Generalization of Machine Learning Model and Various Applications in NLP and Vision

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Masters of Science in the Graduate School of The Ohio State University

By

Soumik Mandal, BE, MS

Graduate Program in Department of Computer Science

The Ohio State University

2021

Master's Thesis Committee:

Rajiv Ramnath, Advisor

Thomas Bihari

Abstract

Generalization is a big issue in machine learning. At first we show that classical notion of U shaped generalization behavior is not true for modern day ML models. Instead modern day ML models show double descent characteristics of generalization. To address issue of generalization for different problems we show different strategies are useful. For information extraction from document we show that data augmentation using rule based technique is highly effective. For prediction of different pandemic related government decision, we show that clustering based feature augmentation improves generalization performance of models. For disinformation detection we show that graph based information propagation improves generalization performance. In general we show different strategies to improve generalization performance for different applications and in this process we also show new way of looking into bias-variance trade-off of ML models.

This is dedicated to the ones I love ... ma, baba ...

Acknowledgments

I would like to thank my advisor Prof. Ramnath for his openness to discuss and helping me in my research projects. I will also thank my collaborators in and outside OSU. I thank my friends Subrata Sarkar, Ritesh Sarkhel for their research discussion with me.

Vita

2010 ..... B.E. Electrical Engineering

2016 ..... M.S. Electrical Engineering and Computer Engineering

Publications

Research Publications

M. Belkin, D. Hsu, S. Ma, S. Mandal "Reconciling modern machine learning practice and the classical bias-variance trade-off". *PNAS 2019.*

M. Belkin et al., S. Ma, S. Mandal "To understand deep learning we need to understand kernel learning". *ICML 2019.*

Fields of Study

Major Field: Department of Computer Science

Table of Contents

Page

List of Tables

List of Figures

Chapter 1: Double Descent Behavior : U Shaped Bias Variance Trade-off Is Not True

1.1 Introduction

Recently Machine Learning has become indispensable to important applications in science, technology and commerce. The focus of machine learning is on the problem of prediction: given a sample of training examples $(x_1, y_1), \ldots, (x_n, y_n)$ from $\mathbb{R}^d \times \mathbb{R}$, we learn a predictor $h_n \colon \mathbb{R}^d \to \mathbb{R}$ that is used to predict the label y of a new point x, unseen in training.

The predictor h_n is commonly chosen from some function class $\mathcal{H}$, such as neural networks with a certain architecture, using *empirical risk minimization (ERM)* and its variants. In ERM, the predictor is taken to be a function $h \in \mathcal{H}$ that minimizes the *empirical (or training) risk* $\frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i)$, where ℓ is a loss function, such as the squared loss $\ell(y', y) = (y' - y)^2$ for regression or zero-one loss $\ell(y', y) = \mathbb{1}_{\{y' \neq y\}}$ for classification.

The goal of machine learning is to find h_n that performs well on new data, unseen in training. To study performance on new data (known as *generalization*) we typically assume the training examples are sampled randomly from a probability distribution P over $\mathbb{R}^d \times \mathbb{R}$, and evaluate h_n on a new test example (x, y) drawn independently from P. The challenge stems from the mismatch between the goals of minimizing the empirical risk (the explicit goal of ERM algorithms, optimization) and minimizing the *true (or test) risk* $E_{(x,y)\sim P}[\ell(h(x), y)]$ (the goal of machine learning).

Conventional wisdom in machine learning suggests controlling the capacity of the function class $\mathcal{H}$ based on the *bias-variance trade-off* by balancing *under-fitting* and *over-fitting* :

1. If $\mathcal{H}$ is too small, all predictors in $\mathcal{H}$ may *under-fit* the training data (i.e., have large empirical risk) and hence predict poorly on new data.

2. If $\mathcal{H}$ is too large, the empirical risk minimizer may *over-fit* spurious patterns in the training data resulting in poor accuracy on new examples (small empirical risk but large true risk).

The classical thinking is concerned with finding the "sweet spot" between under-fitting and over-fitting. The control of the function class capacity may be explicit, via the choice of $\mathcal{H}$ (e.g., picking the neural network architecture), or it may be implicit, using regularization (e.g., early stopping). When a suitable balance is achieved, the performance of h_n on the training data is said to *generalize* to the population P. This is summarized in the classical U-shaped risk curve, shown in fig:double-descent(a) that has been widely used to guide model selection and is even thought to describe aspects of human decision making [9]. The textbook corollary of this curve is that "a model with zero training error is overfit to the training data and will typically generalize poorly" [11, page 221], a view still widely accepted.

Yet, practitioners routinely use modern machine learning methods, such as large neural networks and other non-linear predictors that have very low or zero training risk. In spite of the high function class capacity and near-perfect fit to training data, these predictors often give very accurate predictions on new data. Indeed, this behavior has guided a best practice in deep learning for choosing neural network architectures, specifically that the network should be large enough to permit effortless zero loss training (called *interpolation*) of the training data [21]. Moreover, in direct challenge to the bias-variance trade-off philosophy, recent empirical evidence indicates that neural networks trained to interpolate the training

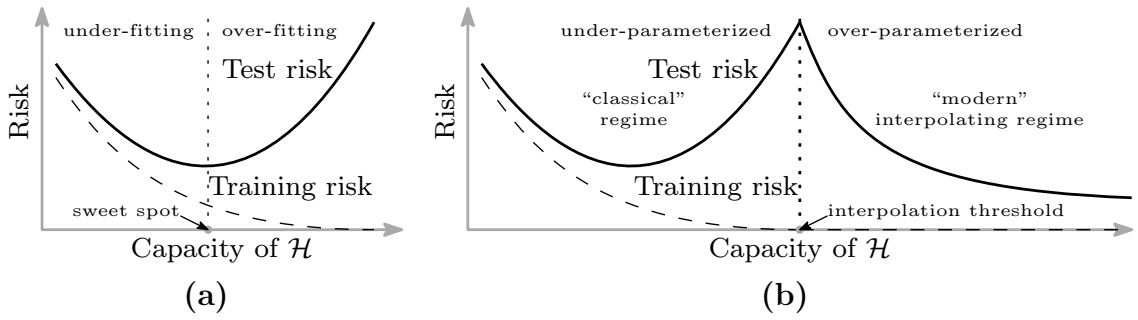

Figure 1.1: **Curves for training risk (dashed line) and test risk (solid line).** (**a**) The classical *U-shaped risk curve* arising from the bias-variance trade-off. (**b**) The *double descent risk curve*, which incorporates the U-shaped risk curve (i.e., the "classical" regime) together with the observed behavior from using high capacity function classes (i.e., the "modern" interpolating regime), separated by the interpolation threshold. The predictors to the right of the interpolation threshold have zero training risk.

data obtain near-optimal test results even when the training data are corrupted with high levels of noise [26, 1].

The main finding of this work is a pattern for how performance on unseen data depends on model capacity and the mechanism underlying its emergence. This dependence, empirically witnessed with important model classes including neural networks and a range of datasets, is summarized in the "double descent" risk curve shown in Figure 1.1(b). The curve subsumes the classical U-shaped risk curve from Figure 1.1(a) by extending it beyond the point of interpolation.

When function class capacity is below the "interpolation threshold", learned predictors exhibit the classical U-shaped curve from Figure 1.1(a). (In this paper, function class capacity is identified with the number of parameters needed to specify a function within the class.) The bottom of the U is achieved at the sweet spot which balances the fit to the training data and the susceptibility to over-fitting: to the left of the sweet spot, predictors are under-fit, and immediately to the right, predictors are over-fit. When we increase the function class capacity high enough (e.g., by increasing the number of features or the size

of the neural network architecture), the learned predictors achieve (near) perfect fits to the training data—i.e., interpolation. Although the learned predictors obtained at the interpolation threshold typically have high risk, we show that increasing the function class capacity beyond this point leads to decreasing risk, typically going below the risk achieved at the sweet spot in the "classical" regime.

All of the learned predictors to the right of the interpolation threshold fit the training data perfectly and have zero empirical risk. So why should some—in particular, those from richer functions classes—have lower test risk than others? The answer is that the capacity of the function class does not necessarily reflect how well the predictor matches the *inductive bias* appropriate for the problem at hand. For the learning problems we consider (a range of real-world datasets as well as synthetic data), the inductive bias that seems appropriate is the regularity or smoothness of a function as measured by a certain function space norm. Choosing the smoothest function that perfectly fits observed data is a form of Occam's razor: the simplest explanation compatible with the observations should be preferred (cf. [24, 2]). By considering larger function classes, which contain more candidate predictors compatible with the data, we are able to find interpolating functions that have smaller norm and are thus "simpler". Thus increasing function class capacity improves performance of classifiers.

1.2 Models That Shows Double Descent Behavior

We start with simple models at first, and the slowly show result for complex models. We first consider a popular class of non-linear parametric models called *Random Fourier Features* (*RFF*) [19], which can be viewed as a class of two-layer neural networks with fixed weights in the first layer. Once we show result for RFF, then we move to more variation of RFF named Random-RELU feature. This is another random model with sinusoids in RFF replaced by RELU. Random-RELU is a very shallow version of modern day fully connected

neural network. Once we have result for Random-RELU model, we show result for fully connected neural network. We also show result for CNN.

