Overparameterized classification problems: How many support vectors do I have, and do increasing margins bode well for generalization?

Adhyyan Narang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2020-116
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-116.html

May 29, 2020

Overparameterized classification problems: How many support vectors do I have, and do increasing margins always bode well for generalization?

by Adhyyan Narang

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Anant Sahai
Research Advisor

05/29/2020

(Date)

* * * * * * *

Professor Laurent El Ghaoui
Second Reader

5/28/2020

(Date)

Abstract

We study phenomena that arise during classification using linear and lifted models in overparameterized settings, presenting new perspectives on the work in Muthukumar et al. [19, 18]. To simulate real-world setups where only some of our features are actually useful, we consider the simplified 1-sparse model. We review a sharp characterization of generalization of min-ℓ_2 interpolation on Gaussian data [18].

In the hard-margin support vector machine (HM-SVM) problem, we show for the stylized ridge featurization that for a sufficient degree of "effective overparameterization", all training points become support vectors. We remark how this simple featurization captures the essential behavior in other featurizations as well. A consequence of this theorem is the HM-SVM problem is indistinguishable from the min-ℓ_2 binary interpolator. Along with prior work that gradient descent initialized at 0 on the squared loss converges to the min-ℓ_2 binary interpolator, and that gradient descent on the logistic loss converges to the HM-SVM, our work conveys that the choice of loss function does not have a large effect on the learned parameters in overparameterized settings.

In the regimes that the above theorem holds, we examine whether margin-based explanations for generalization are able to account for some observations for our problem. We observe for linear models that a) the resultant bounds on the probability of misclassification on a test point exceed 1 and are hence tautological and b) a model with a larger margin often does not have a better generalization performance. For kernel-inspired models, we investigate what the right normalization for margins is, and design two new notions of margin that are appropriate for these.

As a preliminary exposition of ongoing work, we examine the ramifications of our results for adversarial performance on the Fourier featurization. We discover, using visualizations that our learned function exhibits Gibbs-like behavior around jump discontinuities, and this causes adversarial examples to proliferate in the vicinity of training points.

Acknowledgements

I would like to thank my advisor, Prof. Anant Sahai for inspiring me to do research through EE16B, EECS189, EECS290S and for his generous mentorship and guidance over the past year. Not only did he help me find interesting topics to investigate, but also he showed me how to conduct good research and helped me build the confidence to perform independent explorations. I am very thankful to Prof. Laurent El Ghaoui for supporting me during my research journey, generously including me into his team of researchers, and advising me on many optimization-related topics.

I am grateful to Vidya Muthukumar for agreeing to mentor me despite it being an especially busy year for her. If it were not for her mentorship, I would not have had the opportunity to enroll in this MS program and learn and grow in the way that I have. She guided me through the process of doing theoretical research, showed me how to flesh out kernels of ideas into meaningful results, and supported me during difficult times. I would like to thank my other collaborators - Vignesh Subramanian, Prof. Daniel Hsu, Prof. Mikhail Belkin for all the inspiring discussions. Armin Askari's help over the past year has been instrumental to my progress, and I feel very fortunate to have received his mentorship.

My buddy, Yash Bhate always gave me company when I was busy working, and made sure that my days were lively and sunny. My housemates, Kenneth Lai, Micah Caroll and Peter Min Chung made Berkeley feel like home for me - without them, this Masters journey would have been incomplete. Lastly, I would like to thank my parents for their unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The "Unreasonable Effectiveness of Data" identifies how remarkable and surprising it is that so much of human behavior can be explained and predicted by data [12]. However, in recent times, data has also been unreasonably effective in another way: small amounts of data are able to successfully train large and complex models like neural networks! From the perspective of a practitioner, it would seem like a good thing that our learning systems are working better than we might predict from our theory. However, the pessimistic theorist might dampen the practitioner's enthusiasm by pointing out that our inability to predict the behavior of our deep learning systems has negative consequences: training neural networks often requires hours of ad-hoc tuning of hyperparameters and architectures in unpredictable ways, these models take many hours to train even on a GPU, and the learned models are susceptible to adversarial attacks and hence unreliable in sensitive settings. Perhaps this theorist can be more gently described as ambitious rather than pessimistic because (s)he believes that theoretically understanding our models can make them even more useful practically. This thesis aims to take some small steps towards this goal.

Theories of generalization for supervised learning problems assume that there is an underlying joint distribution $(x, y) \sim \mathcal{D}$, from which the training set and test set are drawn. Often it is also assumed that there is a true function f^* that maps the features to the labels. Hence, the training set can be thought of as a set of function-evaluations of f^*. Our goal is use this set to capture the essential structure underlying f^* within a learned model $\hat{f}$. If it is possible to use the evaluations of $\hat{f}$ to successfully perform some task (e.g regression, classification) on samples not contained within the training set, then we say that $\hat{f}$ generalizes well.

To find $\hat{f}$, a family of parameterized functions, $\mathcal{F}$ is first chosen and an optimization problem is then solved to determine what the best choice of parameters is. Folk wisdom from classical machine learning (ML) theory states that if our problem is *overparameterized* i.e the number of parameters of $\mathcal{F}$ is much greater than the number of training points provided, then our goal of identifying the salient structure of f^* is out of reach. However, despite being frequently overparameterized, deep learning problems often generalize well; this means that classical results are not helpful with explaining the remarkable success of deep learning. The success of overparameterized models can be recovered in linear and kernelized models as well, and these will be the setups considered in this thesis. Overparameterization is at the heart of these quandaries, and lies at the backdrop of each of the results presented in this thesis.

Often we know or assume information about f^* in a form other than function evaluations at specific points - for instance, we might know, that many of its parameters are zero (sparsity). We encode this extra information into the learning process as an *inductive bias* that encourages the learner to choose certain models over others. The inductive bias plays a very different role in

underparameterized and overparameterized cases.

In underparameterized setups, the training data is usually not perfectly explainable by the family of hypotheses models being considered. This means that the training error is usually positive, which allows it to act as a "razor" to differentiate between the hypotheses models. A concern here is that the training set does not capture the essence of the distribution from which the data is drawn, either because it contains noise (for example, due to errors in the data collection process) or that the natural variance of the data distribution would play a too-large role in shaping what the sampled training data looks like.

This is where the inductive bias comes in. By encoding our knowledge about f^* into our learning algorithm, we are able to overcome the aforementioned deficiencies of the training data. Often this is done by some kind of regularization - either by adding a regularization penalty term to the optimization objective or some other way. The partially-reliable training data along with the inductive bias allow the learner to select a model that generalizes well.

Contrarily, in the overparameterized case, there are multiple models that achieve zero or near-zero training error. It is often found that some of these models generalize well to the test set but others do not. The training data alone is unable to offer any help in differentiating between these good and bad models. Because of this, the inductive bias must play a larger role in this case to drive the model-selection process. To understand learning in overparameterized setups, we must thus understand what the right notions of inductive bias for our algorithms are.

Recent work has advocated for encouraging the learner to choose functions with large-margins as the right inductive bias to use for classification problems [2, 22, 15, 7]. But are large margins truly the panacea for all our theoretical conundrums regarding generalization in overparameterized classification problems? Chapter 3 adopts the position that while having large margins may indeed be a good sign for many problems, it perhaps does not deserve a status as a "one size fits all" explanation for generalization.

Another interesting question to ask is: Are all problems created equal in terms of how good the inductive bias needs to be? In Chapter 1.4, I recap a proof that, for certain stylized models, classification tasks are less sensitive to having the right inductive bias than regression problems.

In Chapter 2, I make an observation that for multiple different choices of featurizations, for high levels of overparameterization, the inequality constraints of the hard-margin SVM problem are likely to be held with equality. I prove this fact for a stylized featurization, which I argue captures the essence of the phenomenon.

What is the role of the inductive bias in learning adversarially robust solutions? What is the relationship between adversarial robustness and large margins? Answering these questions is the focus of Chapter 4. Before exploring these questions further, I review some core concepts from classical statistical theory in Section 1.1 and make some key distinctions between concepts in Section 1.2 that will help with presentation of the results.

1.1 Historical explanations for generalization

For a particular function class $\mathcal{F}$, *uniform convergence bounds* conservatively approximate the generalization error of $f \in \mathcal{F}$ by that of the least generalizable function in $\mathcal{F}$. For classification problems, the size/complexity of $\mathcal{F}$ is often measured by evaluating the VC dimension, which counts the number of possible labelings on finite datasets [24]. The Rademacher complexity is also often used which is a scale-sensitive measure that can also be used for regression problems [1]. When the complexity of $\mathcal{F}$ is small, then the approximation is good, as evidenced by the success of these bounds to explain the generalization of machine learning models in classical setups. A more detailed

exposition of these ideas can be found in Bousquet et al. [5].

However, when the complexity $\mathcal{F}$ is sufficiently large, then the approximation is worse and the quality of predictions of these bounds suffer. This was first brought into focus by examining the generalization of boosting: a type of ensemble-learning that combines the predictions from many primitive simpler models to generate the final prediction [21]. The main observation was that even after the training 0-1 loss became zero, increasing the number of primitive classifiers in the boosted model still reduced the test error. Since the model complexity is increasing here but the generalization error is reducing, VC-dimension style analysis is unable to offer an explanation for these observations. In Chapter 3, we recall how *margin-based explanations* were able to explain these seemingly surprising boosting results. And how these explanations are being employed to explain the surprising success of overparameterized learning (which includes deep learning) as well; we explore whether margins are explanatory in these cases as well.

1.2 Some key distinctions

1.2.1 Imitating a function v/s succeeding at a task

Often $\hat{f}$ succeeding at its task is conflated with the idea of $\hat{f}$ being either exactly the same or very similar to f^*. Consider the following examples when this may not be the case:

- **Regression:** Let f^* be some continuous and differentiable functions that is defined on the reals, $\mathbb{R}$. And
$$\hat{f}(x) = \begin{cases} \infty, \ x \in \mathbb{Z} \\ f^*(x), \ \text{otherwise} \end{cases}$$
Even though the test mean squared error would be exactly 0 for any continuous distribution, we would probably feel uneasy to claim that f^* and $\hat{f}$ are the same functions.

- **Binary classification:** Let our labels be obtained by $\mathsf{sgn}(f^*x)$. Let our predictions be obtained as $\mathsf{sgn}(\hat{f}(x))$, where $\hat{f}(x) = 100 \times f^*(x)$. On all test points, our labels and predictions would agree; however, f^* and $\hat{f}$ are clearly not the same function.

  In Chapter 4, we will observe a case where the signs of $\hat{f}$ and f^* similarly agree with high probability; however, there are probabilistically unlikely, but very frequent, inputs which are misclassified that can be exploited by an adversary.

1.2.2 Number of parameters v/s Number of features

In different families of parameterized models, the parameters may serve different purposes. Learning in feedforward neural networks can be thought of as a two-step process of a) first learning an appropriate featurization for the problem and b) learning a linear classifier on these learned weights. Usually, most of the parameters of these models are used for part (a) above. In neural networks, it is hence possible to increase the level of overparameterization without increasing the number of features or changing the dimensionality of the input data. Contrarily, for linear models, each parameter of the model is the weight placed on a given feature. Hence, to increase the number of parameters, the numbers of features utilized for prediction must be increased, which increases the input dimension; this plays a large role in the usefulness of margin bounds in Chapter 3.

To understand whether overparameterization is helpful with generalization and robustness tasks, it is instructive to think about the nature of these new features being added. Here, the taxonomy

of "useful/robust" features introduced by Ilyas et al. is helpful [14]. A feature is termed "useful" if it is positively correlated with the labels, and it is termed "robust" if it is not very sensitive to small changes in the inputs. The present work builds heavily on this perspective on features - the term "spurious features" is sometimes used in this thesis as an alternate terminology for "useless features".

In the 1-sparse data generation considered in Chapter 1.4, only one feature is correlated with the output whereas the other features are "useless"; overparameterization is thus detrimental to generalization performance because it includes more "useless" features.

The weak features introduced by Belkin et al. can be thought of as akin to the "useful but non-robust features" of Ilyas et al.. The choice of featurization in Definition 1.3.2 is inspired from here; since each of the features are correlated with the labels, increasing the level of overparameterization by including more features is helpful for generalization.

