A WORKFLOW FOR THE MODELING AND ANALYSIS OF BIOMEDICAL DATA

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the

Graduate School of The Ohio State University

By

Keith Marsolo, B.S., M.S.

* * * * *

The Ohio State University

2007

Dissertation Committee:

Dr. Srinivasan Parthasarathy, Adviser

Dr. Mark A. Bullimore

Dr. Yusu Wang

Approved by

Adviser
Computer and Information Science

ABSTRACT

The use of data mining techniques for the classification of shape and structure can provide critical results when applied biomedical data. On a molecular level, an object's structure influences its function, so classification based on structure can lead to a notion of functional similarity. On a more macro scale, anatomical features can define the pathology of a disease, while changes in those features can illustrate its progression. Thus, structural analysis can play a vital role in clinical diagnosis. When examining the problem of structural or shape classification, one would like to develop a solution that satisfies a specific task, yet is general enough that it can be applied elsewhere.

Before any analysis process can begin, however, the data must be transformed into an appropriate input. There are a wide variety of modeling methods to chose from, but ideally, one would like the transformation process to satisfy the following criteria: 1) capture the clinically- or biologically-relevant features, 2) be computationally feasible, 3) reduce the dimensionality of the data and 4) provide easily interpretable results. It is usually impossible to completely satisfy all four criteria with a single modeling method, so sacrifices must be made. In many fields, there are certain domain-specific methods that are commonly used to model data. While they might provide acceptable performance for the application in question, they may also be specific to a single researcher, making the comparison of results impossible.

In this work, we propose a workflow that can be used to model and analyze biomedical data, both static and time-varying. This workflow consists of four general stages: 1) Modeling, 2) Biomedical Knowledge Discovery, 3) Incorporation of Domain Knowledge and 4) Visual Interpretation and Query-based Retrieval. For each stage we propose either new algorithms or suggest ways to apply existing techniques in a previously-unused manner. Where appropriate, we compare against traditional or accepted standards. Not every technique will be appropriate or effective on all types of data, but we hope that researchers can use our work as a guide in order to determine the most appropriate modeling and analysis path for their particular application.

We present our work as a series of case studies that implement various stages of the above workflow. Through these case studies, we also propose and address a number of specific research questions. We show that generalized modeling methods can be used to effectively represent data from several biomedical domains. We detail a multi-stage classification technique that seeks to improve performance by first partitioning data based on global, high-level details, then classifying each partition using local, fine-grained features. We create an ensemble-learning strategy that boosts performance by aggregating the results of classifiers built from models of varying spatial resolutions. This allows a user to benefit from models that provide a global, coarse-grained representation of the object as well as those that contain more fine-grained details, without suffering from the loss of information or noise effects that might arise from using only a single selection. Finally, we propose a method to model and characterize the defects and deterioration of function that can be indicative of certain diseases.

Dedicated to my parents, who always said I'd spend my life in school.

ACKNOWLEDGMENTS

I would like to thank my advisor, Srinivasan Parthasarathy, as well as Michael Twa and Mark Bullimore for their guidance and assistance throughout my graduate career. Without them, I would not be where I am today.

I would like to thank Chris Johnson for all his domain expertise and assistance in my work with the visual field examinations, particularly his efforts in providing labels for the dataset and his comments on potential improvements and extensions to my algorithms.

I would also like to thank Raghu Machiraju for serving on my candidacy examination committee and Yusu Wang for serving on my final defense committee.

I would like to thank and acknowledge the National Science Foundation and the Australia Academy of Science for giving me with the opportunity to spend a summer conducting research in Melbourne, Australia as part of the East Asia and Pacific Summer Institute (EAPSI). I would like to further thank Ramamohanarao (Rao) Kotagiri for his insights and discussions during my visit and the University of Melbourne for providing me with the needed space and materials.

Portions of this manuscript are based upon work supported by the National Science Foundation under the NSF CAREER Grant IIS-0347662 and the Department of Energy under the DOE Award No. DE-FG02-04ER25611. Any opinions, findings, and conclusions or recommendations expressed here are those of the author and, if applicable, his advisor and collaborators, and do not necessarily reflect the views of the National Science Foundation or the Department of Energy.

Finally, I would like to thank *The* Ohio State University. I gave them 9 years and they gave me four degrees and countless memories. I consider that to be a fair trade.

VITA

March 10, 1980	Born - Columbus, Ohio, USA
2002	B.S. Computer Science and Engineering, The Ohio State University
2005	M.S. Biomedical Engineering, The Ohio State University
2006	M.S. Computer Science, The Ohio State University

PUBLICATIONS

Research Publications

Keith Marsolo and Srinivasan Parthasarathy. On the Use of Structure and Sequence-Based Features for Protein Classification and Retrieval. *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM)*, IEEE, December 2006, pages 394-403.

Keith Marsolo, Srinivasan Parthasarathy and Kotagiri Ramamohanarao. Structure-Based Querying of Proteins Using Wavelets. *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, ACM Press, November 2006, pages 24-33.

Keith Marsolo and Srinivasan Parthasarathy. Protein Classification Using Summaries of Profile-Based Frequency Matrices. *Proceedings of the ACM SIGKDD Workshop on Data Mining and Bioinformatics (BIOKDD)*, ACM Press, August 2006, pages 51-58.

Keith Marsolo*, Michael Twa*, Mark A. Bullimore and Srinivasan Parthasarathy. Spatial Modeling and Classification of Corneal Shape. *IEEE Transactions on Information Technology in Biomedicine* IEEE, Vol. 11, No. 2, March 2007, pages 203-212. (* denotes joint first author)

Keith Marsolo and Srinivasan Parthasarathy. Alternate Representation of Distance Matrices for Characterization of Protein Structure. *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM)*, IEEE, November 2005, pages 298-305.

Keith Marsolo, Srinivasan Parthasarathy, Chris Ding. A Multi-Level Approach to SCOP Fold Recognition. *Proceedings of the IEEE Symposium on BioInformatics and BioEngineering (BIBE)*, IEEE, October 2005, pages 57-64.

Keith Marsolo, Michael Twa, Srinivasan Parthasarathy and Mark Bullimore. Classification of Biomedical Data Through Model-Based Spatial Averaging. *Proceedings of the IEEE Symposium on BioInformatics and BioEngineering (BIBE)*, IEEE, October 2005, pages 49-56.

Keith Marsolo, Srinivasan Parthasarathy, Michael Twa and Mark Bullimore. A Model-Based Approach to Visualizing Classification Decisions for Patient Diagnosis. *Proceedings of the Conference on Artificial Intelligence in Medicine (AIME)*, Springer , July 2005, pages 473-483.

Hongyuan Li, Keith Marsolo, Srinivasan Parthasarathy and Dmitrii Polshakov. A New Approach to Protein Structure Mining and Alignment. *Proceedings of the ACM SIGKDD Workshop on Data Mining and Bioinformatics (BIOKDD)*, ACM Press, August 2004, pages 1-10.

Hui Yang, Keith Marsolo and Sameep Mehta. Discovering Spatial Relations Between Approximately Equivalent Patterns in Contact Maps. *Proceedings of the ACM SIGKDD Workshop on Data Mining and Bioinformatics (BIOKDD)*, ACM Press, August 2004, pages 62-71.

FIELDS OF STUDY

Major Field: Computer and Information Science

Studies in:

Data Mining	Prof. Srinivasan Parthasarathy

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The falling cost and increasing power of modern computing technology has revolutionized the fields of medicine and biology. Doctors and scientists are able to collect and store data as never before. Gene chips, for instance, allow clinicians to simultaneously measure the presence of dozens, or even hundreds, of blood analytes and proteins. Wireless technology gives pacemaker recipients the ability to upload heartrate information directly to their physician, increasing the level of care without requiring the patient to leave the house. Imaging technologies such as ultrasound and fMRI can provide a non-invasive, real-time glimpse into the inner workings of the body. Unfortunately, while both the number of sources and amount of data generated from them has increased by leaps and bounds, the ability of a single person to comprehend this data has not. To that end, the field of *data mining* has arisen to help researchers discover interesting patterns in large datasets.

One particular application of data mining has been its use in the *analysis of structure or shape-based biomedical data.* There are a number of reasons for examining structure-based data. On a molecular level, an object's structure influences its function, so determining the structural similarity between two objects can lead to a notion of functional similarity. On a more macro scale, anatomical features can define the pathology of a disease, changes in those features can illustrate its progression. Thus, one would like to determine and characterize any interesting relationships that exist between the structure of an entity and its function. Most biomedical datasets are so large and complex, however, that it is impossible to find anything but the most trivial of relationships without the aid of advanced computational techniques. By utilizing such procedures, one can implement the semi-supervised or automated discovery of interesting relationships, which greatly speeds up the discovery process and in most cases, improves the quality of results as well.

At the most basic level, the output of a traditional data mining algorithm is a set of features deemed to be "interesting." If the data consists of medical examinations, then these features could differentiate individuals belonging to patient classes such as healthy and diseased. In the protein domain, they might distinguish between different functional groups. While one might be interested in these features in their own right, typically their discovery is just a step in the overall analysis process. Once a set of distinguishing attributes has been obtained, the next stage often involves setting up a system to indicate their presence or absence in an object. This is the principle behind computer-aided diagnosis.

In some applications, the criteria for a diagnosis may be as basic as the detection of a particular antibody in the blood. When this is the case, it is sufficient for a system to provide a simple *yes* or *no* decision. In diseases that are manifested by the existence or absence of certain physical characteristics, however, the decision might not be so cut and dry. A patient could present some of the distinguishing features, but not others, or could present all the features, but to a lesser degree than expected. In these borderline cases, the reasoning behind a decision becomes just as important as the decision itself. Therefore, having a way to interpret the criteria used in classification can provide a great benefit to researchers and clinicians, providing a safety check that could limit some of the liability that might arise from relying purely on a single answer from a computer.

1.1 Challenges in Biomedical Data Analysis

Systems designed to analyze biomedical data can yield enormous benefits, but there are a number of pitfalls that must be addressed if they are to be successfully deployed. The greatest challenges in structure-based biomedical data analysis are usually caused by the data itself.

One of the biggest problems with structure-based biomedical data is noise. Non-invasive imaging techniques like MRI, NMR and ultrasound are inherently noisy. Trying to make a decision based on an image of the inside of the body taken from the outside implies the acquisition of a clean signal is both extremely crucial and incredibly difficult. Even the most advanced instruments and techniques must allow for some degree of uncertainty, so one cannot assume that the data obtained from them will be free of noise.

Another challenge that must be faced is the fact that many datasets contain some missing values. With cross-sectional datasets where one might be comparing isolated examinations, this might not be a serious issue. When one is comparing between individual examinations, this just reduces the size of the dataset. In longitudinal datasets consisting of examinations taken over several years, however, missing values can be much more problematic, as patients miss appointments or drop out of studies.

When this occurs, decisions must be made whether to include these patients in the analysis and if so, how to handle the missing information.

In many structure-based datasets, a single object can be represented by thousands of points. Trying to use traditional analysis techniques on such an object is difficult at best. In general, the raw instrument data is too unwieldy to work with directly, so before any analysis can occur, one must transform it. A researcher can select from one of any number of of modeling methods, but ideally, one would like the transformation process to: 1) capture the clinically- or biologically-relevant features, 2) be computationally feasible 3) reduce the dimensionality of the data and 4) provide easily interpretable results. It is usually impossible to completely fulfill all four requirements with a single modeling method, so sacrifices must be made. However, it should be noted that if the chosen transformation does not highlight those attributes that distinguish between the features of interest, then a different method must be selected, no matter how well it satisfies the other criteria.

Ideally, one would like to use modeling techniques that are as general and as applicable to as many domains as possible. A wide variety of biomedical instruments exist to measure shape and structure and they use a number of different techniques to do so. This means that a single entity can be represented by several different modalities. Having generalized modeling techniques that can be applied to these different modalities would keep researchers from constantly having to reinvent the wheel. Unfortunately, many transformations are chosen specifically for their ability to accentuate certain aspects of the data. Adding to the challenge, even if one allows for different data types, what is considered to be interesting in one domain is often not in another. The adage “one man’s trash is another man’s treasure,” or the more scientific “one man’s signal is another man’s noise[1]” perfectly capture this dilemma.

Along these same lines, there is a growing problem within the scientific community involving the inability or failure of individuals to communicate across disparate domains. While a researcher in vision science and a scientist in protein structural informatics may both be interested in biomedical data modeling and classification, they may not believe that a modeling technique that can be used to classify the shape of the cornea may also be effective in capturing the structure of a protein molecule. Often, if a relationship between the data is not immediately apparent, there is a reluctance to expend any effort to test whether such a “foreign” technique would be applicable. This reluctance causes researchers to focus only on those techniques that have shown to be successful in the past. While a technique may provide an adequate solution to a problem, it might not be the best solution, leading, in essence, to a local optima. It is believed that these challenges, while formidable, can be overcome, which leads to the premise of this document.

Thesis Statement: *The thesis of this work is that it is possible to discover meaningful information in structure-based biomedical data using generalized modeling methods and analysis techniques[2]. Furthermore, we believe that this information can discovered on data obtained from a number of different source domains. Finally, we believe that the results from any analysis can be presented to the user in a visual and easily-interpretable manner.*

[1]Here the traditional term *man* is used, but it actually refers to the gender-neutral *person* or *human*. Its use is simply a matter of preference and no disrespect is intended.

[2]Throughout this document, *we* is used to refer to the author instead of *I*. It is simply a manner of style reflecting the writing preference of the engineering and computer science community. Unless specifically noted, when using *we*, the author is referring to his own work or work done jointly with his advisor.

1.2 A Workflow for the Analysis of Structure-Based Biomedical Data

In this dissertation, we propose a workflow that can be used to model and analyze biomedical data, both static and time-varying. This workflow, shown in Figure 1.1 consists of three main stages and one optional stage. We list each stage along with a short description in the text below.

1. *Data Modeling:* The first stage of the workflow involves modeling the raw source data. The number of ways to represent an object is almost limitless, but since a large amount of structure-based biomedical data is represented as an image or in an image-like form, we evaluate those techniques that have been used previously for image analysis. This is simply an attempt to keep the problem tractable and in some sense, domain-independent. The methods we investigate include polynomial families such as Zernike, pseudo-Zernike and Legendre as well as discrete transformation algorithms such as wavelets [2–5]. Before proceeding, we should note that while we may focus on image analysis techniques to model our data, the original input is usually *not* an image. It can be viewed as an image, but the data was derived by some other means. For example, one can create a topographical picture of Earth using satellite photos or through ground-based elevation readings. While our techniques will work on the former, our data typically takes the form of the latter.

   Within the image analysis community, a transformation is typically judged on its ability to reconstruct an image, often in the presence of additive noise [5–7]. Our standards are slightly different, however. Given two models, if one has a lower fidelity than the other but results in a a cleaner separation among objects of different categories, we are more likely to choose that model as it will likely to lead to higher performance in the analysis stage.

The input to the modeling stage will be a set of data that has been collected and cleaned (i.e. formatting removed, no missing values, etc.). This input can consist of any combination of spatial and temporal information. Therefore, we divide the overall Modeling stage into two sub-stages: spatial and temporal modeling. It is possible to use the same techniques to model both types of data, so this division is more of a semantic distinction. The desired modeling method(s) will be selected and the data will be subsequently transformed into a corresponding spatial-temporal feature vector.

2. *Biomedical Knowledge Discovery:* The purpose of this stage is to discover interesting patterns among the previously-generated spatial-temporal features vectors. It can include cross-sectional data analysis if the input consists only of spatial information, but will employ longitudinal data analysis if there is a temporal component. It will include many of the standard data mining and machine learning techniques such as classification and clustering. Many of these techniques are similar, but there are differences among them. For instance, classification is the process of predicting or assigning a label to an object based on its similarity to a group of previously-labeled objects. Clustering, on the other hand, tries to group objects so that similar objects are close together and dissimilar objects are far apart. A label is then assigned to each group. The difference between the two methods is that clustering can handle unlabeled data, which is useful when working with new information that has yet to be categorized.

There are also ensemble-learning techniques that can be used to improve the quality of results. These techniques rely on the principle that one can create several different individual learning models (e.g. classifiers) and aggregate their output to produce a more accurate final decision. We provide a brief overview of the more commonly used classification, clustering and ensemble-learning techniques in Section 2.2.

Throughout this chapter, we have made mention of the fact that we will work to discover interesting patterns among the data. This leads to the question of "what is an interesting pattern and what makes a pattern interesting?" In the broadest sense, an interesting pattern consists of the features or attributes of a model or representation that distinguish between the different categories in the data. These categories, such as patient class or functional group, will often be domain-specific. When possible, we explain how the features of our model relate to specific structure-based properties of the data.

In summary, using the spatial-temporal feature vector from the previous stage, we apply techniques to discover interesting patterns among the data. Depending on the data and the problem in question, they can include classification, clustering, or various ensemble-learning strategies.

3. *Incorporation of Domain Knowledge:* In many cases, it is possible to improve the quality of the mining results by including domain-specific information. This can be accomplished by using the additional data as a filter, or even by incorporating this information directly into the pattern discovery process. We also view this stage as the location where we would attempt to establish an anatomical

correspondence between the features of our model and the physical attributes of the source data. Depending on the modeling method and application domain, it may not be possible to make such a link, so this stage is considered to be optional.

4. *Visual Interpretation and Query-based Retrieval:* Once a set of interesting patterns have been obtained, they need to be presented so the user can get a sense of the results. This presentation can be provided in numerical form, using traditional approaches from mathematics or statistics, or it can be in a more visual form, using heat maps or decision surfaces to illustrate areas of interest. Furthermore, we intend for this stage to contain additional, higher-level discovery methods such as database queries.

As indicated in the diagram it is possible to return to a previously-visited stage in order to change the modeling or discovery techniques in an attempt to improve the results. The final output of this workflow would be information that can be used to assist in a decision about the data, such as a disease diagnosis or a functional classification. It is not intended to serve as the sole justification for a decision, merely to support it.

For each stage we propose new algorithms or suggest ways to apply existing techniques in a previously-unused manner. Where appropriate, we compare against traditional or accepted standards. While not every technique will be appropriate or effective on all types of data, we hope to provide researchers with results so they can make an informed decision about the modeling and analysis path that may work best for them. Additionally, by creating a workflow that allows for multiple data types and modeling methods, we hope to allow researchers to quickly test whether a previously

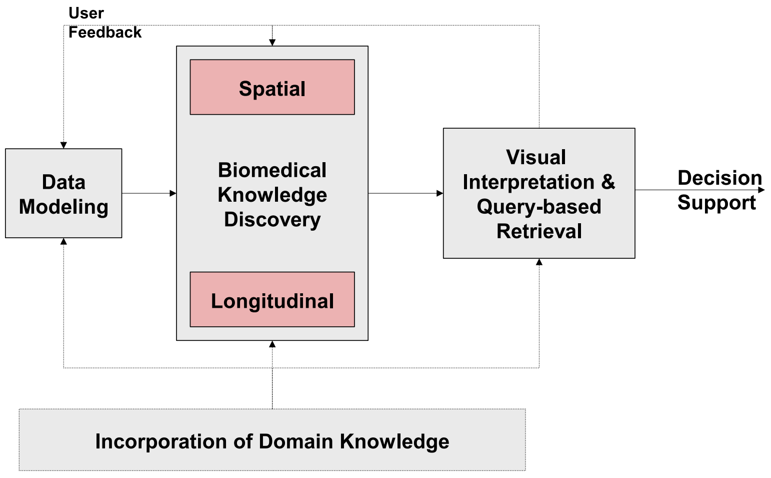

User Feedback
Spatial
Data Modeling
Biomedical Knowledge Discovery
Visual Interpretation & Query-based Retrieval
Decision Support
Longitudinal
Incorporation of Domain Knowledge

Figure 1.1: Diagram of proposed workflow.

unconsidered approach can yield important insights into their respective problems. It is our hope that this flexibility will foster further collaboration or the exchange of ideas and research methods across domains.

Rather than rely on a single dataset, we choose to evaluate our methods on several different data sources. It is extremely difficult to obtain a dataset that is challenging enough to require both complex spatial modeling and robust longitudinal data analysis. Most existing real-world datasets tend to lean one way or the other. Therefore, it is unlikely that a single dataset will require the use of every method discussed here. That is why we propose a generalized workflow and not a specific end-to-end application.

We validate our approach through several case studies and subsequent extensions that illustrate how a researcher would proceed through the workflow. These case studies rely on datasets that were taken from two different biological domains. The first is the protein domain. We study several datasets that were constructed from publicly-available information, primarily in the form of three-dimensional structural coordinates and attributes related to the primary structure or amino acid sequence of each protein. These datasets are derived from from online databases such as the Protein Data Bank (PDB) and Structural Classification of Proteins database (SCOP) [8, 9]. The second domain used in our validation is that of vision science. From this domain we obtain two datasets relating to the human eye. The first consists of information relating to the topography of the corneal surface, with each object labeled as normal, post-operative, or having keratoconus (diseased). The second contains visual field assessments that represent the results of an early-stage glaucoma study. Unlike our other datasets, which are purely spatial in nature, the visual field dataset includes patient examinations taken over a number of years, providing the opportunity to perform longitudinal data analysis. The dataset is known to include patients who are normal (healthy), those who are suffering from glaucoma and individuals who are believed to be in the early stages of the disease.

1.3 Contributions

While we present our work as a series of case studies and extensions, we also address a number of specific research questions. We categorize these contributions by workflow stage. In instances where the research spans several stages, we list them based on the stage where they show the greatest novelty. They are as follows:

1. *Modeling:*

    - We provide a method to generate a compact representation of 3D protein structure. We transform the 3D structural coordinates into a 2D distance matrix and apply a 2D wavelet decomposition, treating the resulting coefficients as a feature vector. Our model yields improved performance over existing representations in terms of both classification and database retrieval accuracy.

- We detail an approach that can be used to create a stand-alone model of protein sequence by applying a wavelet decomposition to the normalized results of the PSI-BLAST alignment algorithm. This model provides comparable classification accuracy to existing solutions that rely on computationally-expensive kernel methods, while also providing a representation that can be used in query-based applications such as database retrieval.
- Previous experiments have shown the effectiveness of using Zernike polynomials to model the surface of the cornea [10–12]. We also examine the use of wavelets, pseudo-Zernike and Legendre polynomials, though they fail to provide improvement in classification accuracy over Zernike-based models.

2. *Knowledge Discovery:*

   - We show that one can use spatial modeling methods and decision tree classifiers to adequately distinguish between the corneas of three patient classes.
   - We create an ensemble-learning strategy that improves classification performance by aggregating the results of individual classifiers built from models of varying spatial resolution or polynomial complexity. This approach

allows a user to benefit from models that provide a global, coarse-grained representation of the object as well as those that contain more fine-grained details, without suffering from the loss of information or noise effects that might arise from using only a single selection.

- We detail a multi-stage classification technique that seeks to improve performance first by partitioning data based on global, high-level details, then classifying each partition using local, fine-grained features. We show the utility of this approach by applying it to a well-studied protein dataset that contains a large number of classes, many of which contain only a few members.
- We propose a tunable method of modeling the visual field that can be used to highlight and detect defects and abnormalities. This model can also be used to characterize the change in a given field over time. Our techniques can be used in applications like classification or the detection of outliers and anomalies.

3. *Incorporation of Domain Knowledge:*

- We show that given a structure-based representation of a protein molecule, one can include physio-chemical properties to improve the accuracy of decision tree classifiers.
- Using a set of defintions that describe a number of common visual field defects [13], we create a series of templates that allow us to characterize how well the visual field of a patient matches a particular category of abnormality.

4. *Visual Interpretation and Query-based Retrieval:*

   - Given a spatial model of the surface of the cornea, we provide a method to visualize the results of a decision tree classifier, highlighting the anatomical features of the surface that are important for classification.

   - We show that our wavelet representations of a protein molecule can be used to allow for fast and efficient structural queries, on both a global and sub-structural level. They can also be used to retrieve objects using sequence-based information. We also show that, compared with existing approaches, this representation results in superior performance in terms of retrieval accuracy, memory usage and query execution time.

   - We detail techniques to visualize abnormalities that are present in a patient's visual field examination.

We also make a number of contributions that are specific to each of the application domains in our study. We feel these efforts will help to spur advances in each field.

- In the *protein domain*, we show that it is possible to use wavelets to derive a model of the protein based on sequence or structure. We also propose an additional technique that can be used to retrieve proteins based on shared local substructures. These representations result in compact, independent feature vectors that could be leveraged to mine the results of large-scale folding simulations [14].

- In representing *corneal topography*, we validate previous work that shows one can use Zernike polynomials to effectively model the surface of the cornea, and that those models can be used to differentiate normal patients from diseased [15]. We develop techniques to highlight the surface features deemed important in this differentiation and we provide a technique that uses classifiers built from multiple surface models to boost classification accuracy above 90%. This is often cited as the minimum level of accuracy at which clinicians are comfortable with working with an automated decision support system.

- Finally, in our work with the *visual field*, we develop a method that allows for the automatic, yet user-tunable, categorization of the visual field examination history of a patient. Our system can assign a label to a patient based on the overall abnormality of their visual field, the progression or change between individual exams, or using both criteria. In addition, our algorithm has the added capability of determining whether an entire examination history is abnormal or anomalous. Through our experimental evaluation, we find that our approach yields excellent results in terms of accuracy, reporting greater than 90% agreement with a domain expert's classification.

1.4 Organization

In the next chapter we provide a brief overview for each of the knowledge discovery algorithms used in the creation of our techniques. We then present several case studies that highlight the methods we developed for the various stages of our proposed workflow. The first covers our experiments on the modeling and classification of corneal shape and an approach to visualizing those classification decisions. Chapter 4 contains our work in protein classification and our initial efforts in the wavelet-based modeling of protein structure. We also detail a technique that uses a multi-stage, hierarchical approach to improve classification performance. Our final case study involves our attempts to model visual field abnormalities and characterize the change in a visual field over time. In Chapter 6, we discuss extensions of our wavelet-based methods of modeling both protein sequence and structure and illustrate how those models can be used for effective query-based protein retrieval. We follow with an ensemble-learning technique that can improve classification accuracy by aggregating the results of classification models built from multiple spatial resolutions of the source data. We conclude with a discussion of our results and detail our plans for future work in Chapter 8.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter contains information on the mathematical methods used to model our biomedical datasets as well as details on the classification algorithms used in this work. Unless otherwise noted, all mathematical formulae are given using the original notation from the source material.

2.1 Modeling Methods

In this section we cover the basic mathematical properties of the *Zernike*, *pseudo-Zernike* and *Legendre* polynomials as well as the *discrete wavelet transform.*

2.1.1 Zernike Polynomials

Zernike polynomials are an infinite series of circular polynomial functions that are orthogonal over a normalized unit circle and consist of three elements [16]. They have been applied to numerous problems including biometric security, rainforest management, and automated character recognition [17–19]. The first element is a normalization coefficient. The second element is a radial polynomial component and the third, a sinusoidal angular component. The general form for Zernike polynomials is given by:

$$Z_n^m(\rho,\theta)=\begin{cases}\sqrt{2(n+1)}R_n^m(\rho)\cos(m\theta) & \text{for } m>0\\ \sqrt{2(n+1)}R_n^m(\rho)\sin(|m|\theta) & \text{for } m<0\\ \sqrt{(n+1)}R_n^m(\rho) & \text{for } m=0\end{cases} \tag{2.1}$$

where n is the radial polynomial order and m represents azimuthal frequency. The normalization coefficient is given by the square root term preceding the radial and azimuthal components. The radial component of the Zernike polynomial (R), the second portion of the general formula, is defined as:

$$R_n^m(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s(n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!}\rho^{n-2s} \tag{2.2}$$

Note that the value of n is a positive integer or zero. For a given n, m can only take the values $-n$, $-n+2$, $-n+4$, $\ldots$, n. In other words, $m-|n| = \mathit{even}$ and $|n| \leq m$. Thus, only certain combinations of n and m will yield valid Zernike polynomials. Any combination that is not valid simply results in a radial polynomial component of zero. The radial component also satisfies the orthogonality relation:

$$\int_0^1 R_n^m(\rho)R_{n'}^m(\rho)\rho d\rho = \frac{1}{2(n+1)}\delta nn' \tag{2.3}$$

where $\delta_{nn'}$ is the Kronecker delta.

Rather than refer to each term with the two indexing scheme using n and m, each term is labeled with a number j that is consistent with the single indexing scheme created by the Optical Society of America [16]. The relation between j, n and m is as follows:

$$\begin{aligned} j &= \tfrac{1}{2}(n(n+2)+m)\\ n &= \lceil \tfrac{1}{2}(-3+\sqrt{9+8j}) \rceil\\ m &= 2j-n(n+2)\end{aligned} \tag{2.4}$$

where $\lceil \cdot \rceil$ is the ceiling operator.

Polynomial	Azimuthal Frequency (m)								
Order (n)	-4	-3	-2	-1	0	1	2	3	4
0					0				
1				1		2			
2			3		4		5		
3		6		7		8		9	
4	10		11		12		13		14

Table 2.1: Combinations of radial polynomial order (n) and azimuthal frequency (m) that yield the 15 terms comprising a 4th order Zernike polynomial.

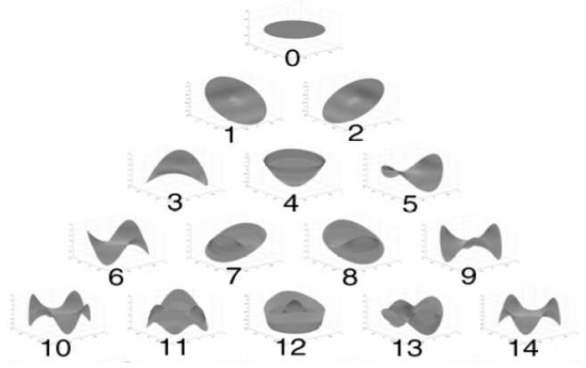


Figure 2.1: Graphical representation of the modes that constitute a 4th order Zernike polynomial. Each mode is labeled using the single indexing convention.

Polynomials that result from fitting our raw data with these functions are a collection of orthogonal circular geometric modes. Examples of the first 15 modes (representing a 4th order Zernike polynomial) can be seen in Figure 2.1. The label underneath each mode corresponds to the jth term listed in Table 2.1. The coefficients of each mode are proportional to its contribution to the overall topography of the original image data. As a result, we can effectively reduce the dimensionality of the data to a subset of polynomial coefficients that represent spatial features from the original data as the magnitude of discrete orthogonal geometric modes.

2.1.2 Pseudo-Zernike Polynomials

Pseudo-Zernike polynomials are another family of orthogonal polynomial functions [5]. Like Zernike polynomials, these functions are an infinite series of circular modes orthogonal a normalized unit circle. The general form for pseudo-Zernike polynomials is given by:

$$P_n^m(\rho,\theta) = \begin{cases} \sqrt{2(n+1)}S_n^m(\rho)\cos(m\theta) & \text{for even } n, m \neq 0 \\ \sqrt{2(n+1)}S_n^m(\rho)\sin(m\theta) & \text{for odd } n, m \neq 0 \\ \sqrt{(n+1)}S_n^m(\rho) & \text{for } m = 0 \end{cases} \tag{2.5}$$

Similarly, the radial portion of the pseudo-Zernike polynomial (S) is defined as:

$$S_n^m(\rho) = \sum_{s=0}^{n-|m|} \frac{(-1)^s(2n+1-s)!}{s!(n-m-s)!(n+m+1-s)!}\rho^{n-s} \tag{2.6}$$

and is related to the radial component of the Zernike series in the following manner:

$$\rho S_n^m(\rho^2) = R_{2n+1}^{2m+1}(\rho) \tag{2.7}$$

One difference between pseudo-Zernike and Zernike polynomials is that with pseudo-Zernike polynomials, the only constraint on m is that $|m| \leq n$. This results in the generation of $(n+1)^2$ linearly independent polynomials for a given degree

Transform	4	5	6	7	8	9	10
Zernike	15	21	28	36	45	55	66
Pseudo-Zernike	25	36	49	64	81	100	121

Table 2.2: Total number of moments for each transformation as a function of the polynomial order.

n, as opposed to the $\frac{1}{2}(n+1)(n+2)$ polynomials that are generated for a Zernike polynomial of the same degree. It has been shown that for the same polynomial order, pseudo-Zernike polynomials are more robust with noisy data than Zernike polynomials [20]. One disadvantage of using pseudo-Zernike polynomials, however, is their greater computational cost. This drawback is partially offset by the fact that the pseudo-Zernike transformation algorithm can be parallelized.

When discussing the above data transformations, we often refer to polynomials of a certain order (n). Each order is comprised of a varying number of moments. We provide the number of moments for each order and transformation that we tested in Table 2.2. Since the number of moments for a given order is not the same for all of the transformations, that is an issue that must be considered when evaluating the utility of these methods for analysis purposes.

2.1.3 Legendre Polynomials

Like the Zernike and pseudo-Zernike transformations, the moments produced by the Legendre transformation are orthogonal. Derived from the Legendre polynomials, and defined using Cartesian coordinates, the Legendre moment of order $m+n$ of a two-dimensional function $f(x, y)$ can be written as:

$$\lambda_{mn} = \frac{(2m+1)(2n+1)}{4} \int_{-1}^{1} \int_{-1}^{1} L_m(x) L_n(y) f(x, y) dx dy \tag{2.8}$$

where $L_m(x)$ represents the Legendre polynomial of degree m. This polynomial is given by the formula:

$$L_m = \frac{1}{2^n n!} \frac{d^n}{dx^n} (x^2 - 1)^n \tag{2.9}$$

The Legendre polynomials form an orthogonal basis set over the interval [-1,1]:

$$\int_{-1}^{1} L_m(x) L_n(x) = \frac{2\delta mn}{2(m+1)} \tag{2.10}$$

where δ_{mn} is the Kronecker delta [5]. Over a square of size N, the discrete approximation of the Legendre moment can be written as [7]:

$$\lambda_{mn} = \frac{(2m+1)(2n+1)}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} L_m(x_i) L_n(y_j) f(i, j) \tag{2.11}$$

with the coordinate transformation computed by the following:

$$x_i = \frac{2i - N + 1}{N - 1}; \; y_i = \frac{2j - N + 1}{N - 1} \tag{2.12}$$

2.1.4 Wavelets

Wavelets are mathematical functions that, like a Fourier transformation, can be used to approximate a signal based on a superposition of functions. As with Fourier transformations, wavelets approximate the data using functions at varying frequencies. Unlike Fourier transformations, however, the frequency components are examined at different resolutions. This allows for the detection of both low-level, global trends, and high-level, local features [21]. Another difference between wavelets and Fourier transforms is in the underlying functions. A wavelet function is a waveform of limited duration, with an average value of zero (often called the mother wavelet). Fourier transforms, on the other hand, use sine or cosine waves, which also have an average value of zero, but have an unlimited duration.

There are two types of wavelet transformations: continuous and discrete. For our experiments, we focus on the discrete wavelet transform and thus do not discuss the continuous version.

Given a decomposition level L and a one-dimensional signal of length N, where N is divisible by 2^L, the discrete wavelet transform consists of L stages. At each stage in the decomposition, two sets of coefficients are produced, the *approximation* and the *detail*. The approximation coefficients are generated by convolving the input signal with a *low-pass* filter and down-sampling the results by a factor of two. The detail coefficients are generated in a similar manner by convolving the input with a *high-pass* filter and down-sampling by a factor of two. If the final stage has not been reached, the approximation coefficients are treated as the new input signal and the process is repeated. This process is illustrated in Figure 2.2 (a). In many cases, once the wavelet transformation is complete, the original signal will be represented using combinations of approximation and detail coefficients.

To operate on a two-dimensional signal (of size N x N), the decomposition proceeds as follows: First, the rows are convolved with a low-pass filter and downsampled by a factor of two, resulting in matrix L ($N/2$ x N). The process is repeated on the original signal using a high-pass filter, which leads to matrix H ($N/2$ x N). The columns of L are convolved two separate times, once with a low-pass filter and again with a high-pass filter. After passing through the filters, the signals are downsampled by a factor of two. This results in a matrix of approximation (LL) and horizontal

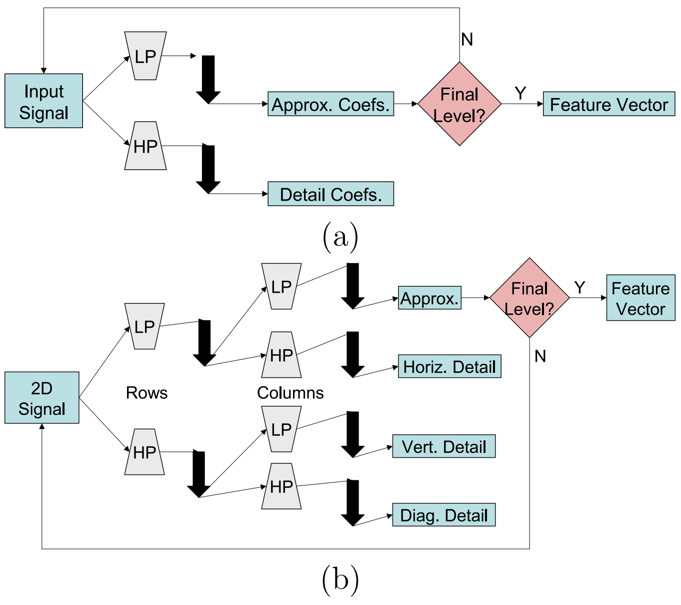


Figure 2.2: Illustration of wavelet decompositions. Top (a): 1D decomposition. Bottom (b): 2D decomposition. *LP* and *HP* refer to the low-pass and high-pass filters, respectively. The large black arrows denote the downsampling process.

detail coefficients (LH), respectively (both of size $N/2$ x $N/2$). These steps are executed once more, this time on the columns of H, resulting in a matrix of diagonal (HH) and vertical (HL) detail coefficients (again of size $N/2$ x $N/2$). The whole procedure can then be repeated on the approximation coefficients contained in matrix LL. Figure 2.2 (b) provides an overview of this process.

Wavelets have been used in a number of applications, ranging from signal processing, to image compression, to numerical analysis [21]. They play a large role in the processing of biomedical instrument data obtained through techniques such as ultrasound, magnetic resonance imaging (MRI) and digital mammography [22]. Wavelets have also been used within the protein domain for various applications, though the reported results have been less than impressive [23].

2.2 Knowledge Discovery - Classification

In the text below we provide a very brief overview of some of the more popular classification techniques used within the data mining community. Rather than go into specific details about each algorithm, we simply cover some of their basic, high-level principles. For a more in-depth discussion, we refer the reader to one of the several books on data mining [24, 25].

Classification is the process of predicting or assigning a label to an object based on that object's similarity to a group of previously labeled objects. An object is assigned the label of the group or class to which it is most similar. The general process involves taking a set of *training data* and constructing a classification model that can discern between objects of different classes based on certain criteria. Then, given a new object or set of objects (often referred to as *target* or *testing data*), using the previously constructed classification model, the classifier assigns a label.

2.2.1 Bayesian Classifiers

Bayesian classifiers are based on Bayes' theorem, which can be formally stated as:

$$P(h|x_i) = \frac{P(x_i|h)P(h)}{P(x_i)} \tag{2.13}$$

where $P(h|x_i)$ represents the posterior probability, $P(h)$ the prior probability associated with hypothesis h and $P(x_i)$ is the probability of occurrence for data value x_i. Finally, $P(x_i|h)$ is the conditional probability that given a hypothesis h, x_i satisfies it.

The simplest Bayesian classification method is the Naïve Bayes classifier [26]. A Naïve Bayes classifier reaches a decision for an object by calculating the probability of that object belonging to each of the possible classes and choosing the label with the highest probability. This is done by calculating the contribution of each feature (or attribute) to a class as a conditional probability. These values are combined to arrive at a prediction. Mathematically, this is written as:

$$P(x_i)|C_j) = \prod_{k=1}^{n} P(f_{ik}|C_j) \tag{2.14}$$

where x_i is a tuple in the dataset, C_j is a class, and f_{ik} represents the value of x_i for attribute f_k.

While a Naï've Bayes classifier is simple and easy to construct, it is based on the inherent assumption that the features are conditionally independent, hence the "naïve" label. This assumption is not always valid, however. Even so, they have a long history in classification problems and have been shown to provide very good results even when the features are not independent [27]. In addition, the classification model is highly interpretable, in that a user can manually exam the probability values assigned to each attribute.

2.2.2 Decision Trees

Decision tree classifiers include some of the most popular classifiers in use today. The principle behind a decision tree is to select an attribute at each node and divide the feature space into increasingly fine-grained regions. The class label of an object is based on the final region in which it lies. One benefit of decision trees is that they are transparent, meaning the user can inspect the actual tree produced by the classifier to examine the features used in classification. This greatly increases the utility of the classifier and is helpful when used as a basis for patient diagnosis.

While there are many different decision tree algorithms, here we focus on the ID3 family of decision trees. ID3-based decision trees [28] rely on the principle of entropy to partition the search space. Entropy (H) represents the amount of disorder in a system. It is formally defined as follows:

$$H(p_1, ..., p_n) = \sum_{i=1}^{n} (p_i \log(1/p_i)) \tag{2.15}$$

In the case of classification, entropy is used to represent the ratio of disagreement between the class labels of the objects in a given partition. If all of the objects in a partition have the same label, the entropy is zero. Thus, at each node in the decision tree, an ID3-based algorithm selects the feature that best minimizes the entropy of the partition. The original ID3 algorithms based their splits on the following equation:

$$Gain(D, S) = H(D) - \sum_{i=1}^{s} P(D_i) H(D_i) \tag{2.16}$$

where D represents the current state and D_i represents the state after choosing one of s possible splits. A drawback to using the above equation as a splitting criteria is that *Gain* would be maximized simply by putting each object into their own partition. This lead to the development of *GainRatio*, which takes into account the size or cardinality of each partition. The GainRatio formula is defined as:

$$GainRatio(D, S) = \frac{Gain(D, S)}{H\left(\frac{|D_i|}{|D|}, ..., \frac{|D_s|}{|D|}\right)} \tag{2.17}$$

When testing ID3-based decision trees, we use the C4.5 algorithm [24, 28, 29], which uses GainRatio. This algorithm is freely available to the public and available as part of several open-source data mining packages.

2.2.3 Neural Networks

The neural network is another method of classification that has seen wide use in patient diagnosis. They are modeled on the way the brain processes information, using several interconnected processing nodes working in parallel to make a decision [30]. Each network consists of a specified number of input nodes that are connected to one of possibly several hidden layers consisting of a varying number of processing nodes. In most of the works that we will examine, the number of hidden layers is typically set to one. The hidden layer is connected to a set of output nodes, one per class label. Given a set of training data, the neural network will then attempt to learn a function that minimizes the predictive error. Neural networks provide a number of benefits: they are highly tunable for a specific application, have been shown to be more robust to noise than decision trees, and when properly configured, can provide very good results. However, they also have several drawbacks, including time and complexity of construction, a lack of overall interpretability, and the potential for overfitting [31]. Despite these drawbacks, their use remains popular within the biomedical community [32, 33].

2.2.4 Support Vector Machines

Support Vector Machines comprise another popular class of statistical learning algorithms. Given a set of binary-labeled examples, the support vector machine attempts to construct a maximum-margin hyperplane between the classes. The resulting hyperplane then functions as a linear classifier. The original support vector machine algorithms were extended to allow for non-linear classification by allowing an n-dimensional feature vector to be mapped to a higher-dimension feature space using a kernel function. Within this feature space, a linear classifier is constructed, even though it may be non-linear in the original input space [34].

2.2.5 Other Methods

There are a number of classification methods that do not fall into any of the above categories. These include many statistical and distance-based methods. One such method is the K-Nearest Neighbor classifier [35]. A nearest-neighbor classifier uses the pairwise distance (or similarity) between objects to assign a classification label. Given a piece of target data, the similarity between each object is computed. Then, the classifier looks at the labels of the K nearest neighbors and assigns to the target the majority label.

Still yet another alternative is a classifier called Voting Feature Intervals [36]. The Voting Feature Intervals algorithm is similar in spirit to the Naïve Bayes classifier in that each feature is considered independently, but rather than calculate a conditional probability for each attribute, the dimensional space of each feature is divided into a series of intervals with a number of votes being assigned to each. Each feature votes for a particular class, and the class with the largest number of votes becomes the predicted class for the object.

2.2.6 Ensemble-learning Techniques

Several techniques have been applied to existing classification methods in an attempt to improve accuracy. These techniques are often called or $ensemble-learning$ techniques because they combine the output of several other methods to achieve their results. The principle behind ensemble-learning is that one can combine a number of classifiers with a larger error to create an overall classifier with a smaller error. The intuition is that a group of experts usually provide more accurate decisions than any single expert.

As an example, suppose we have a ensemble-classifier that takes the decisions of n individual classifiers and makes a final decision based on the label chosen by a majority of those n classifiers. If each individual classifier has an error rate of ϵ, the overall error of the ensemble-classifier can be expressed using the following equation:

$$\sum_{i=\frac{n}{2}+1}^{n} \binom{n}{i} \epsilon^i (1-\epsilon)^{n-i} \tag{2.18}$$

As long as the errors committed by the individual classifiers are uncorrelated and as long as the error rate of the base classifier is less than the probability of a correct random guess (in a binary decision problem, the probability of a correct random guess is 1/2, while in a three-class problem, that probability is 1/3), ensemble-classification will provide an increase in classification accuracy. Ensemble-learning strategies have been shown to yield the greatest benefit when coupled with *weak* classifiers, i.e. classification strategies with error rates only slightly less than the probability of a correct random guess. The weaker the base classifier, the greater the benefit.