1.3 Random Fourier Feature

1.3.1 Model

Sampling from probability distribution : In this section we derive the distribution from which we can the angular frequencies . For a Gaussian kernel $e^{\|\mathbf{x}-\mathbf{y}\|_2^2/\mathbf{2}*\sigma^2}$, $\mathbf{x} \in \mathbb{R}^{D\times 1}$ where D is number of features in original data and σ is the bandwidth is a scalar . So we need Fourier Transform of Gaussian kernel to find the ω's for making RFF .

$$F(g(x)) = G(f) = \frac{1}{2\pi}\int_{-\infty}^{\infty} g(x)e^{-i2\pi fx}\,dx = \frac{1}{2\pi}\int_{-\infty}^{\infty} g(x)e^{-i\omega x}\,dx$$

Fourier Transform of 1-dimensional Gaussian kernel is :

$$\begin{aligned}
G_1(\omega) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-x^2/2\sigma^2}e^{-i\omega x}\,dx \\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-x^2/2\sigma^2}e^{-sx}\,dx \\
&= \frac{e^{\sigma^2 s^2/2}}{2\pi}\int_{-\infty}^{\infty} e^{-\frac{(x+\sigma^2 s)^2}{2\sigma^2}}\,dx \\
&= \frac{e^{-\sigma^2\omega^2/2}}{2\pi}\sigma\sqrt{2\pi} \\
tg &= \frac{\sigma}{\sqrt{2\pi}}e^{-\sigma^2\omega^2/2} \\
&= N(0, 1/\sigma^2)
\end{aligned}$$

Fourier Transform of D-dimensional Gaussian kernel is :

$$
\begin{aligned}
G_D(\omega) &= \left(\frac{1}{2\pi}\right)^D \int_{-\infty}^{\infty} e^{-\|\mathbf{x}\|_2^2/2\sigma^2} e^{-i\omega\mathbf{x}}\, d\mathbf{x} && \text{(dimensions are independent)} \\
&= \left(\frac{1}{2\pi}\right)^D \int_{-\infty}^{\infty} e^{-\frac{x_1^2+\ldots+x_D^2}{2\sigma^2}} e^{-i(\omega_1 x_1+\ldots+\omega_D x_D)}\, d\mathbf{x} \\
&= \frac{\sigma}{\sqrt{2\pi}} e^{-\sigma^2\omega_1^2/2} \ldots \frac{\sigma}{\sqrt{2\pi}} e^{-\sigma^2\omega_D^2/2} \\
&= \left(\frac{\sigma}{\sqrt{2\pi}}\right)^D e^{-\sigma^2\|\omega\|_2^2/2} \\
&= N(\mathbf{0}, diag(1/\sigma^2 \ldots 1/\sigma^2)) && (diag() \in \mathbb{R}^{D\times D} \text{ is covariance matrix})
\end{aligned}
$$

Since, $G_1(\omega)$ and $G_D(\omega)$ are Gaussian distributions, they integrate to 1 and are valid probability distributions. For D dimensional case, we can sample from $N(0, 1/\sigma^2)$ distribution D times and get a vector. So the weight matrix we get is $\mathbf{W} = [\omega_\mathbf{1}\omega_\mathbf{2}\ldots\omega_{\mathbf{a/2}}]$ where $\mathbf{W} \in \mathbb{R}^{D\times a/2}$ and $\omega_\mathbf{i} \in \mathbb{R}^{D\times 1}$ and a is number of RFF. $\mathbf{x_{train}} = [\mathbf{x_1}\ldots\mathbf{x_n}]^T$; $\mathbf{x_{train}} \in \mathbb{R}^{n\times D}$,and $\mathbf{x_{test}} = [\mathbf{x_1}\ldots\mathbf{x_m}]^T$ and $\mathbf{x_{test}} \in \mathbb{R}^{m\times D}$. Training data in RFF domain $\mathbf{Z_{train}} = \sqrt{\frac{2}{a}}[cos(\mathbf{x_{train}W}).sin(\mathbf{x_{train}W})]$ and testing data in RFF domain $\mathbf{Z_{test}} = \sqrt{\frac{2}{a}}[cos(\mathbf{x_{test}W}).sin(\mathbf{x_{test}W})]$. $\mathbf{Z_{train}} \in \mathbb{R}^{n\times a}$ and $\mathbf{Z_{test}} \in \mathbb{R}^{m\times a}$ where n is number of training data and m is number of test data. According to [19] Random Fourier Feature is defined as : $\mathbf{z(x)} \equiv \sqrt{\frac{2}{a}}[\mathbf{cos}(\omega_\mathbf{1}^\mathbf{T}\mathbf{x} + \mathbf{b_1})\ldots\mathbf{cos}(\omega_\mathbf{a}^\mathbf{T}\mathbf{x} + \mathbf{b_a})]^\mathbf{T}$ or $\mathbf{z(x)} \equiv \sqrt{\frac{2}{a}}[\mathbf{cos}(\omega_\mathbf{1}^\mathbf{T}\mathbf{x})\ldots\mathbf{cos}(\omega_{\mathbf{a/2}}^\mathbf{T}\mathbf{x})\mathbf{sin}(\omega_\mathbf{1}^\mathbf{T}\mathbf{x})\ldots\mathbf{sin}(\omega_{\mathbf{a/2}}^\mathbf{T}\mathbf{x})]^\mathbf{T}$. These two equations are equivalent to each other provided the fact that $\mathbf{b_i}$'s are sampled from $U(0, 2\pi)$. a is number of RFF we want and is a variable parameter in our setting.

Dual solution : In dual form we get $\mathbf{K_{train}} = \mathbf{Z_{train}}\mathbf{Z_{train}}^T$ and $\mathbf{K_{test}} = \mathbf{Z_{test}}\mathbf{Z_{train}}^T$. So once we get $\mathbf{K_{train}}$, we solve for dual form $\mathbf{K_{train}} * \alpha = \mathbf{y_{train}}$. While inverting $\mathbf{K_{train}}$, $\mathbf{K_{train}}$ is a matrix with the form $\mathbf{Z_{train}}\mathbf{Z_{train}}^T$. $\mathbf{Z_{train}} \in \mathbb{R}^{n\times a}$ and $\mathbf{K_{train}} \in \mathbb{R}^{n\times n}$ where n is number of training data points and a is number of RFF . As a increases, the rank of $\mathbf{Z_{train}}$ increases. When a is below number of training data points, the Rank($\mathbf{Z_{train}}$) is at most same as number of RFF. Since Rank(AB) $\leq$ min(Rank(A), Rank(B)) , Rank($\mathbf{K_{train}}$) is at most number of

RFF. As a increases more, the rank of $\mathbf{K_{train}}$ increases till RFF becomes same as number of train data points. After this point the rank of $\mathbf{K_{train}}$ is at most number of training data point, even if we vary number of RFF as to be as high as 6 times number of training data. So, when a $\leq$ (number of train data point), $\mathbf{K_{train}}$ is rank deficient and when a $\geq$ (number of train data point), $\mathbf{K_{train}}$ is full rank. We need regularization for solving inverse of $\mathbf{K_{train}}$ in lower range i.e. $(\mathbf{K_{train}} + \lambda * \mathbf{Iden(n)}) * \alpha = \mathbf{y_{train}}$. Without regularization $\mathbf{K_{train}}\alpha = \mathbf{y_{train}} \Rightarrow \mathbf{Z_{train}}\mathbf{Z_{train}}^T\alpha = \mathbf{y_{train}} \Rightarrow \alpha^* = (\mathbf{Z_{train}}\mathbf{Z_{train}}^T)^{-1}\mathbf{y_{train}}$. Once we have solution, $\mathbf{K_{train}}\alpha^* = \mathbf{y_{train-est}} \Rightarrow \mathbf{Z_{train}}\mathbf{Z_{train}}^T\alpha^* = \mathbf{y_{train-est}} \Rightarrow \mathbf{Z_{train}}\mathbf{w}^* = \mathbf{y_{train-est}}$ and similarly $\mathbf{Z_{test}}\mathbf{w}^* = \mathbf{y_{test-est}}$. Here $\mathbf{w}^* = \mathbf{Z_{train}}^T\alpha^* = \mathbf{Z_{train}}^T(\mathbf{Z_{train}}\mathbf{Z_{train}}^T)^{-1}\mathbf{y_{train}}$. When directly solving dual form, we compute approximation error for $\mathbf{K_{train}}$ as $\frac{\|(\mathbf{K_{train-orig}} - \mathbf{K_{train}})\|_\mathbf{F} * \mathbf{100\%}}{\|(\mathbf{K_{train-orig}})\|_F}$, where $\mathbf{K_{train-orig}}$ is the original kernel matrix obtained from evaluating kernel function for pairwise data points and $\mathbf{K_{train}}$ is the kernel matrix using RFF. Approximation error for $\mathbf{K_{test}}$: $\frac{\|(\mathbf{K_{test-orig}} - \mathbf{K_{test}})\|_\mathbf{F} * \mathbf{100\%}}{\|(\mathbf{K_{test-orig}})\|_F}$, where $\mathbf{K_{test-orig}}$ is the original kernel matrix obtained from evaluating kernel function for pairwise data points and $\mathbf{K_{test}}$ is the kernel matrix using RFF. As #RFF increases, the error between original kernel matrix and RFF approximated kernel matrix decreases implying good approximation.

Primal solution :

Instead of solving the dual form , this problem can be solved in RFF space. Loss function is : $\|(\mathbf{Z_{train}}\mathbf{w} - \mathbf{y_{train}})\|_\mathbf{2}^\mathbf{2}$ where $\mathbf{Z_{train}} \in \mathbb{R}^{n \times a}$ and $\mathbf{w} \in \mathbb{R}^a$. Optimal $\mathbf{w}^* = (\mathbf{Z_{train}}^T\mathbf{Z_{train}})^{-1}\mathbf{Z_{train}}^T\mathbf{y_{train}}$. $\mathbf{C} := \mathbf{Z_{train}}^\mathbf{T}\mathbf{Z_{train}}$ where $\mathbf{C} \in \mathbb{R}^{a \times a}$.