1.3 Setup

Throughout this thesis, unless otherwise explicitly stated, I use subscripts to index vectors and matrices: a_i is the i_{th} entry of vector a, b_i is the i_{th} row of matrix B, and B_{ij} is the ij_{th} entry of matrix B. As a matter of convention, I use lowercase to denote scalars and vectors and uppercase to denote matrices. I use Greek letters flexibly to denote scalars, vectors, matrices or functions. Let $\mathbf{1}$ denote the vector of ones and $\mathbf{0}$ denote the vector of zeros, with dimensions implicit; I_d is the $d \times d$ identity matrix. Let e_i denote the i_{th} canonical vector. Further, let $\mathcal{N}(\mu, \sigma^2)$ be the normal distribution with mean μ and standard deviation σ^2, let $U(s, e)$ denote the uniform distribution between start s and end e. The diag$(\cdot)$ function is defined such that if its input is a vector, it returns a diagonal matrix with the input vector's entries on the main diagonal; if it is provided a matrix, it returns a vector that contains the entries on the main diagonal of the input matrix.

1.3.1 Generative model for data

Let $S_{train} = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathcal{X} \times \{-1, +1\})^n$ represent our training dataset. Here, $x_i \sim \mathcal{D}$ are drawn i.i.d from a known distribution. Let

$$X_{\mathsf{train}} = [x_1, x_2 \ldots x_n]$$

be our training matrix and

$$y_{\mathsf{train}} = [y_1, y_2, \ldots y_n]^T$$

be our vector of training labels. Sometimes, $x_i \in \mathbb{R}$ is lifted to a higher dimension and otherwise, $x_i \in \mathbb{R}^d$; hence, the convention above may abused and X_{train} may refer to either a vector or a matrix depending on context.

In each of our cases, we consider $\phi \colon \mathcal{X} \to \mathbb{R}^d$ is a **known** feature-map, and let the featurization of a given point x be indexed as $\phi(x) = \begin{bmatrix}\phi_1(x) & \ldots & \phi_d(x)\end{bmatrix}^T$ for $x \in \mathcal{X}$. Define shorthand for our lifted training matrix

$$\phi_{train} = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_i)^T \\ \vdots \\ \phi(x_n)^T \end{bmatrix}$$

The data is generated by an **unknown** target linear function $f^*(x) = \langle \alpha^*, \phi(x) \rangle$, parameterized by unknown weights $\alpha^* \in \mathbb{R}^d$. Define $Z \in \mathbb{R}^n$ as

$$z_{\mathsf{train}} = [z_1 \ \dots \ z_n] = [f^*(x_1), f^*(x_2) \dots f^*(x_n)]^T.$$

The labels are obtained as $y_i = sgn(z_i)$. We define a linear function with arbitrary weights $\widehat{\alpha}$ as $f_{\widehat{\alpha}}(x) = \langle \widehat{\alpha}, \phi(x) \rangle$, and as a shorthand, we denote f_{α^*} by f^*.

Analogues of the above objects at test time are denoted as X_{test}, y_{test}, ϕ_{test} and z_{test}.

1.3.2 Different choices for $\mathcal{D}$ and ϕ

In different sections of this thesis, I consider different choices of featurizations and data distributions. Here, I list each of these. For the Gaussian and Weak features, there is no lifting.

Definition 1.3.1 (Gaussian features)**.** *Let $x_i \sim \mathcal{N}(0, I_d)$ for $i \in \{1, \dots, n\}$. Since there is no lifting, we have $\phi_{\mathsf{train}} = X_{\mathsf{train}}$.*

The above featurization is used most prominently in future Chapters to provide theoretical characterization of phenomena.

Definition 1.3.2 (Weak features)**.** *for $i \in \{1, \dots, n\}$, we have U_i i.i.d. $\sim$ Unif$(-1, 1)$ and W_i i.i.d. $\sim \mathcal{N}(0, I_d)$.*

$$x_i = U_i \mathbf{1} + W_i. \tag{1.1}$$

Again, since there is no lifting, we have $\phi_{\mathsf{train}} = X_{\mathsf{train}}$.

This definition is inspired by Belkin et al., where a similar featurization was used by the authors to exhibit the phenomenon of *double-descent*: here, if $d > n$, increasing d is beneficial to generalization.

For the two lifted featurizations below, we consider a feature-map $\phi_{\mathsf{fourier}}(x) : \mathbb{R} \to \mathbb{R}^d$:

$$\phi_{\mathsf{fourier}}(x) = [1, \sin(x), \cos(x) \ \dots \ \sin\left(\frac{d-1}{2}x\right), \cos\left(\frac{d-1}{2}x\right)]^T$$

Definition 1.3.3 (Fourier features, regularly spaced train)**.** *We have $x_i = 2 \times \frac{i}{n} - 1$ and*

$$\phi(x) = \phi_{\mathsf{fourier}}(x)$$

Definition 1.3.4 (Fourier features, uniform train)**.** *We have $x_i \sim U(-1, 1)$ and*

$$\phi(x) = \phi_{\mathsf{fourier}}(x)$$

These Fourier featurizations are often used for empirical illustrations. Their usage finds a special relevance in Chapter 4 when investigating adversarial examples. Most of the phenomena under discussion were first discovered in experimentation with Fourier features.

1.3.3 Feature weighting

In addition to having a host of featurizations that we can experiment with, it is useful to be able to adjust the variance of each of the columns. To achieve this end, we perform feature weighting. We define a diagonal matrix $\Sigma = \text{diag}(\lambda_1, \lambda_2 \ \ldots \ \lambda_n)$ where λ_i is the weight on the i_{th} column. Then, if ϕ_{train} and ϕ_{test} are the train and test matrices without weighting respectively, we apply the weighting to each of these to obtain: $\phi_{\mathsf{train}} = \Sigma\phi_{\mathsf{train}}$ and $\phi_{\mathsf{test}} = \Sigma\phi_{\mathsf{test}}$. Two schemes for generating Σ are provided below.

As a sidenote, weightings are sometimes considered as modifying the optimization objective at training time and not as part of the covariates themselves i.e instead of minimizing $\|\alpha\|_2$, we minimize $\|\Sigma\alpha\|_2$. However, this would mean that Σ does not influence the test-time data generation at all; hence, while closely related, this perspective is nevertheless distinct from the one being used here.

These weighting schemes are illustrated in Figure 1.1 and Figure 1.2: λ_i is plotted against i to visualize how the parameters influence the resultant weights. For the bilevel weighting definition below, the parameter q controls how much the early features are preferred.

Definition 1.3.5 (Bilevel weighting)**.** *The weighting scheme is parameterized by $p > 1, 0 < q < 1$ and $0 < r < 1$.*

$$\lambda_j = \begin{cases} \frac{\gamma d}{s}, & 1 \leq j \leq s \\ \frac{(1-\gamma)d}{d-s}, & \textit{otherwise.} \end{cases}$$

In particular, this ensemble sets parameters

$$\begin{aligned} d &:= n^p \\ s &= n^r \\ \gamma &= n^{-q}, \end{aligned}$$

Definition 1.3.6 (Polynomial decay)**.**

$$\lambda_k = \frac{1}{k^m} \ \textit{ for all } k \in \{1, 2, \ldots, d\} \tag{1.2}$$

1.3.4 Risk

Definition 1.3.7 (Regression Risk)**.** *The regression risk of a function $f_{\widehat{\alpha}}$ is the function $\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$:*

$$\mathcal{R}(\widehat{\alpha}) = \mathop{\mathbb{E}}_{x}[\|f^*(x)) - f_{\widehat{\alpha}}(x)\|_2^2]$$

Definition 1.3.8 (Classification Risk)**.** *The classification risk of a function $f_{\widehat{\alpha}}$ is the function $\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$:*

$$\mathcal{C}(\widehat{\alpha}) = \mathop{\mathbb{E}}_{x}[\mathbb{I}(sgn(f^*(x)) = sgn(f_{\widehat{\alpha}}(x)))]$$

1.3.5 Learning algorithms

Definition 1.3.9 (Binary min-ℓ_2 Interpolator)**.** *We define*

$$\begin{aligned} \widehat{\alpha}_{\mathsf{binary}} &:= \mathop{\arg\min}_{\alpha \in \mathbb{R}^d} \|\alpha\|_2 \\ s.t \ \ &\phi_{\mathsf{train}}\alpha = y_{\mathsf{train}} \end{aligned}$$

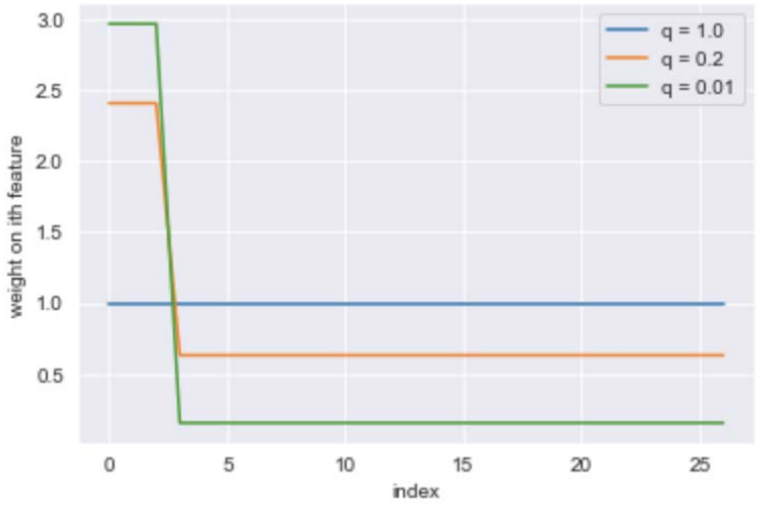


Figure 1.1: Illustration of the bilevel weighting scheme from Definition 1.3.5 for different choices of parameter q. As q increases, the weighting becomes more equal. Since the sum of λ_i's are fixed to always equal d, the area under each of these curves is a constant.

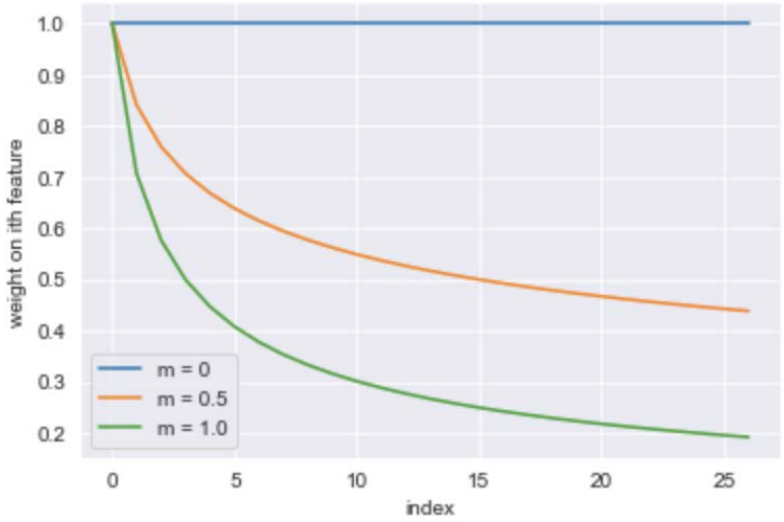


Figure 1.2: Illustration of the polynomial decay weighting scheme from Definition 1.3.6. As the parameter m increases, the weighting becomes less equal and more concentrated on early features.

Definition 1.3.10 (Real min-ℓ_2 Interpolator)**.**

$$\widehat{\alpha}_{\mathsf{real}} := \underset{\alpha \in \mathbb{R}^d}{\arg\min} \|\alpha\|_2$$
$$s.t \;\; \phi_{\mathsf{train}} \alpha = z_{\mathsf{train}}$$

Definition 1.3.11 (Support vector machine)**.**

$$\widehat{\alpha}_{\mathsf{SVM}} := \underset{\alpha \in \mathbb{R}^d}{\arg\min} \|\alpha\|_2$$
$$s.t. \quad y_i f_\alpha(x) \geq 1 \quad for \; i = 1, \ldots, n.$$

1.4 Classification is easier than regression

While the phenomena explored in later chapters are idiosyncratically classification phenomena, I first contrast classification with regression to bring relevant attributes of classification into sharp focus [18].

1.4.1 Analysis of Generalization for Gaussian features

1.4.2 Result

Assumption 1 (1-sparse linear model)**.** *Recall that the bi-level ensemble sets $s := n^r$. For a given*[1] *$t \in \{1, \ldots, s\}$, we assume that $\alpha^* = \frac{1}{\sqrt{\lambda_t}} \cdot e_t$, i.e. the parameter vector α^* is 1-sparse.*

Theorem 1. *Assume that the true data generating process is 1-sparse (Assumption 1). For the bi-level covariance matrix model, the limiting classification and regression error of the minimum-ℓ_2-norm interpolation (of binary labels and real labels respectively) converge in probability, over the randomness in the training data, as a function of the parameters (p, q, r) in the following way:*

1. *For $0 \leq q < (1 - r)$, we have*

$$\lim_{n\to\infty} \mathcal{R}(\widehat{\alpha}_{\mathsf{real}}; n) = 0,$$
$$\lim_{n\to\infty} \mathcal{C}(\widehat{\alpha}_{\mathsf{binary}}; n) = 0.$$

   In this regime, both regression and classification generalize well.

