

CPSC 532D — STATISTICAL LEARNING THEORY

Danica J. Sutherland

University of British Columbia, Vancouver

Fall 2024

For syllabus-type material, see the course website.

This file contains all the notes for the course, and will be updated frequently as we add more sections. Each section is also available as its own standalone file from the course website; these will generally only be updated after class to correct mistakes (and I'll tell you either in class or on Piazza if any of these are important).

These notes are heavily indebted to a variety of sources, including [SSBD14; MRT18; Bach24; Wai19; Tel21; Ma22] as well as papers cited throughout.

Contents

1	Setup; ERM	5
1.1	Linear regression	5
1.2	General problem setup	7
1.3	Empirical Risk Minimization	8
1.4	Error decompositions	9
1.4.1	ERM estimation error	10
2	ERM with finite hypothesis classes	11
2.1	Estimation error: asymptotics	11
2.2	Uniform convergence, bounded loss	12
2.3	Finite \mathcal{H}	13
2.3.1	Is this finiteness assumption reasonable?	14
3	Concentration inequalities	15
3.1	Markov	15
3.2	Chernoff bounds	16
3.3	Subgaussian variables	16
3.3.1	Proof of Hoeffding's lemma	18
4	PAC learning; infinite \mathcal{H}	19
4.1	PAC learning	19
4.2	Covering number bounds	20
4.2.1	Smoothness: Lipschitz functions	21
4.2.2	Putting it together with a set covering	23
4.2.3	Aside: Bounds on covering numbers	26
5	Rademacher complexity	29
5.1	A g-g-g-g-ghost (sample)	29
5.2	Properties of Rademacher complexity	31
5.2.1	Talagrand's contraction lemma	32
5.2.2	Complexity of bounded linear functions	33

For more, visit <https://cs.ubc.ca/~dsuth/532D/24w1/>.

5.3	Concentration	35
6	Growth functions and VC dimension	39
6.1	Zero-one loss	39
6.2	Finite sets	40
6.3	Growth functions	41
6.4	VC dimension	41
6.4.1	Examples of computing VC dimension	42
6.4.2	Growth function bounds in terms of VC: Sauer-Shelah	44
7	Online learning	47
7.1	Online Binary Classification in Realizable Setting	47
7.2	Decision-Theoretic Online Learning and Exponential Weights (Hedge)	52
7.3	Bandits	56
7.3.1	Adversarial bandits.	56
7.3.2	Stochastic bandits	58
8	Lower bounds; no free lunch	65
8.1	No free lunch for high-VC classes	65
8.1.1	Interpretation	67
8.1.2	Aside: “learning is NP-hard”	68
8.2	Lower bounds	69
8.3	The “Fundamental Theorem of Statistical Learning”	70
9	Nonuniform Learning	73
9.1	Structural Risk Minimization	73
9.1.1	With Rademacher bounds	75
9.1.2	Problems with bound minimization	75
9.1.3	Aside: Avoiding the δ dependence	76
9.1.4	Relationship to regularization	77
9.2	Nonuniform learnability	77
9.3	Minimum Description Length	78
9.3.1	Singleton Classes	78
9.3.2	Minimum Description Length	78
10	Universal Approximation	81
10.1	Denseness	81
10.2	Universal Approximators	82
10.3	Universal approximation of neural networks	83
10.3.1	Constructive proofs	83
10.3.2	Non-constructive bound via Stone-Weierstrass	85
10.4	Circuit complexity	86
10.5	Interpretation	86
11	Kernels	89
11.1	Defining function spaces	89
11.2	Polynomial functions	91
11.3	Reproducing kernels	92
11.3.1	Special case: linear kernel	93
11.4	Optimizing in the RKHS	94
11.4.1	Example: kernel ridge regression	94
11.5	Other kernels	95
11.5.1	Some properties	97

12 Is ERM enough?	99
13 Optimization	101
13.1 Gradient descent	101
13.2 β -smooth functions	102
13.3 Aside: convex functions	104
13.3.1 Aside: SGD non-convex convergence	105
13.4 Are deep networks β -smooth?	105
13.5 Is a stationary point enough?	106
14 Neural Tangent Kernels	109
14.1 Simple case	109
14.1.1 Approximation quality	110
14.1.2 The neural tangent kernel	111
14.1.3 More general networks	111
14.2 Optimization	112
14.3 Does this mean deep learning is solved?	114
15 Implicit Regularization and Margins	115
15.1 Gradient descent for linear regression	115
15.2 Separable logistic regression	117
15.2.1 Margin maximization	118
15.2.2 Hinge loss interpolation	119
15.2.3 Margin analysis	120
15.3 Other models/algorithms	121
Bibliography	123

1 Setup; ERM

This course is about: When should we expect machine learning algorithms to work? What kind of problems are possible for machine learning models to represent? What is possible to learn from data?

There are many complementary ways to study these questions. This course takes a primarily theoretical, mathematical approach, but tries to be guided by experimental results.

To phrase these questions more precisely, let's start by thinking about one of the simplest and best-understood machine learning models, linear regression. We'll use this as an example to set up our more general problem and think about how we can address those questions.

Is linear regression “really machine learning”? Obviously Legendre and Gauss didn't use that term in the early 1800s, but one reasonable definition of machine learning is “something you can publish a NeurIPS paper about,” and as someone with multiple NeurIPS papers about linear regression, that makes the answer yes.

1.1 LINEAR REGRESSION

In the typical linear regression setting, we have m training inputs $x_i \in \mathbb{R}^d$ and corresponding outputs $y_i \in \mathbb{R}$. (For one of many possible examples, x_i might be a collection of summary features for a large geographic area, and y_i the number of hectares that burnt in forest fires in a given year.) We'll denote this as

$$S = ((x_1, y_1), \dots, (x_m, y_m)) \subset (\mathbb{R}^d \times \mathbb{R})^m. \quad (1.1)$$

While I used the term “training set” (because it's extremely well-established terminology), we actually want to potentially allow repeated data points. Occasionally, we might also care about the order (e.g. in online learning), so mathematically, we're going to use S as an m -tuple, not a set.

The usual assumption – which is definitely *not* always true, but is overwhelmingly the usual assumption in analyzing these kinds of things – is that these (x, y) pairs are independent samples from a distribution \mathcal{D} . We also write this $S \sim \mathcal{D}^m$, meaning that each of the m entries of S is an independent sample from \mathcal{D} .

\mathcal{D}^m is called a product distribution, and is a distribution over $(\mathcal{R}^d \times \mathcal{R})^m$ of m -length iid sequences.

Our goal is to use S to find a weight vector w such that $w \cdot x \approx y$, for *fresh* (test) samples $(x, y) \sim \mathcal{D}$. Another way to phrase this is that we're looking for a *linear predictor*, i.e. a function $h_w : \mathbb{R}^d \rightarrow \mathbb{R}$ of the form $h_w(x) = w \cdot x$, such that $h_w(x) \approx y$.

You might also want an offset, $w \cdot x + b$, but we'll usually ignore that, since you can just add a constant 1 feature to x .

Statisticians, econometricians, etc. are often most concerned with getting the “right” w vector. For instance, we might assume $y = w^* \cdot x + \text{Gaussian noise}$, or written another way $(y | x) \sim \mathcal{N}(w^* \cdot x, \sigma^2)$, and ask how well we recover w^* , e.g. by showing that $\|w - w^*\|$ goes to zero as $m \rightarrow \infty$.

Machine learners are generally more concerned with getting predictions right. We'll typically measure this with the *squared loss*,

$$\mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{1}{2} (w \cdot x - y)^2 \right] = \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{1}{2} (h_w(x) - y)^2 \right]. \quad (1.2)$$

For example, if $x \sim \mathcal{N}(\mu, \Sigma)$ and $y | x \sim \mathcal{N}(w^* \cdot x, \sigma^2)$, let $M = \mu\mu^\top + \Sigma$. Then $\mathbb{E}(w \cdot x - y)^2 = \sigma^2 + (w - w^*)^\top M (w - w^*) \leq \sigma^2 + \|M\| \|w - w^*\|^2$, where the operator norm $\|M\|$ is constant for a given problem.

Often, recovering the “right” parameter vector, i.e. finding small $\|w - w^*\|$, implies that the predictive error is small (though not always, with nasty enough \mathcal{D}).

There are many situations, though, where you can have large $\|w - w^*\|$ but small predictive error. For instance, imagine that the first dimension of x is a person’s height in metres, and the second dimension is their height in inches, both measured to arbitrary precision. Because one inch is exactly 2.54 centimetres, the w vectors $(1, 0, \dots)$ and $(0, 1/0.0254, \dots)$ give *identical* predictions. So do any (w_1, w_2, \dots) such that $w_1 + .0254w_2 = 1$. There are infinitely many such vectors, and they can be arbitrarily far from w^* . Statisticians call (versions of) this problem *multicollinearity* and talk about how it causes issues with *identifiability*. For the most part, machine learners ignore this kind of problem; the different w s give identical predictions on \mathcal{D} . This is in some ways good, because these problems can be far worse with more complicated kinds of models, such as deep learning; it’s bad in other ways, since although these examples have identical predictions on \mathcal{D} , they can have very different predictions on *other* distributions!

Anyway, we’d ideally like to solve the following optimization problem, which minimizes a quantity known variously as the *risk*, the *population loss*, or various other names:

$$\min_w \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{1}{2} (w \cdot x - y)^2 \right] = \min_h \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[\frac{1}{2} (h(x) - y)^2 \right]. \quad (1.3)$$

We don’t have direct access to \mathcal{D} , though. Instead, the most common choice is to estimate that expectation with an empirical average on the training set (simple Monte Carlo), and instead solve

$$\min_w \frac{1}{2m} \sum_{i=1}^m (w \cdot x_i - y_i)^2 = \min_h \frac{1}{2m} \sum_{i=1}^m (h(x_i) - y_i)^2. \quad (1.4)$$

The thing we minimize here is called the *training loss*, the *empirical risk*, or various other names.

As you’ve probably seen before, we can solve this minimization in closed form. Let $\mathbf{X} \in \mathbb{R}^{m \times d}$ be the matrix whose i th row is x_i^\top , and $\mathbf{y} \in \mathbb{R}^m$ the vector whose i th entry is y_i . Then the objective is $\frac{1}{2m} \|\mathbf{X}w - \mathbf{y}\|^2$, which is a convex problem whose objective has gradient $\frac{1}{m} \mathbf{X}^\top (\mathbf{X}w - \mathbf{y})$. That’s zero iff $\mathbf{X}^\top \mathbf{X}w = \mathbf{X}^\top \mathbf{y}$; if $\mathbf{X}^\top \mathbf{X}$ (which is $d \times d$ of rank at most $\min(n, d)$) is invertible, there’s a unique solution $w = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{X}^\dagger \mathbf{y}$, where \mathbf{X}^\dagger is the [pseudoinverse](#). Otherwise, there are infinitely many minimizers, which can be expressed as $\mathbf{X}^\dagger \mathbf{y} + z$ where z is any vector in the null space of \mathbf{X} . The training set predictions for any of these solutions is the same: $\mathbf{X}w = \mathbf{X} \mathbf{X}^\dagger \mathbf{y} + \mathbf{X}z = \mathbf{X} \mathbf{X}^\dagger \mathbf{y}$.

The metres-and-inches example above is one such case: since one column is 0.0254 times the other, \mathbf{X} is not full-rank.

Having found this solution, we have lots of questions to ask: how well did solving (1.4) do as a proxy for the problem (1.3)? That is, given an estimate \hat{w} or equivalently \hat{h} , what is the loss (1.2) of that predictor? Is it likely to be small, in different situations? Is it as small as any algorithm could hope to achieve? If we try to estimate (1.2) for that solution with a test set, how tight should we expect our estimate to be? If $\mathbf{X}^\top \mathbf{X}$ isn’t invertible (always the case if $d > n$), which of the infinitely many minimizers should we pick?

For linear regression in particular, many of these questions can be tackled with basic linear algebra. You may have seen some of them in statistics courses; Chapter 3 of the textbook of Bach [[Bach24](#)] does some of these analyses framed in the language of learning theory. This is quite interesting, but we’re not going to pursue that

approach in any more detail in this course, instead generalizing to other problems.

1.2 GENERAL PROBLEM SETUP

Our default, general learning problem is as follows:

- Instead of a data distribution \mathcal{D} over $\mathbb{R}^d \times \mathbb{R}$, \mathcal{D} is over some domain \mathcal{Z} . For *supervised learning*, \mathcal{Z} is often a product space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ of (x, y) pairs, where x is an input object (e.g. an image) and y is a label (e.g. whether the image contains a dog). Occasionally, though, we want to learn over some other kind of space that doesn't have clear input-outputs.
- We have m independent, identically distributed samples $z_1, \dots, z_m \sim \mathcal{D}$, collected in a training "set" $S = (z_1, \dots, z_m) \sim \mathcal{D}^m$.
- We have a *hypothesis class* \mathcal{H} . In supervised learning, this is usually a set of predictors $h : \mathcal{X} \rightarrow \hat{\mathcal{Y}}$, a space of prediction functions.

– In linear regression, \mathcal{H} was the set of d -dimensional linear predictors, $\{x \mapsto w \cdot x : w \in \mathbb{R}^d\}$.

– We could use bounded-norm linear predictors, $\{x \mapsto w \cdot x : \|w\| \leq B\}$.

– We could use decision trees of a certain depth, decision forests of a certain size, neural networks of a certain architecture,

– Often, $\hat{\mathcal{Y}} = \mathcal{Y}$, but it might not; for example, it's common to have a problem with binary labels so that $\mathcal{Y} = \{0, 1\}$, but to make probabilistic predictions in $\hat{\mathcal{Y}} = [0, 1]$, or general confidence predictions in \mathbb{R} .

- We have a *loss function* $\ell : \mathcal{H} \times \mathcal{Z} \rightarrow \mathbb{R}$. In supervised learning, this often takes the form $\ell(h, (x, y)) = l(h(x), y)$ for some $l : \hat{\mathcal{Y}} \times \mathcal{Y} \rightarrow \mathbb{R}$. Some common examples:

– Squared loss: $l(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$. (Sometimes the $\frac{1}{2}$ isn't included.)

– Zero-one loss: $l(\hat{y}, y) = \mathbb{1}(\hat{y} \neq y)$, usually used for $\mathcal{Y} = \hat{\mathcal{Y}}$ a discrete set of labels. This corresponds to one minus the *accuracy* of a predictor.

– Logistic loss: $l(\hat{y}, y) = \log(1 + \exp(-\hat{y}y))$ for $\hat{\mathcal{Y}} = \mathbb{R}$, $\mathcal{Y} = \{-1, 1\}$. This loss $\rightarrow 0$ if $\hat{y} \rightarrow y\infty$, i.e. if $y = 1$ and $\hat{y} \rightarrow \infty$, or $y = -1$ and $\hat{y} \rightarrow -\infty$: you're very confidently right. It's $\log 2$ if $\hat{y} = 0$, a totally ambiguous prediction. The loss goes $\rightarrow \infty$ if $\hat{y} \rightarrow (-y)\infty$: you're very confidently wrong.

– Softmax cross-entropy loss, a multi-class generalization of logistic: here $\mathcal{Y} = [k] = \{1, 2, \dots, k\}$, $\hat{\mathcal{Y}} = \mathbb{R}^k$ is the space of logits, and the loss is

$$l(\hat{y}, y) = -\log \frac{\exp(\hat{y}_y)}{\sum_{j=1}^k \exp(\hat{y}_j)} = -\hat{y}_y + \log \sum_{j=1}^k \exp(\hat{y}_j).$$

- $L_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}} \ell(h, z) = \mathbb{E}_{(x,y) \sim \mathcal{D}} l(h(x), y)$ is called the *risk*, the *population loss*, the *true loss*, etc; this was (1.2) in logistic regression.

- $L_S(h) = \frac{1}{m} \sum_{i=1}^m \ell(h, z_i) = \frac{1}{m} \sum_{i=1}^m l(h(x_i), y_i)$ is the *empirical risk*, the *sample loss*, the *training loss* (if S is the training set), etc.

- A learning algorithm \mathcal{A} is a function that takes in a sample S and returns a hypothesis in \mathcal{H} . Ideally, one with low risk.

$x \mapsto 2x + 3$ means "the function which, given the argument x , returns $2x + 3$ "; \mathcal{H} is a set of functions. This is like `lambda x: 2*x+3` in Python.

The function `1` returns one if its boolean argument is true, and zero if not.

Here's a recap of our notation, and a quick reference for how to translate notations across some relevant textbooks.

	These notes	[SSBD14]	[MRT18]	[Bach24]	[Zhang23]
Data distribution	\mathcal{D}	\mathcal{D}	\mathcal{D}	\mathcal{D}	\mathcal{D}
Number of samples	m	m	m	n	n
Sample set	S	S	S	\mathcal{D}_n	S_n
Hypothesis/parameter	$h \in \mathcal{H}$	$h \in \mathcal{H}$	$h \in \mathcal{H}$	$\theta \in \Theta$	$w \in \Omega$
Prediction on x	$h(x)$	$h(x)$	$h(x)$	$f_\theta(x)$	$f(w, x)$
Loss of hypothesis	$\ell(h, z)$	$\ell(h, z)$	–	–	$\phi(w, z)$
Loss of prediction	$l_y(\hat{y})$	–	$L(\hat{y}, y)$	$\ell(y, \hat{y})$	$L(\hat{y}, y)$
Empirical risk	$L_S(h)$	$L_S(h)$	$\hat{R}_S(h)$	$\hat{\mathcal{R}}(\theta)$	$\phi(w, \mathcal{D})$
Population risk	$L_{\mathcal{D}}(h)$	$L_{\mathcal{D}}(h)$	$R(h)$	$\mathcal{R}(\theta)$	$\phi(w, S_n)$

1.3 EMPIRICAL RISK MINIMIZATION

The most common general learning algorithm we'll think about, the general case of (1.4), is *empirical risk minimization*:

$$\text{ERM}(S) \in \arg \min_{h \in \mathcal{H}} L_S(h).$$

If \mathcal{H} is infinite, there might be not be a minimizer. We usually won't worry about this explicitly, but basically everything we talk about could be generalized to approximate minimizers.

If there are ties, by default we think of the algorithm returning any arbitrary choice.

The returned hypothesis, $\text{ERM}(S)$, which we will also often denote \hat{h}_S , is called an empirical risk minimizer ("an ERM").

For example, ERM with the squared loss and $\mathcal{H} = \{x \mapsto w \cdot x\}$ does indeed recover ordinary least squares:

$$\begin{aligned} \text{ERM}(S) &\in \arg \min_{h \in \{x \mapsto w \cdot x : w \in \mathbb{R}^d\}} L_S(h) \\ &= \arg \min_{h \in \{x \mapsto w \cdot x : w \in \mathbb{R}^d\}} \frac{1}{m} \sum_{i=1}^m \ell(h, z_i) \\ &= \arg \min_{h \in \{x \mapsto w \cdot x : w \in \mathbb{R}^d\}} \frac{1}{m} \sum_{i=1}^m l_{y_i}(h(x_i)) \\ &= \left\{ x \mapsto x \cdot \hat{w} : \hat{w} \in \arg \min_{w \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^m l(w \cdot x_i, y_i) \right\} \\ &= \left\{ x \mapsto x \cdot \hat{w} : \hat{w} \in \arg \min_{w \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^m \frac{1}{2} (w \cdot x_i - y_i)^2 \right\}, \end{aligned}$$

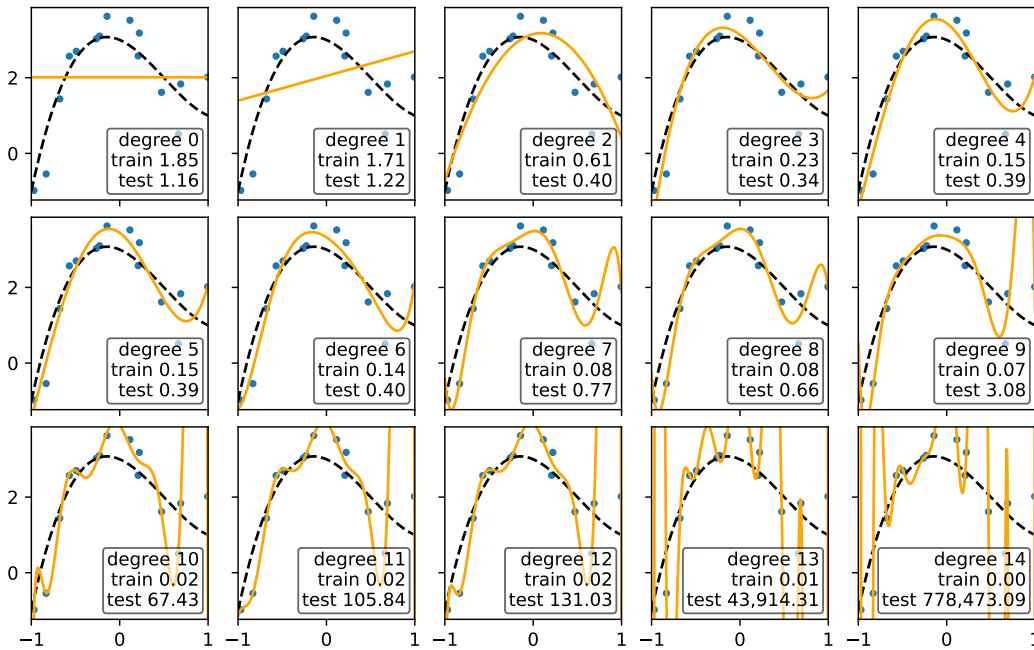
In our notation here, $\text{ERM}(S)$ is returning a function (which makes these last couple of lines slightly tedious); we could equally well have let \mathcal{H} be a set of parameter vectors and define a loss on parameters, $\ell(w, (x, y)) = \frac{1}{2}(x \cdot w - y)^2$.

and now \hat{w} in the last line is probably what your intro stats class wrote down in the first place as the definition of linear regression.

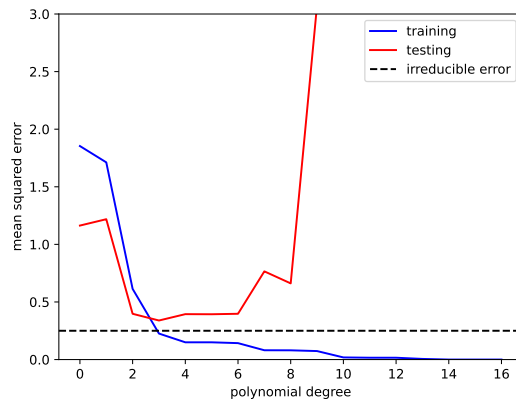
We know that $L_S(\text{ERM}(S))$ is small by definition, but when can we expect $L_{\mathcal{D}}(\text{ERM}(S))$ to be small? The first big chunk of this course is about this question in particular. There are several ways to analyze this; the classic way is by making sure we choose an appropriate hypothesis class \mathcal{H} . If \mathcal{H} is too simple, you'll never be able to learn the pattern you're looking for, but if it's too complicated, you'll *overfit* and pick one that seems good by chance, i.e. has good $L_S(\text{ERM}(S))$ but bad $L_{\mathcal{D}}(\text{ERM}(S))$.

Figure 1.1 illustrates this trade-off for polynomial regression. This is similar to

what you saw in your intro machine learning class; one of the things we'll do in this course is formalize this general intuition and prove theorems about it.



(a) Polynomial regression, $h(x) = w_0 + w_1x + w_2x^2 + \dots + w_kx^k$, for increasing k , to data points shown in blue. ERM fits are in orange; dashed black lines show $\mathbb{E}[y | x]$, a cubic function. Text gives mean squared error for training and testing sets.



(b) Training and test errors from Figure 1.1a.

Figure 1.1: Underfitting to overfitting as \mathcal{H} gets bigger.

1.4 ERROR DECOMPOSITIONS

Here's some standard terminology to know to formalize that intuition. For any estimator $\hat{h}_S \in \mathcal{H}$ (not necessarily just the ERM), we can write

$$\underbrace{L_{\mathcal{D}}(\hat{h}_S) - L_{\text{bayes}}}_{\text{excess error}} = \underbrace{L_{\mathcal{D}}(\hat{h}_S) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h)}_{\text{estimation error}} + \underbrace{\inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_{\text{bayes}}}_{\text{approximation error}}.$$

The *excess error* is how much worse you are than the irreducible error L_{bayes} , also called the Bayes error or the error of the Bayes predictor (see A1 Q2 for more). No predictor, no matter its form, could do better than this: there's just inherent noise in the problem. The general $\ell(h, z)$ form unfortunately doesn't make this

easy to define (the domain is \mathcal{H}), but for $L_y(\hat{y})$ we can say something like $L_{\text{bayes}} = \inf_{h: \mathcal{X} \rightarrow \hat{y} \text{ measurable}} L_y(h(x))$.

The *estimation error*, also called the *statistical error*, is the error that comes about from using your algorithm \hat{h}_S rather than picking the best possible predictor in \mathcal{H} . As $m \rightarrow \infty$, this should (ideally) go to zero.

CPSC 340 used to use “approximation error” for the generalization gap, $L_{\mathcal{D}}(h) - L_S(h)$. This was a nonstandard use; it’s been changed now in 340, and you should wipe it from your memory. :)

The *approximation error* doesn’t (directly) depend on the number of samples you see: it’s a function only of how well your hypothesis class \mathcal{H} can do, regardless of estimation.

For example, in the polynomial regression case of Figure 1.1, using a \mathcal{H} of linear functions results in some approximation error, but not much estimation error (because linear functions are easy to fit). Using a \mathcal{H} of degree-fifteen polynomials has zero approximation error (it contains the Bayes predictor) but really high estimation error (too many parameters to fit).

Intuitively, as \mathcal{H} gets “bigger,” approximation error decreases but estimation error increases. Usually, approximation error is pretty problem-specific, but we’ll see at least a few examples of formal analyses of it later in the course. First, we’ll think about estimation error bounds.

1.4.1 ERM estimation error

Our usual basic way to prove when ERM generalizes well is to take the following decomposition, where we compare the loss of \hat{h}_S to the loss of some arbitrary comparator hypothesis $h^* \in \mathcal{H}$. Note that we’ll usually think of h^* as being roughly the best predictor in \mathcal{H} , but we don’t require that, since it might not exist if \mathcal{H} is infinite; instead we’ll start by just comparing to any arbitrary predictor.

$$\begin{aligned} L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) &= L_{\mathcal{D}}(\hat{h}_S) - \underbrace{L_S(\hat{h}_S)}_0 + \underbrace{L_S(\hat{h}_S) - L_S(h^*)}_0 + L_S(h^*) - L_{\mathcal{D}}(h^*) \\ &= (L_{\mathcal{D}}(\hat{h}_S) - L_S(\hat{h}_S)) + \underbrace{L_S(\hat{h}_S) - L_S(h^*)}_{\leq 0: \hat{h}_S \text{ minimizes } L_S} + (L_S(h^*) - L_{\mathcal{D}}(h^*)) \\ &\leq \underbrace{L_{\mathcal{D}}(\hat{h}_S) - L_S(\hat{h}_S)}_{\text{A: } \hat{h}_S \text{'s overfitting}} + \underbrace{L_S(h^*) - L_{\mathcal{D}}(h^*)}_{\text{B: } h^* \text{'s "underfitting"}}. \end{aligned} \quad (1.5)$$

So, if we can bound A and B, then we can say that \hat{h}_S isn’t too much worse than h^* .

Now, as long as our bound on B doesn’t depend on the particular choice of h^* , then this implies that

$$L_{\mathcal{D}}(\hat{h}_S) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq A + B.$$

The next few weeks will be devoted to bounding A + B, how much worse \hat{h}_S is than the best possible thing in \mathcal{H} .

If you aren’t familiar with inf, it’s like min but makes sense even if there isn’t a minimizer (it’s the largest lower bound). For example, $\inf_{x \in \mathbb{R}: x > 0} x = 0$ even though 0 isn’t in that set.

2 ERM with finite hypothesis classes

In (1.5) we showed that, for any $h^* \in \mathcal{H}$,

$$L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) \leq \left(L_{\mathcal{D}}(\hat{h}_S) - L_S(\hat{h}_S) \right) + \left(L_S(h^*) - L_{\mathcal{D}}(h^*) \right).$$

We'd like to bound these two terms, which would then give us a bound on how much worse \hat{h}_S is than h^* , the best thing ERM could have done. The first thing to note, though, is that anything with an S in it – so everything above except for $L_{\mathcal{D}}(h^*)$ – depends on the draw of the random training set S . It's possible that we could get some ridiculously unlikely training set where everything behaves nonsensically. So we'll need to do some kind of probabilistic bound.

Let's now try to study that formally.

2.1 ESTIMATION ERROR: ASYMPTOTICS

Let's start with the second term from (1.5):

$$L_S(h^*) - L_{\mathcal{D}}(h^*) = \frac{1}{m} \sum_{i=1}^m \ell(h^*, z_i) - \mathbb{E}_{z \sim \mathcal{D}} \ell(h^*, z).$$

Remember that the only thing that's random here is $S = (z_1, \dots, z_m)$, since h^* is just some fixed hypothesis. So, we can frame this as $\frac{1}{m} \sum_{i=1}^m R_i$, where the $R_i = \ell(h^*, z_i)$ are iid random variables with mean $\mathbb{E} \ell(h^*, z_i) = L_{\mathcal{D}}(h^*)$. The law of large numbers therefore guarantees that as $m \rightarrow \infty$, $\frac{1}{m} \sum_{i=1}^m R_i$ converges (almost surely) to $L_{\mathcal{D}}(h^*)$, and so this term in the bound converges to zero.