When a is below number of training data points, the Rank($\mathbf{Z_{train}}$) is at most same as number of RFF and a is above number of training data points, the Rank($\mathbf{Z_{train}}$) is at most same as number of training data . Since Rank(AB) $\leq$ min(Rank(A), Rank(B)) , for a $\leq$ —training data—, Rank($\mathbf{C}$) is at most number of RFF making $\mathbf{C}$ almost full rank . As a increases more, the rank of C increases till number of RFF becomes same as number of

train data points. After this point the rank of $\mathbf{C}$ is at most number of training data point, even if we vary number of RFF as to be as high as 6 times number of training data and this makes $\mathbf{C}$ rank deficient. So, when a $\leq$ |train data point|, $\mathbf{C}$ is almost full rank and when a $\geq$ |train data points|, $\mathbf{C}$ is rank deficient. We need regularization for solving inverse of $\mathbf{C}$ in higher range i.e. $\mathbf{w}^* = (\mathbf{C} + \lambda * \mathbf{Iden(a)})^{-1}{\mathbf{Z_{train}}}^T\mathbf{y_{train}}$. Once we get solution, $\mathbf{y_{train-est}} = \mathbf{Z_{train}}\mathbf{w}^*$ and $\mathbf{y_{test-est}} = \mathbf{Z_{test}}\mathbf{w}^*$.

In both primal and dual setup, once we get the all predictions, we compute the L2 loss by $\sum_{i=1}^{n}\sum_{j=1}^{l}(\hat{y_i^j} - y_i^j)^2$. Here n is the number of data points in training data and l is the total number of options available for labels. We compute classification accuracy by calculating $argmax_j\hat{y_i^j}$ and matching it against the original label of data. Since we don't have multi-label data, the argmax returns just one label.

Model in summary : So in summary, we first consider a popular class of non-linear parametric models called *Random Fourier Features* (*RFF*) [19], which can be viewed as a class of two-layer neural networks with fixed weights in the first layer. The RFF model family $\mathcal{H}_N$ with N (complex-valued) parameters consists of functions $h\colon \mathbb{R}^d \to \mathbb{C}$ of the form

$$h(x) = \sum_{k=1}^{N} a_k \phi(x; v_k) \quad \text{where} \quad \phi(x; v) := e^{\sqrt{-1}\langle v, x\rangle},$$

and the vectors $v_1, \ldots, v_N$ are sampled independently from the standard normal distribution in $\mathbb{R}^d$. (We consider $\mathcal{H}_N$ as a class of real-valued functions with $2N$ real-valued parameters by taking real and imaginary parts separately.) Note that $\mathcal{H}_N$ is a randomized function class, but as $N \to \infty$, the function class becomes a closer and closer approximation to the Reproducing Kernel Hilbert Space (RKHS) corresponding to the Gaussian kernel, denoted by $\mathcal{H}_\infty$. While it is possible to directly use $\mathcal{H}_\infty$ (e.g., as is done with kernel machines [3]), the random classes $\mathcal{H}_N$ are computationally attractive to use when the sample size n is large but the number of parameters N is small compared to n.

Our learning procedure using $\mathcal{H}_N$ is as follows. Given data $(x_1, y_1), \ldots, (x_n, y_n)$ from $\mathbb{R}^d \times \mathbb{R}$, we find the predictor $h_{n,N} \in \mathcal{H}_N$ via ERM with squared loss. That is, we minimize the empirical risk objective $\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2$ over all functions $h \in \mathcal{H}_N$. When the minimizer is not unique (as is always the case when $N > n$), we choose the minimizer whose coefficients $(a_1, \ldots, a_N)$ have the minimum ℓ_2 norm. For problems with multiple outputs (e.g., multi-class classification), we use functions with vector-valued outputs and sum of the squared losses for each output.

1.3.2 Experiments

To demonstrate the double descent risk curve, we train a number of representative models on several widely used datasets that involve images, speech, and text.

Datasets : Table 1.1 describes the datasets we use in our experiments. These datasets are for classification problems with more than two classes, so we adopt the one-versus-rest strategy that maps a multi-class label to a binary label vector (one-hot encoding). For the image datasets—namely MNIST [15], CIFAR-10 [13], and SVHN [17]—color images are first transformed to grayscale images, and then the maximum range of each feature is scaled to the interval $[0, 1]$. For the speech dataset TIMIT [8], we normalize each feature by its

Table 1.1: Descriptions of datasets. In experiments, we use subsets to reduce the computational cost.

Dataset	Size of full training set	Feature dimension (d)	Number of classes
CIFAR-10	$5 \cdot 10^4$	1024	10
MNIST	$6 \cdot 10^4$	784	10
SVHN	$7.3 \cdot 10^4$	1024	10
TIMIT	$1.1 \cdot 10^6$	440	48
20-Newsgroups	$1.6 \cdot 10^4$	100	20

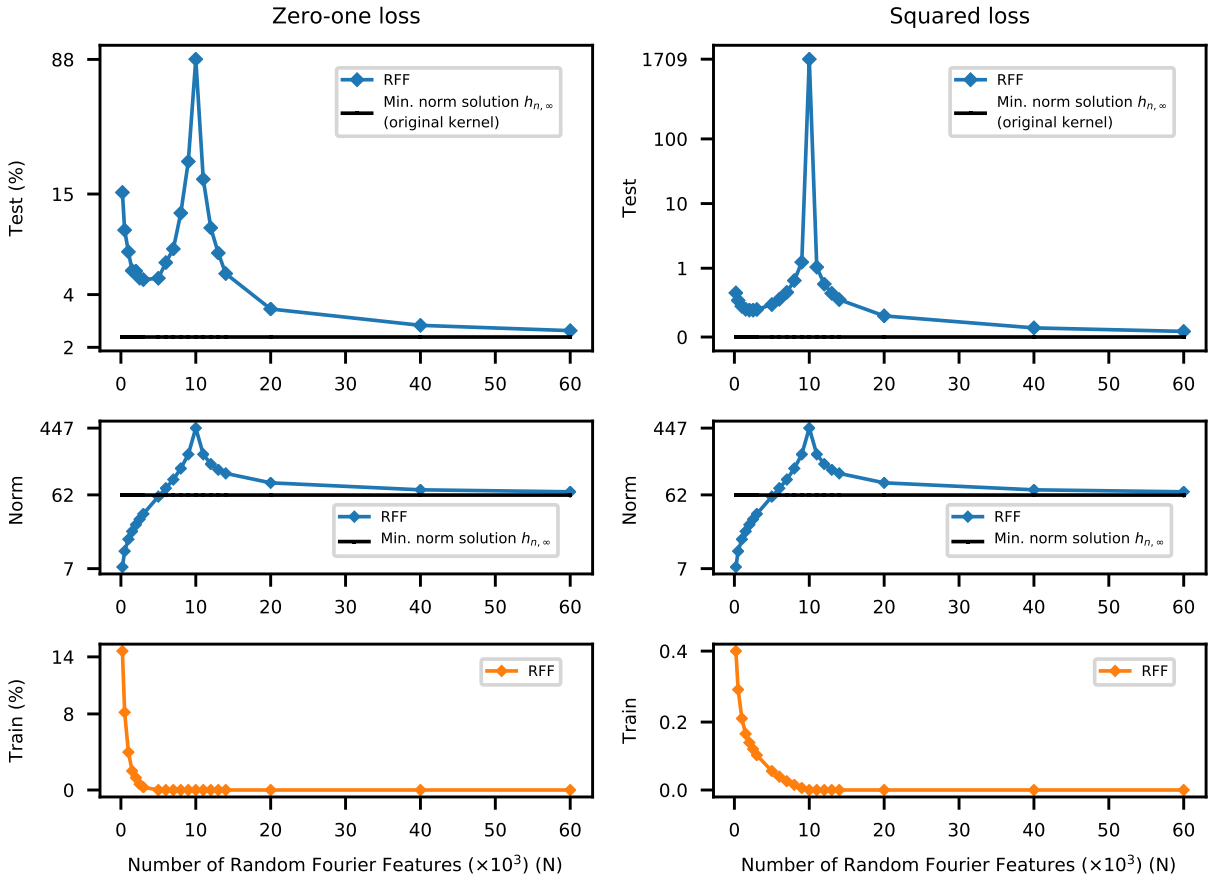

Figure 1.2: **Double descent risk curve for RFF model on MNIST.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the RFF model predictors $h_{n,N}$ learned on a subset of MNIST ($n = 10^4$, 10 classes). The interpolation threshold is achieved at $N = 10^4$.

z-score. For the text dataset 20-Newsgroups [14], we transform each sparse feature vector (bag of words) into a dense feature vector by summing up its corresponding word embeddings obtained from [18].

For each dataset, we subsample a training set (of size n) uniformly at random without replacement. For the 20-Newsgroups dataset, which does not have a test set provided, we randomly pick 1/8 of the full dataset for use as a test set.

Model training : For MNIST, each model is trained to minimize the squared loss on the given training set. Without regularization, such model is able to interpolate the training set when its capacity surpasses certain threshold (interpolation threshold). For comparison, we report the test/train risk for zero-one and squared loss. For rest of datasets we use zero-one loss only and show double descent behavior. The random feature vectors $v_1, \ldots, v_N$ are sampled independently from $\mathcal{N}(0, \sigma^{-2} \cdot I)$, the mean-zero normal distribution in $\mathbb{R}^d$ with covariance $\sigma^{-2} \cdot I$. The bandwidth parameter σ is set to 5, 5, 5, 0.1, and 16 for MNIST, SVHN, CIFAR-10, 20-Newsgroup, and TIMIT, respectively. For all the values of σ we observe same double descent behavior.