2. *For $(1 - r) < q < (1 - r) + \frac{(p-1)}{2}$, we have*

$$\lim_{n\to\infty} \mathcal{R}(\widehat{\alpha}_{\mathsf{real}}; n) = 1,$$
$$\lim_{n\to\infty} \mathcal{C}(\widehat{\alpha}_{\mathsf{binary}}; n) = 0.$$

   In this regime, classification generalizes well but regression does not.

3. *For $(1 - r) + \frac{(p-1)}{2} < q \leq (p - r)$, we have*

$$\lim_{n\to\infty} \mathcal{R}(\widehat{\alpha}_{\mathsf{real}}; n) = 1,$$
$$\lim_{n\to\infty} \mathcal{C}(\widehat{\alpha}_{\mathsf{binary}}; n) = \frac{1}{2}.$$

   In this regime, the generalization is poor for both classification and regression.

1.4.3 Key to proof: Survival, Contamination

The proof for Theorem 1 is detailed in the journal presentation of these results [18]. Here, I introduce the notions of survival and contamination from [19] and highlight the key takeaways from the proof.

The 1-sparsity assumption that we have made on the unknown signal enables us to do this as a function of natural quantities corresponding to the preservation of the true feature (*survival*)

[1] The intuition for this condition, also motivated in prior analyses of minimum-ℓ_2-norm interpolation [19], is that for any reasonable preservation of signal, the true feature needs to be sufficiently preferred, therefore weighted highly.

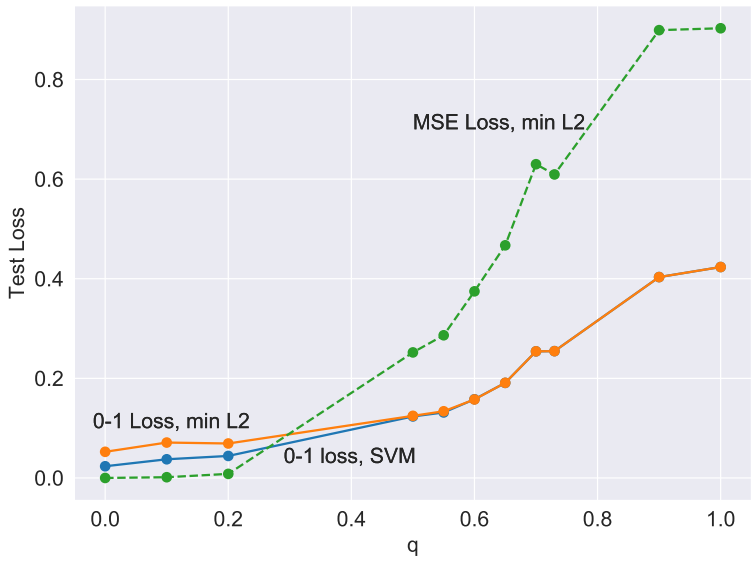


Figure 1.3: Comparison of test classification and regression error on solutions obtained by minimizing different choices of training loss on the bi-level ensemble ($n = 529, d = 12167$ fixed, parameters $(p = 3/2, r = 1/2)$ fixed). The dashed green curve corresponds to $\widehat{\alpha}_{2,\mathsf{real}}$ (Equation 1.3.10), the orange curve corresponds to $\widehat{\alpha}_{2,\mathsf{binary}}$ (Equation 1.3.9), and the solid blue curve corresponds to $\widehat{\alpha}_{\mathsf{SVM}}$ (Definition 1.3.11).

and the pollution due to false features (*contamination*). In terms of the language from Ilyas et al. brought up in Chapter 1, the survival can be thought of as the weight on the "useful" feature and the contamination is a measure of the total weight on the "useless" features. If we assume that the real labels are generated by the t^{th} feature, α_t^*, then we can define these quantities for any solution $\widehat{\alpha}$. First, as classically observed in statistical signal processing, the estimated coefficient corresponding to the true feature α_t^* will experience *shrinkage* and be attenuated by a factor that we denote as *survival*. From Assumption 1, we defined $\boldsymbol{\alpha^*} := \frac{1}{\sqrt{\lambda_t}} \cdot_t$, and so we have

$$\mathsf{SU}(\widehat{\alpha}, t) = \frac{\hat{\alpha}_t}{\alpha_t^*} = \sqrt{\lambda_t}\hat{\alpha}_t \tag{1.3}$$

Second, we have the *false discovery of features*. We measure the effect of this false discovery for prediction on a test point X by a *contamination* term:

$$B = \sum_{j=1, j\neq t}^{d} \widehat{\alpha}_j \boldsymbol{\phi}_j(\mathbf{X}). \tag{1.4}$$

Recall that X is random, and the features $\phi(X)$ are zero-mean. Therefore, B is a zero-mean random variable. Accordingly, we can define the standard deviation of the contamination term on a test point as below:

$$\begin{aligned} \mathsf{CN}(\widehat{\alpha}, t) &= \sqrt{[B^2]} \\ &= \sqrt{\sum_{j=1, j\neq t}^{d} \lambda_j \widehat{\alpha}_j^2}. \end{aligned} \tag{1.5}$$

where the last step follows from the orthogonality of the d features.

In general, the larger the survival and the smaller the contamination, the better the generalization performance is. In regime 1 above, the survival is large enough and contamination is small enough that both regression and classification work. In regime 2, the survival is less than regime 1 - this makes regression fail but classification still works. However, if the survival reduces even further, then both regression and classification fail; this happens in regime 3.

Chapter 2

How many support vectors?

I restate here the hard-margin SVM problem from Definition 1.3.11 in Chapter 1 for easy reference.

$$\begin{aligned} p^* &= \min_{\alpha \in \mathbb{R}^d} \|\alpha\|_2 \\ \text{s.t.} \quad & y_i f_\alpha(x_i) \geq 1 \quad \text{for } i = 1, \ldots, n. \end{aligned}$$

For some training points, $y_i f_\alpha(x_i)$ must be exactly one; otherwise the objective could be reduced without violating any constraints. These points are called *support vectors*. Define the subset of training points that are support vectors as $M : \{x : y_i f_\alpha(x_i) = 1$. When I refer to *the fraction of support vectors*, I am referring to the ratio $F = \frac{|M|}{n}$.

What does it mean for a training point to be a support vector? In Chapter3, we will interpret the quantity $y_i f_\alpha(x_i)$ is a measure of the confidence of prediction on a training point; the support vectors are those training points for which our predictions are least confident. Here, we show that if our optimizer places enough weight on spurious features, then all our training points are likely to become support vectors i.e $\forall(i, j)\ y_i f_\alpha(x_i) = y_j f_\alpha(x_j) = 1$.

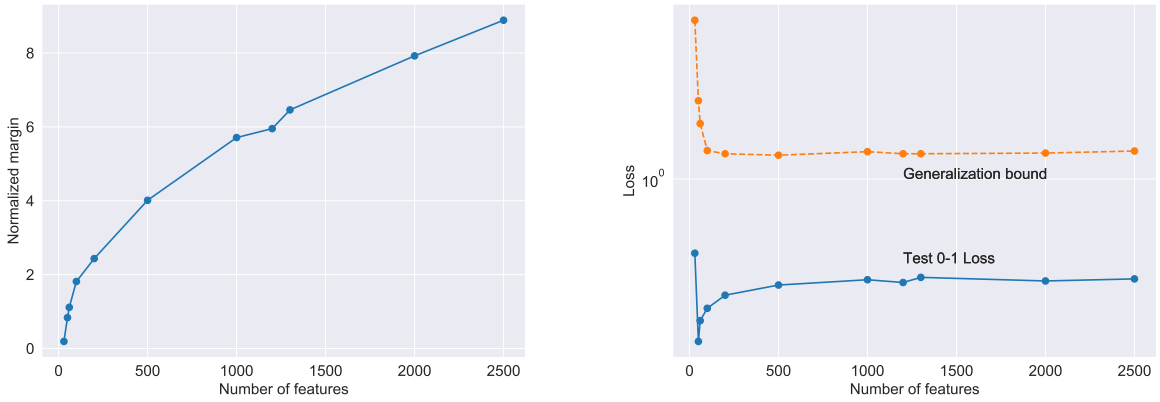
2.1 Empirical observations and their significance

Consider the following fascinating empirical observations. In Figure 2.1, we find that for fixed n and no feature weighting, the fraction of support vectors steadily increases as we increase d. Further, as we make the features more equally weighted for n and d both fixed, the fraction of support vectors increases. The effect from Figure2.1(a) is replicated for Fourier features in 2.2. It certainly does seem that as we increase the "effective overparameterization", the fraction of support vectors increases.

Why might this be an interesting finding? As motivated in Chapter 1, understanding the inductive bias used by our algorithms is of crucial importance in overparameterized settings. Usually this inductive bias is encoded into our algorithm by means of a regularization term in our objective function; however, recent works [15, 22, 11, 25, 20] show that gradient descent also has an "implicit bias" that makes it prefer certain solutions over others. We learn that:

1. If we minimize the logistic loss using gradient descent on separable training data[1], we will converge to the hard-margin SVM [15, 22].

[1]The implicit bias has also been characterized for the more difficult non-separable case [15], but we focus here on separable data as this will always be the case for an overparameterized setting.



(a) The features are isotropic Gaussian with no feature weighting, as in definition 1.3.1.

(b) The features are generated using the bilevel covariance scheme in Definition 1.3.5

Figure 2.1: Fraction of support vectors for Gaussian features. As d is increased, and as feature weighting becomes more equal across features, the fraction of support vectors increases.

2. If we minimize the squared loss on training data using an overparameterized model, we will converge to the minimum-ℓ_2-norm interpolation [8, Theorem 6.1] *provided the initialization is equal to zero.*

In Figure 2.3, these results are represented by the left and right vertical arrows. The empirical results above suggest something new: if all training points are support vectors, $\widehat{\alpha}_{\mathsf{SVM}}$ and $\widehat{\alpha}_{\mathsf{binary}}$ are equivalent as well. This is represented by the horizontal arrow in Figure 2.3, and proving this for a specialized featurization will be the task of the next section. In Muthukumar et al., we proved the result for Gaussian features; however, I choose a different simplistic featurization here to bring the reason for proliferating support vectors into sharp focus [18].

2.2 Setup

Here, $x_i \in \mathbb{R}$ is a scalar and $X_{\mathsf{train}} = [x_1, x_2 \dots x_n]^T$ is a column vector.

Definition 2.2.1 (Ridge features, uniform train)**.** *This featurization is parameterized by λ. We have $x_i \sim U(-1, 1)$ and*

$$\phi(x_i) = [x_i, \lambda e_i^T]$$

Using this definition of ϕ, we can write our lifted train matrix as:

$$\phi_{\mathsf{train}} = [X_{\mathsf{train}} \, | \lambda I_n]$$

Further, assume that $\alpha^* = e_1$. This means that $y_i = \mathsf{sgn}(x_i)$. Only the first feature matters and the others can be thought of as *spurious* features.

This choice of featurization is interesting because each spurious feature has the special property that it only influences the prediction for a single training point. Further, these features have a connection to Tikhonov regularization; Muthukumar et al. show that the minimizer of the squared loss with a Tikhonov regularization term on the original data is equivalent to the binary interpolator, $\widehat{\alpha}_{\mathsf{binary}}$ if ridge features are appended to the data [19].

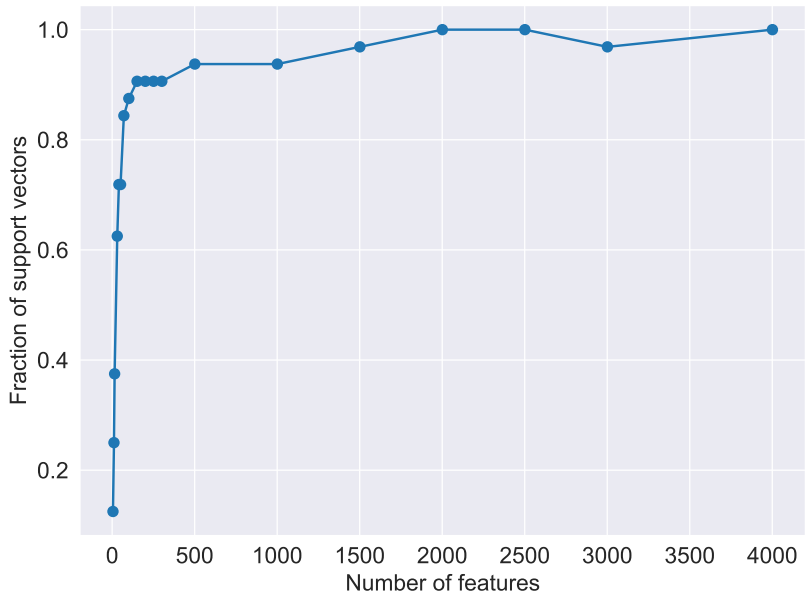


Figure 2.2: Fraction of support vectors for uniformly sampled Fourier features (Definition 1.3.4). The number of training points, $n = 32$ is fixed. As d is increased, the fraction of support vectors increases.