Two ensemble-learning techniques that we employ are boosting [37] and bagging [38]. As is the case with many ensemble-learning strategies, boosting and bagging reach a decision through the aggregation of multiple hypotheses. Both methods take a user-specified classification algorithm and construct several different classification models (that number is also user-specified). The results of these models are combined to reach a final decision. When constructing the classifiers, a different sample of the data is selected (with replacement) at each iteration. The difference between the two is that bagging draws an independent sample from the data at every iteration, whereas boosting draws a weighted sample, with misclassified objects carrying a higher weight during the training process. Since a misclassification represents an object that was not properly captured by the underlying model, we want to put more emphasis on those objects that are "problematic." In addition to drawing a sample that is biased toward misclassified objects, when the final classifier combines the results of the underlying classifiers, boosting will put more weight on the classifiers built from the samples containing the misclassified objects, as they are believed to be more accurate. Bagging, on the other hand, weighs the contribution of all underlying classifiers equally. Since both techniques draw random samples from the data at each iteration, they are most effective when paired with "unstable" classifiers, or classifiers that are sensitive to minor perturbations in the training data. While boosting and bagging have been shown to significantly improve the results of weak classifiers, this improvement comes with the price of reduced interpretability and an increase in time needed to construct the classification model.

Another ensemble-learning strategy that we examine is Random Forests [39]. Unlike boosting and bagging, which can be combined with any weak classification strategy, the Random Forests algorithm relies solely on decision trees. The Random Forests algorithm works by constructing a large number of trees, each of which votes on a class label for an object. A majority vote is then taken to reach an overall decision. Each tree is built from a random sample of the data, much like what occurs when one creates a bagged decision tree. Unlike a bagged decision tree, however, with Random Forests, the splitting attribute for each node is selected from a small random sample of the feature space.

Finally, we consider the RuleFit algorithm [40], which can be viewed as a slightly less-structured approach to ensemble-learning. Rather than base a decision on the votes of several classifiers, the RuleFit algorithm employs a scoring function derived from an ensemble of *rules*. Using a decision trees as an example, a rule is equivalent to the path taken from the root node to any other node (leaf or internal). Traditionally, to create a rule ensemble, one generates a number of decision trees and converts them into rule form. Once this ensemble has been created, the RuleFit algorithm uses a regularized minimization procedure to create a linear weighting. Given this weighting, one can create a scoring function based on the linear combination of the rules, which serves as a classifier.

In many clinical applications, the ability to examine the criteria behind a classification decision can be as important as the decision itself. In such cases, the increased performance gained through ensemble-learning techniques may not be worth the corresponding loss of interpretability. In addition, if a clinician needs to repeatedly construct a classifier, or if they are dealing with a large or growing dataset, then the somewhat poor scalability of many ensemble-learning methods may also be a cause for concern. If the improvement in classification accuracy is not sufficiently large, then the use of ensemble-learning methods may not justify the extra time needed to construct the additional classifiers.

CHAPTER 3

CASE STUDY I: MODELING AND CLASSIFICATION OF CORNEAL SHAPE

This chapter details our attempts to objectively quantify topographical features obtained from biomedical instrument data to support diagnostic classification and clinical decision support for eye disease. Specifically, given a three-class dataset where each record represents the surface features of the cornea, we examine whether it is possible to reconstruct the surface, capturing the relevant anatomical features needed for successful classification, while at the same time reducing the dimensionality of the data.

Our first set of experiments examine the performance of models generated from two different orthogonal polynomial families, the Zernike and pseudo-Zernike polynomials. We compare these models in terms of fidelity to the original source data and evaluate their performance under several different classification and meta-learning strategies [41]. We then present a method that leverages the features of the Zernike-based models to highlight the anatomical features that play a role in classifying and partitioning the different patient groups [42]. Finally, we examine the feasibility of using alternative modeling methods such as wavelets and Legendre polynomials to create surface representations and their subsequent performance in the classification process. We begin by providing some biological background and a discussion of related work.

3.1 Biological Background

Image formation in the human eye begins when entering light is focused by the cornea, passes through the pupil, and is further focused by the lens, forming an image on the retina at the back of the eye. Since light must pass through the cornea, clear vision depends very much on the optical quality of the corneal surface, which is responsible for nearly 75% of the total optical power of the eye [43]. The normal cornea has an aspheric profile that is more steeply curved in the center relative to the periphery. Subtle distortions in the shape of the cornea can have a dramatic effect on vision. For this reason, the shape of the corneal surface is measured clinically for the treatment and diagnosis of eye disease. The process of mapping the surface features on the cornea is known as *corneal topography* [44].

3.1.1 Clinical Use of Corneal Topography

The use of corneal topography has rapidly increased in recent years because of the popularity of refractive surgery and the decreased cost of powerful personal computers. Refractive surgery is an elective surgical treatment intended to reduce one's dependence on glasses or contact lenses. The most common surgical treatment performed for this purpose is *laser assisted in-situ keratomileusis* (LASIK). Corneal topography is used to screen patients for corneal disease prior to this surgery and to monitor the effects of treatment after surgery.

Another clinical application of corneal topography is the diagnosis and management of keratoconus. Keratoconus, a progressive, non-inflammatory corneal disease, distorts corneal shape and results in poor vision that cannot be corrected with ordinary glasses or contact lenses. Patients with keratoconus frequently seek refractive surgery due to their poor vision. However, such treatment exacerbates corneal distortion and frequently leads to corneal transplant. Thus, corneal topography is a valuable tool for diagnosis and management of keratoconus as well as for the prevention of inappropriate refractive surgery in this patient group.

3.1.2 Determination of Corneal Shape

The most common method of determining corneal shape is to record an image of a series of concentric rings reflected from the corneal surface. Any distortion in corneal shape will cause a distortion of the concentric rings. By comparing the size and shape of the imaged rings with their known dimensions, it is possible to mathematically derive the topography of the corneal surface.

Figure 3.1 shows an example of the output produced by a corneal topographer for a member of each patient class. These represent *illustrative* examples of each class, i.e. they are designed to be easily distinguishable by simple visual inspection. The top portion of the figure shows the imaged concentric rings. The bottom portion of the image shows a false color map representing the surface curvature of the cornea. This color map is intended to aid clinicians and is largely instrument-dependent.

Many methods of interpretation exist and in clinical practice, domain expertise is the usual basis for interpretation. There is a lack of broadly accepted standards among domain experts and instrument manufacturers about the best way to interpret the data, which usually results in a qualitative task of pattern recognition from the false color maps.

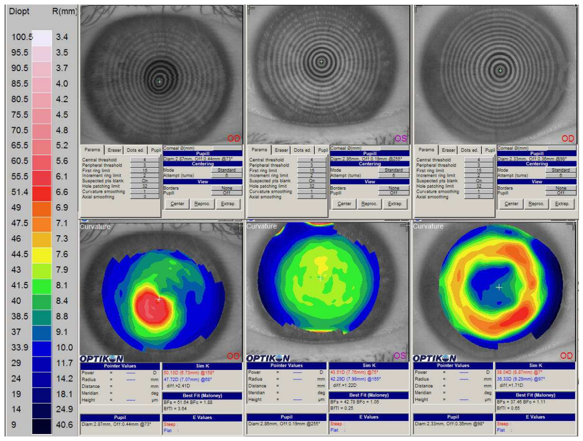


Figure 3.1: Characteristic corneal shapes for each of the three patient groups (easily-distinguishable examples for illustrative purposes). The top image shows a picture of the cornea and reflected concentric rings. The bottom image shows the false color topographical map representing corneal curvature, with an increased curvature given a color in the red spectrum, decreased curvature in blue. From left to right: Keratoconus, indicated by the central steepening of the cornea; Normal; and post-LASIK, with the typical central flattening of the cornea.

3.2 Related Work

Schwiegerling et al. were among the first to model the shape of the cornea with radial polynomials [45]. For a fixed polynomial complexity and using a set of videokeratoscopic height data representing the corneal surface, they computed the Zernike representation at each point on the surface, followed by a least-squares fit of the model to the original data. In a more comprehensive study, Iskander et al. examined the performance of both Zernike and pseudo-Zernike polynomials, with the goal of determining which transformation would provide the most accurate reconstruction of the corneal surface [10]. As one might expect, they found that for a given order, the pseudo-Zernike polynomials yielded a more faithful representation. However, in a follow-up publication, the same group concluded that a 4th order Zernike polynomial was adequate in modeling the majority of corneal surfaces [11]. Smolek and Klyce report that Zernike polynomials may fail to model all the information that influences visual acuity and for this reason, should not be used in clinical diagnosis [46]. While they raise a valid concern, it is unlikely that any domain-independent transformation will be able to adequately capture every possible feature while still reducing the overall dimensionality of the data.

A number of studies that seek to classify corneal shape rely on statistical summaries to represent the data [47–50]. These studies used multi-layer perceptrons or linear discriminant functions in classification. Smolek and Klyce have published a number of studies on modeling and classifying corneal topography [33,51]. They have examined the problem of distinguishing between normal and post-operative corneas through the use of one-dimensional wavelets and neural networks. They report impressive results, but use a very small dataset in their experiments. Carvalho recently published a study where he modeled the shape of the cornea using a 4th order Zernike polynomial and used each of the 15 resulting moments as an input to a neural network with five possible outputs. While his accuracy results are not as high as those reported by Smolek and Klyce, he does achieve an accuracy of close to 80% [32]. Twa et at. were the first to examine whether Zernike polynomials could be used with decision trees for corneal classification [12]. They report 85% accuracy with a single C4.5 decision tree [29], and values above 90% when they combine a decision tree with ensemble learning techniques such as boosting or bagging. In a later study, they modeled a dataset using a 7th order Zernike polynomial and compared the performance of a C4.5 decision tree against several proprietary and domain-specific classification strategies, finding it to be equal to or better than competing approaches [52].

3.3 Dataset

The dataset for our experiments in the classification of corneal shape, which we refer to as the *Topography* dataset, consists of examination data of 254 eyes obtained from a clinical corneal topography system, the Optikon Keratron. The data was examined by an expert and divided into three groups based on the clinical diagnosis. The divisions are given below, with the number of patients in each group listed in parentheses:

1. Normal ($n = 119$)
2. Post-operative myopic LASIK ($n = 36$)
3. Keratoconus ($n = 99$)

The difference in corneal shape between the patient groups is as follows: compared with normal corneas, post-operative myopic LASIK corneas are distorted, but generally have a flatter than normal center curvature with an annulus of greater than normal curvature in the periphery. Corneas that are diagnosed as having keratoconus are also more distorted than normal corneas. Unlike the LASIK corneas, they generally have localized regions of steeper than normal curvature at the site of tissue degeneration, but often have normal or flatter than normal curvature elsewhere.

The data files from the corneal topographer consist of a 3D point cloud of approximately 7000 spatial coordinates arrayed in a polar grid. The height of each point z is specified by the relation $z = f(\rho, \theta)$, where the height relative to the corneal apex is a function of radial distance from the origin (ρ) and the counter-clockwise angular deviation from the horizontal meridian (θ). The inner and outer borders of each concentric ring consist of a discrete set of 256 data points taken at a known angle θ, but a variable distance ρ, from the origin.

The number of data points is a function of the topographer's sampling resolution and should be high enough to capture any surface irregularities that might be present. It is not feasible, however, to use this raw data as a feature vector for classification. There is almost no way to guarantee that point X of one record refers to the same surface location as point X of another. Therefore, in order to make a valid comparison, we must first transform the data.

3.4 Modeling Topography Using Zernike Polynomials

Our polynomial modeling technique is based on methods described in detail by Schwiegerling *et al.* and Iskander *et al.* [10, 45]. In summary, the data is modeled using either polynomial transformation method over a user-selected circular region of variable-diameter centered on the axis of measurement. The generation of the Zernike and pseudo-Zernike model surface proceeds in an iterative fashion, computing a point-by-point representation of the original data at each radial and angular location up to the user-specified limit of polynomial complexity. The polynomial coefficients of the surface that will later be used to represent the proportional magnitude of specific geometric features are computed by performing a least-squares fit of the model to the original data, using standard matrix inversion methods [45].

The polynomials produced in this manner are orthogonal or approximately orthogonal. However, it may not necessarily be the case that polynomials fit to discrete data using the least-squares process are orthogonal. This is especially true when using radial sampling and a small number of data points. However, for a large number of data points, as is the case here, they will be approximately orthogonal [53]. Given a small number of points, however, a Gram-Schmidt orthogonalization procedure can be used to ensure orthogonality [54].

Once we have computed the polynomial coefficient vector, we compute the residual, or root-mean-square (RMS) error between our model and the actual surface data by the following relationship:

$$RMS \ = \ \frac{1}{\sqrt{D}} \| C_O - \hat{C}_{z,pz} \| \tag{3.1}$$

with $\hat{C}_{z,pz}$ representing the surface created using the coefficients computed through least-squares estimation, C_O the original surface elevations (both surfaces stored in vector form), D the number of data points and $\| \cdot \|$ the Euclidean norm. We calculate the RMS to determine whether using a more faithful representation of the corneal surface translates into higher classification accuracy. RMS error is just one of many methods used to evaluate the goodness-of-fit of a model. Alternate measures can be employed if desired.

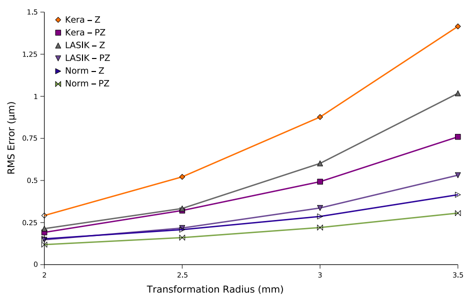
3.5 Determining Model Fidelity

Here we cover our experiments in modeling the Topography dataset with Zernike and pseudo-Zernike polynomials. Unfortunately, there is no consensus within the vision community as to what parameters one should use when computing a Zernike or pseudo-Zernike polynomial transformation on corneal topography data. Therefore, we elect to test and compare a number of different parameter values.

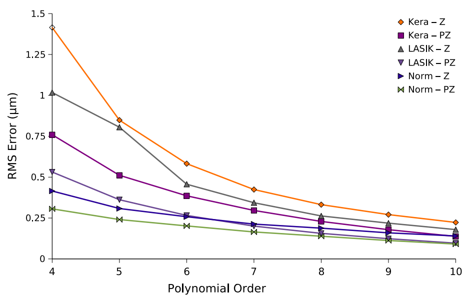
3.5.1 Experiments

When modeling the surface of the cornea, it is possible to vary both the order of polynomial complexity and the maximum radius, i.e. the size of the region of interest. Both values have an effect the computational cost of transforming the data. Changing the former changes the number of moments or coefficients final feature vector. Varying the latter changes the number of datapoints used in the transformation. Care must be taken when selecting this value. If the radius is too small, the relevant surface features may not be captured. If it is too large, there is a chance that the outer rings will not contain a “complete” set of data points. This often occurs due to the presence of a nose or eyelid shadow in the image. Having an incomplete ring will cause errors in the transformation process, producing a number of unwanted edge effects. We elected to test four different radii values that we felt were appropriate for our application: 2.0, 2.5, 3.0 and 3.5 mm.

While we are primarily interested in using these transformations as input for classification, we would like to know the fidelity of our model to the original surface. We computed fourth through tenth order Zernike and pseudo-Zernike transformations on the four radii values mentioned above, yielding a total of 56 transformations for each record (7 orders * 4 radii * 2 series). To test the effectiveness of the polynomials as a modeling method, we compute the residual modeling error as a function of the polynomial order for each radius. The modeling error was calculated by determining the RMS error between the transformation and the original data record.



(a)



(b)

Figure 3.2: (a) RMS error (μm) for a 4th order polynomial fit as a function of transformation radius (mm) for each patient class (Kera - Keratoconus, LASIK, Norm - Normal) and transformation (PZ - pseudo-Zernike, Z - Zernike). (b) RMS error (μm) as a function of polynomial order. The fit radius is fixed at 3.5 mm. The error is shown for each patient class (Kera - Keratoconus, LASIK, Norm - Normal) and transformation (PZ - pseudo-Zernike, Z - Zernike). For clarity, error bars are not shown, but the magnitude of the standard deviation is typically 40 to 50% of the mean for the normal and LASIK values and similar in magnitude to the mean for the keratoconus values.

3.5.2 Empirical Evaluation

We now compare the results of the different spatial transformations on the Topography dataset. We use the residual modeling error as our method of comparison. While it is possible to compute the residual error in modeling left and right eyes, we only compute the error for right eyes. In our dataset, the post-operative LASIK class contains patient records representing only right eyes, so to remain consistent, we just use the right eye transformations when computing error. Please note that is not uncommon for studies of this nature to focus on only right or left eyes for analysis [32].

In Figures 3.2 (a) and (b), we present the average model error as a function of transformation radius (Fig. 3.2 (a)) and polynomial order (Fig. 3.2 (b)). We find that residual error for the pseudo-Zernike transformations is about half that of a Zernike polynomial for the same order. A more sizable difference is found between the different patient groups. Keratoconus corneas are associated with residual errors greater than normal eyes and LASIK eyes have approximately the same or slightly greater residual error than normal. These results have anatomical validity, as a patient suffering from keratoconus is likely to have much greater distortion of corneal shape than either normal or LASIK eyes and these distortions will be more difficult to model accurately.

In addition, residual error increases along with the radius of transformation. As the radius increases, the number of data points that must be considered increases as well. This results in a larger surface that must be modeled and since the complexity of the transformation remains the same, we are left with a larger error. We also find that in most cases, as the radius of the transformation increases, the residual error of the pseudo-Zernike transformation grows at a slower rate than the residual error of a corresponding Zernike transformation. This effect is more pronounced at lower polynomial orders and agrees with previous studies [20].

Also of note is the fact that residual error appears to be related to the number of coefficients used in the transformation. For instance, a 7th order pseudo-Zernike transformation results in a polynomial with 66 coefficients. A 10th order Zernike transformation will yield a polynomial with 64 terms. Comparing the average residual errors for these two transformations, we find that they are very similar. Comparable trends are seen when examining the other transformations with similar numbers of coefficients (6th PZ and 8th Z, 5th PZ and 7th Z, 4th PZ and 6th Z).

3.6 Classification Using Zernike Polynomials

This section details our tests to determine whether one can use a corneal surface model derived from Zernike polynomials to effectively classify the topography dataset. These experiments involve an extensive comparison of several popular classification methods to determine which performed best with respect to classification accuracy, where accuracy is defined as the number of correct classifications divided by the total number of classifications (expressed as a percentage). The classification methods that we test include an ID3-based decision tree (*C4.5*) [28], Voting Feature Intervals (*VFI*) [36], Naïve Bayes (*NB*) [26], Random Forests (*RF*) [39] and a neural network (*NN*) [30]. We run each classifier with boosting (*Bst*) [55] and bagging (*Bag*) [38], except for the neural network and random forests, which we do not boost or bag. We conduct all of our classification experiments using the Weka Data Mining Toolkit v3.4[3] on a 2.4GHz Pentium 4 workstation with 512 MB of RAM running Windows XP and Sun Java2 v1.4.2. All experiments were run using ten-fold cross-validation [56].

For the ID3-based decision tree, we choose C4.5. The Weka version of C4.5 implements Revision 8, the last public release of C4.5 [24, 28, 29]. We use the following parameters in our experiments: binary splitting on attributes, confidence of 0.25, and a minimum number of 2 instances per node. With the VFI experiments, we set the bias parameter to 0.6. We keep Weka's default settings when classifying with Random Forests (10 iterations). For the neural network, we use a multi-layer perceptron trained using back propagation [30]. We evaluate a number of different parameter settings, but achieved the highest accuracy when the perceptron is set to autobuild and run for 500 iterations, with a learning rate of 0.3, a momentum of 0.2, and a

[3]http://www.cs.waikato.ac.nz/∼ml/weka/

single hidden layer containing as many nodes as there are attributes in the feature vector.

For the bagging experiments, we set the bag size parameter to 100% and the number of iterations to 10. The boosting experiments use the AdaBoost.M1 [55] method with a weight threshold of 100, again with 10 iterations. The base classifiers that are boosted and bagged use the same parameters as the non-boosted and non-bagged versions.

3.6.1 Dataset Partitioning

While we are primarily interested in the performance of our models on the entire topography dataset, we are also interested in whether using a simpler, two-class dataset has any effect on accuracy over the more challenging three-class problem. Thus, we repeat all of our classification experiments using different subgroups of the original dataset. The different subgroups, or partitions, are listed in Table 3.1. The *all* partition reflects the original dataset, the others the two-class subgroups. One note on the *NL-K* partition: The *NL-K* group combines the normal and LASIK corneas into one class and places those labeled as having keratoconus into the other. The intent of this test is to see how well we could distinguish normal eyes (both with and without surgical treatment) from diseased.

3.6.2 Empirical Evaluation

We now discuss the results of our experiments. In all of the results tables, "Order" refers to the polynomial order of the transformation (varies from 4 to 10), "Radius" the maximum radius examined in the data file (2.0, 2.5, 3.0 or 3.5 *mm*), "PZ" corresponds to a pseudo-Zernike transformation, and "Z" a Zernike transformation. The

Name	Description
all	Classify between all patient groups (Normal, Keratoconus and LASIK)
K-L	Classify between Keratoconus and LASIK
K-N	Classify between Keratoconus and Normal
L-N	Classify between LASIK and Normal
NL-K	Combine Normal and LASIK into one group and classify against Keratoconus.

Table 3.1: Partitions of the dataset used in the various classification experiments.

values listed under the "Dataset" column are the dataset partitions listed in Table 3.1. For the classifier labels, we generally refer to the full name of the classifier, though we do abbreviate on occasion. In those instances, C4.5 refers to the C4.5 decision tree; VFI to Voting Feature Intervals; NB to the simple Naïve Bayes classifier; RF to Random Forests; and NN to classification by multi-layer perceptron. Unless otherwise noted, all classification results are listed as the percentage of correct classification for the dataset. In each table, we also list, for each polynomial order and transformation, whether the other classifiers are statistically better or worse than a single C4.5 decision tree. Statistical significance is determined using a paired t-test at the 5% significance level. Those classifiers that are better than C4.5 are denoted with a 'v'. Those that are worse are are marked with a '*'.

While we originally conducted our experiments using boosting and bagging with the C4.5, Naïve Bayes and VFI classifiers, we found they only had a positive effect on the decision trees. Thus, we have decided only to present the ensemble-learning results with C4.5. In addition, we focus primarily on the complete (*all*) dataset.

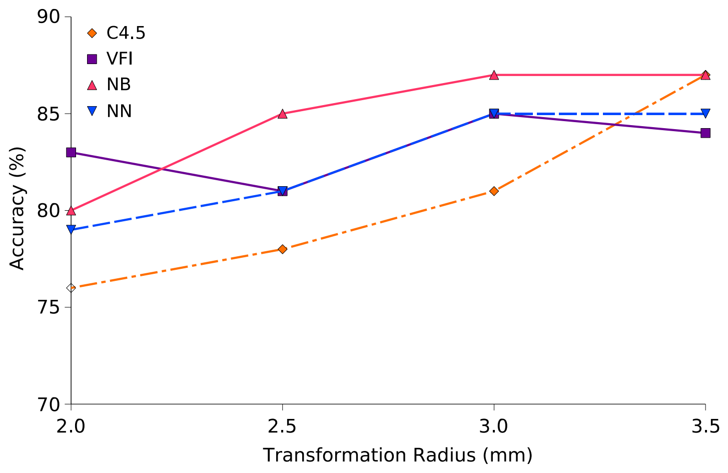


Figure 3.3: Classification accuracy as a function of the transformation radius (in mm) for the standard classifiers tested: C4.5, Voting Feature Intervals (VFI), Naïve Bayes (NB) and a neural network (NN). The graph depicts results for a 4th order Zernike transformation on the *all* dataset partition.

Trying to distinguish between the three patient classes presents the greatest classification challenge. We mention results obtained from the other dataset partitions when germane.

Accuracy versus Radius

When examining the classification results using the standard classifiers, we find that in most cases, *as the radius increases, the classification accuracy increases as well*. We present these results graphically in Figure 3.3. To improve readability, we limit the graph to just the 4th order Zernike results on the *all* partition. The 4th order Zernike results were chosen because they are illustrative of the trends seen with the other transformations. The numbers for each order over the different partitions tend to vary slightly, but the overall trends remain the same. The direct relationship between accuracy and radius is apparent for every classifier except the neural network with the higher order pseudo-Zernike transformations (7th and above). In these cases, accuracy tends to drop as the radius is increased. However, for the Zernike and low-order pseudo-Zernike transformations, the direct relationship holds. This relationship is likely due to the fact that when we increase our surface model, we are better able to capture the surface irregularities that distinguish between the different classes. For example, with a small radius, a cornea suffering from keratoconus might present as normal if the corneal bulge is not central. Using a larger radius value decreases the chance of such a misclassification.

Order versus Accuracy

When discussing the rest of our classification results, we will focus on those achieved with a transformation radius of 3.5 mm. Using this radius ensures that we capture most, if not all, of the irregularities that may be present over the measured diameter of the corneal surface. It also tended to produce the highest accuracy values. Table 3.2 shows the accuracy results of the classifiers tested on that radius for each dataset partition. While increasing the radius generally has a positive effect on classification accuracy, no such trend emerges when the polynomial order of the transformation is increased.

For the C4.5 classifier, we found that classification accuracy stayed fairly constant (within one or two percentage points) as the polynomial order of the transformation was increased from 4 to 10. In addition, there was not a great deal of difference in classification accuracy between the two transformations. The C4.5 decision trees consistently provided classification results above 85%, and for some dataset partitions

(K-L and L-N), had accuracy results above 90%. The Random Forest classifier remains stable as well, with accuracy ranging between 87% and 90%, which is roughly the same as for C4.5 with boosting or bagging.

The VFI classifier provided comparable results to the C4.5 classifier. The only difference is that with the K-L partition, the results became much worse as the polynomial order was increased. This result was especially true for the pseudo-Zernike transformation. In addition, while the C4.5 readings stayed fairly constant as the polynomial order was increased, the VFI results tended to follow a bell-shaped pattern, improving until a 6th or 7th order transformation, then decreasing.

The Naïve Bayes classifier provided acceptable results (above 80%) for lower order transformations, but as the polynomial order was increased, the classification accuracy dropped rather quickly. The only real exception to this result was when classifying the L-N partition. For that dataset, classification accuracy remained above 90%, because the patients within that partition were easily separated by a couple of attributes. Adding the keratoconus patients blurred this separation. As with the VFI classifier, the drop in the accuracy of the Naïve Bayes classifier is likely due to the noise that is included in the more complex polynomial models. For these Bayesian-based strategies, the additional, and largely unnecessary, attributes have an adverse effect on performance. The neural network also suffers a similar fate, though the effect is more pronounced on the pseudo-Zernike datasets. Classifiers like decision trees, on the other hand, essentially ignore the values, which accounts for their stability.

Number of Coefficients versus Accuracy

We next compare the accuracy of Zernike and pseudo-Zernike transformations with a similar number of coefficients. Previous studies that focused on the fidelity

	Data-	C4.5		Bagged C4.5		Boosted C4.5		VFI		Naïve Bayes		Random Forests		Neural Network	
Order	Set	(PZ)	(Z)	(PZ)	(Z)	(PZ)	(Z)	(PZ)	(Z)	(PZ)	(Z)	(PZ)	(Z)	(PZ)	(Z)
4	*all*	86	87	89	89	90	89	89	84	86	87	90	89	86	85
5	*all*	86	88	88	90	90v	89	89	88	81	85	90	89	87	85
6	*all*	87	87	88	90	88	89	88	89	76*	83	88	90	82*	86
7	*all*	88	85	90	88	90	88	88	88	71*	78*	89	88	78*	86
8	*all*	86	86	89	89	89	89	85	88	71*	75*	87	89	76*	84
9	*all*	84	87	89v	89	88	89	84	86	71*	74*	87	88	70*	87
10	*all*	88	86	90	88	89	89	81*	86	68*	73*	87	87	50*	81*
4	*K-L*	92	92	95	92	95	95	95	88	81*	82*	96	94	85	85
5	*K-L*	90	93	93	93	96v	95	93	92	68*	77*	96v	95	85	85
6	*K-L*	91	94	93	94	94	97	91	93	60*	72*	95	97	83	85
7	*K-L*	92	94	93	94	96v	96	90	92	54*	66*	95	96	81*	84
8	*K-L*	92	92	94	93	95	96	82*	89	52*	60*	93	95	81*	85
9	*K-L*	92	92	95	93	96v	96	76*	88	49*	59*	93	95	76*	86
10	*K-L*	93	92	93	93	96	96	69*	84	53*	56*	92	94	58*	85
4	*K-N*	88	89	90	91	91	91	90	91	91	91	90	90	84	86
5	*K-N*	89	88	91	90	90	90	90	92	87	90	90	91	85	85
6	*K-N*	90	87	89	89	89	90	90	92	84	89	90	90	83*	85
7	*K-N*	88	85	90	89	89	88	90	91v	81*	87	90	90	80*	84
8	*K-N*	87	86	89	89	88	89	90	90	77*	84	90	89	80*	84
9	*K-N*	85	85	90v	90v	88	89v	89	90v	75*	83	89	90v	77*	86
10	*K-N*	88	87	91	89	90	90	89	90	73*	80*	89	90	74*	85
4	*L-N*	95	93	97	95	96	96	94	91	96	96	98	96	98	95
5	*L-N*	95	94	97	95	96	96	94	91	95	95	98	96	97	97
6	*L-N*	96	95	97	96	96	96	94	95	95	96	98	98	96	97
7	*L-N*	97	95	98	96	97	96	92	95	94	94	97	98	97	97
8	*L-N*	96	95	97	96	96	96	92	94	94	93	97	97	96	98
9	*L-N*	97	95	97	96	97	95	91	94	93	93	96	96	96	96
10	*L-N*	96	95	97	96	96	95	91	92	91	94	96	96	95	95
4	*NL-K*	89	90	91	91	91	91	89	90	87	88	90	91	88	88
5	*NL-K*	89	90	91	92	91	92	90	90	83*	88	90	91	87	88
6	*NL-K*	87	89	90	91	91	91	89	90	79*	85	89	91	86	88
7	*NL-K*	89	89	90	90	91	90	89	91	77*	83*	89	89	85	87
8	*NL-K*	87	89	90	91	90	91	87	90	77*	80*	88	90	85	87
9	*NL-K*	87	87	89	90	88	90	86	88	75*	78*	88	89	80*	88
10	*NL-K*	87	88	90	90	90	89	84	89	76*	79*	88	89	64*	87

Table 3.2: Classification accuracy for Zernike and pseudo-Zernike transformations at each polynomial order with a radius of 3.5 *mm*. For a given order and transformation, those classifiers that are statistically better than a single C4.5 decision tree are denoted with a 'v', while those that are worse are marked with a '*'. Significance is determined using a paired t-test at 5% significance.

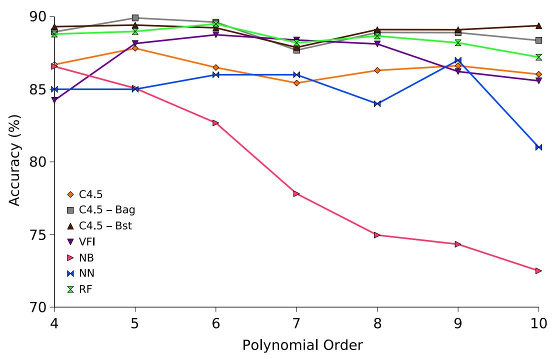


Figure 3.4: Accuracy as a function of polynomial order for the classifiers tested: C4.5, C4.5 with bagging (C4.5 - Bag), C4.5 with boosting (C4.5 - Bst), Voting Feature Intervals (VFI), Naïve Bayes (NB), a neural network (NN) and Random Forests (RF). Graph depicts results for an increasing polynomial order on a 3.5 *mm* radius Zernike transformation on the *all* dataset partition.

of the two transformations largely compared representations of the same polynomial order [10]. Unlike those studies, we attempt to make an “apples to apples” comparison by examining representations with a similar number of coefficients. In Table 3.3, present the results of this study. The numbers in the table reflect those obtained with a transformation radius of 3.5 *mm*. When examining the values of C4.5, C4.5 with boosting or bagging, VFI or Random Forests, one can see that there is no real difference between the accuracy of Zernike or pseudo-Zernike polynomials.

All of the trends discussed above are illustrated using the results of a 4th order Zernike transformation in Figure 3.4. While the accuracy values in the figure represent 3.5 *mm* transformation radius on the *all* dataset partition, the relationships were similar across the tested conditions (i.e. transformation, polynomial order, radius of transformation, and classification method).

		C4.5				Naïve	Random	Neural
Order	Transform	W/O	Bag	Boost	VFI	Bayes	Forests	Network
4	PZ	86	89	90	89	86	90	86
6	Z	87	90	89	89	83	90	86
5	PZ	86	88	90v	89	81	90	87
7	Z	85	88	88	88	78*	88	86
6	PZ	87	88	88	88	76*	88	82
8	Z	86	89	89	88	75*	89	84
7	PZ	88	90	90	88	71*	89	78
10	Z	86	88	89	86	73*	87	81

Table 3.3: Classification accuracy for Zernike and pseudo-Zernike transformations with a similar number of coefficients at each polynomial order with a radius of 3.5 *mm* on the *all* dataset. For a given order and transformation, those classifiers that are statistically better than a single C4.5 decision tree are denoted with a 'v', while those that are worse are marked with a '*'. Significance is determined using a paired t-test at 5% significance.

3.6.3 Discussion

While accuracy is the most important factor in choosing a classification strategy, another attribute that should not be ignored is the overall interpretability of the final results. Decision trees have often been favored over (possibly) more accurate "black-box" classifiers such as neural networks because they provide more understandable results. In medical image interpretation, decision support for a domain expert is often preferred to an automated classification made by an expert system. While it is important for a system to provide a decision, it is often equally important for clinicians to know the basis for an assignment. Decision trees provide a more transparent view of how these classifications are derived as well as the features involved.

In Table 3.4 we summarize the comparisons of this study. We list each classifier that we tested as well as how it fares in the following categories: accuracy; speed, or the time needed to create the model; scalability, or whether increasing the size of the dataset affects the time needed to create a model; interpretability, if the criteria used for classification can be gleaned from the classification model; and stability, how the accuracy varies as the representation changes. Overall, we found the speed, accuracy, stability and interpretability of decision trees preferable to other methods for this dataset. The accuracy of C4.5 was among the highest of the standard classification methods, but it can be further improved through meta-learning techniques such as boosting or bagging. We observed that when the adaBoost algorithm was combined with C4.5, our best classification results were achieved with a fourth order polynomial model. With bagging, the best results were achieved with a fifth order model. This improvement comes at a loss of speed and interpretability, however. Random Forests tended to perform as well as C4.5 with boosting or bagging. The Random Forests algorithm creates multiple decision trees during each classification run, which essentially incorporates a form of bagging into its normal operation, so such results are not surprising. Voting Feature Intervals provides reasonably high accuracy, but the classification model is not as stable as that of a decision tree, with accuracy decreasing as the polynomial order increases. The same was true for the Naïve Bayes classifier. A likely explanation for this result is that higher order polynomial models tend to fit both features and noise, which could reduce accuracy. This could be a big disadvantage in cases where higher order polynomials are needed to faithfully capture features in other applications not considered here.

We found that the neural network was also much less stable than the decision tree algorithms, suffering from a similar drop in accuracy at higher orders as seen with VFI and Naïve Bayes. In addition to the lack of stability, the time needed to construct and test the neural network was much greater than all of the other methods tested. While we could examine an entire dataset (both transformations, all 7 orders and all four radii) in just minutes with C4.5, VFI or Naïve Bayes, or a few hours with the ensemble-learning algorithms, it took days to complete a single neural network test.

The argument could be made that a different set of neural network parameters might yield faster, superior results, especially on the higher order data. Such a claim may be true, but the same argument could be made for every other classifier (except the parameter-free Naïve Bayes). Furthermore, we tested a large number of different network parameter values, varying the number of training iterations and hidden layers, whether the learning rate was set to decay, or a if validation set was used. Those results were much lower than the results reported here – often by 20-25%. While it can be argued that our chosen network structure may been inadequate, there is no systematic method for determining a better one.

We were able to achieve a classification accuracy of 90% using bagged or boosted decision trees, which are constructed in an unsupervised manner. In an application such as patient diagnosis, where there is a possibility of incorporating additional data (as the number of patient screenings increases), an unsupervised classification algorithm would be far more beneficial that one that must be tuned by an expert every time the underlying data changes. For these reasons, we cannot recommend neural networks as a classification method for this task.

In light of our findings, there does not appear to be any benefit to using pseudo-Zernike polynomials as a transformation method. Previous studies stated that the lower RMS error of pseudo-Zernike polynomials implied that they were more robust to

Criteria	C4.5	NN	VFI	NB	META
Accuracy	$+$	$\pm$	$+$	$\pm$	$++$
Speed	$+$	$--$	$+$	$+$	$-$
Scalability	$+$	$--$	$+$	$+$	$\pm$
Interpretability	$++$	$--$	$-$	$-$	$-$
Stability	$++$	$-$	$-$	$-$	$++$

Table 3.4: Comparison of classification methods. C4.5 refers to the decision tree classifier, NN to the neural network, VFI to Voting Feature Intervals and NB to Naïve Bayes. META refers to C4.5 with boosting (or bagging) and Random Forests.

noise and therefore would provide a more faithful surface model [20]. Those comparisons were only made over equivalent polynomial orders, however. While our studies do support that conclusion, when we compare representations with a similar number of coefficients, we find that a Zernike transformation yields an equal or higher-fidelity model than the corresponding pseudo-Zernike polynomial. In any event, the additional model fidelity does not translate into increased classification performance. If the additional coefficients gained from the pseudo-Zernike transformation were indeed useful in distinguishing between the patient classes, classification accuracy would increase along with the polynomial order of the transformation, but it does not. One can make a similar argument against using the higher-order Zernike polynomials as well. The highest accuracy generally occurs when classifying with a 4th, 5th or 6th order Zernike model, regardless of patient category. These results suggest that overall, a fourth order polynomial has both sufficient complexity and the fidelity needed to correctly distinguish the clinical conditions studied here.

3.7 Visualization of Classification Decisions

When discussing experiments relating to the Biomedical Knowledge Discovery stage of our workflow, accuracy is the primary metric used for evaluation. While accuracy is an important factor in choosing a classification strategy, another attribute that should not be ignored is the interpretability of the final results. We have spoken about the transparency of decision trees and why they may be favored over (possibly) more accurate "black-box" classifiers because they provide more understandable results. In medical image interpretation, decision support for a domain expert is preferable to an automated classification made by an expert system. While it is important for a system to provide a decision, it is often equally important for clinicians to know the basis for an assignment. It is in that spirit that we extend our work on the classification of corneal shape to include a method for the visualization of classification criteria to aid in clinical decision support.

Taking the Topography dataset, we model the corneal surface using Zernike polynomials. We then use the polynomial coefficients for each geometric mode as an input feature for a decision tree. Since the polynomial coefficients used as splitting attributes represent the proportional contributions of specific geometric modes, we are able to create a surface representation that reflects the spatial features deemed "important" in classification. These features discriminate among the patient classes and give an indication as to the specific reasons for a decision.

We have designed and developed a method of visualizing our decision tree results. We partition our dataset based on the path taken through the decision tree and create a surface from the mean values of the polynomial coefficients of all the patients falling into that partition. For each patient, we create an individual polynomial surface

Polynomial	Azimuthal Frequency (m)								
Order (n)	-4	-3	-2	-1	0	1	2	3	4
0					0				
1				1		2			
2			3		4		5		
3		6		7		8		9	
4	10		11		12		13		14

Table 3.5: Combinations of radial polynomial order (n) and azimuthal frequency (m) that yield the 15 terms comprising a 4th order Zernike polynomial.

from the patient's coefficient values that correspond to the splitting attributes of the decision tree. We contrast this surface against a similar surface containing the mean partition values for the same splitting coefficients and provide a measure to quantify how "close" a patient lies to the mean.

3.7.1 Algorithm

Figure 3.6 shows an example decision tree. In this case, the Z value in each node (diamond) refers to the jth term of a Zernike polynomial. For clarity, we repeat the table and figure from Chapter 2 that illustrate the terms in a 4th order Zernike polynomial as well as the relation between the two Zernike indexing schemes.

Classification occurs by comparing a patient's Zernike coefficient values against those in the tree. For instance, the value of Z_7 for a patient is compared against the value listed in the top node. If the patient's value is less than or equal to 2.88, we traverse the right branch. If not, we take the left (in all cases, we take the right branch with a *yes* decision and the left with a *no*). With this example, a *no* decision would lead to a classification of K, or Keratoconus. If the value of Z_7 *was* less than or equal to 2.88, we would proceed to the next node and continue the process until

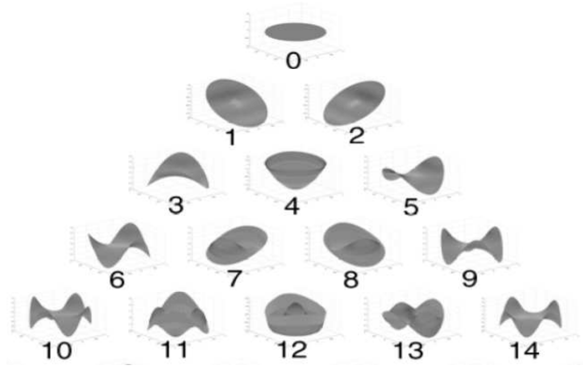


Figure 3.5: Graphical representation of the modes that constitute a 4th order Zernike polynomial. Each mode is labeled using the single indexing convention.

we reached a leaf and the corresponding classification label (K = Keratoconus, N = Normal, L = LASIK).

We can treat each possible path through a decision tree as a rule for classifying an object. Using the tree in Fig. 3.6, we construct rules for the first four leaves, with the leaf number (arbitrarily assigned) listed below the class label:

1. $Z_7 > 2.88 \rightarrow \mathit{Keratoconus}$
2. $Z_7 \leq 2.88 \ \wedge \ Z_{12} > -4.04 \ \wedge \ Z_5 > 9.34 \rightarrow \mathit{Keratoconus}$
3. $Z_7 \leq 2.88 \ \wedge \ Z_{12} \leq -4.04 \ \wedge \ Z_0 \leq -401.83 \rightarrow \mathit{Keratoconus}$
4. $Z_7 \leq 2.88 \ \wedge \ Z_{12} \leq -4.04 \ \wedge \ Z_0 > -401.83 \rightarrow \mathit{LASIK}$

In each rule, the "$\wedge$" symbol is the standard *AND* operator and the "$\rightarrow$" sign is an implication stating that if the values of the coefficients for a patient satisfy the expression on the left, the patient will be assigned the label on the right. For a given dataset, a certain number of patients will be classified by each rule. These patients

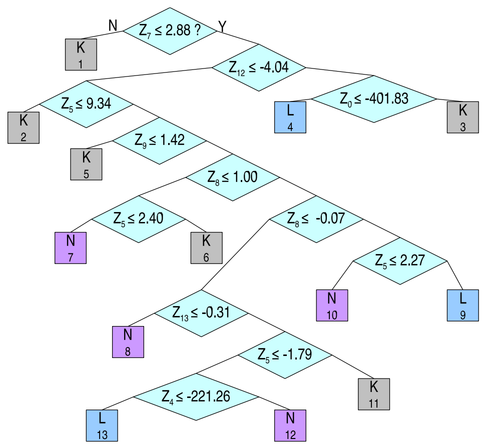


Figure 3.6: Decision tree for a 4th order Zernike polynomial.

will share similar surface features. As a result, one can compare a patient against the mean attribute values of all the other patients who were classified using the same rule. This comparison will give clinicians some indication of how “close” a patient is to the rule average. To compute these “rule mean” coefficients, denoted $\overline{rule}$, we partition the training data and take the average of each coefficient over all the records in that particular partition.

For a new patient, we compute the Zernike transformation and classify the record using the decision tree to determine the rule for that patient. Once this step has been completed, we apply our visualization algorithm,to produce five separate images (illustrated in Fig. 3.7). The first panel is a topographical surface representing the Zernike model for the patient. It is constructed by plotting the 3-D transformation surface as a 2-D topographical map, with elevation denoted by color. The second section contains a topographical surface created in a similar manner by using the

$\overline{rule}$ coefficients. These surfaces are intended to give an overall picture of how the patient's cornea compares to the average cornea of all similarly-classified patients.

The next two panels in Fig. 3.7 (rows 3 and 4) are intended to highlight the features used in classification, i.e. the distinguishing surface details. We call these surfaces the *rule surfaces*. They are constructed from the value of the coefficients that were part of the classification rule (the rest are zero). For instance, if a patient were classified using the second rule in the list above, we would keep the values of Z_7, Z_{12} and Z_5. The first rule surface (third panel of Fig. 3.7) is created by using the relevant coefficients, but instead of using the patient-specific values, we use the values of the $\overline{rule}$ coefficients. This surface will represent the mean values of the distinguishing features for that rule. The second rule surface (row 4, Fig. 3.7) is created in the same fashion, but with the coefficient values from the patient transformation, not the average values.

The actual construction of the surfaces is achieved by running our transformation algorithm in reverse, updating the values in the coefficient matrix depending on the surface being generated. Once the values have been computed, we are left with several 3-D polar surfaces. In the surfaces, the base elevation is green. Increasing elevation is represented by red-shifted colors; decreased elevation is represented by blue-shifted colors.