Results : In Figure 1.2, we show the test risk of the predictors learned using $\mathcal{H}_N$ on a subset of the popular data set of handwritten digits called MNIST. The same figure also shows the ℓ_2 norm of the function coefficients, as well as the training risk. We see that for small values of N, the test risk shows the classical U-shaped curve consistent with the bias-variance trade-off, with a peak occurring at the interpolation threshold $N = n$.

The interpolation regime connected with modern practice is shown to the right of the interpolation threshold, with $N \geq n$. The model class that achieves interpolation with fewest parameters ($N = n$ random features) yields the least accurate predictor. (In fact, it has no predictive ability for classification.) But as the number of features increases beyond n, the

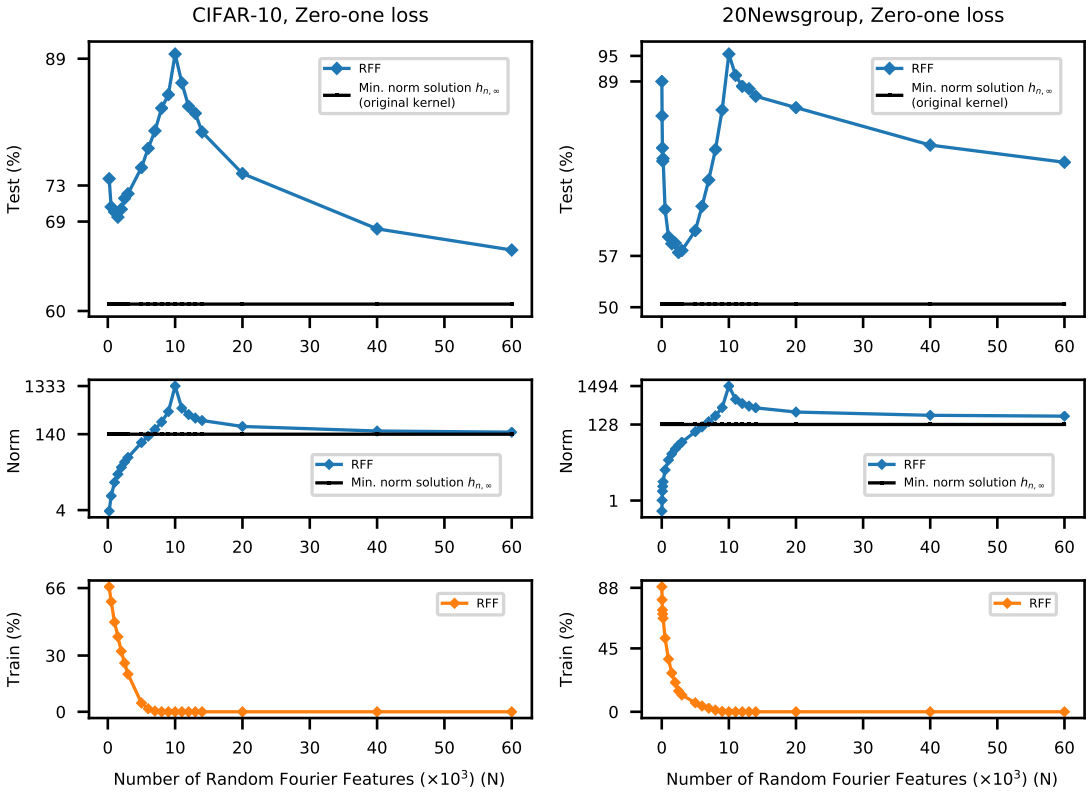

Figure 1.3: **Double descent risk curve for RFF model.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the RFF model predictors $h_{n,N}$ learned on subsets of CIFAR-10 and 20Newsgroups ($n = 10^4$). The interpolation threshold is achieved at $N = 10^4$.

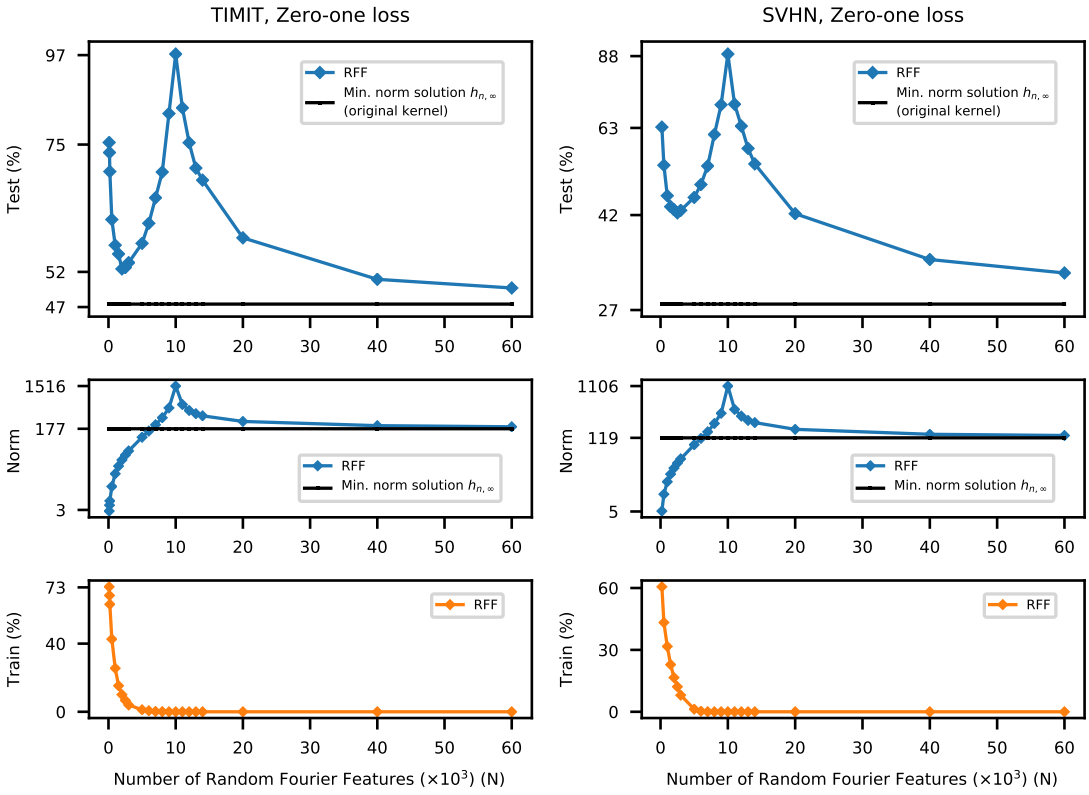

Figure 1.4: **Double descent risk curve for RFF model.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of RFF model predictors $h_{n,N}$ learned on subsets of TIMIT and SVHN ($n = 10^4$). The interpolation threshold is achieved at $N = 10^4$.

accuracy improves dramatically, exceeding that of the predictor corresponding to the bottom of the U-shaped curve. The plot also shows that the predictor $h_{n,\infty}$ obtained from $\mathcal{H}_\infty$ (the kernel machine) out-performs the predictors from $\mathcal{H}_N$ for any finite N.

What structural mechanisms account for the double descent shape? When the number of features is much smaller then the sample size, $N \ll n$, classical statistical arguments imply that the training risk is close to the test risk. Thus, for small N, adding more features yields improvements in both the training and test risks. However, as the number of features approaches n (the interpolation threshold), features not present or only weakly present in the data are forced to fit the training data nearly perfectly. This results in classical over-fitting as predicted by the bias-variance trade-off and prominently manifested at the peak of the curve, where the fit becomes exact.

To the right of the interpolation threshold, all function classes are rich enough to achieve zero training risk. The model continues to improve as function complexity grows.

Figure 1.3 illustrates double descent behavior for CIFAR-10 and 20Newsgroup. Figure 1.4 shows similar curves of zero-one loss for TIMIT and SVHN.

1.4 Random RELU Feature

1.4.1 Model

We show that the double descent risk curve also appears with Random ReLU feature networks [5]. Such networks are similar to the RFF models, except that they use the ReLU transfer function. Specifically, the Random ReLU features model family $\mathcal{H}_N$ with N parameters consists of functions $h\colon \mathbb{R}^d \to \mathbb{R}$ of the form

$$h(x) = \sum_{k=1}^{N} a_k \phi(x; v_k) \quad \text{where} \quad \phi(x; v) := \max(\langle v, x \rangle, 0).$$