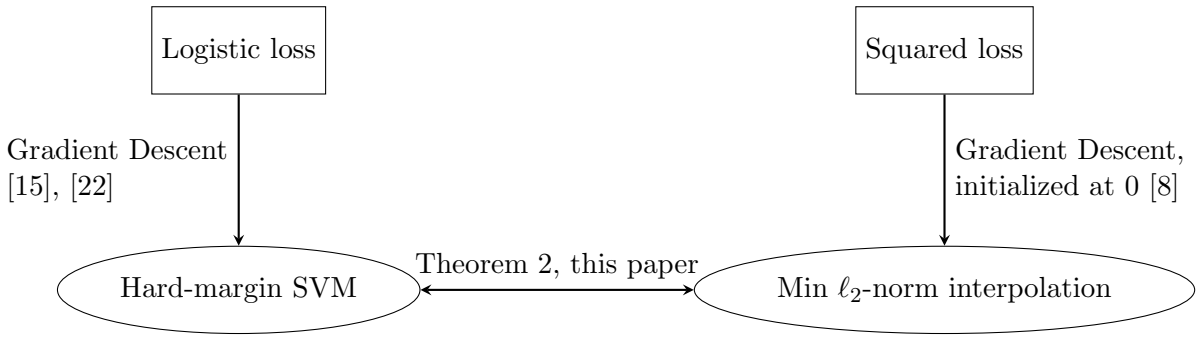


Figure 2.3: Equivalence of training procedures in overparameterized settings. Theorem 2 in this paper highlights exact equivalence between the hard-margin SVM and minimum-ℓ_2-norm interpolation under sufficient *effective overparameterization* for the special case of ridge features.

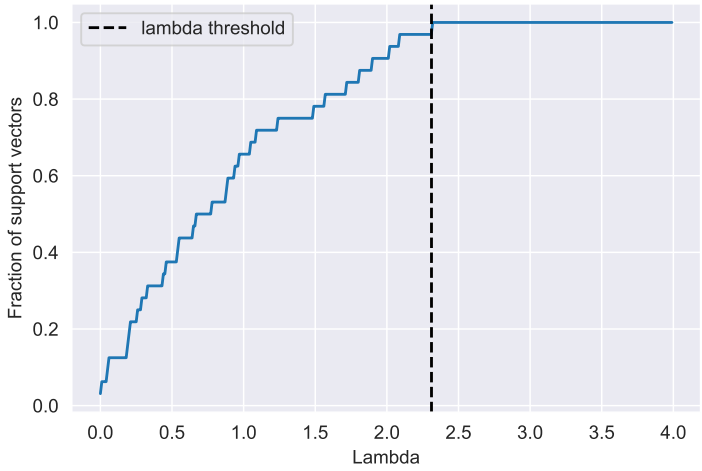


Figure 2.4: As λ is increased, the strength of the spurious features increases, making these more attractive to the optimizer. This causes the fraction of support vectors to increase until all training points are support vectors.

2.3 Characterization for ridge features

Theorem 2 (Condition for all support vectors)**.** *For the ridge featurization from Definition 2.2.1 with no weighting and $\alpha^* = e_1$, we have that $|S| = n$ if and only if:*

$$\lambda > \sqrt{\sum_{i=1}^{n} |x_i|(\|X_{\mathsf{train}}\|_\infty - x_i)} = \lambda_T \tag{2.1}$$

Before we understand what the theorem intuitively states, consider the constraint for the i_{th} training point in the SVM problem: $y_i(\alpha_1 x_i) + y_i(\lambda\alpha_{i+1}) \geq 1$. Since the optimizer is trying to keep the values in α small, as the value of λ increases for fixed x_i's, α_{i+1} will increase and α_1 will decrease. In other words, the larger λ is, the cheaper the spurious features are for the optimizer.

Theorem 2 informally asserts that as energy is redirected onto spurious features from true features (i.e as contamination increases), the fraction of support vectors increases. Figure 2.4 displays corroborates the theorem empirically; we see that are theorem is strong and is able to exactly predict the threshold of λ before all training points become support vectors.

Figure 2.5 and Table 2.3 illustrate the effects of λ on $\widehat{\alpha}$ and the fraction of support vectors by means of a simple example. The choice of $\lambda = \lambda_T$ corresponds to the orange line in Figure 2.5 and the second row in Figure 2.3. As λ increases, the weight on the true feature, $\widehat{\alpha}_1$ decreases and the weights on the spurious features increase. When $\lambda > \lambda_T$, all training points become support vectors.

Further we recognize that the simple ridge featurization preserves the order of Theorem 1 in Muthukumar et al. which states that $d > O(nlog(n))$ for all support vectors. Here, we expect $\|X_{\mathsf{train}}\|_\infty$ to scale as $log(n)$ and x_i to scale as n; hence our threshold on λ here is $O(\sqrt{nlog(n)})$, remarkably similar to the result on Gaussian features.

This intuition is explanatory for the other featurizations in Figure 2.1 and Figure 2.2 as well. For both Fourier and Gaussian features, as d increases, the number of spurious features are increasing,

λ	Num support	$\widehat{\alpha}_1$	$\widehat{\alpha}_2$	$\widehat{\alpha}_3$	$\widehat{\alpha}_4$	$\widehat{\alpha}_5$
0.01	2.0	4.00	0.00	-0.08	0.08	0.00
0.50	4.0	1.33	0.00	-1.33	1.33	0.00
2.00	4.0	0.38	-0.36	-0.45	0.45	0.36

Table 2.1: For the purposes of illustration of Theorem 2, data is chosen to be regularly spaced: $X_{\text{train}} = [-0.75, -0.25, 0.25, 0.75]^T$. In this case, we have $\lambda_T = 0.5$, which corresponds to the second row here. As λ increases, we can see that $\widehat{\alpha}_1$ decreases and the weights on the spurious features increase.

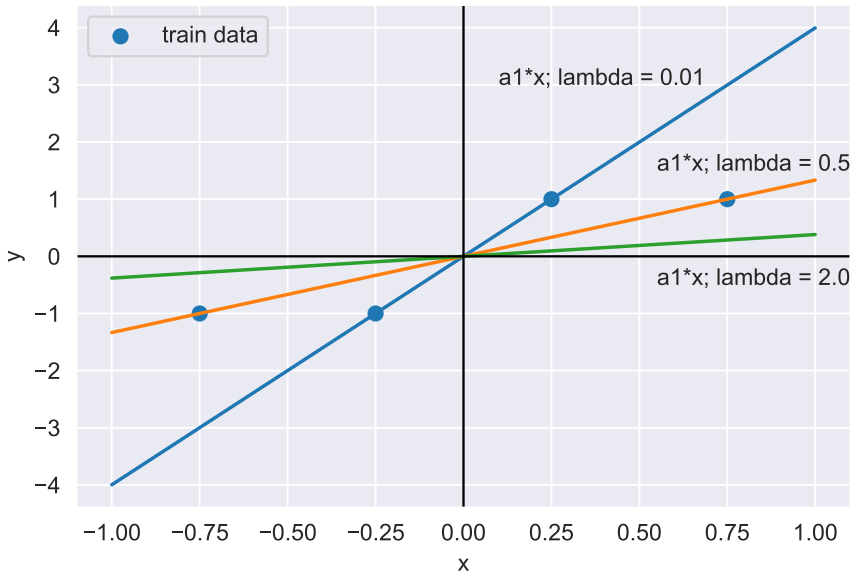


Figure 2.5: The experimental setup is the same as Table 2.3. As λ is increased, the weight on the true feature decreases, as evidenced by the decreasing slopes of the lines $\widehat{\alpha}_1 x$ above. When $\widehat{\alpha}_1 < \frac{1}{0.75}$ (see Lemma 2.4), then all training points become support vectors; this happens at the orange line.

and this pulls energy from the true featurization. For a fixed d, reducing the weighting on early features makes the spurious features cheaper; this can be thought of as analogous to increasing λ.

2.3.1 Intuition for the result

Since true features all correlate with the labels, they also all correlate with each other. Hence, if we use only true features, we are likely to have high confidences on some points. However, weak/useless features are not likely to agree with each other: even on strong points, some of these features will have low confidences. Thus, if we place more weight on such features, we are less likely to have a large distribution of confidences over the training points. Loosely restated: If we are making a decision and only have experts to advise us, we will be more confident (because all the experts agree) than if we had many amateurs on our advisory board (amateurs don't agree and tend to confuse!).

Let us understand the dependence on the different terms in the theorem. Again looking at the i_{th} constraint we see that, the larger that x_i is, the less likely the i_{th} point is to be a support vector. Hence, of all the training points, the $\arg\max_i x_i$ point will be most unwilling to become a support vector; this explains the dependence on the $\max_i |x_i|$ term. However, it is not only the point with maximum x_i that matters. To understand this, consider two choices of X_{train} with the same maximum x_i: $[1, 2, 4]^T$ and $[1, 2, 3, 4]^T$. We realize that the placing weight on the true feature helps with increasing the margin of each training point; however, placing weight on any single spurious feature only changes the margin of a single training point. Hence, in the two examples considered, the true feature is more attractive in the second; thus, a higher λ_T would be required to entice the optimizer towards the spurious features instead.

We can think of each of the spurious features as analogous to the "slack variables" in the soft-margin SVM problem; while the actual features are applied to each of the points, each slack variable is only applied to one point each. As the penalty on the slacks becomes smaller, they become more attractive to the optimizer and less weight is placed on the other features. Further, if a slack variable is used, the corresponding training point must be a support vector; this is analogous to the usage of a spurious features increasing $|M|$. The takeaway here is that there exist features that purport to be true features, but actually behave more like slack variables because they do not actually carry much information about the labels.

2.4 Proof of Theorem 2

Here is an outline of the proof:

1. We can write the optimal weights on each of the spurious features, $\{\widehat{\alpha}_i, i \neq 1\}$ as a function of the weight on the true feature, $\widehat{\alpha}_1$.

2. This allows us to reduce our support vector machine optimization problem into a 1-d unconstrained convex problem in α_1, that is solved in closed form.

3. We obtain a iff condition on $\widehat{\alpha}_1$ for all the training points to be support vectors. This is done in Lemma 2.4. See Figure 2.5 and Table 2.3 for an illustration of this lemma.

4. Since we have an explicit expression for $\widehat{\alpha}_1$ from above, this yields an equivalent condition in λ, which is our theorem.

First we introduce some notation that will help us with the proof. Recall that the i_{th} training point is said to be a support vector iff the above constraint holds with equality, and that the optimal

solution is denoted by $\widehat{\alpha}$. Realize that for ridge features parameterized by λ, the constraint for the i_{th} point can be rewritten as:

$$y_i(\alpha_1 x_i + \alpha_{i+1}\lambda) \geq 1 \tag{2.2}$$

Let M be the set of training points that are support vectors. Let $S_1 = \sum_{x_i \in M} |x_i|$ and $S_2 = \sum_{x_i \in M} x_i^2$. Let $r_i = 1 - \alpha_1 |x_i|$.