Finally, we provide a measure to show how close the patient lies to those falling in the same rule partition. For each of the distinguishing coefficients, we compute a relative error between the patient and the $\overline{rule}$. We take the absolute value of the difference between the coefficient value of the patient and the value of the $\overline{rule}$ and divide the result by the $\overline{rule}$ value. We provide a bar chart of these errors for each

Rule	Class Label	Expression	Support	Support Percentage
1	K	$Z_7 > 2.88$	47	66.2%
8	N	$Z_7 \leq 2.88 \ \wedge \ Z_{12} > -4.04 \ \wedge \ Z_5 \leq 9.34 \ \wedge$ $Z_9 \leq 1.42 \ \wedge \ -0.07 > Z_8 \leq 1.00 \ \wedge \ Z_{13} > -0.31$	74	89.2%
4	L	$Z_7 \leq 2.88 \ \wedge \ Z_{12} \leq -4.04 \ \wedge \ Z_0 > -401.83$	19	79.2%

Table 3.6: Strongest Decision Tree Rules and Support

coefficient (the error values of the the coefficients not used in classification are set to zero). This plot is intended to provide a clinician with an idea of the influence of the specific geometric modes in classification and the degree that the patient deviates from the mean. To provide further detail, if the patient's coefficient value is less than the $\overline{rule}$ coefficient, we color the bar blue. If it is greater, we color it red.

3.7.2 Results

Using a C4.5 decision tree generated from the classification of the dataset on a 4th order Zernike transformation (Fig. 3.6), we provide an example of our visualization technique on a record from each patient class.

As stated previously, we must reclassify our training data after generating the decision tree in order to partition the data by rule. We expect that certain rules will be *stronger* than others, i.e. a larger portion of the class will fall into those rule's partition. Table 3.6 provides a list of the strongest rule for each class. We provide the rule number, its class label and its *support*, or the number of patients that were classified using that rule. We also provide the *support percentage* of the rule, which is equal to support divided by the total number of patients in that class. These numbers reflect the classification of the training data *only*, not the entire dataset (the data was

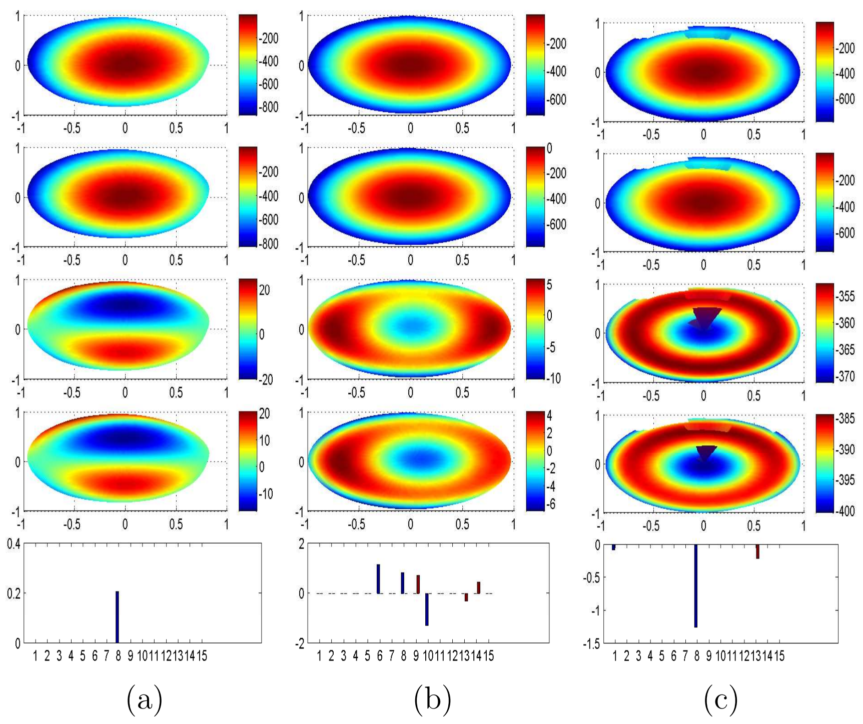


Figure 3.7: Example surfaces for the strongest rule in each patient class. Figure (a) represents a Keratoconic eye (Rule 1), (b) a Normal cornea (8), and (c) a post-operative LASIK eye (4). The top panel contains the Zernike representation of the patient. The next panel illustrates the Zernike representation using the $\overline{rule}$ values. The third and fourth panels show the *rule surfaces*, using the patient and $\overline{rule}$ coefficients, respectively. The bottom panel consists of a bar chart showing the deviation between the patient's *rule surface* coefficients and the $\overline{rule}$ values.

divided into a 70%/30% training/testing split). Table 3.6 also provides the expression for each rule.

Using a patient that was classified by each *strong* rule, we execute our visualization algorithm and create a set of surfaces. These surfaces are shown in Fig. 3.7. Figure 3.7 (a) shows a Keratoconic patient classified by Rule 1. Figure 3.7 (b) gives an example of a Normal cornea classified under Rule 8. Finally, Fig. 3.7 (c) illustrates a LASIK eye falling under Rule 4.

Since these are the strongest, or most discriminating rules for each class, we would expect that the *rule surfaces* would exhibit surface features commonly associated with corneas of that type. Our results are in agreement with expectations of domain experts. For instance, corneas that are diagnosed as keratoconic often have a cone shape protruding from the surface. This aberration, referred to as "coma" is modeled by terms Z_7 and Z_8 (vertical and horizontal coma, respectively). As a result, we would expect one of these terms to play some role in any keratoconus diagnosis. Rule 1, the strongest keratoconus rule, is characterized by the presence of a large Z_7 value. The *rule surfaces* (rows 3 and 4) of Fig. 3.7 (a) clearly show such a feature. Corneas that have undergone LASIK surgery are characterized by a generally flat, plateau-like center. They often have a very negative spherical aberration mode (Z_{12}). A negative Z_{12} value is part of Rule 4, the strongest LASIK rule. The elevated ring structure shown in the *rule surfaces* (rows 3 and 4) of Fig. 3.7 (c), is an example of that mode. These rule surfaces look very similar to the false color maps of the patients show in Figure 3.1.

3.7.3 Discussion

In order to address malpractice concerns and be trusted by clinicians, a classification system should consistently provide high accuracy (at least 90%, preferably higher). When coupled with meta-learning techniques, decision trees can provide such accuracy. The motivation behind the application presented here was to provide an additional tool that can be used by clinicians to aid in patient diagnosis and decision support. By taking advantage of the unique properties of Zernike polynomials, we are able to create several decision surfaces which reflect the geometric features that discriminate between normal, keratoconic and post-LASIK corneas. We have shown our results to several practicing clinicians who indicated that the important corneal features found in patients with keratoconus were indeed being used in classification and can be seen in the decision surfaces. This creates a degree of trust between the clinician and the system. In addition, the clinicians stated a preference for this technique over several start-of-the-art systems available to them. Most current classification systems diagnosis a patient as either having keratoconus or not. An off-center post-LASIK cornea can present itself as one that has keratoconus and vice versa. Misclassifying the post-LASIK cornea is bad, but misclassifying a keratoconic cornea as normal (in a binary classifier) can be catastrophic. Performing LASIK surgery on a cornea suffering from keratoconus can lead to corneal transplant. Our method can be leveraged by clinicians to serve as a safety check, limiting the liability that results from relying on an automated decision from an expert system.

3.8 Alternate Representations of Corneal Topography

Here we describe several methods of representing the Topography dataset using wavelets and Legendre polynomials. Our intent is to use these models as feature vectors for classification. As a result, our concern about the faithfulness of the model to the original data is less than it was with the Zernike-based methods. Due to certain intrinsic properties of the Zernike transformation, it can be used to visualize and highlight certain anatomical properties of the data. The wavelet transformation is domain-independent and we do not plan to use the resulting models for any kind of anatomical analysis. Therefore, here we are only concerned with the how the models perform during the knowledge discovery process.

3.8.1 Modeling Methods

The wavelet decomposition allows us to treat the input as either a 1D or 2D signal. We begin our discussion by focusing on variations that transform the data into a a 1D signal and follow with those that rely on a 2D input.

1D Wavelets

In the topography dataset, every object is represented as a series of concentric rings, where each ring consists of 256 points, and each point on the ring has an associated angle, radius and elevation. The simplest way to convert this polar-based data into a 1D signal is to "unwrap" the each ring, starting with the inner-most ring and moving outward to the next largest. When a ring is unwrapped, one moves in a given direction (for instance, clockwise) and traces along, concatenating each point to the end of the signal. With this approach, only the elevation values are used. Information about the angle and radius is ignored. The 1D wavelet decomposition is then applied to this signal, using the final level of approximation coefficients as the data model.

One drawback to the above approach is that the smaller rings tend to be more heavily sampled. Since each ring is composed of 256 points regardless of the radius, the larger rings will be more sparse. In an attempt to counter this issue, Smolek and Kylce developed a method that involves sampling each ring at equal angles [33]. They took the first 25 rings and sampled at equi-angle positions along each one until they had sampled a total of 1024 points. They used least-squares interpolation between adjacent points to generate the elevation values. This signal was decomposed using the s8 Symmlet filter (we typically use the Haar). After decomposing the signal, they took the 6th level approximation coefficients (16 total), the 4th, 5th and 6th level detail coefficients (64, 32, and 16, respectively) and used these 128 coefficients as a feature vector. The stated reason for using these values was that the authors wanted to focus on the large-scale coefficients to eliminate the potential for a ring-shift sampling artifact and to avoid long classifier training times. They divided the dataset into 50% training and 50% testing and used these vectors to train a neural network. They report accuracy, specificity and sensitivity values of greater than 99%, with only 1 misclassification. We test their algorithm in the next chapter to see whether we achieve the same classification performance on our own data.

One problem with both of the methods described above is that any spatial locality that exists between the rings tends to get lost in the concatenation process. Adjacent points within a ring remain together, but neighboring points between rings are widely separated. In order to retain some of this spatial locality, space-filling curves can be used to sample the data [57]. There are a number of different space-filling curves, but we focus on the the Hilbert curve. Hilbert curves can be used to generate a 1D signal that visits every point in a 2D space. Figure 3.8 shows the Hilbert curves of first, second and third order. These curves can be used to sample a matrix of size 4, 16, and 81, respectively. Higher-order curves can be used to sample larger input matrices, or the input data can be scaled to fit accordingly.

We utilize the Hilbert curve by converting the original polar data into Cartesian coordinates and interpolating those values into a 2D matrix. We create a Hilbert curve to fit this matrix and trace along the curve to generate our 1D signal. As with the ring-based method, we apply a 1D wavelet decomposition and use the final level approximation coefficients as our data model.

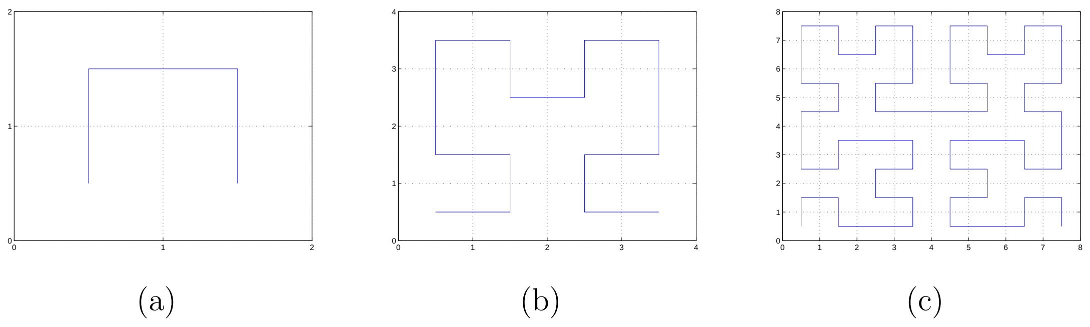


Figure 3.8: Graphical representation of First (a), Second (b), and Third-Order Hilbert Curves (c). Curves used to sample matrices of size 4, 16, and 81, respectively.

2D Wavelets

We also look to create a wavelet-based data model using a 2D input signal. As we did when generating the Hilbert curve, we convert the polar data to Cartesian coordinates and interpolate those values into a 2D matrix. We apply a 2D wavelet decomposition to these matrices and use the final level of approximation coefficients as our model.

Legendre Polynomials

As one final way to represent the topography data, we examine the use of Legendre polynomials as a modeling method. To create this model, we convert each object into a 1D signal by using the ring-based unwrapping technique described in the previous section. For a given order, we then calculate the Legendre polynomial for each point. We compute the least-squares fit between the polynomial and the original data and use the resulting coefficients as a feature vector.

3.8.2 Classification Performance

Using the above methods, we generate a number of different variations and test their performance with a Boosted C4.5 decision tree trained using 10-fold cross-validation. In the following list, we describe each of the representations tested and the modeling method used to create them:

1. **Legendre:** We generate representations that range in polynomial order from 5 to 10.

2. **2D Wavelets:** We convert the polar topography data to Cartesian coordinates and interpolate the converted data into a 2D matrix. We generate matrices of size *64x64* and *128x128*. We apply a 2D wavelet decomposition to these matrices and create feature vectors that contain the 3rd, 4th or 5th order approximation coefficients.

3. **1D Wavelets:** We create a number of different representations using the 1D wavelet decomposition. Each representation requires that we generate a different 1D signal before applying the actual decomposition.

   In the simplest approach, we start with the smallest inner ring and "unwrap' the signal, moving outward to each consecutive ring. Given this ring-based signal, we apply the 1D decomposition and generate feature vectors that contain the 5th, 6th, 7th, or 8th order approximation coefficients.

   We also test whether a Hilbert-based sampling method will affect the overall results. To generate such a signal, we convert the data into a 2D matrix as described above. We create a Hilbert curve to fit this matrix and trace along the curve to generate our 1D signal. We generate Hilbert curves for both the

64x64 and *128x128* matrices and create feature vectors containing 5th, 6th, 7th or 8th order approximation coefficients.

Finally, we create 1D representation based on the methods described by Smolek and Klyce in their topography work [33].

Results

After running our classification experiments, we found that, despite the number of representations tested, none of them were able to outperform the Zernike-based models. Classification accuracy using those representations was in the high-80% to low 90% range. The values with the alternative models were much lower. Therefore, rather than report each individual result, we just focus on the general trends.

The models based on the Legendre polynomials fared the worst in our tests, yielding accuracies in the mid to high-60% range. For the 2D wavelet representations, the 3rd order transformation performed the best, roughly around 80%. This was true for both matrices. For the 1D wavelet models, there was little difference between the ring and Hilbert-based models. The accuracy for each method was almost identical, around 85%. The representation created using the method of Smolek and Klyce performed slightly worse, around 83%. Thus, as we see from these results, of the methods tested, a Zernike-based approach appears to be the most effective way to represent the data, at least for the task of classification.

3.9 Conclusions

In Figure 3.9, we illustrate how various aspects of this case study implement the stages of our proposed workflow. We attempt to model the topography data using Zernike, pseudo-Zernike and Legendre polynomials, as well as 1D and 2D wavelets.

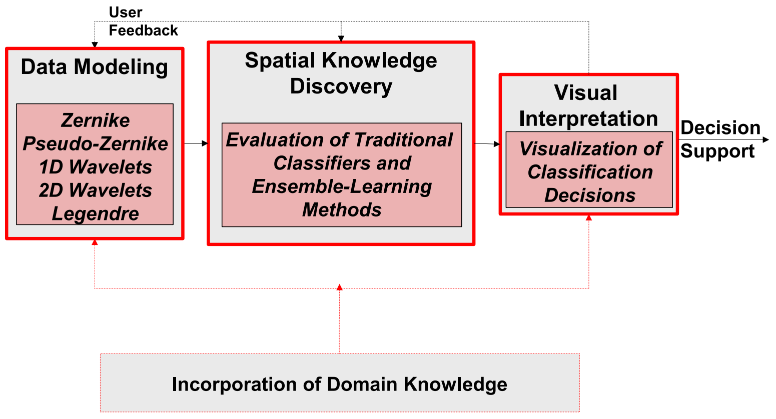

User Feedback
Data Modeling
Zernike
Pseudo-Zernike
1D Wavelets
2D Wavelets
Legendre
Spatial Knowledge Discovery
Evaluation of Traditional Classifiers and Ensemble-Learning Methods
Visual Interpretation
Visualization of Classification Decisions
Decision Support
Incorporation of Domain Knowledge

Figure 3.9: Workflow as implemented in first case study.

We test the effectiveness of these representations and also evaluate the performance of a number of classification strategies. Finally, we create a method to highlight and visualize those features that are important for distinguishing between the various patient groups. Since several of the lower-order Zernike terms directly correspond to anatomical surface features of the cornea, this technique indirectly incorporates domain knowledge into the visualization process. In addition, many of the methods developed in this chapter serve as the foundation of our Spatial Averaging algorithm, which we cover in Chapter 7.

CHAPTER 4

CASE STUDY II: CLASSIFICATION OF PROTEINS

We now introduce some of our early work in the protein domain, which focuses mainly on the classification of protein structure. The structural classification of protein molecules is an underlying component of many applications in computational biology, including drug discovery and function prediction. The structure of a protein molecule is known to influence its function, so if one can determine that protein belongs to a particular structural class, it might be possible to gain insight into its biological role. Our efforts in structural classification are divided into two major areas. The first involves a multi-stage classification strategy that leverages the hierarchical nature of protein structure taxonomies to improve accuracy [58]. We demonstrate the effectiveness of this approach using a dataset and representation that are well-known in the literature. We then describe two methods of modeling protein structure and illustrate their effectiveness using our multi-stage classification strategy [59]. These representations are incorporated into our later work in the protein retrieval, which is described in the following chapter.

4.1 Biological Background

Proteins are comprised of a varying sequence of 20 different amino acids. Individual amino acids are referred to as residues. Each type of amino acid is composed of a number of different atoms, but every single one contains a central Carbon atom that is called the C_α atom. The position of this atom is often used as an abstraction for the position of the entire residue. The sequential connection of the C_α atoms is called the protein backbone. Amino acids combine to form interacting subunits called secondary structures. These secondary structures interact to form the global, tertiary structure of the protein. Two of the most common secondary structures are α-helices and β-sheets. Residues that belong to these secondary structures are said to have a secondary structure assignment. There are a number of different labeling systems used to assign residues, but in this work, we use the simplest system: a residue can either be part of an α-helix, a member of a β-sheet, or have no assignment. All of the 3D information and secondary structure assignments for a protein can be obtained from the PDB.

There are several public databases that provide information on protein structure. Two of the most popular are the Protein Structure Classification database (CATH) [60] and the Structure Classification of Proteins database (SCOP) [9]. In our experiments, we use the classifications from SCOP database as our ground truth. The SCOP database arranges proteins into several hierarchical levels. The first four are *Class*, *Fold*, *SuperFamily* and *Family*. Proteins in the same Class share similar secondary structure information, while proteins within the same Fold have similar secondary structures that are arranged in the same topological configuration. Proteins in the same SuperFamily show clear structural homology and proteins belonging to the same Family exhibit a great deal of sequence similarity and are thought to be evolutionarily related.

4.2 Dataset

The dataset used in our sequence-based experiments is based on the dataset initially described in the work by Ding and Dubchak [1]. There, a training set was taken from the 27 most populated SCOP folds of the PDB_Select set, in which no two proteins share more than 35% sequence identity for aligned subsequences longer than 80 residues. This training set contained 311 proteins. Ding and Dubchak used an independent test set derived from the PDB_40D set, which consists of all SCOP sequences having less than 40% sequence identity with each other. Using the same 27 SCOP folds, 385 proteins were selected, and any PDB_40D protein having more than 35% sequence identity with the proteins in the training set was excluded. When combined together, the training and test sets yield a total of 696 proteins.

Since the publication of the work by Ding and Dubchak [1], the protein identifiers used in the SCOP database have changed. We were looking to use this dataset for a different set of experiments and needed to match the original identifiers to those currently used by SCOP. Using the latest labels from SCOP (version 1.67), a Perl script was written to automate the matching process, but we were unable to find a match for every protein. Rather than attempt to manually match the proteins that remained, we simply removed them from consideration, leaving a total of 653 proteins. We call this set the *Ding* dataset.

We represent each protein in our dataset using the same sequence properties as those listed in previous experiments [1, 61–63]. The feature vectors characterize the

Symbol	Property	Dim.
c	Amino Acid Composition	20
h	Hydrophobicity	21
p	Polarity	21
s	Predicted Secondary Structure	21
v	van der Waals Volume	21
z	Polarizability	21

Table 4.1: Sequence-based Properties used as feature vectors for classification.

following properties for each protein (the symbol for each descriptor given in parentheses): amino acid composition (c), hydrophobicity (h), polarity (p), predicted secondary structure (s), van der Waals volume (v), and polarizability (z). For a more detailed discussion on the derivation of these properties, the reader is referred to the work by Dubchak et al. [64] or Ding and Dubchak [1]. The feature vector for the amino acid composition consists of 20 dimensions. All of the rest have 21. These properties are presented in Table 4.1.

We test each descriptor individually and also combine them to create longer feature vectors, concatenating the feature vector of one descriptor onto the end of another. The combined datasets are referenced by their combined symbols (cs, csh, etc.). These combined vectors ranged in size from 41 dimensions (cs) to 125 (cshpvz).

4.3 A Multi-Stage Approach to SCOP Fold Recognition

Proteins are an important functional unit in countless cellular and biological processes. Rather than play a singular role, many proteins participate as a member of a group, or complex. High-yield genomic and proteomic experiments have led to the identification of large numbers of protein complexes. While the role of an individual protein within a given complex has been determined for a few molecules, for the vast majority, that role remains a mystery. It is believed, however, that a protein's function is strongly influenced by its structure. Therefore, it is reasonable to believe that proteins with a similar structure might have a similar function. As such, grouping molecules based on their structure will likely help predict a protein's function.

One way of determining similarity is through alignment. A number of different alignment methods exist, but they generally fall into two different categories: sequence-based or structure-based. While each are effective in certain cases, there are drawbacks to both approaches. Sequence-based alignment will fail when two proteins are structurally-similar but share little in the way of sequence homology. On the other hand, structure-based methods rely on data derived from a solved structure. Unfortunately, the number of proteins whose structure has been solved is much smaller than the number of proteins that have been sequenced. As of April 2005, the number of solved proteins in the PDB stands at just over 30,000, while there are more than 2.1 million sequenced proteins in the PIR-NREF database.

Thus, to be truly effective, any method of comparing proteins should not be limited only to molecules with a solved structure. The exact nature of the relationship between a protein's sequence and its structure remains one of the open challenges in computational biology.

While we are primarily concerned with the modeling and analysis of structure-based data in this work, we have proposed certain pattern discovery techniques that were validated on sequence-based protein data. Therefore, we detail some of the related work in sequence-based protein classification so we can later draw comparisons between our method and existing techniques.

One approach to the classification of protein structure is to examine the sequence-based properties of proteins whose structure is known. Using these proteins to construct and validate a classifier, one can use the resulting model on unclassified proteins to assign a structure-based label. Such a strategy has been employed by a number of researchers [1, 61–63]. Given a set of proteins from the Structural Classification of Proteins (SCOP) database, each classifier attempts to correctly predict a protein's *fold*.

Most of the recent work on this problem uses machine-learning techniques such as Support Vector Machines (SVMs) or Neural Networks (NNs). These methods are highly-tunable and have been shown to provide excellent performance in certain applications. A drawback to both of these techniques is that they are most effective when dealing with a binary, or two-class decision problem. In order to handle a multi-class (or in this case, multi-fold) dataset, a variation of one-vs-others (OvO) or all-vs-all (AvA) classification is often chosen.

With an one-vs-others approach, a classifier is constructed to decide between two classes: the class in question (the "true" class) and the rest (the "others"). Given k classes, k different classifiers are constructed and an input protein is assigned the label of whatever classifier returns a yes vote. In the case of a multiple yes votes, a number of different tie-breaking solutions have been proposed. For example, Ding and Dubchak describe a "unique" one-vs-others (uOvO) strategy that uses a series of 2-way classifiers to decide amongst those classes with a yes vote [1]. With OvO classification, the number of objects in the true class is often very small compared to the number of others. In a dataset with k different classes and an equal number of objects per class, this results in a classifier that tries to distinguish just $1/k$ of the objects from the rest. Given an unequal number of objects per class, this number can be even lower. As a result, an individual classifier can have high accuracy even if it misclassifies everything in the true class.

Using an all-vs-all strategy and a dataset with k classes, a classifier is constructed for every possible pair of classes, resulting in $k(k-1)/2$ different classifiers. Given an input object, it is tested with every classifier, and the class returning the largest number of "yes" classifications is assigned to the object. The drawback here is that AvA requires the construction of a large number of classifiers, while at the same time using a smaller number of datapoints in the construction of those classifiers, which can lead to over-fitting. As a result, care must be taken in order for either of these approaches to be effective.

Ding and Dubchak were one of the earliest groups to report on the problem of fold recognition, comparing the classification accuracy of a neural network and a support vector machine (testing the effectiveness of both OvO and AvA classification) on a set of proteins taken from the 27 most-populated SCOP folds, with less than 25% sequence identity between every protein in the set [1]. They were able to recognize the correct fold with an accuracy of approximately 56% using a number of sequence-based properties as feature vectors for their classifier. Using a slightly smaller dataset and many of the same input features, Tan et al. developed a rule-based classifier that combines the best classifiers from the OvO and AvA methods to generate a single classifier for each of the possible folds. Their best results improve fold recognition to roughly 60% [61].

Classifying the same dataset and input features, but this time employing a Bayesian Network-based approach, Chinnasamy et al. improve on the average fold recognition results reported by Ding and Dubchak, but fail to increase accuracy above 60% [62]. Finally, again starting with the same dataset and input features, Shi et al. employ an evolutionary algorithm to select the most relevant features and classify using a SVM, but their average accuracy remains around 56% for the 27-fold problem [63].

As an alternative to the above techniques, we propose a tiered, or multi-stage, approach to classification. Just as proteins in the SCOP database are grouped in a hierarchical fashion, it is possible to first partition a dataset based on certain high-level similarities. For each partition, one can then create a more specific, fine-grained classifier. To our knowledge, it is the first time such a technique has been applied to this particular problem. Here we report our method and results using a two-level classification strategy.

In the first stage, we classify proteins based on their SCOP class. For each class of proteins, we then classify by fold. Our approach is flexible, allowing for any classification strategy to be used at any stage. We test our method with a Naïve Bayes classifier and several decision tree-based meta-learning strategies. Using the Ding dataset, we achieve an improvement in accuracy of 15-20% compared with previously published results, predicting the correct fold with an accuracy up to 74%. We only need to create a single classifier for the Class level and then one for each Fold, for a total of five classifiers. As opposed to previous methods, our model is fast, scalable, and can easily be recreated if there is a large change in the dataset or input features.

4.3.1 Multi-Level Classification

Databases like SCOP group proteins in a hierarchical, tree-like fashion based on shared structural characteristics. The top of the hierarchy is referred to as the Class level. The classes are divided into Folds, which are further divided into *SuperFamilies.* Each SuperFamily is itself composed of several individual *Families.* Since proteins at each level are grouped based on shared characteristics, we would expect a protein to have a higher structural similarity to those proteins that are within the same class (intra-class) than those that are without (inter-class). The same would hold true at the fold level and on down through the hierarchy. For this reason, we believe a classification scheme should take advantage of such a natural hierarchy. Most of the existing work on this problem have taken a "flat" view toward classification, focusing only on the fold level, totally ignoring any class-level information.

The only reason it is currently possible to classify proteins at the fold level with a binary classifier is that there are few folds with the requisite number of members needed to effectively train a classifier. The current version of the SCOP database (May 2005) lists 887 possible folds. As the number of solved protein structures increases, so will the number of folds that can be classified. When this occurs, the use of binary classification strategies will become less and less practical. When presented with 887 potential folds, an OvO strategy will require the training of 887 2-way classifiers. An AvA method, on the other hand, will require the construction of a total of 392,941 different classifiers. The use of cross-validation will increase that number even more. With n-fold cross validation, $1/n$ of the dataset is set aside as a testing set (a *fold*) and a classifier is trained on the remaining members. This process is repeated n times with a different $1/n$ of the dataset set aside in each iteration (an object can only be a member of one testing fold). Therefore, when using n-fold cross-validation and an OvO classification strategy on k possible classes, a total of kn classifiers must be trained, while AvA increases that number to $nk(k-1)/2$. Finally, if one wishes to classify proteins further down the hierarchy, where the number of potential choices is even larger, the problem quickly becomes intractable.

As stated previously, classification techniques that can effectively handle multi-class data, such as Bayesian or decision tree-based methods, have a difficult time distinguishing between a large number of possible classes. One way to try and reduce the potential confusion is to pre-process or pre-partition the data before classification occurs. By using a *multi-tiered* or multi-level classification strategy, one can cut down on the number of potential outcomes, which is useful when faced with noisy, real-world data that is not clearly separable. The multi-tier strategy we present here is meant to be general. It can be applied to any domain where the data falls into a natural hierarchy (or one where such a hierarchy can be readily deduced). In addition, any classification strategy can be used at the different levels of the hierarchy. If a certain method is found to be more effective at one stage, it can be used there and replaced with something else at the others.

Algorithm

We provide the implementation details of our multi-level strategy in Section 4.3.2. The general idea, however, is to first classify proteins at the class level, grouping them based on global, high-level features. We refer to this stage as *Classification by SCOP Class.* Once this partitioning is complete, we subdivide further, classifying each protein by fold (denoted *Class-specific Fold Classification*). The intent of this step is to improve accuracy by using a increasingly fine-grained classification model, separating the data based on more local, fold-specific attributes than a typical decision tree that is forced to distinguish between all possible classes (which we call *All-Folds Classification*). With this approach, the classifier tends to employ two different sets of features in classification. As we show below, at the Class level, the focus is on those elements that can partition the dataset on global, high-level features. At the Fold level, more emphasis is placed on the features that allow a distinction based on more local, specific attributes.

Validation

The intuition behind our classification strategy is based on several well-known and empirically-validated assumptions [65, 66], which are listed below:

- Multi-stage classifiers may be more accurate than single-stage classifiers.
- Smaller decision trees have higher accuracy than larger trees.
- Locally optimizing information tends to produce small, shallow, accurate trees.

Of particular importance are the last two assumptions. By dividing the overall task of global fold recognition into the smaller sub-tasks of recognizing class and class-specific folds, we hope to produce smaller, more accurate decision trees that avoid the problem of getting stuck in a local minima. Large decision trees often suffer from *noise*, *fragmentation* and *subtree replication* [66, 67]. Noise can cause irrelevant features to be used as selection criteria, which in turn can lead to overfitting. A similar effect can arise due to tree fragmentation, where there are a large number of leaf nodes that only represent a few objects. Finally, a large decision tree can suffer from subtree replication, where a certain piece of the tree is repeated throughout the overall structure, which can also cause fragmentation. We now provide results that motivate the development of our multi-tier strategy and empirically validate our approach.

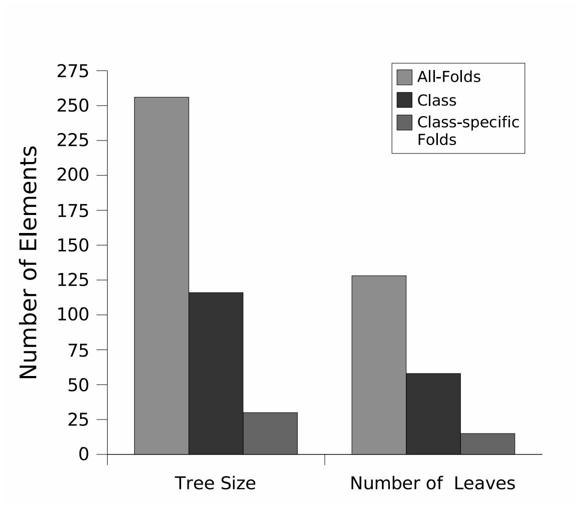


Figure 4.1: Average decision tree size and average number of leaves for each of the classification variations tested (Class, All-Folds, Class-specific Folds). Results reflect a boosted C4.5 decision tree.

In Figure 4.1 we provide the average decision tree size and average number of leaf nodes for the different single-stage classification strategies mentioned above: Class, Class-specific Fold and All-Folds. These values were obtained when running the boosted C4.5 experiments discussed in Section 4.3.2 on one of the single property datasets. While the actual numbers may differ depending on the property chosen, the overall trends and differences between classification strategies remain.

Figure 4.1 clearly shows the drawback to using a single-stage classifier on a large multi-class decision problem. The size of the average tree for All-Folds classification is greater than 250 nodes, and it contains over 125 leaves. With a dataset of approximately 650 proteins (the size of our set), fragmentation becomes a real concern. In contrast, in Figure 4.1 one can see that using Class or Class-specific Fold classification results in a significantly smaller decision tree and far fewer leaf nodes. The Class

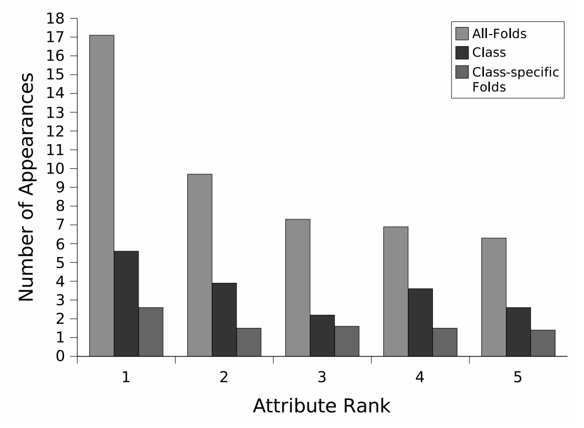


Figure 4.2: Average number of appearances per tree for each of the top five splitting criteria in the All-Folds method.

tree is less than half the size of the All-Folds tree, and the Class-specific Fold tree is roughly an eighth the size of the original.

The problem of subtree replication is illustrated in Figure 4.2. Using one of the single-property datasets as an example, we list the top five decision tree attributes in terms of average number of occurrences per tree (results again from the boosted C4.5 experiments). Shown are the top five attributes for the All-Folds method and the corresponding counts of those attributes using the other methods. The top attribute appears an average of 17 times in each tree when classifying using All-Folds. That same attribute occurs less than six times when classifying by SCOP Class and fewer than three times in a Class-specific-Fold tree. In general, the All-Folds attributes appear 2-3 times more often than in the Class trees and 4-6 times more often than the Class-specific Fold trees. It should be noted that the top attributes in the All-Folds trees are not necessarily the top attributes in the trees of the other methods (and vice versa).

Class	Class-Specific Fold	All-Folds
1	1	1
2	–	2
3	–	4
4	–	5
5	–	–

Table 4.2: Rank of the top 5 attributes of a *Class* decision tree in the *Class-specific Fold* and *All-Folds* methods. A "–" indicates a rank below 5.

In Table 4.2, using the same single-property dataset as in Figure 4.2, we list the top 5 attributes (in terms of average number of occurrences per tree) in the Class method. We also list the corresponding rank of those attributes in the Class-specific-Fold and All-Folds methods. A "–" indicates that the attribute is not among the top 5 attributes. As one can see in Table 4.2, there is a high degree of overlap between the Class and All-Folds methods and almost no overlap between the Class and Class-specific Fold trees. Similar results are seen with the other datasets. This gives credence to our belief that using a multi-level classification strategy allows a classifier to focus on different features at different levels: global features at the top level, local features at the lower level.

In light of these results, we feel that a multi-stage strategy like the one proposed here will eliminate many of the issues that arise from a large multi-class decision problem. Furthermore, by breaking the task of classification into a number of stages, we can more finely-tune our overall classifier, taking advantage of specific strengths that an individual classifier may have at a particular level, using a different method for each one.

4.3.2 Experiments

In order to show the effectiveness of our classification strategy, we need to run a number of different tests in order to establish a series of baseline results. As such, we conduct the following classification experiments using the Ding dataset:

1. **Classification by SCOP class.** In this test, each SCOP fold is replaced by its corresponding SCOP class label (α, β, $\alpha + \beta$, α/β). Table 4.3 lists the identification number of the SCOP folds contained in our dataset and their corresponding class. The purpose of this test is to establish that it is possible to improve accuracy by classifying proteins at an "upper," or more global level. This mode of classification is analogous to the "Class" method of the previous section.

2. **Classification by SCOP fold after partitioning by class.** We repeat the previous experiment, but rather than classify the entire dataset at once, we manually partition by class and then distinguish between folds. We hope that classification accuracy will improve after removing those proteins that fall into different areas of the structural hierarchy. This experiment is equivalent to Class-specific Fold classification.

3. **Classification by SCOP fold.** This experiment is an attempt to replicate the work reported elsewhere in the literature [1, 61–63]. Here, we create a classifier to distinguish among the 27 different folds in our dataset. Rather than employ a multi-classifier variation of OvO or AvA, we use a single classifier for this task, deciding between all possible classes at once. The results from this experiment are the same as the All-Folds results discussed in the previous section.

4. **Multi-Level Classification.** The final experiment is a test of our multi-tiered classification strategy. First, we classify each protein based on its SCOP class, using 10-fold cross-validation. Then, for each SCOP class, we take all of the proteins that were correctly classified and construct a classifier that distinguishes between the folds present in that class (see Table 4.3 for the relationship between class and fold).

   For each SCOP class, we take a number of random samples (twenty in our tests) of the data, dividing it into 70%/30% training/testing splits. While a protein can appear in more than one sample, we do not allow them to appear more than once within a given sample (i.e. they are randomly selected without replacement). Finally, in an attempt to ensure that every protein is adequately represented, we limit the number of samples in which a protein can be present to half the total number of samples.

   To compute the accuracy of our multi-stage method, we take the number of proteins that were misclassified at the SCOP class level and add to that the average number of misclassifications at the fold level. To compute the average, we sum the number of misclassifications for each random sample and divide by the total number of samples. By using the ideas behind Experiments 1 and 3, we hope to show a substantial improvement in classification accuracy over the results of Experiment 2.

All of our experiments were conducted on a PC with a 2.8 GHz Pentium 4 CPU and 1.5 GB RAM, running Debian Linux with a custom 2.6.9 kernel. The classification process was done using the WEKA data mining toolkit, version 3.4[4] and Sun Java

[4]http://www.cs.waikato.ac.nz/~ml/weka/

Class	Folds	Total
α	1, 2, 4, 7, 9, 11	6
β	20, 23, 26, 30, 31, 32, 33, 35, 39	9
α/β	46, 47, 48, 51, 54, 57, 59, 62, 69	9
$\alpha+\beta$	72, 87, 110	3

Table 4.3: SCOP Classes and their corresponding Folds

1.4.2. Classification accuracy is given as the True Positive Rate (TPR), or the number of correct classifications divided by the total. We test each experiment using a number of different classifiers, including Naïve Bayes, Random Forests, C4.5 with Bootstrap Aggregation (bagging) and C4.5 with Adaptive Boosting (boosting) [26,29,38,39,55]. We left all of the classifiers on their default setting, except for the C4.5 algorithm, which we changed to use binary splits. All of the proteins were combined into one large dataset and classified using 10-fold cross-validation.

4.3.3 Results

We now present the results of our classification experiments. We found that the classification accuracy of Random Forests and Bagged C4.5 were very close to that of Boosted C4.5, though when there was a difference, Boosted C4.5 tended to provide the highest accuracy. Thus, in order to improve clarity, we omit the results from Random Forests and Bagged C4.5.

Classification by SCOP class

In Figure 4.3, we provide the results of our experiments in trying to recognize the correct SCOP class. As we show next, the accuracy when classifying by SCOP class is higher than the accuracy when classifying by fold. This is due to the fact that we

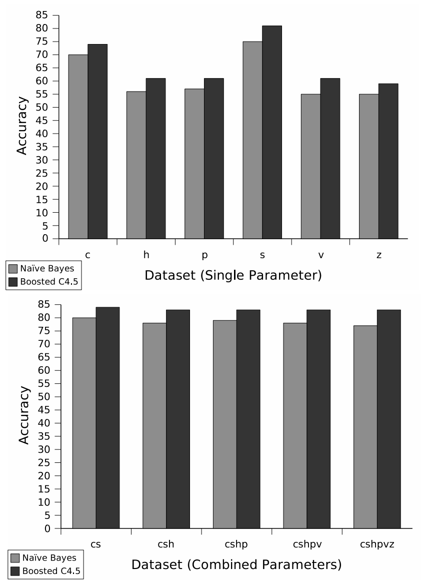


Figure 4.3: Accuracy when classifying by SCOP class.

are dealing with 4 classes, as opposed to 27 folds. For most of the single parameter datasets, the accuracy hovers around 55% when using the Naïve Bayes classifier and approximately 60% when using a Boosted C4.5 decision tree. The *c* and *s* datasets perform much better than the others, with accuracy above 70% for both classifiers. Performance can be improved further by combining the datasets into larger feature vectors. Figure 4.3, shows that combining the feature vectors increases accuracy to 80% with the Naïve Bayes classifier and to 84% with Boosted C4.5. This is 3 to 5% better than the best single parameter dataset for the Boosted and Bayes classifiers, respectively.

Classification by SCOP fold after partitioning by class

We provide the accuracy of trying to recognize the correct SCOP fold when the dataset has already been partitioned by class. In theory, this should be an easier problem, as there are fewer folds to choose from and the dataset for each class is smaller than the original. As one can see in Figure 4.4, the accuracy for most of the class-specific datasets is higher than that of the multi-class (listed under the *All Classes* column). There are a few instances where this does not hold, most notably for the Naïve Bayes classifier and the β class. Overall, however, classification accuracy is substantially improved. Since it is unlikely that the user would ever be presented with a dataset that is already partially classified, these results may not seem to be very useful. However, they are presented here to give some intuition as to why a multi-level classification strategy might be more effective than a single-stage strategy when dealing with a large number of possible folds.

Classification by SCOP fold

In Ding and Dubchak [1], the highest accuracy for the 27-fold problem is obtained using an all-vs-all support vector machine. This requires the construction of 351 different classifiers. The best classification accuracy with the AvA SVM was around 56% using the *csh* and *cshp* datasets. The results in Figure 4.5 show that we are able to achieve the same classification accuracy with a single classifier.

Our accuracy on the single parameter datasets are much higher than the results of Ding and Dubchak [1]. There is only one case where the Naïve Bayes classifier does not outperform the SVM and the Boosted C4.5 classifier always outperforms the SVM by at least 10%. These results are also higher than those reported elsewhere [63].

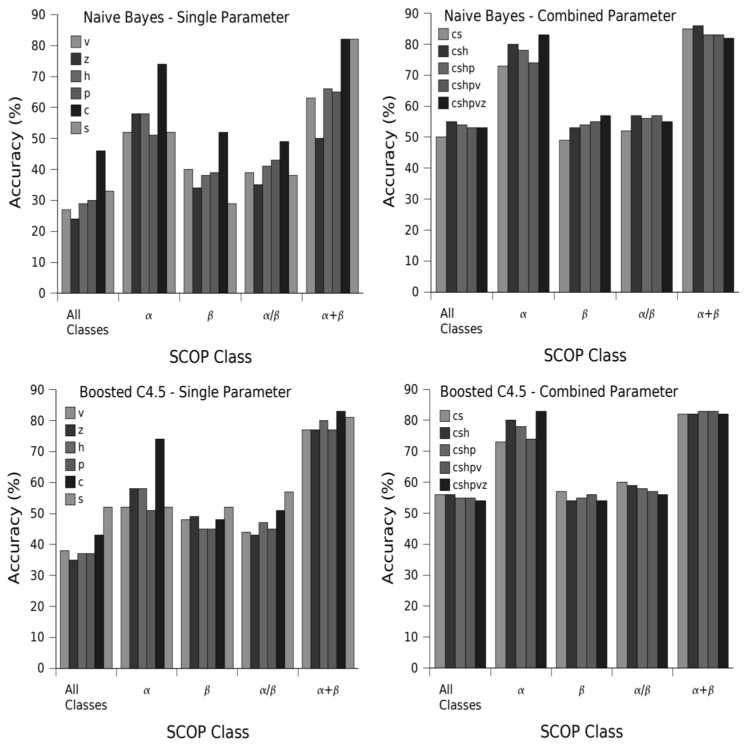


Figure 4.4: Accuracy for SCOP Fold recognition by Class. Top figure represents Naïve Bayes results, the bottom, Boosted C4.5. The left graphs contain the single parameter datasets and the combined datasets are on the right.

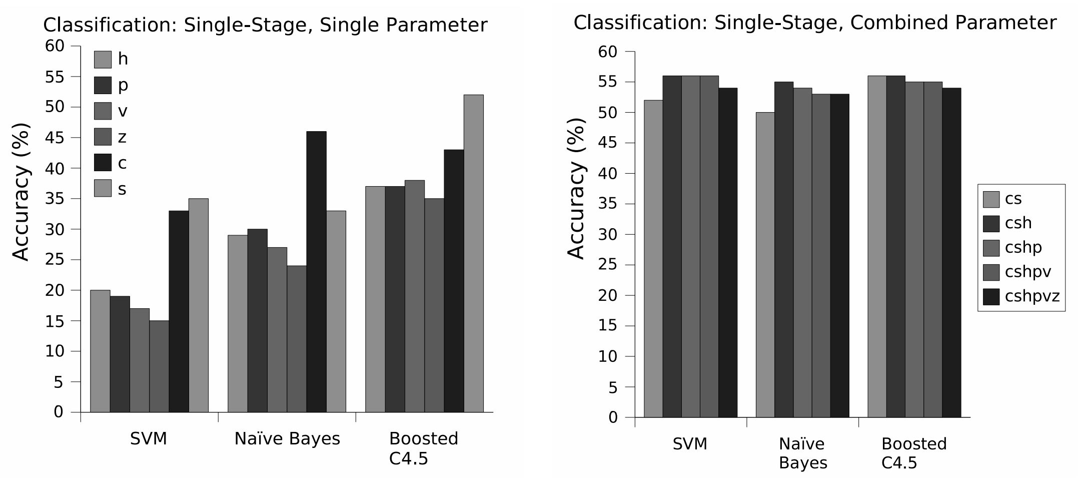


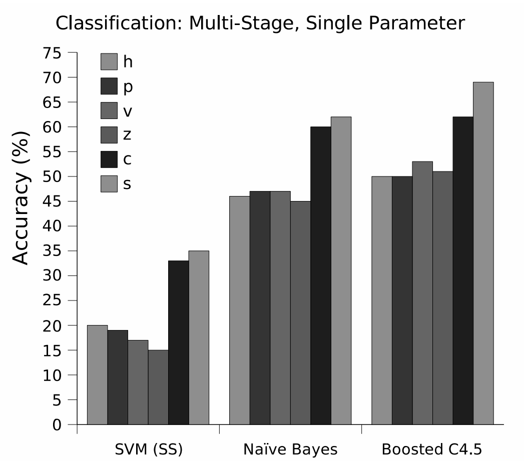
Figure 4.5: Accuracy for SCOP Fold recognition when using a Single-Stage Naïve Bayes or Boosted C4.5 classifier compared to previously-published SVM values [1]. The left graphs show the results using the single parameter datasets, while the combined datasets are shown on the right.