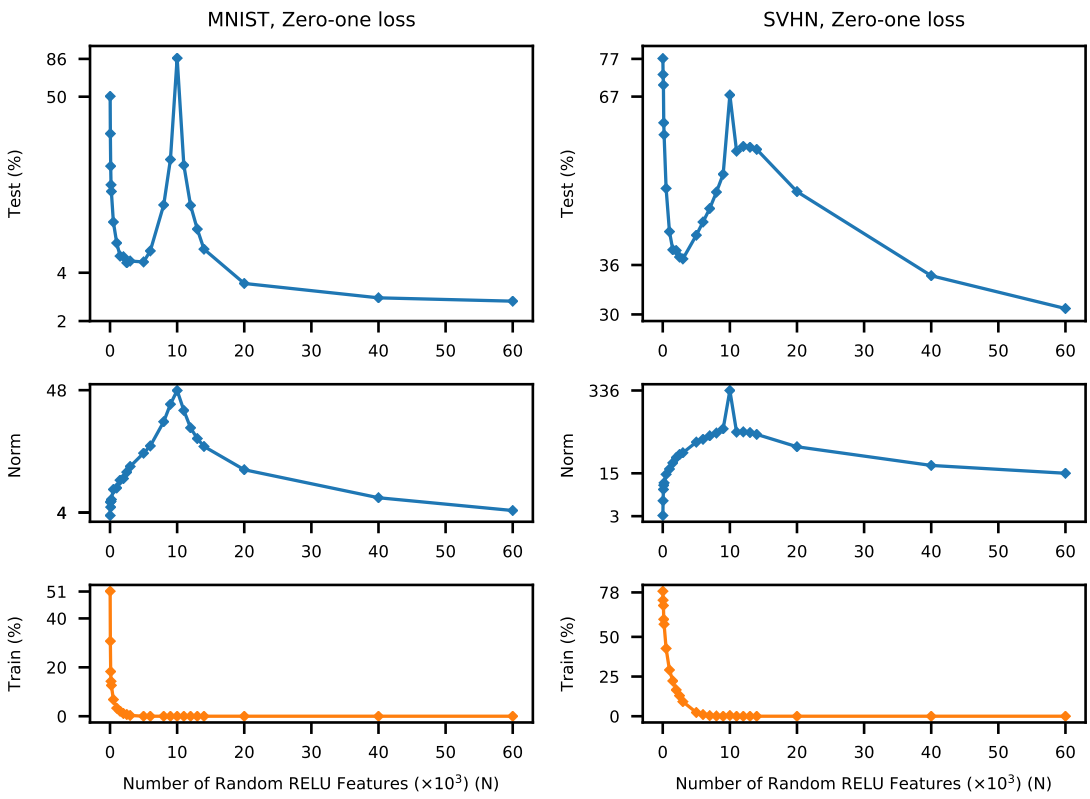

Figure 1.5: **Double descent risk curve for Random ReLU model.** Test risks (log scale), coefficient ℓ_2 norms (log scale), and training risks of the Random ReLU Features model predictors $h_{n,N}$ learned on subsets of MNIST and SVHN data ($n = 10^4$). The interpolation threshold is achieved at $N = 10^4$. Regularization of $4 \cdot 10^{-6}$ is added for SVHN to ensure numerical stability near interpolation threshold.

The vectors $v_1, \ldots, v_N$ are sampled independently from uniform distribution over surface of unit sphere in $\mathbb{R}^d$. The coefficients a_k are learned using linear regression. All details in Random-RELU is same as RFF except the RELU transformation instead of sinusoid.

1.4.2 Experiments

Figure 1.5 illustrates zero-one loss with Random ReLU features for MNIST and SVHN data. Ridge regularization with parameter $\lambda = 4 \cdot 10^{-6}$ is added in SVHN experiments to ensure numerical stability near the interpolation threshold. For MNIST experiments, no

regularization is added. We observe that the resulting risk curves and the norm curves are very similar to those for RFF.

1.5 Neural Networks and Backpropagation

1.5.1 Model

In our experiments, we use fully connected neural networks with a single hidden layer. To control the capacity of function class, we vary the number of hidden units. We use stochastic gradient descent (SGD) to solve the ERM optimization problem in this setting.

In general multilayer neural networks (beyond RFF or ReLU random feature models), a learning algorithm will tune all of the weights to fit the training data, typically using versions of stochastic gradient descent (SGD), with backpropagation to compute partial derivatives. This flexibility increases the representational power of neural networks, but also makes ERM generally more difficult to implement. Nevertheless, as shown in Figure 1.6, we observe that increasing the number of parameters in fully connected two-layer neural networks leads to a risk curve qualitatively similar to that observed with RFF models.

The computational complexity of ERM with neural networks makes the double descent risk curve difficult to observe. Indeed, in the classical under-parametrized regime ($N \ll n$), the non-convexity of the ERM optimization problem causes the behavior of local search-based heuristics, like SGD, to be highly sensitive to their initialization. Thus, if only suboptimal solutions are found for the ERM optimization problems, increasing the size of a neural network architecture may not always lead to a corresponding decrease in the training risk. This suboptimal behavior can lead to high variability in both the training and test risks that masks the double descent curve. Hence to smooth out this variability, we use weight reuse scheme. For smaller model, we save the weights and initialize larger model with these weights.

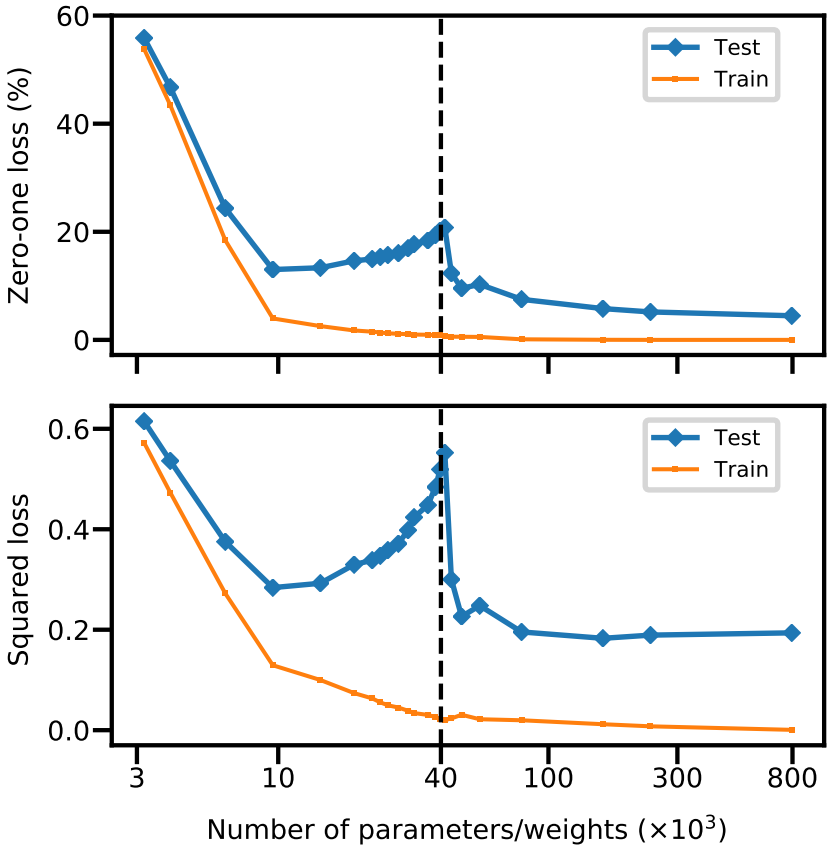

Figure 1.6: **Double descent risk curve for fully connected neural network on MNIST.** Training and test risks of network with a single layer of H hidden units, learned on a subset of MNIST ($n = 4 \cdot 10^3$, $d = 784$, $K = 10$ classes). The number of parameters is $(d + 1) \cdot H + (H + 1) \cdot K$. The interpolation threshold (black dotted line) is observed at $n \cdot K$.

1.5.2 Experiments

Model training : The ERM optimization problem in this setting is generally more difficult than that for RFF and ReLU feature models due to a lack of analytical solutions and non-convexity of the problem. Consequently, SGD is known to be sensitive to initialization. To mitigate this sensitivity, we use a "weight reuse" scheme with SGD in the under-parametrized regime ($N < n$), where the parameters obtained from training a smaller neural network are used as initialization for training larger networks. This procedure, detailed below, ensures decreasing training risk as the number of parameters increases. In the over-parametrized regime ($N \geq n$), we use standard (random) initialization, as typically there is no difficulty in obtaining near-zero training risk.

We now provide specific details below. We use SGD with standard momentum (parameter value 0.95) implemented in [6] for training. In the weight reuse scheme, we assume that we have already trained a smaller network with H_1 hidden units. To train a larger network with $H_2 > H_1$ hidden units, we initialize the first H_1 hidden units of the larger network to the weights learned in the smaller network. The remaining weights are initialized with normally distributed random numbers (mean 0 and variance 0.01). The smallest network is initialized using standard Glorot-uniform distribution [10]. For networks smaller than the interpolation threshold, we decay the step size by 10% after each of 500 epochs, where an epoch denotes a pass through the training data. For these networks, training is stopped after classification error reached zero or 6000 epochs, whichever happens earlier. For networks larger than interpolation threshold, fixed step size is used, and training is stopped after 6000 epochs.

Results : Fully connected neural network results have been shown in Figure 1.6 with weight reuse scheme in action. This shows double descent behavior for MNIST dataset. More additional experimental results for fully connected neural networks are shown in Figure 1.7.