2.4.1 Steps 1 and 2

Since all our decision variables other than α_1 appear in only a single constraint, we can rewrite our optimization problem as:

$$\min_{\alpha_1 \ldots \alpha_{n+1}} \alpha_1^2 + \sum_{i=1}^{n} p_i(\alpha_1) \tag{2.3}$$

Here, $p_i : \mathbb{R} \to \mathbb{R}$ is defined as:

$$\begin{aligned} p_i(\alpha_1) = &\min \alpha_{i+1}^2 \\ &s.t\ y_i(\alpha_1 x_i + \alpha_{i+1}\lambda) \geq 1 \end{aligned} \tag{2.4}$$

Lemma 2.1 (Solution to inner optimization)**.**

$$\forall i, \widehat{\alpha}_{i+1}^2 = \max\left(0, \frac{1 - \widehat{\alpha}_1 |x_i|}{\lambda}\right)^2$$

Proof. The constraint in (2.4) above can be rewritten as

$$\mathsf{sgn}(x_i)\alpha_{i+1}\lambda \geq \frac{r_i}{\lambda}$$

Consider the two cases:

- $r_i \leq 0$
- $r_i \geq 0$

Since we are trying to minimize α_{i+1}^2 in the objective, in the first case we would just set $\alpha_{i+1} = 0$. Now consider the second case. If $y_i = 1$, our constraint is $\alpha_{i+1} \geq \frac{r_i}{\lambda}$; to minimize $|\alpha_{i+1}|$, we would set $\alpha_{i+1} = \frac{r_i}{\lambda}$ exactly. Similarly if $y_i = -1$, we would set $\alpha_{i+1} = -\frac{r_i}{\lambda}$. Upon squaring, we obtain our desired expression. □

Lemma 2.2 (Solution to outer optimization)**.**

$$\widehat{\alpha}_1 = \frac{\frac{S_1}{\lambda^2}}{1 + \frac{S_2}{\lambda^2}}$$

Proof. We can plug in our solution to (2.4) from Lemma 2.1 to simplify (2.3) to a 1-d convex optimization in only α_1:

$$\min_{\alpha_1 \ldots \alpha_{n+1}} \alpha_1^2 + \sum_{i=1}^{n} \max\left(0, \frac{1 - \widehat{\alpha}_1 |x_i|}{\lambda}\right)^2$$

Setting the gradient of the objective with respect to α_1 to 0, we obtain the desired. □

2.4.2 Step 3

Lemma 2.3. *i_{th} training point is a support vector* $\iff \widehat{\alpha}_1 \leq |\frac{1}{x_i}|$

Proof. Start with the forward direction.

$$\begin{aligned} & y_i(\widehat{\alpha}_1 x_i + \widehat{\alpha}_{i+1}\lambda) = 1 \\ \implies & y_i \widehat{\alpha}_1 x_i \leq 1 \\ \implies & |\widehat{\alpha}_1 x_i| \leq 1 \\ \implies & |\frac{1}{x_i}| \leq 1 \end{aligned}$$

The third implication follows because we know that $\alpha_1 > 0$ from Lemma 2.2 and that $y_i = \mathsf{sgn}(x_i)$. Now, moving to the backward direction.

$$|\frac{1}{x_i}| \leq 1 \implies |\widehat{\alpha}_1 x_i| \leq 1$$

By plugging our expression for α_{i+1} from Lemma 2.1, our consequent follows. □

Lemma 2.4. $\widehat{\alpha}_1 \leq \frac{1}{\|X_{\mathsf{train}}\|_\infty} \iff$ *all training points are support vectors*

Proof. Consider the forward direction. From the antecedant, we have that $\widehat{\alpha}_1 < |\frac{1}{x_i}|$. Invoking Lemma 2.3, our result follows.

Now, we move to the backward direction. Let $i^{\mathsf{max}} = \arg\max_i x_i$. As a special case of the antecedent, we have that the i^{max}_{th} point is a support vector. By apppealing to Lemma 2.3, our result follows. □

2.4.3 Step 4

First, we prove the forward direction. If all training points are support vectors, our expression for α_1 from Lemma 2.2 simplifies to:

$$\alpha_1 = \frac{\frac{\sum_{i=1}^n |x_i|}{\lambda}}{1 + \frac{\sum_{i=1}^n x_i^2}{\lambda^2}}$$

Setting $\alpha_1 < \frac{1}{\|X_{\mathsf{train}}\|_\infty}$ and solving for λ, we obtain the desired condition in Theorem 2.

Now, we prove the backward direction. We assume the condition on λ from Theorem 2. For the sake of contradiction, assume that all training points are not support vectors i.e $M \neq S_{train}$. Then, from Lemma 2.4,

$$\alpha_1 > \frac{1}{\|X_{\mathsf{train}}\|}. \tag{2.5}$$

Using the expression for α_1 from Lemma 2.2, (2.5) is shown to be equivalent to

$$\lambda^2 < \sum_{x_i \in M} \|X_{\mathsf{train}}\|_\infty (|x_i| - x_i^2)$$

This is contradictory to our assumed condition on λ from Theorem 2; hence, $M = S_{train}$ must be true. This completes our proof.

Chapter 3

Do large margins indicate good generalization?

An explanation that is often provided for generalization in classification problems is that classifiers with larger *margin* on the training data generalize better than those with smaller margin. In this Chapter, I first recap the intuition behind this idea and the problems for which it has been successful. Then, I take a closer look at margins in overparameterized settings; I examine where the explanation might fall short, and why.

3.1 Definitions and geometry

Consider a classifier f (does not have to be linear) that makes its prediction by first learning a real-valued function $g \in \mathcal{G}$ and then discretizing the outputs of g. In this section, we restrict our attention to the binary case: $f(x) = \mathsf{sgn}(g(x))$. Let us call g the *score function* associated with f.

Definition 3.1.1 (Pointwise margin)**.** *The margin of a classifier f of the above form at a point x with label y is: $\gamma_{\mathsf{pw}}(x) = yg(x)$.*

The intuition is that the large the magnitude of the score $|g(x_i)|$ is on a correctly classified point is, the more confident the prediction on that point is. Most often "margin" is spoken about in the context of a classifier $\gamma = min_{x_i}\gamma_{\mathsf{pw}}(x_i)$, and γ is often referred to as the "margin of f". It is important to be cognizant of this distinction between the min-margin γ and the distribution of pointwise margins $\gamma_{\mathsf{pw}}(x_i)$.

A problem that comes up here is that the margin of f could be arbitrarily increased by scaling the function by a positive scalar a: $f_2 = \mathsf{sgn}(ag)$. However, the binary outputs of f and f_2 are exactly the same for any given input; so, we would not expect any reasonable indicator of generalization to be vastly different for these two functions. To resolve this issue, instead of looking at γ, we first normalize the margin by the lipschitz constant of f. Informally, the Lipschitz constant is a measure of how sensitive the outputs of f are to changes in its inputs. Often, deducing the right normalization is the key to obtaining a margin expression that explains generalization performance - we will explore this further in Section 3.4.

Margins are most commonly spoken about in the context of hard-margin SVM's. For a linear classifier $f(x) = \alpha^T x$, the Lipschitz constant is $\|\alpha\|$. In this case, the pointwise normalized margin $\gamma(x_i)$ is exactly equal to the Euclidean distance of x_i from the decision boundary hyperplane $\mathcal{H} = \{x : \alpha^T x = 0\}$.

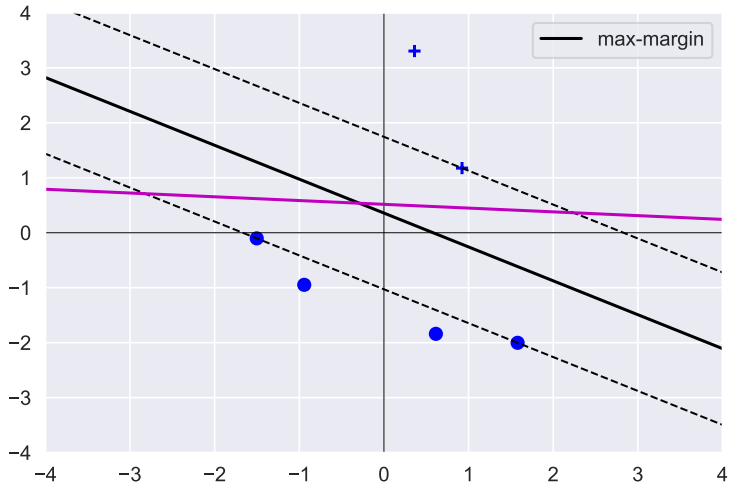


Figure 3.1: The data is i.i.d $\mathcal{N}(0, 2)$ and $\alpha^* = [2, 3]^T$. Both the black and the magenta decision boundaries perfectly seperate the training data; However, the black decision boundary has the largest margin, denoted by the dotted lines.

	$x_i[0]$	$x_i[1]$	y_i	γ_{pw}
0	0.36	3.31	1	2.29
1	0.92	1.18	1	1.00
2	0.61	-1.84	-1	1.31
3	-1.50	-0.10	-1	1.00
4	1.58	-2.00	-1	1.00
5	-0.94	-0.95	-1	1.36

Table 3.1: Data corresponding to Figure 3.1. Points indexed $1, 3, 4$ are the support vectors.

Consider Figure 3.1, which works with a small dataset reproduced in Table 3.1. While both the purple and black line separate the training data perfectly, the purple line has a small margin $= 0.72$ and is very close to training point number 3. Contrastingly, the black line has a margin of 1.18 which cannot be improved upon; the margin is shown by the dashed black lines. The γ_{pw} of the max-margin black line are shown in the last column of the table; we identify that points $\{1, 3, 4\}$ are the support vectors.

The notion of choosing a classifier, whose decision boundary is far from training points is intimately connected with the notion of adversarial robustness, explored further in Chapter 4: the amount of noise that needs to be added to a training point to flip its prediction is largest for the max-margin classifier.

3.2 The historical role of margins

Let us revisit the surprising observations regarding boosting brought up in Chapter 1. If a new theory could argue either that the relevant model complexity is not actually increasing or that the relevant measure of training error is not actually zero, then they could explain these observations.

Margin-based explanations were able to make both these arguments [21]. To do this, the surrogate ramp loss l_γ was utilized, parameterized by margin γ

Definition 3.2.1 (Ramp Loss)**.**

$$l_\gamma(z) := \begin{cases} 1 \text{ if } z \leq 0 \\ 1 - \frac{z}{\gamma} \text{ if } 0 < z \leq \gamma \\ 0 \text{ if } z > \gamma. \end{cases}$$

Generalization guarantees were developed which bounded the test error by a sum of two terms: a training loss term, under the ramp loss of g and a Rademacher complexity term of $\mathcal{G}$. In the boosting examples, it was found that Rademacher metric of $\mathcal{G}$ scaled more slowly with the number of primitive models than the VC-dimension of $\mathcal{F} = \mathsf{sign}(\mathcal{G})$, which is very sensitive to small real-valued changes in output that cause changes in sign. Furthermore, it was found that even though the classification training accuracy of f became zero, the ramp training loss of g reduced as the number of primitive models increased[1].

3.3 Linear case: Empirical exploration

To evaluate the predictive power of margins, I consider a sample margin bound below. Other margin bounds have the same terms and are very similar, but this is the strongest one I encountered for a linear classifier in overparameterized settings.

Theorem 3 (Theorem 21, [1])**.** *For a random test point (x, y) drawn from the same distribution as the training data, the following holds with probability $(1 - \delta)$ over the training data $\{x_i, y_i\}_{i=1}^n$:*

$$\Pr[\mathsf{sgn}(f_{\widehat{\alpha}}(x)) \neq y] \leq \frac{1}{n} \sum_{i=1}^{n} l_\gamma(f_{\widehat{\alpha}}(x) \cdot y_i) + \frac{4}{\gamma_N} \cdot \frac{\|\phi_{\mathsf{train}}\|_{\mathsf{F}}}{n} + \left(\frac{8}{\gamma} + 1\right) \cdot \sqrt{\frac{\ln(4/\delta)}{2n}} \tag{3.1}$$

l_γ is the ramp loss function in Definition 3.2.1.

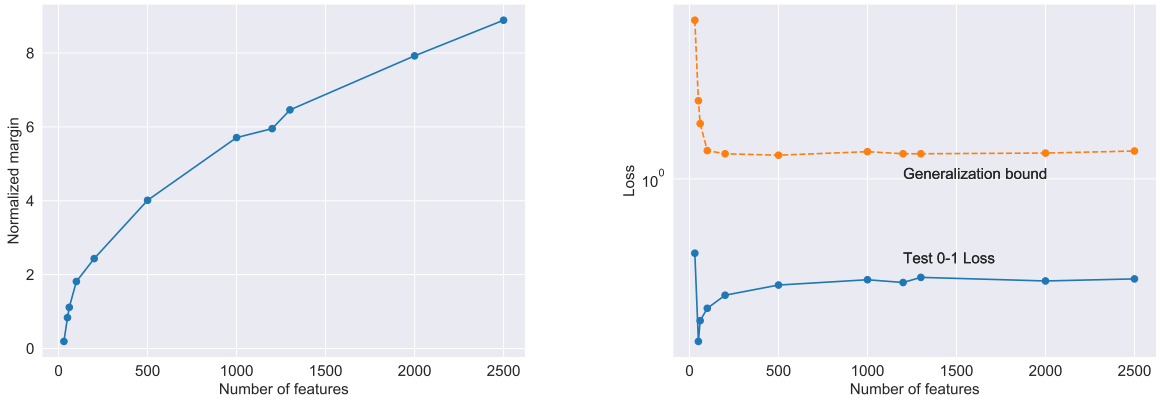
When the training data are separable, we apply the above bound setting the first (average training loss) term to 0, and only consider the the second term in the bound, i.e. we ignore the high-probability term. Equation (3.1) reminds us that there is a critical dependence on the intrinsic data dimension, captured by the term $\|\phi_{\mathsf{train}}\|_{\mathsf{F}}$. We will shortly see that this dependence is critical to track in the overparameterized regime.