In fact, for many \mathcal{H} and ℓ , the other term will also have the same property, implying (if h^* is a minimizer of $L_{\mathcal{D}}$) that $L_{\mathcal{D}}(\hat{h}_S) \rightarrow L_{\mathcal{D}}(h^*)$. Various formalizations of this last property are called *consistency*, and it's a nice property to have: eventually, your learning algorithm works as well as it could have. One problem with this notion, though, is that this is *all* it tells you. There's no guarantee about what happens with $m = 1,000$, or when going from $m = 1,000$ to $m = 1,000,000$, or anything at all other than “eventually it works.”

A more precise analysis might use the central limit theorem. Let $\sigma^2 = \text{Var}[R_i]$ and assume this is finite; informally, the CLT then says that $\frac{1}{m} \sum_{i=1}^m R_i$ behaves like

$\mathcal{N}(0, \sigma^2/m)$. In fact, it's often true that the first term is also asymptotically normal. This is a nicer result than before: it still doesn't say anything particular for a finite m (maybe the CLT takes a long time to kick in), but it tells us a lot about the asymptotic behaviour, including both its limiting value but also roughly how much variation

Formally, we'd write $\frac{1}{\sqrt{m}} \sum_{i=1}^m (R_i - \mathbb{E} R_i) \xrightarrow{d} \mathcal{N}(0, \sigma^2)$.

we can expect around that value.

It can be tough to find these exact limiting distributions in general, though, and they're not always true (e.g. the one I didn't state for the first term above has some kind-of strict requirements on the way that h is parameterized). A similar but somewhat looser style of bound is to say that the excess error is $\mathcal{O}_p(1/\sqrt{m})$, which is implied by the CLT result above, but can also be much easier to show. Again, this doesn't imply anything for a finite m (just like how \mathcal{O} analyses don't say anything for finite input size on your algorithms), but they do say things like, for reasonably large m , observing four times as much data should roughly halve your excess error.

You can check [the wiki page](#) for a formal definition of \mathcal{O}_p , but it roughly means "with any constant probability, a sequence of sampled random variables is $\mathcal{O}(1/\sqrt{m})$."

The *most* preferred kind of result, though, is usually one with explicit constants: something like

$$\forall \delta > 0. \quad \Pr_{S \sim \mathcal{D}^m} \left(L_{\mathcal{D}}(\hat{h}_S) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq \sqrt{\frac{2}{m} \log \frac{|\mathcal{H}| + 1}{\delta}} \right) \geq 1 - \delta$$

or, where B is a problem parameter,

$$\mathbb{E}_{S \sim \mathcal{D}^m} L_{\mathcal{D}}(\hat{h}_S) \leq \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \sqrt{\frac{8B^2}{m}}.$$

These results give you a rate, but also apply to *any* m , not just eventually. (They might not be meaningful for small m , though; if you're using 0-1 loss, it's not very helpful to say the excess error is less than four!)

2.2 UNIFORM CONVERGENCE, BOUNDED LOSS

We're first going to assume that $\ell(h, z) \in [a, b]$ for all h, z ; usually $a = 0$ (but it won't hurt us to be more general), and e.g. for the 0-1 loss we have $b = 1$. For something like the square loss, it isn't "automatically" bounded, but it might be depending on \mathcal{H} and \mathcal{D} ; we'll discuss this later.

Recall that we have two things to bound in (1.5):

$$L_{\mathcal{D}}(\hat{h}_S) - L_S(\hat{h}_S) = \mathbb{E}_{z \sim \mathcal{D}} \ell(\hat{h}_S, z) - \frac{1}{m} \sum_{i=1}^m \ell(\hat{h}_S, z_i) \quad (\text{A})$$

and

$$L_S(h^*) - L_{\mathcal{D}}(h^*) = \frac{1}{m} \sum_{i=1}^m \ell(h^*, z_i) - \mathbb{E}_{z \sim \mathcal{D}} \ell(h^*, z). \quad (\text{B})$$

As we discussed, (B) is an average of iid random variables. We can bound this with the following form of *Hoeffding's inequality*, which we'll prove soon:

PROPOSITION 2.1 (Hoeffding, simple form). *Let (X_1, \dots, X_m) be independent with mean*

μ and almost surely bounded in $[a, b]$. Define $\bar{X} = \frac{1}{m} \sum_{i=1}^m X_i$. Then

$$\begin{aligned} \Pr\left(\bar{X} \leq \mu + (b-a)\sqrt{\frac{\log(1/\delta)}{2m}}\right) &\geq 1 - \delta \\ \Pr\left(\bar{X} \geq \mu - (b-a)\sqrt{\frac{\log(1/\delta)}{2m}}\right) &\geq 1 - \delta \\ \Pr\left(|\bar{X} - \mu| \leq (b-a)\sqrt{\frac{\log(2/\delta)}{2m}}\right) &\geq 1 - \delta. \end{aligned}$$

The first of these results immediately implies the other two: use the random variables $Y_i = -X_i$ for the second, and then use a union bound, Lemma 2.3, to get the third.

Applying this to the random variables $X_i = \ell(h^*, z_i)$ handles the bound for (B).

It's tempting to also try to apply this result directly to (A), which would then complete our bound and everything would be really simple. The problem is that the $\ell(\hat{h}_S, z_i)$ aren't independent! The choice of \hat{h}_S depends on *all* of S , i.e. on all of the other z_j , so $\ell(\hat{h}_S, z_1)$ and $\ell(\hat{h}_S, z_2)$ are *not* independent.

So, how can we bound this? The most common way is called *uniform convergence*. The idea is, if we know that $L_{\mathcal{D}}(h) - L_S(h)$ is small for *all* $h \in \mathcal{H}$, then it'll be small for \hat{h}_S , no matter how we pick it – since it's something in \mathcal{H} . That is, if we know that

$$\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq \varepsilon$$

then we also have that $L_{\mathcal{D}}(\hat{h}_S) - L_S(\hat{h}_S) \leq \varepsilon$. Or, stating it another way,

$$\Pr_{S \sim \mathcal{D}^m} \left(L_S(\hat{h}_S) - L_{\mathcal{D}}(\hat{h}_S) > \varepsilon \right) \leq \Pr_{S \sim \mathcal{D}^m} \left(\exists h \in \mathcal{H}. L_S(h) - L_{\mathcal{D}}(h) > \varepsilon \right), \quad (2.1)$$

and so bounding the right-hand side bounds the left-hand side.

How can we bound that?

2.3 FINITE \mathcal{H}

To start, we'll make a kind of drastic assumption: that \mathcal{H} is finite, i.e. we're only considering $|\mathcal{H}|$, say 500, possible hypotheses.

PROPOSITION 2.2. *Suppose $\ell(z, h)$ is almost surely bounded in $[a, b]$, \mathcal{H} is finite, and \hat{h}_S is any ERM in \mathcal{H} . Then for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ it holds that*

$$L_{\mathcal{D}}(\hat{h}_S) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq (b-a)\sqrt{\frac{2}{m} \log \frac{|\mathcal{H}| + 1}{\delta}}.$$

Proof. For any hypothesis h , we can allow it a “failure probability” of $\delta/(|\mathcal{H}| + 1)$ in Hoeffding's inequality:

$$\Pr_{S \sim \mathcal{D}^m} \left(L_{\mathcal{D}}(h) - L_S(h) > (b-a)\sqrt{\frac{1}{2m} \log \frac{|\mathcal{H}| + 1}{\delta}} \right) \leq \frac{\delta}{|\mathcal{H}| + 1}.$$

If we do this for *each* hypothesis $h \in \mathcal{H}$, we know that the probability of each particular h being bad is low. We then want to combine them into the probability that *anything* is bad; we can do this with a union bound.

This *fact* is really useful. **LEMMA 2.3** (Union bound). For any two events A and B , $\Pr(A \cup B) \leq \Pr(A) + \Pr(B)$.

Combining all of them together, the probability that *any* h happens to look way better than it is can be bounded as

$$\Pr_{S \sim \mathcal{D}^m} \left(\exists h \in \mathcal{H}. \quad L_{\mathcal{D}}(h) - L_S(h) > (b-a) \sqrt{\frac{1}{2m} \log \frac{|\mathcal{H}|+1}{\delta}} \right) \leq |\mathcal{H}| \frac{\delta}{|\mathcal{H}|+1}.$$

But we'll also need the other direction for (B): h^* in particular doesn't look way worse than it actually is. Giving it the same failure probability to make things nice,

$$\Pr_{S \sim \mathcal{D}^m} \left(L_S(h^*) - L_{\mathcal{D}}(h^*) > (b-a) \sqrt{\frac{1}{2m} \log \frac{|\mathcal{H}|+1}{\delta}} \right) \leq \frac{\delta}{|\mathcal{H}|+1}.$$

Now, if (A) $\leq \varepsilon_A$ and (B) $\leq \varepsilon_B$, then (1.5) tells us that $L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) \leq (A) + (B) \leq \varepsilon_A + \varepsilon_B$. Using another union bound,

$$\begin{aligned} \Pr_{S \sim \mathcal{D}^m} \left(L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) > (b-a) \sqrt{\frac{1}{2m} \log \frac{|\mathcal{H}|+1}{\delta}} + (b-a) \sqrt{\frac{1}{2m} \log \frac{|\mathcal{H}|+1}{\delta}} \right) \\ = \Pr_{S \sim \mathcal{D}^m} \left(L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) > (b-a) \sqrt{\frac{2}{m} \log \frac{|\mathcal{H}|+1}{\delta}} \right) \\ \leq \frac{|\mathcal{H}|}{|\mathcal{H}|+1} \delta + \frac{1}{|\mathcal{H}|+1} \delta = \delta. \quad \square \end{aligned}$$

Another way to state Proposition 2.2 is that with m samples, we can achieve excess error at most ε with probability at least $(|\mathcal{H}|+1) \exp\left(-\frac{m\varepsilon^2}{2(b-a)^2}\right)$.

Or, alternately, we can say that we can achieve excess error at most ε with probability at least $1 - \delta$ if we have at least $\frac{2(b-a)^2}{\varepsilon^2} \log \frac{|\mathcal{H}|+1}{\delta}$ samples.

2.3.1 Is this finiteness assumption reasonable?

Every \mathcal{H} we use in practice is finite. Our models are represented on a computer with bounded memory, so we consider no more than $2^{\text{max number of bits}}$ hypotheses.

On the other hand, $|\mathcal{H}|$ might be really large. Typical vision CNNs are around a few hundred megabytes: 100 megabytes is 800,000,000 bits, and $\log(|\mathcal{H}|+1) \approx \log 2^{800,000,000} = 800,000,000 \log 2 \approx 554,517,744$ is quite big. For 0-1 loss, this would mean that for our bound to show that ERM learns a 100-MB network even to within an extremely loose $\varepsilon = 20\%$ additive error with probability at least $1 - \delta = 50\%$, we'd need

$$m \geq \frac{2}{0.2^2} \left(\log(|\mathcal{H}|+1) + \log \frac{1}{0.5} \right) \approx 50 (554 \text{ million} + 0.7) \approx 28 \text{ billion}.$$

100 MB is a relatively small model these days (ViTs are usually a few gigabytes), and 28 billion is a *lot* of samples.

But the union bound we did over \mathcal{H} ignores *all* structure in \mathcal{H} . If we change just one parameter by 0.00001, then we're treating the error of that new hypothesis totally separately, when in reality those two errors are tightly correlated. We'll approach that soon, with various techniques that will also allow us to handle \mathcal{H} with infinite size; but first, we'll go back and prove Hoeffding's inequality.

3 Concentration inequalities

We'll now prove Hoeffding's inequality (Proposition 2.1), and learn a bunch of useful stuff along the way.

3.1 MARKOV

We'll start with the following surprisingly simple bound, which turns out to be the basis for just about everything:

PROPOSITION 3.1 (Markov's inequality). *If X is a nonnegative-valued random variable, then $\Pr(X \geq t) \leq \frac{1}{t} \mathbb{E} X$ for all $t > 0$.*

Proof. We know $X \geq 0$. We also know, if $X \geq t$, then $X \geq t$. Combining those two statements, we can write $X \geq t \mathbf{1}(X \geq t)$. Now take the expectation of both sides of that inequality, giving $\mathbb{E} X \geq t \mathbb{E} \mathbf{1}(X \geq t) = t \Pr(X \geq t)$. Rearrange. \square

This was actually proved by Markov's PhD advisor Chebyshev. Luckily, though, Chebyshev has another inequality named after him:

PROPOSITION 3.2 (Chebyshev's inequality). *For any X , $\Pr(|X - \mathbb{E} X| \geq \varepsilon) \leq \frac{1}{\varepsilon^2} \text{Var } X$.*

Proof. $(X - \mathbb{E} X)^2$ is a nonnegative random variable; applying Markov gives $\Pr((X - \mathbb{E} X)^2 \geq t) \leq \frac{1}{t} \mathbb{E}(X - \mathbb{E} X)^2$. Change variables to $t = \varepsilon^2$. \square

Equivalently, with probability at least $1 - \delta$, $|X - \mathbb{E} X| < \sqrt{\text{Var}[X]/\delta}$.

Let's consider iid X_1, \dots, X_m , each with mean μ and variance σ^2 . Then the random variable $\bar{X} = \frac{1}{m} \sum_{i=1}^m X_i$ has mean μ and variance σ^2/m , so Chebyshev gives that $|\bar{X} - \mu| \leq \sigma/\sqrt{m\delta}$. This is $\mathcal{O}_p(1/\sqrt{m})$, as expected, so sometimes this is good enough.

But the dependence on δ is really quite bad compared to what we'd like. For instance, if the X_i are normal so that \bar{X} is too, then in (3.2) below we'll obtain $\bar{X} - \mu \leq \frac{\sigma}{\sqrt{m}} \sqrt{2 \log \frac{1}{\delta}}$. To emphasize the difference:

δ	0.1	0.01	0.001	0.0001	0.00001
$1/\sqrt{\delta}$	3.2	10.0	31.6	100.0	316.2
$\sqrt{2 \log \frac{1}{\delta}}$	2.2	3.0	3.7	4.3	4.8

Chebyshev's inequality is sharp, meaning that it can be an equality in certain cases; this happens for random variables of the form $\Pr(X = 0) = 1 - \delta$, $\Pr(X = 1/\sqrt{\delta}) = \Pr(X = -1/\sqrt{\delta}) = \frac{1}{2}\delta$. This X has mean 0 and variance 1, but it still has a big probability of being really far from zero. "Typical" random variables, like Gaussians, don't look like this. So here's another analysis that takes this into account.

3.2 CHERNOFF BOUNDS

Perhaps the most useful category of results are called Chernoff bounds; they're based on

$$\Pr(X \geq \mathbb{E}X + \varepsilon) = \Pr\left(e^{\lambda(X - \mathbb{E}X)} \geq e^{\lambda\varepsilon}\right) \leq e^{-\lambda\varepsilon} \mathbb{E} e^{\lambda(X - \mathbb{E}X)}, \quad (3.1)$$

where we applied Markov to the nonnegative random variable $\exp(\lambda(X - \mathbb{E}X))$ for any $\lambda > 0$.

The quantity $M_X(\lambda) = \mathbb{E} e^{\lambda(X - \mathbb{E}X)}$ is known as the centred *moment-generating function*; recalling that $e^t = 1 + t + \frac{t^2}{2!} + \frac{t^3}{3!} + \dots$ and writing $\mu = \mathbb{E}X$, we have

$$M_X(\lambda) = \mathbb{E} e^{\lambda(X - \mu)} = 1 + \lambda \mathbb{E}[X - \mu] + \frac{\lambda^2}{2!} \mathbb{E}[(X - \mu)^2] + \frac{\lambda^3}{3!} \mathbb{E}[(X - \mu)^3] + \dots$$

So, taking the k th derivative of the centred mgf and then evaluating at $\lambda = 0$ gives $M_X^{(k)}(0) = \mathbb{E}[(X - \mu)^k]$.

PROPOSITION 3.3. *If $X \sim \mathcal{N}(\mu, \sigma^2)$, then $\mathbb{E} e^{\lambda(X - \mu)} = e^{\frac{1}{2}\lambda^2\sigma^2}$.*

Proof. Let's start with $X \sim \mathcal{N}(0, 1)$. We can write

$$\begin{aligned} \mathbb{E}_{X \sim \mathcal{N}(0,1)} e^{\lambda X} &= \int \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} e^{\lambda x} dx \\ &= \int \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2 + \lambda x - \frac{1}{2}\lambda^2 + \frac{1}{2}\lambda^2} dx \\ &= e^{\frac{1}{2}\lambda^2} \int \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x - \lambda)^2} dx \\ &= e^{\frac{1}{2}\lambda^2}, \end{aligned}$$

since the last integral is just the total probability density of an $\mathcal{N}(\lambda, 1)$ random variable. To handle $Y = \mathcal{N}(\mu, \sigma^2)$, note that this is equivalent to $\sigma X + \mu$, so

$$e^{\lambda(Y - \mathbb{E}Y)} = e^{\lambda(\sigma X + \mu - \mathbb{E}(\sigma X + \mu))} = e^{\lambda(\sigma X)} = e^{(\lambda\sigma)X} = e^{\frac{1}{2}\sigma^2\lambda^2}. \quad \square$$

Plugging Proposition 3.3 into (3.1), for $X \sim \mathcal{N}(\mu, \sigma^2)$, it holds for any $\lambda > 0$ that

$$\Pr(X \geq \mu + \varepsilon) \leq e^{-\lambda\varepsilon} e^{\frac{1}{2}\sigma^2\lambda^2}.$$

The value of λ only appears on the right-hand side, not the left. So we might as well find the best value of λ to use: the one that gives the tightest bound. Let's optimize this in λ : noting that exp is monotonic, we can just check that $\frac{1}{2}\sigma^2\lambda^2 - \lambda\varepsilon$ has derivative $\sigma^2\lambda - \varepsilon$, which is zero when $\lambda = \varepsilon/\sigma^2 > 0$. (And this is indeed a max, since the second derivative is $\sigma^2 > 0$.) Plugging in that value of λ , we get the bound

$$\Pr(X \geq \mu + \varepsilon) \leq \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right). \quad (3.2)$$

Equivalently, $X < \mu + \sigma\sqrt{2 \log \frac{1}{\delta}}$ with probability at least $1 - \delta$.

3.3 SUBGAUSSIAN VARIABLES

In fact, the only place we used the Gaussian assumption in this argument was in that $\mathbb{E} e^{\lambda(X - \mathbb{E}X)} \leq e^{\frac{1}{2}\lambda^2\sigma^2}$. So we can generalize the result to anything satisfying that

condition, which we call *subgaussian*:

DEFINITION 3.4. A random variable X with mean $\mu = \mathbb{E}[X]$ is called *subgaussian with parameter* $\sigma \geq 0$, written $X \in \mathcal{SG}(\sigma)$, if its centred moment-generating function $\mathbb{E}[e^{\lambda(X-\mu)}]$ exists and satisfies that for all $\lambda \in \mathbb{R}$, $\mathbb{E}[e^{\lambda(X-\mu)}] \leq e^{\frac{1}{2}\lambda^2\sigma^2}$.

Watch out with other sources; notation for subgaussians is not very standardized, in particular whether the parameter is σ or σ^2 . Also “ $X \in \mathcal{SG}(\sigma)$ ” is kind of weird; probably “ $\text{Law}(X) \in \mathcal{SG}(\sigma)$ ” would be better, but oh well.

As we just saw, normal variables with variance σ^2 are $\mathcal{SG}(\sigma)$. Notice also that if $\sigma_1 < \sigma_2$, then anything that’s $\mathcal{SG}(\sigma_1)$ is also $\mathcal{SG}(\sigma_2)$.

PROPOSITION 3.5 (Hoeffding’s lemma). *If $\Pr(a \leq X \leq b) = 1$, X is $\mathcal{SG}\left(\frac{b-a}{2}\right)$.*

Proof. See Section 3.3.1; we’ll probably skip this in class. □

Here are some useful properties about building subgaussian variables:

PROPOSITION 3.6. *If $X_1 \in \mathcal{SG}(\sigma_1)$ and $X_2 \in \mathcal{SG}(\sigma_2)$ are independent random variables, then $X_1 + X_2 \in \mathcal{SG}(\sqrt{\sigma_1^2 + \sigma_2^2})$.*

Proof. $\mathbb{E}[e^{\lambda(X_1+X_2-\mathbb{E}[X_1+X_2])}] = \mathbb{E}[e^{\lambda(X_1-\mathbb{E}[X_1])}] \mathbb{E}[e^{\lambda(X_2-\mathbb{E}[X_2])}]$ by independence. Bounding each expectation, this is at most $e^{\frac{1}{2}\lambda^2\sigma_1^2} e^{\frac{1}{2}\lambda^2\sigma_2^2} = e^{\frac{1}{2}\lambda^2(\sigma_1^2 + \sigma_2^2)}$. □

PROPOSITION 3.7. *If $X \in \mathcal{SG}(\sigma)$, then $aX + b \in \mathcal{SG}(|a|\sigma)$ for any $a, b \in \mathbb{R}$.*

Proof. $\mathbb{E}[e^{\lambda(aX+b-\mathbb{E}[aX+b])}] = \mathbb{E}[e^{(a\lambda)(X-\mathbb{E}[X])}] \leq e^{\frac{1}{2}(a\lambda)^2\sigma^2} = e^{\frac{1}{2}\lambda^2(|a|\sigma)^2}$. □

PROPOSITION 3.8 (Chernoff bound for subgaussians). *If $X \in \mathcal{SG}(\sigma)$, then $\Pr(X \geq \mathbb{E}X + \varepsilon) \leq \exp\left(-\frac{\varepsilon^2}{2\sigma^2}\right)$ for $\varepsilon \geq 0$.*

Proof. Exactly as the argument leading from (3.1) to (3.2). □

Since $-X$ is also $\mathcal{SG}(\sigma)$ by Proposition 3.7, the same bound holds for a lower deviation $\Pr(X \leq \mathbb{E}X - t)$. A union bound then immediately gives $\Pr(|X - \mu| \geq t) \leq 2 \exp\left(-\frac{t^2}{2\sigma^2}\right)$.

PROPOSITION 3.9 (Hoeffding). *If X_1, \dots, X_m are independent and each $\mathcal{SG}(\sigma_i)$ with mean μ_i , for all $\varepsilon \geq 0$*

$$\Pr\left(\frac{1}{m} \sum_{i=1}^m X_i \geq \frac{1}{m} \sum_{i=1}^m \mu_i + \varepsilon\right) \leq \exp\left(-\frac{m^2\varepsilon^2}{2 \sum_{i=1}^m \sigma_i^2}\right).$$

Proof. By Propositions 3.6 and 3.7, $\frac{1}{m} \sum_{i=1}^m X_i \in \mathcal{SG}\left(\frac{1}{m} \sqrt{\sum_{i=1}^m \sigma_i^2}\right)$. Then apply Proposition 3.8. □

If the X_i have the same mean $\mu_i = \mu$ and parameter $\sigma_i = \sigma$, this becomes

$$\Pr\left(\frac{1}{m} \sum_{i=1}^m X_i \geq \mu + \varepsilon\right) \leq \exp\left(-\frac{m\varepsilon^2}{2\sigma^2}\right), \quad \text{(Hoeffding)}$$

which can also be stated as that, with probability at least $1 - \delta$,

$$\frac{1}{m} \sum_{i=1}^m X_i < \mu + \sigma \sqrt{\frac{2}{m} \log \frac{1}{\delta}}. \quad (\text{Hoeffding'})$$

The form of Hoeffding we saw before, Proposition 2.1, follows immediately from Proposition 3.5 and (Hoeffding').

3.3.1 Proof of Hoeffding's lemma

Wikipedia's proof is similar, but I think a little less clean. Other proofs are based more explicitly on convexity, but use either opaque changes of variable [SSBD14, Lemma B.7] or compute some pretty nasty derivatives [MRT18, Lemma D.1]. There's also a proof strategy based on "exponential tilting" (see [BLM13, Lemma 2.2], [Rag14, Lemma 1], or [Wai19, Exercise 2.4]) which is quite related but just overall a little more annoying. There are also proofs based on symmetrization (see [Wai19, Examples 2.3-2.4] or [Rom21]), which are nice but (a) have a worse constant and (b) require symmetrization, which is an important idea we'll cover soon but kind of hard to understand.

This proof roughly follows Zhang [Zhang23, Lemma 2.15].

LEMMA 3.10. *Let $X \sim \text{Bernoulli}(p)$. Then X is $\mathcal{SG}(1/2)$.*

Proof. The logarithm of the (uncentred) moment-generating function of X is

$$\psi(\lambda) = \log \mathbb{E} e^{\lambda X} = \log((1-p)e^0 + pe^\lambda).$$

This has derivatives

$$\begin{aligned} \psi'(\lambda) &= \frac{pe^\lambda}{(1-p)e^0 + pe^\lambda} \\ \psi''(\lambda) &= \frac{pe^\lambda}{(1-p)e^0 + pe^\lambda} - \frac{(pe^\lambda)^2}{((1-p)e^0 + pe^\lambda)^2} = \psi'(\lambda)(1 - \psi'(\lambda)). \end{aligned}$$

Since the function $x(1-x)$ has maximum $1/4$, $\psi''(\lambda) \leq 1/4$. By Taylor's theorem (in the Lagrange form), for any λ there exists some ξ_λ such that

$$\psi(\lambda) = \underbrace{\psi(0)}_0 + \lambda \underbrace{\psi'(0)}_p + \frac{1}{2} \lambda^2 \underbrace{\psi''(\xi_\lambda)}_{\leq 1/4} \leq \lambda p + \frac{1}{8} \lambda^2.$$

Thus the centred mgf satisfies

$$\mathbb{E} e^{\lambda(X - \mathbb{E} X)} = e^{-\lambda p} \mathbb{E} e^{\lambda X} \leq e^{-\lambda p} (e^{\lambda p + \frac{1}{8} \lambda^2}) = e^{\frac{1}{8} \lambda^2}. \quad \square$$

PROPOSITION 3.5 (Hoeffding's lemma). *If $\Pr(a \leq X \leq b) = 1$, X is $\mathcal{SG}(\frac{b-a}{2})$.*

Proof. Using $(X-a)/(b-a)$ and Proposition 3.7, we need only consider $a=0, b=1$.

Let $f(\lambda) = \mathbb{E} e^{\lambda X}$ be the (uncentred) mgf of X , and $g(\lambda) = (1-\mu)e^0 + \mu e^\lambda$ that of a Bernoulli(μ) variable, where $\mu = \mathbb{E} X$. For $\lambda \geq 0$,

$$f'(\lambda) = \frac{d}{d\lambda} \mathbb{E}[e^{\lambda X}] = \mathbb{E} \left[\frac{d}{d\lambda} e^{\lambda X} \right] = \mathbb{E}[X e^{\lambda X}] \leq \mathbb{E}[X e^\lambda] = \mu e^\lambda = g'(\lambda),$$

using in the inequality that $\lambda \geq 0$ and $0 \leq X \leq 1$. and that $0 \leq X \leq 1$. The same steps give $f'(\lambda) \geq g'(\lambda)$ for $\lambda \leq 0$. As $f(0) = 1 = g(0)$, it follows that $f(\lambda) \leq g(\lambda)$ everywhere. The conclusion follows by Lemma 3.10. \square

You can interchange this derivative and expectation, but it's trickier to prove than usual, requiring e.g. Theorem 3 here.

4 PAC learning; infinite \mathcal{H}

Recall that we previously showed Proposition 2.2:

PROPOSITION 2.2. *Suppose $\ell(z, h)$ is almost surely bounded in $[a, b]$, \mathcal{H} is finite, and \hat{h}_S is any ERM in \mathcal{H} . Then for any $\delta > 0$, with probability at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ it holds that*

$$L_{\mathcal{D}}(\hat{h}_S) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq (b - a) \sqrt{\frac{2}{m} \log \frac{|\mathcal{H}| + 1}{\delta}}.$$

Another way to state this result is that with m samples, we can achieve estimation error at most ε with probability at least $1 - (|\mathcal{H}| + 1) \exp\left(-\frac{m\varepsilon^2}{2(b-a)^2}\right)$.

Or, alternately, we can say that we can achieve estimation error at most ε with probability at least $1 - \delta$ if we have at least $\frac{2(b-a)^2}{\varepsilon^2} \log \frac{|\mathcal{H}|+1}{\delta}$ samples. This last way establishes the *sample complexity* of learning to a given estimation error ε with a given confidence $1 - \delta$.

4.1 PAC LEARNING

This last statement corresponds to one of the standard notions of learnability. Here, we're going to use a general idea of a learning algorithm as some function that takes a sample $S \in \mathcal{Z}^*$ (the set of sequences of any length from \mathcal{Z}) and returns a hypothesis in \mathcal{H} .