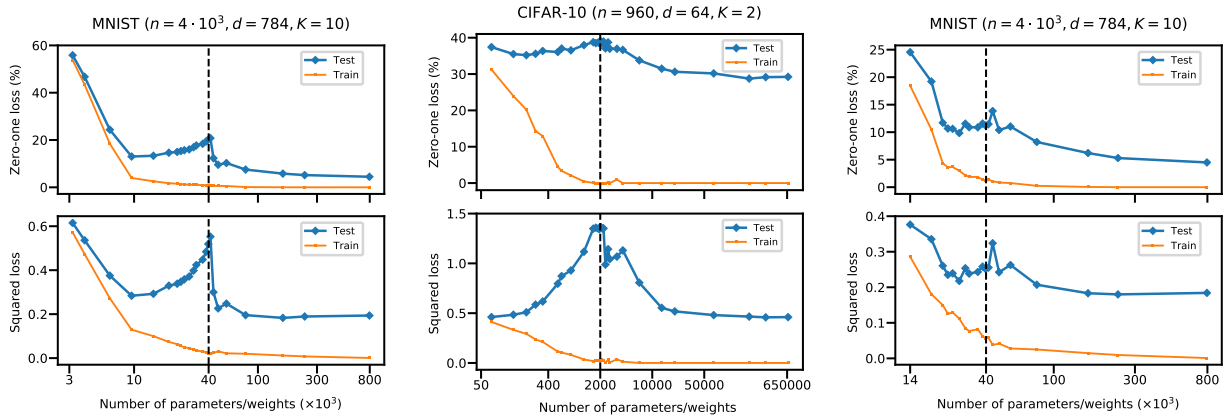

Figure 1.7: **Double descent risk curve for fully connected neural networks.** In each plot, we use a dataset with n subsamples of d dimension and K classes for training. We use networks with a single hidden layer. For network with H hidden units, its number of parameters is $(d+1) \cdot H + (H+1) \cdot K$. The interpolation threshold is observed at $n \cdot K$ and is marked by black dotted line in figures. (**a**) Weight reuse before interpolation threshold and random initialization after it on MNIST. (**b**) Same, on a subset of CIFAR-10 with 2 classes (cat, dog) and downsampled image features (8×8). (**c**) No weight reuse (random initialization for all ranges of parameters).

Results for MNIST and CIFAR-10 with weight reuse are reported in Figure 1.7(a) and Figure 1.7(b). Results for MNIST without weight reuse are reported in Figure 1.7(c). In this setting, all models are randomly initialized. While the variance is significantly larger, and the training loss is not monotonically decreasing, the double descent behavior is still clearly discernible.

1.6 Related Literature

Bias variance trade-off for machine learning models goes back to the day of statistical machine learning. There people used to deal with toy machine learning models. Most of the statisticians mainly dealt with as parameters increases how the combined effect of bias-variance vary. The classical work shows that U shaped generalization behavior. This has effected all models in various ways. Usage of regularization is one of the motivating factor

for trade-off. In linear regression case one can use p-norm of the solution to constraint the solution. Same goes true for logistic regression, SVM also. For decision tree and related algorithm, one can control the depth of trees and number of trees to achieve regularization. In Bayesian models one can use priors to act as regularizer. However, since the models were small and datasets were small, mostly there were theoretical works. With the advent of GPU and deep neural network and internet, comes real dataset and big models. To best of my knowledge this is first work which shows double descent behavior for modern day machine learning models.

1.7 Conclusion

It is common to use neural networks with extremely large number of parameters [4]. But to achieve interpolation for a single output (regression or two class classification) one expects to need at least as many parameters as there are data points. Moreover, if the prediction problem has more than one output (as in multi-class classification), then the number of parameters needed should be multiplied by the number of outputs. This is indeed the case empirically for neural networks shown in Figure 1.6. Thus, for instance, data sets as large as ImageNet [20], which has $\sim 10^6$ examples and $\sim 10^3$ classes, may require networks with $\sim 10^9$ parameters to achieve interpolation; this is larger than many neural network models for ImageNet [4]. In such cases, the classical regime of the U-shaped risk curve is more appropriate to understand generalization. For smaller data sets, these large neural networks would be firmly in the over-parametrized regime, and simply training to obtain zero training risk often results in good test performance [26].

Chapter 2: Information Extraction from Document

2.1 Introduction

Information extraction from document has been around for many years in research community. Given a document, we can extract various information like paragraph, triple, QA on these documents. A document can be in image format or language format. If the document is in image format, typically, these tasks are solved as joint vision and NLP extraction tasks. If the document is in text format, the we just extract the language from the document using standard tools and apply ML on these to perform several downstream jobs. The problem becomes far more interesting if the document is in image format and there is text inside them. We try to solve this problem. We assume that the vision part has been given to us. This is a reasonable assumption given the fact that OCR engine performances are very high these days. Now given the OCR engine output, we process the text using some algorithm. The end goal is to do information extraction from these documents.

Let's look at the overall flow. A data can be given as (x,y) where x is the document and y are the label(in our case information to be extracted). But we have plethora of documents, and do not have labels for them. This is typical data generation task that modern machine learning/AI systems face, and try to solve this. For example, in NLP community, given a paragraph, we can generate QA pair for solving question answer task. Researchers have tried to generate artificial QA pair to augment the original job of QA on NL. In vision, given

an image we can generate rules/heuristics to generate instance segmentation in an image. Label/data generation is a research problem in ML, and here we are facing same problem. Hence, we try to generate fields given an invoice/structured document.

At first we write rule based solution to extract labels with high precision. Since these rules have high precision, if they return empty for certain documents, we manually annotate them. At the end of this stage, 70% documents were labeled automatically by the rules, and rest 30% we manually annotated. This reduces the expensive, and time consuming manual annotation part of documents, and useful for business: they have lower yield time to deliver to the customer, lower cost because we need less manual annotation. Also, from research point of view, label generation for any data is an important task. This way we can generate data in unsupervised fashion. Once we have the data ready, i.e., both x and y are ready, we can train our favorite ML algorithm: deep neural network, support vector machine, BERT, Bayesian models etc. This model, given a document, rolls out the information to be extracted. For example, information like vendor name, invoice number, invoice date, total amount, tax amount, final date, address etc. are extracted by the model given a financial document.

2.2 Problem Statement

In this work we focus on financial documents. The financial documents are in image format, and there are texts inside them. Given a financial document, we want to extract invoice number, invoice date, total amount, tax amount, final date, address etc. In Figure 2.1, we show an example of a document and the information to be extracted from it; invoice number is INV02081, invoice date is 11/11/18, etc. We train a machine learning model on these documents. However, there is data annotation problem as data annotation is expensive. The basic pipeline is: we apply OCR on the document followed by machine

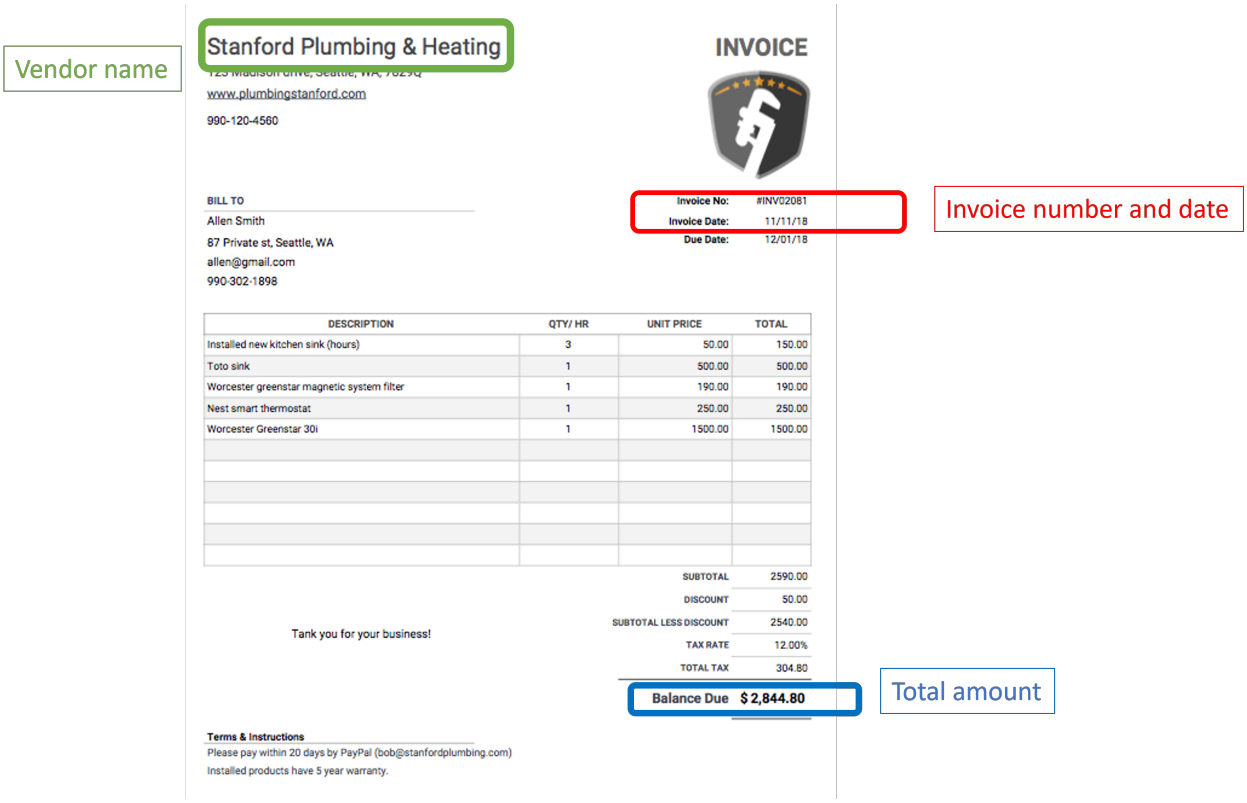

Figure 2.1: Information to be extracted from document.

learning algorithm to extract the information. OCR output is word with bounding box of coordinate information. We are trying to learn patterns on top of OCR output. One problem with OCR is that the output of this does not have any structure information that is present in the document. However, we can detect the structure information using some technique. Also another observation is that the vocabulary of the information to be extracted is not as varied as natural language. We have variations here, but they are pretty limited.