3.3.1 Can margin explain the benefits of overparameterization?

We now investigate whether this generalization bound is effective in tracking the true test classification error for the hard-margin SVM in our setting for a number of choices of featurization. We plot the solution $\widehat{\alpha}_{\mathsf{binary}}$ in sufficiently overparameterized settings under which all training points become support vectors with high probability; therefore, the un-normalized margin $\gamma = 1$ and the normalized margin of the binary interpolating solution is exactly equal to $\gamma_N = \frac{1}{\|\widehat{\alpha}\|_2}$.

We study the evolution of margin, the ensuing upper bound in Equation (3.1), and the true test classification error as we increase the level of overparameterization for two choices of featurizations:

[1]This understanding of margins was inspired by Alexander Rakhlin's Generalization 3 lecture at the Simon's Institute in 2019.



(a) Normalized margin.

(b) Comparison of the generalization bound (3.1) with true test classification loss.

Figure 3.2: Evolution of normalized margin, ensuing generalization bound and true classification test loss as a function of number of features d for isotropic Gaussian features ($n = 32$ fixed). Observe that the terms $\|\phi_{\mathsf{train}}\|_{\mathsf{F}}$ and $\|\widehat{\alpha}\|_2$ cancel each other's effect on the bound, leading to a roughly constant bound. The true test error increases as d is increased.

isotropic Gaussian features (Definition 1.3.1) which generalize poorly according to Theorem 1 and weak features (Definition 1.3.2), which are known to exhibit the double-descent behavior. For the case of isotropic features, we retain our 1-sparse assumption from Section 1.4. For the case of weak features, we consider $y_i = \mathsf{sgn}(U_i)$ for $i \in \{1, \ldots, n\}$.

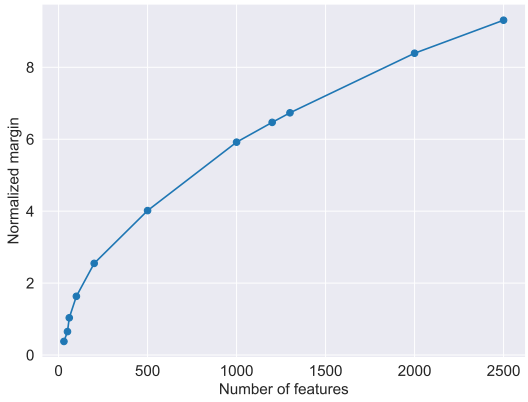
Figure 3.2 plots the isotropic case, and Figure 3.3 plots the weak features case. For both figures, we hold the number of training points, n constant and vary the number of features, d, to tune the extent of overparameterization. In both Figures 3.2(a) and 3.3(a), the normalized margin increases with increasing d, since the optimizer can use more features to meet the constraint in Equation (**??**). The generalization bounds in Figure 3.2(b) and Figure 3.3(b) are consequently very similar as well. However, while the test classification loss increases with d for isotropic features, it decreases with d for weak features.

Figures 3.2 and 3.3 together show that the relationship between margin and generalization is more complex than typically assumed in highly overparameterized regimes. We highlight a few observations:

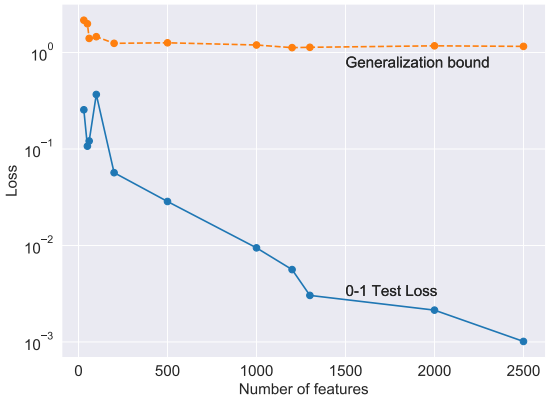
1. In both featurizations, the generalization bounds are always greater than 1, and hence, tautological. The $\|\phi_{\mathsf{train}}\|_{\mathsf{F}}$ term, which represents the scale of the data, effectively cancels out any beneficial effect of increasing normalized margin[2]. Intuitively, it is clear that *feature-space* margin-based bounds will have to scale with the intrinsic input dimension, which itself is overparameterized for Gaussian featurization.

2. Whether margin is *qualitatively* predictive of generalization is also unclear, as evidenced by the contrasting examples of weak features and isotropy. Under both featurizations, the normalized margin increases with increased overparameterization; but the actual test error behaves very differently (decreasing for weak features, but increasing for isotropy). In the

[2]In fact, for the case of minimum-ℓ_2-norm interpolation and isotropic features, this can be verified quantitatively, as we know that $||\widehat{\alpha}||_2 \sim \sqrt{\frac{n}{d}}$ and $||\phi_{\mathsf{train}}||_{\mathsf{F}} \sim \sqrt{nd}$ with high probability.



(a) Normalized margin.



(b) Comparison of the generalization bound (3.1) with true test classification loss.

Figure 3.3: Evolution of normalized margin, ensuing generalization bound and true classification test loss as a function of number of features d for weak features ($n = 32$ fixed, $\sigma = 0.1$). Observe that the terms $\|\phi_{\mathsf{train}}\|_{\mathsf{F}}$ and $\|\widehat{\alpha}\|_2$ cancel each other's effect on the bound, leading to a roughly constant bound. The true test error decreases as d is increased.

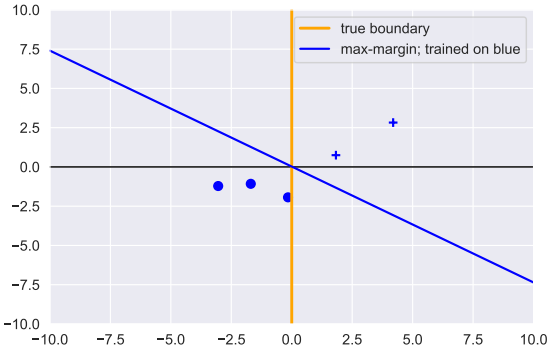
next subsection, I present an illustration of a simple case where higher margin does not mean better generalization.

3.3.2 Is max-margin always the right inductive bias?

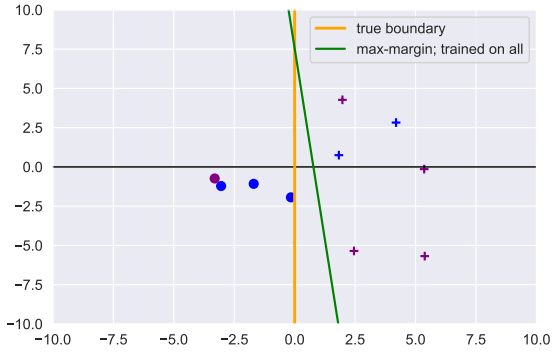
In Section 1.4, we find that generalization performance depends on survival - how much weight we place on our first feature. Does maximizing margin increase our survival? Figure 3.4 investigates through a simple 2-dimensional setup, similar to Figure 3.1 when maximizing margin may not be the best strategy. In Figure 3.4(a), we have 5 data points and the ones with class "+1" are denoted by "+" and those with class "-1" are denoted by circles. We again choose a 1-sparse $\alpha^* = [1, 0]^T$, so that only our first feature actually influences our labels; this yields the orange vertical axis as the true decision boundary of our data. However the margin-maximizing α_{SVM} learns the less generalizable blue boundary instead. What happens as if we obtain more datapoints? Figure 3.4(b) considers the same setup as Figure 3.4(a), but includes the new purple points into the training set in addition to the old blue ones. The new training data now diminishes the effect of the margin-maximizing inductive bias, and now our learned $\widehat{\alpha}$ is much more similar to α^*. However, for this to happen we needed n considerably greater than d, which would never happen in an overparameterized setup.

In Figure 3.5, we fix n and d and encourage the learner to prefer early features by increasing m in the polynomial weighting scheme; the generalization error reduces due to increasing survival but the margin decreases, confirming for our setup what Figure 3.4 suggests. This effect of having a wrong inductive bias can be understood as an asymmetrical looseness of bounds: the bound is tighter for the classifier with larger margin, which leads to smaller error estimates than other classifiers.

The **main takeaway** from these Figures is that if our problem is sparse and our training matrix is filled with "useless" features (see Section 1.2.2), then max-margin is likely to be the wrong prior because it place non-zero weight on these "useless" features.

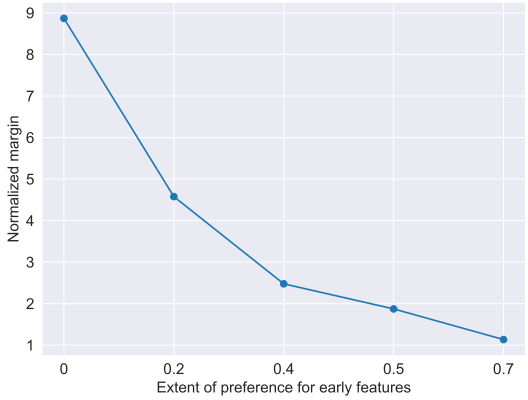


(a) The max-margin solution places a non-negligible weight on the second feature and chooses the blue decision boundary instead of the orange one, to the detriment of its generalizability.

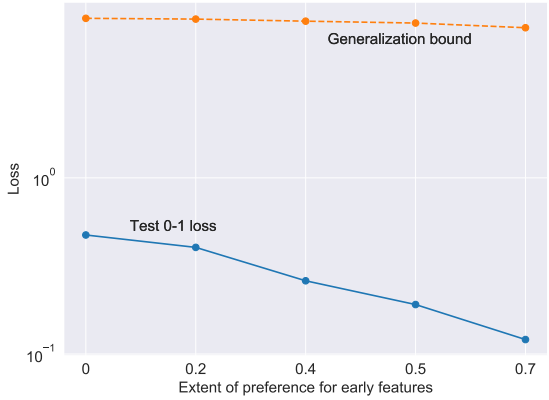


(b) Upon including new training points, the max-margin solution becomes more similar to the true decision boundary.

Figure 3.4: The data is drawn from $\mathcal{N}(0, 3)$ and $e_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ are the true weights. The '+' denote the points with class $+1$ and the circles denote the points with class -1.



(a) Normalized margin.



(b) Comparison of the generalization bound (3.1) with true test classification loss.

Figure 3.5: The number of training points, $n = 529$ and the number of features, $d = 12167$ is fixed. The strength of preference for lower features is varied, as determined by rate of weight decay $\lambda_k = \frac{1}{k^m}$ in Definition 1.3.6. As we increase m, the normalized margin decreases, but so does the actual test loss

3.4 Kernelized case: Proposed modifications

We find that the appropriate normalization for the case of linear classifiers is the ℓ_2 norm of the weights vector. In the kernelized case where there is a feature-map ϕ that is not the identity, it is less clear what the right notion of normalization is. While in the linear case, we obtained nice geometrical understanding of margin in the *native space*, $\mathcal{X}$, now there is also a distinct *feature space*: the image of our feature-map, $\text{Im}(\phi)$. Bartlett et al. face a similar task when finding the right notion of normalized margin for a feedforward neural network, and they choose an approximation of the Lipschitz-constant as their normalization [2]. Recalling that in the linear case, the normalized margin is the minimum perturbation of a training point required for misclassification, we can start here in the kernelized case.