DEFINITION 4.1. An algorithm $\mathcal{A} : \mathcal{Z}^* \rightarrow \mathcal{H}$ *agnostically PAC learns* \mathcal{H} with a loss ℓ if there exists a function $m : (0, 1)^2 \rightarrow \mathbb{N}$ such that, for every $\varepsilon, \delta \in (0, 1)$, for every distribution \mathcal{D} over \mathcal{Z} , for any $m \geq m(\varepsilon, \delta)$, we have that

$$\Pr_{S \sim \mathcal{D}^m, \mathcal{A}} \left(L_{\mathcal{D}}(\mathcal{A}(S)) > \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon \right) < \delta,$$

where the randomness is both over the choice of S and any internal randomness in the algorithm \mathcal{A} . That is, \mathcal{A} can *probably* get an *approximately correct* answer, where "correct" means the best possible loss in \mathcal{H} .

If \mathcal{A} runs in time polynomial in $1/\varepsilon$, $1/\delta$, m , and some notion of the size of h^* , then we say that \mathcal{A} *efficiently agnostically PAC learns* \mathcal{H} .

DEFINITION 4.2. A hypothesis class \mathcal{H} is *agnostically PAC learnable* if there exists an algorithm \mathcal{A} which agnostically PAC learns \mathcal{H} .

So, ERM agnostically PAC-learns finite hypothesis classes, with the sample complexity $m(\varepsilon, \delta) = \frac{2(b-a)^2}{\varepsilon^2} \log \frac{|\mathcal{H}|+1}{\delta}$. Notice that in the definition of agnostic PAC learning, there's no limitation on the distribution – there needs to be an $m(\varepsilon, \delta)$ that works for

any \mathcal{D} . Proposition 2.2 satisfies this, but in general, it's an extremely worst-case kind of notion.

Often it's nicer to think about cases where we can make some assumptions on \mathcal{D} . For example, maybe the number of samples you need depends on "how hard" the particular problem is. We'll talk about this more a little later in the course. For now, it's worth mentioning one common special case:

A1 Q4 was partly about this setting. **DEFINITION 4.3.** Consider a nonnegative loss $\ell(h, z) \geq 0$. A distribution \mathcal{D} is called *realizable* by \mathcal{H} if there exists an $h^* \in \mathcal{H}$ such that $L_{\mathcal{D}}(h^*) = 0$.

This version is the "privileged" version that doesn't need a modifier because it's was introduced first [Val84]. **DEFINITION 4.4.** An algorithm $\mathcal{A} : \mathcal{Z}^m \rightarrow \mathcal{H}$ PAC learns \mathcal{H} with a loss ℓ if there exists a function $m : (0, 1)^2 \rightarrow \mathbb{N}$ such that, for every $\epsilon, \delta \in (0, 1)$, for every *realizable* distribution \mathcal{D} over \mathcal{Z} , for any $m \geq m(\epsilon, \delta)$, we have that

$$\Pr_{S \sim \mathcal{D}^m, \mathcal{A}} (L_{\mathcal{D}}(\mathcal{A}(S)) > \epsilon) < \delta,$$

where the randomness is both over the choice of S and any internal randomness in the algorithm \mathcal{A} . That is, \mathcal{A} can *probably* get an *approximately correct* answer, where "correct" means zero loss.

If \mathcal{A} runs in time polynomial in $1/\epsilon, 1/\delta, m$, and some notion of the size of h^* , then we say that \mathcal{A} *efficiently (realizably) PAC learns* \mathcal{H} .

DEFINITION 4.5. A hypothesis class \mathcal{H} is *PAC learnable* if there exists an algorithm \mathcal{A} which PAC learns \mathcal{H} .

Sometimes people say "realizable PAC learnable" or similar, to emphasize the difference versus agnostic PAC. The name "agnostic" is because the definition doesn't care whether there's a perfect h^* or not. (Notice that if \mathcal{A} agnostically PAC learns \mathcal{H} , then it also PAC learns \mathcal{H} .)

The emphasis here on "how many samples for a given error" is also kind of a TCS-style framing, whereas statisticians more often ask "how much error for a given number of samples"; I tend to prefer the latter, but it's all equivalent. If you read [SSBD14] or other work by computational learning theorists, there tends to be a lot of focus on just being learnable versus not being learnable. That problem has been solved, though, as we'll see not too much later in class; recent work focuses much more on rates than on just learnability or not, and tends to be willing to make *some* assumptions on \mathcal{D} rather than either being totally general or assuming only realizability.

We've shown that anything finite is agnostically PAC learnable. That's only an upper bound, though; it *doesn't* mean that infinite things aren't learnable. Which is good, because that's what we usually want to learn!

Lemma 6.1 of [SSBD14] gives a really simple example of realizably PAC learning an infinite class, if you're curious to see that style of proof. I tried to do an agnostic version of that, but it was more complicated than I hoped, so let's do something more interesting instead.

4.2 COVERING NUMBER BOUNDS

This is more convenient than $\mathcal{Y} = \{0, 1\}$ here... In *logistic regression*, our data is in a subset of \mathbb{R}^d , our labels are in $\mathcal{Y} = \{-1, 1\}$ and we try to predict with a confidence score in $\widehat{\mathcal{Y}} = \mathbb{R}$. Our predictors are linear functions of the form $h_w(x) = w \cdot x$, and the logistic loss is given by

You usually want an intercept term, $w \cdot x + w_0$, but you can achieve that by padding x with an always-one dimension.

$$\ell_{\log}(h, (x, y)) = l_y^{\log}(h(x)) = \log(1 + \exp(-h(x)y)). \tag{4.1}$$

We'll use the hypothesis class $\mathcal{H} = \{h_w = x \mapsto w \cdot x : w \in \mathbb{R}^d, \|w\| \leq B\}$ for some constant B ; this avoids overfitting by using really-really complex w , and is basically equivalent to doing L_2 -regularized logistic regression (we'll talk about this more later). This \mathcal{H} is still infinite, but it has finite volume.

Now, our analysis is going to be based on the idea that if w and v are similar predictors, i.e. $h_w(x) \approx h_v(x)$ for all x , then they'll behave similarly: $L_{\mathcal{D}}(h_w) \approx L_{\mathcal{D}}(h_v)$ and $L_S(h_w) \approx L_S(h_v)$. Thus we don't have to do a totally separate concentration bound on their empirical risks; we can exploit that they're similar.

The fundamental idea is going to be one of a "set cover," or an " ε -net." To handle an infinite \mathcal{H} that's nonetheless bounded, we're going to choose some *finite* set \mathcal{H}_0 such that everything in \mathcal{H} is close to something in \mathcal{H}_0 , use Proposition 2.2 to say that $L_{\mathcal{D}}(h) - L_S(h)$ isn't too big for anything in \mathcal{H}_0 , and then argue that since $L_{\mathcal{D}}(h) - L_S(h)$ is smooth, this means it can't be too big for anything in \mathcal{H} at all.

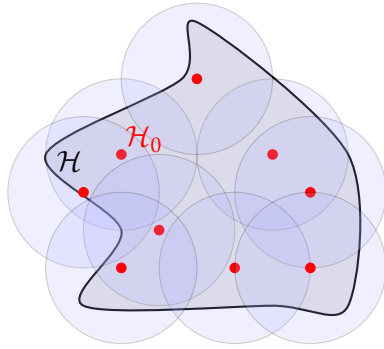


Figure 4.1: A (non-minimal) set cover.

4.2.1 Smoothness: Lipschitz functions

To formalize the idea that similar weight vectors give similar loss, we'll want a bound like

$$|L_{\mathcal{D}}(h) - L_{\mathcal{D}}(g)| \leq M \rho_{\mathcal{H}}(h, g),$$

for some notion of a distance metric on \mathcal{H} . This is called a Lipschitz property.

DEFINITION 4.6. A function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is M -Lipschitz with respect to $\rho_{\mathcal{X}}$ and $\rho_{\mathcal{Y}}$ if for all $x, x' \in \mathcal{X}$, $\rho_{\mathcal{Y}}(f(x), f(x')) \leq M \rho_{\mathcal{X}}(x, x')$. The smallest M for which this inequality holds is the Lipschitz constant, denoted $\|f\|_{\text{Lip}}$.

If \mathcal{X} and/or \mathcal{Y} are subsets of \mathbb{R}^d , ρ is Euclidean distance unless otherwise specified.

So, for example, $x \mapsto |x|$ is a 1-Lipschitz function, since $||x| - |y|| \leq |x - y|$.

The notation $\|f\|_{\text{Lip}}$ is justified by the following result. If you're not sure about function spaces / norms / etc, don't worry about it (we'll come back to this later in the course); the takeaway is the two properties shown in the proof.

LEMMA 4.7. Consider a vector space of functions $\mathcal{X} \rightarrow \mathcal{Y}$, where \mathcal{Y} is a normed space, such that $f + g$ is the function $x \mapsto f(x) + g(x)$ and af is the function $x \mapsto af(x)$. $\|\cdot\|_{\text{Lip}}$ is a seminorm on this space with respect to $\|\cdot\|_{\mathcal{Y}}$.

Proof. There are two properties to show. First, subadditivity (which implies the

triangle inequality):

$$\begin{aligned} \|f + g\|_{\text{Lip}} &= \sup_{x \neq x'} \frac{\|f(x) + g(x) - f(x') - g(x')\|}{\rho_{\mathcal{X}}(x, x')} \\ &\leq \sup_{x \neq x'} \frac{\|f(x) - f(x')\|}{\rho_{\mathcal{X}}(x, x')} + \frac{\|g(x) - g(x')\|}{\rho_{\mathcal{X}}(x, x')} \leq \|f\|_{\text{Lip}} + \|g\|_{\text{Lip}}. \end{aligned}$$

Second, absolute homogeneity:

$$\|af\|_{\text{Lip}} = \sup_{x \neq x'} \frac{\|af(x) - af(x')\|}{\rho_{\mathcal{X}}(x, x')} = \sup_{x \neq x'} \frac{|a| \|f(x) - f(x')\|}{\rho_{\mathcal{X}}(x, x')} = |a| \|f\|_{\text{Lip}}. \quad \square$$

It isn't a proper norm because $\|x \mapsto a\|_{\text{Lip}} = 0$ for all constant functions.

So, what is $\|L_{\mathcal{D}}\|_{\text{Lip}}$? When $z = (x, y)$ and $\ell(h, (x, y)) = l_y(h(x))$, we have

$$\begin{aligned} |L_{\mathcal{D}}(h) - L_{\mathcal{D}}(g)| &= \left| \mathbb{E}_{z \sim \mathcal{D}} \ell(h, z) - \mathbb{E}_{z \sim \mathcal{D}} \ell(g, z) \right| \\ &\leq \mathbb{E}_{z \sim \mathcal{D}} |\ell(h, z) - \ell(g, z)| \\ &= \mathbb{E}_{(x, y) \sim \mathcal{D}} |l_y(h(x)) - l_y(g(x))| \\ &\leq \mathbb{E}_{(x, y) \sim \mathcal{D}} \|l_y\|_{\text{Lip}} \rho_{\hat{y}}(h(x), g(x)). \end{aligned} \quad (4.2)$$

So, in particular settings we want to find $\|l_y\|_{\text{Lip}}$ and bound $\rho_{\hat{y}}(h(x), g(x))$ in terms of some notion of similarity between h and g .

For the first problem, since for logistic regression $l_y^{\text{log}} : \mathbb{R} \rightarrow \mathbb{R}$, this result will help:

LEMMA 4.8. *Let $\mathcal{X} \subseteq \mathbb{R}$ be a connected, closed set. If a function $f : \mathcal{X} \rightarrow \mathbb{R}$ is continuous and differentiable everywhere on the interior of \mathcal{X} , $\|f\|_{\text{Lip}} = \sup_{x \in \mathcal{X}} |f'(x)|$.*

Proof. We apply the fundamental theorem of calculus:

$$|f(x') - f(x)| = \left| \int_x^{x'} f'(x) dx \right| \leq \int_x^{x'} |f'(x)| dx \leq \int_x^{x'} \|f\|_{\text{Lip}} dx = \|f\|_{\text{Lip}} |x' - x|. \quad \square$$

We won't need this today, but it's worth noting that if $\mathcal{X} \subseteq \mathbb{R}^d$, the same proof idea gives us that $\|f\|_{\text{Lip}} = \sup_{x \in \mathcal{X}} \|\nabla f(x)\|$.

LEMMA 4.9. *For any $y \in \{-1, 1\}$, $\|l_y^{\text{log}}\|_{\text{Lip}} \leq 1$.*

Proof. l_y^{log} is differentiable everywhere on \mathbb{R} , and so using Lemma 4.8,

$$\begin{aligned} \left| \frac{d}{d\hat{y}} l_y^{\text{log}}(\hat{y}) \right| &= \left| \frac{d}{d\hat{y}} \log(1 + \exp(-y\hat{y})) \right| = \left| \frac{1}{1 + \exp(-y\hat{y})} \exp(-y\hat{y})(-y) \right| \\ &= \left| \frac{\exp(-y\hat{y})}{1 + \exp(-y\hat{y})} \times \frac{\exp(y\hat{y})}{\exp(y\hat{y})} \right| |-y| = \left| \frac{1}{1 + \exp(y\hat{y})} \right| \leq 1. \quad \square \end{aligned}$$

Plugging into (4.2), we get

$$|\mathbb{L}_{\mathcal{D}}(h_w) - \mathbb{L}_{\mathcal{D}}(h_v)| \leq \mathbb{E}_{(x,y) \sim \mathcal{D}} \|l_y\|_{\text{Lip}} |h_w(x) - h_v(x)|.$$

That is, if the predictions are similar, the losses are too. We can further say that if w and v are close, then their predictions are similar:

$$|h_w(x) - h_v(x)| = |w \cdot x - v \cdot x| = |(w - v) \cdot x| \leq \|w - v\| \|x\|$$

by Cauchy-Schwarz. Thus

$$|\mathbb{L}_{\mathcal{D}}(h_w) - \mathbb{L}_{\mathcal{D}}(h_v)| \leq \left(\mathbb{E}_{(x,y) \sim \mathcal{D}} \|x\| \|l_y\|_{\text{Lip}} \right) \|w - v\|,$$

giving that $\mathbb{L}_{\mathcal{D}}$ is $(\mathbb{E}_{(x,y) \sim \mathcal{D}} \|x\| \|l_y\|_{\text{Lip}})$ -Lipschitz with respect to $\rho_{\mathcal{H}}(h_w, h_v) = \|w - v\|$, and similarly \mathbb{L}_S is $(\frac{1}{m} \sum_{i=1}^m \|x_i\| \|l_{y_i}\|_{\text{Lip}})$ -Lipschitz. (We could repeat the argument with empirical averages instead of \mathbb{E} , but a slicker way is to note that \mathbb{L}_S is exactly $\mathbb{L}_{\hat{\mathcal{D}}_S}$ for the *empirical distribution* $\hat{\mathcal{D}}_S$, the discrete distribution that puts $1/m$ probability at each member of S .) Thus we know that

$$\|\mathbb{L}_{\mathcal{D}} - \mathbb{L}_S\|_{\text{Lip}} \leq \mathbb{E}_{(x,y) \sim \mathcal{D}} \|x\| \|l_y\|_{\text{Lip}} + \frac{1}{m} \sum_{i=1}^m \|x_i\| \|l_{y_i}\|_{\text{Lip}}. \quad (4.3)$$

If we assume for simplicity that the distribution is bounded, $\Pr_{(x,y) \sim \mathcal{D}}(\|x\| \leq C) = 1$, and that $\|l_y\|_{\text{Lip}} \leq M$ for each y (as with logistic loss, where $M = 1$), then $\mathbb{L}_{\mathcal{D}} - \mathbb{L}_S$ is guaranteed to be $(2CM)$ -Lipschitz.

4.2.2 Putting it together with a set covering

Now the question is: how big does \mathcal{H}_0 have to be? We'll use the following concept:

DEFINITION 4.10. An η -cover of a set U is a set $T \subseteq U$ such that, for all $u \in U$, there is a $t \in T$ with $\rho(t, u) \leq \eta$. The *covering number* $N(U, \eta)$ is the size of the smallest η -cover for U .

We want to cover $\mathcal{H}_B = \{h_w = (x \mapsto w \cdot x) : \|w\| \leq B\}$ with the metric $\rho(h_w, h_v) = \|w - v\|$. We can immediately construct this kind of cover if we have a cover for the Euclidean ball of radius B . Section 4.2.3 bounds how big this cover needs to be:

LEMMA 4.11. Let $\eta \in (0, B]$ and $p \in [1, \infty]$. The covering number of the radius- B p -norm ball in \mathbb{R}^d , $U = \{x \in \mathbb{R}^d : \|x\|_p \leq B\}$, satisfies

$$\left(\frac{B}{\eta}\right)^d \leq N(U, \eta) \leq \left(\frac{2B}{\eta} + 1\right)^d \leq \left(\frac{3B}{\eta}\right)^d.$$

(When $\eta \geq B$, trivially $N(U, \eta) = 1$.)

We now have all the tools we need for the following result about linear models with bounded Lipschitz losses.

PROPOSITION 4.12. Let $h_w(x) = w \cdot x$ and $\mathcal{H} = \{h_w : \|w\| \leq B\}$ for some $B > 0$. Consider a loss $\ell(h, (x, y)) = l_y(h(x))$ for functions $l_y : \mathbb{R} \rightarrow \mathbb{R}$ which each have Lipschitz constant at most M and are bounded in $[a, b]$. Assume that $\|x\| \leq C$ almost surely under \mathcal{D} . Then,

with probability at least $1 - \delta$,

$$\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq \frac{1}{\sqrt{2m}} \left[\text{BCM} + (b - a) \sqrt{\log \frac{1}{\delta} + \frac{d}{2} \log(72m)} \right].$$

Proof. We'll first choose a η -cover $\mathcal{H}_0 = \{w_1, \dots, w_{N_\eta}\} \subset \{w \in \mathbb{R}^d : \|w\| \leq B\}$, where η is a parameter to be set later. Then, for any $h \in \mathcal{H}$, let $\text{nn}_{\mathcal{H}_0}(h) \in \arg \min_{h' \in \mathcal{H}_0} \rho(h, h')$, using $\rho(h_w, h_v) = \|w - v\|$. Define the function $\Delta(h) := L_{\mathcal{D}}(h) - L_S(h)$ for brevity. Then

$$\begin{aligned} \sup_{h \in \mathcal{H}} \Delta(h) &= \sup_{h \in \mathcal{H}} \Delta(h) - \Delta(\text{nn}(h)) + \Delta(\text{nn}(h)) \\ &\leq \sup_{h \in \mathcal{H}} [\Delta(h) - \Delta(\text{nn}(h))] + \sup_{h' \in \mathcal{H}_0} \Delta(h') \\ &\leq 2\text{CM}\eta + \sup_{h' \in \mathcal{H}_0} \Delta(h'), \end{aligned}$$

where the first term is because of (4.3) and \mathcal{H}_0 being an η -cover.

The other term is uniform convergence over a finite hypothesis class \mathcal{H}_0 , as in Proposition 2.2. We can apply Hoeffding to each element of \mathcal{H}_0 , giving it a failure probability of δ/N_η , and obtain that with probability at least $1 - \delta$,

$$\begin{aligned} \sup_{h \in \mathcal{H}} \Delta(h) &\leq 2\text{CM}\eta + (b - a) \sqrt{\frac{1}{2m} \log \frac{N_\eta}{\delta}} \\ &\leq 2\text{CM}\eta + (b - a) \sqrt{\frac{1}{2m} \left[\log \frac{1}{\delta} + d \log \frac{3B}{\eta} \right]}. \end{aligned}$$

Now, we could try to exactly optimize the value of η , but I think we won't be able to do that analytically. Instead, let's notice that if η is $o(1/\sqrt{m})$, the first term being smaller doesn't really help in rate since the other term is $1/\sqrt{m}$ anyway – but choosing a smaller η makes the $\log \frac{1}{\eta}$ worse. Also, the dependence on η there is only in a log term, so it's probably okay-ish to choose $\eta = \alpha/\sqrt{m}$ for some $\alpha > 0$, giving us

$$\sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] \leq \frac{1}{\sqrt{m}} \left[2\text{CM}\alpha + \frac{b - a}{\sqrt{2}} \sqrt{\log \frac{1}{\delta} + d \log \frac{3B\sqrt{m}}{\alpha}} \right].$$

Picking $\alpha = B/(2\sqrt{2})$ and using $\log A = \frac{1}{2} \log(A^2)$ gives the desired result. \square

For our motivating problem of logistic regression, $M = 1$, but there's one catch: we can use $a = 0$ but there isn't an "inherent" upper bound for b . Given that we know

$\|x\| \leq C$ and $\|w\| \leq B$, though, we have that $|h(x)| = |w \cdot x| \leq BC$. Thus

$$\begin{aligned} \ell(h, (x, y)) &= \log(1 + \exp(-yh(x))) \leq \log(1 + \exp(BC)) =: b \\ \ell(h, (x, y)) &= \log(1 + \exp(-yh(x))) \geq \log(1 + \exp(-BC)) =: a \\ b - a &= \log(1 + \exp(BC)) - \log(1 + \exp(-BC)) \\ &= \log\left(\frac{1 + \exp(BC)}{1 + \exp(-BC)} \times \frac{\exp(BC)}{\exp(BC)}\right) \\ &= \log\left(\frac{1 + \exp(BC)}{\exp(BC) + 1} \times \exp(BC)\right) = \log \exp(BC) = BC. \end{aligned} \quad (4.4)$$

Plugging into Proposition 4.12 gives us that with probability at least $1 - \delta$, logistic regression with bounded-norm weights on bounded-norm data satisfies

$$\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_{\mathcal{S}}(h) \leq \frac{BC}{\sqrt{2m}} \left[1 + \sqrt{\log \frac{1}{\delta} + \frac{d}{2} \log(72m)} \right] = \mathcal{O}_p \left(BC \sqrt{\frac{d \log m}{m}} \right). \quad (4.5)$$

Treating everything but m as a constant, the rate is $\mathcal{O}_p \left(\sqrt{\frac{\log m}{m}} \right)$. That $\sqrt{\log m}$ factor is actually unnecessary, but getting rid of it with covering number-type arguments requires some more advanced machinery. Instead, soon we'll see a simpler way to show a $\mathcal{O}_p(1/\sqrt{m})$ rate – in fact, a $\mathcal{O}_p(BC/\sqrt{m})$ rate, also dramatically improving the dependence on d – that will also be very generally applicable.

This machinery is called “chaining”; we probably won't cover it in class, but Wainwright [Wai19, Section 5.3.3] has a reasonable overview.

ERM BOUND We only wrote this proof here for $\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_{\mathcal{S}}(h)$, but since the loss is bounded, this implies exactly as in (1.5) an upper bound on the generalization error of any ERM $\hat{h}_{\mathcal{S}}$. Using the general result from Proposition 4.12 with probability $\delta/2$, and plain Hoeffding with probability $\delta/2$ on the $L_{\mathcal{S}}(h^*) - L_{\mathcal{D}}(h^*)$ term, gives us

$$L_{\mathcal{D}}(\hat{h}_{\mathcal{S}}) - L_{\mathcal{D}}(h^*) \leq \frac{1}{\sqrt{2m}} \left[BCM + (b - a) \sqrt{\log \frac{2}{\delta} + \frac{d}{2} \log(72m)} \right] + (b - a) \sqrt{\frac{1}{2m} \log \frac{2}{\delta}},$$

and using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ we can simplify to

$$L_{\mathcal{D}}(\hat{h}_{\mathcal{S}}) - L_{\mathcal{D}}(h^*) \leq \frac{1}{\sqrt{2m}} \left[BCM + (b - a) \sqrt{\frac{d}{2} \log(72m)} + 2(b - a) \sqrt{\log \frac{2}{\delta}} \right].$$

Specializing to logistic regression, we can plug in $M = 1$, $b - a = BC$ so that

$$L_{\mathcal{D}}(\hat{h}_{\mathcal{S}}) - L_{\mathcal{D}}(h^*) \leq \frac{BC}{\sqrt{m}} \left[\frac{1}{\sqrt{2}} + \frac{1}{2} \sqrt{d \log(72m)} + \sqrt{2 \log \frac{2}{\delta}} \right] = \mathcal{O}_p \left(BC \sqrt{\frac{d \log m}{m}} \right). \quad (4.6)$$

A question for yourself here: does this imply that ERM agnostically PAC-learns logistic regression?

MORE GENERAL VERSIONS We used the following properties about the problem:

- A bounded loss, to apply Hoeffding. This could be weakened in various ways, e.g. another kind of subgaussianity, or other ways to show concentration for a finite number of points.
- A Lipschitz loss. Some form of this is definitely necessary. You could poten-

tially use a locally Lipschitz loss (where the constant varies through space), but then you have to be more careful in bounding (4.3) or similar.

- A parameterization for \mathcal{H} with a covering number bound. We framed this as covering the parameter set for linear models, but you could use more general notions of covering for \mathcal{H} , as long as they're compatible with the metric you use for Lipschitzness in the previous part. This generality is often useful, e.g. for nonparametric \mathcal{H} .

4.2.3 *Aside: Bounds on covering numbers*

We'll now prove our upper bound on covering numbers. Recall their definition:

DEFINITION 4.10. An η -cover of a set U is a set $T \subseteq U$ such that, for all $u \in U$, there is a $t \in T$ with $\rho(t, u) \leq \eta$. The *covering number* $N(U, \eta)$ is the size of the smallest η -cover for U .

We'll also use *packing numbers*: how many balls can we squeeze into a set T ?

DEFINITION 4.13. An η -packing of a set U is a set $T \subseteq U$ such that, for all $t, t' \in T$ with $t \neq t'$, we have $\rho(t, t') > \eta$. The *packing number* $M(U, \eta)$ is the maximal size of any η -packing.

PROPOSITION 4.14. A maximally-sized η -packing T of a set U is also a η -cover of U .

Proof. Suppose there were some point $u \in U$ such that $\rho(u, t) > \eta$ for all $t \in T$. Then we could add u to the η -packing, producing a packing of size one larger; this contradicts that T was maximal. \square

We're now ready to prove the result:

LEMMA 4.11. Let $\eta \in (0, B]$ and $p \in [1, \infty]$. The covering number of the radius- B p -norm ball in \mathbb{R}^d , $U = \{x \in \mathbb{R}^d : \|x\|_p \leq B\}$, satisfies

$$\left(\frac{B}{\eta}\right)^d \leq N(U, \eta) \leq \left(\frac{2B}{\eta} + 1\right)^d \leq \left(\frac{3B}{\eta}\right)^d.$$

(When $\eta \geq B$, trivially $N(U, \eta) = 1$.)

Proof. By Proposition 4.14, we have that $N(U, \eta) \leq M(U, \eta)$; we'll first prove the upper bound on the packing number M . Let T be a maximal η -packing of the B -ball $U = \{w \in \mathbb{R}^d : \|w\|_p \leq B\}$. Thus the open $\eta/2$ -balls centered at each $t \in T$, $\{w \in \mathbb{R}^d : \|w - t\|_p < \eta/2\}$, are disjoint: if they weren't, you could get from one t to another in distance less than η , contradicting that T is an η -packing. These balls are also all contained within the ball of radius $(B + \eta/2)$, since each $\|t\|_p \leq B$. Thus

$$\sum_{t \in T} \text{vol}(\{w \in \mathbb{R}^d : \|w - t\|_p < \eta/2\}) \leq \text{vol}(\{w \in \mathbb{R}^d : \|w\|_p < B + \eta/2\}).$$

But we know that the volume of a p -norm ball of radius R in d dimensions is $R^d V_1$,

where $V_1 = \text{vol}(\{w \in \mathbb{R}^d : \|w\|_p < 1\})$. Thus

$$\sum_{t \in T} \left(\frac{\eta}{2}\right)^d V_1 = M(U, \eta) \left(\frac{\eta}{2}\right)^d V_1 \leq \left(B + \frac{\eta}{2}\right)^d V_1$$

$$\text{so } M(U, \eta) \leq \left(\frac{2B}{\eta} + 1\right)^d = \left(\frac{2B + \eta}{\eta}\right)^d \leq \left(\frac{3B}{\eta}\right)^d,$$

using at the end that $\eta \leq B$ to get a simpler form.

For the lower bound, it holds for a minimal cover T of any set U that

$$\text{vol}(U) \leq \text{vol}\left(\bigcup_{t \in T} \{w : \|w - t\|_p < \eta\}\right) \leq \sum_{t \in T} \text{vol}(\{w : \|w - t\|_p < \eta\}) = N(U, \eta) V_\eta,$$

where $V_\eta = \text{vol}(\{w : \|w\|_p < \eta\})$. Thus $N(U, \eta) \geq \text{vol}(U)/V_\eta$. Plugging in for U being a $\|\cdot\|_p$ ball in \mathbb{R}^d , we obtain the desired lower bound. \square

A similar upper bound holds more generally for any finite-dimensional **Banach space**, getting $(4B/\eta)^d$ [CS02, Proposition 5]. I don't know about a lower bound there. For infinite-dimensional Banach spaces, the lower bound is infinite [Isr15], so to use covering numbers another setup is necessary.

I don't know if the above proofs can be generalized or not.

5 Rademacher complexity

Last time (Section 4.2) was our first time showing a uniform convergence bound, one on $\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h)$, for an infinite \mathcal{H} . We can then easily turn that into a bound on the estimation error of ERM, $L_{\mathcal{D}}(\hat{h}_S) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$, as in (4.6).