We believe a hybrid system with machine learning model and high precision rules is suitable for this job. But we don't have annotated data for ML model. Hence we need to collect data. But data collection is expensive, and in long run that is not a viable option. Hence can we develop high precision method for doing data annotation. This leads to the methods in next sections. Once we have the data, a large fraction of data is covered by these high precision rules, and rest documents where the high precision rules can not have a prediction, we annotate manually. Once the entire data is ready, we train any machine learning model and obtain desired result. As a by product of these high precision rules is that we can have a hybrid system of rule and machine learning model. But this is not our focus. We focus on data generation part for the document information extraction.

2.3 Proposed Solution

We now explain in details the proposed solution for the hybrid system. We construct high precision rules based upon spatial and language patterns in the data. These high precision rules help in efficient data augmentation and act as a first stage in the hybrid system. Once we obtain the data, we train a deep learning system on the overall data.

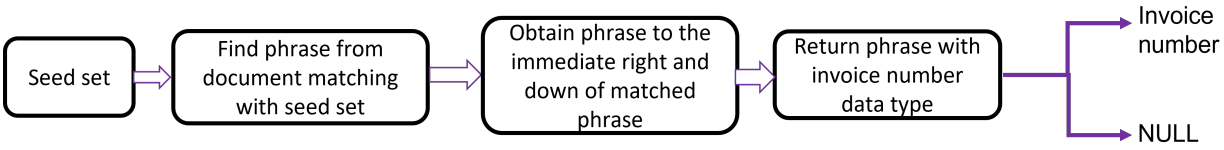

Figure 2.2: Invoice number extraction pipeline.

2.3.1 Invoice Number and Date

We now describe data generation for invoice number and date. As we mentioned before since the vocabulary variability for information is smaller compared to natural language, we have the flexibility to write high precision rules through which we can generate data.

In Figure 2.2 we show the overall diagram of the data generation of invoice number and date. We at forst obtain a good seed set of phrases that correspond to invoice number and date. Then we find the phrase (RBP) to the immediate right or down of this seed matched phrase. To make it high precision, we do a data type match of the RBP. For invoice number RBP should be a number, for date it should be of date data type. Here we use three aspects - the vocabulary variability for phrases is less, the structural information of phrases, data type of the information. These helps us to write high precision data generation.

2.3.2 Amount Detection

We now describe data generation for amount detection. As we mentioned before since the vocabulary variability for information is smaller compared to natural language, we have the flexibility to write high precision rules through which we can generate data.

Like the previous section, we at first obtain a good seed set of phrases that correspond to total amount. Then we find the phrase (RBP) to the immediate right, left or down of this seed matched phrase. To make it high precision, we do a data type match of the RBP. Here the data type will be money. If we don't have result out of this rule, we do partial

table column detection. For table column detection we start with highly occurring phrases in table and based upon that we detect the table rightmost column. Once we have found the rightmost column, maximum element out of this is the amount we are interested in. Similar to the date and number, here we use three aspects - the vocabulary variability for phrases is less, the structural information of phrases, data type of the information. In addition to these, we use partial table detection to arrive at the result. These helps us to write high precision data generation. These rules described above is very high precision, that is if they can say it they say with very high confidence. If they can not say, then they simply return null. Detecting vendor name is similar to detecting amount detection.

2.4 Experimental Results

We use data that consists of several thousands of documents from which we extract these information. We pass the documents through OCR engine and rule modules.

At first we manually annotate 500 documents, and then we pass these documents thorough the data generation rules to observe precision and coverage of these rule modules. The result is shown in In Figure 2.3 where we observe that with very high precision, we can generate the data for the information to be extracted. For example for the invoice number let's assume there are 1000 documents. Then for 870 documents the rules are able to generate the number, and for rest 130 document it returns empty string. For the 130 documents we do human annotation for obtaining the labels. Same arguments hold true for the rest of the information to be extracted.

Once we obtain all the data collected from the documents, we can train deep learning model (variant of BERT or so) on this data. Some advantages of this approach are : cost of expensive human annotation decreases, time to annotate decreases, yield time to customer decreases, and we can augment data to a ML/DL model to obtain better performance.

Field	Precision	Coverage
Invoice number	0.96	0.87
Invoice date	0.95	0.89
Amount Due	0.86	0.80
vendor name	0.91	0.72
Overall average	0.92	0.82

Figure 2.3: Precision and coverage for the rule modules on different fields.

2.5 Related Literature

There are variety of literature for this job. One line of research says that we can obtain a joint vision and language detection model on these. The advantage of this is that since model is trained joint, the error rate goes down. A downside of this approach is that we now need a very big model since vision and language part has to be handled at the same time. This is not good for maintainability and interpretibility. Another approach has started it's way into research community is optical character recognition based method. Since computer vision, especially OCR is a evolved and mature technology, with accuracy nearing more than 95%, it is easier for industrial settings to do detection on top of OCR output. Now one advantage if this method is that the developer or researcher need not focus on vision part of the model. Just machine learning on language part will suffice. However, a major drawback of this approach is that there is no structure information in the OCR output. All these leads to added complexity. Another few approaches are - fixed template-based

information extraction,and feature based Conditional Random Field (CRF) where we extract string features and do ML on top this [16, 23].

2.6 Conclusion

Information is an important task to solve, and data augmentation in this scenario is an important issue for machine learning models. Using the rules mentioned above one can generate labeled data to be augmented with original data for training machine learning model. In this way we can do better training of ML models. Also as a byproduct of this process, the high precision rules also act as an integral part of hybrid system (rule+ML based solution). We show on real dataset that the rule based system performs well although coverage is an issue.

Chapter 3: Mandates from Government Prediction Using COVID Data

3.1 Introduction

COVID 19 has affected all of us. It has led to many effects which are beyond our imagination. Due to different death rate, different state governments have closed the state/issued stay at home order/partial or full closing of state at different time. This decision can be taken by simply observing COVID confirmed/death/recovered cases or it can be taken in more complex way such as projection of possible cases in future and possible economic damage this closure order may cause. It is an optimization between death and financial damage. In this complex decision making part, demographic information from USA census data can be useful since different age group gets affected differently by this pandemic. Since now we know that this pandemic has social and economic consequences, here are some points. At first, let's look at some economic aspect of it. Economic aspect of pandemic: • Throughout the world different places have different industries, different businesses, different sources of income. Some places thrive on agriculture, while some other on technology industry. With in technology industry also there are differences – core sector (like mechanical, electrical, civil) and software/computer science sector. So, it is obvious that different industries are going to be affected differently by pandemic. Since different states have different major industries, they are going to be affected differently. For example, California or Washington in

USA is mostly dominated by software/internet companies while Michigan is dominated by car industry. So due to stay home order Michigan is going to be affected more by pandemic, and contrary to that California/Washington is not going to be that affected by pandemic money wise. Now coming to the question of which state should enforce stay at home order and when, depends on the job people do there. So different states might have different time frame to implement stay at home order. • Another part government should possibly consider is shut down different businesses at different time. We have different categories of business – high priority(medical/grocery), medium priority, and low priority. As a result, low priority businesses are going to be shut down first, gradually followed by medium priority businesses. What are the possible reasons for this gradual shutting down of businesses to ensure containment of disease and overall less financial loss. Since different states have different proportion of medium and low priority businesses, they will have different shutting down time. As a result, how people there are getting affected? Once pandemic is over how exactly governments are going to reopen the businesses? Social aspect of pandemic: • People, mostly above 60, have been affected by this pandemic. As in different states the number of infected people is on the rise, we have to be attentive to the primary health care worker's condition. Due to the outbreak and implementing public health measures, health workers often are getting infected which in return makes more pressure on health system. So, the governments should be more careful about the health workers. One way to ensure is give more proper PPE, hand sanitizer and other precautionary and protective agents to them. But initially, since different states have different rate of increase of pandemic, and there was shortage of precautionary and protective equipments, governments took different measures at different time. How did these different timing affect the overall health of health care workers is an social aspect we can possibly look into. • Another important aspect of this pandemic is the underlying drivers of fear, anxiety and stigma that fuel misinformation,

particularly through social media. Social media and internet are now a day one of the major sources of information. Due to fear, stigma people have been spreading misinformation regarding various aspects of pandemic. For example, there was a rumor that Coronavirus is not going to survive in hot and humid climate condition which neither does has any scientific basis nor empirical basis. So, such a misinformation must be addressed since it gives false hope to the people. Did governments take care of this misinformation? One way to know that it did is by reading the official statements or public briefing. Some selected group of persons also have been heckled as a source/originator of pandemic with out any valid reason. Did governments address this? If it did not what was the possible social implications. We can possibly get these information from crawling social media, news articles, and websites in general, make models to identify these misinformation, take a note of government policy on addressing these issues, if not addressed what were the implications. • Another social aspect can be unfortunate more death of the poor in society because they live in crowded space, don't have blue collar job where they can work from home, and inadequate money to survive cover health cost in case they gets infected. I think demography of states plays huge role in this. Governments should be extra careful where the per capita income is less, and people also live in crowded place. So how did governments handled this situation, what measures did they take to address this, and if there was delay in this procedure what was the effects on poor?

3.2 Problem Statement and Results

The task here is given any day's features for a state and day can we decide whether the government should have 'STAY AT HOME', 'OTHERBUSINESSCLOSE' , 'GATHRESTRICT10', 'GATHRESTRICTANY' order in that place or not. So, we have supervised binary classification task where the classification model is Logistic Regression, SVM with

RBF, and Random Forest. Once we have the result with full features, we remove one feature at a time, retrain the model, predict performance, and find difference of this performance with that of full feature. This is how we find feature importance. For Random Feature we also can have feature importance based upon contribution to entropy decrease – but we do not use this to consistently evaluate all kinds of classifiers. The features that was used are $ICUbed - mean$, $democrat$, $republican - vote - prct$, $newICU - mean$, $admis - mean$, $allbed - mean$, $icuover - mean$, $InvVen - mean$, $bedover - mean$, $total - claims$, $initial - claims$, and $total - claims - rate$. We also show that proactivity or reactivity of a state is important to predict the different decision we are trying to predict.