Multi-Level Classification

We present the results of our multi-level classification strategy in Figure 4.6. As one can see, these results are much higher than those of the single-stage solution (given in Figure 4.5). For the combined datasets, the multi-stage results are a full 12%-18% higher than the corresponding single-stage value. In addition, *our best Multi-Stage results are around 14% higher than any result we have seen in the literature on this problem.*

4.3.4 Discussion

Trying to predict the fold of a protein is extremely challenging. There are a large number of possible folds, and many folds contain a small number of members.



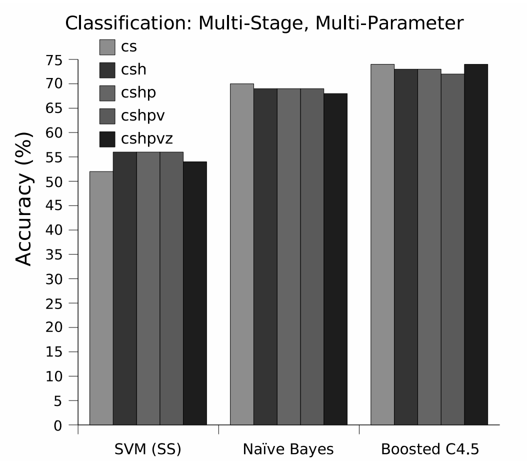


Figure 4.6: Accuracy for SCOP Fold recognition when using Naïve Bayes or Boosted C4.5 classifier with the Multi-Level classification strategy (compared to a single-stage SVM [1]). The left graphs show the results of the single parameter datasets while the combined values are on the right.

Most existing solutions to this problem rely on some type of all-vs-all or one-vs-others classification strategy. As stated previously, both of these strategies have their drawbacks. Our proposed method leverages the fact that folds that fall within the same SCOP class share similar structural properties. Therefore, *we can use high-level, global similarities to partition the data at the class level.* This partitioning simplifies the classification problem, both in the number of proteins that must be classified and in the number of potential folds. It is true that the proteins within each partition will be more structurally similar, which would seem to present a more difficult classification challenge. However, in a given partition, *we can use local, more discriminating features to distinguish between proteins and assign the correct fold.* As additional protein structures are crystallized and labeled, this strategy can also be used to help classify unknown proteins at the SuperFamily or Family level.

Our strategy can be used with any type of classifier, it need not be Naïve Bayes or Boosted C4.5. It is also possible to use different classifiers for each tier. If new data is acquired, it is relatively simple to regenerate the classification model. One can also use this strategy in a semi-supervised environment, using unlabeled data to either tune the classifier or to make predictions about the possible fold/function of an unknown protein. Also, if desired, one can create a classifier that is biased toward a particular class or fold. In this manner, one can more reliably identify proteins of that fold, with a possible detriment to the identification accuracy of the other folds.

4.4 Zernike and 1D Wavelet-Based Representations of Protein Structure

In this section, we present our first attempts at modeling the structure of proteins. We propose two dimensionality-reducing methods of decomposition that allow us to group proteins sharing certain common structural elements while at the same time distinguishing between those that are structurally dissimilar. Both methods use the distances between the C_α atoms of a protein as their basis. The first method maps the C_α distances to a two-dimensional matrix and creates a representational model of that matrix with Zernike polynomials. The second starts by treating the protein as a one-dimensional signal, using the distances between C_α atoms as the amplitude. We apply a wavelet transformation to the signal and use the corresponding coefficients as our representation.

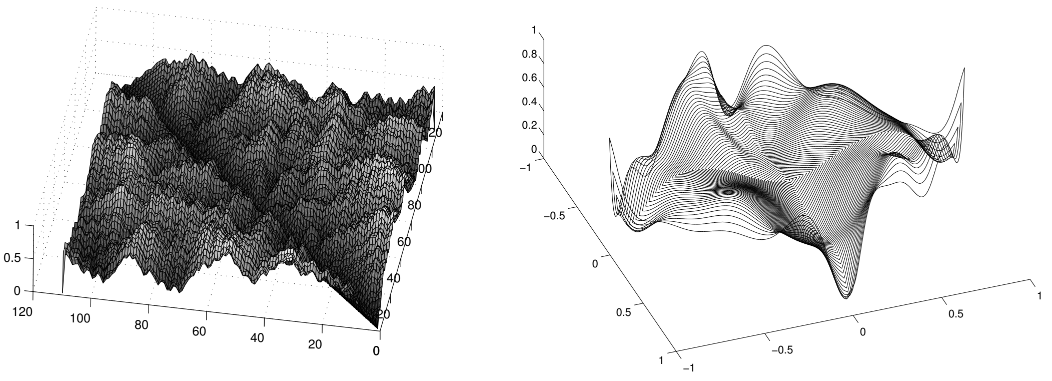

(a) (b)

Figure 4.7: Left: Example distance matrix for protein pdb1CCR (a). Elevation values have been normalized between 0 and 1, with lower values representing closer proximity between residues. Right: 10th order Zernike representation (b).

4.4.1 Algorithms

Here we detail the algorithms used to create our structural representations. We first describe our method of modeling a protein with Zernike polynomials. We follow with the wavelet-based approach.

Zernike-Based Approach

In addition to using the Zernike polynomials to model the surface of the cornea, we have also looked at using it to model the structure of a protein. Rather that model the structure directly, however, we convert the structure into a distance matrix and then apply the transformation. That conversion process is as follows: First, we use the 3D coordinates of the protein (obtained from the PDB) to calculate the distance between the C_α atom of each residue. These values are placed into an n x n matrix D, where n represents the number of residues in the protein and *D(i,j)* represents the distance between the C_α atoms of residues i and j. Once we have populated the matrix, we divide by the maximum distance to normalize the values between 0 and 1. Figure 4.7 (a) provides a graphical depiction of this matrix, with higher elevations (greater distance) having a lighter color. An illustration of a 10th order representation of the matrix shown on the left. As one can see, the Zernike representation adequately captures the global shape of the matrix, but does not do as well in representing local features. This realization led us to develop our wavelet-based method, described below.

1D Wavelet-Based Model

The use of wavelets is natural in applications that require a high-degree of compression without a corresponding loss of detail, or where the detection of subtle distortions and discontinuities is crucial. Their use in knowledge discovery applications is less cut and dry, however. Wavelets can be applied to signals of any length, and as such, will produce a varying number of coefficients. Deciding which coefficients to use in a feature vector becomes difficult. Does one keep the top k coefficients? How does one identify a "top" coefficient? What if a wavelet transformation yields less than k coefficients? Are we limited by the size of the smallest signal in the dataset? Does one consider the coefficients from every level, or just the last one? Should one ignore the detail coefficients and just consider the approximation coefficients, or vice versa? In order to make a true "apples-to-apples" comparison, feature i of object x should correspond to the same property in object y. Simply selecting the top k wavelet coefficients does not provide such a guarantee.

In the end, we elect to use the approximation coefficients of the last level of the wavelet decomposition as our feature vector. In order to guarantee that we are left with an equal number of coefficients per object, each input signal must be normalized to have the same length. With datasets such as our topography dataset, this process is relatively easy, since every object consists of approximately the same number of data points. Proteins, on the other hand, can be of vastly different lengths. However, since proteins belonging to the same SCOP fold have the same SSEs in the same topological order, intuitively this normalization makes sense. Before describing our method of converting a proteins into a signal, it should be noted that to normalize the lengths, we will have to downsample, discarding some information. In rare cases when given an extremely short protein, we may have to upsample in order to obtain an adequate number of datapoints.

The first step in the creation of our 1D wavelet-based feature vectors involves converting the protein structure into a distance matrix using the technique described in the previous section. Figure 4.8 (a) provides a graphical depiction of another matrix (for protein pdb153L), with higher elevations, or larger distances, having a lighter color.

In addition to representing a protein as a distance matrix, it is also possible to represent a structure as a “contact map,” which is simply a binary distance matrix, where a value *D(i,j)* is set to 1 if the distance between residues i and j is less than a certain threshold and 0 otherwise ($7\mathring{A}$ is a commonly used threshold value). In a contact map, secondary structures such as α-helices and parallel β-sheets emerge as thick bands along (or parallel to) the main diagonal, while anti-parallel β-sheets appear perpendicular to it. Figure 4.8 (b) provides the contact map version of the distance matrix shown in Figure 4.8 (a). Examples of the different secondary structures are evident in this figure.

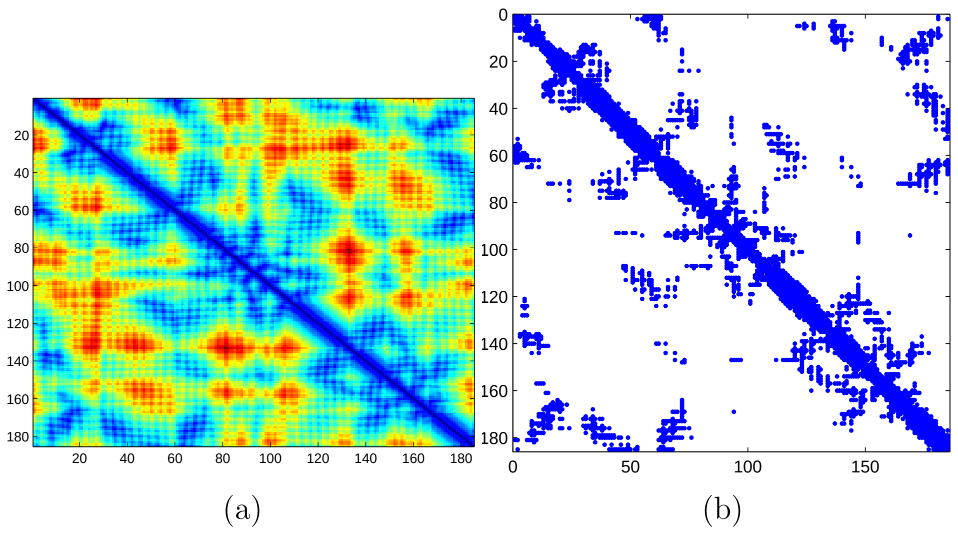


Figure 4.8: Left: Distance matrix for protein pdb153L. Larger distances a denoted with a lighter color. Right: Example contact map for protein pdb153L. Marked positions denote inter-residue distance of less than $7\AA$.

The interactions between secondary structures influence the global structure of the protein. Thus, we want our conversion process to capture as many of these interactions as possible. To that end, we implement two conversion techniques. The first is designed to capture the interactions and secondary structures that are *parallel* to the main diagonal, while the second focuses on the *anti-parallel.* To convert the matrix to a parallel signal, we start with position *D(0,1)* of the matrix and place it at position 0 of the signal. We then place *D(1,2)* at position 1 and proceed down the diagonal. *D(0,2)* will be mapped to position n, *D(1,3)* to n*+1*, and so on, until the matrix has been converted. In this manner, we first add the values closest to the diagonal and gradually move toward the corner. This process is illustrated in the upper right triangle of Figure 4.9.

The anti-parallel conversion process is shown in the lower left triangle of the matrix in Figure 4.9. We normally perform this operation on the upper triangle (as with the parallel process), but in order to match the figure, we use the lower triangle in our description. We start with position *D(1,0)* and place it at position 0 of the signal. We then place *D(2,0)* at position 1. We move right to *D(2,1)*, place it at position 3 and travel away from the diagonal, placing *D(3,0)* at position 4. We continue this stair-step approach, moving our starting point down the diagonal and adding points in a perpendicular fashion until we reach the edge of the matrix (the movement of the starting point is illustrated by the double arrow in Figure 4.9). Since there are *n-1* points that are a distance 1 from the diagonal, and *n-2* points that are distance 2, we must take a total of $2n - 3$ "steps" before reaching the bottom of the matrix.

In both conversions, if a residue is not part of a secondary structure, we simply skip over that position. The global structure of the protein is determined based on interactions between secondary structure elements. Therefore, we hypothesize that we will get better results by only looking at the residues involved in those structures, a belief that has been validated through empirical testing.

Implementation

All of the wavelet transformations were calculated using functions from the Matlab Wavelet Toolbox (Toolbox Version 3.0.1). We tested a number of different wavelet families, but finally settled on using a Haar wavelet, since it the simplest and easiest to compute, but still maintains the useful property of being an orthonormal basis function.

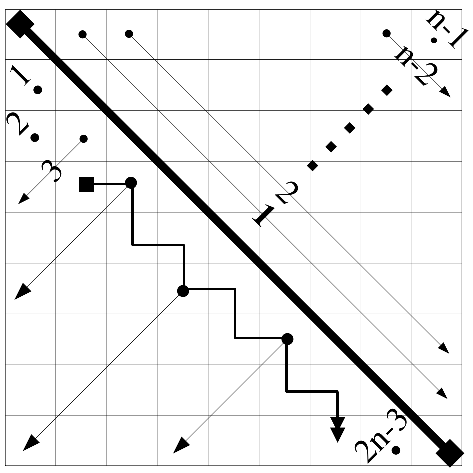


Figure 4.9: Two methods for converting a distance matrix to a one-dimensional signal. The parallel conversion process is shown in the upper triangle, the anti-parallel version in the bottom.

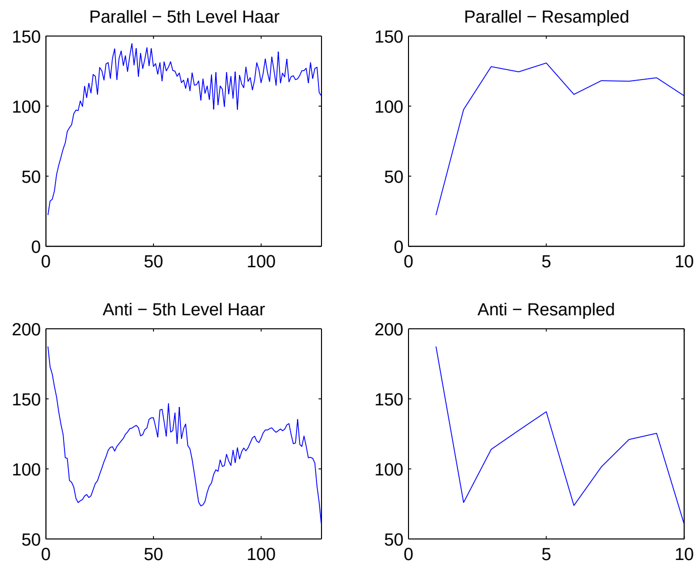


Figure 4.10: Parallel and Anti-Parallel windows (top and bottom, respectively). The figures on the left represent a 5th level Haar decomposition. The final reduced signals are shown on the right.

Both the 1D wavelet and Zernike transformations will result in a multi-resolution representation of the input data. The difference is that a Zernike polynomial provides a wavefront-based approach applied to the entire distance matrix, whereas the wavelet decomposition results in a high-level description created from localized summaries created after converting the 2D matrix to a 1D signal. The idea is that the wavefront-based approach may be able to capture more high-level trends corresponding to long-distance, tertiary relationships among structure elements, while wavelets may be more effective at characterizing fine-grained details that are lost in a global model. Further improvement may be achieved by using both representations in classification, therefore capturing both global and local trends. We explore this issue in our experiments.

4.4.2 Validation

In the text below, we show the effectiveness of our 1D wavelet and Zernike-based models of protein structure. In addition, we test our Multi-Stage classification strategy using a structure-based features. Before proceeding, however, we cover some related work in structure-based protein classification.

Most of the structure-based methods of protein analysis are concerned with the problem of pair-wise or multi-way alignment. These methods, such as DALI [68] and CE [69] (to name just two) can be considered as a type of classification, but they are primarily focused on providing a measure of similarity between small groups of proteins (usually pair-wise at first, then extended to multi-way).

A number of recent publications have attempted to create a true structure-based classifier. Some, such as SGM [70], use a nearest-neighbor classification strategy, where a protein is assigned the label of the cluster to whose mean representative it is most similar. Others use a more traditional classification strategy, creating a structure-based "fingerprint," that can be used as a feature vector in classification [71, 72]. Aung and Tan validate their technique on a dataset that is very similar, though less-challenging (fewer total folds), than the one used by Ding and Dubchak [1]. However, they identify the correct fold only 45% of the time [72].

4.4.3 Experiments

The following experiments were conducted using the Ding Dataset:

1. **Single Stage Classification.** This experiment is an attempt to replicate the earlier work on sequence-based classification [1, 61, 62]. Here, we create a classifier to distinguish among the 27 different folds in our dataset. Unlike previous studies, which typically employ an OvO or AvA strategy, we use a single classifier and attempt to distinguish between all classes at once.

2. **Multi-Stage Classification.** We test our representations using our recently proposed multi-stage classification strategy [58]. For clarity, the implementation details of that method are repeated here. First, we classify each protein based on its SCOP class, using 10-fold cross-validation. Then, for each SCOP class, we take all of the proteins that were correctly classified and construct a classifier that distinguishes between the folds present within that class.

   For each SCOP class, we take twenty random samples of the data, dividing it into 70%/30% training/testing splits. While a protein can appear in more than one sample, we do not allow them to appear more than once within a given sample (i.e. they are randomly selected without replacement). Finally, in an attempt to ensure that every protein is adequately represented, we limit the number of samples in which a protein can be present to half the total number of samples.

To compute the accuracy of our multi-stage method, we take the number of proteins that were misclassified at the SCOP class level and add to that the average number of misclassifications at the fold level. To compute the average, we sum the number of misclassifications for each random sample and divide by the total number of samples.

3. **Classification with Sequence-based Properties.** A number of previous studies have attempted to classify proteins using a series of sequence-based physio-chemical properties [1, 58, 61, 62]. They looked at classification accuracy using each individual property as a separate feature vector and also at combining them together to improve results. These studies found that using several properties at once does in fact improve classification accuracy. To build on those results, we test whether we can improve classification accuracy by extending our feature vectors with a dataset of these combined properties.

   The physio-chemical descriptors characterize the following properties (symbol and number of dimensions given in parentheses): amino acid composition (c, 20), predicted secondary structure (s, 21), and hydrophobicity (h, 21). For a more detailed discussion on the derivation of these properties, the reader is referred to the work by Ding and Dubchak. [1]. To create this combined dataset, we concatenate the attributes of one descriptor onto the end of another. We refer to this dataset by the symbols of the combined properties: csh.

Experimental Setup

All of our classification experiments were conducted on a PC with a 2.8 GHz Pentium 4 CPU and 1.5 GB RAM, running Debian Linux with a custom 2.6.9 kernel. The classification was done using the WEKA data mining toolkit, version 3.4 and Sun Java 1.4.2. Classification accuracy is given as the True Positive Rate (TPR), or the number of correct classifications divided by the total. We conduct our single-stage classification experiments using Naïve Bayes and C4.5 with Adaptive Boosting (boosting) [29,55]. We also use Boosted C4.5 as the base for our Multi-Stage classifier. For the single-stage tests, all of the proteins were combined into one large dataset and classified using 10-fold cross-validation.

4.4.4 Results

We now discuss the results of our classification experiments. We examined both the original wavelet coefficients and the denoised values, but found the denoised wavelet coefficients performed better than the original wavelets. As a result, we only present the results of that dataset, which we denote *wd*. In addition, while we tested the accuracy of a 4th through 10th order Zernike transformation, we found that the 5th order polynomial provided much better results than the others, so to improve the clarity, we only show those values (we refer to this dataset as $5z$). We also combined the wavelet and Zernike coefficients into a single dataset, which we label *wz*. Finally, we found that a Boosted C4.5 decision tree greatly outperformed the Naïve Bayes classifier, so we focus solely on that classifier.

Single-Stage Classification

The results of our single-stage classification experiments are presented in Figure 4.11. The values of the structure-based representations are given in the *Structure* columns while the results of the structure representation combined with the physio-chemical properties are given in the *Structure + csh* columns. As we can see when examining the pure structure results, the denoised wavelet dataset (*wd*) clearly outperforms the Zernike coefficients (*5z*), 57% to 45%. The Zernike coefficients provide roughly the same accuracy as the method proposed by Aung and Tan [72] and the denoised wavelet coefficients provide a slightly higher accuracy than the combined physio-chemical property dataset used by Ding and Dubchak [1]. We did not see any increase in accuracy from the combined wavelet-Zernike dataset. When we combine the structural datasets with the physio-chemical values, we can increase the accuracy of all datasets. We improve accuracy to 62% for the Zernike coefficients, 63% for the wavelet coefficients, and unlike the pure structure case, we can improve accuracy to 65% for the wavelet-Zernike dataset. All of these values are higher than any result previously seen on this dataset with purely structural or physio-chemical-based features.

Multi-Stage Classification

Figure 4.12 illustrates the benefits of a multi-stage classification strategy. With the multi-stage classifier, we are able to improve the classification accuracy of the structure-based datasets. The denoised wavelet coefficients again outperform the Zernike values, 70% to 54%, but there still is an improvement in accuracy in both datasets, 10% and 9%, respectively. The accuracy of the wavelet-Zernike dataset is

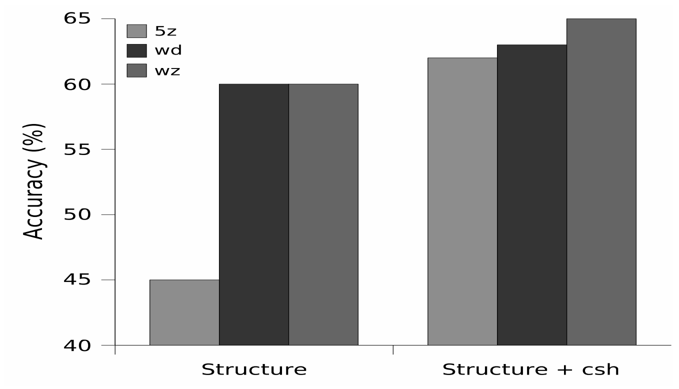


Figure 4.11: Accuracy of the single-stage Boosted C4.5 classifier on the denoised (*wd*), Zernike (*5z*) and wavelet-Zernike (*wz*) datasets (*Structure* columns). Also shown are the results of those datasets combined with the physio-chemical properties (*Structure + csh* columns).

actually worse than the pure-wavelet dataset, but only by 1%, which is not statistically significant. These results give credence to our belief that a multi-level approach to classification can be very beneficial in areas where the data falls within a natural hierarchy.

Even more impressive results are seen when we combine the structural datasets with the physio-chemical values. When combined, we can improve the accuracy of the denoised wavelet and Zernike datasets to 77% and 76%, respectively, and we can improve the wavelet-Zernike accuracy to 78%. *These values represent an improvement of 8% over the best structure-based multi-stage value, 15% larger than any single-stage classification result, and a full 34% larger than previously reported structure-based classification results* [72].

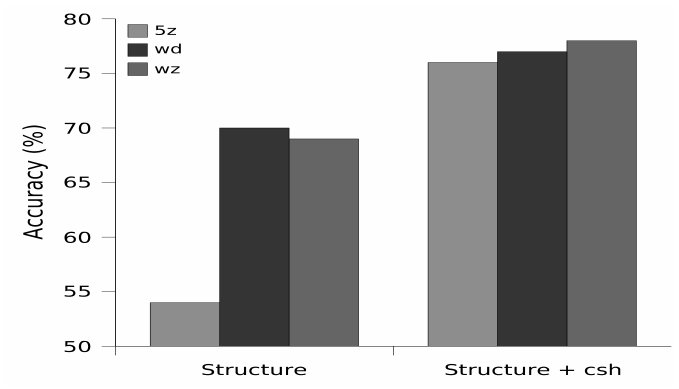


Figure 4.12: Accuracy of the multi-stage Boosted C4.5 classifier on the denoised (*wd*), Zernike (*5z*) and wavelet-Zernike (*wz*) datasets (*Structure* columns). Also shown are the results of those datasets combined with the physio-chemical properties (*Structure + csh* columns).

4.4.5 Discussion

In the above experiments, we evaluate two strategies to represent protein structure, one using Zernike polynomials, the other based on wavelet decompositions. We demonstrate the effectiveness of our approach by using our models as input features for the classification of a well-known protein dataset. In addition, when combining our methods with additional physio-chemical descriptors and using a multi-stage classification strategy, we can improve accuracy by almost 20% over existing sequence-based techniques and 34% over published structure-based results.

Most structure-based methods of protein analysis involve some sort of alignment process. Very few strategies exist that allow for the direct comparison of large numbers of protein structures. Our proposed representation yields a compact, effective description that can be used as part of any classification strategy. In addition, the coefficients can be combined with other datasets or descriptors to further increase

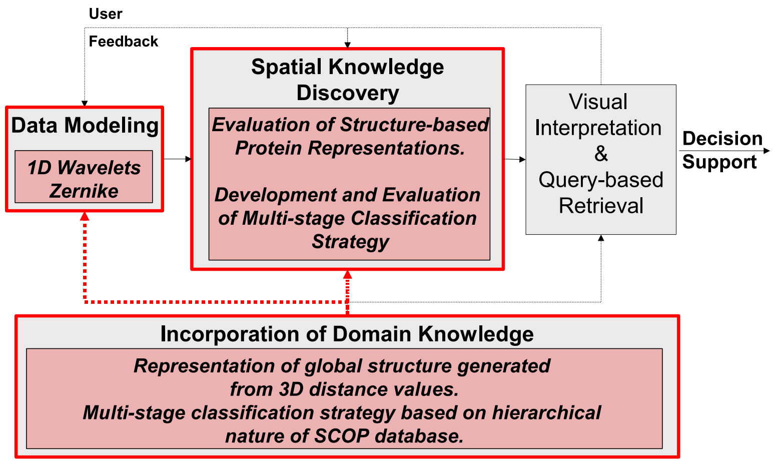

User
Feedback
Data Modeling
1D Wavelets
Zernike
Spatial Knowledge Discovery
Evaluation of Structure-based Protein Representations.
Development and Evaluation of Multi-stage Classification Strategy
Visual Interpretation & Query-based Retrieval
Decision Support
Incorporation of Domain Knowledge
Representation of global structure generated from 3D distance values.
Multi-stage classification strategy based on hierarchical nature of SCOP database.

Figure 4.13: Diagram of workflow as implemented in second case study.

performance. The representations described here can be computed quickly and efficiently, and can be tuned to the specific needs of any application. While the Zernike polynomials do not provide as high a classification accuracy as the wavelet method, there may be some cases where such a representation would still be useful.

4.5 Conclusions

As with our first case study, in Figure 4.13, we highlight the stages of our proposed workflow that were implemented in our experiments in protein structure classification. In this work, we incorporate domain knowledge into the modeling process, converting the raw structural information into a distance matrix before we model the object with wavelets or Zernike polynomials. Our work in Biomedical Knowledge Discovery tests several traditional classifiers, as well as a multi-stage classification strategy that leverages the hierarchical nature of protein structure databases to improve performance.

CHAPTER 5

CASE STUDY III: THE MODELING AND CLASSIFICATION OF VISUAL FIELD EXAMINATIONS

In this chapter, we present methods for the modeling and analysis of visual field examinations. These examinations are typically used in determining whether an individual suffers from a disease such as glaucoma. Often a diagnosis or course of treatment is based upon two criteria derived from these exams. The first is whether a patient's visual field contains an abnormality or defect. The second is whether that field is deteriorating, or the defect worsening, over time. Both factors can be related, but they can also be treated independently to a large degree. Thus, our efforts in this area begin with an algorithm to determine whether a patient's visual field is abnormal. We incorporate features of this algorithm into a second technique that characterizes the changes in that patient's visual field over time.

Our work with the visual field represents a slight departure from our earlier efforts in the modeling and analysis of proteins and corneal topography in that the visual field does not correspond to a physical structure. Each object in the topography dataset represents the surface of a patient's cornea and every protein is described by a set of 3D coordinates. Even our sequence-based protein transformation, which models the protein's amino acid sequence, can be viewed as structure-based, as the sequence has a direct influence on the final structure. Our work with the visual field can be viewed through the same lens, though the connection is less direct. Visual field loss can be caused by defects in the underlying structure of the eye, so we are still dealing with the relationship between structure and function as we did before. The difference is that here we are modeling the function in an attempt to determine whether there is a problem with the underlying structure.

Throughout this chapter, one of the issues that we hope to make clear is the difficulty in defining and detecting abnormalities in a visual field. One might assume that it is fairly simple to determine whether an abnormality exists in a visual field. Unfortunately, a visual field can be highly variable from one examination to the next. Thus, before declaring that a portion of a field contains an abnormality, one must be sure that the defect is substantial and more importantly, not transient. In other words, it should appear in successive exams and in roughly the same place. Many clinical systems that are used to diagnose abnormality are cross-sectional, but they still assume that any defect that is present remains in the same location and will get progressively worse over time. However, this not always the case. Indeed, to rule out so-called false positives, many clinical studies require that a patient have a defect in the same location over the course of 3 exams before their field is labeled as abnormal [13].

Our work is based on a manual classification system that was developed by Keltner et al. to categorize the characteristic abnormalities or defects seen in the visual fields of patients with glaucoma [13]. Examining a large dataset of patients, they derived a total of 17 different categories of abnormality, with some being more prevalent than others. For each category, they established a rule, often based on location or severity, to determine whether a potential defect would qualify or not. Each category was intended to be mutually exclusive, meaning that a defect would receive only one label, but it is possible for a defect to qualify for multiple categories. In such cases, a domain expert can usually determine which category is the most appropriate. However, even among experts, there is occasionally disagreement in their initial classifications, especially when the defects fall on the border between categories [13]. Despite these limitations, we feel that this classification system provides the best starting point for any type of automated system. Before presenting the rest of our solution, however, we cover additional information on the biological background of this problem as well as some of the existing work in the domain.

5.1 Biological Background

Glaucoma is a disease that affects the optic nerve and is one of the leading causes of blindness in the U.S. The exact causes of glaucoma are unknown, but treatment options exist if diagnosed early. If left untreated or undiagnosed, a loss of visual function will occur as the disease progresses. Unfortunately, no single standard or universal definition exists to determine if an individual is diseased or if a diseased patient is progressing. Despite the lack of a single standard, a number of alternatives do exist and many of them rely on the automated perimetry test, commonly called the visual field examination.

For patients being treated for, or suspected to have, glaucoma, visual field measurements are usually taken at yearly or half-yearly intervals. These assessments are examined by a clinician, who determines whether progression is occurring. To support their decision, a clinician can rely on any number of methods, including, but not limited to, summary statistics, point-wise regression, and regression techniques that rely on spatially-based clusters [73–75].

Despite this potential support, clinicians working in this area face a number of challenges. One major, and largely unavoidable challenge, is that a patient's visual field is highly (and naturally) variable. Therefore, a patient may seem to be progressing in one exam, but appear normal in the next. In addition, the sheer number of available support systems gives rise to a second challenge: when provided the same set of patient data, they may not yield the same overall assessment. Thus, the final decision of a clinician can be influenced by the method of assistance. In this work, we hope to develop a system that is robust enough to handle any natural field variability, yet performs with accuracy comparable to a domain expert.

5.1.1 Capturing the Visual Field

The visual field test is used to measure the sensitivity of the retina to light. A patient sits facing a spherical bowl that encompasses their field of vision and a light is flashed at various locations in a regular grid pattern (∼50-60 points). Retinal sensitivity at each location is a function of the brightness as reported by the patient [76]. This value is scored on a decibel scale and can range from 0 dB (blindness) to 35 dB (superior).

5.1.2 Clinical Display

The results of a visual field exam are typically displayed as either a gray-scale map of sensitivities or the raw sensitivity values themselves. This display often includes a secondary map that provides the probability of an abnormality occurring at each location in the patient's field based on their sensitivity values.

We now cover the steps taken to create these maps using a procedure described in Johnson et al. [77]. In that work, a best-fit least-squares linear function was created that mapped sensitivity as a function of age at each location in the field. The slope of this linear age vs. sensitivity function was then used to adjust the values of each patient to those of an average 45-year-old. By taking the difference between these age-adjusted values and those of an average 45-year-old, one is left with a set of values denoted as the patient's *Total Deviation* (TD). If one takes the 85th percentile sensitivity value (the 7th highest) and adjusts the Total Deviation numbers by the difference between the patient's 85th percentile and the average 85th percentile, one is left with a related measure known as *Pattern Deviation* (PD), which adjusts for the mean threshold surface of the entire field. With pattern deviation, any overall depression of visual threshold caused by age or cataract would not obscure any mean-adjusted pattern of field loss.

Given the deviation values, one can then determine whether a patient's sensitivity values fall below normal levels at any of the locations in the field. If it does, the location is marked on the probability map. Typically, a location must fall below normal at the 5% probability level to be marked (these probabilities are derived from normal control groups, not abnormal patient groups). Once a location meets this threshold, further abnormality is denoted by color, with different values used for the 5, 2, 1 and 0.5% probability levels.

Figure 5.1 provides an example of the results of a visual field examination that might be presented to a clinician. This exam corresponds to a right eye. The figure in the top left contains the raw sensitivity values for each test location in the visual field. The values in the top right have been adjusted for age. The middle left and middle right plots contain the Total and Pattern Deviation values, respectively. The bottom plots correspond to the Total and Pattern Deviation-based probability maps. In each figure, the '<' symbol denotes the location of the blind spot. Figure 5.1 also contains additional measures, such as *Mean Deviation* (MD), *Pattern Standard Deviation* (PSD) and the *Glaucoma Hemifield Test* (GHT), that can be used to assist a clinician when diagnosing a patient.

5.2 Related Work

In this section we cover work related to characterizing the change in a patient's visual field over time as well as work that attempts to model and detect change in anatomical shapes. We include such work here because a visual field can be viewed as a surface by treating the sensitivity thresholds as elevation values.

5.2.1 Visual Field Progression

A number of different researchers have looked into predicting visual field progression in glaucoma patients using the standard statistical techniques of longitudinal data analysis. Some methods excel at monitoring field summary measures, others at characterizing the rate of loss in sensitivity [73]. However, since there is no one analysis technique that stands out above the rest, a number of groups have attempted to create alternative methods of analysis. Gardiner and Crabb developed a method of point-wise linear regression (PLR) that provides better performance than standard PLR in applications where a high degree of specificity is required [73]. Nouri-Mahdavi et al. tried to predict progression in visual field test data by using the PLR method

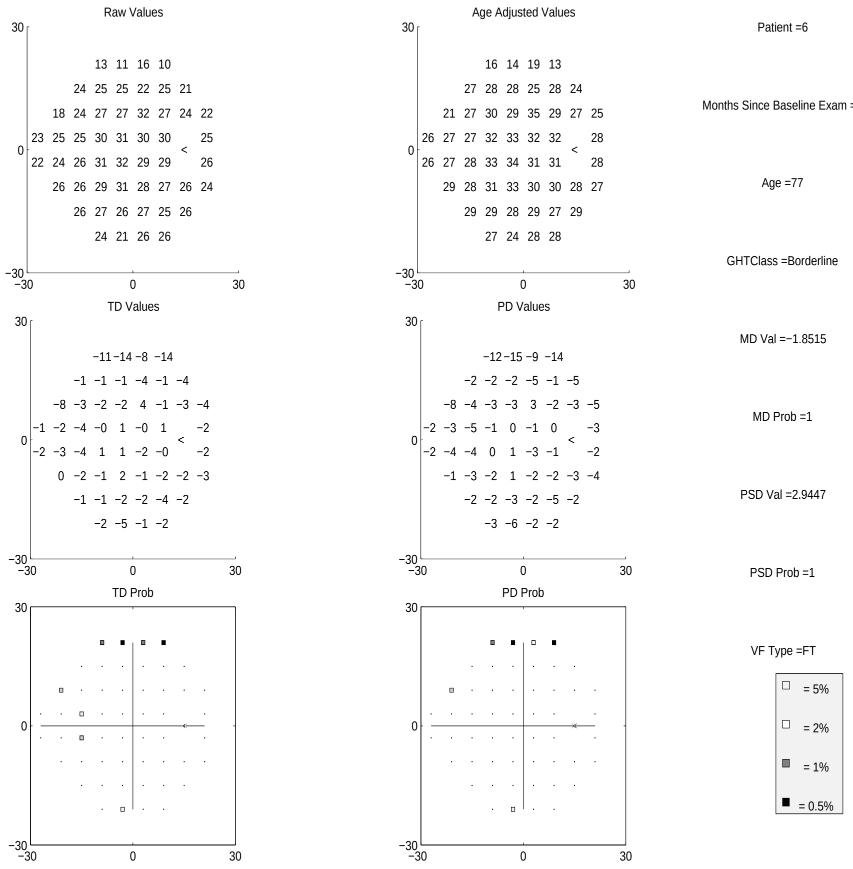


Figure 5.1: Results of a visual field examination (right eye). The figure in the top left provides the raw sensitivity value at each location. The plot on the top right shows the age-adjusted values. The values in the middle plots contain the Total and Pattern Deviation values (TD and PD, respectively). The bottom figures show the probability maps derived from the deviation values contained in the middle plots. The locations that are marked fall below normal at the the 5, 2, 1 and 0.5% probability thresholds. In each figure, the '<' symbol signifies the location of the blind spot. The measures listed on the right edge of the figure can also be used to assist a clinician in their diagnosis.

proposed by Gardiner and Crabb along with several other clinical and perimetric measures [74]. Goldbaum et al. used features derived from the raw visual field sensitivity values as input to several machine learning classifiers to see how they compared with the current analysis standards [78]. They tested support vector machines, mixture of Gaussian and generalized Gaussian classifiers. They used as input both the entire visual field as well as a summary that was created by reducing the original field with PCA, but they found that there was no statistically-significant difference between the variations. Other groups have reported similar findings [79]. Finally, Tucker et al. attempted to learn a Bayesian Network for classification and prediction, again using the raw visual field sensitivity values as input [75, 76].

5.2.2 Modeling and Detecting Change in Anatomical Shapes

Given image-based biomedical data, a number of researchers have implemented methods that look to model the overall shape of an object and then determine how much that object differs from a given population. If an object has been modeled over time, those measurements can also be used to quantify the degree of shape change. Such techniques would be useful in our analysis and characterization of the longitudinal changes seen in a visual field.

One method for detecting abnormalities or outliers in a dataset involves converting the boundary of an object into a time series and applying subsequence clustering techniques to the results [80, 81]. One can then calculate the amount of discord between objects or use dimensionality-reduction techniques to find outliers or abnormal shapes. These techniques generally require that each object be represented by many points, fall into relatively distinct clusters and be largely free of noise. Unfortunately, the visual field dataset does not meet any of these criteria.

Other techniques look to compute a medial representation [82], or capture variation from a mean using shape-based representations derived using spherical harmonics or spherical wavelets [83, 84]. After creating the shape descriptor, characterizing the differences between groups becomes a simple machine learning problem [85]. Unfortunately, these techniques require that each object be represented by a large number of points to generate an appropriate model. We are limited by the relatively small number of points that comprise each visual field. Also, the visual field exams are highly variable, meaning a deformation seen in one exam can become a protrusion in the next. Most of the above techniques are not intended for structures with such high variability.

While the above techniques may not be directly applicable to the visual field dataset, we can still use many of the underlying principles to guide our analysis. For instance, the idea of a medial representation is are already present in the visual field. The age-adjustment that is used to calculate the total deviation values implies that a median already exists. In fact, groups have attempted to use the raw age-adjusted values as input to several SVM variants [79], though the improvement over current methods was not statistically significant. This means that some degree of modeling will be necessary to improve performance over existing techniques.

5.3 Representing Visual Field Abnormalities

A number of different groups have looked to characterize and classify either single visual field exams or an overall examination history based on some form of the raw sensitivity values [74–76, 78, 79]. Despite these efforts, no one methods stands out above the rest. Part of the reason for this is that visual field exams are highly variable and patient can appear to be progressing in one exam and improving the next. Another problem is that there is no true consensus on what constitutes a normal or abnormal exam. Such a designation is often left to the judgement of the clinician.

To try and address this problem, Keltner et al. developed a manual classification system that attempts to describe and characterize the common defects and abnormalities appear in the visual fields of glaucoma patients [13]. Using a set of visual field examinations, they defined criteria for primary and secondary levels of abnormality. If a visual field met both criteria, it was then classified into one of 17 categories based on a set of rules that were devised by the study participants. As part of our analysis, one of our goals is to create a system that can automatically label a field as normal or abnormal. Our efforts at solving this problem begin with an attempt to codify the rules developed by Keltner et al.

At this point, we are interested only in determining abnormality. We are not trying to assign each field to a specific category, though we do plan on such an extension in the future. Part of the reason why we avoid making a specific assignment is that while each of the 17 categories are intended to be mutually exclusive (i.e. a defect would only receive one label), as currently defined, it is possible for an abnormality to satisfy the qualifying criteria for more than one rule.

Faced with a choice between a subset of categories, it is fairly easy for domain experts to determine the choice that best fits a given field. However, even with a set of well-defined rules, borderline or marginal cases are a source of disagreement between domain experts. In addition, over time, some defects actually progress and become a different type of defect. All of these challenges complicate the decision process of an automated system. Therefore, until there is more consensus within the vision science community as to the relationship and borders between the different defects, we will refrain from making specific category assignments for each field.

5.3.1 Defining Categories of Abnormality

The original work by Keltner et al. defined a total of 17 categories of abnormality. After consulting with domain experts, we decided that given our dataset, which is skewed toward patients who are normal or suspected to be in the early stages of glaucoma, we were unlikely to see patients from every single category. As a result, we focus on a smaller subset of 7 abnormalities that were deemed the most likely to be present in our data.

Before listing these abnormalities and their qualifying rules, however, we detail some of the location-based terminology used in their definitions. Figure 5.2 provides an example visual field, along with a numbered location of each position (locations 18 and 31 refer to the blind spot). Also provided is a label for each quadrant. The upper half of the field is denoted as the *superior* hemifield, while the lower half is referred to as the *inferior* hemifield. The *temporal* region of the visual field corresponds to the edge closest to the blind spot, while the other side, closest to the nose, is called the *nasal* region. And while it is not labeled on this particular figure, the area between the temporal and nasal regions contains the optic nerve and we refer to it as the *nerve fiber bundle* region.

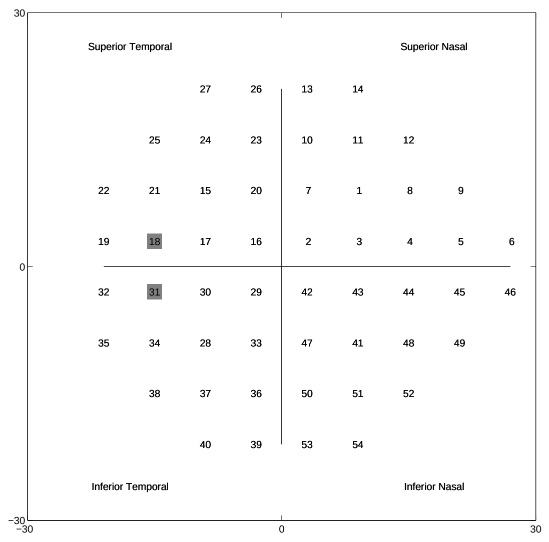


Figure 5.2: Visual field test locations. Positions 18 and 31 denote the blind spot.

Some of the abnormalities are contained within a single hemifield and can thus be defined as a superior or inferior defect. For such abnormalities, we only provide rules that correspond to the superior hemifield. To derive the rules for the inferior hemifield, the locations are simply reflected across the horizontal midline. Finally, when a point or location is defined as *abnormal*, that means that it has a Total or Pattern Deviation probability level 5% or less. We provide the qualifying rules for each defect in Figure 5.3

In Figure 5.4, we provide examples of the four most prevalent defect types in our dataset. They are, in no particular order, the Nasal Step, Paracentral, Partial Arcuate and Arcuate defects (a through d, respectively). We present each example in the form of a Total Deviation probability map, where each abnormal point is represented by

Superior Nasal Step:
1. At least one point in the range [4-6] must be abnormal.
2. Less than two points in the group [15,17,20,21,23,24] are abnormal.

Superior Partial Arcuate:
1. At least one abnormal point in the range [4-6] or [15,21].
2. At least one abnormal point in the temporal visual field ([15-27]).

Superior Arcuate:
1. At least one abnormal point in the range [4-6] and [15,21].
2. A contiguous line of abnormal points between [4-6] and [15,21].
   The contiguous points must include [20,23,24,7,10,11,1,8]
   (but not necessarily all points).

Superior Paracentral:
1. No abnormal points in [4-6].
2. At least one abnormal point in [7,10,11,15,17,20,23,24].

Partial Peripheral Rim:
1. 3 or more, but 15 or fewer contiguous abnormal points
   in [6,9,12,14,13,26,27,25,22,19,32,35,38,40,39,53,54,52,49,46].