We're now going to develop a technique that's less intuitive, but will show a better result (no $\sqrt{d \log m}$), is somewhat more general, and once you understand it can be easier to use.

We'll start with a bound on the *mean* worst-case generalization gap. That is, we'll show that

$$\mathbb{E}_{S \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq \varepsilon(m).$$

This gives us, for instance, that if \hat{h}_S is an ERM then

$$\mathbb{E} L_{\mathcal{D}}(\hat{h}_S) = \underbrace{\mathbb{E} [L_{\mathcal{D}}(\hat{h}_S) - L_S(\hat{h}_S)]}_{\leq \varepsilon(m)} + \underbrace{\mathbb{E} [L_S(\hat{h}_S) - L_S(h^*)]}_{\leq 0} + \underbrace{\mathbb{E} [L_S(h^*)]}_{= L_{\mathcal{D}}(h^*)} \leq L_{\mathcal{D}}(h^*) + \varepsilon(m).$$

We'll use this to prove a high-probability bound on $\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h)$ in Section 5.3.

5.1 A G-G-G-G-GHOST (SAMPLE)

Using that $L_{\mathcal{D}}(h) = \mathbb{E}_{S \sim \mathcal{D}^m} L_S(h)$:

$$\mathbb{E}_{S \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) = \mathbb{E}_{S \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} \mathbb{E}_{S' \sim \mathcal{D}^m} L_{S'}(h) - L_S(h).$$

S' here is sometimes called a "ghost sample."

Now, we'll exploit the following general fact:

LEMMA 5.1. *Let f_y be a class of functions indexed by y , and X be some random variable. Then when the expectations exist,*

$$\sup_y \mathbb{E}_X f_y(X) \leq \mathbb{E}_X \sup_y f_y(X).$$

This should be intuitive, once you think about it a bit: if the optimization can see what particular sample you got, it can "overfit" better than if it has to optimize on average.

Proof. For any y , we have $f_y(X) \leq \sup_{y'} f_{y'}(X)$ by definition, no matter the value of X . Taking the expectation of both sides, for any y , $\mathbb{E}_X f_y(X) \leq \mathbb{E}_X \sup_{y'} f_{y'}(X)$. So it's also true if we take the supremum over y . \square

Applying this, we see that

$$\mathbb{E}_{S \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq \mathbb{E}_{S \sim \mathcal{D}^m} \sup_{S' \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{S'}(h) - L_S(h). \quad (5.1)$$

The right-hand-side of (5.1) is itself a natural thing to think about: how much does anything in \mathcal{H} overfit relative to a test set?

Now, $S = (z_1, \dots, z_m)$ and $S' = (z'_1, \dots, z'_m)$ are composed of independent samples from the same distribution. So, if we decided to swap z_3 and z'_3 , this would still be a “valid,” equally likely sample for S and S' . Rademacher complexity is based on this idea.

Watch out that σ_i has nothing to do with a standard deviation or subgaussian parameter σ ; we'll refer to the vector $(\sigma_1, \dots, \sigma_m)$ as σ , or $\vec{\sigma}$ in handwriting. Unfortunate, but no option is great here.

Notationally, let $\sigma_i \in \{-1, 1\}$ for $i \in [m]$, and define $(u_i, u'_i) = \begin{cases} (z_i, z'_i) & \text{if } \sigma_i = 1 \\ (z'_i, z_i) & \text{if } \sigma_i = -1. \end{cases}$

Then, for any choice of $\sigma = (\sigma_1, \dots, \sigma_m)$, we have

$$\ell(h, z'_i) - \ell(h, z_i) = \sigma_i(\ell(h, u'_i) - \ell(h, u_i)).$$

So, for any value of S , S' , and σ , defining $U = (u_1, \dots, u_m)$ and $U' = (u'_1, \dots, u'_m)$ accordingly, we have

$$L_{S'}(h) - L_S(h) = \frac{1}{m} \sum_i \sigma_i [\ell(h, u'_i) - \ell(h, u_i)].$$

Since this holds for *any* choice of σ , it also holds if we pick them at random and then take a mean over that choice. We'll choose them according to a Rademacher distribution, also written $\text{Unif}(\pm 1)$, which is 1 half the time and -1 the other half. Thus,

$$\mathbb{E}_{S, S' \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{S'}(h) - L_S(h) = \mathbb{E}_{\sigma} \mathbb{E}_{S, S' \sim \mathcal{D}^m} \mathbb{E}_{U, U'} \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_i \sigma_i [\ell(h, u'_i) - \ell(h, u_i)] \mid S, S', \sigma \right].$$

Here we're writing U and U' as random variables, even though they're actually deterministic conditional on S , S' , and σ . The marginal distributions of U and U' are each exactly \mathcal{D}^m , though, the same as S and S' . So, it makes sense for us to switch the order of the expectations. $\sigma \mid U, U'$ is still just random signs; given σ and U, U' , S and S' become deterministic. This gives us

This switch is allowed by Fubini's theorem as long as $\mathbb{E} |\sup_h L_{S'}(h) - L_S(h)|$ is finite, which is always true e.g. for a bounded loss.

$$\mathbb{E}_{S, S' \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{S'}(h) - L_S(h) = \mathbb{E}_{U, U' \sim \mathcal{D}^m} \mathbb{E}_{\sigma} \mathbb{E}_{S, S'} \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_i \sigma_i [\ell(h, u'_i) - \ell(h, u_i)] \mid U, U', \sigma \right].$$

But... S and S' no longer appear at all, so we can forget about that expectation on the right. Continuing,

This proof technique of introducing a random sign is called symmetrization.

$$\mathbb{E}_{S, S' \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{S'}(h) - L_S(h) = \mathbb{E}_{U, U' \sim \mathcal{D}^m} \mathbb{E}_{\sigma} \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_i \sigma_i [\ell(h, u'_i) - \ell(h, u_i)]$$

$$\leq \mathbb{E}_{U, U' \sim \mathcal{D}^m} \mathbb{E}_{\sigma} \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_i \sigma_i \ell(h, u'_i) + \sup_{h' \in \mathcal{H}} \frac{1}{m} \sum_i (-\sigma_i) \ell(h', u_i) \right]$$

$$= \mathbb{E}_{U, U' \sim \mathcal{D}^m} \mathbb{E}_{\sigma} \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_i \sigma_i \ell(h, u'_i) + \mathbb{E}_{U, U' \sim \mathcal{D}^m} \mathbb{E}_{\sigma} \sup_{h' \in \mathcal{H}} \frac{1}{m} \sum_i \sigma_i \ell(h', u_i)$$

$-\sigma$ and σ have the same distribution

Renaming U to S

$$= 2 \mathbb{E}_{S, S' \sim \mathcal{D}^m} \mathbb{E}_{\sigma} \sup_{h \in \mathcal{H}} \frac{1}{m} \sum_i \sigma_i \ell(h, z_i)$$

$$=: 2 \mathbb{E}_{S, S' \sim \mathcal{D}^m} \text{Rad}((\ell \circ \mathcal{H})|_S).$$

We're defining some notation at the end: $\ell \circ \mathcal{H} = \{z \mapsto \ell(h, z) : h \in \mathcal{H}\}$ is a set of

functions from \mathcal{Z} to \mathbb{R} , and $\mathcal{F}|_S$ denotes $\{(f(z_1), \dots, f(z_m)) : f \in \mathcal{F}\} \subseteq \mathbb{R}^m$, so that

$$(\ell \circ \mathcal{H})|_S = \{(\ell(h, z_1), \dots, \ell(h, z_m)) : h \in \mathcal{H}\} \subseteq \mathbb{R}^m.$$

DEFINITION 5.2. The Rademacher complexity of a set $V \subseteq \mathbb{R}^m$ is given by

$$\text{Rad}(V) = \mathbb{E}_{\sigma \sim \text{Unif}(\pm 1)^m} \sup_{v \in V} \frac{1}{m} \sum_{i=1}^m \sigma_i v_i = \mathbb{E}_{\sigma \sim \text{Unif}(\pm 1)^m} \sup_{v \in V} \frac{\sigma \cdot v}{m}.$$

Many sources define Rad with an absolute value around the sum. This is the more common modern definition, since it makes some things nicer.

One way to think of it is a measure of how much a set V extends in the direction of a random binary vector. $\text{Rad}(\mathcal{F}|_S)$ measures how well \mathcal{F} can align with random signs on the particular set S , or equivalently how well it can separate a random subset of S from the rest.

For intuition, it might be nice to compare to the closely-related *Gaussian complexity* [BM02], which uses $\sigma \sim \mathcal{N}(0, I_m)$ instead of a Rademacher vector. That's maybe more natural to see as a notion of the size of a set: "if I look in a random direction, how far do I get?" (Remember that the norm of a random Gaussian concentrates tightly in high dimensions.) For Rademacher, "looking in any direction" versus "looking along 'binary' directions" isn't so different.

Finally, notice that nothing here depended on the structure of the actual functions $z \mapsto \ell(h, z) \in \ell \circ \mathcal{H}$, and so we've proved the following result for general function classes (rather than just those of the form $\ell \circ \mathcal{H}$).

THEOREM 5.3. For any class \mathcal{F} of functions $f : \mathcal{Z} \rightarrow \mathbb{R}$, and any distribution \mathcal{D} over \mathcal{Z} with $S = (z_1, \dots, z_m) \sim \mathcal{D}^m$, we have

$$\mathbb{E}_{S \sim \mathcal{D}^m} \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{z \sim \mathcal{D}} [f(z)] - \frac{1}{n} \sum_{i=1}^m f(z_i) \right) \leq 2 \mathbb{E}_{S \sim \mathcal{D}^m} \text{Rad}(\mathcal{F}|_S).$$

In particular, in our standard learning setup,

$$\mathbb{E}_{S \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq 2 \mathbb{E}_{S \sim \mathcal{D}^m} \text{Rad}((\ell \circ \mathcal{H})|_S).$$

5.2 PROPERTIES OF RADEMACHER COMPLEXITY

First, note that

$$\text{Rad}(\{v\}) = \frac{1}{m} \mathbb{E}_{\sigma} \sigma \cdot v = 0 :$$

no matter the vector, a singleton set has no complexity. (In terms of generalization: any given hypothesis is equally likely to over- or under-estimate the risk.)

On the other extreme, for the vertices of a hypercube,

$$\text{Rad}(\{-1, 1\}^m) = \frac{1}{m} \mathbb{E}_{\sigma} \sup_v \sum_{i=1}^m \sigma_i v_i = \frac{1}{m} \mathbb{E}_{\sigma} m = 1.$$

As we'll see later (Proposition 6.1), this is highly related to considering the complexity of the hypothesis class of all possible $\{-1, 1\}$ -valued functions; if we tried to do ERM in the set of "all possible classifiers," we'd get that the expected zero-one loss is ≤ 1 . Exciting!

Letting $cV = \{cv : v \in V\}$ for any $c \in \mathbb{R}$, we have that

$$\text{Rad}(cV) = \frac{1}{m} \mathbb{E} \sup_{\sigma} \sup_{v \in V} \sigma \cdot (cv) = \frac{1}{m} \mathbb{E} \sup_{\sigma} |c| (\text{sign}(c)\sigma) \cdot v = |c| \text{Rad}(V) \quad (5.2)$$

since $\text{sign}(c)\sigma$ has the same distribution as σ .

For $V + W = \{v + w : v \in V, w \in W\}$, also called the Minkowski sum, we get

$$\text{Rad}(V + W) = \frac{1}{m} \mathbb{E} \sup_{\sigma} \sup_{\substack{v \in V \\ w \in W}} \sigma \cdot (v + w) = \frac{1}{m} \mathbb{E} \sup_{\sigma} \sup_{v \in V} \sigma \cdot v + \frac{1}{m} \mathbb{E} \sup_{\sigma} \sup_{w \in W} \sigma \cdot w = \text{Rad}(V) + \text{Rad}(W).$$

Combined with the fact that $\text{Rad}(\{v\}) = 0$, this means that translating a set by a constant vector doesn't change its complexity.

5.2.1 Talagrand's contraction lemma

How do we compute $\text{Rad}(\ell \circ \mathcal{H}|_S)$ for practical losses and hypothesis classes? The first key step is usually to “peel off” the loss, getting a bound in terms of $\text{Rad}(\mathcal{H}|_{S_x})$. We can do that with the following lemma, which is also *very* helpful for bounding $\text{Rad}(\mathcal{H})$ for \mathcal{H} that are defined compositionally, like deep networks.

The major way to do that is with the following results, for Lipschitz losses (Definition 4.6). For example, recall from Lemma 4.9 that logistic loss, used in logistic regression, is 1-Lipschitz.

A 1-Lipschitz function is called a contraction: it doesn't increase the distance between any points, but (usually) contracts at least some.

LEMMA 5.4 (Talagrand). *Let $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^m$ be given by $\phi(t) = (\phi_1(t_1), \dots, \phi_m(t_m))$, where each ϕ_i is M -Lipschitz. Then*

$$\text{Rad}(\phi \circ V) = \text{Rad}(\{\phi(v) : v \in V\}) \leq M \text{Rad}(V).$$

Our proof will be based on the following special case:

LEMMA 5.5. *If $\varphi : \mathbb{R} \rightarrow \mathbb{R}$ is 1-Lipschitz, $\text{Rad}(\{(\varphi(v_1), v_2, \dots, v_m) : v \in V\}) \leq \text{Rad}(V)$.*

Proof of Lemma 5.4, assuming Lemma 5.5. First notice that “rotating” the vectors in V doesn't change its complexity, since σ has iid entries:

$$\text{Rad}(\{(v_2, \dots, v_m, v_1) : v \in V\}) = \text{Rad}(V).$$

Now, notice that each component of $\frac{1}{M}\phi(t) = (\frac{1}{M}\varphi_1(t_1), \dots, \frac{1}{M}\varphi_m(t_m))$ is 1-Lipschitz. So, start by applying Lemma 5.5 to V with $\frac{1}{M}\varphi_1$, then rotating, to obtain

$$\text{Rad}\left(\left\{(v_2, \dots, v_m, \frac{1}{M}\varphi_1(v_1)) : v \in V\right\}\right) \leq \text{Rad}(V).$$

Repeat these steps with $\frac{1}{M}\varphi_2$, then $\frac{1}{M}\varphi_3$, and so on, until we obtain

$$\text{Rad}\left(\left[\frac{1}{M}\phi\right] \circ V\right) \leq \text{Rad}(V).$$

Finally, scale by M , which by (5.2) means

$$\text{Rad}(\phi \circ V) = M \text{Rad}\left(\left[\frac{1}{M}\phi\right] \circ V\right) \leq M \text{Rad}(V). \quad \square$$

Proof of Lemma 5.5. Let $\phi(v) = (\varphi(v_1), v_2, \dots, v_m)$ so that $\phi \circ V = \{(\varphi(v_1), v_2, \dots, v_m) :$

$v \in V$ }. Using Python-like notation where $v_{2:}$ means $(v_2, v_3, \dots, v_m) \in \mathbb{R}^{m-1}$, we have

$$\begin{aligned} m \operatorname{Rad}(\phi \circ V) &= \mathbb{E}_{\sigma} \sup_{v \in V} [\sigma_1 \phi(v_1) + \sigma_{2:} \cdot v_{2:}] \\ &= \frac{1}{2} \mathbb{E}_{\sigma_{2:}} \sup_{v \in V} [\phi(v_1) + \sigma_{2:} \cdot v_{2:}] + \frac{1}{2} \mathbb{E}_{\sigma_{2:}} \sup_{v' \in V} [-\phi(v'_1) + \sigma_{2:} \cdot v'_{2:}] \\ &= \frac{1}{2} \mathbb{E}_{\sigma_{2:}} \sup_{v, v' \in V} [\phi(v_1) - \phi(v'_1) + \sigma_{2:} \cdot (v_{2:} + v'_{2:})]. \end{aligned}$$

Now, for points arbitrarily close to the supremum, $\phi(v_1) - \phi(v'_1)$ will always be nonnegative: if it were negative, simply swapping v and v' would make that term positive, and wouldn't affect the rest of the expression, making the objective bigger. Thus we can write

$$\begin{aligned} m \operatorname{Rad}(\phi \circ V) &= \frac{1}{2} \mathbb{E}_{\sigma_{2:}} \sup_{v, v' \in V} |\phi(v_1) - \phi(v'_1)| + \sigma_{2:} \cdot (v_{2:} + v'_{2:}) \\ &\leq \frac{1}{2} \mathbb{E}_{\sigma_{2:}} \sup_{v, v' \in V} |v_1 - v'_1| + \sigma_{2:} \cdot (v_{2:} + v'_{2:}) \end{aligned}$$

since ϕ is 1-Lipschitz. Now, notice that the objective of the maximization is identical if we swap v and v' , so for any point close to the supremum with $v_1 \leq v'_1$, there's an exactly equivalent one with $v_1 \geq v'_1$. Thus

$$\begin{aligned} m \operatorname{Rad}(\phi \circ V) &\leq \frac{1}{2} \mathbb{E}_{\sigma_{2:}} \sup_{v, v' \in V} v_1 - v'_1 + \sigma_{2:} \cdot (v_{2:} + v'_{2:}) \\ &= \frac{1}{2} \mathbb{E}_{\sigma_{2:}} \left(\sup_{v \in V} [v_1 + \sigma_{2:} \cdot v_{2:}] + \sup_{v' \in V} [-v'_1 + \sigma_{2:} \cdot v'_{2:}] \right) \\ &= \mathbb{E}_{\sigma} \sup_{v \in V} v \cdot \sigma = m \operatorname{Rad}(V). \quad \square \end{aligned}$$

How do we use this? Well, remember that for typical supervised learning losses,

$$\begin{aligned} (\ell \circ \mathcal{H})|_{\mathcal{S}} &= \{(\ell(h, z_1), \dots, \ell(h, z_m)) : h \in \mathcal{H}\} \\ &= \{(l_{y_1}(h(x_1)), \dots, l_{y_m}(h(x_m))) : h \in \mathcal{H}\} \\ &= (\mathbf{1}_{\mathcal{S}_y} \circ \mathcal{H})|_{\mathcal{S}_x}, \end{aligned}$$

where $\mathbf{1}_{\mathcal{S}_y}$ is a vectorized version of these losses (like ϕ above) for the vector of particular labels $\mathcal{S}_y = (y_1, \dots, y_m)$. Then we have a function of x only, so we apply it to $\mathcal{S}_x = (x_1, \dots, x_m)$. If the functions l_{y_i} are all M -Lipschitz, then Talagrand's lemma gives us that

Note that M here might depend on the particular \mathcal{S}_y !

$$\operatorname{Rad}((\ell \circ \mathcal{H})|_{\mathcal{S}}) \leq M \operatorname{Rad}(\mathcal{H}|_{\mathcal{S}_x}). \quad (5.3)$$

5.2.2 Complexity of bounded linear functions

When studying covering numbers, we considered logistic regression using the hypothesis class of bounded-norm linear functions,

$$\mathcal{H}_B = \{x \mapsto \langle w, x \rangle : \|w\| \leq B\}.$$

To analyze that with Rademacher complexity, the key term is

$$\operatorname{Rad}((\ell_{\log} \circ \mathcal{H}_B)|_{\mathcal{S}}) \leq \operatorname{Rad}(\mathcal{H}_B|_{\mathcal{S}_x}),$$

using (5.3) with Lemma 4.9 that logistic loss is 1-Lipschitz. Now let's bound that latter term:

$$\begin{aligned}
 m \operatorname{Rad}(\mathcal{H}_B|_{S_x}) &= \mathbb{E}_\sigma \sup_{\|w\| \leq B} \sum_i \sigma_i \langle w, x_i \rangle \\
 &= \mathbb{E}_\sigma \sup_{\|w\| \leq B} \left\langle w, \sum_i \sigma_i x_i \right\rangle \\
 &\leq \mathbb{E}_\sigma \sup_{\|w\| \leq B} \|w\| \left\| \sum_i \sigma_i x_i \right\| \\
 &= B \mathbb{E}_\sigma \left\| \sum_i \sigma_i x_i \right\| \\
 &\leq B \sqrt{\mathbb{E}_\sigma \left\| \sum_i \sigma_i x_i \right\|^2} \\
 &= B \sqrt{\mathbb{E}_\sigma \sum_{ij} \sigma_i \sigma_j \langle x_i, x_j \rangle} \\
 &= B \sqrt{\sum_i \underbrace{\mathbb{E}[\sigma_i^2]}_1 \|x_i\|^2 + \sum_{i \neq j} \underbrace{\mathbb{E}[\sigma_i \sigma_j]}_0 \langle x_i, x_j \rangle}.
 \end{aligned}$$

using Cauchy-Shwartz

using $(\mathbb{E} T)^2 \leq \mathbb{E} T^2$ so
 $|\mathbb{E} T| \leq \sqrt{\mathbb{E} T^2}$

Dividing both sides by m , we can rewrite this final inequality as

$$\operatorname{Rad}(\mathcal{H}_B|_{S_x}) \leq \frac{B}{\sqrt{m}} \sqrt{\frac{1}{m} \sum_i \|x_i\|^2}, \quad (5.4)$$

so this bound on the complexity depends on the particular S_x that you see, similar to the issue we had with covering numbers.

a.s. is “almost surely” = One solution (as we did before) is to assume that \mathcal{D} is such that $\|x\| \leq C$ (a.s.), “with probability one” something often true in practice. This would imply that $\operatorname{Rad}(\mathcal{H}_B|_{S_x}) \leq BC/\sqrt{m}$ (a.s.). Note that this gives us an expected-case bound on the excess error of ERM for logistic regression of

$$\mathbb{E}_{S \sim \mathcal{D}^m} L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) \leq \frac{2BC}{\sqrt{m}}; \quad (5.5)$$

we'll see in Section 5.3 that, in this case, we can convert this into a bound saying that, with probability at least $1 - \delta$,

$$L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) \leq \frac{BC}{\sqrt{m}} \left[2 + \sqrt{2 \log \frac{2}{\delta}} \right] = \mathcal{O}_p \left(\frac{BC}{\sqrt{m}} \right). \quad (5.6)$$

Compare this to the covering number-based bound we showed in (4.6):

$$L_{\mathcal{D}}(\hat{h}_S) - L_{\mathcal{D}}(h^*) \leq \frac{BC}{\sqrt{m}} \left[\frac{1}{\sqrt{2}} + \frac{1}{2} \sqrt{d \log(72m)} + \sqrt{2 \log \frac{2}{\delta}} \right] = \mathcal{O}_p \left(BC \sqrt{\frac{d \log m}{m}} \right).$$

Sometimes, though, we don't want to assume this hard upper bound on $\|x\|$; for example, what if our data is Gaussian? Again using that $\mathbb{E} X \leq \sqrt{\mathbb{E} X^2}$ for nonnegative

X , we can bound the expected value of (5.4) as

$$\mathbb{E}_S \text{Rad}(\mathcal{H}_B|_{S_x}) \leq \frac{B}{\sqrt{m}} \mathbb{E}_S \sqrt{\frac{1}{m} \sum_i \|x_i\|^2} \leq \frac{B}{\sqrt{m}} \sqrt{\mathbb{E} \|x\|^2}. \quad (5.7)$$

This only works for the average Rademacher complexity, which is the only thing we've seen to care about yet, but in some settings you do want a high-probability bound on $\text{Rad}(\mathcal{H}|_{S_x})$ rather than an average-case one.

This allows for much broader data distributions, as long as you can bound $\mathbb{E} \|x\|^2$. For example, for a Gaussian $x \sim \mathcal{N}(\mu, \Sigma)$ this is $\mathbb{E} \|x\|^2 = \|\mu\|^2 + \text{Tr}(\Sigma)$.

We've thus shown an average-case estimation error bound for bounded-norm linear problems with Lipschitz losses with a rate of $\mathcal{O}(1/\sqrt{m})$.

5.3 CONCENTRATION

Now let's prove that high-probability bound. We'll need a new tool: *McDiarmid's inequality*, which lets us show concentration of things *other* than sample averages.

THEOREM 5.6 ([McD89]). *Let X_1, \dots, X_m be independent, and let $f(X_1, \dots, X_m)$ be a real-valued function satisfying the bounded differences condition*

$$\forall i \in [m]. \quad \sup_{x_1, \dots, x_m, x'_i} |f(x_1, \dots, x_m) - f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_m)| \leq c_i.$$

Then, with probability at least $1 - \delta$,

$$f(X_1, \dots, X_m) \leq \mathbb{E} f(X_1, \dots, X_m) + \sqrt{\frac{1}{2} \left(\sum_{i=1}^m c_i^2 \right) \log \frac{1}{\delta}}.$$

Proof. Use $X_{i:j}$ to denote (X_i, \dots, X_j) . For any $k \in [m]$, freeze some arbitrary values for $x_{1:k-1} = (x_1, \dots, x_{k-1})$. We're going to consider $\mathbb{E}_{X_{k+1:m}} f(x_{1:k-1}, X_k, X_{k+1:m})$ as a random variable, which is random depending *only* on the value of X_k : the earlier arguments are frozen, and the later ones are being averaged over.

This proof has deep connections to martingale methods, but we won't talk any more about that. If you take Nick Harvey's randomized algorithms course, you can learn some more! Or read Section 2.2 of [Wai19] for a very brief intro, or read [McD89].

First, we know this variable is bounded: it can vary only in an interval of length at most c_k . By assumption, for any particular values for $x_{1:k-1}$ and $x_{k+1:m}$,

$$c_k \geq \sup_{x_k} f(x_{1:m}) - \inf_{x_k} f(x_{1:m}).$$

This is true for *any* values of $x_{k+1:m}$, so it's also true on average:

$$\begin{aligned} c_k &\geq \mathbb{E}_{X_{k+1:m}} \sup_{x_k} f(x_{1:k-1}, x_k, X_{k+1:m}) - \inf_{x_k} f(x_{1:k-1}, x_k, X_{k+1:m}) \\ &\geq \mathbb{E}_{X_{k+1:m}} \sup_{x_k} f(x_{1:k-1}, x_k, X_{k+1:m}) + \sup_{x_k} (-f(x_{1:k-1}, x_k, X_{k+1:m})) - \inf_{x_k} (-f(x_{1:k-1}, x_k, X_{k+1:m})) \\ &= \mathbb{E}_{X_{k+1:m}} \sup_{x_k, x'_k} f(x_{1:k-1}, x_k, X_{k+1:m}) - f(x_{1:k-1}, x'_k, X_{k+1:m}) \\ &\geq \sup_{x_k, x'_k} \mathbb{E}_{X_{k+1:m}} f(x_{1:k-1}, x_k, X_{k+1:m}) - f(x_{1:k-1}, x'_k, X_{k+1:m}) \\ &= \sup_{x_k} \mathbb{E}_{X_{k+1:m}} f(x_{1:k-1}, x_k, X_{k+1:m}) - \inf_{x_k} \mathbb{E}_{X_{k+1:m}} f(x_{1:k-1}, x_k, X_{k+1:m}). \end{aligned}$$

Lemma 5.1

Thus, by Hoeffding's lemma (Proposition 3.5), this variable is $\mathcal{SG}(c_k/2)$. That is, mul-

tipling the definition of subgaussianity (Definition 3.4) by $e^{\lambda \mathbb{E} X}$ for convenience,

$$\mathbb{E}_{X_k} \exp \left(\lambda \mathbb{E}_{X_{k+1:m}} f(x_{1:k-1}, X_k, X_{k+1:m}) \right) \leq \exp \left(\lambda \mathbb{E}_{X_k} \mathbb{E}_{X_{k+1:m}} f(x_{1:k-1}, X_k, X_{k+1:m}) + \frac{1}{8} \lambda^2 c_k^2 \right).$$

This inequality holds for any $x_{1:k-1}$, so let's take the expectation of both sides:

$$\mathbb{E}_{X_{1:k}} \exp \left(\lambda \mathbb{E}_{X_{k+1:m}} f(X_{1:m}) \right) \leq \mathbb{E}_{X_{1:k-1}} \exp \left(\lambda \mathbb{E}_{X_{k:m}} f(X_{1:m}) + \frac{1}{8} \lambda^2 c_k^2 \right).$$

That inequality holds for each choice of k . Let's take the log of each one, and add them all up:

$$\sum_{k=1}^m \log \mathbb{E}_{X_{1:k}} \exp \left(\lambda \mathbb{E}_{X_{k+1:m}} f(X_{1:m}) \right) \leq \sum_{k=1}^m \left[\log \mathbb{E}_{X_{1:k-1}} \exp \left(\lambda \mathbb{E}_{X_{k:m}} f(X_{1:m}) \right) + \frac{1}{8} \lambda^2 c_k^2 \right].$$

Letting $a_k = \log \mathbb{E}_{X_{1:k}} \exp(\lambda \mathbb{E}_{X_{k+1:m}} f(X_{1:m}))$, we have

$$\sum_{k=1}^m a_k \leq \sum_{k=1}^m a_{k-1} + \sum_{k=1}^m \frac{1}{8} \lambda^2 c_k^2.$$

Most of the terms cancel, leaving us a_m on the left and a_0 on the right:

$$\log \mathbb{E}_{X_{1:m}} \exp(\lambda f(X_{1:m})) \leq \log \exp \left(\lambda \mathbb{E}_{X_{1:m}} f(X_{1:m}) \right) + \frac{1}{8} \lambda^2 \sum_{k=1}^m c_k^2.$$

Taking the exponential of both sides and rearranging,

$$\mathbb{E}_{X_{1:m}} \exp \left(\lambda \left(f(X_{1:m}) - \mathbb{E}_{X_{1:m}} f(X_{1:m}) \right) \right) \leq \exp \left(\frac{1}{2} \lambda^2 \cdot \frac{1}{4} \sum_{k=1}^m c_k^2 \right).$$

This is exactly the definition of $f(X_{1:m}) \in \mathcal{SG} \left(\frac{1}{2} \sqrt{\sum_{i=1}^m c_i^2} \right)$. The Chernoff bound for subgaussians (Proposition 3.8) then tells us that with probability at least $1 - \delta$,

$$f(X_{1:m}) \leq \mathbb{E} f(X_{1:m}) + \frac{1}{2} \sqrt{\sum_{i=1}^m c_i^2} \cdot \sqrt{2 \log \frac{1}{\delta}}. \quad \square$$

Considering $-f$ gives an identical form for the lower bound, and a union bound gives an absolute value version by replacing $\frac{1}{8}$ with $\frac{2}{8}$.