STAY AT HOME Government encouraged stay at home so that the pandemic does not spread. Table 3.2 shows stay at home order prediction from government of different US states for different models.

OTHER BUSINESS CLOSE Government form different states took precautionary measure so that essential business remains open like hospital, groceries, petrol pumps etc. On the other hand non essential business mostly restaurant, show business, and different other kinds should be shut down. Table 3.2 shows non essential business close orders prediction from government of different US states for different models.

Method	Precision	Recall	F1-score
Random forest	0.96	0.96	0.96
Logistic regression	0.75	0.75	0.75
SVM with RBF	0.97	0.97	0.97

Table 3.1: Stay at home order prediction for all states on each day.

GATHRESTRICT 10 There are states where government thought that may be gathering of 10 people won't be serious. Hence Table 3.2 shows no more than 10 people gather restriction orders prediction from government of different US states for different models.

GATHRESTRICTANY The states with very serious surge of COVID, government decided to shut down all movements of people. Hence Table 3.2 shows any number of people gather restriction orders prediction from government of different US states for different models.

3.2.1 Proactive and Reactive Nature of States

We have to decide some of the states as proactive and rest as reactive. Since we don't have ground truth, we have to do it in unsupervised way. The following algorithm shows how we defined proactive or reactiveness of a state. Few definitions: $D_{10percent}$ – the day when 10 percent of total death has happened, D_{sah} – the day when stay at home order has been issued . Algorithm:

1. Find difference (D) of 2 days ($D_{10percent}$, D_{sah}) from people dying curve for each state. Final output is in number of days. So, for each state we have delay days (D).

2. We define a threshold (T) on top of D to define a state as proactive or reactive. If $D < T$ then that state is proactive, else reactive. However, we don't know optimal T.

Method	Precision	Recall	F1-score
Random forest	0.95	0.95	0.95
Logistic regression	0.7	0.7	0.7
SVM with RBF	0.96	0.96	0.96

Table 3.2: Other business close prediction for all states on each day.

3. To find reasonable T we vary T and observe effect of it. For a given T, we cluster the states in 2 clusters using Gaussian Mixture Model (GMM). To do cluster we need features for state. The features for each state will be - summary statistics from ($D_{10percent+T}$) day to ($D_{10percent+T+30}$) day for total-death , total-hospitalization , total-unemploy and many more.

4. Since with each T, dataset of clustering is different, we get different clustering scores (for GMM we use AIC/BIC). For each T we get a clustering score. We take the $T_{optimal}$ = T where clustering score is minimum.

5. Since we got $T_{optimal}$, we assign proactive or reactive status for all states as 1/0 status.

6. Append this proactive/reactive status as feature. Run random forest model for several downstream classification tasks we hope to improve.

Final result comparison with and without proactive and reactive features The states with proactive and reactive feature shows improvement on the four decisions we are trying to make. Hence Table 3.2.1 shows results for stay at home, business close, any gather 10 and any gather decision from government of different US states for Random Forest model with and without proactive/reactive feature.

Method	Precision	Recall	F1-score
Random forest	0.95	0.95	0.95
Logistic regression	0.6	0.6	0.6
SVM with RBF	0.96	0.96	0.96

Table 3.3: Gathering of more than 10 people restriction prediction for all states on each day.

Method	Precision	Recall	F1-score
Random forest	0.96	0.96	0.96
Logistic regression	0.85	0.85	0.85
SVM with RBF	0.97	0.97	0.97

Table 3.4: Gathering of any people restriction prediction for all states on each day.

Classification task	With proactive/reactive feature	Without proactive/reactive feature
Stay at home	96.5	96.0
Business close	95.0	94.5
Any gather of 10	95.3	95.0
Any gather	96.5	96.0

Table 3.5: F1 score using Random forest with and without proactive and reactive feature.

Chapter 4: Misinformation Detection from Social Media

4.1 Introduction

With the advent of internet, initially there were very few people o the internet and mostly it was used for social good and used to portray nice side of people. It was mostly dominated by rich people doing nice things to connect the world. But then came information age. With the democratization of internet, and rapidly decreasing technological advancement to give almost free internet gave people access to the wonderful world of internet. It is a parallel universe where people can do almost anything these days. Our world has always revolved around good and bad deeds. With democratization came evil actors in the internet whose purpose is to spread rumor and fake news thereby corrupting the niceness of internet. Political propaganda, social unrest, maligning somebody, fake review of products to name a few of the misinformation types. These misinformation has both social and economic aspects associated to them. Propagating misinformation about a person can do social harm to the point of maligning and physical harm. Misinformation about a product can ruin a companies personal brand and do financial damage. Since these are detrimental to various aspects of society, they need to be curbed. Hence there has been proliferation of misinformation detection tasks on social media platforms like Facebook, Google, Twitter to name a few. These companies invest billions of dollars into making their platform misinformation free so that balance is maintained.

There has been different types of misinformation detection. Initially people just took raw tweet or social media post and classify it. One of the main modalities we can augment with text is graph of the internet. Graph is an essential part of internet based upon which internet was made. Hence misinformation detection on social media is very much dependant upon graph and text part. With the proliferation of image and text research one can imagine that image will be also a part of the misinformation where a corrupted text is added to right image or a corrupted image is added to a right text. Hence multimodal misinformation detection is slowly getting popular.

Given a misinformation dataset which is mainly natural language, we can classify a Transformer type model to arrive at a decent performance. However, if the graph component can be augmented wit this text based part, we can get better result. Hence we can pass the graph component to Graph Convolution Network (GCN) and get the embeddings out of it. Then we can augment BERT embeddings for the text with the GCN embeddings and obtain better result for misinformation detection (DD) job. However a very serious concern in this kind of research is that since multimodal components are getting added, how to do effective augmentation. A recent work has shown that the graph component can augmented to the self attention layer to find better model for graph+text type classification job. This work has been shown to have good performance for semantic parsing job where the relations from the database and NL to table mapping relations form graph relation and the classification job is to predict right first order logic form detection.

Misinformation detection also can be formulated as graph information with natural language information. The BERT with relational embedding mentioned above can be used as a new way of detecting disinformation.

4.2 Problem Statement

Misinformation in social media platform as Twitter is very concerning these days. In Twitter there is main user, followers of that main user and the tweet/main text/post that main user created or shared. Hence form social graph or relation point of view there are edge types - 'create' edge between main user and tweet, 'follow' edge between main user and followers, 'share' edge between followers of the main user and tweet. Since this relation information is available, we can pass this social graph through graph convolution network and get embeddings for the tweet nodes. This propagates the neighborhood graph information from few hops away and as a result of this the tweet node has information from the main user node and followers node. But a main disadvantage of this approach is that GCN [12] depends only on graph node embeddings - it does not take into consideration the relation types. Also if the nodes has some other context embedding, it does not know how to propagate that information through graph. Once we obtain the tweet node embeddings, we concatenate with BERT [7] embeddings and classify - which leads to the general question of merging embeddings together effectively.

A recent work RalationalBERT [25, 22] tries to the relation embedding along with natural language embeddings, and merge then together inside BERT. This model has flexibility to do natural language modeling, graph/relation information propagation, merging them together inside BERT model in one go. This type of model has shown improved performance on text to SQL job [25, 22]. In this work we investigate the efficacy of relational-BERT for disinformation detection job.

4.3 Experimental Results

For misinformation detection form social media, we test two different kinds data on different models. We test the misinformation detection performance using LSTM, BERT,

GCN+BERT and RelationalBERT model. We expect that RelationalBERT model should have the best result.

4.4 Future Direction

In the context of social media, different types of approaches has been adopted - text based, separate graph based. In this work we do model graph and text in same model. However, how to effectively integrate graph inside a language model is still an unsolved problem. In future we wish to investigate more on this. Multimodal disinformation detection is getting more prominence day by day. In future we also plan to investigate on this.

Bibliography

[1] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 541–549, 2018.

[2] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

[3] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[4] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[5] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pages 342–350, 2009.

[6] François Chollet et al. Keras. https://keras.io, 2015.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[8] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. *NIST speech disc*, 1-1.1, 1993.

[9] Gerd Gigerenzer and Henry Brighton. Homo heuristicus: Why biased minds make better inferences. *Topics in cognitive science*, 1(1):107–143, 2009.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[11] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*, volume 1. Springer, 2001.

[12] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[13] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[14] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings*, pages 331–339. Elsevier, 1995.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[16] Bodhisattwa Majumder, Navneet Potti, Sandeep Tata, James B. Wendt, Qi Zhao, and Marc Najork. Representation learning for information extraction from form-like documents. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 6495–6504, 2020.

[17] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop*, volume 2011, page 4, 2011.

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543, 2014.

[19] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pages 1177–1184, 2008.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[21] Ruslan Salakhutdinov. Deep learning tutorial at the Simons Institute, Berkeley, https://simons.berkeley.edu/talks/ruslan-salakhutdinov-01-26-2017-1, 2017.

[22] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[23] Sandeep Tata, Navneet Potti, James B. Wendt, Lauro Beltrao Costa, Marc Najork, and Beliz Gunel. Glean: Structured extractions from templatic documents. In *Proceedings of the VLDB Endowment*, pages 997–1005, 2021.

[24] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[25] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers, 2021.

[26] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.