Peripheral Rim:
1. More than 15 contiguous abnormal points
   in [6,9,12,14,13,26,27,25,22,19,32,35,38,40,39,53,54,52,49,46].

Widespread:
1. A glaucoma hemifield test classification of
   "*General Reduction in Sensitivity.*"

OR

1. A mean deviation probability of less than 5%.
2. A pattern standard deviation probability greater than or equal to 5%.
3. More than one abnormal point in each quadrant:
   Quadrant 1: [1:14]
   Quadrant 2: [15:27] (excluding 18 - Blind Spot)
   Quadrant 3: [28:40] (excluding 31 - Blind Spot)
   Quadrant 4: [41:54]

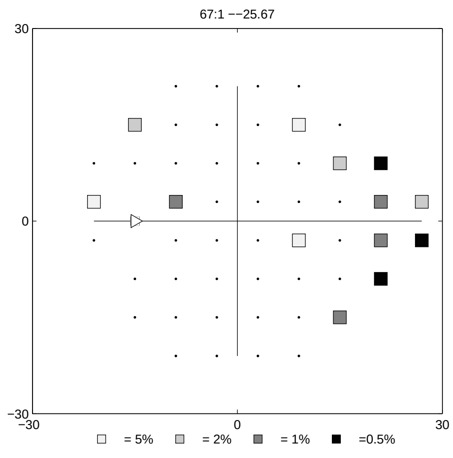
Figure 5.3: Qualifying rules for each defect category.

a grayscale square, with the color corresponding to the probability of abnormality. Normal points are represented by a dot and the blind spot is denoted with a '>' symbol.

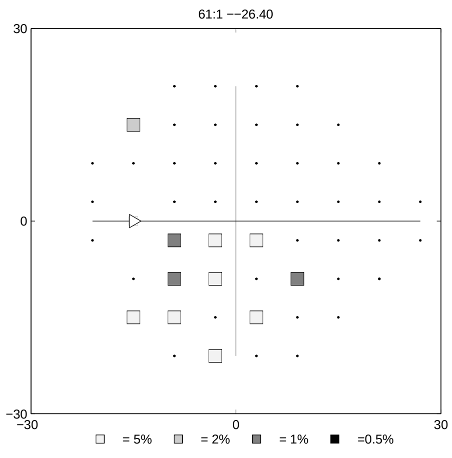
After our initial experiments, we also added one additional category that was designed to capture the overall loss or reduction in visual field sensitivity. We call this the *General Reduction* abnormality. We calculate a score for this category for every field, i.e. it does not have any qualifying criteria. In our current evaluation, however, we ignore the General Reduction category as it does not capture any physical defect. It is intended for use in the future analysis of this dataset.

As stated above, given a field, we want to determine whether it has a defect that matches the qualifying criteria of any of the abnormality categories, and if so, we wish to assign a score that characterizes how well the defect matches the definition. The paper by Keltner et al. contains enough information to determine whether a defect satisfies the qualifying rule for a given category. It does not, however, provide information on how to determine how *well* an abnormality matches that definition. Since a defect may satisfy multiple rules, one would prefer a system where the category that best matches the defect would receive the highest score.

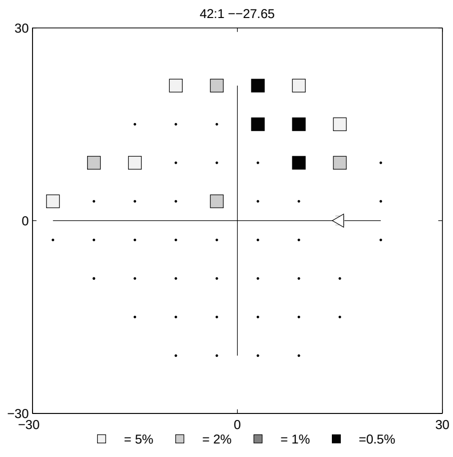
Therefore, we decided to create a template for each category that would assign a probability score to each location in the visual field. The locations that were highly likely to appear in a particular defect or abnormality would receive a high score, while those locations that were slightly less likely would receive a lower score. Those locations that were unlikely or very unlikely to appear would receive no score or a negative value. The idea behind the negative values is to penalize and lower the overall score, making the field appear like less of a match.



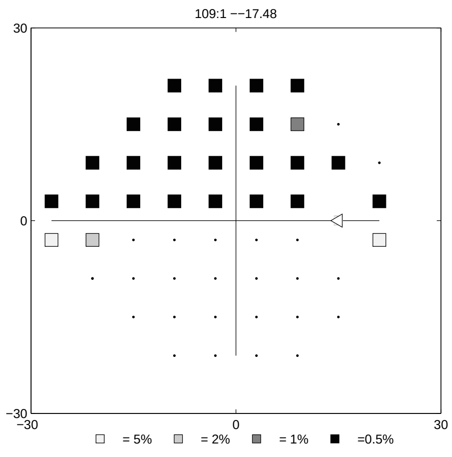
(a)



(b)



(c)



(d)

Figure 5.4: Total Deviation probability plots of the four most common defect categories in the visual field dataset. The defects are as follows: (a) Superior and Inferior Nasal Step, (b) Inferior Paracentral, (c) Superior Partial Arcuate, and (d) Superior Arcuate. abnormality categories. Abnormal points are represented by a square, with the probability level denoted by color. Each dot corresponds to a normal location and the '>' character denotes the blind spot.

The probability maps for each category, whose values were assigned by a domain expert, are provided in Figure 5.5. Each color-coded location in the map can have a positive score ranging from 1-4, a neutral score (0), or a negative score of -1 or -2. For locations with a positive score, a value of 1 indicates a position where a given defect is likeliest to appear abnormal. As the value increases, the likelihood of abnormality decreases. For locations with a negative score, the opposite is true. Locations with a score of -2 are penalized more than locations with a -1. Neutral locations are ignored when assigning a score. Please note that the Arcuate and Partial Arcuate defects share a probability map. The same is true for Peripheral Rim and Partial Peripheral Rim. Finally, the Widespread and General Reduction categories use a map where every location is assigned an equal probability level.

5.3.2 Calculating Individual Category Scores

Given a visual field, we calculate a score for each abnormality category using the procedure described below. We first take the raw threshold values and apply an age-adjustment to normalize them to those of a 45 year-old. Using the adjusted sensitivity scores, we calculate the Total Deviation values and probabilities for each location in the field. We also calculate the other domain-specific information that is required to determine whether a field satisfies the criteria for certain categories (particularly Widespread). This information includes the Mean Deviation (value and probability), Pattern Standard Deviation (value and probability) and Glaucoma Hemifield Test classification[5].

[5]The code to calculate these measures is proprietary and was provided by Michael Twa and Chris Johnson. The routines used by this software are detailed in Johnson et al. [77]

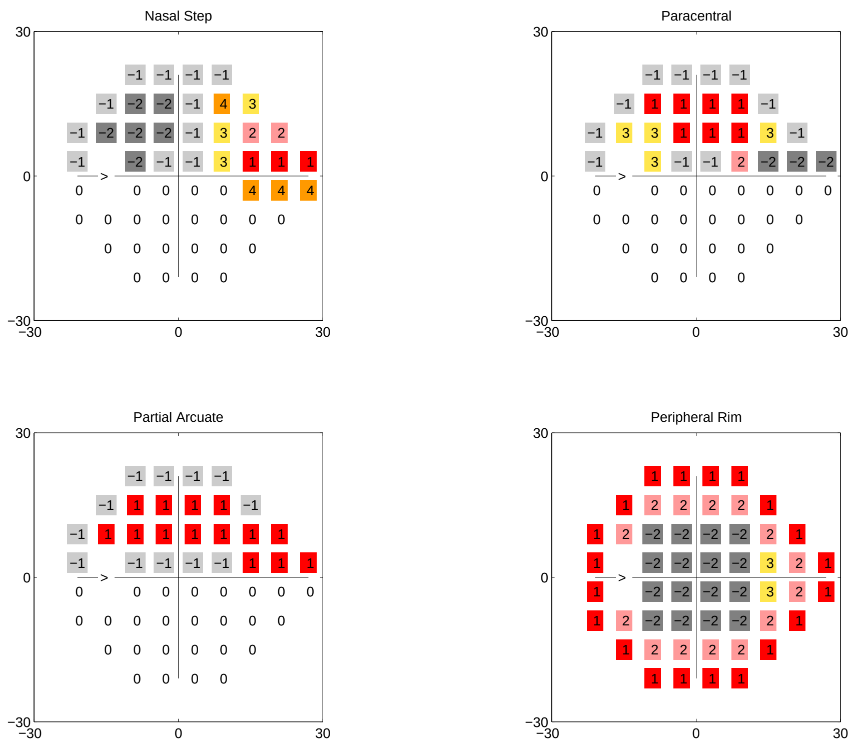


Figure 5.5: Probability map templates for the various abnormality categories. The map for Nasal Step is given in the top left, Paracentral in the top right, Partial Arcuate in the bottom left and Partial Peripheral Rim in the bottom right. Each location is assigned a positive, neutral, or negative score based on the likelihood of abnormality for that category (the '>' character denotes the blind spot).

Once we have completed these calculations, we determine whether a field matches the criteria for any of the defect categories. If it does, we take the Total Deviation sensitivity values, and for each location, we multiply the sensitivity value by a scaling factor that is based on the value in the probability map. These scaling factors, determined empirically, are provided in Table 5.1. We take a sum over each field and use this value as our overall abnormality score (the Total Deviation sensitivity values are positive if a patient has better than average sensitivity at a particular location. *We only consider those locations where a patient has less than average sensitivity.* These values are negative, so we use their absolute value when calculating each abnormality score).

As opposed to the original work by Keltner et al., where a field would be classified only if it satisfied two initial tests for abnormality, we pass every field through our scoring procedure. A normal field would not satisfy any of the qualifying criteria and would thus receive an abnormality score of zero. Checking every field removes the need to implement additional checks for abnormality.

In our initial tests, we would only allow a point to contribute to the overall score if it was abnormal at the 5% probability level (or less) on the Total Deviation plot. If it was, we would assign a score based on the probability level itself (5, 2, 1, or 0.5%), not the underlying sensitivity value. During consultations with others in the vision science community, concern was raised that we could miss potentially useful information from locations that were not quite abnormal - those that were below normal at a probability level of 6%, for instance. In addition, it was felt that using the Total Deviation sensitivity values, and not the probability level, would provide a more continuous score that could better capture the varying degrees of abnormality.

Probability Value	-2	-1	0	4	3	2	1
Scaling Factor	-0.25	-0.1	0	0.1	0.15	0.25	0.5

Table 5.1: Probability map values and associated scaling factors.

As a result, we elected to include all locations in our calculations and use the actual sensitivity values instead of the probability levels.

5.3.3 Calculating Total Field Scores

Once we have calculated a score for each qualifying category, we can begin to determine whether the *overall field* is abnormal. In this work, our goal is not a cross-sectional analysis where we associate each field with a specific defect. We are more interested in assigning a label based on a longitudinal analysis of a patient's examination history. It is possible that a defect can change from one type of abnormality to another as the disease progresses. For instance, an abnormality that begins as a Partial Arcuate can progress into an Arcuate defect. This is the type of change we hope to capture through our analysis. To facilitate this effort, we group the defects based on their location in the visual field.

To create these groupings, we divide the visual field vertically into three different regions: nasal, temporal, and nerve fiber bundle. We next divide the field horizontally into inferior and superior hemifields. We then create groupings based on the defects that generally occur in each location. We also create a separate group for the defects that affect the entire visual field. These groupings, and the defect categories that comprise them, are listed in Table 5.2. Combining the abnormalities in this way allows us to characterize a patient as having a defect in a particular location without

Group	Composite Abnormality Categories
Superior Nasal	Superior Nasal Step
Superior Nerve Fiber Bundle Region	Partial Arcuate, Arcuate, Paracentral (all Superior)
Inferior Nasal	Inferior Nasal Step
Inferior Nerve Fiber Bundle Region	Partial Arcuate, Arcuate, Paracentral (all Inferior)
Whole Field	Widespread, Peripheral Rim, Partial Peripheral Rim
General Reduction	General Reduction

Table 5.2: Location-based defect groups and their corresponding abnormality categories.

```
1. Age-adjust sensitivity values.
2. Compute domain-specific measures, including:
     Total Deviation, Pattern Deviation, Mean Deviation,
     Pattern Standard Deviation, as well as a classification
     by the Glaucoma Hemifield Test.
3. Check whether the field satisfies any of defect criteria.
4. Compute an abnormality score for each the qualifying categories.
5. Calculate the location-based group summary scores.
```

Figure 5.6: Algorithm to model visual field defects.

having to determine whether the defect switches from category X to category Y between exams. Currently, we do not list either of the Temporal groups. This is because the primary defect located in the Temporal region, the Temporal Wedge, was not one of the defects likely to be present in our dataset. When we begin to test for this defect, we will add the requisite groups. We calculate a score for each group by taking the sum of the abnormality scores of the underlying categories. Our approach to modeling a visual field based on its underlying defects is summarized in the pseudo-code provided in Figure 5.6.

5.4 Characterization and Classification of Visual Field Examinations

In the previous section, we describe a technique to characterize the abnormalities that may be present in a visual field. We define several categories of abnormality and for each one we create a template that contains the probability of abnormality at each location in the visual field. Given a field, we compare it to each template and assign a score based on how well it matches. Those efforts were limited to a single visual field. Here we present an approach to characterize a patient's entire visual field examination history. This longitudinal analysis characterizes both the degree of field abnormality as well as the rate of change in the field between exams. *This system is automatic, yet tunable, allowing clinicians to have control over the sensitivity of the results.* We begin with our approach to determining field abnormality and follow with a description of how we characterize the rate of change.

5.4.1 Determining Field Abnormality

Executing the scoring procedure described in the previous section will result in a set of five location-based group summary scores. These values summarize the scores of the individual abnormality categories and are intended to characterize the amount of abnormality in the Nasal and Nerve Fiber Bundle regions (Inferior and Superior), as well as any Whole Field defects that may be present.

Before determining whether a particular field is normal or abnormal, we generate a series of summary statistics to determine what constitutes normality. To do this, we execute our scoring procedure to compute location-based group scores for every field in our dataset. We then calculate the median and standard deviation of each group.

To determine whether a patient's visual field is abnormal, we examine the score for each location-based group. If the score for that group is greater than $X * \sigma$, where X is a user-defined threshold and σ the standard deviation, we say that an abnormality exists in that group (or location). Inherent in this calculation is the assumption that normal fields have summary scores of zero.

This process is repeated for every one of a patient's exams. If a patient has a single abnormality in any group, we declare that patient's field to be abnormal. One can relax the abnormality criteria by requiring that a patient have several abnormal exams in a particular group as opposed to just one. However, our dataset was labeled so that a patient's field was declared abnormal if a single defect was present and we elect to use the same standard. We should note that it is certainly possible for a patient to be declared abnormal in multiple categories. This simply implies that any defects that are present are likely to be widespread and span multiple locations within the field.

5.4.2 Characterizing Rate of Change

To determine the rate of change in a patient's visual field, we take the group scores for each field and compute the difference between consecutive exams, as well as between the first and last. We then use these differences to calculate the percentage change for each group. For instances where the summary score changes from some value to zero, or from zero to some value (resulting in a percentage change of -100 or infinity, respectively), we use the median value of the group instead of zero.

Type of Change	Criteria
Progression	$C \geq X\%$.
Probable Progression	$Y\% < C < X\%$.
Relative Stability	$-Y\% \geq C \leq Y\%$.
Probable Improvement	$-X\% < C < -Y\%$.
Improvement	$C \leq -X\%$.

Table 5.3: Categories of inter-exam field change.

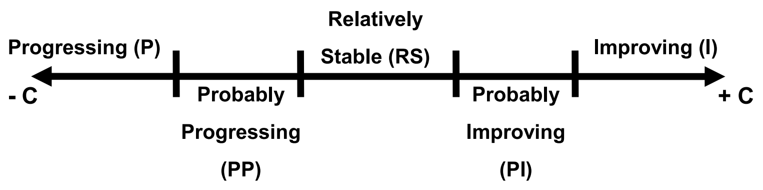


Figure 5.7: Line diagram of inter-exam categories of change.

Using these percentages, we compute a tally for the patient based on the magnitude and direction of the change. We have specified five different types of change that can occur. We list these changes and the criteria for each in Table 5.3 (assume C represents the inter-exam percentage change). A graphical view of these categories is presented in Figure 5.7.

The inter-exam change for each summary group will fall into one of the categories listed in Table 5.3. While these categories are similar to the labels used to characterize the change in our dataset, we cannot simply take a majority vote and assign that category to the patient. If an individual has a field defect that is present and progressing in only one group (i.e. the other two group scores are zero), a majority of votes for that patient would fall into the *Relatively Stable* category, when that individual should probably be classified as *Progressing.* On the other hand, if a patient had a defect that fell into the *Probably Progressing* category, but for only a few exams, then one might want to actually classify that patient as *Relatively Stable.* As one can see, assigning a category to characterize the change in visual fields is not a straightforward process.

To address this problem, we propose the following: First, we compute a tally for the inter-exam change of each summary group. Then, we calculate a score S using the equation given below:

$$S = -Z * P + (-1/Z) * PP + (1/Z) * PI + Z * I \tag{5.1}$$

where *P, PP, PI,* and I represent the number of votes in the *Progressing, Probably Progressing, Probably Improving* and *Improving* categories, respectively. Z represents a user-defined weight value (set to 2 in our experiments). The votes for *Progressing* and *Improving* receive a larger weight because they represent a more significant change.

The value for S will be either positive (implying field improvement), negative (defect progression), or zero. A zero value will occur when there is either no abnormality present, or the number of votes on the progression side of the equation cancel out those on the improvement side. Once we compute S, we use that value as well as the number of *Relatively Stable* (RS) votes to assign a label for each patient. The pseudo-code for assigning a label is provided in Figure 5.8. In this algorithm, the sign of S is only used to denote whether a patient is leaning toward progression or improvement. *When assigning a label, we treat S as a negative value with respect to RS.* Thus, for the rest of the discussion, assume that in all calculations, the value of S is negative. We refer to the original sign of S only to determine whether a patient is progressing or improving. A graphical, flowchart version of the pseudo-code is presented in Figure 5.9.

To summarize our approach, if the value of $S + RS < 0$, then that patient is said to be *Progressing* or *Improving*, depending on the original sign of S. If $(S + RS)/(\textit{Number of Exams}) \leq (NG - 1)$, where $(NG - 1)$ is equal to the number of location-based groups minus 1 (4 in our case), then a patient is said to be *Probably Stable (Progressing or Improving)*, depending on the original sign of S. Otherwise, we label a patient as *Stable*. It might seem like the value of $(NG - 1)$ is a rather arbitrary selection, but there is some logic behind the choice. First of all, $(S + RS)/(\textit{Number of Exams})$ will always be less than NG, since we are only considering the 5 location-based summary groups. If $(S + RS)/(\textit{Number of Exams}) > (NG - 1)$, that implies that at least four of the groups are Relatively Stable (RS) and the fifth is not "unstable" enough to be definitely progressing or improving (there could be some instability in all 5 groups, but overall it would appear to be relatively minor). We want to designate such patients as *Stable*.

5.4.3 Additional Applications

Though we present our approach in a classification context, we are not limited solely to this task. Our methods can also be employed in other applications, such as detecting outliers or discovering anomalous fields and patient histories in unlabeled data. The declaration of a visual field as normal or abnormal does not rely on any previous labeling of the dataset. The same applies to the labels quantifying the degree

```
If $S < 0$
    If $S + RS < 0$
        *Progressing*
    Elseif $(S + RS)/(Number\ of\ Exams) > (NG - 1)$
        *Stable*
    Else
        *Probably Stable (Progressing)*
    End
Elseif $S > 0$
    If $-S + RS < 0$
        *Improving*
    Elseif $(-S + RS)/(Number\ of\ Exams) > (NG - 1)$
        *Stable*
    Else
        *Probably Stable (Improving)*
    End
Elseif $S == 0$ && $RS > 0$
    *Stable*
Else
    *Probably Stable*
End
```

Figure 5.8: Algorithm for assigning change-based labels.

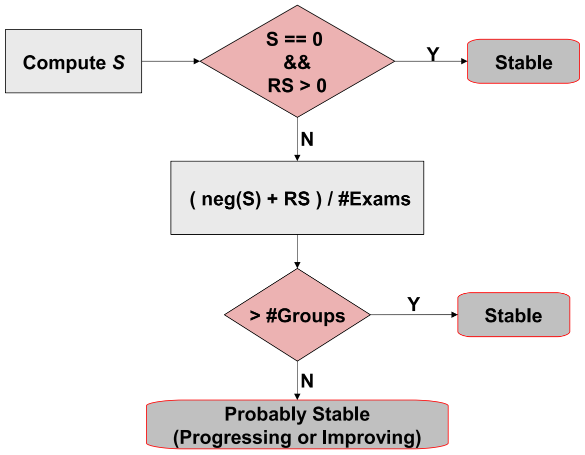


Figure 5.9: Flowchart for assigning change-based labels.

of inter-exam change. This means that our technique could be used to flag anomalous cases such as:

- Patients whose visual field is marked as abnormal in every location-based group.
- A patient with a normal visual field that is declared to be *Progressing.*
- Similarly, a patient with an abnormal but *Improving* field.

One could also retrieve the top-k abnormal visual fields, based on the total or per-group abnormality scores. Finally, one might also be interested in finding those patients whose visual fields are abnormal in more than $X\%$ of their exams or whose exams alternate between *Progressing* and *Improving.* Any patient with one of the above history types might warrant additional scrutiny by a clinician.

5.5 Dataset

The dataset in our visual field experiments is composed of a group of normal patients as well as individuals who have been diagnosed with glaucoma, with most people having between 1 and 3 years of follow-up exams. The complete dataset contains 192 individuals and a total of 1166 exams, with the number of exams roughly divided equally between right and left eyes. Using this dataset, we wish to perform two types of analysis. The first is more cross-sectional in nature and involves determining whether a patient's visual field can be considered normal or abnormal. The second analysis is a longitudinal study that involves characterizing the change in a patient's visual field over time. We felt it would not be possible to adequately categorize the change in patient's field using just two exams, so we focus only on those individuals that have participated in at least three examinations.

This dataset was originally provided without labels, so in order to do any type of validation, it needed to be analyzed by a domain expert. To ease the burden imposed by this process, we limit ourselves to working with a single eye for each individual. Instead of simply choosing just right or left eyes, however, we select the "worst" eye, which is defined to be the one with the lowest baseline Mean Deviation value. If that eye is associated with less than 3 exams, however, we use the other. In the rare instance where the deviation value is the same for both eyes, we simply select one at random. After removing those patients with less than 3 exams and pruning the dataset of the "good" eyes, we are left with a total of 135 patients and 501 examinations.

We asked a domain expert to assign two different labels to each patient, one based on the defects present in the visual field, the other on the changes in that field over time. The labels and assignment criteria are listed below. We also provide the number of patients assigned to each category in parentheses.

- **Abnormality:** If any of a patient's visual field exams contain one of the defects listed in Keltner et al. [13], that patient was labeled as *Abnormal* (68). If all of their exams were defect-free, they were considered to be *Normal* (67).

- **Change:** Each patient was also labeled based on the changes in their visual field sensitivity over time. This was done by analyzing their examination history and assessing the relative stability of the field. A patient was placed into one of the following categories:

    1. *Stable* (60)

    2. *Probably Stable* (56)

3. *Probably Progressing* (7)

4. *Progressing* (2)

5. *Probably Improving* (8)

6. *Improving* (2)

An assignment of *Stable* or *Probably Stable* implies that there is little change in a patient's visual field over time. A patient that is said to be *Progressing* or *Probably Progressing* would show some degree of deterioration in their visual field or have a worsening defect, while an individual who is labeled as *Improving* or *Probably Improving* would appear to get "better" over time. This could mean that a defect that was present during a baseline exam was no longer evident in subsequent exams, or that a serious defect became less severe in later tests.

5.6 Validation

To validate our approach, we perform two sets of experiments. In the first set, we focus on determining whether the visual field of a patient is normal or abnormal. In second, we switch our attention to characterizing the rate of change in a patient's field.

5.6.1 Experiments

In our first experiments, we execute our abnormality scoring procedure and assign a label of normal or abnormal to each patient. We then compare our classification with the assignment provided by our domain expert. Again, we note that in these experiments, we ignore the General Reduction group as it does not correspond to any physical defect. Therefore, a field can only be declared abnormal based on defects in the Whole Field, Nasal or Nerve Fiber Bundle groups (Superior and Inferior).

In our experiments, we use a value of 0.6 for the X parameter of the Whole Field and two Nerve Fiber Bundle groups. A value of 2.1 was chosen for the two Nasal groups. We tried a number of different values and these selections provided the best tradeoff between the number of false positives and false negatives. If one wished to weight the algorithm so that it minimized false positives at the expense false negatives (or vice versa), alternate values of X could be employed.

In our second set of experiments we characterize the rate of inter-exam change in a visual field (or lack thereof) by assigning a progression-based label to each patient in our dataset. As stated previously, our approach is tunable, allowing the user to set several parameters to control the quality of results. In assigning a change-based label, we allow the user to specify the cutoff threshold between the categories of *Progression* and *Probably Progressing* and between *Probably Progressing* and *Relatively Stable* (X and Y, respectively). In these experiments, we used a value of 60% for X and 20% for Y. The parameters for both sets of experiments were chosen empirically, but we plan a more robust evaluation in the future to try and determine the optimal values. We will likely use a genetic, gradient descent or simulated annealing-based algorithm for this task [86, 87]. With such an approach, the input is the set of m parameter values and the output is the degree or percentage agreement with the expert classification. This presents an $(m + 1)$-dimensional response surface where one searches over the m input dimensions to determine the optimal output value. This procedure has been used to determine the optimal configuration in a number of applications, ranging from vehicle design to network traffic and the distribution of electricity [88, 89].

Like our first set of experiments, after choosing a label for each patient, we compare our selection to that of the domain expert. As originally labeled, a patient could be assigned to one of six different categories. In our algorithm, however, we limit an individual to one of five possibilities. This is largely a result of the relatively small number of patients who are labeled as *Progressing* or *Improving*.

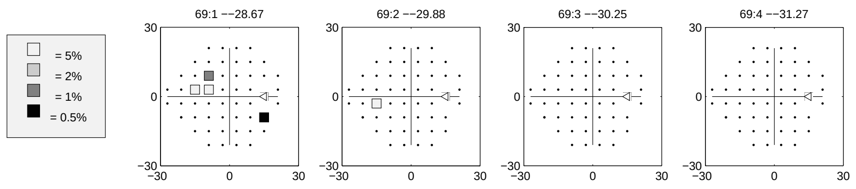
Thus, when determining whether our assignment is in agreement, we allow for some fuzziness and overlap. For instance, a label of *Stable* in one system and *Probably Stable* in another is considered to be a match. In addition, a label of *Probably Progressing* or *Progressing* by our expert could be matched by a label of either *Probably Stable (Progressing)* or *Progressing* by our approach (similarly for *Improving*).

5.6.2 Results

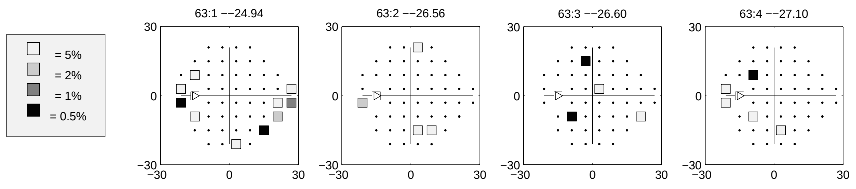
Overall, we found that our algorithm performed very well in detecting field abnormalities. Of the 135 patients in the dataset, we only disagreed with the expert classification in 14 cases. These misclassifications were almost evenly split between false positives and false negatives (8 and 6, respectively), and the defects that we missed tended to match the overall distribution of defects in our dataset (i.e. we did not consistently miss one defect over the others). This translates into a roughly 90% agreement between our system and the domain expert. We examined the patients where the classifications disagreed, and we found that with 3 of the false negatives, the defects labeled by the clinician did not technically satisfy the qualifying rules for that particular defect category. Such a "defect" would never be detected by an automatic system such as ours without changing or relaxing the qualifying criteria. These cases illustrate some of the danger that is inherent with any approach that relies on strict classification rules. Even so, our results are very promising, as studies in the literature report that even among multiple domain experts, agreement tends to hover around 87-88% [13].

In our second set of experiments to characterize inter-exam change, we also report very promising results. Using the somewhat fuzzy matching defined above, we found that we were in agreement with the expert classification roughly 91% of the time, misclassifying 12 patients in total. In all cases but one, when there was disagreement between the two system, the expert classification was *Probably Stable* compared to an assignment of *Progressing* or *Improving* by our algorithm. In no cases did we classify a *Stable* patient as *Progressing* or *Improving.* In fact, in several of the instances where there was disagreement, one could make an argument that the patient's field does indeed appear to be progressing or improving. In a traditional setting, these disagreements would likely be re-examined by the rating experts and adjudicated until a consensus was reached. It also serves to highlight the imprecise nature of assigning progression-based labels to patients. However, despite the misclassifications, we are very encouraged the performance of our algorithm.

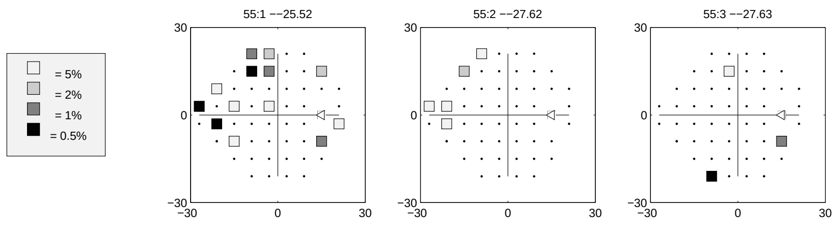
In Figure 5.10, we present examples where our abnormality or progression assignments differed from those of the domain expert. Each history is represented as a series of Total Deviation probability plots, with the abnormal points again represented by grayscale squares. As before, the probability of abnormality is denoted by color. In the top history (a), we show a patient that was classified as having a Superior Paracentral defect by the expert, but normal in our system. We draw particular attention to this case because it represents one of the instances where the defect specified by the expert technically fails to meet the qualifying criteria for that category. In the first,



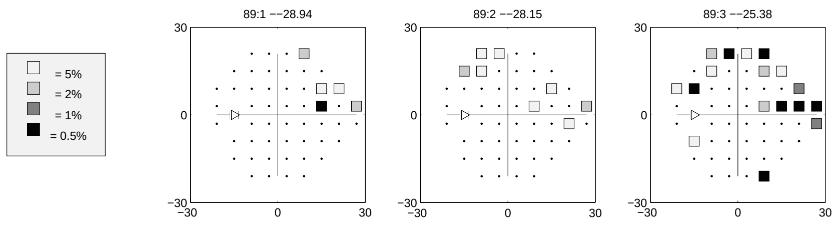
(a)



(b)



(c)



(d)

Figure 5.10: Example examination histories for the patients where our abnormality or progression assignments disagreed with the expert. The top history (a) represents a patient classified as abnormal by the domain expert but normal by our system and (b) corresponds to the opposite case. We considered the patient in (c) to be Improving and (d) as Progressing. Both were labeled as Probably Stable by the domain expert.

or left-most exam, which is the one said to contain the defect, the abnormal point in location 6 (the left-most abnormal point above the horizontal midline) should disqualify this defect from consideration in the paracentral category. We present this example to show the difficulty presented by abnormalities that straddle the border between categories.

The history in in Figure 5.10 (b) corresponds to a patient that was classified as abnormal by our system and normal by the domain expert. While the patient's fields contain a number of abnormal points that would seem to indicate a defect, none of the points are consistently abnormal. As a result, one cannot definitively say that a specific defect exists. This shows that while one can consider each patient's visual field in isolation to detect defects, there may be instances where several fields must be considered in total to determine whether a defect really exists or if the supposed abnormality simply represents natural variability.

The last two examples (Figure 5.10 (c) and (d)) are patients where our progression-based assignment disagreed with the expert. Both of these patients were classified as *Probably Stable* by the expert, but *Improving* and *Progressing* by our system. While there does seem to be improvement or progression in each field, it is difficult to say whether the defect is truly diminishing or worsening. Thus, one could make the argument that the defects are probably stable. We show these examples to illustrate the challenges in making progression-based assignments.

5.7 Conclusions

In this section, we present our approach to model the defects that may be present in a patient's visual field. This model can be used to classify a patient's entire field as normal or abnormal and by looking at the inter-exam changes in this model, one can also assign a label that characterizes the direction and degree of change. We find that the labels assigned by our approach largely agree with those assigned by a domain expert. We are in agreement with the expert classification 90% of the time in terms of labeling a field as normal or abnormal and 91% of the time in characterizing inter-exam change. These values compare favorably with those reported between domain experts, and outperform many of the automated systems in use today.

In addition, our approach contains a number of user-tunable parameters, which can be changed to minimize false positives and false negatives, depending on the task at hand. This is critical, as our evaluations are somewhat preliminary, being limited to a single dataset. Datasets that contain a different ratio of diseased patients versus normal individuals, or datasets that contain a greater number of patients with additional or more severe defects are likely to require a different set of parameters in order to achieve the most effective performance. Thus, it is likely that our parameter selections do not represent the globally-optimal solution. We did check a range of values during our tests, however, and believe that we have found at least a locally-optimal solution. While this might imply that we have overfit our algorithm to the data, until we can evaluate our technique on multiple datasets, we are left with our current selections. Once we have the opportunity to expand our efforts, we plan to use a genetic, gradient descent, or simulated annealing-based algorithm to ensure that the optimal input parameters have been chosen [86,87]. By treating the user-tunable parameters as a set of inputs and the percent agreement with the expert labels as an output, one can check a range of values to determine those that yield an optimal output. We feel that such an effort is outside the focus of this study, however.

Despite the promising results seen in our evaluation, our approach does contain several limitations that we intend to address in the near future. The most pressing need is to expand the capabilities of our scoring procedure in declaring fields as normal or abnormal. Currently, we assign a label to a field based on the sum total of all the defects that might be present in a particular region or hemifield. Realistically, only one defect will be present in a given location. Thus, we must create a method to normalize the abnormality scores of each category so that the category matching the defect returns the highest score. This will require additional work with clinicians to clearly define the borders between the different defects as well as how to handle patients whose defects evolve from one type to another.

Part of problem with creating a method to normalize the abnormality scores is that in using the Total Deviation sensitivity values, there is no maximally-abnormal score for each location (other than 0 dB, or blindness). When we computed the score based on the probability of abnormality, we could say that a location that was abnormal at the 0.05% probability level exhibited the maximum abnormality possible. With the "continuous" sensitivity values, it is much more difficult to make such a declaration. Despite this limitation, we feel that using the actual sensitivity values provide the user with a much better idea as to the true nature of an abnormality. Such a view could be masked by relying on the discrete and relatively sparse probability levels.

While we limit our evaluations to comparing performance against a labeled dataset, our approach can also be applied to unlabeled data. Though further research is needed before this system is capable of labeling an abnormal field with a single, specific defect category, it can be used to retrieve examinations that appear to be outliers or anomalous in comparison to the rest of the dataset. Using the abnormality scores assigned by our algorithm, one could retrieve the most abnormal individual examinations, or if a patient has a certain percentage of abnormal fields, an entire examination history. One could either specify a level at which an abnormal exam is declared to be an outlier, or, based on the scores of all the exams in the dataset, simply retrieve the k-most abnormal fields or examination histories. Such a feature would allow researchers or clinicians to quickly flag patients on whom they might wish to focus additional treatment or attention.

As a further extension, we can also use our technique as a method of flagging early disease cases by removing or relaxing the qualifying rules for each defect category. As we saw in our experimental evaluation, there were several instances where an individual was labeled as abnormal by the domain expert because of the presence of a certain defect, but our approach failed to reach the same conclusion because the defect in question did not technically satisfy the qualifying criteria for that category. While a clinician can use their judgement to handle such borderline cases, it illustrates the challenge they present for any automated system. By relaxing or removing the rules for each defect category, these so-called boundary defects will no longer pose a problem, and we may even be able to detect preliminary or early-stage defects that would normally be overlooked. In such a system, every patient would receive a score for every category, so relaxing the qualifying criteria would likely require a change to our probability templates to ensure that we still correctly identify any defects that are actually present and while minimizing the number of potential false positives. Since many clinicians are not used to looking for such early stage defects, this extension would also require additional collaboration with the vision science community to

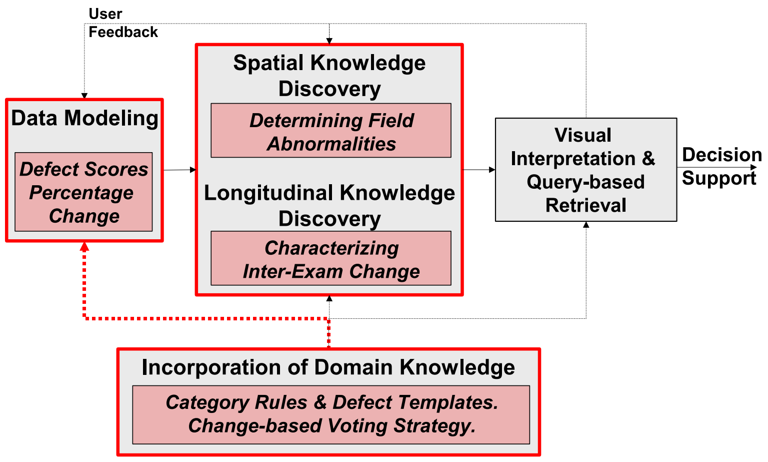

User Feedback
Data Modeling
Defect Scores Percentage Change
Spatial Knowledge Discovery
Determining Field Abnormalities
Longitudinal Knowledge Discovery
Characterizing Inter-Exam Change
Visual Interpretation & Query-based Retrieval
Decision Support
Incorporation of Domain Knowledge
Category Rules & Defect Templates. Change-based Voting Strategy.

Figure 5.11: Diagram of workflow as implemented in the final case study.

address any questions or concerns. Even so, it remains a promising avenue of future work.

As with our other case studies, we provide a graphical summary of our workflow and highlight the implemented stages in Figure 5.11. Unlike the previous studies, however, the methods used in the Data Modeling stage are not true modeling techniques like Zernike polynomials or wavelets. In this case study, we are dealing with functional information, in the form of sensitivity values, as opposed to the structural information used in our previous efforts. As a result, many of the traditional modeling methods do not apply. We rely on simpler techniques such as computing summary scores or calculating percentage change, as these measures provide an adequate characterization of both patient function and functional change. Since the purpose of this case study is to characterize and detect visual field abnormalities, it is heavily dependent on the source domain and the Incorporation of Domain Knowledge. This occurs in creating and coding the rules for each defect category as well as the corresponding defect templates. In addition, the source domain also influences the voting strategy used to characterize the inter-exam change of a patient's field. In terms of Biomedical Knowledge Discovery, this case study contains both a spatial and a longitudinal aspect. The spatial component arises in the determination of individual field defects while the longitudinal element is found in the characterization of inter-exam change. Also, while we use probability maps to visualize a patient's field and any defects that may be present, these maps are fairly standard within the domain of vision science. Thus, this case study does not include an implementation of the Visual Interpretation and Query-based Retrieval stage.

CHAPTER 6

EXTENSION I: QUERY-BASED RETRIEVAL OF PROTEIN SEQUENCE AND STRUCTURE

In recent years, bioinformatics, or the use of methods from information technology on large-scale biological datasets, has become a major showcase for the real-world application of many ideas from computer science. Two areas of research within the field of bioinformatics involve genomic, or gene-based and proteomic, or protein-derived, data. Genes, which encode an individual's DNA, are comprised of nucleic acids, while proteins, which play a crucial functional role in countless cellular processes, are composed of amino acids. Since certain patterns in DNA can be an indicator of disease, much of focus on genes has been on their sequence. Due to their functional significance, there is interest in both the structure and sequence of proteins. The amino acid sequence of a protein is referred to as its primary structure. Sequential acids fold into small sub-domains, which are called secondary structure elements (SSEs). These secondary structures interact to form the global, tertiary structure of the protein.

The past decade has seen an explosion in the amount of publicly-available genomic and proteomic information. Low-cost shotgun sequencing has led to the determination of entire genomes, including cat, dog and human. The completion of these projects has generated vast repositories of sequence information. Scientists believe that these sequences hold the key to determining the cause and treatment of many diseases. The subsequent desire to mine this information has led to the development of query and search tools like BLAST and its variants [90, 91].

The protein domain has seen a similar, though smaller, increase in information. For the past decade, the UniProt[6] database, a repository of non-redundant protein sequences, has doubled in size every two years, to a current total of 2.8 million entries. Due to their functional significance, as well as the belief that there is a link between structure and function, there is also considerable interest in the structure of proteins. Solving the structure of a protein molecule, typically through techniques such as X-ray crystallography or Nuclear Magnetic Resonance (NMR), is much more difficult than determining its sequence. Consequently, the number of solved protein structures trails that of determined sequences.

In most cases, once a protein structure has been solved, it is deposited into the Protein Data Bank (PDB)[7]. As of May 2006, the PDB contained over 36,000 structures. While this number is far lower than the number of known sequences, in the past ten years, the size of the PDB has increased seven-fold. The expected development of high-throughput crystallization techniques means that the number of solved structures should increase dramatically. In addition, projects such as Folding@Home[8] are looking to simulate the folding of a protein as it progresses from an unfolded amino acid sequence to its final structure [14]. These simulations are expected to generate a tremendous amount of data and a corresponding increase in intermediate structures that will need to be examined.

[6] http://pir.uniprot.org

[7] http://www.rcsb.org

[8] http://folding.stanford.edu

6.1 Related Work

Our second protein-related case study involves research efforts that focus on the protein retrieval and the querying of protein datasets [92, 93]. In the past, there has been a large effort in the database community to develop tools to search for similarity in protein databases. One of the first works proposed an algebra to handle queries to determine similarity among protein sequences [94,95]. While useful on sequences, it is not immediately apparent how to extend such an algebra to structure-based datasets. For instance, sequence queries allow for partial matches that include gaps. There is a clear notion on the meaning of a gap in terms of sequence, but not when applied to a 3D structure. Çamoğlu, Kahveci and Singh proposed a method of finding similarity that created feature vectors based on triplets derived from SSEs [96]. These vectors were placed into an R^*-tree, which was used to retrieve similar proteins as a pruning step for the VAST alignment algorithm [97], greatly reducing the time needed to conduct a pair-wise alignment of small datasets. However, pair-wise alignment is not feasible on large databases. Finally, Tan and Tung proposed a method to convert a protein into a series of 3D substructures. These substructures were clustered and sequential substructures were merged [98]. However, this method again relies on pair-wise alignment and does not scale well to large datasets.

There have been a number of recent publications that have looked at indexing proteins for large-scale structure retrieval [99–104]. Three of the most recent are PSIST [102], ProGreSS [103] and Protdex2 [104]. In PSIST, a feature vector is generated for each protein based on the distances and angles between residues. These vectors are placed into a suffix-tree, which is used as the indexing structure. With ProGreSS, a sliding window is placed on the backbone and several structure and sequenced-based features are extracted. Retrieval is handled using a maximum bounding rectangle (MBR)-based index structure. Protdex2, on the other hand, converts the distances between residues into a matrix and uses SSE-based sub-matrices to derive a feature vector, including information such as angle, area and amino acid type. Queries are then executed on an inverted file index. Despite these efforts, however, no one indexing method stands out among the rest.

6.2 Modeling Protein Structure

In the following section, we present several methods that can be used to represent protein structure using wavelets. We begin with a transformation to generate a global description of protein structure using a 2D decomposition. We follow with a technique to represent local substructures using the 1D models described in the previous chapter.

6.2.1 2D Global Descriptor

In this section, we describe a method to create a global representation of protein structure using a 2D wavelet decomposition. Unlike the 1D transformation described above, this method is intended to provide a more high-level structural description. The first step in generating this representation involves converting the protein structure into a distance matrix. Figure 6.1(a) provides a graphical depiction of this matrix (for protein 1HLB), with higher elevations, or larger distances, having a lighter color. The image in Figure 6.1(a) represents a pixelated version of the distance matrix. In contrast to this discretized approach, or the related binary 'contact map' representation, where distances below a certain threshold are set to 1 (0 otherwise), we operate on the actual distance values.

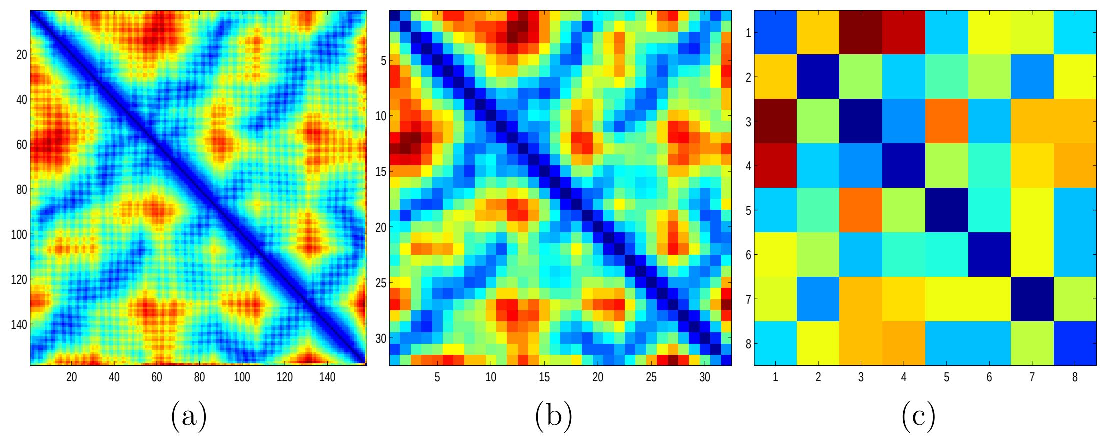


Figure 6.1: Left (a): Raw distance matrix for protein 1HLB. Larger distances a denoted with a lighter color. Middle (b), Right (c): Example 2D decompositions for protein 1HLB. The figure on the left (b) represents a 2nd level Haar decomposition. The image on the right (c) corresponds to a Haar decomposition of level 4. The distance matrix on the left is of size 158x158. The figure in the middle contains 1024 coefficients (32x32), while the one on the right is composed of just 64 (8x8).