Notice that if $c_i = c$ for all i , then $\sqrt{\sum_{i=1}^m c_i^2} = c \sqrt{m}$.

(It's also worth checking for yourself that when $f(X_{1:m}) = \frac{1}{m} \sum_{i=1}^m X_i$, you exactly recover the bounded version of Hoeffding's inequality.)

Now that we know McDiarmid's inequality, we can *directly* apply it to get a high-probability bound:

THEOREM 5.7. *Suppose that $\ell(h, z) \in [a, b]$ for all h, z . Then, with probability at least $1 - \delta$,*

$$\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq \mathbb{E} \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] + (b - a) \sqrt{\frac{1}{2m} \log \frac{1}{\delta}}. \quad (5.8)$$

Thus, if \hat{h}_S is an ERM, we have with probability at least $1 - \delta$ that

$$L_{\mathcal{D}}(\hat{h}_S) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq \mathbb{E} \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] + (b - a) \sqrt{\frac{2}{m} \log \frac{2}{\delta}}. \quad (5.9)$$

Proof. Let $S^{(i)} = (z_1, \dots, z_{i-1}, z', z_{i+1}, \dots, z_m)$. Now, we have

$$L_{\mathcal{D}}(h) - L_S(h) = L_{\mathcal{D}}(h) - L_{S^{(i)}}(h) + L_{S^{(i)}}(h) - L_S(h);$$

thus, expanding out $L_{S^{(i)}}(h) - L_S(h)$,

$$\sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] - \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_{S^{(i)}}(h)] \leq \sup_{h \in \mathcal{H}} \frac{1}{m} [\ell(h, z') - \ell(h, z)] \leq \frac{b - a}{m}$$

because the loss is bounded. The same holds in the other direction:

$$\sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_{S^{(i)}}(h)] - \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] \leq \sup_{h \in \mathcal{H}} \frac{1}{m} [\ell(h, z) - \ell(h, z')] \leq \frac{b - a}{m}.$$

Therefore the worst-case generalization gap, $f(S) = \sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h)$, satisfies the bounded differences condition with $c = (b - a)/m$. Equation (5.8) follows by applying McDiarmid.

The other result follows as usual for our ERM bounds: we know that for any $h^* \in \mathcal{H}$,

$$\begin{aligned} L_{\mathcal{D}}(\hat{h}_S) &\leq L_S(\hat{h}_S) + \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] \\ &\leq L_S(\hat{h}_S) + \mathbb{E} \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] + (b - a) \sqrt{\frac{1}{2m} \log \frac{2}{\delta}} && (5.8), w/ \text{prob. } 1 - \delta/2 \\ &\leq L_S(h^*) + \mathbb{E} \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] + (b - a) \sqrt{\frac{1}{2m} \log \frac{2}{\delta}} && \text{definition of ERM} \\ &\leq L_{\mathcal{D}}(h^*) + (b - a) \sqrt{\frac{1}{2m} \log \frac{2}{\delta}} + \mathbb{E} \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] + (b - a) \sqrt{\frac{1}{2m} \log \frac{2}{\delta}}, && \text{Hoeffding, w/ prob. } 1 - \delta/2 \end{aligned}$$

and the result follows since h^* was arbitrary. \square

For bounded-norm bounded-data logistic regression, using (5.5) and (4.4) in (5.9) gives (5.6).

6 Growth functions and VC dimension

So far, we've mainly talked about logistic regression. We proved some bounds that ERM obtains nearly the optimal value of the logistic loss over a bounded ball of weight vectors, but we haven't actually said anything yet about 0-1 loss (i.e. accuracy).

We're going to focus for now on binary classifiers, i.e. h that output a binary label, not a continuous one. If we're doing logistic regression, we're thinking about the "hard prediction," not the logit or the predicted probability.

6.1 ZERO-ONE LOSS

If $h(x) \in \{-1, 1\}$ and $y \in \{-1, 1\}$, then the 0-1 loss is

$$l_y(\hat{y}) = \begin{cases} 0 & \hat{y} = y \\ 1 & \hat{y} \neq y. \end{cases}$$

This isn't a function on \mathbb{R} , so applying Talagrand's lemma is a little weird. The trick is, though: for computing the loss, we can just extend the function l_y to \mathbb{R} in any way at all, and the loss will be exactly the same – it just doesn't care what l_y does for *other* values of \hat{y} .

So, let's just pick a Lipschitz function on \mathbb{R} that agrees at the points we need, by linear interpolation:

$$l_y(\hat{y}) = \begin{cases} 0 & y\hat{y} \geq 1 \\ \frac{1}{2} - \frac{1}{2}y\hat{y} & -1 \leq y\hat{y} \leq 1 \\ 1 & y\hat{y} \leq -1. \end{cases}$$

This has $\|l_y\|_{\text{Lip}} = \frac{1}{2}|y| = \frac{1}{2}$. Thus:

PROPOSITION 6.1. *If $\mathcal{H}_{-1,1}$ is a hypothesis class with outputs in $\{-1, 1\}$,*

$$\text{Rad}((\ell_{0-1} \circ \mathcal{H}_{-1,1})|_{\mathcal{S}}) \leq \frac{1}{2} \text{Rad}(\mathcal{H}_{-1,1}|_{\mathcal{S}_x}). \quad (6.1)$$

If instead $\mathcal{H}_{0,1}$ maps to $\{0, 1\}$,

$$\text{Rad}((\ell_{0-1} \circ \mathcal{H}_{0,1})|_{\mathcal{S}}) \leq \text{Rad}(\mathcal{H}_{0,1}|_{\mathcal{S}_x}). \quad (6.2)$$

Proof. The first result just applies Talagrand's contraction lemma (Lemma 5.4) to the extended l_y above.

For the $\{0, 1\}$ case, we can either do the same thing with a slightly different 1-Lipschitz function, or we can note that we can convert a $\{0, 1\}$ classifier to a $\{-1, 1\}$

classifier by taking $2h - 1$ and use basic properties of Rademacher complexity to see

$$\text{Rad}(\mathcal{H}_{-1,1}|_{S_x}) = \text{Rad}((2\mathcal{H}_{0,1} - 1)|_{S_x}) = 2 \text{Rad}(\mathcal{H}_{0,1}|_{S_x}). \quad \square$$

6.2 FINITE SETS

How do we bound $\text{Rad}(\mathcal{H}|_{S_x})$ for binary classifiers?

One major way is to note that, for binary classifiers,

$$\mathcal{H}|_{S_x} = \{(h(x_1), \dots, h(x_n)) : h \in \mathcal{H}\} \subseteq \{0, 1\}^m$$

– and so it can't be too big. There are only 2^m possible bitvectors of behaviour on the particular set S_x , *even if \mathcal{H} is infinite*. In fact, there may be many fewer possible things that \mathcal{H} is able to do on this particular S_x .

So, let's first try bounding the Rademacher complexity of an arbitrary finite set based on its size.

LEMMA 6.2. *If V is finite and $\|v\| \leq B$ for all $v \in V$, then*

$$\text{Rad}(V) \leq \frac{B}{m} \sqrt{2 \log |V|}.$$

Proof. We have

$$\text{Rad}(V) = \mathbb{E} \max_{v \in V} \sum_{i=1}^m \frac{\sigma_i v_i}{m}.$$

Considering any one v for now, $\sum_{i=1}^m \sigma_i v_i$ is a random variable (depending on σ). It has mean zero, and since σ_i is $\mathcal{SG}\left(\frac{1-(-1)}{2}\right) = \mathcal{SG}(1)$ by Hoeffding's lemma, $v_i \sigma_i / m$ is $\mathcal{SG}(|v_i|/m)$. The $v_i \sigma_i / m$ for each i are independent of one another, so this means

$$\sum_{i=1}^m \frac{v_i \sigma_i}{m} \in \mathcal{SG}\left(\sqrt{\sum_{i=1}^m \left(\frac{|v_i|}{m}\right)^2}\right) = \mathcal{SG}\left(\frac{\|v\|}{m}\right) \subseteq \mathcal{SG}\left(\frac{B}{m}\right).$$

We now want to find the expected max of these $|V|$ random variables. Each is mean zero and $\mathcal{SG}(B/m)$; they're dependent, since they all use the same σ , but that's okay. Lemma 6.3 handles exactly this situation, giving our desired result. \square

LEMMA 6.3. *Let X_1, \dots, X_n be zero-mean random variables that are each $\mathcal{SG}(\sigma)$, which are **not** necessarily independent. Then $\mathbb{E}[\max_{i \in [n]} X_i] \leq \sigma \sqrt{2 \log(n)}$.*

Proof. This is Assignment 2, Question 2.4. \square

We can then specialize this to the binary classifier case:

COROLLARY 6.4. *For binary classifiers mapping to $\{-1, 1\}$, $\text{Rad}(\mathcal{H}_{-1,1}|_{S_x}) \leq \sqrt{\frac{2}{m} \log |\mathcal{H}|_{S_x}}$.*

For binary classifiers mapping to $\{0, 1\}$, $\text{Rad}(\mathcal{H}_{0,1}|_{S_x}) \leq \sqrt{\frac{1}{2m} \log |\mathcal{H}|_{S_x}}$.

Proof. For binary classifiers mapping to ± 1 , $|h(x)| = 1$ so $\|h|_{S_x}\| = \sqrt{m}$; the result follows by plugging in to Lemma 6.2. For binary classifiers mapping to $\{0, 1\}$, it's half of that, by the same scaling-and-translating conversion as before. \square

Thus Proposition 6.1 and Theorems 5.3 and 5.7 give that for binary classifiers and zero-one loss,

$$\mathbb{E}_{S \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] \leq \mathbb{E}_{S \sim \mathcal{D}^m} \sqrt{\frac{2}{m} \log |\mathcal{H}|_{S_x}} \quad (6.3)$$

You might want to check for yourself that this same equation holds whether \mathcal{H} maps to $\{0, 1\}$ or $\{-1, 1\}$, or indeed any other two-element set.

$$\Pr_{S \sim \mathcal{D}^m} \left(L_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq \mathbb{E}_{S \sim \mathcal{D}^m} \sqrt{\frac{2}{m} \log |\mathcal{H}|_{S_x}} + \sqrt{\frac{2}{m} \log \frac{2}{\delta}} \right) \geq 1 - \delta. \quad (6.4)$$

Using just that $|\mathcal{H}|_{S_x} \leq |\mathcal{H}|$, this becomes that with probability at least $1 - \delta$

$$L_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq \sqrt{\frac{2}{m} \log |\mathcal{H}|} + \sqrt{\frac{2}{m} \log \frac{2}{\delta}}.$$

This is a tiny bit worse than the much more direct bound of Proposition 2.2,

$$L_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq \sqrt{\frac{2}{m} \log \frac{|\mathcal{H}| + 1}{\delta}} \leq \sqrt{\frac{2}{m} \left[\log |\mathcal{H}| + \log \frac{2}{\delta} \right]},$$

using $|\mathcal{H}| + 1 \leq 2|\mathcal{H}|$.

6.3 GROWTH FUNCTIONS

The bound $|\mathcal{H}|_{S_x} \leq |\mathcal{H}|$ is potentially very, very loose, though. For instance, we know the left-hand side can't be more than 2^m , even if the right-hand side is infinite.

Plugging in that 2^m bound would only give $\mathbb{E}_S \sup_h L_{\mathcal{D}}(h) - L_S(h) \leq \sqrt{2 \log 2} \approx 1.2$, which is not very interesting since the generalization gap is trivially at most 1! But, when $|\mathcal{H}|_{S_x} = o(2^m)$, this is far more interesting.

DEFINITION 6.5. The *growth function* $\Gamma_{\mathcal{H}}(m)$ of a hypothesis class \mathcal{H} is given by

$$\Gamma_{\mathcal{H}}(m) = \sup_{x_1, \dots, x_m \in \mathcal{X}} |\mathcal{H}|_{(x_1, \dots, x_m)}.$$

By definition, $|\mathcal{H}|_{S_x} \leq \Gamma_{\mathcal{H}}(m)$ for any S_x of size m ; thus for binary classifiers with zero-one loss, we immediately know that the expected worst-case generalization gap is at most $\sqrt{\frac{2}{m} \log \Gamma_{\mathcal{H}}(m)}$.

Unlike our previous Rademacher or covering number bounds, we've now dropped *all* dependence on the particular distribution \mathcal{D} ; this is a purely combinatorial notion. That's helpful if we're trying to show PAC learning.

It's sometimes possible to compute growth functions directly – you'll do this in an assignment – but it's usually much easier to get a bound with the VC dimension.

6.4 VC DIMENSION

DEFINITION 6.6. A hypothesis class \mathcal{H} is said to *shatter* a set $S_x \subseteq \mathcal{X}$ if it can achieve all possible labellings of S_x , i.e. $|\mathcal{H}|_{S_x} = 2^m$.

DEFINITION 6.7. The *VC dimension* of \mathcal{H} is the size of the largest set \mathcal{H} can shatter:

$$\text{VCdim}(\mathcal{H}) = \max(\{m \geq 0 : \Gamma_{\mathcal{H}}(m) = 2^m\}).$$

VC is for Vladimir Vapnik and Alexey Chervonenkis, Soviet mathematicians who developed this theory starting in the 60s (well before PAC learning); the English translation of the first key paper is [VC71].

If \mathcal{H} can shatter unboundedly large sets, we say its VC dimension is infinite.

It turns out that we can bound the growth function in terms of the VC dimension: $\Gamma_{\mathcal{H}}(m) = \mathcal{O}(m^{\text{VCdim}(\mathcal{H})})$, which then gives us that the expected worst-case generalization gap is $\tilde{\mathcal{O}}(\sqrt{2 \text{VCdim}(\mathcal{H})/m})$. We'll prove this in Section 6.4.2; let's first explore how to compute the VC dimension for some different \mathcal{H} .

6.4.1 Examples of computing VC dimension

It will be useful for all of our examples below to note that if you can't shatter any set of size m , you also can't shatter any set of size $m' > m$: if you could, then by definition you could shatter any size- m subset of the larger set.

6.4.1.1 Threshold functions

Let $h_a : \mathbb{R} \rightarrow \{0, 1\}$ be a threshold function $h_a(x) = \mathbb{1}(x \geq a)$, and let $\mathcal{H} = \{h_a : a \in \mathbb{R}\}$.

We can shatter any set of size 1, but for VC dimension we only have to show that we can shatter one particular set of that size.

To start: we can shatter, say, $S_x = \{0\}$, because $h_{-1}(0) = 1$ and $h_1(0) = 0$. Thus $\text{VCdim}(\mathcal{H}) \geq |S_x| = 1$.

But we can't shatter any set S_x of size $|S_x| \geq 2$. Let $a, b \in S_x$ with $a < b$. We can't get $h(a) = 1$ and $h(b) = 0$ with the same $h \in \mathcal{H}$, since all $h \in \mathcal{H}$ are nondecreasing. Thus no S_x of size 2 can be shattered, and so $\text{VCdim}(\mathcal{H}) < 2$.

Thus $\text{VCdim}(\mathcal{H}) = 1$.

6.4.1.2 Circles

For $\mathcal{X} = \mathbb{R}^2$, consider $\mathcal{H} = \{h_{r,c} : r > 0, c \in \mathbb{R}^2\}$ with $h_{r,c}(x) = \mathbb{1}(\|x - c\| \leq r)$, the set of indicator functions of circles.

We can shatter any set of size two, since we can draw a circle that includes both points, one that includes either point, or one that includes neither point.

A bunch of these examples are easier to see if you draw them out! I'll try to add some TikZ pictures, but in the meantime you can draw them yourself.

We can also shatter *some* sets of size three, since if we put them in an equilateral triangle we can pick out none, or any one, two, or all three points. (If we put the three points in a line, we can't pick out the two edges but not the middle – but that's okay, VC dimension is about *the largest* set you can shatter.)

Claim: we cannot shatter any set of size four, and so $\text{VCdim}(\mathcal{H}) = 3$. If we think of the points as lying roughly in a rectangle, then we can't pick out opposite corners without including at least one of the other points. (Ideally you'd formalize this argument, but let's not do that now.)

6.4.1.3 Homogeneous linear threshold functions in \mathbb{R}^2

Let $\mathcal{X} = \mathbb{R}^2$ and consider $\mathcal{H} = \{x \mapsto \text{sgn}(w \cdot x) : w \in \mathbb{R}^2\}$: hyperplanes passing through the origin. We're using $\mathcal{Y} = \{-1, 1\}$, and we're going to define a function sgn which is like the sign except that $\text{sgn}(0) = 1$ – yeah, that sucks but so do all the other options. If you want to use $\mathcal{Y} = \{0, 1\}$, then instead write $\mathbb{1}(w \cdot x \geq 0)$; that's nicer to write down, but more annoying to work with.

We can shatter at least some sets of size 2: e.g. $\{(-1, 1), (1, 1)\}$, we can put the hyperplane along the x -axis to get both the same sign, or put it in along the y -axis to get them with opposite signs.

We can't shatter any sets of size 3. If the convex hull of the points contains the origin, then we can't get them all with the same sign; if the hull doesn't contain the origin, then we can't label them like $(1, 0, 1)$.

A convex hull of a set is the smallest convex set containing the original set: $\text{conv}(V) = \{\alpha v + (1 - \alpha)v' : v, v' \in V, \alpha \in [0, 1]\}$. If you have some points in \mathbb{R}^2 , you draw straight lines connecting the "outside" points to include all the points.

So homogenous 2-d linear threshold functions have VC dimension 2.

6.4.1.4 Homogeneous linear threshold functions in \mathbb{R}^d

PROPOSITION 6.8. *Let $\mathcal{H} = \{x \mapsto \text{sgn}(w \cdot x) : w \in \mathbb{R}^d\}$. Then $\text{VCdim}(\mathcal{H}) = d$.*

Proof. We can shatter a set of size d : take the set $\{e_1, \dots, e_d\}$ for e_i the i th standard basis vector, i.e. the one-hot vector with a 1 in the i th position and 0 everywhere else. Then we can achieve an arbitrary labeling $(y_1, \dots, y_d) \in \{0, 1\}^d$ by setting $w_i = y_i$: we get $w \cdot e_i = y_i$.

To show that we cannot shatter any set of size $d + 1$, let x_1, \dots, x_{d+1} be a set of $d + 1$ points in \mathbb{R}^d . Then they can't be linearly independent: there must be some $\alpha_1, \dots, \alpha_{d+1}$ such that $\sum_{i=1}^{d+1} \alpha_i x_i = 0$, with not all the α_i zero. Let $\mathcal{I}_+ = \{i \in [d + 1] : \alpha_i > 0\}$, $\mathcal{I}_0 = \{i \in [d + 1] : \alpha_i = 0\}$, and $\mathcal{I}_- = \{i \in [d + 1] : \alpha_i < 0\}$.

Now, if \mathcal{H} can shatter $\{x_1, \dots, x_{d+1}\}$, we can ask it to assign 1 to the x_i with $i \in \mathcal{I}_+ \cup \mathcal{I}_0$, and -1 to the x_i with $i \in \mathcal{I}_-$. Then we'd have

$$0 = w \cdot 0 = w \cdot \sum_{i=1}^{d+1} (\alpha_i x_i) = \sum_{i \in \mathcal{I}_+} \underbrace{\alpha_i}_{>0} \underbrace{w \cdot x_i}_{\geq 0} + \sum_{i \in \mathcal{I}_0} \underbrace{\alpha_i}_0 \underbrace{w \cdot x_i}_0 + \sum_{i \in \mathcal{I}_-} \underbrace{\alpha_i}_{<0} \underbrace{w \cdot x_i}_{<0}.$$

Suppose that \mathcal{I}_- is nonempty, or that there are any $i \in \mathcal{I}_+$ such that $w \cdot x_i > 0$. Then the sum on the right-hand side is strictly positive, contradicting that it equals 0.

Thus, either we cannot shatter $\{x_1, \dots, x_{d+1}\}$, or we can but $w \cdot x_i = 0$ for all $i \in \mathcal{I}_+$ and $\mathcal{I}_- = \{\}$. Considering the second case, we must have $\sum_{i \in \mathcal{I}_+} \alpha_i x_i = 0$. Now, find some \tilde{w} that labels all these points as negative, $\tilde{w} \cdot x_i < 0$ for all $i \in \mathcal{I}_+$; this must be possible if the set is shattered. Then we'd have

[SSBD14] misses analyzing this case :(, since they just pretend $w \cdot x = 0$ is impossible.

$$0 = \tilde{w} \cdot 0 = \tilde{w} \cdot \left(\sum_{i \in \mathcal{I}_+} \alpha_i x_i \right) = \sum_{i \in \mathcal{I}_+} \underbrace{\alpha_i}_{>0} \underbrace{\tilde{w} \cdot x_i}_{<0} < 0,$$

a contradiction. We're left to conclude that \mathcal{H} cannot shatter $\{x_1, \dots, x_{d+1}\}$. □

6.4.1.5 Inhomogeneous linear threshold functions in \mathbb{R}^d

What about if we don't enforce that the hyperplane passes through the origin, $\mathcal{H} = \{x \mapsto \text{sgn}(w \cdot x + b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$?

We could analyze this directly, similarly to what we did above; this is Example 3.12 of [MRT18] if you want to see it.

But we can also reduce to the set of homogeneous linear classifiers: if we have d -dimensional data, we can model that as homogeneous linear classifiers on $(d + 1)$ -dimensional data with an extra "dummy feature" that's always 1. The weight w_0 corresponding to that feature will just be the offset b .

Using this reduction, we can see:

PROPOSITION 6.9. For $x \in \mathbb{R}^d$, $\text{VCdim}(\{x \mapsto \text{sgn}(w \cdot x + b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}) = d + 1$.

Proof. First, we can shatter the set $\{0, e_1, \dots, e_d\}$, which has size $d + 1$, like before. We set $w_0 = y_0/2$ and $w_i = y_i$; the $y_0/2$ only affects the sign if all the other weights are “off”, i.e. only on the 0 vector.

Also, we can't shatter any set of size $d + 2$. If we could, then there would be $d + 2$ vectors in \mathbb{R}^{d+1} shattered by the class of homogeneous thresholds; but that class has VC dimension $d + 1$ by Proposition 6.8, so that's not possible. \square

6.4.2 Growth function bounds in terms of VC: Sauer-Shelah

As mentioned before, we're going to show that $\Gamma_{\mathcal{H}}(m)$ is $\mathcal{O}(m^{\text{VCdim}(\mathcal{H})})$. Remember that for $m \leq \text{VCdim}(\mathcal{H})$, we know that $\Gamma_{\mathcal{H}}(m) = 2^m$; this means that $\Gamma_{\mathcal{H}}$ always grows exponentially up to some point, then drops off to just polynomial growth.

This e is $\exp(1) \approx 2.7$. **COROLLARY 6.10.** If $m \geq d = \text{VCdim}(\mathcal{H})$, then $\Gamma_{\mathcal{H}}(m) \leq \left(\frac{em}{d}\right)^d$.

Plugging into (6.3) and (6.4) gives

THEOREM 6.11. Let \mathcal{H} be a class of binary classifiers with $\text{VCdim}(\mathcal{H}) = d$, and use the zero-one loss. For any $m \geq d$, we have that

$$\begin{aligned} \mathbb{E}_{S \sim \mathcal{D}^m} \sup_{h \in \mathcal{H}} [L_{\mathcal{D}}(h) - L_S(h)] &\leq \sqrt{\frac{2d}{m} [\log m + 1 - \log d]} \\ \Pr_{S \sim \mathcal{D}^m} \left(L_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) - \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \leq \sqrt{\frac{2d}{m} [\log m + 1 - \log d]} + \sqrt{\frac{2}{m} \log \frac{2}{\delta}} \right) &\geq 1 - \delta. \end{aligned}$$

When $d \geq 3$, we can replace $\log m + 1 - \log d$ with simply $\log m$ above.

We'll prove Corollary 6.10 as a corollary to the following:

LEMMA 6.12 (Sauer-Shelah). Let $\text{VCdim}(\mathcal{H}) \leq d < \infty$. Then $\Gamma_{\mathcal{H}}(m) \leq \sum_{i=0}^d \binom{m}{i}$.

Proof of Corollary 6.10 given Lemma 6.12. We need to show that $\sum_{i=0}^d \binom{m}{i} \leq \left(\frac{em}{d}\right)^d$ for $m \geq d$. We can do this by

$$\begin{aligned} \sum_{i=0}^d \binom{m}{i} &\leq \sum_{i=0}^d \binom{m}{i} \left(\frac{m}{d}\right)^{d-i} && \text{multiply each term by } \geq 1 \\ &\leq \sum_{i=0}^m \binom{m}{i} \left(\frac{m}{d}\right)^{d-i} && \text{add nonnegative terms} \\ &= \left(\frac{m}{d}\right)^d \sum_{i=0}^m \binom{m}{i} \left(\frac{d}{m}\right)^i \\ &= \left(\frac{m}{d}\right)^d \left(1 + \frac{d}{m}\right)^m && \text{binomial theorem} \\ &\leq \left(\frac{m}{d}\right)^d e^d && 1 + x \leq \exp(x). \quad \square \end{aligned}$$

This might be our first time using that $1 + x \leq \exp(x)$; it follows e.g. from Taylor's theorem, and is a useful thing that comes up a lot.

Now, we'll actually prove Lemma 6.12 itself as a corollary to the following result:

LEMMA 6.13 (Pajor). *For all finite $S \subseteq \mathcal{X}$, $|\mathcal{H}|_S \leq |\{T \subseteq S : T \text{ is shattered by } \mathcal{H}\}|$.*

If S is shattered, both sides of the inequality are $2^{|S|}$; otherwise, it's not obvious that these things should be related.

Proof of Lemma 6.12 given Lemma 6.13. To bound the number of shattered subsets of S in Lemma 6.13, recall there can't possibly be any with size larger than $d = \text{VCdim}(\mathcal{H})$; the number of sets it can shatter is thus upper-bounded by the number of subsets of S of size at most d , which is just $\sum_{i=0}^d \binom{m}{i}$ for $m = |S|$. \square

Proof of Lemma 6.13. We'll proceed by (strong) induction on $|\mathcal{H}|_S$.

Base case: $|\mathcal{H}|_S = 1$. For the right-hand side, the empty set is trivially shattered by any \mathcal{H} , so the RHS is always at least 1 as well, and the inequality holds.

Inductive case: $|\mathcal{H}|_S \geq 2$ and the inequality holds for any T with $|\mathcal{H}|_T < |\mathcal{H}|_S$. Then, since there are two distinct labelings, there must be at least one point $x \in S$ that achieves both $h(x) = 1$ and $h'(x) = 0$ for some $h, h' \in \mathcal{H}$. Partition \mathcal{H} into $\mathcal{H}_+ = \{h \in \mathcal{H} : h(x) = 1\}$ and $\mathcal{H}_- = \{h \in \mathcal{H} : h(x) = 0\}$. Now,

$$|\mathcal{H}|_S = |\mathcal{H}_+|_S + |\mathcal{H}_-|_S,$$

since the two produce disjoint labelings on S (they always disagree on x). They also produce fewer labelings than $\mathcal{H}|_S$ itself (there's at least one labeling in each), so we can apply the inductive hypothesis to each.

Defining $\text{Shat}_{\mathcal{H}}(S) = \{T \subseteq S : T \text{ is shattered by } \mathcal{H}\}$, we've shown that

$$|\mathcal{H}|_S \leq |\text{Shat}_{\mathcal{H}_+}(S)| + |\text{Shat}_{\mathcal{H}_-}(S)|.$$

Note the right-hand side is exactly, keeping track of the "double-counted" sets,

$$|\text{Shat}_{\mathcal{H}_+}(S) \cup \text{Shat}_{\mathcal{H}_-}(S)| + |\text{Shat}_{\mathcal{H}_+}(S) \cap \text{Shat}_{\mathcal{H}_-}(S)|;$$

it remains to argue that this is at most $|\text{Shat}_{\mathcal{H}}(S)|$. To see this, first note that $\text{Shat}_{\mathcal{H}_+}(S) \cup \text{Shat}_{\mathcal{H}_-}(S) \subseteq \text{Shat}_{\mathcal{H}}(S)$.