Definition 3.4.1 (Native margin)**.** *Native margin is the minimum perturbation in the native space prior to featurization, $\mathcal{X}$ that induces a misclassification by the model. For $f_{\widehat{\alpha}}$, the native margin on a point x with correct label y is:*

$$\begin{aligned}\gamma_{\mathsf{native}}(x) = &\min_{\Delta} \Delta \\ &s.t\ \mathsf{sgn}(y) \neq \mathsf{sgn}(\widehat{\alpha}^T \phi(x+\Delta))\end{aligned}$$

Notice that our definition is point-wise and not a summary statistic for the classifier. As a special case, we realize that for already misclassified training points, $\gamma_{\mathsf{native}} = 0$. While γ_{native} is our quantity of interest, this optimization problem is difficult to solve because the ϕ_j's might be arbitrary non-convex functions in Δ. Let us try to take a linear approximation to ϕ to alleviate this trouble. Before we do that, first we lift our problem:

$$\begin{aligned}&\min_{\Delta, u} \Delta \\ &s.t\ \mathsf{sgn}(y) \neq \mathsf{sgn}(\widehat{\alpha}^T u) \\ &\phi(x+\Delta) = u\end{aligned}$$

Now, we can perform the first-order-approximation for each ϕ_i:

$$\phi_i(x+\Delta) - \phi_i(x) \approx \frac{d\phi_i(x)}{dx} \times \Delta.$$

Define $S = \text{diag}(\frac{d\phi(x)}{x})$. Using the element-wise approximation above, we can rewrite the second constraint of our problem to reformulate our problem as:

$$\begin{aligned}&\min_{\Delta, u} \Delta \\ &s.t\ \mathsf{sgn}(y) \neq \mathsf{sgn}(\widehat{\alpha}^T u) \\ &\phi(x) + \Delta S\mathbf{1} = u\end{aligned}$$

Using our second inequality, we would like to remove Δ from our optimization entirely. However, we realize that our second constraint (obtained through an approximation) contains d equations containing Δ:

$$\frac{\phi_i(x+\Delta) - \phi_i(x)}{\frac{d\phi_i(x)}{dx}} \approx \Delta$$

Remembering that these were each obtained by an approximation, instead of just relying on any one of these, we aggregate all of them:

$$\sum_{i=1}^{n}\left(\frac{\phi_i(x+\Delta)-\phi_i(x)}{\frac{d\phi_i(x)}{dx}}\right)^2 \approx d\Delta^2$$
$$\frac{1}{\sqrt{d}}\left\|S^{-1}(u-\phi(x))\right\|_2 \approx \Delta.$$

Replacing Δ in our objective and removing our second constraint:

$$\frac{1}{\sqrt{d}}\min_u \left\|S^{-1}(u-\phi(x))\right\|_2$$
$$s.t\ \mathsf{sgn}(y) \neq \mathsf{sgn}(\widehat{\alpha}^T u)$$

This can be equivalently framed as below.

Definition 3.4.2 (Gradient Weighted Margin)**.**

$$\gamma_{\mathsf{GW}}(\widehat{\alpha},x) = \frac{1}{\sqrt{d}}\min_u \left\|S^{-1}(u-\phi(x))\right\|_2$$
$$s.t\ \widehat{\alpha}^T u = 0$$

The quality of this approximation to the native margin depends on the effectiveness of the Taylor approximation. Recall the interpretation of the feature-space margin as the minimum perturbation to cause a misclassification; the above resembles this. However, the "perturbation" u is weighted in the objective by S^{-1} i.e the low gradient directions are penalized more heavily; this inspires the name of the new margin. Using the change of co-ordinates and the projection theorem, we can solve the optimization in closed form.

Theorem 4 (Gradient-weighted margin in closed form)**.**

$$\gamma_{\mathsf{GW}}(\widehat{\alpha},x) = \frac{1}{\sqrt{d}}\frac{\widehat{\alpha}^T x}{\|S\widehat{\alpha}\|_2} \tag{3.2}$$

Proof. Starting from the definition of γ_{GW}:

$$\gamma_{\mathsf{GW}}(\widehat{\alpha},x) = \frac{1}{\sqrt{d}}\min_u \left\|S^{-1}(u-\phi(x)\right\|_2 : \widehat{\alpha}^T u = 0$$

Performing the change of co-ordinates $\widehat{\beta} = S\widehat{\alpha},\ \widetilde{u} = S^{-1}u,\ \widetilde{\phi}(x) = S^{-1}\phi(x)$, we rewrite the above as

$$= \frac{1}{\sqrt{d}}\min_u \left\|\widetilde{u}-\widetilde{\phi}(x)\right\|_2 : \widehat{\beta}^T u = 0$$

Using the projection theorem, we can write this in closed form and then change our co-ordinates back to obtain the desired.

$$= \frac{1}{\sqrt{d}}\frac{\widehat{\beta}^T\widetilde{\phi}(x)}{\left\|\widehat{\beta}\right\|_2}$$
$$= \frac{1}{\sqrt{d}}\frac{w^T\phi(x)}{\left\|\widehat{\beta}\right\|_2}$$

□

3.4.1 Experimental inspection of proposed notions

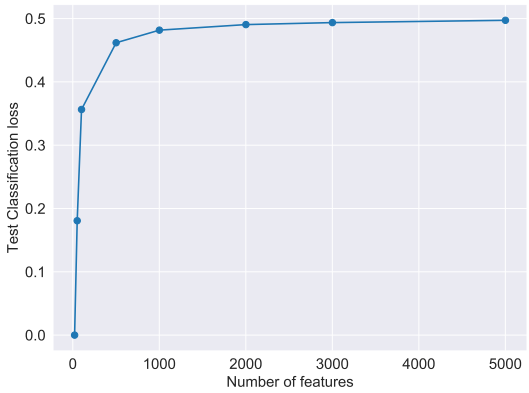
We test the ability of the proposed γ_{GW} and γ_{native} to explain generalization in the kernalized setup with uniformly sampled Fourier features. We recognize that increasing d is not beneficial to generalization because of the addition of more "useless" features that reduce our survival. We make the following observations about the usual normalized margin in Figure 3.6(b) and Figure 3.6(c). Since both the gradient weighted and native margins are evaluated for each training point individually, we plot the medians and include the 40_{th} and 60_{th} percentile as error bars.

1. The naive normalized margin is unable to capture the trend in generalization performance; while we would expect decreasing margins to indicate the worsening generalization, the margins increase.

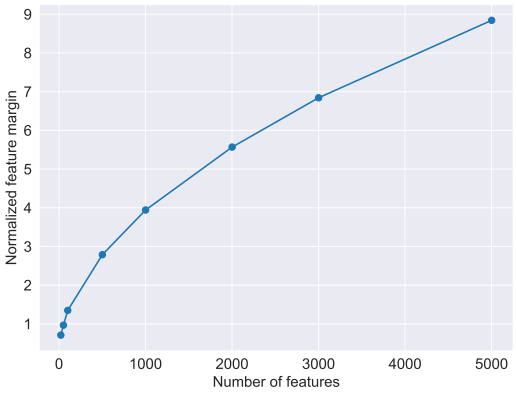
2. The native and gradient-weighted margins both decrease as we would desire.

3. The gradient weighted margin, even though designed to approximate the native margin, is consistently an underestimate. This suggests that the first order approximation does not hold as well as we might have hoped.

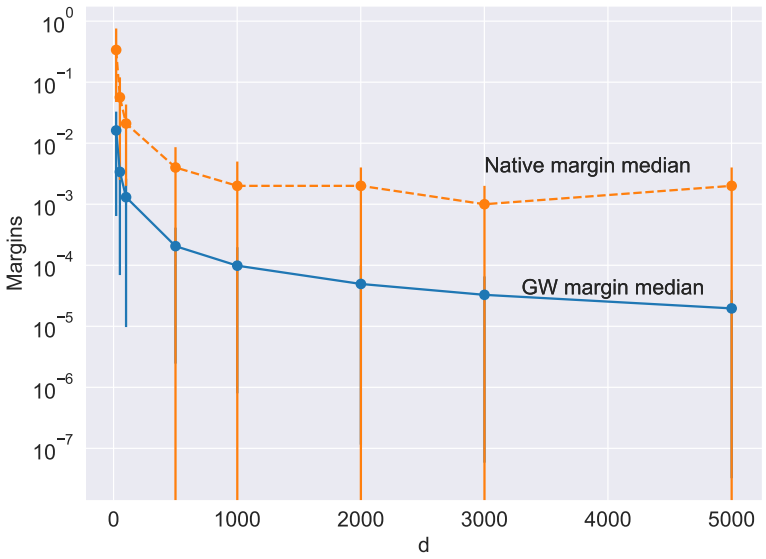
4. The error bars for the native margin are extremely large in the bottom direction, indicating a large distribution in native margins between training points.



(a) As d increases, the test classification loss becomes worse.



(b) The feature margin increases even though the classification loss gets worse.



(c) The native margin and gradient weighted margin both decrease as d is increased. The median over all training points is plotted and the top and bottom error bars are the 40_{th} and 60_{th} percentile respectively.

Figure 3.6: Examination of ability of original and new margins to predict generalization error in a kernelized setting. Here, we use uniformly sampled Fourier features (Definition 1.3.4). We fix $n = 32$, use no weighting and increase d.

Chapter 4

Adversarial examples

Almost a decade ago, Szegedy et al. discovered that small perturbations to input images, that are imperceptible to the human eye, can trick image-classification convolutional neural networks at test time into making misclassifications. The term *adversarial examples* was coined for these benign-looking adversarially perturbed images, and this term is now used more generally outside of image-classification tasks as well. The discovery of adversarial examples came as a surprise to many, and inspired investigation into the creation of adversarial examples, the design of classifiers that are resistant to such attacks, and other work that tries to interpret adversarial examples.

While it was commonly believed that adversarial examples were an artifact of neural networks, Figure 4.1 illustrates that it is possible to construct such attacks for otherwise performant kernel classifiers as well. Specifically, for $m \in [0.15, 0.25]$, it is observed that despite low test classification loss, test adversarial loss is very high. Hence, we see that these kernel models are a good playground in which to investigate adversarial examples; a big benefit of identifying adversarial examples for a problem with one-dimensional input is that the learned function and adversarial instances can be easily visualized, like in Figure 4.2. This chapter presents a preliminary result from ongoing work that aims to study robustness in relation to the findings from earlier chapters on generalization performance, margins and equivalence between different training formulations.

4.1 Notation

We specify a set of allowable perturbations for the adversary for point x as $\mathcal{X}(x)$. Now, we can define the adversarial version of the classification loss for test sample (x, y) as follows:

$$\mathcal{C}_{adv}(\widehat{\alpha}) := \mathbb{E}[\max_{\tilde{x} \in \mathcal{X}(x)} \mathbb{I}[Y \neq \mathsf{sgn}(\langle \phi(\tilde{x}), \widehat{\alpha} \rangle)]] \tag{4.1}$$

This is our quantity of interest for this section: the worst-case expected loss if x is allowed to be perturbed by small amounts. The extent of allowable perturbation is governed by the choice of $\mathcal{X}$.

In this section, I will be working with regularly spaced Fourier features (Definition 1.3.3). Hence, we choose $x \in \mathbb{R}$ and we define

$$\mathcal{X}(X) = \{\tilde{x} : |\tilde{x} - x| \leq \epsilon\}.$$

We choose $\epsilon = 0.05$. For higher dimensional input, the absolute value is usually replaced with some choice of norm, the l_∞ norm being the most common. Let the chosen norm be $\|\cdot\|_p$, then let $\|\cdot\|_*$ be the dual norm.

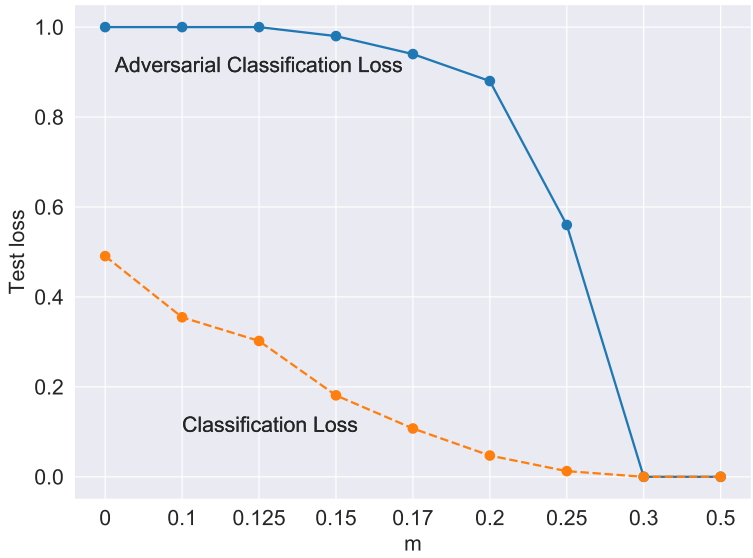


Figure 4.1: The regularly spaced fourier featurization, as in Definition 1.3.3 was is used here with fixed $n = 32$ and $d = 2000$. The feature weighting, as in Definition 1.3.6 is tuned and classification adversarial performance of the Binary Interpolation in Definition 1.3.9 is compared.

4.2 Related works

Here, I present related works on adversarial examples in the context of the findings from previous chapters. I focus on works that aim to understand what properties of the learned function influence susceptibility to adversarial examples; it does not review the literature on algorithms to create and defend against adversarial examples.