To create our global structure representation, we apply a 2D decomposition to the distance matrix. In order to use the results of this transformation as a feature vector, the final number of coefficients must be the same for every protein. This can be achieved either by normalizing the size of the input (the distance matrix), or the output (the approximation coefficients). It is not immediately clear how to normalize a variable-size coefficient matrix while still preserving the necessary spatial correlations. Thus, we elect to normalize the input signal, fixing the size of the distance matrix at 128x128. This normalization occurs either through interpolation or extrapolation, depending on whether the input protein is shorter or longer than 128 residues, respectively. We choose a value of 128 for our signal length because the wavelet decomposition requires that the total length be divisible by 2^L. Since the optimal value for L is unknown, using a length that is a power of 2 gives us the greatest flexibility in choosing an appropriate decomposition level. Using the next higher power of 2, 256, is also an option, but most of the proteins in our datasets are shorter than 256 residues. We prefer to interpolate and smooth or average the excess points, over extrapolation, where we would be forced to generate additional data. Thus our choice of 128 for our normalized signal length.

We perform a multi-level 2D decomposition on this normalized matrix and use the final level of approximation coefficients as our feature vector. There are a fairly large number of wavelet families that can serve as filters. We tested several, but found that the simplest, the Haar, worked well for our purposes. Examples of the wavelet decomposition for protein 1HLB can be seen in Figure 6.1. The figure on the left illustrates the original distance matrix, while the figures in the middle and on the right correspond to the approximation coefficients produced from a 2nd and 4th level decomposition, respectively. We only focus on the approximation values produced by the final level of the wavelet decomposition, so as the decomposition level increases, the number of coefficients decreases by a factor of 4. Despite this large reduction in data, important features such as secondary structures are still present in the figures. Since the matrix is symmetric across the diagonal, we only need to keep the coefficients in the upper (or lower) triangle, plus those that fall on the diagonal itself.

To summarize, the steps taken to create our global representation are as follows:

1. Compute distance matrix using pair-wise distance between C_α atoms.

2. Normalize matrix to size 128 x 128.

3. Apply 2D wavelet decomposition.

4. Extract final level approximation coefficients from the upper half of the matrix plus those that fall on the diagonal.

We calculate the approximation coefficients for several different levels, from 2-8, to see how performance, in terms of accuracy, varies as the dimensionality of the data is reduced. Again, for a given 2D signal, as the level of the wavelet decomposition is increased, the final number of approximation coefficients is reduced by a factor of 4. If this level is set too low, the decomposition will result in a large feature vector, making tasks such as classification difficult. Setting it too high will leave too few attributes, leading to a large amount of information loss and the discarding of potentially-distinguishing information. Both are situations we wish to avoid. Of all the decomposition levels that we tested, we found that the 4th level provided the best performance. Thus, to construct our *global descriptor*, we take those coefficients in the upper triangle of the matrix plus those that fall on the diagonal, which, given a 4th level decomposition, leaves us with a total of 36 coefficients per structure.

6.2.2 Local Substructure Descriptor

Typical protein retrieval methods base similarity on overall structure. Indeed, the techniques presented thus far are designed for that very purpose. However, there may be cases where a researcher is more interested in similarity based on small pieces of a protein - a particular subsequence, for instance. To that effect, we propose an alternate approach that uses a windowing strategy and two 1D wavelet decompositions to generate substructure-based feature vectors.

Figure 6.2 shows a distance matrix for a hypothetical protein sequence. The letters correspond to the secondary structure assignment of each residue (H - helix,

B - sheet, $-$ - no assignment). If we were interested in a seven residue sequence with SSE values of "BBBHHHH," we would only be concerned with the distances contained between the thick black lines. Everything in gray can be ignored. In this manner, every subsequence corresponds to a windowed distance matrix.

While one can manually select a particular query subsequence, to construct subsequences on a target dataset, we need to employ a sliding window strategy. Given a subsequence of length n, for each protein, we start at the first residue, and if the first n residues have the same SSE values as the query, we create a windowed distance matrix for those residues. If not, we move to the next residue of the sequence and repeat the process. We only generate distance matrices for the matching subsequences in order to limit the number of target features that must be generated for each query. This pruning strategy can be disabled to allow for approximate subsequence matches, however.

There is no straightforward way to apply a 2D transformation to a windowed, non-square distance matrix like the one in Figure 6.2. Thus, we elect to use our two 1D wavelet representations, which were designed to capture the interactions between secondary structures.

When we apply these transformations to each windowed distance matrix, we simply ignore the values that fall outside the window. Since the signals generated by this process will be of varying length, we use symmetric extension to ensure that they are divisible by 32. From there, we apply a 5th-level 1D Haar decomposition to each signal and reduce the final level of approximation coefficients to 10 values using interpolation. We concatenate the parallel and anti-parallel signals and treat the resulting 20 attributes as our *local descriptor*.

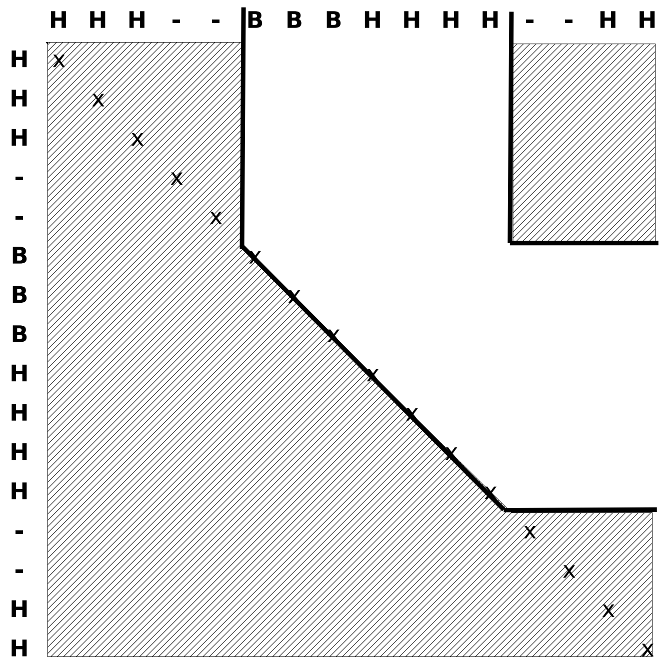

H H H - - B B B H H H H - - H H
H H H - - B B B H H H H - - H H
x

Figure 6.2: Illustration of distance values selected using windowing strategy. Hatched matrix values are ignored.

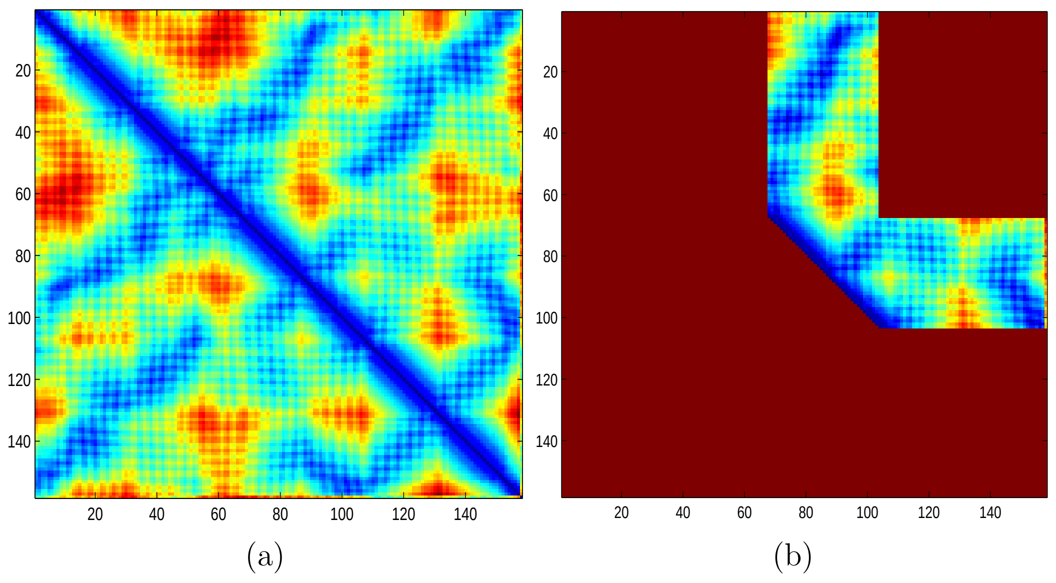


Figure 6.3: Left (a): Original distance matrix. Right (b): Windowed distance matrix.

To summarize the local transformation process:

1. Select query sequence.
2. Generate windowed distance matrices.
3. Convert each 2D windowed matrix to 1D parallel and anti-parallel signals.
4. Apply a 1D wavelet decomposition to each signal, extending them if they are not divisible by 32.
5. Interpolate last level of approximation coefficients to 10 values.
6. Concatenate parallel and anti-parallel coefficients to create local substructure representation.

We provide a graphical representation of the transformation process used to create the local descriptors in Figures 6.3 and 4.10. Figure 6.3(a) represents the original distance matrix for protein 1HLB while Figure 6.3(b) shows the same matrix after it has been "windowed." This window represents a sequence of 36 residues selected from the middle of the protein for illustrative purposes. The results of our parallel and anti-parallel conversions can be seen in Figure 4.10 (top and bottom, respectively). The left images contain the results of the 5th level Haar decomposition, while the rightmost images depict the final, reduced descriptors.

6.3 Modeling Protein Sequence

Here we provide an overview of the method used to create our wavelet-based sequence representation. Since it is based on the results of the PSI-BLAST alignment algorithm, we begin with a brief discussion of PSI-BLAST and follow with the steps used to normalize the results. We conclude with specific details of the wavelet transformation.

Generation of PSI-BLAST Profiles: PSI-BLAST is one of the more popular methods used to determine the similarity of a protein to a database of sequences. This similarity is returned in the form of a profile, or scoring matrix. In PSI-BLAST, a sequence is tested against a database to identify conserved patterns, or motifs. For each position in each of these conserved regions, the algorithm computes a score for each amino acid type. In highly conserved regions, those amino acids that are highly conserved receive a high positive score, while the others receive high negatives. In weakly conserved regions, residues receive scores near zero. These scores are calculated based on amino acid frequency information. Evolution-based substitution matrices such as BLOSUM [105] can also be used when calculating the scores. This process can be iterative, running a profile against the database to refine the results. The final output of the algorithm is a profile that includes a position-specific scoring matrix (PSSM) and a position-specific amino acid frequency matrix (PSFM), as well as a sequence of Z-scores or E-values that denote the statistical significance of the alignment. When discussing both the PSFM and PSSM matrices, we will refer to them as the PSI-BLAST profile matrices (PBPM).

The first step in creating our sequence-based representation involves generating a PSI-BLAST profile for each protein. Using version 2.2.13 of the algorithm, we compute a multi-way alignment against the non-redundant (nr) protein database. Downloaded in March 2006, this database contains almost 3.5 million sequences. We run PSI-BLAST for five iterations with the ϵ parameter set to 0.001. Since our representation is derived partly from the position-specific frequency matrix (PSFM),

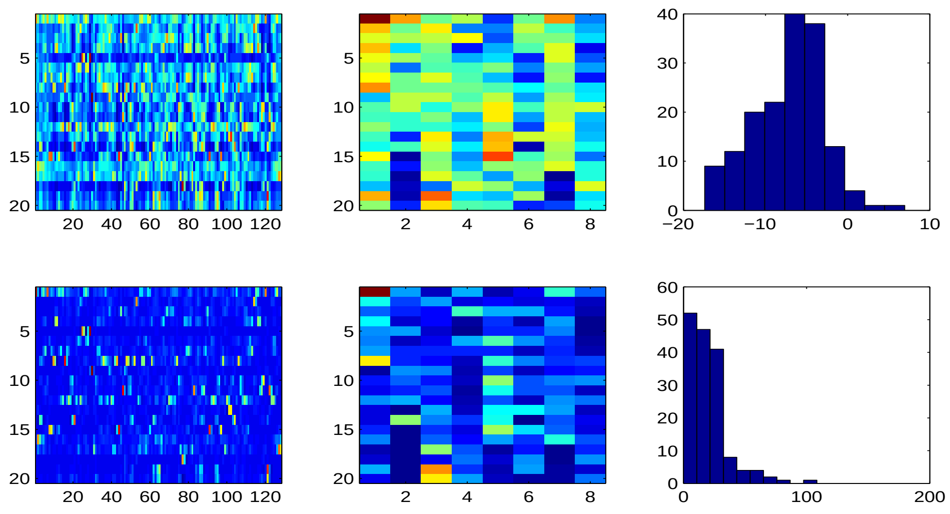


Figure 6.4: Graphical illustration of the PSI-BLAST profile and corresponding wavelet-based summaries for protein 1CCR. The images on the top correspond to the PSSM matrix, while those on the bottom reflect the PSFM values. The matrices on the left show the original profiles while the ones in the middle correspond to the wavelet representations. The graphs on the right show the matrix values in histogram form. The values are divided into bins, the midpoints of which are listed on the x-axis. The frequency count of each bin is provided on the y-axis.

we set the program to output both that and the position-specific scoring matrix (PSSM), in addition to the standard alignment information. We provide a graphical illustration of the matrices for protein 1CCR in the left-most plots of Figure 6.4. Each row corresponds to an amino acid, while each column represents a position in the query sequence. The PSSM representation is given on the top, while the PSFM version lies on the bottom. These images are false color representations, so lower values have darker colors, while higher values appear lighter. Once the profile has been generated, we create a summary of the frequency information by applying a 1D wavelet decomposition to the PSFM matrix. We create a similar summary of the profile scores from the PSSM matrix.

Profile Normalization: Before we can apply the wavelet decomposition, however, we must normalize the size of the matrix to ensure that we are left with the same number of coefficients for each protein. This will allow us freely compare between the resulting feature vectors. The PSFM and PSSM matrices are of size nx20, where n refers to the number of amino acids in the protein sequence and 20 corresponds to one of the different amino acid types. We fix n to be 128 and normalize each PBPM matrix to 128x20. The normalization occurs either through interpolation or extrapolation, depending on whether the input protein is shorter or longer than 128 residues, respectively. We choose a value of 128 for our signal length because the wavelet decomposition requires that the total length be divisible by 2^L. Since the optimal value for L is unknown, using a length that is a power of 2 gives us the greatest flexibility in choosing an appropriate decomposition level. Using the next higher power of 2, 256, is also an option, but most of the proteins in our dataset are shorter than 256 residues. We prefer to interpolate and smooth or average the excess points, over extrapolation, where we would be forced to generate additional data. Thus our choice of 128 for our normalized signal length. We provide a more thorough examination of signal length and decomposition level in Section 6.7.3.

Once the matrix has been normalized we transpose it (20 rows x 128 columns) and apply a 4th level Haar 1D decomposition to each row. We only use the final level of approximation coefficients, so each decomposition will produce 1 coefficient for every 16 input values (2^4), or 8 coefficients per row/amino acid. This results in feature vectors of size 160. These representations can also be combined together into a single profile summary of 320 attributes. An example of the wavelet-based representations can be seen in the middle of Figure 6.4.

Transformation Details: The wavelet decomposition creates a frequency signature for each amino acid. Using a 4th level Haar decomposition results in 8 coefficients. The Haar wavelet filter is an averaging filter, so each of these coefficients will represent the average frequency value for 12.5% of the sequence. One could vary the decomposition or the normalization factor to change the number of coefficients per amino acid, which would change the representation percentage, but a tradeoff must be made between the total number of coefficients and the overall accuracy of the representation. As shown in Figure 6.5, we varied the decomposition level from 2-7 and found that a 4th level decomposition gave us the best overall results.

Since the Haar wavelet filter is orthonormal, each coefficient represents a non-overlapping segment of the protein sequence. If one wanted to create a representation that includes such an overlap, this could be done by manually calculating the average using a sliding window, or through other techniques such as n-grams. If n were set to 16, one would start at position one, and then compute an n-gram for the first 16

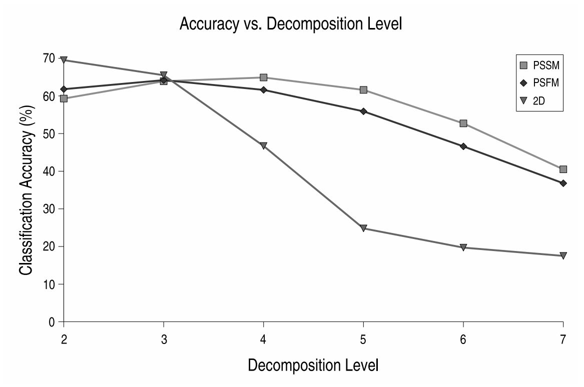


Figure 6.5: Classification accuracy by decomposition level for the different wavelet summaries (*2D* corresponds to the structure-based summary). Values reported represent the accuracy of a SVM trained using 10-fold CV on the 63_Fold dataset (see Section 6.5).

points, slide to the right one position and compute a second n-gram. This process would repeat down the profile. Such an approach would increase the total number of coefficients, however. We use wavelets instead of manually computing the summary because of their speed and relative ease of implementation.

Many SVM kernels that are based on the results of a PSI-BLAST alignment focus solely on values derived from the PSSM matrix [106, 107], though there are approaches that incorporate both matrices [108]. Our own testing on the matter has been inconclusive. As we show in Section 6.7, it is not entirely clear whether either representation is preferable over the other, though in most cases, they can be combined together for superior performance.

We surmise that the PSFM summary outperforms the PSSM-based measure in certain cases because the entries in the scoring matrix represent log-odd ratios, which scales the data and removes some of the variance that may help distinguish between the different groups. As an example, for one of our datasets, all of the PSSM values fall between -7 and 13. The PSFM matrices, on the other hand, contain values between 0 and 100. This information is provided in the histograms on the left of Figure 6.4. The wider variance of the PSFM matrix is at times preferable to the tighter range of the PSSM values. This is analogous to often preferring the covariance matrix over correlation for EM-based missing value analysis [109].

A summary workflow for our global wavelet-based transformations is presented in Figure 6.6. To create a global structural descriptor for a protein, we transform the 3D coordinates into a 2D distance matrix that is then normalized. After normalization, we decompose the matrix using a 2D wavelet decomposition. We extract the final level of approximation coefficients, which serve as our feature vector. To create the sequence-based representations for a given protein, we begin by generating a PSI-BLAST alignment profile. We extract the frequency and scoring matrices and normalize them. A 1D wavelet decomposition is applied to each matrix and we again extract the final level of approximation coefficients. As with the structural descriptor, these coefficients serve as our sequence-based feature vectors.

6.4 Global Hybrid Representation

To create the hybrid variant of our global representations, we simply concatenate the PBPM-summaries to the end of the structural descriptors. This results in an overall feature vector with 456 attributes that characterizes both the sequence and structure of a protein in a concise, normalized manner. One can also create a hybrid version of the individual sequence summaries. In these cases, we concatenate the

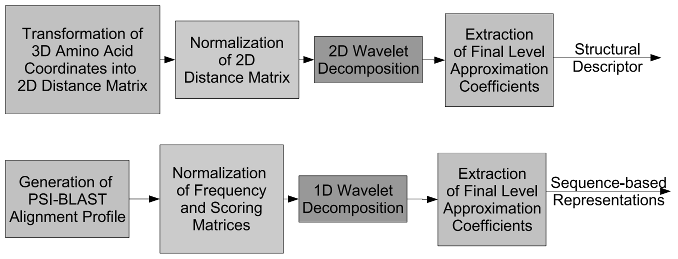

Transformation of 3D Amino Acid Coordinates into 2D Distance Matrix
Normalization of 2D Distance Matrix
2D Wavelet Decomposition
Extraction of Final Level Approximation Coefficients
Structural Descriptor
Generation of PSI-BLAST Alignment Profile
Normalization of Frequency and Scoring Matrices
1D Wavelet Decomposition
Extraction of Final Level Approximation Coefficients
Sequence-based Representations

Figure 6.6: Workflow for the structural and sequence-based transformations.

structure descriptors to the end of the PSSM and PSFM-based representations. Each of these feature vectors contains a total of 296 attributes.

6.5 Datasets

In this section we review the different protein datasets used in our experiments. There is some overlap between the different proteins in the datasets. The feature vectors are constructed using different properties, however. The sequence-based dataset only contains information that can be derived directly from the amino acid sequence. The structure-based dataset, on the other hand, used 3D information based on the solved structure of the protein.

6.5.1 Sequence-Based

The dataset used in these sequence-based experiments is a subset of the ASTRAL database [110]. ASTRAL is a repository derived from SCOP that filters proteins based on their shared sequence similarity. One version of the ASTRAL database consists of proteins with less than 40% sequence identity, the other with those that share less than 95%. Both versions have been examined in previous classification experiments [106, 107, 111], but we choose the version containing proteins with less than 40% sequence identity, as we feel that poses a greater classification challenge. Using version 1.65 of the ASTRAL database, we take all the proteins that share less than 40% sequence identity, which results in a set of approximately 5600 proteins. From there, we select all the proteins that belong to Folds with at least 20 members, removing the rest. After this pruning step, we are left with a total of 2915 proteins that belong to 63 different Folds. This dataset, which we call the *63_Fold* dataset, is essentially the same as the one used in the SVM experiments of Han et al. [111].

6.5.2 Structure-Based

To evaluate our wavelet-based structural representations, we conduct experiments on three different datasets, comparing our results with two of the more recently-published techniques [102, 104]. In order to make a valid comparison, we try to use the exact datasets tested in those works. Due to discrepancies or errors in the original data files, there are some differences, which we note below.

The first two datasets were used by Aung and Tan in their Protdex2 experiments [104]. The first of these datasets contains 200 proteins taken from version 1.59 of the SCOP database that have less than 40% sequence homology (similarity) to each other. We prefer to test proteins with a low sequence similarity because proteins with a high sequence similarity are likely to be almost identical structurally and would thus be a poor choice for retrieval experiments. 10 proteins belonging to the *Globins* family and 10 to the *Serine/Threonin Kinases* were selected. These 20 proteins comprise a set of queries to be tested against 180 proteins randomly selected from the other families within SCOP. In trying to establish a link between the dataset

Dataset	Class	Fold	SuperFamily	Family
2K	4	133	181	281
33K	7	623	960	1632

Table 6.1: Number of unique groups for the 2K and 33K datasets at different levels of the SCOP hierarchy.

ID and the SCOP ID, we were unable to match the names of 11 of these proteins, so our variant, which we refer to as the *Small* set, contains 189 members. We use this dataset for our substructure matching experiments. The second dataset, which we call the *33K* set, originally contained 34,055 members. 108 proteins belonging to 108 medium-sized families (having ≥ 40 members and ≤ 180 members) were chosen as representative queries. Our version of this dataset contained 33,010 proteins and 107 queries.

The final dataset has been examined in a number of publications [96, 103], most recently by Gao and Zaki [102]. Again taken from the SCOP database (though a different version than the one used in constructing the Small and 33K sets), this set contains a total of 1810 proteins, with 10 proteins selected from 181 different SuperFamilies. When running queries on this set, 1 protein was randomly selected from each SuperFamily. As before, when trying to establish a link between the dataset ID and the SCOP ID, we could not match 22 proteins, which leaves us with a total of 1798. Approximately 1600 proteins from this dataset, which we call the *2K* set, are present in the *33K* set. Table 6.1 provides the number of unique groups for the 2K and 33K datasets over the different levels of the SCOP hierarchy.

6.6 Structure-Based Querying of Proteins Using Wavelets

As we mentioned previously, tools such as BLAST are used to determine similarity between DNA sequences. While genomic tools such as BLAST have been adapted to work with protein amino acid sequences, proteome researchers are also interested in determining the *structural* similarity between proteins. Here we present two methods of protein representation that allow for the fast, efficient retrieval of similar structures, based on either global or local features. We begin by computing the pair-wise distance between residues to transform each 3D structure into a 2D distance matrix. To create our global representation, we apply a 2D wavelet decomposition on this matrix to generate a set of approximation coefficients, which serve as our feature vector. Our local representation, which allows us to effectively match proteins based purely on shared substructures, is computed using two separate 1D decompositions. This provides additional functionality that is not available through existing retrieval methods. We evaluate our technique by running classification experiments and nearest-neighbor queries on three previously-examined protein datasets, using labels from the Structural Classification of Proteins database (SCOP) [9], as our basis for comparison. *We find that our global method significantly outperforms the retrieval performance of existing approaches [102, 104], in terms of retrieval accuracy, memory usage and execution time.* Using a k-d tree and running a 10-nearest-neighbor search on a dataset of 33,000 proteins against itself, we see an average accuracy of 89% at the SuperFamily level and an individual query time that is up to 350 times faster than previously published results. We find our retrievals based on local substructure to be highly accurate as well. Finally, while we limit this work to protein data, our techniques can be applied to any structural dataset.

6.6.1 Benefits & Use Cases

Wavelets are used frequently within the vision and image processing community for clustering and retrieval. In many cases, images are retrieved based on a subset of the most dominant coefficients corresponding to some underlying property of the image. While effective in some applications, we cannot rely on such a subset. It is true that part of our rationale for using wavelets is that the secondary structure information will be contained in the dominant coefficients. However, databases like SCOP classify proteins based on the topological arrangement of secondary structures. Many proteins share the same types of secondary structures, and therefore, will have dominant coefficients that are almost identical. Thus, we must also take into account the relative position and location of the secondary structures. Our transformation was designed to capture and retain this topological information. It also has the added benefit of being robust in the presence of noise. If the noise is uniformly distributed, it will be spread throughout the transformation and will not affect the relative values of the results.

Another benefit to using wavelets is that the transformation algorithm is largely parameter-free. Aside from choosing the filter type and initial matrix size, little else is required from the end user. It is true that they must choose which coefficients to use as a feature vector, but it is possible to compute everything at once and make a decision at a later date. Switching from a feature vector composed of 4th level approximation coefficients to one of 5th level values does not affect the overall transformation, only the experimental validation. We feel this is a tremendous advantage over other representations that contain a large number of tunable parameters, all of which affect the computation time and final quality of the overall transformation.

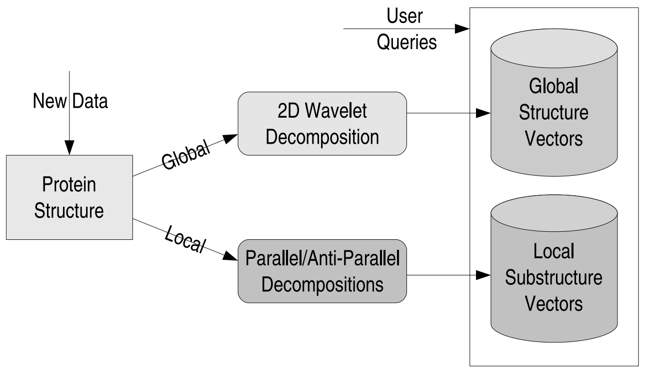


Figure 6.7: Proposed Query Workflow.

Given a database of protein structures, there are several tests that can be used to evaluate the effectiveness of any representation. We evaluate our approach using *k-nearest-neighbor* (KNN) and *range*-based query retrievals [35]. Applied to a target database, a KNN search looks to find the k "nearest" neighbors to a given query. A range query is similar to KNN retrieval, except that instead of providing a value for k, the user specifies a range r and the search returns all the objects less than a distance r from the query. In our evaluation, distance refers to the Euclidean distance between attributes. Allowing the user to specify the range means they have more fine-tuned control over the quality of the results, which can be more effective when the number of potential matches is either quite low or very high. The drawback is that the number of matches is now dependent on the distance between the objects, making the range parameter a function of the dataset and the representation. This dependency occurs regardless of the transformation or model that is chosen.

We envision our transformation techniques being used in a database context, allowing users to submit queries in order to retrieve proteins based on structural similarity. Figure 6.7 provides an illustration of how this system would be implemented. Given a new protein structure, the global descriptors would be generated using a 2D wavelet decomposition. This can be done offline. The local substructure descriptors, generated using the windowed, 1D conversion, can also be generated offline, though it would be impractical to generate local descriptors for every possible subsequence. A more intelligent solution would be to create and store the local descriptors of the more popular subsequences offline and then wait to generate the descriptors of any other subsequence until after a user has submitted a specific substructure query. To avoid the need for continuous regeneration, the descriptors of the more frequent queries can be cached, similar to the strategy employed by BLAST. In either case, once the descriptors have been generated, all of the user interaction would be on the reduced feature vector space, not the original structure data.

6.6.2 Experiments & Results

To evaluate our representations, we conduct experiments on three different datasets, the 2K, 33K and Small dataset, comparing our results with two of the more recently-published techniques [102, 104]. We validate our approach with four different sets of experiments. The first is designed to showcase the predictive ability of our technique by using it to classify selected query proteins according to their SCOP labels. The next two illustrate the robustness of our feature vector by using it, given a query protein, to retrieve a large number of structurally-similar proteins (as defined by SCOP). Our final experiments show how our window-based representation can be used to search for specific substructure sequences. We conclude with a performance analysis in terms of query execution time and overall memory usage.

All experiments were conducted on a 2.4 GHz Pentium 4 PC with 1.5 GB RAM running Debian Linux on a 2.6.9 kernel. We use a *k-d tree* as our target data structure [112]. The k-d tree is a generalization of the binary search tree. Both start with a root node and place the data into left and right subtrees based on the value of a particular attribute, or key. This process is repeated recursively until the data is divided into mutually exclusive subsets. While a binary search tree uses the same key on all nodes of all subtrees, a k-d tree will use a different key at each level. This leads to a data structure that effectively partitions the data but still allows the user to search for best matches without having to make a large number of comparisons. Rather than re-implement this data structure ourselves, we use k-d tree code obtained from the Auton Lab[9].

We also wish to compare our technique with PSIST and Protdex2. We were able to obtain the source code for PSIST[10], but were unable to do so for Protdex2. We were, however, able to download an evaluation version of Protdex2, though it provides limited functionality. As a result, we only use it to generate timing results, which we discuss at the end of this section.

To evaluate PSIST, we generate the input features required by the program and test our datasets. Unfortunately, while we were able to obtain results with the 2K dataset, to process the 33K dataset, the suffix tree generated by PSIST required more memory than was available on our machine (1.5 GB), and we were forced to abort

[9]http://www.autonlab.org

[10]Our thanks to F. Gao and M. Zaki for their assistance.

the tests. It may be possible for PSIST to run on a machine with more memory, but both Protdex2 and our own solution require less than 100 MB of memory to process the same dataset.

The PSIST algorithm allows the user to tune a number of parameters, all of which have effects on both the accuracy and computation time. Using the parameter values reported in the original work [102], we evaluated the performance of the algorithm on the 2K dataset. We found that the default settings yielded an accuracy equivalent to those listed previously as the "best," along with a drastic reduction in running time. While another set of parameter values might provide better performance, we have no intuition to guide their selection. Thus, we report results obtained with the program defaults.

Classification

We examine the predictive accuracy our technique by using the k-d tree as a nearest-neighbor classifier. Given a protein, we retrieve from the target database the three nearest neighbors to the query. We then compare the SCOP labels of the query and the retrieved proteins. If the labels of at least two of the neighbors match, we say that the query has been classified correctly (the query proteins are not included in the target database). Similar classification experiments are detailed in the paper on PSIST [102]. In that work, proteins were only compared at the Class and SuperFamily level. We also examine the accuracy of our method at the Fold and Family levels. As one progresses down the SCOP hierarchy, the protein structures become increasingly similar and the distinctions between them more fine-grained. A truly useful representation should not lose accuracy during this descent, even though the classification problem itself becomes more difficult. We examine the

Dataset	Class	Fold	SuperFamily	Family
2K - 2D	98.3	96.6	96.6	95.5
2K - PSIST	98.3	−−	93.9	−−
33K - 2D	100	92.5	91.5	89

Table 6.2: Classification accuracy for PSIST and the 2D representation on the 2K dataset at different levels of the SCOP hierarchy. Results are provided for the 2D representation on the 33K dataset as well. Accuracy is shown as a percentage of correct classification (total correct out of 181 for the 2K dataset, 107 for the 33K).

effect of our approach on both the 33K and 2K datasets and compare against the results returned by PSIST on the 2K dataset. Unfortunately, as the PSIST code only provides accuracy values for Class and SuperFamily, we do not have results for Fold or Family.

Table 6.2 shows the classification accuracy for our representation on the 2K and 33K datasets at different levels of the SCOP hierarchy. Our technique strategy provides exceptional performance at the Class level, and highly accurate results as one progresses down the hierarchy. Our results on the 2K dataset are equal to those of PSIST at the Class level and almost 3% higher at the SuperFamily level. These accuracies are essentially equivalent, but we do see a greater difference in performance in the nearest-neighbor experiments, which we discuss in the section below.

For the Fold, SuperFamily and Family levels, our performance on the 33K dataset is about 5% lower than the values reported on the 2K data. This is to be somewhat expected, though, as the 33K dataset presents a much tougher classification challenge than the 2K data. At the Class level, we do see a higher accuracy on the 33K dataset than the 2K dataset. This fact is not as odd as it may appear, as the 2K dataset is

Dataset	Queries	$k = 4$	$k = 10$	$k = 50$	$k = 100$
2K - PSIST	181	3.75	7.04	7.50	7.74
2K - 2D	181	3.85	7.74	8.17	8.45
33K - 2D	107	3.75	8.59	29.4	42.5
33K - 2D	33K	3.86	8.93	32.7	51.5

Table 6.3: Average number of proteins in matching SuperFamily for varying values of k as returned by a nearest-neighbor query.

not a complete subset of the 33K dataset and they are evaluated with different sets of query proteins.

There are a number of reasons why we typically expect to see lower classification performance on the 33K data, especially at the lower levels of the SCOP hierarchy. First, the dataset is roughly 18 times larger than the 2K set. Second, there are many more potential categories in the 33K dataset. As illustrated in Table 6.1, at the SuperFamily level, the 2K dataset contains 181 unique groups, one per query. The 33K dataset, on the other hand, contains 960 unique SuperFamilies. The 108 query SuperFamilies represent just 11% of the total. At the Family level, that percentage drops to less than 7.

Nearest-Neighbor Retrieval

For our nearest-neighbor retrieval tests, we again use SCOP labels to determine whether a neighbor is correct. We examine the number of correct matches for k equal to 4, 10, 50 and 100. As before, these values have been used in previous retrieval experiments [102]. We evaluate the 2K and 33K datasets in our tests and compare against the 2K values returned by PSIST.

Dataset		$r = 25$	$r = 50$	$r = 125$
2K	Match	2.87	3.79	5.43
(181 Queries)	Total	2.87	3.79	5.43
33K	Match	5.46	7.8	17.4
(107 Queries)	Total	5.46	7.8	17.4
33K	Match	16.6	28.0	56.2
(33K Queries)	Total	16.6	28.0	56.2

Table 6.4: Average number of objects returned by a range query for different range values (r).

The results of our retrieval tests are presented in Table 6.3. Shown in the table are the average number of nearest-neighbors that belong to the same SuperFamily as the query protein for different values of k. *On the 2K dataset, we return a larger number of correct neighbors as PSIST for all the tested values of k.* For $k = 10$ and larger, we retrieve an additional 0.7 correct neighbors. There is little performance gain for values of k larger than 10, but that is because the target database only contains 10 members for each SuperFamily.

We also report good performance on the 33K dataset. When $k = 10$, the number of correct neighbors is between 8.5 and 9, depending on the query load. As k increases, we continue to return a larger number of nearest neighbors, increasing from around 9 when $k = 10$, to approximately 30 and 40-50 when $k = 50$ and $k = 100$, respectively. Performance tapers off to some degree for large k, partly because many SuperFamilies contain less than 50 members.

Range Queries

We test our approach with a range query for three different range values: 25, 50 and 125. The range is largely dependent on the representation. If set too low, very few objects will be returned. If the chosen value is too high, it is possible to return the entire dataset as a match for each query. These values, determined empirically, were selected because the number of objects returned covers between 25-50% of the possible total. We report the average number of objects retrieved (total number of objects retrieved divided by the total number of queries) as well as the average number of matches among those objects. A match occurs when the object and query belong to the same SuperFamily. The results of this experiment are provided in Table 6.4.

As we can see, range queries are highly accurate, but the number of retrievals is also highly variable. For instance, when the range value is set to 25, we retrieve an average of 2.87 objects on the 2K dataset, but 5.46 or 16.6 on the 33K dataset, depending on the number of queries. However, even as the range value is increased to 125, our accuracy still remains at 100% (average number of matches / average number of retrievals). Considering that the 2K dataset only contains 10 members per SuperFamily, and the 33K dataset contains between 40 and 108, *we are retrieving half of the possible total with no error.*

Substructure Matching

The results presented above illustrate the effectiveness of our technique at matching protein structures based on global similarity. Here we highlight the ability of our approach to retrieve structures based on local similarity. To test our proposed technique, we take the Small dataset and generate windowed distance matrices for

Window Type	SCOP Family ID	Number of Queries	Number in Database	Query Accuracy
HHHHHHHHHH	56113	21	231	100
($F = 112$, $n = 4832$)	46463	46	372	100
BBBBBBBBBB	51534	4	3	75
($F = 45$, $n = 333$)	53851	4	17	100
$--BBBBBB--$	56113	2	6	100
($F = 76$, $n = 160$)	51751	1	9	100

Table 6.5: Membership information for the substructure window datasets. For each window type, the number of different families (F) and size of the target database (n) are given. Each window type has an associated query family, provided for which are the SCOP ID and number of query and database members. The last column lists the query accuracy for the three different substructure windows at the Family level. The values represent percentage of correct classification on a 3-NN classifier.

all subsequences of size 10. For each matrix, we create a local descriptor using the process described in Section 6.2.2.

The Small dataset contains 189 proteins, but there are over 44,000 windows of size 10. We select three different subsequences, *HHHHHHHHHH*, *BBBBBBBBBB* and $--BBBBBB--$ as our queries. These subsequences were chosen because they provide test sets large enough to validate our approach. There are 4899 *HHHHHHHHHH* windows, 341 *BBBBBBBBBB* and 163 $--BBBBBB--$. We select 67 of the *HHHHHHHHHH* windows, 8 of the *BBBBBBBBBB*, and 3 from the $--BBBBBB--$ group to act as queries. The remaining windows serve as a target database. In this manner, a separate database is created for each window type.

Having removed the query windows from the database, we repeat our classification experiments, retrieving 3 nearest neighbors and comparing SCOP labels to determine accuracy. We provide information on the number of members in each Family in the query and target databases in Table 6.5. While we test this method on the entire Small dataset, we envision it being used as a refinement to the results of a nearest-neighbor retrieval. Generating window-based feature vectors on only a subset of the entire dataset would drastically reduce the number of overall features produced.

The results of our substructure classification experiments are presented in right column of Table 6.5. *Our local representation performs very well, misclassifying only one query out of 78.* Despite the fact that the datasets contained a relatively large number of families, we were able to choose the correct one in almost every case. When generating datasets for a given substructure window, there may not be many objects in the database that belong to the same Family as the query. In these cases, a range query may be more appropriate than a strict nearest-neighbor query.

Performance

In this section we provide an analysis of query execution time and memory usage of the tested methods. As stated previously, we were only able to obtain an evaluation version of the Protdex2 code. Therefore, we simply use the software to generate timing results. In addition, while all techniques were evaluated on the same machine, we had to execute the Protdex2 software using Windows. The other two methods were tested under Linux.

Query Execution Time

Here we discuss the running time for our KNN experiments where k equals 100, the longest of all our retrieval tests. We list these values in Table 6.6. The numbers in the *Total Time* column include the time necessary to read both datasets (target and query) into memory, build the *k-d tree*, execute the queries, and write the results

Database Size	Number of Queries	Total Time	Query Time	Time per Query
33K - 2D	33,010	160	135	0.004
33K - 2D	107	8	3	0.028
33K - Protdex2	107	150	–	1.4
1798 - 2D	181	2	< 1	< 0.006
1798 - PSIST	181	107	92	0.51

Table 6.6: Execution time (in seconds) for various database and KNN query workloads ($k = 100$)

to disk. The column labeled *Query Time* shows only the time spent conducting the queries.

To process 181 queries on the 2K dataset, the total query time required by PSIST is 92 seconds, or roughly 0.51 seconds per query. Our wavelet-based approach, in conjunction with a k-d tree, however, requires less than 1 second to process the same queries. This yields an average query time of less than 0.006 seconds, *an improvement of more than 80-fold.* To execute 107 queries on the 33K dataset, Protdex2 takes a total of 150 seconds, or an average of 1.4 seconds per query (the program does not provide a breakdown of the time spent in the query phase). On the same machine, our method requires only 3 seconds to execute these queries, *a 50-fold decrease in running time.* In addition, we see that increasing the number of queries actually reduces the average time spent on an individual query. Querying the entire 33K dataset against itself increases the total number of queries by a factor of 308. The total query time, however, only increases by a factor of 45. Thus, we see a reduction in the average time per query, from 0.03 seconds to just 0.004. *This represents a 350-fold improvement over the average time per query reported by Protdex2.*

Memory Usage

Our representation is much less memory-intensive than PSIST and comparable to Protdex2. On the 2K dataset, PSIST requires roughly 80MB of memory, compared with around 12 MB for our technique. With 33,000 proteins, PSIST requests 1.6 GB of memory, maxing out our test machine and forcing us to abort the experiments. Protdex2 requires around 20MB of memory to execute a single query. Our wavelet representation, coupled with the k-d tree, requires just 80 MB to load the dataset and process 33K queries. The 33K dataset is roughly 18 times larger than the 2K set. PSIST requests 20 times the memory, which would imply a linear relationship between the size of the dataset and the memory required to process it. Our approach on the other hand, shows sublinear growth. *An 18-fold increase in the size of the dataset leads to just a 7-fold increase in memory usage.* Figure 6.8 provides a graphical representation of these relationships. This indicates that our representation is very scalable, and can easily be deployed as a simple, but accurate, system for determining structure similarity.

Another factor to consider is the amount of disk space needed to store the query representations as well as the results. Protdex2 requires that each query be represented using a PDB-style coordinate file (PSIST requires a somewhat similar, though more compact, format). For the 107 query proteins tested, these files requires around 19MB of storage space. To evaluate 33K queries with Protdex2, one would need roughly 6GB to hold all the query information. This is in contrast to our wavelet-based approach, which can represent the entire 33K dataset in just a few MB. In addition, Protdex2 generates a score for the similarity of each query to every object in the database. On the 33K dataset, this yields a 2MB scoring summary for each

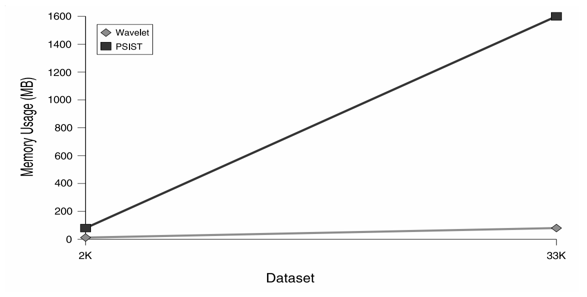


Figure 6.8: Memory usage (in MB) of the 2D wavelet approach versus PSIST on the 2K and 33K datasets.

query. For 107 queries, these summaries require a few hundred MB of storage space. To process 33K queries, on the other hand, one would need 65GB just to store the results. While it is true that storage is relatively inexpensive, managing and processing that much information is not a trivial task. With our approach, we only provide the number of nearest-neighbors requested by the user, drastically reducing the amount of data generated.

6.6.3 Discussion

We present two methods of protein representation that allow for quick and efficient structure queries. We provide implementations for retrieving proteins based on the similarity of either their global shape or smaller substructures, an option that is not provided in any of the leading strategies. When viewed against current techniques, our method is superior in accuracy and more importantly, can answer user queries in a fraction of the time. We have tested our approach by running a number of queries on several different datasets of protein structures. We have validated our results against the labels and categorizations of a leading structural database. We find that even as we progress down the hierarchy of the database, there is not a significant decrease in the accuracy of our method, indicating that the proteins we retrieve are truly correct.

In the future, we plan to test and refine our substructure matching technique. While not immediately obvious, we would like to determine whether one can obtain meaningful results if we allow substructure matching with gaps in the query sequence. In addition, we plan to evaluate the use of space-filling curves [57] as a sampling method to see how they perform compared to our combined parallel/anti-parallel approach.

Thus far, we have focused only on finding similarity among solved protein structures. We do not feel that our technique is limited solely to this task, however. We believe that the methods presented here can be used on a number of different applications, particularly simulations that operate on structure-based data. Two examples include defect tracking in molecular dynamics (MD) simulations and the modeling of protein folding pathways [14, 113]. MD simulations are used to model the behavior of spurious atoms as they move through a lattice of a base material, often silicon. Given a set of simulation frames, we could apply our algorithm to create a sequence describing that set, which could then be compared against other sets. Being able to characterize the behavior of these defects would be a boon to those who work in the manufacture of semiconductors.