Now, consider a set $T \in \text{Shat}_{\mathcal{H}_+}(S) \cap \text{Shat}_{\mathcal{H}_-}(S)$, i.e. one that's been double-counted. Then note that $T' = T \cup \{x\}$ is not in either $\text{Shat}_{\mathcal{H}_+}(S)$ or $\text{Shat}_{\mathcal{H}_-}(S)$, since these classes by definition cannot shatter $\{x\}$, and so can't shatter a superset of $\{x\}$ either. But \mathcal{H} can shatter T' : there's a hypothesis in \mathcal{H}_- to achieve any desired labeling with $h(x) = 0$ (since $T \in \text{Shat}_{\mathcal{H}_-}(S)$), and likewise there's a hypothesis in \mathcal{H}_+ for any labeling with $h(x) = 1$. So $T' \in \text{Shat}_{\mathcal{H}}(S)$. Also, each such double-counted T corresponds to a different T' , since $x \notin T$ for each of these T s. Thus

$$|\text{Shat}_{\mathcal{H}_+}(S) \cap \text{Shat}_{\mathcal{H}_-}(S)| \leq \left| \text{Shat}_{\mathcal{H}}(S) \setminus (\text{Shat}_{\mathcal{H}_+}(S) \cup \text{Shat}_{\mathcal{H}_-}(S)) \right|,$$

and so $|\mathcal{H}|_S \leq |\text{Shat}_{\mathcal{H}}(S)|$ as desired. \square

7 Online learning

This chapter was primarily written by Bingshan Hu.

In batch (offline) learning, usually, we have a learning phase and a prediction phase. Learner first receives a batch of i.i.d. training samples and then uses these data to learn a hypothesis, e.g., an ERM (learning phase). The learned hypothesis will be used for predicting the labels of future samples (prediction phase). In contrast, in online learning, there is no separation between the learning phase and the prediction phase. We blend these two phases together by specifying a learning protocol that regulates all parties participating in the learning.

7.1 ONLINE BINARY CLASSIFICATION IN REALIZABLE SETTING

Still, we have an instance space \mathcal{X} , a label space $\mathcal{Y} = \{-1, 1\}$, a hypothesis class $\mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$, and the 0-1 loss function $\ell_{0-1}(y, \hat{y}) = \mathbf{1}\{y \neq \hat{y}\}$. We play this game sequentially with the following learning protocol.

Chapter 21.1 of [SSBD14].

In each round $t = 1, 2, \dots, T$,

1. Nature (Adversary/Environment) selects $x_t \in \mathcal{X}$ and reveals it to Learner ;
2. Learner chooses a hypothesis $f_t \in \mathcal{H}$ and predicts $\hat{y}_t = f_t(x_t) \in \{-1, 1\}$;
3. Nature plays label y_t and reveals it to Learner ;
4. Learner obtains a data sample (x_t, y_t) and suffers loss $\ell(y_t, \hat{y}_t) = \mathbf{1}\{y_t \neq \hat{y}_t\}$.

This is a picture for this.

If $\ell(y_t, \hat{y}_t) = \mathbf{1}\{y_t \neq \hat{y}_t\} = 1$, we say Learner makes a mistake in round t , as its predicted label is not correct. The goal of Learner is to make as few mistakes as possible. Intuitively, to choose a good predictor f_t in round t , Learner should use all the data samples attained in previous rounds $S_{t-1} := ((x_1, y_1), (x_2, y_2), \dots, (x_{t-1}, y_{t-1}))$ and even the input x_t in round t .

The data sequence $((x_1, y_1), (x_2, y_2), \dots, (x_T, y_T))$ does not need to be iid, which is quite different from the assumption usually batch learning needs.

No statistical assumption.

Since Adversary can decide y_t based on \hat{y}_t , it is hopeless to learn for some some hypothesis class \mathcal{H} , e.g., two constant functions $x \mapsto +1$ and $x \mapsto -1$ are in \mathcal{H} . Because Adversary can make Learner unhappy in each round by declaring $y_t = -\hat{y}_t$.

So, we need to put some constraints for Adversary.

We get started with a simple learning problem setup. It is about the aforementioned binary classification problem under the assumption of *realizability*, i.e., the true label function $f^* \in \mathcal{H}$, i.e., $y_t = f^*(x_t)$ for all $t \in [T]$, and Learner knows \mathcal{H} and the fact

With the assumption of realizability, setting $y_t = -\hat{y}_t$ for all $t \in [T]$ may not always be possible.

that $f^* \in \mathcal{H}$.¹ With this restriction on how Adversary generates the data sequence, Learner should make as few mistakes as possible. We are interested in how many mistakes Learner makes after playing this sequential game after T rounds.

HYPOTHESIS CLASS \mathcal{H} IS FINITE. Since we know the true label function $f^* \in \mathcal{H}$, i.e., there is a perfect hypothesis incurring zero loss in \mathcal{H} , it is quite natural to eliminate any hypothesis in \mathcal{H} that has made a mistake after observing (x_t, y_t) at the end of each round t . This intuition gives us an idea, called *version space*, to develop learning algorithms. The version space at the end of round t is defined as $\mathcal{H}_t = \{f \in \mathcal{H} : f(x_s) = y_s, \forall s \in [t]\}$. Then, we can maintain a sequence of version spaces $\mathcal{H} = \mathcal{H}_0 \supseteq \mathcal{H}_1 \supseteq \dots \supseteq \mathcal{H}_{t-1} \supseteq \mathcal{H}_t \supseteq \dots \supseteq \{f^*\}$. Hypotheses that have made mistakes will be kicked out from the version space at the end of each round t . We eliminate hypotheses from \mathcal{H} during the learning once we are confident they are not good almost surely!

Let's see **Consistent Algorithm** first.

Initially, we set $\mathcal{H}_0 = \mathcal{H}$.

In each round $t = 1, 2, \dots, T$,

1. Learner receives x_t ;
2. Learner chooses $f_t \in \mathcal{H}_{t-1}$ arbitrarily and predicts $\hat{y}_t = f_t(x_t)$;
3. Nature reveals true label $y_t = f^*(x_t)$ and Learner updates $\mathcal{H}_t = \{f \in \mathcal{H}_{t-1} : f(x_t) = y_t\}$.

Note that the number of mistakes does not depend on the learning horizon T . For any length of T , the number of mistakes is at most $|\mathcal{H}| - 1$.

Now, we exam how many mistakes (the total loss $\sum_{t=1}^T \ell(y_t, \hat{y}_t) = \sum_{t=1}^T \ell(y_t, f_t(x_t))$) that **Consistent Algorithm** makes by the end of round T . It is not hard to see that the number of mistakes is at most $|\mathcal{H}| - 1$, as each mistake forces Learner to eliminate at least one hypothesis from the version space at the end of that round.

$f(x) = o(g(x))$ is equivalent to $\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = 0$ if $g(x) > 0$.

One may ask is it possible to reduce the number of mistakes from $|\mathcal{H}|$ to $o(|\mathcal{H}|)$? The answer is yes!

Actually, we can do better if we do not pick f_t in an arbitrary way! We introduce another idea, called *majority vote*. Since some hypotheses in \mathcal{H}_{t-1} predict +1 while some predict -1, we simply count which side has more supporters. The side of more supporters wins! Combining majority vote with version space, we have a much nicer algorithm, called **Halving Algorithm**, which reduces the number of mistakes from $\mathcal{O}(|\mathcal{H}|)$ to $\log |\mathcal{H}|$.

Initially, we set $\mathcal{H}_0 = \mathcal{H}$.

In round $t = 1, 2, \dots, T$,

1. Learner receives x_t ;
2. Learner predicts $\hat{y}_t \in \arg \max_{y \in \{-1, 1\}} |\{f \in \mathcal{H}_{t-1} : f(x_t) = y\}|$;
3. Nature reveals true label $y_t = f^*(x_t)$ and Learner updates $\mathcal{H}_t = \{f \in \mathcal{H}_{t-1} : f(x_t) = y_t\}$.

THEOREM 7.1. *Halving Algorithm makes at most $\log_2(|\mathcal{H}|)$ mistakes.*

¹Nature is regulated to play a label function that is consistent with the history data S_{t-1} .

Proof. According to the majority vote, the version space is halved on each mistake. If Learner makes a mistake in round t , we have $|\mathcal{H}_t| \leq \frac{1}{2} |\mathcal{H}_{t-1}|$. Let M be the total number of mistakes. We have

$$1 \leq |\mathcal{H}_T| \leq |\mathcal{H}_0| 2^{-M} = |\mathcal{H}| 2^{-M} \quad ,$$

which yields $M \leq \log_2(|\mathcal{H}|)$. Note that f^* is always in the version spaces. So, we have $1 \leq |\mathcal{H}_T|$. \square

GENERAL HYPOTHESIS CLASS INCLUDING \mathcal{H} IS INFINITE. Still under the assumption of realizability, now, let us consider a hypothesis class \mathcal{H} that may be infinite in size. For some hypothesis class \mathcal{H} , Learner is able to have a strategy that guarantees it makes a finite number of mistakes. For other hypothesis classes, Nature can force Learner to make infinitely many mistakes. To gain an understanding of which hypothesis class falls in the former category and which falls in the latter, we need to have new concepts, something like VC dimension to measure the richness of a hypothesis class. We aim at characterizing online learnability. In particular, we target the following question: What is the optimal online classification learning algorithm for a given hypothesis class \mathcal{H} ?

Littlestone dimension characterizes online learnability. A hypothesis \mathcal{H} is online learnable if it has a finite Littlestone dimension denoted as $Ldim(\mathcal{H})$. The idea of Littlestone dimension is to view online learning as a 2-player sequential game between Learner and Adversary. The job of Adversary is to force Learner to make mistakes while preserving realizability.

How does Adversary choose x_t to force Learner to make the maximum number of mistakes, while ensuring realizability? It is easy as Adversary can always choose $y_t = -\hat{y}_t$ for the first $Ldim(\mathcal{H})$ rounds in the sequential game.

The strategy for Adversary can be formally described by using a complete binary tree. Each node of the tree is associated with an instance $x_t \in \mathcal{X}$. If Learner predicts $\hat{y}_t = +1$, Adversary will declare the prediction is wrong, i.e., $y_t = -\hat{y}_t = -1$ and will traverse to the right child of the current node. If Learner predicts $\hat{y}_t = -1$, Adversary will set $y_t = -\hat{y}_t = +1$ and will traverse to the left child of the current node.

Add a picture of the binary tree.

To introduce the definition of Littlestone dimension, let us give the definition of an \mathcal{H} shattered tree first.

DEFINITION 7.2. (\mathcal{H} shattered tree.) A shattered tree of depth d is a sequence of inputs $v_1, v_2, \dots, v_{2^d-1} \in \mathcal{X}$ such that for every root-to-leaf path, $\exists f^* \in \mathcal{H}$ such that all labels along the path are achieved.

The depth of the tree is defined as the number of edges in a path from the root to a leaf, i.e., the number of layers in the tree.

DEFINITION 7.3. (Littlestone dimension.) The Littlestone dimension of hypothesis class \mathcal{H} , $Ldim(\mathcal{H})$, is the maximal integer d such that there exists a shattered tree of depth d that \mathcal{H} shatters. If there is no such largest d , then the Littlestone dimension is infinite.

THEOREM 7.4. Any algorithm makes at least $Ldim(\mathcal{H})$ mistakes.

Proof. Let $d_* = Ldim(\mathcal{H})$. Since \mathcal{H} has Littlestone dimension d_* , we know there exists an \mathcal{H} shattered tree with depth d_* that is shattered by \mathcal{H} . Adversary will “walk” on the tree for the first d_* rounds. In each round $t \in [d_*]$, Adversary sets $y_t = -\hat{y}_t$. Since the walk continues for d_* rounds, Learner makes d_* mistakes. Now, we check the

assumption of realizability. Since \mathcal{H} shatters the tree, there $\exists f_* \in \mathcal{H}$ such that all the labels along the root-to-leaf path selected by Adversary can be realized. \square

Now, we show a learning algorithm, **Standard Optimal Algorithm (SOA)**, makes at most $\text{Ldim}(\mathcal{H})$ mistakes. The algorithm is similar to **Halving Algorithm**. We partition the version space by the end of round $t - 1$ into two sub-version spaces. Let $\mathcal{H}_{t-1}^{(-1)} := \{f \in \mathcal{H}_{t-1} : f(x_t) = -1\}$ and $\mathcal{H}_{t-1}^{(+1)} := \{f \in \mathcal{H}_{t-1} : f(x_t) = +1\}$.

Initially, we set $\mathcal{H}_0 = \mathcal{H}$.

In round $t = 1, 2, \dots, T$,

1. Learner receives x_t ;
2. Learner predicts $\hat{y}_t \in \arg \max_{y \in \{-1, 1\}} \text{Ldim}(\mathcal{H}_{t-1}^{(y)})$;
3. Nature reveals true label $y_t = f^*(x_t)$ and Learner updates $\mathcal{H}_t = \{f \in \mathcal{H}_{t-1} : f(x_t) = y_t\}$.

THEOREM 7.5. *SOA makes at most $\text{Ldim}(\mathcal{H})$ mistakes.*

Proof. We claim that whenever Learner makes a mistake in some round t , we have $\text{Ldim}(\mathcal{H}_t) \leq \text{Ldim}(\mathcal{H}_{t-1}) - 1$ at the end of that round. Without loss of generality, we assume Learner predicts -1 , but the true label is $+1$ in that round.

We use contradiction to complete the proof of the claim. Suppose $\text{Ldim}(\mathcal{H}_t) = \text{Ldim}(\mathcal{H}_{t-1})$.

We have the following inequalities.

1. $\text{Ldim}(\mathcal{H}_{t-1}^{(-1)}) \leq \text{Ldim}(\mathcal{H}_{t-1})$;
2. $\text{Ldim}(\mathcal{H}_t) = \text{Ldim}(\mathcal{H}_{t-1}^{(+1)})$;
3. $\text{Ldim}(\mathcal{H}_{t-1}^{(-1)}) \geq \text{Ldim}(\mathcal{H}_{t-1}^{(+1)})$.

From inequalities 2 and 3 above, we have $\text{Ldim}(\mathcal{H}_{t-1}^{(-1)}) \geq \text{Ldim}(\mathcal{H}_t) = \text{Ldim}(\mathcal{H}_{t-1})$, where the equality uses the assumption that $\text{Ldim}(\mathcal{H}_t) = \text{Ldim}(\mathcal{H}_{t-1})$. Combining with inequality 1, we have

$$\text{Ldim}(\mathcal{H}_{t-1}^{(-1)}) = \text{Ldim}(\mathcal{H}_{t-1}) = \text{Ldim}(\mathcal{H}_t) = \text{Ldim}(\mathcal{H}_{t-1}^{(+1)}).$$

Since $\text{Ldim}(\mathcal{H}_{t-1}^{(-1)}) = \text{Ldim}(\mathcal{H}_{t-1}^{(+1)})$, we can construct a shattered tree of depth $\text{Ldim}(\mathcal{H}_{t-1}) + 1$ that can be shattered by \mathcal{H}_{t-1} , which yields a contradiction. \square

COROLLARY 7.6. *Let \mathcal{H} be any hypothesis class. Then, SOA makes exactly $\text{Ldim}(\mathcal{H})$ mistakes.*

Proof. From Theorem 7.4, we know SOA makes at least $\text{Ldim}(\mathcal{H})$ mistakes. From Theorem 7.5, we know SOA makes at most $\text{Ldim}(\mathcal{H})$ mistakes. Combining these two results concludes the proof. \square

VC DIMENSION VS LITTLESTONE DIMENSION.

EXAMPLE 1. If \mathcal{H} is a finite hypothesis class, we have $\text{Ldim}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$. **why?**

EXAMPLE 2. Let $\mathcal{X} = [0, 1]$ and $\mathcal{H} = \{x \mapsto \mathbf{1}\{x < a\} : a \in [0, 1]\}$ be the class of thresholds on the interval $[0, 1]$. Then, we have $\text{VCdim}(\mathcal{H}) = 1$, but $\text{Ldim}(\mathcal{H}) = \infty$.

There is a picture for this.

THEOREM 7.7. For any hypothesis class \mathcal{H} , we have $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$. Further, the gap can be arbitrarily large.

Proof. We first prove that $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$. Suppose $\text{VCdim}(\mathcal{H}) = d$ and let $\{x_1, x_2, \dots, x_d\}$ be a shattered set by \mathcal{H} . Now, we construct a complete binary tree of inputs $v_1, v_2, \dots, v_{2^d-1}$, where all nodes at depth i are set to be x_i .

There is a picture for this.

Now, the definition of a shattered set clearly implies that we constructed a valid shattered tree of depth d and conclude that $\text{VCdim}(\mathcal{H}) \leq \text{Ldim}(\mathcal{H})$.

The class of threshold functions on the unit interval has VC dimension of 1, whereas its Littlestone dimension is infinite. \square

7.2 DECISION-THEORETIC ONLINE LEARNING AND EXPONENTIAL WEIGHTS (HEDGE)

Practically, we should not assume realizability always holds, i.e., the true labels $y_t = f^*(x_t), \forall t \in [T]$, are generated by using f^* . Similar to agnostic setting in supervised learning, now, we can compare with the best predictor in \mathcal{H} . In online learning, we use the notion of regret, defined as

$$\mathcal{R}(T) := \sup_{((x_1, y_1), \dots, (x_T, y_T))} \left\{ \sum_{t=1}^T \ell(y_t, f_t(x_t)) - \min_{f \in \mathcal{H}} \sum_{t=1}^T \ell(y_t, f(x_t)) \right\}, \quad (7.1)$$

to measure the performance gap between $\sum_{t=1}^T \ell(y_t, f_t(x_t))$, the total loss Learner has made, and $\min_{f \in \mathcal{H}} \sum_{t=1}^T \ell(y_t, f(x_t))$, the total loss of the best predictor in hindsight. If a learning algorithm achieves an $o(T)$ regret bound, we say it is a *no-regret* algorithm. The intuition is if you play this game long enough, you can compete with the best predictor in hindsight. Note that $o(T)$ regret implies $\lim_{T \rightarrow \infty} \frac{o(T)}{T} = 0$.

To control the regret at Learner's side, generally, we have two key principles.

1. *Randomization.* Deterministic Learner fails for some learning problems. Let hypothesis class $\mathcal{H} = \{f_+ : x \rightarrow +1, f_- : x \rightarrow -1\}$ only contain two constant functions. Adversary can always give Learner $y_t = -\hat{y}_t$ to force Learner to suffer loss. So, Learner suffers in total T loss. Now, we investigate the total loss in hindsight of the best predictor in \mathcal{H} , that is, $\min_{f \in \mathcal{H}} \sum_{t=1}^T \ell(y_t, f(x_t)) \leq \frac{1}{2} \cdot \left(\sum_{t=1}^T \ell(y_t, f_-(x_t)) + \sum_{t=1}^T \ell(y_t, f_+(x_t)) \right) = 0.5T$, which yields the regret is at least $0.5T$.

Since Learner always fails if revealing the predicted label \hat{y}_t , to make the problem interesting, we need to give some power to Learner. Now, we change the learning protocol a bit by allowing Learner to reveal only a probability distribution over $\{-1, 1\}$ instead of revealing the predicted label \hat{y}_t itself.

2. *Exploitation.* We want to track the empirical performance of each predictor, but we do not eliminate any of them during the learning, as some of them may not perform well at the beginning of the learning, but later turns out to be the best one.

Since we do not plan to eliminate predictors during the learning, we can maintain a (*data-dependent*) *distribution* over all the predictors in \mathcal{H} taking account of each predictor's total loss suffered so far. For some predictor performing poorly, we put a small mass on it. We eliminate some predictor in a soft way!

DECISION-THEORETIC ONLINE LEARNING (DTOL). Since Learner is allowed to reveal a probability distribution over the labels and revealing a distribution over $\{-1, 1\}$ can be translated to revealing a distribution over \mathcal{H} , now, we can re-formulate the online learning problem slightly and the modification will be, generally, useful for bandit problems (and even reinforcement learning problems). We do not use training samples at all. We get rid of the input space, label space, and hypothesis class. Instead, we have a fixed set of K actions, denoted by $[K]$. The learning protocol

Here, exploitation refers to utilizing the already learned information, i.e., the total loss of each predictor observed so far.

If the data sequence are i.i.d. according to some fixed but unknown distribution, we can eliminate predictors.

You can view each action in $[K]$ as a hypothesis in \mathcal{H} .

is modified as follows.

In each round $t = 1, 2, \dots, T$,

1. Learner plays a probability distribution $p_t \in [0, 1]^K$ over all actions ;
2. Nature plays loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \dots, \ell_{K,t})$;
3. Learner observes ℓ_t and suffer a loss $\langle p_t, \ell_t \rangle = \sum_j p_{j,t} \ell_{j,t}$.

Sometimes, Learner chooses $p_t \in \{e_1, e_2, \dots, e_K\}$.

Remarks. (1) A probability distribution $p_t = (p_{1,t}, p_{2,t}, \dots, p_{K,t})$ is a vector with all $p_{j,t} \geq 0$ and $\sum_j p_{j,t} = 1$. (2) In Step 2, Nature can give ℓ_t based on all the past information and even p_t .

For DTOL, we adapt the notion of regret shown in (7.1) to

$$\mathcal{R}(T) := \sup_{\ell_1, \ell_2, \dots, \ell_T} \left\{ \sum_{t=1}^T \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{t=1}^T \ell_{j,t} \right\} . \quad (7.2) \quad \ell_{j,t} = \langle e_j, \ell_t \rangle.$$

EXPONENTIAL WEIGHTS (HEDGE). There is a no-regret algorithm for DTOL. It has many names such as ‘‘Exponential Weights’’ and ‘‘Hedge’’. The idea of Hedge is to maintain weights over actions, and the weight of an action decays exponentially in the total loss incurred by that action over all previous rounds.

Input: learning rate $\eta \in (0, 1]$.

Initialize: $w_{j,0} = 1$ for all $j \in [K]$.

For $t = 1, 2, \dots, T$,

1. Set $p_{j,t} = \frac{w_{j,t-1}}{\sum_{j'} w_{j',t-1}}$ for all $j \in [K]$;
2. Observe loss vector ℓ_t for Nature ;
3. Suffer loss $\langle p_t, \ell_t \rangle = \sum_j p_{j,t} \ell_{j,t}$;
4. Update $w_{j,t} = w_{j,t-1} \cdot e^{-\eta \ell_{j,t}}$ for all $j \in [K]$.

THEOREM 7.8. For any sequence of loss vectors $(\ell_1, \ell_2, \dots, \ell_T) \in ([0, 1]^K)^T$, for any $\eta \in (0, 1]$, we have

$$\sum_{t=1}^T \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{t=1}^T \ell_{j,t} \leq \frac{\eta}{2} \sum_{t=1}^T \langle p_t, \ell_t^2 \rangle + \frac{\log K}{\eta} .$$

With $\eta = \sqrt{\frac{2 \log K}{T}}$, we have $\mathcal{R}(T) \leq \sqrt{2T \log K}$.

Proof. of Theorem 7.8: Fix a sequence of loss vectors $(\ell_1, \ell_2, \dots, \ell_T)$. Let $Z_t =$

$\sum_{j=1}^K w_{j,t} = \sum_{j=1}^K w_{j,t-1} \cdot e^{-\eta \ell_{j,t}} = \sum_{j=1}^K e^{-\eta \sum_{s=1}^t \ell_{j,s}}$ be the total weight at the end of round t .

Note that $Z_0 = K$. We have $\log Z_T = \sum_{t=1}^T \log \frac{Z_t}{Z_{t-1}} + \log Z_0$.

Now, we construct the following upper bound to upper bound $\log Z_T$. We have

$$\begin{aligned}
\log \frac{Z_t}{Z_{t-1}} &= \log \frac{\sum_{j=1}^K w_{j,t-1} \cdot e^{-\eta \ell_{j,t}}}{\sum_{j'=1}^K w_{j',t-1}} \\
&= \log \left(\sum_{j=1}^K \frac{w_{j,t-1}}{\sum_{j'=1}^K w_{j',t-1}} \cdot e^{-\eta \ell_{j,t}} \right) \\
&= \log \left(\sum_{j=1}^K p_{j,t} \cdot e^{-\eta \ell_{j,t}} \right) \\
&\stackrel{(a)}{\leq} \log \left(\sum_{j=1}^K p_{j,t} \cdot \left(1 - \eta \cdot \ell_{j,t} + \frac{\eta^2 \cdot \ell_{j,t}^2}{2} \right) \right) \\
&= \log \left(1 - \eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \langle p_t, \ell_t^2 \rangle \right) \\
&\stackrel{(b)}{\leq} -\eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \langle p_t, \ell_t^2 \rangle,
\end{aligned}$$

where step (a) uses $e^{-x} \leq 1 - x + x^2/2$ and step (b) uses $\log(1 + x) \leq x$.

We also have a lower bound on $\log Z_T$, which is

$$\begin{aligned}
\log Z_T &= \log \left(\sum_{j=1}^K e^{-\eta \sum_{t=1}^T \ell_{j,t}} \right) \\
&\geq \log \left(\max_{j \in [K]} \left\{ e^{-\eta \sum_{t=1}^T \ell_{j,t}} \right\} \right) \\
&= \max_{j \in [K]} \left\{ -\eta \sum_{t=1}^T \ell_{j,t} \right\}.
\end{aligned} \tag{7.3}$$

Now, we have

$$\begin{aligned}
\sum_{t=1}^T -\eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \langle p_t, \ell_t^2 \rangle &\geq \sum_{t=1}^T \log \frac{Z_t}{Z_{t-1}} = \log Z_T - \log Z_0 \geq \max_{j \in [K]} \left\{ -\eta \sum_{t=1}^T \ell_{j,t} \right\} - \log K. \\
\Rightarrow \sum_{t=1}^T \langle p_t, \ell_t \rangle - \min_{j \in [K]} \left\{ \sum_{t=1}^T \ell_{j,t} \right\} &\leq \frac{\eta}{2} \langle p_t, \ell_t^2 \rangle + \frac{\log K}{\eta}.
\end{aligned} \tag{7.4}$$

□

Remark. Note that in order to achieve the $\sqrt{2T \log K}$ regret bound, **Hedge** needs to input the learning rate $\eta = \sqrt{\frac{2 \log K}{T}}$, depending on the learning horizon T . Later, we will show how to use doubling-trick to get rid of it!

HEDGE WITH DOUBLING-TRICK. The regret bound $\mathcal{R}(T) = \mathcal{O}(\sqrt{T \log K})$ in Theorem 7.8 relies on inputting the learning rate $\eta = \sqrt{\frac{2 \log K}{T}}$ which relies on the knowledge of the time horizon T .

One may be curious to know is it possible to have a learning algorithm that does not need to know T in advance, but still preserving the same regret bound?

The answer is yes!!!

We introduce a useful idea, called *doubling-trick*, for solving online learning problems. Using (*Geometric*) *doubling-trick*, we can still achieve an $\mathcal{O}(\sqrt{T \log K})$ regret

bound. The idea is to run the algorithm in epochs of lengths $2^0, 2^1, \dots, 2^r, \dots$ until stopping.

At the beginning of each epoch $r \geq 0$, we set the learning rate $\eta_r = \sqrt{\frac{2 \log K}{2^r}}$ based on 2^r the length of the current epoch. At the end of epoch r , we reset the algorithm, that is, we forget all the stuff we have learned. Progressing in this way, we will run the algorithm for epochs $r = 0, 1, \dots, d$, where $d = \lceil \log_2(T + 1) - 1 \rceil = \mathcal{O}(\log T)$.

THEOREM 7.9. *The regret of Hedge with doubling-trick is $\mathcal{O}(\sqrt{T \log K})$.*

Proof. For any $r \geq 0$, from (7.8), we have

$$\sum_{t=2^{r+1}}^{2^{r+1}} \langle p_t, \ell_t \rangle - \min_{j_r \in [K]} \sum_{t=2^{r+1}}^{2^{r+1}} \ell_{j_r, t} \leq \frac{\eta_r}{2} \sum_{t=2^{r+1}}^{2^{r+1}} \langle p_t, \ell_t^2 \rangle + \frac{\log K}{\eta_r} \leq \mathcal{O}(\sqrt{2^r \log K}) . \quad (7.5)$$

The regret is

$$\begin{aligned} \mathcal{R}(T) &= \sup_{\ell_1, \ell_2, \dots, \ell_T} \left\{ \sum_{t=1}^T \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{t=1}^T \ell_{j, t} \right\} \\ &= \sup_{\ell_1, \ell_2, \dots, \ell_T} \left\{ \sum_{r \geq 0} \sum_{t=2^{r+1}}^{2^{r+1}} \langle p_t, \ell_t \rangle - \min_{j \in [K]} \sum_{r \geq 0} \sum_{t=2^{r+1}}^{2^{r+1}} \ell_{j, t} \right\} \\ &\leq \sup_{\ell_1, \ell_2, \dots, \ell_T} \left\{ \sum_{r \geq 0} \sum_{t=2^{r+1}}^{2^{r+1}} \langle p_t, \ell_t \rangle - \sum_{r \geq 0} \min_{j_r \in [K]} \sum_{t=2^{r+1}}^{2^{r+1}} \ell_{j_r, t} \right\} \\ &= \sup_{\ell_1, \ell_2, \dots, \ell_T} \left\{ \sum_{r \geq 0} \left(\sum_{t=2^{r+1}}^{2^{r+1}} \langle p_t, \ell_t \rangle - \min_{j_r \in [K]} \sum_{t=2^{r+1}}^{2^{r+1}} \ell_{j_r, t} \right) \right\} \\ &= \sup_{\ell_1, \ell_2, \dots, \ell_T} \left\{ \sum_{r \geq 0} \mathcal{O}(\sqrt{2^r \log K}) \right\} \\ &= \sum_{r \geq 0} \mathcal{O}(\sqrt{2^r \log K}) \\ &= \mathcal{O}(\sqrt{T \log K}) . \end{aligned} \quad (7.6)$$

The last epoch may not have a full length, but we can add $\vec{0}$'s to make the last epoch have a full length

□

7.3 BANDITS

Last time we have talked about Hedge/DTOL learning. We usually say it is a full information game as Learner is able to observe each individual loss in $\ell_t = (\ell_{1,t}, \ell_{2,t}, \dots, \ell_{K,t})$. Starting from this lecture, we will talk about multi-armed bandit (MAB) problems, where only some entry in the loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \dots, \ell_{K,t})$ is revealed in each round.