4.2.1 Relation to margin

The interpretation of margin introduced in Chapter 3 as the amount of perturbation needed to induce a misclassification on a training point suggests a relationship between margins and adversarial examples.

The max-margin SVM solution can be thought of as a robust optimization solution to the problem of finding a linear decision boundary that perfectly seperates the data. Adversarial training, a defense against adversarial examples that has proved to be successful, is an iterative method that alternatively finds adversarial examples for the model iterate, and then updates the model by training on these adversarial examples [17]; this is also derived from a robust optimization formulation of the problem. Charles et al. find that adversarial training on a binary linear classification problem actually does increase the margin of the network at a much faster rate than ordinary training [7]. Further, in the same work where Bartlett et al. provide a margin-based generalization bound for neural networks, they also conjecture that having large margins indicates adversarial robustness [2].

4.2.2 Relation to contamination

We conjecture that higher contamination and lower survival must increase susceptibility to adversarial examples. In support of this hypothesis, we provide two arguments by utilizing findings from related works.

Ford et al. relate robustness to adversarial noise and robustness to stochastic noise; they assert that unless a classifier is robust to stochastic noise, it cannot be robust to adversarial noise [10]. Their arguments proceeds that unless your performance in stochastic environments is very almost perfect, the probability of there existing a misclassified point in a ball around a test point is very high. In the present setup, adding random noise to x should not change test performance much. And we showed in Chapter 1 that higher contamination leads to poorer test performance; hence, it should lead to poorer adversarial performance as well. In Figure 4.1, we observe that unless the test classification loss is very close to 0, adversarial examples proliferate.

Ilyas et al. argue that the adversarial performance of a neural network is determined by the robustness of the featurizations it learns [14]. They define what it means for a feature to be robust, and if very high weight is placed on non-robust features, they find the classifier to be non-robust. In the case of the Fourier featurization considered in Figure 4.2, the higher-frequency features are more sensitive to perturbations in the inputs and hence more non-robust in the sense of Ilyas et al.. Contamination precisely measures how much weight we place on later features - hence, increasing contamination should signal poorer adversarial performance.

4.2.3 Transferrability, Geometry, and other theoretical results

A fascinating observation was made by Szegedy et al. in their seminal paper that adversarial examples that were designed for one classifier transfer to another classifier i.e they are misclassified for the second classifier as well [23]. Liu et al. investigated this finding for multiclass classification problems through an extensive set of experiments [16]. They corroborated these findings for *untargeted adversarial examples*, which are designed to simply be misclassified without a preference for which wrong class is chosen. However, they discover that *targeted adversarial examples*, that are designed to trick the classifier into making a specific prediction on them rarely receive the same prediction for a new classifier.

He et al. and Fawzi et al. investigate the geometrical properties of the decision boundaries of the learned classifier to understand how this influences adversarial robustness [13, 9]. He et al. attempted to estimate where the decision boundary lay by perturbing test examples until they are just misclassified - this must mean that they just crossed the decision boundary. Fawzi et al. investigate the shape of the decision boundary of neural networks - they find that in general, it is quite flat but around adversarial examples, it is curved.

There are also some theoretical results for stylized setups that make hypotheses about what may influence adversarial susceptibility. Yin et al. evaluate the Rademacher complexity of the adversarial loss class, and find that it is considerably larger than the non-adversarial one; by doing this, they imply that learning an adversarially robust classifier must require a much higher complexity than a non-robust classifier [26]. This hypothesis is supported by Madry et al., who provide an intuitive toy example to make a similar argument [17]. Bubeck et al. believe that learning an adversarially robust classifier is computationally difficult, and in some cases intractable; according to them, this explains the empirical difficulty of learning adversarially robust classifiers [6]. Belkin et al. find that wrong labels in the training dataset causes regions of misclassification around these mislabeled training examples, and assert that these regions are where adversarial examples thrive in [3]. Figure 4.2(a) shows that even in the absence of label noise, adversarial examples can proliferate; we explore this

further in the next section.

4.3 Adversarial examples can proliferate near training points

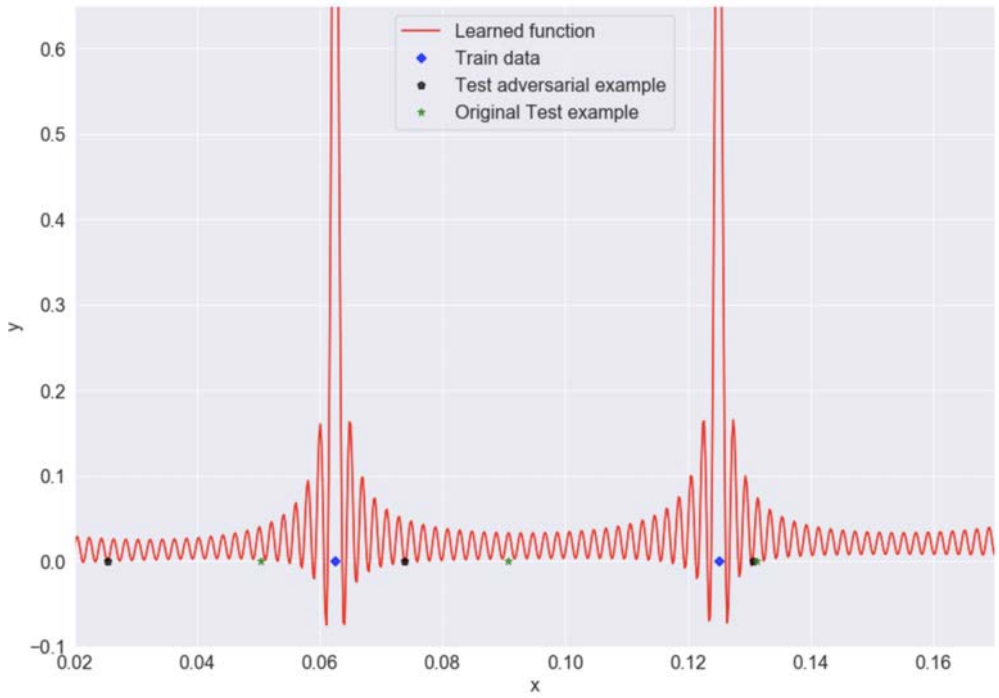
In Figure 4.2, we plot our learned predictor $f_{\widehat{\alpha}}(x)$ on the independent axis and x on the dependent axis. The blue diamonds denote the location of the train points and the red curves represent the predictions. We choose ϕ to be the Fourier featurization. We choose $m = 0.2$ as the polynomial decay weighting parameter because we want adversarial examples to be present.

We wish to examine where $\mathsf{sgn}(f_{\widehat{\alpha}}(x))$ does not match with $\mathsf{sgn}(x)$. These are regions of misclassification. The objective of the adversary is to perturb x into one of these regions. We can clearly observe the existence of these misclassification regions in the vicinity of the training points. This is quite a general phenomenon, and is observed for other learning algorithms (SVM), other feature families (Legendre) and for real labels.

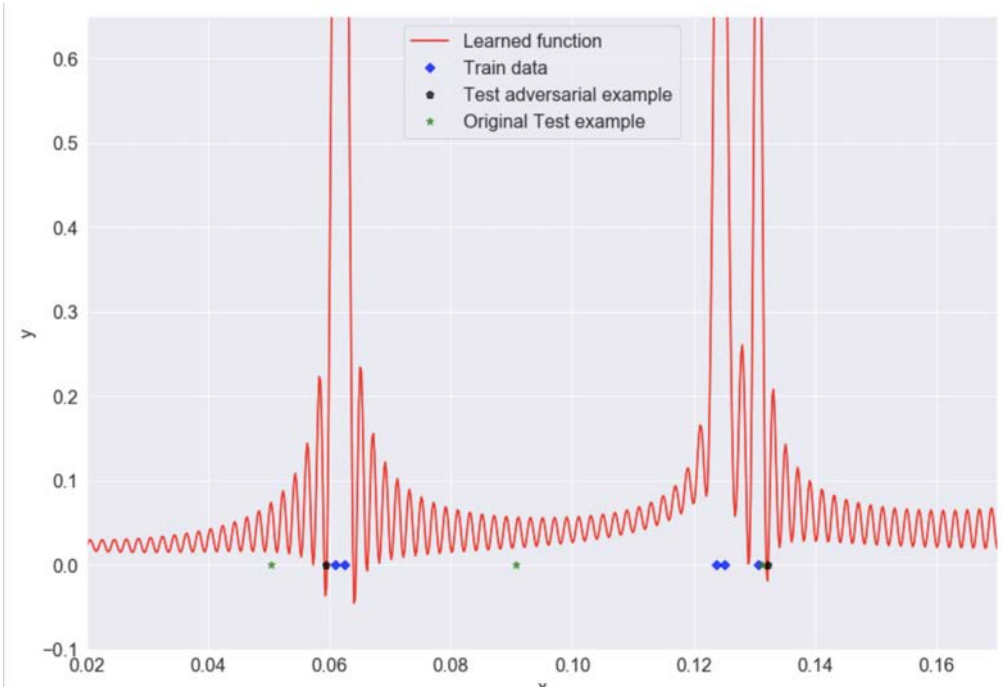
Since $f_{\widehat{\alpha}}$ is very close to the zero-function generally, around training points, we have almost a jump discontinuity. We see that these spikes to fit the training points are accompanied by troughs, resembling the shape of the sinc function. This behavior has been replicated for other featurizations as well and we conjecture that it captures a general phenomenon.

These troughs allow for the function to cross the horizontal axis in the neighbourhood of training points, which leads to a change of sign and misclassification regions. This permits adversarial examples to proliferate. The adversarial examples are represented by the black pentagons. Hence, the closer the function is to the horizontal axis and the larger the troughs are, the more likely it is to find adversarial examples.

The function plots in Figure 4.2 indicate that if we are able to suppress the oscillations around the training points that allow for zero-crossings, then we might be able to defend against adversarial examples. In Figure 4.2(b), we see that including the adversarial examples into the training set has this desired effect.



(a) Around training points, there is a spike and a trough, which spawns adversarial examples nearby; this is reminiscent of Gibb's phenomenon.



(b) Upon inclusion of the adversarial points (black pentagons) from (a) into the training set,the Gibbs-like behavior from (a) can be controlled to reduce the proliferation of adversarial examples.

Figure 4.2: Visualization of adversarial examples for regularly spaced Fourier features (Definition 1.3.3). d is fixed at 2000 and $m = 0.2$ in the polynomial decay weighting scheme from Definition 1.3.6. We fix $= 0.05$.

Bibliography

[1] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[2] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[3] Mikhail Belkin, Daniel Hsu, and Partha Mitra. Overfitting or perfect fitting? risk bounds for classification and regression rules that interpolate, 2018.

[4] Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019.

[5] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207. Springer, 2003.

[6] Sébastien Bubeck, Eric Price, and Ilya Razenshteyn. Adversarial examples from computational constraints, 2018.

[7] Zachary Charles, Shashank Rajput, Stephen Wright, and Dimitris Papailiopoulos. Convergence and margin of adversarial training on separable data, 2019.

[8] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[9] A. Fawzi, S. Moosavi-Dezfooli, P. Frossard, and S. Soatto. Empirical study of the topology and geometry of deep networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3762–3770, 2018.

[10] Nic Ford, Justin Gilmer, Nicholas Carlini, and Ekin Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise. *CoRR*, abs/1901.10513, 2019. URL `http://arxiv.org/abs/1901.10513`.

[11] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841, 2018.

[12] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[13] Warren He, Bo Li, and Dawn Song. Decision boundary analysis of adversarial examples. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=BkpiPMbA-`.

[14] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019.

[15] Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Learning Theory*, pages 1772–1798, 2019.

[16] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks, 2016.

[17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[18] Vidya Muthukumar, Adhyyan Narang, Vignesh Subramanian, Mikhail Belkin, Daniel Hsu, and Anant Sahai. Classification vs regression in overparameterized regimes: Does the loss function matter?, 2020.

[19] Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai. Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, 2020.

[20] Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry. Convergence of gradient descent on separable data. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3420–3428, 2019.

[21] Robert E Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5): 1651–1686, 1998.

[22] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[23] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[24] Vladimir Naumovich Vapnik. An overview of statistical learning theory. *IEEE transactions on Neural Networks*, 10(5):988–999, 1999.

[25] Blake Woodworth, Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Kernel and deep regimes in overparametrized models. *arXiv preprint arXiv:1906.05827*, 2019.

[26] Dong Yin, Kannan Ramchandran, and Peter L. Bartlett. Rademacher complexity for adversarially robust generalization. *CoRR*, abs/1810.11914, 2018. URL `http://arxiv.org/abs/1810.11914`.