It is well-established that the amino acid sequence of a protein determines its final structure. What is unknown, however, is the true nature of the role that the primary structure plays in the folding process. In addition, the intermediate steps that a protein undergoes as it transitions from an unfolded chain to its final formed structure remain a mystery. On occasion, a protein will "misfold," resulting in an abnormal shape. These abnormalities can lead to diseases such as Alzheimer's, Cystic Fibrosis, and Creutzfeldt-Jakob's (the human equivalent of Bovine Spongiform Encephalopathy or Mad Cow). As a result, there has been tremendous effort to understand the protein folding process through computer simulations like Folding@Home. As these programs grow in popularity, they will generate an enormous amount of simulation data. Fast and efficient characterization techniques such as the one proposed here will be crucial if there is to be any hope at processing the results of these simulations in a timely fashion.

Given a set of simulation frames, we can apply our technique on the results from different proteins and cluster them based on similarity. Large clusters may be indicative of certain key points along the folding pathway. Clusters of consistently abnormal structures may give researchers a clue as to how and why proteins misfold. In addition, there may be instances where two groups of proteins share a portion of the pathway and then split before folding into their final shape. Such a characterization could provide insight into the evolution of protein structures and remains a rich direction of future study.

6.7 Classification of Proteins using Sequence and Structure-Based Features

When applying knowledge discovery techniques to the protein domain, one chooses a representation based on the particular task at hand. As we have discussed previously, those representations can be derived from properties based either on sequence or on structure. With structure-based models, one typically focuses tasks such as database retrieval, as we do in the following chapter. With representations derived from sequence-based information, much of the work has involved predicting the final structure of an input sequence. Some of the more popular methods include Hidden Markov Models (HMMs) [114], profile-based alignment algorithms such as PSI-BLAST [91] support vector machine (SVM) classification [106–108, 111]. In the following experiments, we use SVM classification to evaluate the performance of our sequence-based, and to a lesser extent, our structure-based protein representations.

Structure prediction using support vector machines generally falls into two categories. The first consists of those methods that take a protein and compute a representation encompassing a large number of sequence-based properties [111]. Once the properties have been determined, a feature vector is created by calculating the pair-wise similarity between them. The feature vectors are then used to train the SVM. The second form of SVM structure prediction involves the use of specialized kernel functions [106–108]. Kernel functions are designed to compute a high-dimensional similarity between objects in the dataset. A hyperplane can be constructed in this high-dimensional feature space, partitioning the object space. While kernel functions can be used on pair-wise feature vectors generated from several different properties (as in the former), they generally rely on just one. These functions have been shown to be quite effective, but they can be computationally expensive. While some users might be willing to trade speed for accuracy, there is another drawback to these kernels in that they are essentially a "black-box" solution, leading to a loss in the interpretability of both the results and the decision-making process. Moreover, there is no intermediate representation that one can leverage for other purposes such as retrieval. The closest thing to an independent feature vector is a pair-wise matrix that illustrates the relationship between every object in the dataset. As the size of the dataset increases, any attempts at manipulating this matrix quickly becomes infeasible. In addition, any updates to the source dataset requires the recomputation of this matrix.

Even alignment algorithms such as PSI-BLAST cannot be used to directly compare between a large number of proteins. PSI-BLAST alignments are based on the overall length of the protein, which make the widespread comparison of results impossible. Thus, it is imperative to have a normalized, stand-alone sequence-based representation. Such feature vectors could be used by alternative classification algorithms or in applications such as nearest-neighbor or range-based similarity search. Of course, it would also be desirable if this representation could be used for tasks such as SVM-based classification while still providing results that are comparable to the state-of-the-art.

Our sequence-based protein representation was designed to address the challenges listed above. It results in a compact, stand-alone feature vector that can be used in multiple applications. In the following chapter, we evaluate their effectiveness in tasks such as database retrieval. In this section, however, we focus on their performance in SVM-based classification and prediction. We also examine the use of our structure-based model, as well as the hybrid variant that combines the two for instances where both sequence and structure-based information is available.

6.7.1 Datasets

To evaluate the classification performance of our representations, we rely on the 63_Fold dataset. Taking the 63_Fold dataset, we classify proteins using their SCOP labels as our gold standard. We initially focus on the Fold level, but we are also interested in determining the performance of our representations at the other levels of the SCOP hierarchy. If a representation is truly effective, there should be little change in the overall performance as one progresses down the hierarchy. This is despite the fact that the classification challenge itself is increased, with the lower levels containing a larger number of potential classes with fewer members per class.

As it is originally constructed, the 63_Fold dataset is not suited for classification at the lower levels of the SCOP hierarchy. While each Fold was chosen because it contains at least 20 members, there is no such requirement on the underlying Families and SuperFamilies. Some Families, for instance, contain a single member, which makes classification at this level impossible.

Therefore, we construct two additional subsets of our 63_Fold dataset to classify at the SuperFamily and Family levels. For the first subset, we select the proteins belonging to those SuperFamilies that contain more than 10 members. We classify this subset, denoted *SF>10*, at the SuperFamily level. The second subset contains all the proteins that belong to Families with more than 10 members. This subset is classified at the Family level and denoted *Fam>10*. We provide the membership information for these datasets (as well as the original 63_Fold dataset) in Table 6.7. We could classify the SuperFamily subset at the Fold level and the Family subset at the SuperFamily and Fold levels but do not. Classifying the smaller Family subset at the Fold level, for instance, will simply result in improved performance over the more general 63_Fold dataset. The original 63_Fold dataset presents the greatest classification challenge at the Fold level, and the SuperFamily and Family subsets pose the greatest challenges at their respective levels.

Dataset	Proteins	Folds	SFs	Families
63_Fold	2951	63	390	996
SF>10	2116	54	70	538
Fam>10	1014	31	37	55

Table 6.7: Membership information for the 63_Fold dataset as well as the SuperFamily and Family subsets (SF>10 and Fam>10, respectively) at different levels of the SCOP hierarchy (SF = SuperFamily).

6.7.2 Experiments

All of our experiments were conducted on a 2.4 GHz Pentium 4 PC with 1.5 GB RAM running Ubuntu Linux on a 2.6.12 kernel. When referring to our different representations, we use *PSFM* and *PSSM* to denote the PSFM and PSSM sequence summaries, respectively, and *2D* to refer to our 2D structural decomposition. We also combine the structural descriptor with the individual sequence summaries (*2D & PSFM*, *2D & PSSM*). We denote the combined PSSM and PSFM descriptors as *PBPM* and finally, the sequence-structure hybrid as *2D & PBPM*.

For our experiments, we use the SVM provided by WEKA, version 3.4 [115]. WEKA uses Platt's sequential minimal optimization (SMO) algorithm for SVM training [116]. We use a linear kernel with the default complexity parameter (1.0) and train each classifier in a one-vs-rest fashion. As with our database experiments, we test each representation separately, as well as combined into a single feature vector. We report classification performance in terms of the average accuracy, with accuracy values reported as the number of true positives divided by the number of true positives plus the number of true negatives (TP/(TP+TN)), returned as a percentage.

We perform two different classification tests. For the first set of tests, we train an SVM using 10-fold cross-validation and report the average classification accuracy. In these experiments, there is at least one member from every class included in the training data. We view such tests as *recognition* experiments. This is in contrast to our second set of tests, which we refer to as *homology detection*.

In these experiments, we take the SF>10 and Fam>10 datasets and set aside 9 SuperFamilies and 8 Families, respectively. The proteins in these groups are completely untouched during the training phase. We take the remaining members of each dataset and train a classifier one level "up" the hierarchy. In other words, we train an SVM on the SF>10 dataset using Fold labels and the Fam>10 dataset using SuperFamily labels. After the classifier is trained (using 10-fold cross-validation), we test the independent validation sets and if the classifier can identify the correct Fold of the held-out SuperFamilies and the correct SuperFamily for the held-out Families, we state that the classification has been successful. Similar homology tests have been reported elsewhere [106, 108] and are considered to be a more challenging exercise in structure prediction.

The SuperFamilies that were set aside were selected because they belong to Folds with more than one SuperFamily, giving us an opportunity to train a classifier to recognize that Fold. The same process was used for selecting the Families, except that they were required to belong to a SuperFamily with more than one Family. After dividing the SF>10 and Fam>10 datasets using the above criteria, we are left with training sets of 1911 and 877 proteins and testing sets of 203 and 137 proteins, respectively.

Features	63_Fold	SF>10	Fam>10
2D	65.5	67.9	80.9
PSFM	61.6	70.8	84.6
PSSM	64.9	72.4	83.3
2D & PSFM	78.5	81.5	89.3
2D & PSSM	77.4	81.8	89.2
PBPM	68.7	76.4	86.6
2D & PBPM	78.8	82.9	90.4

Table 6.8: Accuracy of recognition experiments at the Fold, SuperFamily and Family levels (63_Fold, SF>10, Fam>10, respectively) for 2D, PSFM, PSSM and combined representations.

Recognition: We present the results of our recognition experiments in Table 6.8. As one can see, Fold-level classification presents the greatest challenge, where we report accuracy values of around 66%, 62% and 65% for the 2D, PSFM and PSSM representations, respectively. At the SuperFamily level, the accuracy for the PSSM and PSFM representations improves to roughly 72%, while the performance of our structure-based representation is essentially constant. We see additional improvement progressing down to the Family level, with the accuracy of the 2D representation increasing to 81% and the PSSM and PSFM summaries to almost 85%. While the PSSM summary outperforms the PSFM representation at the Fold level, that edge is erased at the other levels of the hierarchy. Combining the PSSM and PSFM representations together results in an improvement of approximately 3%.

Table 6.8 also shows the accuracy of our hybrid sequence-structure representation. It is quite surprising how much one can improve the overall performance by combining sequence with structure. *At the Family level, the stand-alone PSFM and PSSM* *summaries yields an accuracy of around 84%. Combined with the structural features, that value increases to 89%, an improvement of almost 6%.* Even though the PBPM representation has an accuracy of 87% at the Family level, that value can be increased to more than 90% by adding structure. One can see even greater increases at the SuperFamily and Fold levels; 10 and 14%, respectively. While structural information would not be available to a researcher trying to predict the global fold of a protein, our results do indicate that the information contained in our 2D representation could be used in a semi-supervised manner to refine the sequence-based classification results, potentially improving the prediction process [117]. In addition, there may be instances where a researcher has access to information on both the sequence and the structure of a protein. In such cases, our methods can be combined to improve the results over any individual representation.

Homology Detection: The results of our homology detection experiments are provided in Table 6.9. While it is unlikely that a researcher would use structural information to detect homologs, we include the values for the sake of completeness, but do not discuss them further. As we see from the results, homology detection is a harder classification challenge, with accuracy values 10-20% lower than those of the recognition experiments. There is little difference between the PSSM and PSFM representations, however. With the combined PBPM representation, we see an increase in Fold-level accuracy, but that gain is erased at the SuperFamily level. Our results are comparable to homology results reported elsewhere on similar, though not identical, datasets [108]. *We see these results without having to resort to a task-specific kernel function or representation, which highlights the general applicability of our approach.*

Features	Fold Detection	SF Detection
2D	55.2	35.7
PSFM	41.8	54.7
PSSM	40.9	54.0
2D & PSFM	59.6	48.1
2D & PSSM	58.1	58.4
PBPM	46.7	53.2
2D & PBPM	59.1	59.1

Table 6.9: Accuracy for homology detection experiments. Values represent accuracy of trained SVM on an untouched test set. Results provided for detection of Fold and SuperFamily (SF)-level homologs.

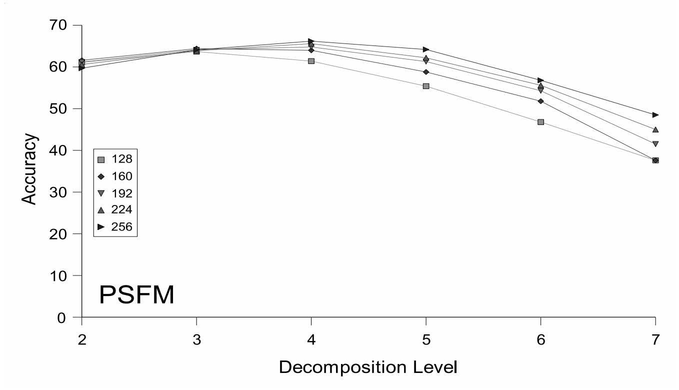
6.7.3 Transformation Parameters

In our original experiments, we elected to normalize the length of each protein to 128 and use a 4th and 3rd level wavelet decomposition for the sequence and structure-based representations, respectively (see Section 4.4.1). As a follow up to that work, we conducted a more robust set of experiments where we evaluate the performance of several different protein lengths (128, 160, 192, 224, 256) over a range of decomposition levels (2-7). However, we do not repeat all of our experiments for every possible combination of length and decomposition level. We only repeat the full set of classification experiments on those combinations that appear most promising. To determine those combinations, we repeat the process detailed in the *Transformation Details* subsections of Section 4.4.1.

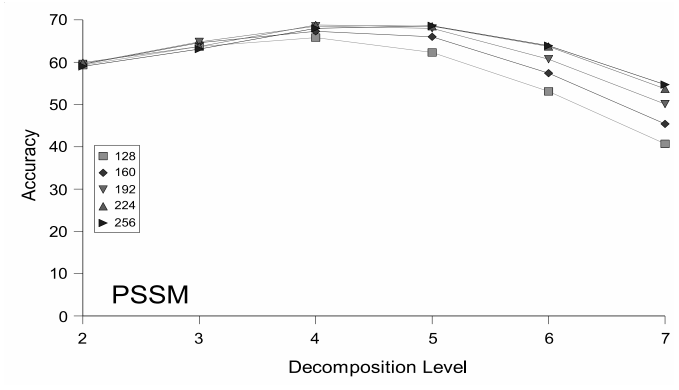
First, for each normalized protein length, we generate feature vectors for the sequence and structure-based representations for every wavelet decomposition level between 2 and 7 (inclusive). This leaves us with a total of 15 feature vectors for each candidate length (3 representations times 5 decomposition levels). We take these feature vectors and use them to train an SVM to classify the 63_Fold dataset with 10-fold cross-validation.

We use the accuracy values from the classification experiments to determine the suitability of each combination of level and length. For the two combinations that we deem the most promising, we generate the additional hybrid representations and repeat our full set of classification experiments to determine how these alternatives compare against our original results. While accuracy is the primary criteria used to just the suitability of of each combination, we also feel it is necessary to take other details, such as the overall number of attributes, into account. The size of each feature vector will affect the classification training time and the amount of storage and memory needed to process our database queries.

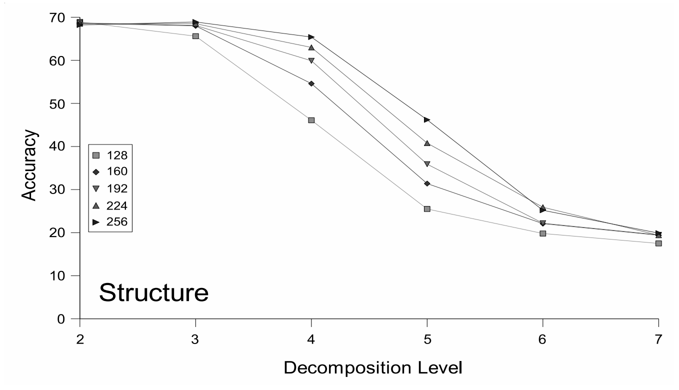
In Figure 6.9, we present the SVM classification accuracies on the 63_Fold dataset for each combination of level and length. The PSFM, PSSM and structural values are presented in Figures 6.9 (a), (b) and (c), respectively. The accuracy values are listed on the y-axis and the decomposition levels on the x-axis. Each line in the figures corresponds to a different length. As one can see, the trends are very similar to those presented in Figure 6.5. The highest values for the sequence-based representations are seen at levels 4 or 5, while the best performance for the structural descriptors occurs at levels 2 or 3. For each representation, the difference in accuracy between the two levels is typically marginal, so when selecting a value for the final decomposition level, we choose the one that will result in a smaller number of total attributes: level 4 for



(a)



(b)



(c)

Figure 6.9: Top (a): SVM classification accuracy of the PSFM transformation for various wavelet decomposition levels and normalized signal lengths on the 63_Fold dataset using 10-fold cross-validation. Each line in the graph represents a different signal length. The accuracy values are provided on the y-axis, the decomposition levels on the x-axis. Middle (b): SVM accuracy values for the PSSM transformation. Bottom (c): SVM accuracy for the structure-based representation.

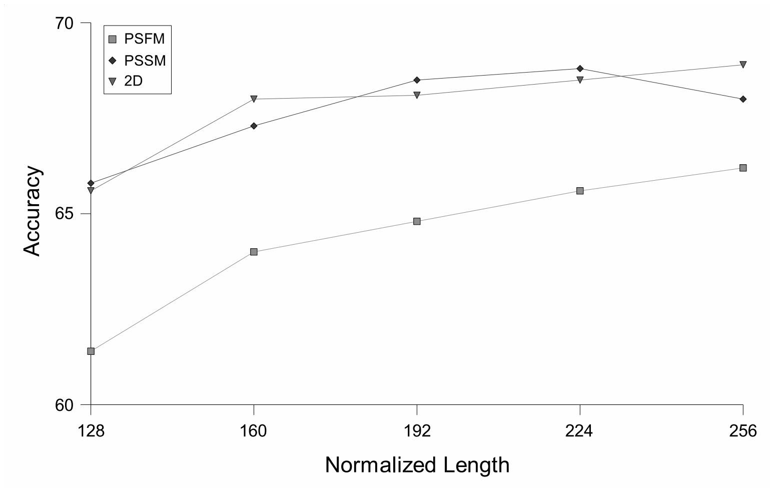


Figure 6.10: SVM classification accuracy using 10-fold cross-validation on the 63 Fold dataset for various signal lengths for each of the different transformations. The lengths are listed on the x-axis, accuracy on the y-axis. For the PSSM and PSFM representations, the decomposition level was set to four. To generate the structure-based features, it was set to three.

the sequence-based representations and 3 for the structure-based descriptor. These were also the decomposition values used in our original experiments.

After determining the decomposition level for each transformation, it becomes necessary to select a normalized length for the representations. While we could use a different length for each individual representation, we feel that in order to make a valid comparison, the same value should be used for all the transformations. Figure 6.10 provides a closer view of the classification accuracy for each length and representation at a single decomposition level (4 for sequence, 3 for structure). Unlike Figure 6.9, here each line in the plot corresponds to a representation while the lengths are listed on the x-axis. The y-axis again provides the accuracy values, though at a different scale than in Figure 6.9. As one can see from the figure, using a longer length generally

Features	$N = 128$	$N = 160$	$N = 224$
2D	136	210	406
PSFM	160	200	280
PSSM	160	200	280
2D & PSFM	296	410	686
2D & PSSM	296	410	686
PBPM	320	400	560
2D & PBPM	456	610	966

Table 6.10: Number of attributes for each representation for the different signal lengths tested (*N=128,160,224*).

results in a higher accuracy, though the difference may be small. Based on the graphs, we elect to repeat our experiments using length values of 160 and 224. We choose 160 as that seems to be the optimal value for the structural descriptor, in terms of accuracy and the overall number of attributes. Our selection of 224 was motivated by the performance of the sequence-based representations at that value. The number of attributes in each representation are listed in Table 6.10. We now repeat our original tests using these new lengths to see how they compare to the representations generated using a length of 128. We begin with a re-evaluation of our *recognition* and *remote homology detection* classification experiments.

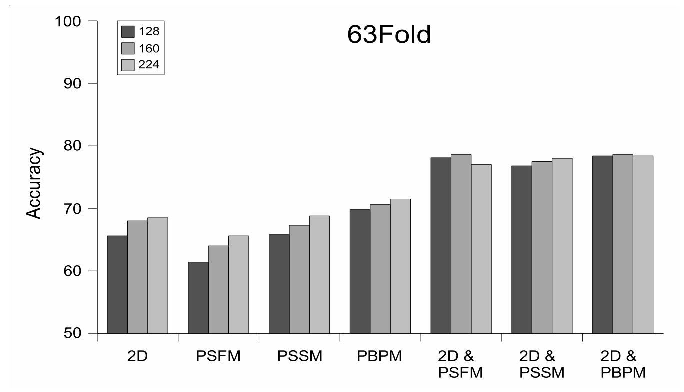
Classification: Figure 6.11 contains the results of our *recognition* experiments, with the Fold, SuperFamily and Family-level results given in Figures 6.11 (a), (b) and (c), respectively. As one can see, with the sequence, structure and combined sequence representations (PSFM, PSSM, 2D, PBPM), a longer length generally results in a higher accuracy. When combining sequence with structure (2D & PSSM/PSSM/PBPM), the reverse trend appears to hold.

While these trends tend to hold across the Fold, SuperFamily and Family levels, we should note that at the Family level, there is little difference in accuracy for the different lengths with the PSSM and PSFM representations. In addition, the PBPM representation present behavior opposite that seen at the Fold and SuperFamily level. The smaller-length representations yield a higher accuracy than those based on the larger values.

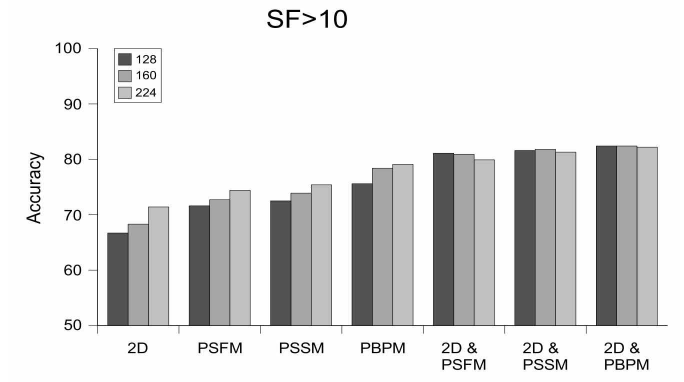
Unfortunately, the experiments in *homology detection* do not produce trends that are as clear-cut as those seen in our *recognition* tests. Figure 6.12 provides the accuracy results for our attempts at homology detection. In general, with Fold Detection (Figure 6.12 (a)), a shorter length yields a higher accuracy. The same holds true at the SuperFamily level (Figure 6.12 (b)) for the 2D and sequence-structure hybrids, but not for the purely sequence-based representations. For those descriptors, a longer length is more effective.

Information Loss: One may have concerns with the amount of information loss that occurs with our transformations. There are two primary areas that contribute to this loss: when normalizing the protein lengths and when applying the wavelet decomposition. Here we attempt to allay some of those concerns by first addressing the latter and then focusing on the former.

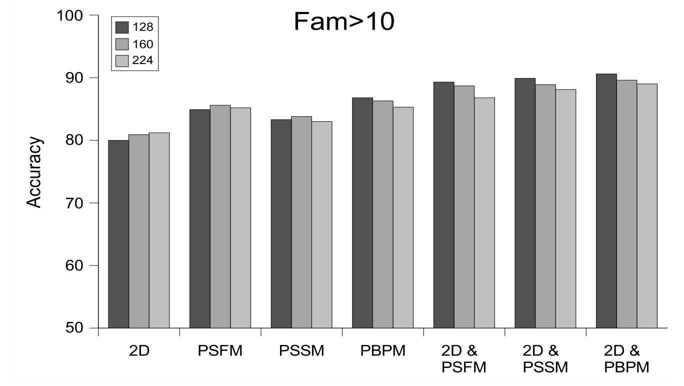
For the sequence-based representations, after normalizing the length of the protein, applying a 4th level wavelet decomposition reduces the total number of attributes or amount of information by a factor of 16 (2^4). With the structural descriptors, the



(a)

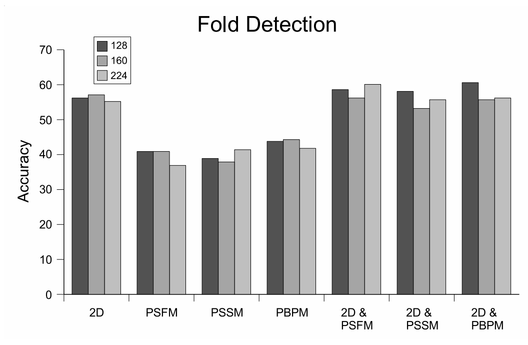


(b)

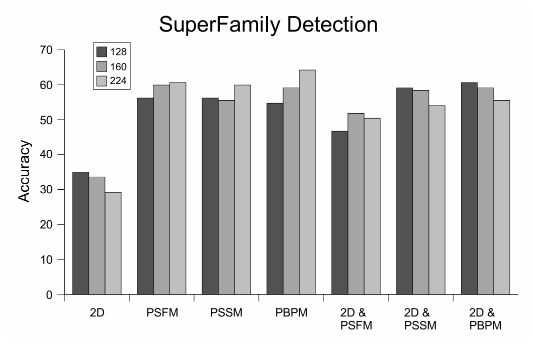


(c)

Figure 6.11: Results of the *recognition* experiments. Left (a): Classification accuracy for the different representations on the 63_Fold dataset (Fold level). Middle (b): Classification accuracy on the $SF > 10$ dataset (SuperFamily level). Right (c): Accuracy for the different representations on the $Fam > 10$ dataset (Family level). All accuracy values were obtained from an SVM trained using 10-fold cross-validation.



(a)



(b)

Figure 6.12: Results for the experiments in *homology detection*. Left (a): Accuracy of the different representations at the Fold level. Right (b): Accuracy of the different representations for at the SuperFamily level.

total amount of information is approximately reduced by a factor of 60. For example, given a normalized matrix of size 128x128, we can immediately remove half the points since it is symmetric across the diagonal. In this case, we would be left with a total of 8192 points. A 3rd level 2D wavelet decomposition yields 136 coefficients, a reduction of approximately 60-fold. This is about 4 times greater than the reduction seen with the sequence-based representations. Since each level of the 2D wavelet decomposition reduces the total number of points by a factor of 4, one could make the information loss between the different methods equal by either reducing the final level of the structural descriptor from 3 to 2, or by increasing the final level for the sequence-based representations from 4 to 6. However, since there is little difference in accuracy between levels 2 and 3 for the structural descriptors (see Figure 6.9 (c)), we do not feel that such a change is necessary.

When normalizing the length of the protein, it would appear that the amount of information loss is much greater for the structural descriptors, since we are reducing an NxN distance matrix to size LxL (where L is the normalization length) compared with the sequence-based profile matrix that is being reduced from Nx20 to Lx20. However, since the distance matrix is computed from the 3D coordinates of the amino acids, we really are reducing the total amount of information from Nx3 (the xyz coordinates of each amino acid) to Lx3. This is similar to the amount of information loss in our sequence-based representation, but we are currently looking at several information-theoretic measures to better characterize these values.

6.8 On the Use of Structure and Sequence-Based Features for Protein Retrieval

Given a protein, the ability to effectively retrieve similar molecules, or to identify the functional group to which it belongs, is needed in many biomedical applications. A key requirement in this process is defining a suitable metric of similarity. Protein molecules are highly complex and a number of different measures can be used to define similarity. One could calculate the geometric similarity between the structure of proteins, for instance, or use a measure based on certain sequence-related properties. In many cases, the application or the available data will dictate the type of similarity required. For instance, if a protein is found to prevent a disease based on a key chemical property, researchers may look for proteins that have a similar sequence in the hope that they can also be used for prevention. On the other hand, if a protein is designed to be part of a drug delivery system, it might also be possible to use proteins with similar structural features to effectively deliver medicinal payloads to sites within the body. In folding simulations, similar structures between proteins could be indicative of a common folding pathway. Structural representations are particularly critical for processing the results of folding simulations, where the sequence of a protein is essentially unchanged.

Previously, we presented a representation that could be used to retrieve proteins using either global or local substructure-based similarity. In an expansion to that work, we present an additional representation that can be used to retrieve proteins using sequence-based similarity. Here we demonstrate how each representation can be employed to quickly and efficiently retrieve molecules using similarity based on sequence, structure or both.

6.8.1 Datasets

As in our previous indexing experiments, to evaluate database performance, we focus on the *2K* dataset. In this set, a total of 181 SuperFamilies were selected from SCOP, with each SuperFamily having at least ten members. Ten proteins were selected from each SuperFamily, resulting in a total dataset size of 1810 proteins. To construct a query set, one protein was selected at random from each of the SuperFamilies, yielding a query subset of 181 proteins.

6.8.2 Experiments

All of our experiments were conducted on a 2.4 GHz Pentium 4 PC with 1.5 GB RAM running Ubuntu Linux on a 2.6.12 kernel. When referring to our different representations, we use *PSFM* and *PSSM* to denote the PSFM and PSSM sequence summaries, respectively, and *2D* to refer to our 2D structural decomposition. We also combine the structural descriptor with the individual sequence summaries (*2D*

& PSFM, *2D & PSSM*). We denote the combined PSSM and PSFM descriptors as *PBPM* and finally, the sequence-structure hybrid as *2D & PBPM*.

To illustrate the effectiveness of our representations as an indexing method, we validate our approach using two different query retrieval tests, nearest-neighbor and range. For the nearest-neighbor tests, given a query, we retrieve the k nearest-neighbors from the target database and report the number of correct matches. For the range queries, we specify a range value r and retrieve from the target database those objects that lie within a distance r to the query. In both tests, distance refers to the standard Euclidean distance.

We use a *k-d tree* as our target data structure [112]. The k-d tree is a generalization of the binary search tree. Both start with a root node and place the data into left and right subtrees based on the value of a particular attribute, or key. This process is recursively repeated until the data is divided into mutually exclusive subsets. While a binary search tree uses the same key on all nodes of all subtrees, a k-d tree will use a different key at each level. This leads to a data structure that effectively partitions the data but still allows the user to search for best matches without having to make a large number of comparisons. For our tests, we use k-d tree code obtained from the Auton Lab.

As in our structure-based query experiments, we also compare our techniques with PSIST. As before, we generate a feature vector for each protein using the default parameter values.

Nearest-Neighbor Retrieval: For our nearest-neighbor retrieval tests, we use the SCOP labels at the SuperFamily level to determine whether a neighbor is correct. We

Features	$k = 4$	$k = 10$	$k = 50$	$k = 100$
PSIST	3.75	7.04	7.50	7.74
2D	3.86	7.71	8.17	8.41
PSFM	3.88	8.07	8.33	8.44
PSSM	3.90	8.14	8.51	8.71
PBPM	3.88	8.12	8.35	8.44
2D & PSFM	3.87	7.85	8.28	8.46
2D & PSSM	3.86	7.72	8.19	8.40
2D & PBPM	3.86	7.85	8.29	8.48

Table 6.11: Average number of proteins in matching SFs for varying k as returned by a NN query on the 2K dataset.

examine the number of correct matches for k equal to 4, 10, 50 and 100. Similar values have been used in previous retrieval experiments [102]. We evaluate our structure and sequence-based representations on the 2K dataset and compare against the values returned by PSIST.

The results of our retrieval tests are presented in Table 6.11. Shown in the table are the average number of nearest-neighbors that belong to the same SuperFamily as the query protein for different values of k. *On the 2K dataset, we see that both our representations return a larger number of correct neighbors than PSIST for all the tested values of k* and that the sequence-based summaries outperform those derived from structure. There is little performance gain for values of k larger than 10, but that is because the target database only contains 10 members for each SuperFamily. We find that PSSM provides the best overall performance, though PSFM is comparable. The structure-based values are slightly worse, and we see no improvement in combining sequence with structure.

Range Queries: We also test our representations with short, medium and long range queries using the 2K dataset. We empirically define a query as short, medium or long based on the number of objects returned. If the range value is too small, very few objects will be returned. Too large, and the query will return the entire database. Since the choice of the actual range value is dependent on the representation, we cannot use the same value of r for every test. Therefore, we list the range values used for each representation along with our results. We report the average number of objects retrieved (total number of objects retrieved divided by the total number of queries) as well as the average number of matches among those objects. A match occurs when the object and query belong to the same SuperFamily. The results of this experiment are provided in Table 6.12.

As we can see, range queries are highly accurate, but the number of retrievals is variable. For instance, for the short range values, we retrieve 3-5 objects with 100% accuracy. Even at the long range values, our accuracy remains very high. We see an accuracy of almost 100% with the the 2D features, and 84-98% with the sequence-based representations. Each SuperFamily in the 2K dataset contains only 10 members. As we from the table, *we can retrieve half of the possible total with no error.*

Finally, while we do not report the values here, we find that our indexing approach is far superior than PSIST in terms of both query execution time and memory usage.

6.8.3 Discussion

In this paper, we present two methods of protein representation, one based on sequence information, the other derived from the solved structure. We find that

Features		*short*	*med.*	*long*
2D	Match	3.71	4.54	5.90
$r = (75, 125, 250)$	Total	3.71	4.54	5.93
PSFM	Match	3.73	5.58	6.96
$r = (25, 75, 110)$	Total	3.73	5.58	8.27
PSSM	Match	3.78	5.33	7.33
$r = (10, 20, 35)$	Total	3.78	5.33	7.88
PBPM	Match	3.69	5.35	6.76
$r = (25, 75, 110)$	Total	3.69	5.35	6.88
2D & PSFM	Match	4.60	5.62	7.20
$r = (150, 250, 450)$	Total	4.60	5.62	7.47
2D & PSSM	Match	3.61	4.52	5.89
$r = (75, 125, 250)$	Total	3.61	4.52	5.92
2D & PBPM	Match	4.56	5.61	7.20
$r = (150, 250, 450)$	Total	4.56	5.61	7.46

Table 6.12: Average number of objects returned by a range query for short, medium and long ranges (*short, med., long*, respectively). *Total* lists the average number of objects returned and *Match* the average that are correct. The range values for each query are listed in parentheses.

the PSSM and PSFM representations provide roughly the same overall performance, though there are instances where the PSSM summary yields a higher accuracy. We also find that even though the PSSM scores are derived from the PSFM values, combining the summaries can result in a fairly dramatic increase in accuracy. We believe this is because the combined PBPM representation merges information about the overall amino acid frequency counts contained in the PSFM summary as well as the PSSM scoring information pertaining to the overall conserved-ness of each region. The PBPM summaries are comparable to our 2D structural descriptor. It is also possible, as evident in our classification experiments, to combine information on both sequence and structure into a *hybrid representation* that can provide an improvement over any individual variant. Such a technique can be useful in applications where the researcher is not limited to sequence or structure and further emphasizes the general applicability of our approach.

CHAPTER 7

EXTENSION II: MODEL-BASED SPATIAL AVERAGING

In this section we present an algorithm that is designed to increase classification accuracy by taking advantage of the spatial properties inherent to many of the modeling methods explored in this work. It entails an ensemble-learning technique that improves classification performance by aggregating the decisions of multiple classifiers built from surface models of varying spatial resolution. We validate this algorithm by applying it to two of our case study domains, protein structure and corneal topography.

One of the benefits of modeling image-based or image-like biomedical data is that it can be modeled using spatial transformations at several possible resolutions. Spatial transformations allow the user to create a model of the original data, capturing the correlations and features that are important in distinguishing between groups of interest. At the same time, they can reduce the overall dimensionality of the data and provide a representation that can serve an a feature vector for tasks such as classification. There are a large number of possible transformations, and choosing which one to use when modeling a dataset is not a trivial task. One can make the process easier by relying on a specific group of transformations and choosing the one that can best represent the underlying properties of the dataset. The challenges do not end with the selection of the transformation, however. One must still decide at what resolution the data should be modeled.

A model of high resolution will result in a more faithful representation of the original data with much finer detail, but will yield a larger number of coefficients and could include a large degree of noise. An increase in the number of coefficients or the amount of noise can lead to a degradation in classification performance. A lower-resolution model, on the other hand, will be much less specific, putting more emphasis on global features and trends. These representations are typically less affected by noise, but can miss out on some of the features and trends that can distinguish between certain problem cases or classes, such as diseased vs. non-diseased.

With classification, users have typically been forced to choose a single resolution or a single model before proceeding. In this work, we present an technique that removes that requirement. This method, which we term *spatial averaging* is designed to aggregate the decisions of classifiers constructed from multiple spatial representations or resolutions of the same data[11].

The Spatial Averaging algorithm can be leveraged with any classification strategy that can take advantage of ensemble-learning methods, though here we focus on decision trees. Spatial averaging can also be used in place of, or even in conjunction with, traditional ensemble-learning approaches. We envision this strategy as a way to improve classification performance without completely sacrificing the underlying interpretability of algorithms such as Naïve Bayes or decision trees. Spatial averaging requires the construction of multiple classifiers, but examining multiple decision trees

[11]The algorithms and results detailed in this section are an expanded version of work presented at the 2005 IEEE Symposium on BioInformatics and BioEngineering (BIBE) [118]

can still be more intuitive than attempting to understand and interpret a "black-box" system such as a neural network or a support vector machine.

We show the utility of our approach by applying it to datasets taken from two different biomedical domains. The first contains data relating to the corneal topography of three different patient groups. This topography is modeled using two polynomial transformations at several different orders (see Chapter 3 for further detail). The second dataset that we examine is a set of several thousand proteins divided into roughly 60 different groups based on their 3D structure. These proteins are represented by two different sets of features, one sequence-based, the other derived from their structure. These features are generated by applying a wavelet decomposition to the original data and treating the final level of approximation coefficients as the representative model (see Chapter 6 for more information).

With this technique, we are not interested in the performance of the original transformations per se. We have applied these transformations to these datasets in the past [41, 93, 118]. Our intent is to show that it is possible to improve classification performance given a set of existing representations. Such an approach can remove some of the uncertainty that results when faced with having to choose a single representation from one of many seemingly-similar options.

7.1 Approach

Our ensemble-learning strategy is based on the idea of averaging the results from classifiers built on multiple resolutions of a single spatial transformation to reach a decision. We refer to this approach as *Spatial Averaging*. Unlike boosting and bagging, which seek to improve classification accuracy by aggregating the results of classifiers constructed from multiple random sub-samples of the dataset after it has been modeled at a single spatial resolution, we aggregate the results of classifiers constructed from *multiple spatial resolutions* of different transformations *derived from a single sample of the dataset.* This strategy was designed to smooth out perturbations due to *noise effects.*

Intuitively, for a given transformation, the lower order resolutions result in a model that is less detailed than those generated from the higher orders. The higher order resolutions, however, contain noise that is not present in the models derived from the transformations of lower order. Thus, assuming that the classification errors are uncorrelated, our belief is that including the additional features will help minimize the misclassifications that result from certain problem cases. We now proceed to list the steps of the Spatial Averaging algorithm.

Generation of Model Representations: First, a model of each individual record in the dataset is created for each transformation at every resolution that is to be included in the Spatial Averaging classifier. Thus, if we wish to create a Spatial Averaging classifier that aggregates the predictions of classifiers constructed using models of 4th-10th order Zernike and pseudo-Zernike polynomials, we would create a total of 14 representations for each member of the dataset (7 Zernike and 7 pseudo-Zernike). This is a pre-processing step that only needs to be completed once, as the models themselves will not change. Once the representations have been generated, the actual classification process can begin.

Dataset Partitioning: The first step is to divide the dataset into N folds, or splits, of approximately equal size, with each dataset member present in one, and only one fold. As stated above, we generate a number of model representations of the original

```
1. Partition the original dataset into $N$ splits.
   Repeat the same partitioning on each of the $r$
   representations that are to be included in the
   Spatial Averaging classifier.
2. For each of the $r$ representations,
   obtain a test prediction for each object.
   (For each $r$, create $N$ classifiers using one
   of the $N$ splits as a held-out test set and
   training the classifier on the remaining splits.)
3.     If using Top K or weighted voting,
       compute the average training accuracy for
       each representation by also obtaining a
       prediction for each object in the training set.
   end for
4. Compile the test predictions of all the
   representations, yielding $r$ predictions
   for each object.
5. Take the modal label of the $r$ predictions
   to determine the final label of each object.
```

Figure 7.1: Algorithm for base version of Spatial Averaging technique.

dataset. In order to create a valid Spatial Averaging classifier, we require that each of the representations be partitioned in the same manner. Thus, given a certain ordering of the dataset, if object X lies in Split 1 of one representation, it will be placed in Split 1 of all the remaining representations. As we will explain later, we also implement versions of our Spatial Averaging algorithm based on bagged classifiers. This requires that that the algorithm be executed a number of times. To ensure that we generate unique splits for each bagging iteration, we randomize the dataset before partitioning. *Classifier Construction:* After we have divided the dataset, we construct a set of initial classifiers. In the experiments described here, we use decision trees, but any classification strategy can be employed. For each representation, we train a total of N classifiers, treating each of the N folds as a held-out test set and using the remaining $(N-1)$ folds for training. After training the classifier, we retrieve a prediction for each object in the *training* set and a separate set of predictions for each object in the *test* set. Since each object in the dataset belongs to one, and only one fold, these N classifiers yield *one* test prediction for each object and $(N-1)$ training predictions. This step is repeated for each representation.

Computation of Training Accuracy: We compile all of the training predictions and use them to compute the average training accuracy for each representation. This step is only needed when prioritizing or ranking the different representations. It is used with the weighted and Top-K voting schemes, which are described below.

Compilation of Test Predictions: The next step in the Spatial Averaging algorithm is to tally all of the predictions in the test sets. For every object in the dataset, we compile the predictions from every representation that is part of the overall Spatial Averaging classifier. Thus, if we are testing 14 representations, we will have a total of 14 predictions for each object. We have also implemented a version of the Spatial Averaging algorithm that relies on the predictions of a bagged classifier. Therefore, we generate a number of predictions for each representation. In these instances, we will end up with a total of $r * m$ predictions for each object, where r is the number of representations and m refers to the number of iterations used to create the underlying bagged classifier.

Voting: Once we have tallied the predictions for each object in the dataset, we count the votes of the test predictions. We have implemented three different versions of the Spatial Averaging classifier. The first is a simple base strategy. This version of the Spatial Averaging classifier tallies a single vote for each representation and the label with the most votes is selected as the overall prediction for that object. If we have representations that belong to more than one transformation (i.e. Zernike and pseudo-Zernike), we aggregate the results of each transformation separately, and also compute a *combined* classifier that aggregates the decisions of all the representations.

Thus, if a dataset is represented by multiple transformations, for instance, both Zernike and pseudo-Zernike models, we compute a separate prediction for both the Zernike and pseudo-Zernike representations, as well as a single prediction for the combined representation. The idea behind the combined classifier is to minimize the overall effects of noise on a dataset by including transformations that are richer and more-detailed, but are more likely to be affected by noise with those that offer a less-detailed model, but are more robust and noise-tolerant. In addition, we give user the ability to average the results of the top K classifiers. If this option is selected, we simply average the predictions of the K classifiers with the highest training accuracy.
Other Variants: The second version of the Spatial Averaging classifier relies upon the predictions of bagged classifiers to reach a final decision. We refer to this version as *Spatial Averaging + Bagging.* Assume that we run the Spatial Averaging algorithm with r representations for a total of m iterations. Thus, for each representation r_i, we are left with a total of m predictions for each object. We take the modal label of those m predictions and use that value as the single prediction for representation r_i. We then tally the prediction of each of the r representations and take the modal label to be the overall prediction for that object.

The final variation creates a number of base Spatial Averaging classifiers and uses the predictions from those classifiers to determine an overall prediction for each object.

We denote this variant as *Bagged Spatial Averaging.* The Bagged Spatial Averaging algorithm can be viewed as follows: assume, as with the Spatial Averaging + Bagging variant, that we execute m iterations of the base Spatial Averaging algorithm (with r representations). Thus, for each iteration m_i, we are left with a total of r predictions for each object. We compute a prediction for each iteration as with the base Spatial Averaging method. We tally the prediction for each of the m iterations and take the modal label to be the overall prediction for that object.

Weighted Voting: The default voting scheme for each Spatial Averaging classifier is that each of the underlying representations receives an equal vote. This method works well when the representations are approximately equal in terms of classification accuracy. There may be cases, however, where certain classifiers significantly outperform the others. When this occurs, we want to weigh the vote of these classifiers much more than the rest. In addition, when there are classifiers that are significantly worse than the average, we would like to give them less emphasis. We do not want to totally ignore their predictions, since they may add potentially useful information, but we want to avoid dragging down the overall results by heavily relying on them.

As a result, we have implemented a heuristic function that assigns a weight to the vote of classifier based on its performance above or below the average accuracy of the other classifiers. The training accuracy of each representation is used to compute the average accuracy. Our heuristic is intended to mimic a geometric function, in that the weight assigned will rapidly decrease as the accuracy of the underlying classifier approaches the average. In this manner, we assign weights much greater than 1 to representations that have classifiers with higher-than-average accuracy. Those representations with an accuracy close to or less than the average will receive a weight between 0 and 1. To compute the weight of a given classifier, we use the following function:

$$\frac{1}{[1-(\mathrm{acc}_i-\mathrm{acc}_{\mathrm{avg}})]^{\frac{1}{(1-\mathrm{acc}_i)}}} \tag{7.1}$$

where acc_i is the training accuracy of a given representation and $\mathrm{acc}_{\mathrm{avg}}$ is the average training accuracy of all the representations. The training accuracy of each representation, defined as (1 - probability of error for the classifier of that representation), is used to compute this average. We tested a number of different weighting strategies, but found that the above function provided the best overall results.

When the training accuracy of a representation is significantly higher than the average, this function will give the vote of that representation a much greater weight than one that is close to the average. We do not view this as a drawback, however, as it rewards those classifiers with excellent performance. In addition, if a classifier is significantly lower than the average, the weight of its vote will be very small. If the accuracy of each classifier is roughly the same, however, the weight of their votes will be approximately equal. To illustrate the effects of this weight function, consider a Spatial Averaging classifier that is constructed from 5 underlying classifiers with the following training accuracies: 25, 25, 50, 75 and 90%. The average training accuracy for these classifiers is 53%. Our weight function would assign the following values to the votes of these classifiers: 0.72, 0.72, 0.94, 2.7 and 101.5, respectively.