Actions and arms are interchangeable. You can view an arm as a hypothesis.

You can view Step 1 and Step 2 occur simultaneously.

In Step 3, the remaining losses other than $\ell_{j,t}$ are still hidden to Learner, which is the key difference between DTOL and bandits.

LEARNING PROTOCOL. We have a fixed arm set $[K]$.

In each round $t = 1, 2, \dots, T$,

1. Adversary/Environment selects a loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \dots, \ell_{K,t})$ that is hidden to Learner ;
2. Learner pulls an arm $J_t \in [K]$;
3. Learner suffers/observes loss $\ell_{J_t,t}$ associated with the pulled arm J_t .

The goal of Learner to pull a sequence of arms (J_1, J_2, \dots, J_T) to minimize the total loss by the end of round T .

Regret is defined as

$$\mathcal{R}(T) := \sum_{t=1}^T \ell_{J_t,t} - \min_{j \in [K]} \sum_{t=1}^T \ell_{j,t} \quad , \quad (7.7)$$

which is a random variable as J_1, J_2, \dots, J_T and all loss vectors are random.

Based on how the loss vectors $(\ell_1, \ell_2, \dots, \ell_T)$ are generated, we have

1. Adversary bandits: no distributional assumption is made.
2. Stochastic bandits: all ℓ_t are i.i.d. over time according to a fixed but unknown probability distribution.

Learner needs to make a good balance between

1. *Exploitation*: Learner needs to pull arms that have smaller losses, as the goal is to minimize the total loss.
2. *Exploration*: Learner needs to pull arms that have not been observed too often to gain information.

7.3.1 Adversarial bandits.

In adversarial bandits, the loss vectors can be generated adversarially. We use *pseudo regret* to measure performance of the algorithm used by Learner, defined as

$$\bar{\mathcal{R}}(T) := \mathbb{E} \left[\sum_{t=1}^T \ell_{J_t,t} \right] - \min_{j \in [K]} \mathbb{E} \left[\sum_{t=1}^T \ell_{j,t} \right] \quad , \quad (7.8)$$

Sometimes people call it expected regret as it has an expectation. I am following the notation of Bubeck and Cesa-Bianchi [BC12].

where the expectation is taken over (J_1, J_2, \dots, J_T) and all loss vectors over T rounds.

EXP3 (EXPONENTIAL WEIGHTS FOR EXPLORATION AND EXPLOITATION). It is quite similar to Hedge, but in EXP3, only the weight associated with the pulled arm will be updated at the end of each round t , as only the loss for that arm is revealed.

Importance sampling We construct an estimator $\tilde{\ell}_t = (\tilde{\ell}_{1,t}, \tilde{\ell}_{2,t}, \dots, \tilde{\ell}_{K,t})$ with each entry $\tilde{\ell}_{j,t} = \ell_{j,t} \frac{\mathbf{1}\{J_t=j\}}{p_{j,t}}$.

It is not hard to see that for all arms not played in round t , we set $\tilde{\ell}_{j,t} = 0$. For the pulled arm in round t , we set $\tilde{\ell}_{j,t} = \frac{\ell_{j,t}}{p_{j,t}}$ instead of $\ell_{j,t}$ by making it more “important”. Actually, the constructed estimator $\tilde{\ell}_{j,t}$ is an unbiased estimator of $\ell_{j,t}$, as we have

$$\mathbb{E}_{J_t \sim p_t} [\tilde{\ell}_{j,t}] = \mathbb{E}_{J_t \sim p_t} \left[\ell_{j,t} \frac{\mathbf{1}\{J_t=j\}}{p_{j,t}} \right] = \ell_{j,t} \quad . \quad (7.9)$$

Input: learning rate $\eta \in (0, 1]$. Initialize: $w_{j,0} = 1$ for all $j \in [K]$.

For $t = 1, 2, \dots, T$,

1. Adversary/Environment selects a loss vector $\ell_t = (\ell_{1,t}, \ell_{2,t}, \dots, \ell_{K,t})$ that is hidden to Learner ;
2. Learner computes $p_{j,t} = \frac{w_{j,t-1}}{\sum_{j'} w_{j',t-1}}$ for all $j \in [K]$;
3. Learner plays arm $J_t \in [K]$ according to $p_t = (p_{1,t}, p_{2,t}, \dots, p_{K,t})$;
4. Learner computes loss estimates $\tilde{\ell}_{j,t} = \frac{\ell_{j,t}}{p_{j,t}} \mathbf{1}\{J_t = j\}$ for all $j \in [K]$;
5. Update $w_{j,t} = w_{j,t-1} \cdot e^{-\eta \tilde{\ell}_{j,t}}$ for all $j \in [K]$.

THEOREM 7.10. Assume that all loss vectors are bounded with $[0, 1]$ support. If EXP3 is run with learning rate $\eta = \sqrt{\frac{\log K}{KT}}$, the pseudo regret is at most $\sqrt{2TK \log K}$.

Proof.

$$\begin{aligned} \bar{\mathcal{R}}(T) &= \mathbb{E} \left[\sum_{t=1}^T \ell_{J_t,t} \right] - \min_{j \in [K]} \mathbb{E} \left[\sum_{t=1}^T \ell_{j,t} \right] \\ &= \mathbb{E} \left[\sum_{t=1}^T \langle p_t, \tilde{\ell}_t \rangle \right] - \min_{j \in [K]} \mathbb{E} \left[\sum_{t=1}^T \mathbb{E}_{J_t \sim p_t} [\tilde{\ell}_{j,t}] \right] \\ &= \mathbb{E} \left[\sum_{t=1}^T \langle p_t, \tilde{\ell}_t \rangle \right] - \min_{j \in [K]} \mathbb{E} \left[\sum_{t=1}^T \tilde{\ell}_{j,t} \right] \\ &\leq \mathbb{E} \left[\sum_{t=1}^T \langle p_t, \tilde{\ell}_t \rangle \right] - \mathbb{E} \left[\min_{j \in [K]} \sum_{t=1}^T \tilde{\ell}_{j,t} \right] \\ &= \underbrace{\mathbb{E} \left[\sum_{t=1}^T \langle p_t, \tilde{\ell}_t \rangle - \min_{j \in [K]} \sum_{t=1}^T \tilde{\ell}_{j,t} \right]}_{\text{regret of Hedge}} \\ &\stackrel{(a)}{\leq} \mathbb{E} \left[\frac{\eta}{2} \sum_{t=1}^T \langle p_t, \tilde{\ell}_t^2 \rangle + \frac{\log K}{\eta} \right] \\ &= \stackrel{(b)}{\mathbb{E}} \left[\frac{\eta}{2} \sum_{t=1}^T \sum_{j \in [K]} p_{j,t} \frac{\ell_{j,t}^2}{p_{j,t}} \right] + \frac{\log K}{\eta} \\ &\leq \frac{\eta}{2} KT + \frac{\log K}{\eta}. \end{aligned} \quad (7.10)$$

Tuning $\eta = \sqrt{\frac{\log K}{KT}}$ gives the stated regret bound.

Step (a) uses Theorem 7.8 in previous lecture. Note that from (7.9), we know $\mathbb{E}_{J_t \sim p_t} [\tilde{\ell}_{j,t}] \in [0, 1]$. So, it is safe to use Theorem 7.8 directly.

Step (b) uses $\mathbb{E}_{J_t \sim p_t} [\tilde{\ell}_{j,t}^2] = \mathbb{E}_{J_t \sim p_t} \left[\ell_{j,t}^2 \frac{\mathbf{1}\{J_t=j\}}{p_{j,t}^2} \right] = \frac{\ell_{j,t}^2}{p_{j,t}}$. □

REMARK. Note that inputting $\eta = \sqrt{\frac{\log K}{KT}}$ into EXP3 means that it is not an anytime learning algorithm. To make it anytime, you can set the learning rate $\eta_t = \sqrt{\frac{\log K}{tK}}$ in each round t . The regret analysis is much more complicated (refer to Theorem 3.1 in [BC12]).

7.3.2 Stochastic bandits

In stochastic bandits, we have a fixed arm set $[K]$ and each arm $j \in [K]$ is associated with a reward distribution v_j . We can use $\Theta_K := (v_1, v_2, \dots, v_K)$ to specify a K -armed stochastic bandit problem instance.

In each round $t = 1, 2, \dots, T$,

1. Environment generate a reward vector $X_t = (X_{1,t}, X_{2,t}, \dots, X_{K,t})$ with each $X_{j,t} \sim v_j$. This reward vector is hidden to Learner ;
2. Learner pulls an arm $J_t \in [K]$;
3. Learner obtains/observes $X_{J_t,t}$, the reward of the pulled arm J_t .

We still use pseudo regret to measure performance of **Alg** , defined as

$$\bar{\mathcal{R}}(\mathbf{Alg}; \Theta_K; T) := \max_{j \in [K]} \mathbb{E} \left[\sum_{t=1}^T X_{j,t} \right] - \mathbb{E} \left[\sum_{t=1}^T X_{J_t,t} \right] , \quad (7.11)$$

where the randomness is taken over J_1, J_2, \dots, J_T and all reward vectors. Note that since we have statistical assumptions on reward vectors, different Θ_K may give different regret. That is also to say, for a fixed algorithm, when working over different problem instances, the regret could be different.

The goal of Learner is to pull arms sequentially to minimize regret.

Let $\mu_j = \mathbb{E}_{X_j \sim v_j}[X_j]$ denote the mean reward of arm j . Without loss of generality, we assume the first arm is the optimal one, that is, $\mu_1 > \mu_j$ for all $j \neq 1$.

For any $j \neq 1$, let $\Delta_j := \mu_1 - \mu_j$ denote the mean reward gap. We also call it the sub-optimality gap between the optimal arm 1 and the sub-optimal arm j . Let $\Delta_1 = 0$.

Now, we can rewrite $\bar{\mathcal{R}}(T)$ as

$$\begin{aligned} \bar{\mathcal{R}}(T) &= \max_{j \in [K]} \mathbb{E} \left[\sum_{t=1}^T X_{j,t} \right] - \mathbb{E} \left[\sum_{t=1}^T X_{J_t,t} \right] \\ &= T \cdot \mu_1 - \sum_{t=1}^T \mathbb{E} [\mu_{J_t}] \\ &= \sum_{t=1}^T \mathbb{E} [\mu_1 - \mu_{J_t}] \\ &= \sum_{t=1}^T \mathbb{E} [\Delta_{J_t}] \\ &= \sum_{t=1}^T \sum_{j \in [K]} \mathbb{E} [\mathbf{1}\{J_t = j\}] \cdot \Delta_j \\ &= \sum_{j \in [K]: \Delta_j > 0} \mathbb{E} \left[\underbrace{\sum_{t=1}^T \mathbf{1}\{J_t = j\}}_{=: n_{j,T}} \right] \cdot \Delta_j . \end{aligned} \quad (7.12)$$

Let $n_{j,t-1} := \sum_{s=1}^{t-1} \mathbf{1}\{J_s = j\}$ denote the total number of pulls for arm j by the end of round $t - 1$. Then, $\mathbb{E}[n_{j,T}]$ is the expected number of pulls of arm j by the end of learning and Δ_j is the single round performance loss when pulling a sub-optimal arm j .

From the last step in (7.12), it is not hard to see, to minimize the regret, it is important to control the number of pulls of sub-optimal arms. But we have no idea which arms are sub-optimal. So, we have to pull each arm a certain amount of times in order to learn whether they are sub-optimal or not confidently. We need information!

Here, I should mention exploitation-vs-exploration.

UPPER CONFIDENCE BOUND (UCB). It is inspired by the principle of being optimistic in the face of uncertainty. Usually, all UCB-based algorithms are optimistic learning algorithms and follow a template to decompose regret. Also, they can be justified by concentration inequalities, e.g., Hoeffding's inequality.

Recall $\Theta_K := (v_1, v_2, \dots, v_K)$ is the bandit instance we are interested in. Actually, for developing regret minimization algorithm, we are interested in $(\mu_1, \mu_2, \dots, \mu_K)$. Note that only the mean reward gaps appear in the regret.

Recall $n_{j,t-1} = \sum_{s=1}^{t-1} \mathbf{1}\{J_s = j\}$ is the number of pulls of arm j by the end of round $t - 1$.

Now, let $\hat{\mu}_{j,n_{j,t-1}} := \frac{1}{n_{j,t-1}} \sum_{s=1}^{t-1} X_{j,s} \mathbf{1}\{J_s = j\}$ be the empirical mean of arm j by the end of round $t - 1$, i.e., the average of $n_{j,t-1}$ iid random variables according to v_j .

Now, we can construct an empirical model $\hat{\Theta}_t = (\hat{\mu}_{1,n_{1,t-1}}, \hat{\mu}_{2,n_{2,t-1}}, \dots, \hat{\mu}_{K,n_{K,t-1}})$.

exploitation-vs-exploration

Assume all reward distributions have a $[0, 1]$ support. The idea of UCB1², an algorithm in UCB family, is to construct an optimistic model $\bar{\Theta}_t = (\bar{\mu}_{1,t}, \bar{\mu}_{2,t}, \dots, \bar{\mu}_{K,t})$ with each $j \in [K]$

Hoeffding's inequality

upper confidence bound

w.t.p. $\bar{\mu}_{1,t} \geq \mu_1$

$$\bar{\mu}_{j,t} = \hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2 \ln(t)}{n_{j,t-1}}} . \quad (7.13)$$

In each round $t = 1, 2, \dots, T$,

1. Environment generate a reward vector $X_t = (X_{1,t}, X_{2,t}, \dots, X_{K,t})$ with each $X_{j,t} \sim v_j$. This reward vector is hidden to Learner ;
2. Learner constructs the upper confidence bound $\bar{\mu}_{j,t} = \hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2 \ln(t)}{n_{j,t-1}}}$ for all $j \in [K]$;
3. Learner pulls the arm with the highest upper confidence bound, i.e., $J_t \in \arg \max_{j \in [K]} \bar{\mu}_{j,t}$;
4. Learner observes $X_{J_t,t}$, the reward of the pulled arm J_t ;
5. Learner updates $n_{J_t,t} = n_{J_t,t-1} + 1$ and the empirical mean $\hat{\mu}_{J_t,n_{J_t,t}}$.

²UCB1 works for all Sub-Gaussian reward distributions.

THEOREM 7.11. *If all reward distributions in Θ_K have a $[0, 1]$ support, we have*

$$\overline{\mathcal{R}}(\text{UCB1}; \Theta_K; T) \leq \sum_{j \in [K]: \Delta_j > 0} \frac{8 \ln T}{\Delta_j} + \text{Constant} \quad .$$

Proof. Fix a sub-optimal arm j , we upper bound $\mathbb{E}[n_{j,T}]$.

You can view L_j as the amount of observations needed to conclude that this arm is not the optimal one. We have

Let $L_j := \frac{\square \cdot \ln T}{\Delta_j^2}$, where \square is a constant that will be tuned later.

We have

$$\begin{aligned} \mathbb{E}[n_{j,T}] &= \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j\} \right] \\ &= \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j, n_{j,t-1} \leq L_j\} \right] + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j, n_{j,t-1} > L_j\} \right] \\ &= \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j, n_{j,t} > n_{j,t-1}, n_{j,t-1} \leq L_j\} \right] + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j, n_{j,t} > n_{j,t-1}, n_{j,t-1} > L_j\} \right] \\ &\leq L_j + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j, n_{j,t-1} > L_j\} \right] \\ &\leq L_j + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j, \bar{\mu}_{j,t} \geq \bar{\mu}_{1,t}, n_{j,t-1} > L_j\} \right] \\ &\leq L_j + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{J_t = j, \bar{\mu}_{j,t} \geq \mu_1, n_{j,t-1} > L_j\} \right] + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{\bar{\mu}_{1,t} \leq \mu_1, n_{j,t-1} > L_j\} \right] \\ &\leq L_j + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{\bar{\mu}_{j,t} \geq \mu_j + \Delta_j, n_{j,t-1} > L_j\} \right] + \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{\bar{\mu}_{1,t} \leq \mu_1\} \right] \quad . \end{aligned} \tag{7.14}$$

Now, we set $L_j = \frac{8 \ln(T)}{\Delta_j^2}$. Then, we have $\Delta_j = \sqrt{\frac{8 \ln T}{L_j}}$.

$$\begin{aligned} &\mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{\bar{\mu}_{j,t} \geq \mu_j + \Delta_j, n_{j,t-1} > L_j\} \right] \\ &= \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \left\{ \hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2 \ln(t)}{n_{j,t-1}}} \geq \mu_j + \sqrt{\frac{8 \ln T}{L_j}}, n_{j,t-1} > L_j \right\} \right] \\ &\leq \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \left\{ \hat{\mu}_{j,n_{j,t-1}} + \sqrt{\frac{2 \ln(T)}{n_{j,t-1}}} \geq \mu_j + \sqrt{\frac{8 \ln T}{L_j}}, n_{j,t-1} > L_j \right\} \right] \\ &\leq \mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \left\{ \hat{\mu}_{j,n_{j,t-1}} \geq \mu_j + \sqrt{\frac{2 \ln T}{n_{j,t-1}}}, n_{j,t-1} > L_j \right\} \right] \\ &\leq \sum_{t=1}^T \sum_{h=L_j}^{t-1} \mathbb{E} \left[\mathbf{1} \left\{ \hat{\mu}_{j,h} \geq \mu_j + \sqrt{\frac{2 \ln T}{h}} \right\} \right] \\ &\leq T^2 \cdot e^{-2 \cdot 2 \ln T} \\ &= \mathcal{O}(1) \quad . \end{aligned} \tag{7.15}$$

Similarly, we have $\mathbb{E} \left[\sum_{t=1}^T \mathbf{1} \{\bar{\mu}_{1,t} \leq \mu_1\} \right] = \mathcal{O}(1)$.

If problem-dependent parameters, e.g., all Δ_j , appear in the regret bound, we say it is a problem-dependent regret bound.

Now, we have $\mathbb{E}[n_{j,T}] \leq L_j + \mathcal{O}(1) = \frac{8 \ln T}{\Delta_j^2} + \mathcal{O}(1)$, which gives

$$\begin{aligned} \overline{\mathcal{R}}(\text{UCB1}; \Theta_K; T) &= \sum_{j \in [K]: \Delta_j > 0} \mathbb{E}[n_{j,T}] \cdot \Delta_j \\ &\leq \sum_{j \in [K]: \Delta_j > 0} \frac{8 \ln T}{\Delta_j} + \text{Constant} \quad . \end{aligned} \tag{7.16}$$

□

WORST-CASE REGRET BOUND FOR UCB1. Let us consider a 2-armed bandit problem, where the first arm is the optimal one and the second arm has a mean reward gap $\Delta = \frac{1}{T}$. Clearly, according to Theorem 7.11, we have $\overline{\mathcal{R}}(\text{UCB1}; \Theta_K; T) = 8T \ln T$, which is even worse than T . Does it mean UCB1 fails this learning task?

To answer this question, we are motivated to study the worst-case regret bound, defined as

$$\sup_{\Theta_K \in \Pi^K} \overline{\mathcal{R}}(\text{UCB1}; \Theta_K; T) \quad , \quad (7.17)$$

where Π is a set of distributions with a $[0, 1]$ support.

THEOREM 7.12. *We have*

$$\sup_{\Theta_K \in \Pi^K} \overline{\mathcal{R}}(\text{UCB1}; \Theta_K; T) \leq \mathcal{O}(\sqrt{KT \ln T}) \quad . \quad (7.18)$$

Proof. Fix Θ_K and set $\Delta := \sqrt{\frac{K \ln T}{T}}$. We have

$$\begin{aligned} \overline{\mathcal{R}}(\text{UCB1}; \Theta_K; T) &= \sum_{t=1}^T \sum_{j \in [K]} \mathbb{E}[\mathbf{1}\{J_t = j\}] \cdot \Delta_j \\ &= \sum_{t=1}^T \sum_{j \in [K]: \Delta_j \leq \Delta} \mathbb{E}[\mathbf{1}\{J_t = j\}] \cdot \Delta_j + \sum_{t=1}^T \sum_{j \in [K]: \Delta_j > \Delta} \mathbb{E}[\mathbf{1}\{J_t = j\}] \cdot \Delta_j \\ &\leq T \cdot \Delta + \sum_{t=1}^T \sum_{j \in [K]: \Delta_j > \Delta} \mathbb{E}[\mathbf{1}\{J_t = j\}] \cdot \Delta_j \\ &\leq T \cdot \Delta + \sum_{j \in [K]: \Delta_j > \Delta} \left(\frac{8 \ln T}{\Delta_j} + \mathcal{O}(1) \right) \\ &\leq T \cdot \Delta + \sum_{j \in [K]: \Delta_j > \Delta} \left(\frac{8 \ln T}{\Delta} + \mathcal{O}(1) \right) \\ &\leq T \cdot \Delta + K \frac{8 \ln T}{\Delta} + \mathcal{O}(K) \\ &= \mathcal{O}(\sqrt{KT \ln T}) \quad . \end{aligned} \quad (7.19)$$

□

One may ask whether UCB1 is optimal in the worst case sense or not. The definition of minimax optimality, a joint of property between a family of algorithms and distributions, can answer this question. We skip the proof here and only show the conclusion: UCB1 is minimax optimal up to an extra $\sqrt{\ln T}$ factor.

ARM ELIMINATION ALGORITHM. As UCB1 only has an $\mathcal{O}(\sqrt{KT \ln T})$ worst-case regret bound, now, we show an algorithm that enjoys an $\mathcal{O}(\sqrt{KT \ln K})$ worst-case regret bound, which is slightly better than UCB1.

Suppose we have a special K -armed bandit problem with one arm having a mean reward μ_* and all the remaining arms having the same mean reward $\mu_* - \Delta$, i.e., the mean reward gap for any sub-optimal arm is Δ . *Learner knows Δ and μ_* but does not know which arm is the optimal one.* To solve this special bandit problem, we can use the following **Arm Elimination** algorithm [AO10]:

Input: $[K]$, μ_* , Δ , and T .

1. Pull each arm $n = \frac{2 \ln(T\Delta^2)}{\Delta^2}$ times and compute the empirical mean $\hat{\mu}_{j,n}$ of each arm $j \in [K]$;
2. Commit to the arm with the highest empirical mean until the end of the learning, i.e., pull arm $J = \arg \max_{j \in [K]} \hat{\mu}_{j,n}$ for the remaining $T - Kn$ rounds.

THEOREM 7.13. *Arm Elimination enjoys an $\mathcal{O}\left(\frac{K \ln(T\Delta^2)}{\Delta} + \frac{K}{\Delta}\right)$ problem-dependent regret bound. It also enjoys an $\mathcal{O}(\sqrt{KT \ln K})$ worst-case regret bound.*

Proof. Let i_* denote the index of the optimal arm. Without loss of generality, we assume it is unique. We first upper bound the probability that the committed arm is not the optimal one. We have

$$\mathbb{P}\{J \neq i_*\} \leq \mathbb{P}\left\{\max_{j \in [K] \setminus \{i_*\}} \hat{\mu}_{j,n} \geq \hat{\mu}_{i_*,n}\right\} \leq \sum_{j \in [K] \setminus \{i_*\}} \mathbb{P}\{\hat{\mu}_{j,n} \geq \hat{\mu}_{i_*,n}\} \leq (K-1) \cdot 2e^{-n \frac{\Delta^2}{2}}. \quad (7.20)$$

The problem-dependent regret $\bar{\mathcal{R}}(T)$ is

$$\begin{aligned} & \underbrace{(K-1) \cdot n \cdot \Delta}_{\text{regret in Step 1}} + \underbrace{\mathbb{P}\{J \neq i_*\} \cdot (T - Kn) \cdot \Delta}_{\text{regret in Step 2}} \\ & \leq K \cdot \frac{2 \ln(T\Delta^2)}{\Delta^2} \cdot \Delta + K \cdot 2e^{-n \frac{\Delta^2}{2}} \cdot T \cdot \Delta \\ & = 2K \frac{\ln(T\Delta^2)}{\Delta} + K \cdot \frac{2}{T\Delta^2} \cdot T \cdot \Delta \\ & = \mathcal{O}\left(\frac{K \ln(T\Delta^2)}{\Delta} + \frac{K}{\Delta}\right). \end{aligned} \quad (7.21)$$

Let $\tilde{\Delta} := \frac{e\sqrt{K}}{\sqrt{T}}$. If $\Delta \leq \tilde{\Delta}$, we have the regret is at most $T \cdot \tilde{\Delta} = e\sqrt{KT}$. If $\Delta > \tilde{\Delta}$, we have $\mathcal{O}\left(\frac{K \ln(T\Delta^2)}{\Delta} + \frac{K}{\Delta}\right) \leq \mathcal{O}\left(\frac{K \ln(T\tilde{\Delta}^2)}{\tilde{\Delta}} + \frac{K}{\tilde{\Delta}}\right) = \mathcal{O}(\sqrt{KT \ln K})$, where the inequality uses the fact that $f(x) = \frac{\ln(Tx^2)}{x}$ is a decreasing function when $Tx^2 \geq e^2$. \square

ARM ELIMINATION ALGORITHM WITH DOUBLING-TRICK. Since we cannot assume we know Δ in advance and all sub-optimal arms have the same Δ , we cannot use **Arm Elimination** directly for solving practical learning problems. A good thing is we can introduce **doubling-trick** into **Arm Elimination** to make it work by estimating Δ_j .

Input: $[K]$ and T .

Initialization: Set $\hat{\Delta}_1 = 0.5$ and $B_1 = [K]$.

For epochs $r = 1, 2, \dots$ up to $\log T$,

1. For each arm $j \in B_r$, pull it until the total number of pulls hits $n_r = \frac{2 \ln(KT\hat{\Delta}_r^2)}{\hat{\Delta}_r^2}$;
2. All arms $i \in B_r$ such that

$$\underbrace{\hat{\mu}_{i,n_r} + \sqrt{\frac{\log(KT\hat{\Delta}_r^2)}{2 \cdot n_r}}}_{\text{upper confidence bound of arm } i} \geq \underbrace{\max_{j \in B_r} \hat{\mu}_{j,n_r} - \sqrt{\frac{\log(KT\hat{\Delta}_r^2)}{2 \cdot n_r}}}_{\text{lower confidence bound of } j_r^* \in \arg \max_{j \in B_r} \hat{\mu}_{j,n_r}} . \quad (7.22)$$

will be kept in B_{r+1} .

Set $\hat{\Delta}_{r+1} = \frac{\hat{\Delta}_r}{2} = 0.5^{r+1}$.

If $|B_{r+1}| = 1$, commit to that arm until the end of the learning.

It is also fine to set $\hat{\Delta}_0 = 0$ and start from $r = 0$.

You can view B_r as a version space in batch learning.

This is a picture for this.

THEOREM 7.14. *Arm Elimination with Doubling-Trick* has a $\sum_{j \in [K]: \Delta_j > 0} \mathcal{O}\left(\frac{\ln(T\Delta_j^2)}{\Delta_j} + \frac{\ln K}{\Delta_j}\right)$ problem-dependent regret bound and an $\mathcal{O}(\sqrt{KT \ln K})$ worst-case regret bound.

Proof. Let i_* denote the index of the unique optimal arm. Fix a sub-optimal arm j . Let $r_j = \left\lceil \log\left(\frac{1}{\Delta_j}\right) \right\rceil$. Then, we have $0.5\Delta_j \leq 0.5^{r_j} = \hat{\Delta}_{r_j} \leq \Delta_j$.