Key Intuition: The rationale behind Spatial Averaging is similar to that of bagging. Breiman notes that bagging works by creating an aggregate predictor based on a sequence of underlying predictors formed from several replicates of the training data, or in cases where that those replicates do not exist, by taking a series of random samples from the original training set [38]. By coupling the different samples with an unstable classifier, the goal is to cause perturbations in the predictor that will lead to improvements in the overall classification accuracy. The same principle applies to Spatial Averaging. By using an increasingly detailed data transformation over the *same* sample, we add additional features to our classification input that will (potentially) help distinguish problem cases. To use Breiman's terminology, we can consider each representation as a replicate of the original training data. Therefore, instead of having to take random samples to obtain our replicate training data, we can simply apply different transformations to the same sample.

In his review on bagging and its ability to increase classification accuracy, Domingos argues that the improvement is due to a shift in complexity from an overly simplistic decision tree model to one that is more complex [119]; in effect, the supposedly beneficial *simplicity bias* that is a function of many decision tree algorithms can actually degrade classification performance. A simple decision tree may be more interpretable, but limiting or favoring a small number of leaves can cause a "smoothing over" of the learner's model. A tree with a large number of leaves might be able to more accurately partition this space, but a more simplistic and therefore coarser-grained model is preferred. While bagging achieves this increase in complexity through resampling, we achieve a similar increase by modeling the data at varying resolutions.

7.2 Experiments and Results

We now illustrate the effectiveness of our Spatial Averaging algorithm by testing it on the Topography and 63_Fold datasets. Unless otherwise noted, all results are listed as the percentage of correct classification of the testing data.

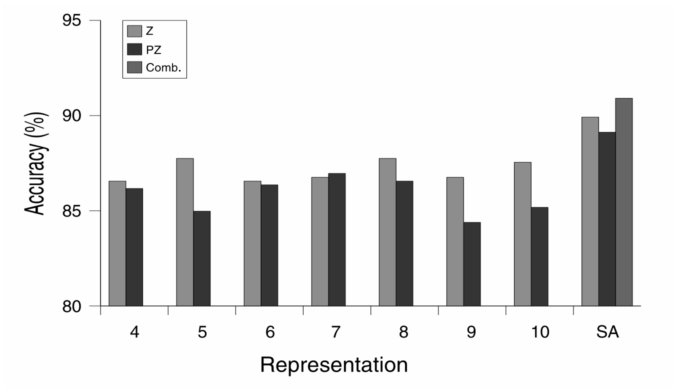
Like many ensemble-learning algorithms, our Spatial Averaging technique can be applied to a number of underlying classification strategies. In these experiments, we elected to use C4.5 decision trees [29]. This choice was made for several reasons. Decision trees are fast, easy to construct, and highly interpretable, which is a plus when trying to explain the criteria for a classification decision.

When dividing our datasets, we set the number of folds to 10, and when creating our Spatial Averaging + Bagging and Bagged Spatial Averaging classifiers, the number of iterations is also set to 10. We compare our Spatial Averaging classification strategy against the performance of a single decision tree as well as a bagged decision tree. We examine whether the Spatial Averaging + Bagging and Bagged Spatial Averaging variants provide any improvement over a base Spatial Averaging classifier. We explore the effects on accuracy when using just a subset of the representations (i.e. the Top-K) and determine the utility of our weighted voting scheme.

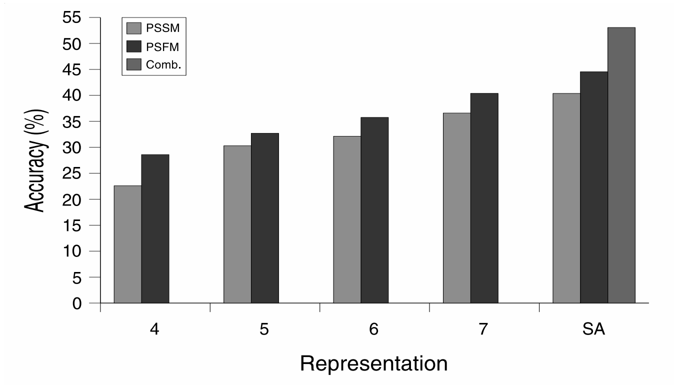
7.2.1 Spatial Averaging vs. C4.5

The first set of experiments involve testing the performance of our Spatial Averaging technique against a standard decision tree-based classifier. The standard decision tree is intended to serve as a baseline. We want to examine how well our Spatial Averaging strategy performs as an ensemble-learning technique.

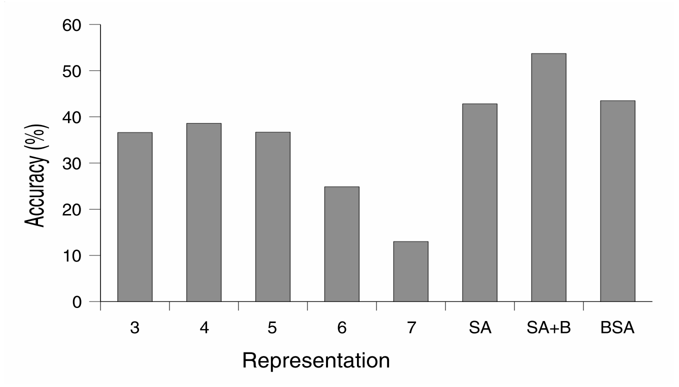
The results of our baseline comparisons are provided in Figure 7.2. The values in the top figure correspond to the corneal topography dataset and the remaining figures to the protein dataset. The results of the sequence-based representations are given in Figure 7.2 (b) and the numbers for the structure-based transformation are shown in Figure 7.2 (c). As shown in Figure 7.2, the Spatial Averaging technique provides much



(a)



(b)



(c)

Figure 7.2: Accuracy for Spatial Averaging (SA) classifier compared with a C4.5 decision tree. Figure (a) corresponds to the corneal topography dataset, (b) to the sequence-based protein representations and (c) to the structure-based wavelet transformation. Accuracy values are given for each order and transformation. Also shown are the results for the combined Spatial Averaging classifier. The y-axis scales have been set to highlight the differences in accuracy.

higher accuracy than standard C4.5 for all the datasets that we examined. On the corneal topography dataset, a single decision tree yields and accuracy of around 86%. For the Spatial Averaging trees, we see an accuracy of 89-90%. Using a paired t-test at 5% significance, we find that the Spatial Averaging classifiers are more accurate, in a statistically-significant manner, than the underlying Zernike and pseudo-Zernike representations.

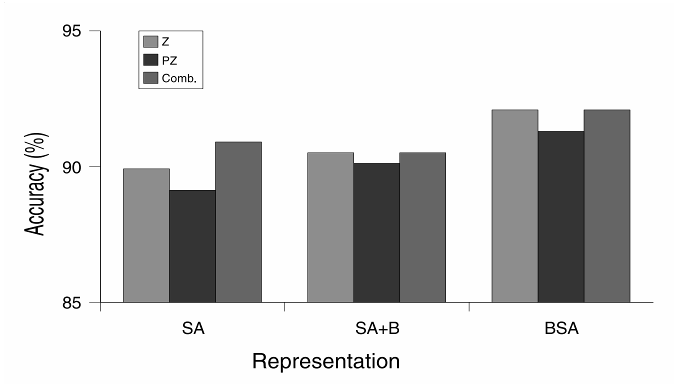
With the topography dataset, we see that the accuracy for the different decision trees is fairly constant, falling in the mid-80% range. With the protein dataset, on the other hand, we see a large difference in accuracy. Using the sequence-based representations (Figure 7.2 (b)), we see individual decision tree accuracies between 22-40% and 13-38% for the structure-based transformations (Figure 7.2 (c)). Despite this wide range of values, we still see an improvement with our Spatial Averaging technique. There is a 4% improvement with the structure-based transformation, and similar increases on the sequence-based representations with the Spatial Averaging classifiers derived from a single transformation. Again using a paired t-test at 5% significance, we find that the accuracy of our Spatial Averaging classifiers is always significantly higher than the stand-alone representations. *With the Spatial Averaging classifier that combines the two sequence representations, we see an accuracy of 53%, a full 13% improvement over any individual decision tree.*

7.2.2 Spatial Averaging and Ensemble-Learning

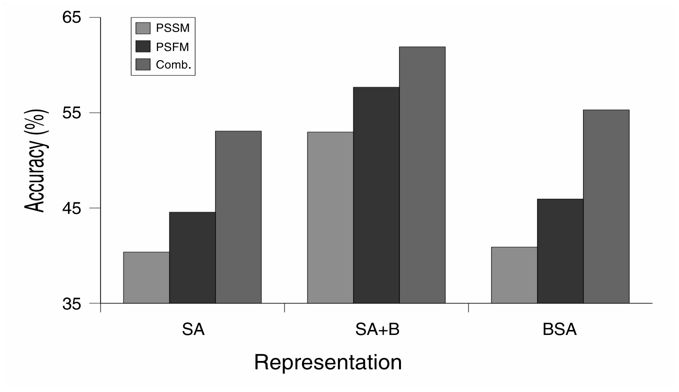
While we would expect our Spatial Averaging algorithm to perform well compared to traditional C4.5, we would also like to determine how it compares to other ensemble-learning techniques. First, we want to directly compare how Spatial Averaging fares against traditional ensemble-learning strategies such as bagging. The data is not shown, *but we find that with the protein datasets, bagging has almost no effect on classification accuracy.* On the topography dataset, bagging increases classification accuracy to roughly 90%, which is comparable to the results returned by our Spatial Averaging classifiers.

We would also like to determine whether it is possible to use existing ensemble-learning techniques in conjunction with Spatial Averaging in order to improve classification accuracy over a traditional Spatial Averaging-classifier or a standard C4.5-based ensemble-learner. This was the motivation behind the Spatial Averaging + Bagging and Bagged Spatial Averaging variants of our algorithm.

The results of these experiments are shown in Figure 7.3 for the corneal topography dataset and sequence-based protein representation and Figure 7.2 (c) for the structure-based protein transformation. On the corneal topography dataset, we see that the accuracy of Spatial Averaging + Bagging is roughly the same as that of a single Spatial Averaging classifier. Bagged Spatial Averaging, on the other hand, provides a improvement in accuracy of up to 2%, though it is not statistically significant. With the protein dataset, the situation is reversed. The accuracy of a Bagged Spatial Averaging classifier is almost identical to that of standard Spatial Averaging. *With Spatial Averaging + Bagging, however, we see a dramatic 10% improvement over a single Spatial Averaging classifier*, to almost 54% on the structure-based transformation and 63% for the representations based on sequence. This represents a statistically significant increase in accuracy.



(a)



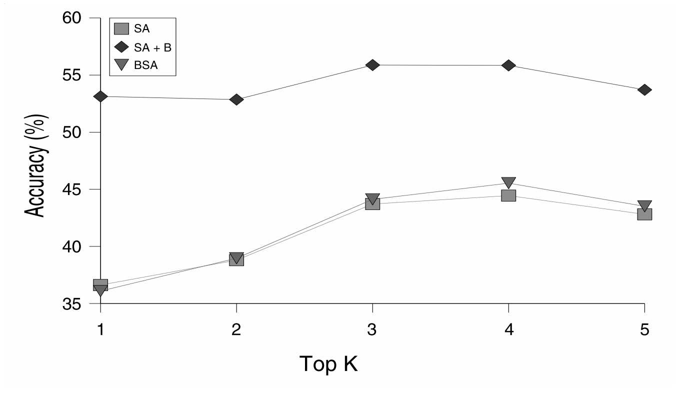
(b)

Figure 7.3: Accuracy for Spatial Averaging combined with ensemble-learning techniques. SA corresponds to Spatial Averaging, $SA + B$ to Spatial Averaging + Bagging and BSA refers to Bagged Spatial Averaging. Figure (a) represents the corneal topography dataset while (b) corresponds to the sequence-based protein representations. The structure-based protein values can be found in Figure 7.2 (c). The y-axis scales have been set to highlight the differences in accuracy.

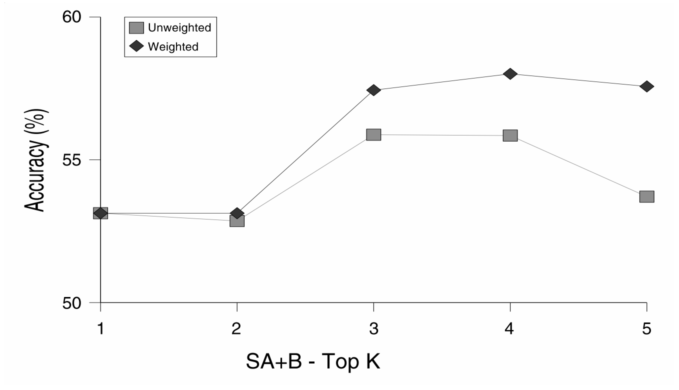
Traditional ensemble-learning techniques try to increase accuracy by repeatedly resampling the training data. By drawing enough samples, the intent is to increase the predictive or descriptive ability of the overall classification model. If one viewed each object as a row in a matrix and each attribute in that object's feature vector as a column, techniques such as boosting and bagging are designed to sample from the row space. Spatial Averaging on the other hand, can be viewed as drawing samples from the column space, where each sample corresponds to the features of a particular representation. Therefore, it can be coupled with techniques such as bagging, providing a greater benefit and a larger increase in predictive ability than any single ensemble-learner.

7.2.3 Top-K and Weighted Voting

Our final set of experiments are designed to determine the effect of using the predictions of the top-K representations to reach a final prediction and whether a weighted voting scheme has any effect on classification accuracy. To simplify the discussion, we focus solely on the structure-based protein representations. As shown in Figure 7.4 (a), we do see an improvement with the top-K strategy. We find that on this dataset, using the top 3 representations, (the wavelet decompositions of level 3, 4 and 5, respectively), to reach a decision provides roughly the same accuracy as using all 5 and that the highest accuracy occurs when using just 4 (levels 3, 4, 5 and 6). This result implies that in instances where one or two classifiers perform significantly worse than the others, it may be beneficial to exclude them from the Spatial Averaging voting process.



(a)



(b)

Figure 7.4: (a) Accuracy for Top-K Spatial Averaging-classification combined with ensemble-learning techniques on the structure-based protein dataset. The values on the x-axis (K) list the number of underlying representations included in the overall Spatial Averaging classifier. SA corresponds to Spatial Averaging, SA + B to Spatial Averaging + Bagging and BSA refers to Bagged Spatial Averaging. (b) Top-K Spatial Averaging + Bagging accuracy on the structure-based protein dataset with and without the weighted voting function. The y-axis values have been selected to highlight the difference in accuracy.

Figure 7.4 (b) shows the effect of weighted voting on the top-K Spatial Averaging + Bagging classification results. We find that weighted voting can improve classification accuracy 1-4% over the default scheme. In addition, the decline that occurs between the top-4 and top-5 classifiers is less than 0.5% compared to a greater than 2% decline in the standard scheme. This means that weighted voting can be used to offset any decline that might occur by including all the representations in the spatial averaging algorithm. As an alternative to simply choosing the top-K representations, we also plan to evaluate the RuleFit algorithm, which provides a structured approach to evaluating the effectiveness of the different representations.

7.3 Conclusions

In this section, we present our algorithm for ensemble-learning classification through model-based spatial averaging. Unlike traditional ensemble-learning techniques, which work to increase accuracy by continually sampling the training data, our method looks to achieve the same result by modeling the data at varying resolutions. As a result of this difference, our technique can be coupled with traditional ensemble-learning methods in order to further improve performance. Empirical results have shown that our standard algorithm can increase accuracy more than 10% over a single decision tree, and when coupled with other ensemble-learners, can provide an additional increase of up to 10%. It is unclear, however, whether the Spatial Averaging + Bagging variant is superior to Bagged Spatial Averaging or vice versa. We also find that in situations where a user has a number of unbalanced classifiers, one can employ our weighting scheme to ensure that the poor classifiers do not skew the results.

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

The application of data mining techniques on structure-based biomedical datasets has the potential to unlock a vast array of hidden and potentially useful information. Working with structure-based data presents a host of challenges, however. Many biomedical datasets are extremely large and complex, or are composed of complex objects that are represented in a high or multi-dimensional format. In addition, structure-based data that has been obtained experimentally will contain some degree of noise and uncertainty and many objects or datasets may have missing or partial values. Structure-based data can also be highly variable, failing to conform to expectations on how it should behave over time. Also, while many biomedical datasets are themselves composed of many members, they can often be divided into a large number of different groups or classes. This can cause problems when trying to develop methods of identifying whether a certain object belongs to a particular group. It becomes especially challenging when a dataset is dominated by a few groups with large memberships, particularly when a researcher is more interested in those that are smaller and more sparsely-populated.

As stated in our thesis statement, we believe that despite these challenges, it is possible to discover useful information in structure-based biomedical data and that furthermore, one can use generalized techniques that are applicable to multiple domains. To illustrate this we propose a generalized workflow for the modeling and analysis of biomedical data. This workflow consists of three main stages: 1) Modeling, 2) Biomedical Knowledge Discovery and 3) Visual Interpretation and Query-based Retrieval, as well as one optional stage: Incorporation of Domain Knowledge.

As validation, we present our work as a series of case studies and extensions spanning a number of domains. These include the modeling and classification of corneal shape (Chapter 3), the classification and retrieval of protein molecules (Chapters 4 and 6, respectively) and the modeling and longitudinal analysis of visual fields (Chapter 5). While we try to make our methods as general and as widely-applicable as possible, every research project will have its own set of goals, datasets and domains and thus require slightly different techniques in order to achieve maximum performance. Our intent in this work is to provide a blueprint for researchers and illustrate how many techniques can be applied to a number of different areas. We show that many of these works share common themes and elements and that not every analysis will require a completely new domain-specific solution.

Even so, certain transformations and modeling methods may be more appropriate for some data types than others. For instance, we modeled our protein and corneal topography datasets using both wavelets and Zernike polynomials. We found that for the topography data, Zernike polynomials provided the most effective representation. With the protein-based data, the opposite was the case. We surmise that wavelets are a better choice for modeling proteins because their 3D structures can be somewhat irregular and span a large range of potential shapes. The wavelet transformation is largely data-dependent and thus may be more capable of handling the wide variety of shapes than a Zernike polynomial, which models objects with a series of fixed modes. The topography dataset contains objects that are much more regular with respect to one another, so one can use differences in a fixed-mode Zernike-based model to differentiate between the different patient classes. So while generalized techniques can be used on multiple data types, some of these techniques may have underlying properties that make them more effective in certain cases.

8.1 Contributions

In our case studies and extensions, we show how generalized modeling techniques can be applied to structure-based data from a number of domains and how those models can be used in conjunction with knowledge discovery algorithms to discover interesting and useful information. We also show how our techniques can be used on objects such as the visual field or protein sequence. While these objects are not specifically derived from structure, they play a key role in the relationship between structure and function. Gaining an understanding of this relationship is an underlying goal to many biomedical knowledge discovery applications. Through our efforts, we make a number of contributions to the specific application domains, as well as the overall scientific community. We list these contributions below.

- In the *protein* domain, we show that generalized modeling methods such as wavelets can be used to efficiently represent sequence and structure. We develop a model that allows for protein retrieval based on local substructures. We also present representations that allow for fast, efficient protein retrieval based on either sequence or structure (or both). The structure-based techniques will be especially helpful in processing and mining the results of large-scale molecular simulation experiments, including those that deal with non-proteins. We also detail a multi-stage classification technique that seeks to improve performance by first partitioning data based on global, high-level details, then classifying each partition using local, fine-grained features. We show the utility of this approach by applying it to a sequence-based protein dataset that contains a large number of classes, many of which contain only a few members.

- To represent *corneal topography* we create and classify models using Zernike polynomials, validating previous work [15]. We develop new techniques that show it is possible to visualize classification decisions or class-differentiating surface features by creating surface models built from the splitting criteria of decision trees.

- We propose a method to model the *visual field* that can be used to highlight and detect defects and abnormalities. This model can also be used to characterize the change in a given field over time. Both of these techniques can be used for classification, but also for detecting outliers and anomalies. Our method is automatic and tunable, allowing clinicians to control the overall sensitivity of the results.

- Finally, we create a general ensemble-learning strategy that improves performance by aggregating the results of classifiers built from models of varying spatial resolutions. This allows a user to benefit from models that provide a global, coarse-grained representation of the object as well as those that contain more fine-grained details, without suffering from the loss of information or noise effects that might arise from relying on a single resolution.

8.2 Future Work

In the future, we plan to expand our efforts and address some of the limitations of our current work. At a minimum, we hope to conduct a further analysis of the visual field dataset as it represents our most recent efforts. At a minimum, we plan a robust, machine learning-based evaluation to determine how we should set the user-defined input parameters to yield the optimal or most accurate output or classification. If possible, we would like to obtain additional datasets that contain a broader mix of abnormal patients. The abnormal patients in our current dataset are limited to a few defect categories. In order to fully validate our technique, we need to test our approach on visual fields that span all possible categories.

In addition, instead of simply stating that a patient has an abnormality in a certain location, we want to provide the user with a set of probabilities indicating the likelihood for each of the defect categories associated with that location. While it may be relatively easy for a clinician to decide among a subset of categories in many instances, giving a set of probabilities along with that subset could help confirm or dissuade their decision for borderline cases. As an extension, we would also like to work on characterizing the relationship among the different defect categories. For example, we want to expand our classification system to allow patients to evolve from one defect category to another over time. Along those same lines, we also plan to give the user the option of relaxing the qualifying criteria for each category, allowing for the possible detection of early disease cases. Finally, since it is usually easier to obtain unlabeled or partially-labeled datasets, we plan to further evaluate our technique in a semi-supervised manner, using it on unlabeled data to detect abnormal or anomalous patient histories. This would permit us, for example, to flag histories that include patients with a visual field that is marked as normal but progressing or abnormal but improving. These patients could then be targeted for special attention by the clinician. If the dataset is unlabeled, having not yet been evaluated by a domain expert, this could allow for a more rapid treatment or intervention than might normally occur.

We would like to continue to refine and evaluate our Spatial Averaging technique with additional datasets and representations. As part of this evaluation, we would also like to develop the statistical and theoretical justification needed to show why this particular technique appears to be so effective as well as any limitations that might prevent its use. Recent efforts have provided such a foundation for ensemble-learning strategies such as boosting [120] and we hope to establish similar underpinnings for Spatial Averaging. In addition, we also intend to use techniques such as RuleFit [40] to determine which of the individual classifiers are the most important to the overall Spatial Averaging classifier. By focusing on a smaller subset of classifiers, we can reduce the number of models that a user needs to examine to determine the rationale behind a classification decision, improving the overall interpretability of the Spatial Averaging algorithm.

We also plan to apply our structure-based protein representations to molecular folding simulation data. By grouping different representations, one can determine whether molecules have similar folding pathways or share certain intermediate structures. Also, given a set of structures, outliers could signify abnormal structures, which might be indicative of structure that is lethal or detrimental to the organism. Work by others in this area is ongoing [14,121], but it still remains a rich direction for future study.

BIBLIOGRAPHY

[1] C. H. Q. Ding and I. Dubchak, “Multi-class protein fold recognition using support vector machines and neural networks,” *Bioinformatics*, vol. 17, no. 4, pp. 349–358, April 2001.

[2] F. Zernike, “Beugungstheorie des schneidenverfahrens und seiner verbesserten form, der phasenkontrastmethode,” *Physica*, vol. 1, pp. 689–704, 1934.

[3] M. Born and E. Wolf, *Principles of optics : electromagnetic theory of propagation, interference and diffraction of light*, 6th ed. Oxford ; New York: Pergamon Press, 1980.

[4] A. B. Bhatia and E. Wolf, “On the circle polynomials of zernike and related orthogonal sets,” in *Proc. Cambridge Philos. Soc.*, vol. 50, 1954, pp. 40–53.

[5] C.-H. Teh and R. T. Chin, “On image analysis by the methods of moments,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 496–513, 1988.

[6] G. Demoment, “Image reconstruction and restoration: Overview of common estimation structures and problems,” *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 37, no. 12, pp. 2024–2036, 1989.

[7] R. Mukundan, S. H. Ong, and P. A. Lee, “Discrete vs. continuous orthogonal moments for image analysis,” in *The 2001 International Conference on Imaging Science, Systems and Technology: Computer Graphics (CISST)*, 2001, pp. 23–29.

[8] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, E. F. M. Jr., M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi, “The protein data bank: A computer-based archival file for macromolecular structure,” *J. Mol. Biol.*, vol. 112, pp. 535–542, 1977.

[9] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, “SCOP: a structural classification of proteins database for the investigation of sequences and structures,” *J. Mol. Biol*, vol. 247, pp. 536–540, 1995.

[10] D. R. Iskander, M. J. Collins, and B. Davis, "Optimal modeling of corneal surfaces with zernike polynomials," *IEEE Trans Biomed Eng*, vol. 48, no. 1, pp. 87–95, 2001.

[11] D. R. Iskander, M. J. Collins, B. Davis, and R. Franklin, "Corneal surface characterization: How many zernike terms should be used? (ARVO abstract)," *Invest Ophthalmol Vis Sci*, vol. 42, no. 4, p. S896, 2001.

[12] M. D. Twa, S. Parthasarathy, T. W. Raasch, and M. A. Bullimore, "Automated classification of keratoconus: A case study in analyzing clinical data," in *SIAM International Conference on Data Mining (SDM)*, San Francisco, CA, 2003.

[13] J. L. Keltner, C. A. Johnson, K. E. Cello, M. A. Edwards, S. E. Bandermann, M. A. Kass, and M. O. Gordon, "Classification of visual field abnormalities in the ocular hypertension treatment study," *Arch Ophthalmol*, no. 5, pp. 643–650, 2003.

[14] S. M. Larson, C. D. Snow, M. Shirts, and V. S. Pande, "Folding@home and genome@home: Using distributed computing to tackle previously intractable problems in computational biology," in *Computational Genomics*, R. Grant, Ed. Horizon Press, 2002.

[15] M. D. Twa, "Spatial pattern recognition of videokeratography by decision tree classification of zernike polynomials," Master's thesis, The Ohio State University, 2002.

[16] L. N. Thibos, R. A. Applegate, J. T. Schwiegerling, and R. Webb, "Standards for reporting the optical aberrations of eyes," *J. Refract. Surg.*, vol. 18, no. 5, pp. S652–60, 2002 Sept-Oct.

[17] J. N. K. Liu, M. Wang, and B. Feng, "ibotguard: an internet-based intelligent robot security system using invariant face recognition against intruder," *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 35, no. 1, pp. 97–105, 2005.

[18] D. H. Hoekman and C. Varekamp, "Observation of tropical rain forest trees by airborne high-resolution interferometric radar," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 39, no. 3, pp. 584–594, 2001.

[19] H. Hse and A. R. Newton, "Sketched symbol recognition using zernike moments," in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 1, 2004, pp. 367–370.

[20] D. R. Iskander, M. R. Morelande, M. J. Collins, and B. Davis, "Modeling of corneal surfaces with radial polynomials," *IEEE Trans Biomed Eng*, vol. 49, no. 4, pp. 320–328, 2002.

[21] I. Daubechies, *Ten Lectures on Wavelets.* Philadelphia: Soc. Indust. Appl. Math., 1992.

[22] A. F. Laine, "Wavelets in temporal and spatial processing of biomedical images," *Annu. Rev. Biomed. Eng.*, vol. 02, pp. 511–550, 2000.

[23] F. Schwarzer and I. Lotan, "Approximation of protein structure for fast similarity measure," in *RECOMB'03.* ACM, 2003, pp. 267–276.

[24] I. H. Witten and E. Frank, *Data mining : practical machine learning tools and techniques with Java implementations.* San Francisco, CA: Morgan Kaufmann, 2000.

[25] J. Han and M. Kamber, *Data mining : concepts and techniques.* San Francisco: Morgan Kaufmann Publishers, 2001.

[26] G. H. John and P. Langley, "Estimating continuous distributions in (bayesian) classifiers," in *11th Conf on Uncertainty in AI*, 1995, pp. 338–345.

[27] I. Rish, "An empirical study of the naive bayes classifier," IBM, Tech. Rep. RC22230, 2001.

[28] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[29] ——, *C4.5 : programs for machine learning.* San Mateo, Calif.: Morgan Kaufmann Publishers, 1993.

[30] C. Bishop, *Neural Networks for Pattern Recognition.* Oxford: Oxford University Press, 1995.

[31] G. Schwarzer, W. Vach, and M. Schumacher, "On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology," *Stat. Med.*, vol. 19, no. 4, pp. 541–561, 2000.

[32] L. A. Carvalho, "Preliminary results of neural networks and zernike polynomials for classification of videokeratography maps," *Optometry and Vision Science*, vol. 82, no. 2, pp. 151–158, 2005.

[33] M. K. Smolek and S. D. Klyce, "Screening of prior refractive surgery by a wavelet-based neural network," *J Cataract Refract Surg*, vol. 27, no. 12, pp. 1926–31, 2001.

[34] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines.* Cambridge Univeristy Press, 2000.

[35] D. Aha and D. Kibler, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[36] G. Demiroz and H. A. Guvenir, "Classification by voting feature intervals," in *European Conf. on Machine Learning.* Springer, 1997, pp. 85–92.

[37] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[38] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 26, no. 2, pp. 123–140, 1996.

[39] ——, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[40] J. Friedman and B. E. Popescu, "Predictive learning via rule ensembles," Stanford University, Tech. Rep., 2004.

[41] K. Marsolo, M. Twa, M. A. Bullimore, and S. Parthasarathy, "Spatial modeling and classification of corneal shape," *IEEE Transactions on Information Technology in Biomedicine*, vol. 11, no. 2, pp. 203–212, 2007.

[42] K. Marsolo, S. Parthasarathy, M. D. Twa, and M. A. Bullimore, "A model-based approach to visualizing classification decisions for patient diagnosis," in *Proceedings of the 10th Conference on Artificial Intelligence in Medicine (AIME).* Springer, 2005, pp. 473–483.

[43] P. M. Kiely, G. Smith, and L. G. Carney, "The mean shape of the human cornea," *Journal of Modern Optics*, vol. 29, no. 8, pp. 1027–1040, 1982.

[44] R. B. Mandell, "A guide to videokeratography," *Int Contact Lens Clin*, vol. 23, no. 6, pp. 205–28, 1996.

[45] J. Schwiegerling, J. E. Greinvenkamp, and J. M. Miller, "Representation of videokeratoscopic height data with zernike polynomials," *J. Opt. Soc. Am. A*, vol. 12, no. 10, pp. 2105–2113, 1995.

[46] M. K. Smolek and S. D. Klyce, "Zernike polynomial fitting fails to represent all visually significant corneal aberrations," *Invest Ophthalmol Vis Sci*, vol. 44, no. 11, pp. 4676–81, 2003.

[47] N. Maeda, S. D. Klyce, M. K. Smolek, and H. W. Thompson, "Automated keratoconus screening with corneal topography analysis," *Invest Ophthalmol Vis Sci*, vol. 35, no. 6, pp. 2749–57, 1994.

[48] Y. S. Rabinowitz and K. Rasheed, "KISA% - minimal topographic criteria for diagnosing keratoconus," *J Cataract Refract Surg*, vol. 25, no. 10, pp. 1327–35, 1999.

[49] P. Accardo and S. Pensiero, "Neural network-based system for early keratoconus detection from corneal topography," *J. Biomedical Informatics*, vol. 35, pp. 151–159, 2003.

[50] N. Maeda, S. D. Klyce, and M. K. Smolek, "Neural network classification of corneal topography. preliminary demonstration," *Invest Ophthalmol Vis Sci*, vol. 36, no. 7, pp. 1327–35, 1995.

[51] M. K. Smolek and S. D. Klyce, "Current keratoconus detection methods compared with a neural network approach," *Invest Ophthalmol Vis Sci*, vol. 38, no. 11, pp. 2290–9, 1997.

[52] M. D. Twa, S. Parthasarathy, C. Roberts, A. M. Mahmoud, T. W. Raasch, and M. A. Bullimore, "Automated decision tree classification of corneal shape," *Optometry and Vision Science*, vol. 82, no. 12, pp. 1038–1046, 2005.

[53] J. Y. Wang and D. E. Silva, "Wave-front interpretation with zernike polynomials," *Applied Optics*, vol. 19, no. 9, pp. 1510–1518, 1980.

[54] G. Williams, *Linear algebra with applications.* Newton, MA: Allyn and Bacon, Inc., 1984.

[55] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *13th Intl Conf on Machine Learning*, 1996, pp. 148–156.

[56] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Intl Joint Conf. on AI*, 1995, pp. 1137–1145.

[57] L. Platzman and J. Bartholdi, "Spacefilling curves and the planar travelling salesman problem," *J. Assoc. Comput. Mach*, vol. 46, pp. 719–737, 1989.

[58] K. Marsolo, S. Parthasarathy, and C. Ding, "A multi-level approach to SCOP fold recognition," in *Proceedings of the 5th IEEE Symposium on Bioinformatics & Bioengineering (BIBE05).* IEEE, 2005, pp. 57–64.

[59] K. Marsolo and S. Parthasarathy, "Alternate representation of distance matrices for characterization of protein structure," in *Proccedings of the 5th IEEE International Conference on Data Mining (ICDM05).* IEEE, 2005, pp. 298–305.

[60] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton, "CATH: A hierarchic classification of protein domain structures," *Structure*, vol. 5, no. 8, pp. 1093–1108, 1997.

[61] A. C. Tan, D. Gilbert, and Y. Deville, “Multi-class protein fold classification using a new ensemble machine learning approach,” *Genome Informatics*, vol. 14, pp. 206–217, 2003.

[62] A. Chinnasamy, W. K. Sung, and A. Mittal, “Protein structure and fold prediction using tree-augmented naive bayesian classifier,” in *Proc. of PSB 2004*. Stanford, CA: World Scientific Press, 2004, pp. 387–398.

[63] S. Y. M. Shi, P. N. Suganthan, and K. Deb, “Multi-class protein fold recognition using multi-objective evolutionary algorithms,” in *Proc. of IEEE Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2004)*. IEEE, 2004, pp. 61–66.

[64] I. Dubchak, I. Muchnik, S. Holbrook, and S.-H. Kim, “Prediction of protein folding class using global description of amino acid sequence,” *Proc. Natl. Acad. Sci. USA*, vol. 92, pp. 8700–8704, September 1995.

[65] S. K. Murthy, “Automatic construction of decision trees from data: A multi-disciplinary survey,” *Data Mining and Knowledge Discovery*, vol. 2, no. 4, pp. 345–389, 1998.

[66] L. A. Breslow and D. W. Aha, “Simplifying decision trees: A survey,” NCARAI, Tech. Rep. AOC-96-014, 1996.

[67] G. Pagallo and D. Haussler, “Boolean feature discovery in empirical learning,” *Machine Learning*, vol. 5, no. 1, pp. 71–99, March 1990.

[68] L. Holm and C. Sander, “Protein structure comparision by alignment of distance matrices,” *J. Mol. Biol*, vol. 233, pp. 123–138, 1993.

[69] I. N. Shindyalov and P. Bourne, “Modern protein structure alignment by incremental combinatorial extension (ce) of the optimal path,” *Protein Eng.*, vol. 11, no. 9, pp. 739–747, 1998.

[70] O. Røgen and B. Fain, “Automatic classification of protein structure by using gauss integrals,” *PNAS*, vol. 100, no. 1, pp. 119–124, 2003.

[71] J. Huan, W. Wang, A. Washington, J. Prins, and A. Tropsha, “Accurate classification of protein structure families using coherent subgraph analysis,” in *Proceedings of the 9th Pacific Symposium on Biocomputing (PSB’04)*. World Scientific Press, 2004, pp. 411–422.

[72] Z. Aung and K.-L. Tan, “Automatic protein strutcture classification through structural fingerprinting,” in *Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering (BIBE’04)*. IEEE, 2004, pp. 508–515.

[73] S. K. Gardiner and D. P. Crabb, "Examination of differential pointwise linear regression methods for determining visual field progression," *IOVS*, vol. 43, no. 5, pp. 1400–1407, May 2002.

[74] K. Nouri-Mahdavi, D. Hoffman, D. Gaasterland, and J. Caprioli, "Prediction of visual field progression in glaucoma," *IOVS*, vol. 45, no. 12, pp. 4346–4351, December 2004.

[75] A. Tucker, X. Liu, and D. Garway-Heath, "Spatial operators for evolving dynamic bayesian networks from spatial-temporal data," in *Genetic and Evolutionary Computation Conference - GECCO 2003.* Springer, 2003, pp. 2360–2371.

[76] ——, "Bayesian classification and forecasting of visual field deterioration," in *Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP)*, 2003.

[77] C. A. Johnson, P. A. Sample, G. A. Cioffi, J. R. Liebmann, and R. N. Weinreb, "Structure and function evaluation (SAFE): I. criteria for glaucomatous visual field loss using standard automated perimetry (SAP) and short wavelength automated perimetry (SWAP)," *Am J Ophthalmol*, vol. 134, no. 2, pp. 177–185, 2002.

[78] M. H. Goldbaum, P. A. Sample, K. Chan, J. Williams, T.-W. Lee, E. Blumenthal, C. A. Girkin, L. M. Zangwill, C. Bowd, T. Sejnowski, and R. N. Weinreb, "Comparing machine learning classifiers for diagnosing glaucoma from standard automated perimetry," *IOVS*, vol. 43, no. 1, pp. 162–169, January 2002.

[79] A. Turpin, E. Frank, M. Hall, I. A. Witten, and C. A. Johnson, "Determining progression in glaucoma using visual fields," in *Proceedings of PAKDD'01.* Springer, 2001, pp. 136–147.

[80] L. Wei, E. Keogh, and X. Xi, "SAXually explict images: Finding unusual shapes," in *Proceedings of ICDM'06.* IEEE, 2006, pp. 711–720.

[81] D. Yankov and E. Keogh, "Manifold clustering of shapes," in *Proceedings of ICDM'06.* IEEE, 2006, pp. 1167–1171.

[82] S. Pizer, G. Gerig, S. Joshi, and S. Aylward, "Multiscale medial shape-based analysis of image objects," in *Proceedings of the IEEE*, vol. 91, no. 10, 2003, pp. 1670–1679.

[83] G. Gerig, M. Styner, and G. Székely, "Statistical shape models for segmentation and structural analysis," in *Proceedings of the IEEE Symposium on Biomedical Imaging (ISBI 2002).* IEEE, 2002, pp. 18–21.

[84] D. Nain, S. Haker, A. Bobick, and A. Tannenbaum, "Multiscale 3d shape analysis using spherical wavelets," in *Proceedings of MICCAI'05.* Springer Verlag, 2005, pp. 459–467.

[85] P. Golland, W. E. L. Grimson, M. E. Shenton, and R. Kikinis, "Detection and analysis of statistical differences in anatomical shape," *Medical Image Analysis*, vol. 9, pp. 69–86, 2005.

[86] M. H. Dunham, *Data Mining: Introductory and Advanced Topics.* Prentice Hall, 2003.

[87] W. Press, B. Flannery, S. Teukolsky, and W. T. Vetterling, *Numerical Recipes: the art of scientific computing.* Cambridge University Press, 1986.

[88] D. Thompson, S. Parthasarathy, R. Machiraju, and S. Lawrence, "Improvements to response-surface based vehicle design using a feature-centric approach," in *International Conference on Computational Science 2004.* Springer, 2004, pp. 764–770.

[89] E. Carpaneto and G. Chicco, "Performance of the simulated annealing-based algorithms for the optimal reconfiguration of distribution systems," *IEEE Power Tech Proceedings*, vol. 3, pp. 50–56, 2001.

[90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *J. of Mo. Biol.*, vol. 215, pp. 403–410, 1990.

[91] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Anang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, pp. 3389–3402, 1997.

[92] K. Marsolo, S. Parthasarathy, and K. Ramamohanarao, "Structure-based querying of proteins using wavelets," in *Proceedings of CIKM'06.* IEEE, 2006, pp. 24–33.

[93] K. Marsolo and S. Parthasarathy, "On the use of structure and sequence-based features for protein classification and retrieval," in *Proceedings of ICDM'06.* ACM, 2006, pp. 394–403.

[94] L. Hammel and J. Patel, "Searching on the secondary structure of protein sequences," in *VLDB*, 2002, pp. 634–645.

[95] S. Tata and J. Patel, "PiQA: An algebra for querying protein data sets," in *SSDBM*, 2003, pp. 141–150.

[96] O. Çamoğlu, T. Kahveci, and A. Singh, "Towards index-based similarity search for protein structure databases," in *2nd IEEE Computer Society Bioinformatics Conference (CSB).* IEEE, 2003, pp. 148–158.

[97] T. Madej, J.-F. Gibrat, and S. Bryant, "Threading a database of protein cores," *Protein Struct. Funct. Genet.*, vol. 23, no. 3, pp. 356–369, Nov. 1995.

[98] Z. Tan and A. K. H. Tung, "Substructure clustering on sequential 3D object datasets," in *International Conference on Data Engineering (ICDE).* Boston: IEEE, 2004, pp. 634–645.

[99] I. Choi, J. Kwon, and S. Kim, "Local feature frequency profile: A method to measure structural similarity in proteins," *PNAS*, vol. 101, no. 11, pp. 3797–3802, 2004.

[100] T. Can and Y. Wang, "Ctss: A robust and efficient method for protein structure alignment based on local geometrical and biological features," in *Proceedings of the IEEE Computer Society Bionfinformatics Conference (CSB 2003).* IEEE, 2003, pp. 169–179.

[101] P.-H. Chi, G. Scott, and C.-R. Shyu, "A fast protein structure retrieval system using image-based distance matrices and multidimensional index," in *Proceedings of the 4th IEEE Symposium on BioInformatics and BioEngineering (BIBE 2004).* IEEE, 2004, pp. 522–532.

[102] F. Gao and M. Zaki, "PSIST: Indexing protein structures using suffix trees," in *IEEE Computational Systems Bioinformatics Conference (CSB).* Palo Alto, CA: IEEE, August 2005, pp. 212–222.

[103] A. Bhattacharya, T. Can, T. Kahveci, A. Singh, and Y. Wang, "ProGreSS: Simultaneous searching of protein databases by sequence and structure," in *Pacific Symposium on Biocomputing (PSB 2004)*, vol. 9. World Scientific Press, 2004, pp. 264–275.

[104] Z. Aung and K.-L. Tan, "Rapid 3d protein structure database searching using information retrieval techniques," *Bioinformatics*, vol. 20, pp. 1045–1052, 2004.

[105] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proc. Natl. Acad. Sci*, vol. 89, pp. 10 915–10 919, 1992.

[106] E. Ie, J. Weston, W. S. Noble, and C. Leslie, "Multi-class protein fold recognition using adaptive codes," in *Proceedings of the 22nd Intl Conf on Machine Learning.* Bonn, Germany: ACM, 2005, pp. 329–336.

[107] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. S. Leslie, "Profile-based string kernels for remote homology detection and motif extraction," in *Proceedings of CSB 2004*. IEEE, 2004, pp. 152–160.

[108] H. Rangwala and G. Karypis, "Profile-based direct kernels for remote homology detection and fold recognition," *Bioinformatics*, vol. 21, no. 23, pp. 4239–47, 2005.

[109] S. Parthasarathy and C. C. Aggarwal, "On the use of conceptual reconstruction for mining massively incomplete data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1512–1521, 2003.

[110] S. E. Brenner, P. Koehl, and M. Levitt, "The ASTRAL compendium for sequence and structure analysis," *Nucleic Acids Research*, vol. 28, pp. 254–256, 2000.

[111] S. Han, B.-C. Lee, S. T. Yu, C.-S. Jeong, S. Lee, and D. Kim, "Fold recognition by combining profile-profile alignment and support vector machine," *Bioinformatics*, vol. 21, no. 11, pp. 2667–2673, 2005.

[112] J. L. Bentley, "Multidimensional binary search trees used for associate searching," *Comm. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[113] S. Mehta, S. Barr, A. Choy, H. Yang, S. Parthasarathy, R. Machiraju, and J. Wilkins, "Dynamic classification of anomalous structures in molecular dynamics simulation data," in *Proceedings of the SIAM Conference on Data Mining*. SIAM, 2005, pp. 161–172.

[114] K. Karplus, C. Barrett, and R. Hughley, "Hidden markov models for detecting remote protein homologies," *Bioinformatics*, vol. 14, pp. 846–856, 1998.

[115] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[116] J. Platt, *Advances in Kernel Methods - Support Vector Learning.* Cambridge, MA, USA: MIT Press, 1998, ch. Fast Training of Support Vector Machines using Sequential Minimal Optimization, pp. 185–208.

[117] J. Weston, C. Leslie, D. Zhou, and W. S. Noble, "Semi-supervised protein classification using cluster kernels," in *Advances in Neural Information Processing Systems (NIPS) 16*. NIPS, 2004, pp. 595–602.

[118] K. Marsolo, M. Twa, S. Parthasarathy, and M. Bullimore, "Classification of biomedical data through model-based spatial averaging," in *Proceedings of the 5th IEEE Symposium on Bioinformatics & Bioengineering (BIBE05)*. IEEE, 2005, pp. 49–56.

[119] P. Domingos, “Why does bagging work? a bayesian account and its implications,” in *Third International Conference on Knowledge Discovery and Data Mining*. Newport Beach, CA: AAAI Press, 1997, pp. 155–158.

[120] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors),” *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.

[121] H. Yang, S. Parthasarathy, and D. Ucar, “A spatio-temporal mining approach towards summarizing and analyzing protein folding trajectories,” *Algorithms for Molecular Biology*, vol. 2, no. 3, 2007.