We claim that the probability that this arm j is kept in B_{r_j+1} is very low. Formally, we have

$$\begin{aligned} & \mathbb{P}\{j \in B_{r_j+1}\} \\ &= \mathbb{P}\left\{j \in B_{r_j}, \hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \max_{j \in B_{r_j}} \hat{\mu}_{j,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}\right\} \\ &\leq \mathbb{P}\left\{\hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \max_{j \in B_{r_j}} \hat{\mu}_{j,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}, i_* \in B_{r_j}\right\} \\ &+ \mathbb{P}\left\{\hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \max_{j \in B_{r_j}} \hat{\mu}_{j,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}, i_* \notin B_{r_j}\right\} \\ &\leq \underbrace{\mathbb{P}\left\{\hat{\mu}_{j,n_{r_j}} + \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}} \geq \hat{\mu}_{i_*,n_{r_j}} - \sqrt{\frac{\log(KT\hat{\Delta}_{r_j}^2)}{2 \cdot n_{r_j}}}\right\}}_{\text{UCB analysis, Hoeffding's inequality}} \end{aligned} \quad (7.23)$$

$$\begin{aligned} &+ \mathbb{P}\{i_* \notin B_2\} + \mathbb{P}\{i_* \in B_2, i_* \notin B_3\} + \dots + \mathbb{P}\{i_* \in B_2, \dots, i_* \in B_{r_j-1}, i_* \notin B_{r_j}\} \\ &\leq \frac{2}{KT\hat{\Delta}_j^2} + \sum_{r=1}^{r_j-1} \frac{2}{T\hat{\Delta}_r^2} \\ &\leq \sum_{r=1}^{r_j} \frac{2}{T\hat{\Delta}_r^2} \\ &= \sum_{r=1}^{r_j} \frac{2}{T \cdot 0.5^{2r}} \\ &= \mathcal{O}\left(\frac{1}{T \cdot 0.5^{2r_j}}\right) \\ &= \mathcal{O}\left(\frac{1}{T \cdot \Delta_j^2}\right) . \end{aligned}$$

The total regret from this sub-optimal arm j is at most

$$\begin{aligned}
& \underbrace{n_{r_j} \cdot \Delta_j}_{\text{regret until the end of epoch } r_j} + \underbrace{T \cdot \mathbb{P}\{j \in \mathcal{B}_{r_j+1}\}}_{\text{regret for the remaining rounds}} \cdot \Delta_j \\
& \leq \frac{2 \ln(KT\hat{\Delta}_{r_j}^2)}{\hat{\Delta}_{r_j}^2} \cdot \Delta_j + T \cdot \mathcal{O}\left(\frac{1}{T \cdot \Delta_j^2}\right) \cdot \Delta_j \\
& \leq \frac{2 \ln(KT\Delta_j^2)}{0.25\Delta_j^2} \cdot \Delta_j + T \cdot \mathcal{O}\left(\frac{1}{T \cdot \Delta_j^2}\right) \cdot \Delta_j \\
& = \mathcal{O}\left(\frac{\ln(T\Delta_j^2)}{\Delta_j} + \frac{\ln K}{\Delta_j}\right).
\end{aligned} \tag{7.24}$$

Summing over all the sub-optimal arms, we have the problem-dependent regret is at most

$$\bar{\mathcal{R}}(T) = \sum_{j \in [K]: \Delta_j > 0} \mathcal{O}\left(\frac{\ln(T\Delta_j^2)}{\Delta_j} + \frac{\ln K}{\Delta_j}\right). \tag{7.25}$$

Let $\tilde{\Delta} := \frac{e\sqrt{K \ln K}}{\sqrt{T}}$. We have

$$\begin{aligned}
\bar{\mathcal{R}}(T) & \leq T \cdot \tilde{\Delta} + \sum_{j \in [K]: \Delta_j > \tilde{\Delta}} \mathcal{O}\left(\frac{\ln(T\Delta_j^2)}{\Delta_j} + \frac{\ln K}{\Delta_j}\right) \\
& \leq e\sqrt{KT \ln K} + \sum_{j \in [K]: \Delta_j > \tilde{\Delta}} \mathcal{O}\left(\frac{\ln(T\tilde{\Delta}^2)}{\tilde{\Delta}} + \frac{\ln K}{\tilde{\Delta}}\right) \\
& \leq e\sqrt{KT \ln K} + \mathcal{O}\left(\frac{K \ln(T\tilde{\Delta}^2)}{\tilde{\Delta}} + \frac{K \ln K}{\tilde{\Delta}}\right) \\
& \leq e\sqrt{KT \ln K} + \mathcal{O}\left(\frac{K \ln(e^2 K \ln K)}{\frac{e\sqrt{K \ln K}}{\sqrt{T}}} + \frac{K \ln K}{\frac{e\sqrt{K \ln K}}{\sqrt{T}}}\right) \\
& = \mathcal{O}(\sqrt{KT \ln K}).
\end{aligned} \tag{7.26}$$

We have

$$\begin{aligned}
\mathbb{P}\{i^* \notin \mathcal{B}_2\} & = \mathbb{P}\left\{\hat{\mu}_{i^*, n_1} + \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}} < \max_{j \in \mathcal{B}_1 \setminus \{i^*\}} \hat{\mu}_{j, n_1} - \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}}\right\} \\
& \leq \sum_{j \in \mathcal{B}_1 \setminus \{i^*\}} \mathbb{P}\left\{\hat{\mu}_{i^*, n_1} + \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}} < \hat{\mu}_{j, n_1} - \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}}\right\} \\
& \leq \sum_{j \in \mathcal{B}_1 \setminus \{i^*\}} \left(\mathbb{P}\left\{\hat{\mu}_{i^*, n_1} + \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}} \leq \mu_1\right\} + \mathbb{P}\left\{\hat{\mu}_{j, n_1} - \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}} \geq \mu_j\right\}\right) \\
& \leq \frac{2}{T\hat{\Delta}_1^2}.
\end{aligned} \tag{7.27}$$

$$\begin{aligned}
& \mathbb{P}\{i^* \in \mathcal{B}_2, i^* \notin \mathcal{B}_3\} \\
& = \mathbb{P}\left\{\hat{\mu}_{i^*, n_1} + \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}} \geq \max_{j \in \mathcal{B}_1} \hat{\mu}_{j, n_1} - \sqrt{\frac{\log(KT\hat{\Delta}_1^2)}{2 \cdot n_1}}, \hat{\mu}_{i^*, n_2} + \sqrt{\frac{\log(KT\hat{\Delta}_2^2)}{2 \cdot n_2}} < \max_{j \in \mathcal{B}_2 \setminus \{i^*\}} \hat{\mu}_{j, n_2} - \sqrt{\frac{\log(KT\hat{\Delta}_2^2)}{2 \cdot n_2}}\right\} \\
& \leq \mathbb{P}\left\{\hat{\mu}_{i^*, n_2} + \sqrt{\frac{\log(KT\hat{\Delta}_2^2)}{2 \cdot n_2}} < \max_{j \in \mathcal{B}_2 \setminus \{i^*\}} \hat{\mu}_{j, n_2} - \sqrt{\frac{\log(KT\hat{\Delta}_2^2)}{2 \cdot n_2}}\right\} \\
& \leq \frac{2}{T\hat{\Delta}_2^2}.
\end{aligned} \tag{7.28}$$

Similarly, we have

$$\mathbb{P}\{i^* \in \mathcal{B}_2, \dots, i^* \in \mathcal{B}_{r_j-1}, i^* \notin \mathcal{B}_{r_j}\} \leq \frac{2}{T\hat{\Delta}_{r_j-1}^2}. \quad \square$$

8 Lower bounds; no free lunch

Armed with our knowledge of online learning from Chapter 7, let's now return to the offline setting from Chapters 1, 2 and 4 to 6.

In the offline setting, so far we've only done upper bounds: in this case, we know we can learn at least this well. But if we only know upper bounds, we never really know how tight they are, and so we can never really know if one algorithm is better than another, or if a learner will really fail in some situation or if it's just that our proof wasn't good enough.

Should this have just come before Chapter 7? Probably, but the teaching timing wasn't right....

One way to approach this problem is with asymptotic results, as described e.g. by [Bach24] who summarizes and translates results from the classic textbook of van der Vaart [vdV98]. For instance, if $\mathcal{H} = \{h_w : w \in \mathcal{W}\}$ for some open set of possible parameters $\mathcal{W} \subseteq \mathbb{R}^D$, the loss is sufficiently "nice" as a function of w , and there's a minimizer $h^* = h_{w^*}$, then as long as some extra "niceness" assumptions also hold, it's true for the ERM that

$$\mathbb{E}_{S \sim \mathcal{D}^m} L_D(\hat{h}_S) - L_D(h^*) = \Theta \left(\frac{1}{m} \text{Tr} \left[\left[\nabla_w^2 L_D(h^*) \right]^{-1} \mathbb{E}_{z \sim \mathcal{D}} \left[(\nabla_w \ell(h_w, z)) (\nabla_w \ell(h_w, z))^T |_{w=w^*} \right] \right] \right).$$

This gives a fast $1/m$ rate – better than the $1/\sqrt{m}$ we've gotten so far (except in A1 Q4) – and along the way it actually also tells us that $w - w^*$ is asymptotically Gaussian, and some other nice things. If we can evaluate the stuff inside the trace, we could also then explicitly say "this \mathcal{H} converges faster than that one," or compare to an asymptotic rate for some different algorithm. But: the "niceness" assumptions don't always hold, the expressions aren't always easy to analyze, and they're purely asymptotic results, so we don't know whether they're a good approximation after $m = 20$ or only after $m = 100,000,000,000$.

Instead, let's use a different route to *lower bounds*, specifically focusing on binary classifiers where these things are easiest.

8.1 NO FREE LUNCH FOR HIGH-VC CLASSES

THEOREM 8.1. *Let \mathcal{H} be a hypothesis set of binary classifiers over \mathcal{X} . Let $m \leq \text{VCdim}(\mathcal{H})/2$. This result is similar to*

Then, using 0-1 loss,

$$\inf_{\mathcal{A}} \sup_{\mathcal{D} \text{ realizable by } \mathcal{H}} \Pr_{S \sim \mathcal{D}^m, \mathcal{A}} \left(L_D(\mathcal{A}(S)) \geq \frac{1}{8} \right) \geq \frac{1}{7},$$

Theorem 5.1 of [SSBD14], but incorporating the idea of VC dimension (which they haven't introduced yet at that point).

where the infimum over \mathcal{A} is over all (possibly randomized) learning algorithms which return hypotheses in \mathcal{H} , and the probability is over both the sampling of a training set and any internal randomness in \mathcal{A} .

Before we prove this, let's unpack the quantifiers a bit. For any m and any learning

algorithm \mathcal{A} , there is some realizable distribution \mathcal{D} such that \mathcal{A} has at least constant probability of failing with m samples, i.e. getting at least $1/8$ error. Note that this distribution *depends on m* and on \mathcal{A} .

This result immediately implies the following:

COROLLARY 8.2. *Any \mathcal{H} with $\text{VCdim}(\mathcal{H}) = \infty$ is not PAC learnable.*

This doesn't necessarily mean that there's any single \mathcal{D} that \mathcal{A} fails on forever. But, at any m , there's still *some distribution* that's too hard. This removes the possibility of PAC learning, which needs to work for *all* distributions at a uniform rate.

Proof of Theorem 8.1. We're first going to pick a shatterable set of size $2m$, $\tilde{\mathcal{X}} = \{\tilde{x}_1, \dots, \tilde{x}_{2m}\} \subseteq \mathcal{X}$; at least one such set must exist, since $2m \leq \text{VCdim}(\mathcal{H})$. Then we'll pick the marginal distribution of x , \mathcal{D}_x , to be a discrete uniform distribution on $\tilde{\mathcal{X}}$. To construct our hard \mathcal{D} , we're going to use this \mathcal{D}_x and then somehow assign a y for each x .

Since we're being totally generic with respect to \mathcal{A} , it's going to be hard to say which $y \mid x$ labeling rule in particular is going to be hard for \mathcal{A} to learn. So, as a proof technique, we're going to start with a *random* labeling rule, and then settle on a particular one later. Specifically, for each vector of possible labels $y \in \{0, 1\}^m$, choose some particular $f \in \mathcal{H}$ such that $f(x_j) = y_j$ for all j ; there must be at least one, since \mathcal{H} shatters $\tilde{\mathcal{X}}$. Let \mathcal{F} be the set of these functions (of size exactly 2^m), and choose $f \sim \text{Unif}(\mathcal{F})$, i.e. we're picking a labeling function uniformly from \mathcal{F} . For any f , let the distribution $\mathcal{D}_{(f)}$ denote the distribution that you get by sampling $x \sim \mathcal{D}_x$ and then assigning $y \mid x = f(x)$.

Now, for any sample of inputs $S_x = (x_1, \dots, x_m)$, we can implicitly construct a sample of pairs $S = ((x_1, f(x_1)), \dots, (x_m, f(x_m)))$. Run the algorithm \mathcal{A} to get $\hat{h}_S = \mathcal{A}(S)$, which itself might be random given S . Its expected loss over the process of choosing a distribution, sampling a training set, and running the algorithm is

$$\mathbb{E}_{f \sim \text{Unif}(\mathcal{F})} \mathbb{E}_{S \sim \mathcal{D}_{(f)}^m} \mathbb{E}_{\mathcal{A}} L_{\mathcal{D}_{(f)}}(\mathcal{A}(S)) = \mathbb{E}_f \mathbb{E}_S \mathbb{E}_{\mathcal{A}} \mathbb{E}_{x \sim \mathcal{D}_x} \mathbb{1}([\mathcal{A}(S)](x) \neq f(x)).$$

Using the law of total expectation, let's break this expectation up based on whether the test x is in the training data S or not:

$$\begin{aligned} \mathbb{E}_{f, S, \mathcal{A}} \mathbb{E}_x \mathbb{1}(\hat{h}_S(x) \neq f(x)) &= \mathbb{E}_{f, S, \mathcal{A}} \left[\Pr(x \notin S_x) \mathbb{E}_{x \sim \mathcal{D}_x} [\mathbb{1}(\hat{h}_S(x) \neq f(x)) \mid x \notin S_x] \right. \\ &\quad \left. + \Pr(x \in S_x) \mathbb{E}_{x \sim \mathcal{D}_x} [\mathbb{1}(\hat{h}_S(x) \neq f(x)) \mid x \in S_x] \right]. \end{aligned}$$

For the second term, we're not going to worry about what the algorithm does on the data it's actually seen, since the algorithm might be good: we'll just bound this as being at least zero.

For the first term, we know since \mathcal{D}_x is uniform and $|S_x| \leq m$ that

$$\Pr(x \notin S_x) = \frac{|\tilde{\mathcal{X}} \setminus S_x|}{|\tilde{\mathcal{X}}|} \geq \frac{m}{2m} = \frac{1}{2}.$$

Also, since our labels $f(\tilde{x}_j)$ are uniformly random and totally independent of one another, and S is independent of those labels for points $\tilde{x} \notin S$, whether \hat{h}_S agrees

with f is just a pure coin flip: $\mathbb{E}_x[\mathbb{1}(\hat{h}_S(x) \neq f(x)) \mid x \notin S_x] = \frac{1}{2}$.

Combining, we know that

$$\mathbb{E}_{f \sim \text{Unif}(\mathcal{F})} \mathbb{E}_{S \sim \mathcal{D}_{(f)}^m} L_{\mathcal{D}_{(f)}}(\hat{h}_S) \geq \frac{1}{4}.$$

But, if the *average* over f of the expected loss $\mathbb{E}_{S \sim \mathcal{D}_{(f)}^m} L_{\mathcal{D}_{(f)}}(\hat{h}_S)$ is at least $\frac{1}{4}$, then there must be at least one *particular* f such that the expected loss is at least $\frac{1}{4}$! Pick one and call it g ; this will be the labeling function claimed by the theorem.

This proof technique is known as the probabilistic method, and often attributed to Paul Erdős.

We've now shown the average loss is large, but we still want to show that the loss has high probability of being large. Now, $L_{\mathcal{D}_{(g)}}(\hat{h}_S)$ is a random variable bounded in $[0, 1]$, and we already know one way to bound those variables in terms of their means: Markov's inequality. But Markov's inequality bounds the probability of things being *big*, and we want to bound the probability of this being *small*. So we'll need to switch it around, which is sometimes called "reverse Markov":

$$\Pr(L_{\mathcal{D}_{(g)}}(\hat{h}_S) \leq \frac{1}{8}) = \Pr\left(1 - L_{\mathcal{D}_{(g)}} \geq 1 - \frac{1}{8}\right) \leq \frac{1 - \mathbb{E} L_{\mathcal{D}_{(g)}}(\hat{h}_S)}{\frac{7}{8}} \leq \left(1 - \frac{1}{4}\right) \frac{8}{7} = \frac{6}{7}.$$

Thus, for the realizable $\mathcal{D}_{(g)}$ we picked above,

$$\Pr_{S \sim \mathcal{D}_{(g)}^m} \left(L_{\mathcal{D}_{(g)}}(\hat{h}_S) > \frac{1}{8}\right) \geq \frac{1}{7}. \quad \square$$

8.1.1 Interpretation

Theorem 8.1 is sometimes called a "no free lunch" theorem, in that there is no algorithm that *always* works (in the sense of PAC learning): every algorithm fails on at least one distribution.

In fact, basically this same proof strategy implies [Wol96] that, if you only care about the "off-sample" error (the average error on $(x, y) \mid x \notin S_x$), there are just as many possible distributions where your predictor is right as where it's wrong, regardless of your learning algorithm. If you don't assume *anything* about the world, all algorithms perform the same on average over all possible worlds.

This is in some ways a deep philosophical problem, called the [problem of induction](#) and generally credited to David Hume. The fact that the sun rose every day so far doesn't, from "pure first principles," imply anything about whether it will rise tomorrow: we just decide to prefer "simple" explanations, i.e. we choose some \mathcal{H} that we like. But that doesn't really answer which \mathcal{H} would be good.

Actually, VC or Rademacher theory can't answer that problem either: it's preferable to choose a \mathcal{H} with small complexity, but since $\text{Rad}((\mathcal{H} + \{f\})|_S) = \text{Rad}(\mathcal{H}|_S)$, and $\text{VCdim}(\mathcal{H}) = \text{VCdim}(\{x \mapsto h(x)f(x) : h \in \mathcal{H}\})$ for ± 1 -valued h and f , we haven't actually seen any objective notion of a "simple hypothesis": only ways to say that *sets* of hypotheses are all similar enough to one another.

Sometimes people get a little mystical about no free lunch theorems, though – e.g. <https://no-free-lunch.org> says that this result "calls the whole of science into question." But the world is *not* uniformly random; we know from experience that some kinds of \mathcal{H} tend to work better than others. so, although there is *some* distribution that every algorithm fails on, it's not the case in the world we live in that all algorithms are the same as each other. (And, interestingly, there *are* (impractical)

learning algorithms that are always at least as good as any other algorithm, up to (huge) constants: `free-lunch.org` used to (but, alas, no longer) point to the paper of Nakkiran [Nak21].)

8.1.2 *Aside: “learning is NP-hard”*

Another example of this kind of claim (based on a different underlying theorem) is given by van Rooij et al. [vRoo+24], who say (in reaction to recent progress of LLMs):

[We present] a mathematical proof of inherent intractability (formally, NP-hardness) of the task that [...] AI engineers set themselves. This intractability implies that any factual AI system created in the short-run (say, within the next few decades or so) is so astronomically unlikely to be anything like a human mind, or even a coherent capacity that is part of that mind, that claims of ‘inevitability’ of AGI within the foreseeable future are revealed to be false and misleading. We realize that this implication may appear counterintuitive given everyday experiences and interactions with currently impressive AI systems, but we will explain why it is not. As we will carefully unpack later in the paper, it is a mistake to assume that AI systems’ performance is either currently human-level, or will simply continue to improve and the systems will soon constitute human-level A(G)I. The problem is that—in line with our intractability result—the performance cannot scale up.

What they actually prove (their Theorem 2) can be rephrased roughly as follows:

THEOREM 8.3 (“Ingenia Theorem”, [vRoo+24]). *Let $\mathcal{X} = \{0, 1\}^N$ and \mathcal{Y} a fixed finite set. For each x , define $\mathcal{Y}_x \subsetneq \mathcal{Y}$ to be the set of “acceptable” responses to an input x . Let \mathcal{H} be a hypothesis class containing all functions implemented by circuits with complexity at most a parameter D ; for instance, for each N and D there exists a class of feedforward neural networks satisfying this. A realizable distribution \mathcal{D} is one where $\Pr_{(x,y) \sim \mathcal{D}}(y \in \mathcal{Y}_x) = 1$ and there exists $h^* \in \mathcal{H}$ with $\Pr_{(x,y) \sim \mathcal{D}}(h^*(x) \in \mathcal{Y}_x) = 1$. Suppose that there exists a polynomial-time algorithm, allowed to randomly sample from \mathcal{D} as a constant-time operation, which with probability at least $\Omega(1/N^\alpha)$ for some $\alpha > 0$ successfully identifies a hypothesis $h \in \mathcal{H}$ satisfying*

$$\Pr_{(x,y) \sim \mathcal{D}}(h(x) \in \mathcal{Y}_x) \geq \frac{|\mathcal{Y}_x|}{|\mathcal{Y}|} + \epsilon_N,$$

for some $\epsilon_N = \Omega(1/N^\beta)$, for some $\beta > 0$. Then $NP \subseteq BPP$.

This conclusion contradicts a *very* common assumption in complexity theory. So, although we don’t 100% know this for a fact, we should probably think that this implies there is no polynomial-time algorithm satisfying the above properties, i.e. that can improve on random guessing.

Does this imply that “AI” is computationally infeasible? **Not really.** Assuming $NP \not\subseteq BPP$, it implies that for any given polynomial-time learning algorithm, there exist *some* distributions which cannot be efficiently learned. (This is true even for distributions which are themselves efficiently computable. The universal induction approach considered e.g. by Nakkiran [Nak21] finds computationally-efficient hypotheses but it does so in an extremely computationally-inefficient way.)

This obviously doesn’t mean, though, that *every* distribution can’t be efficiently

learned. For instance, the distribution that always says “banana please” in response to any input at all can be. Is “human-like behaviour” a distribution that can be efficiently learned by some algorithm? I don’t know (other than to say that, well, humans do it), and this theorem doesn’t say either!

8.2 LOWER BOUNDS

Theorem 8.1 only applies when $m \leq \text{VCdim}(\mathcal{H})/2$. We can use it, though, to also get a quantitative lower bound for higher m :

THEOREM 8.4. *Let \mathcal{H} be a set of binary classifiers over \mathcal{X} such that $\text{VCdim}(\mathcal{H}) \geq 2$. For any $m > \text{VCdim}(\mathcal{H})/2$,*

$$\inf_{\mathcal{A}} \sup_{\mathcal{D} \text{ realizable by } \mathcal{H}} \Pr_{S \sim \mathcal{D}^m} \left(L_{\mathcal{D}}(\mathcal{A}(S)) > \frac{\text{VCdim}(\mathcal{H}) - 1}{32m} \right) > \frac{1}{100}$$

where $L_{\mathcal{D}}$ uses zero-one loss, and the infimum over \mathcal{A} is over all learning algorithms returning hypotheses in \mathcal{H} .

Proof. Choose a set $\tilde{\mathcal{X}} = \{\tilde{x}_1, \dots, \tilde{x}_d\}$ of size $d = \text{VCdim}(\mathcal{H})$ which can be shattered by \mathcal{H} . We’re going to choose a distribution that puts most of its probability mass on \tilde{x}_1 , in such a way that we’re likely to see less than half of the *other* points from the distribution. Specifically, for an $\varepsilon > 0$ to choose later,

$$\Pr_{x \sim \mathcal{D}_x} (x = \tilde{x}_1) = 1 - \varepsilon, \quad \text{for all } i > 1, \quad \Pr_{x \sim \mathcal{D}_x} (x = \tilde{x}_i) = \frac{\varepsilon}{d - 1}.$$

Now, let $\tilde{\mathcal{D}}$ be the distribution over $\{\tilde{x}_2, \dots, \tilde{x}_d\}$ selected by Theorem 8.1 with $m = (d - 1)/2$, and let $f \in \mathcal{H}$ be the labeling function chosen in $\tilde{\mathcal{D}}$. Our distribution will be found by sampling $x \sim \mathcal{D}_x$ and then letting $y \mid x = f(x)$.

Now, we’re going to prove that it’s fairly likely that samples from \mathcal{D}_x contain at most $(d - 1)/2$ of the non- \tilde{x}_1 points. How many points we *don’t* see is a little annoying to characterize exactly, but we can get a bound based on

$$Q = \sum_{i=1}^m \mathbb{1}(x_i \neq \tilde{x}_1);$$

if we repeat any of the non- \tilde{x}_1 points, Q will double-count them, but it’s a valid upper bound on the number of non- \tilde{x}_1 points we see. Notice that $\Pr(x_i \neq \tilde{x}_1) = \varepsilon$, and each of the indicators is iid Bernoulli(ε), so $Q \sim \text{Binomial}(m, \varepsilon)$.

A standard tail bound for binomial variables, Proposition 8.5 with $\gamma = 1$, shows that

$$\Pr(Q \geq 2m\varepsilon) \leq \exp\left(-\frac{1}{3}m\varepsilon\right).$$

To use this result, we want $2m\varepsilon = \frac{1}{2}(d - 1)$; so, pick $\varepsilon = (d - 1)/(4m)$. This is valid, since $m > d/2$ implies that $\varepsilon < \frac{1}{2} \cdot \frac{d-1}{d} < \frac{1}{2}$. Then we see less than half of the non- \tilde{x}_1 points with probability at least

$$1 - \exp\left(-\frac{m}{3} \cdot \frac{d - 1}{4m}\right) = 1 - \exp\left(-\frac{d - 1}{12}\right) \geq 1 - \exp\left(-\frac{1}{12}\right) > 0.07,$$

since $1 - \exp(-1/12) \approx 0.07995$.

So, with more than 7% probability, a sample of size m from \mathcal{D} will contain at most

This theorem roughly follows [MRT18, Theorem 3.20]. That result merges this result with Theorem 8.1 in a way I find really hard to follow; their theorem statement is also obviously incorrect when $m < (\text{VCdim}(\mathcal{H}) - 1)/32$. [SSBD14, Theorem 6.8] states a similar result, but leaves this part as an exercise.

$(d - 1)/2$ of the non- \tilde{x}_1 points. Then, Theorem 8.1 tells us that with probability at least $1/7$, $L_{\mathcal{D}}(\mathcal{A}(S)) \geq \frac{1}{8}$. If this happens, this implies that $L_{\mathcal{D}}(\mathcal{A}(S)) \geq \frac{1}{8}\varepsilon = \frac{d-1}{32m}$, since the total probability of the non- \tilde{x}_1 points is exactly ε . So, we have more than a $\frac{1}{7} \cdot 7\% = 1\%$ chance of seeing $\frac{d-1}{32m}$ error on \mathcal{D} , as desired. \square

PROPOSITION 8.5. *If $X \sim \text{Binomial}(m, p)$, then for any $\gamma > 0$ it holds that*

$$\Pr(X \geq (1 + \gamma)mp) \leq \exp\left(-\frac{1}{3}mp\gamma^2\right).$$

This is an immediate consequence of the multiplicative Chernoff bound, which is e.g. Theorem D.4 of [MRT18]. The proof technique is different from how we proved Hoeffding/etc, and I don't know if it holds as generally, but you should be able to follow their proof (which uses their Theorem D.3) just fine.

AGNOSTIC CASE You can get a bigger error if you don't require \mathcal{D} to be realizable: Theorem 3.23 of [MRT18] gives that for any m and \mathcal{H} ,

$$\inf_{\mathcal{A}} \sup_{\mathcal{D}} \Pr\left(L_{\mathcal{D}}(\mathcal{A}(S)) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) \geq \sqrt{\frac{d}{320m}}\right) \geq \frac{1}{64}. \quad (8.1)$$

Section 28.2 of [SSBD14] is similar.

MORE GENERALLY These styles of theorems are sometimes called “minimax bounds,” and algorithms are called “minimax-optimal” or simply “minimax” if they achieve the lower bound (usually only up to constants, though that's also sometimes called “rate-optimal”). In the VC notes we showed that ERM gets error $\tilde{O}_p(\sqrt{d/m})$, which combined with the agnostic result above shows that ERM is (up to log factors) rate-optimal for finite-VC classes. Although we haven't shown this (see Section 28.3 of [SSBD14] or 6.5 of [Zhang23]), ERM for binary classifiers achieves $\tilde{O}_p(d/m)$ error in the realizable setting, so by Theorem 8.4 ERM is also (up to log factors) minimax rate-optimal for realizable distributions too.

Minimax rates are also available for various other problems, including things like linear regression, density estimation, and optimization. We won't talk a lot about lower bounds in this course, but they can be really nice to know whether your learning algorithm is “good” or not. (The problem, though, is they tend to be extremely “worst-case,” and might not be too informative about problems you're likely to actually see – similar to no free lunch arguments.)

8.3 THE “FUNDAMENTAL THEOREM OF STATISTICAL LEARNING”

We've now shown all the necessary parts for a pretty complete qualitative understanding of PAC learning for binary classifiers.

This name is only, as far as I know, used by [SSBD14]. **THEOREM 8.6** (Fundamental Theorem of Statistical Learning). *For \mathcal{H} a class of functions $h : \mathcal{X} \rightarrow \{0, 1\}$ and with the 0-1 loss, the following are equivalent:*

1. *Uniform convergence: for all $\varepsilon, \delta \in (0, 1)$, we have that $\sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) < \varepsilon$ with probability at least $1 - \delta$ as long as $m \geq m^{\text{UC}}(\varepsilon, \delta) < \infty$.*
2. *Any ERM rule agnostically PAC-learns \mathcal{H} .*
3. *\mathcal{H} is agnostically PAC learnable.*
4. *Any ERM rule PAC-learns \mathcal{H} .*
5. *\mathcal{H} is PAC learnable.*

[SSBD14] use two-sided uniform convergence: in the setting of the theorem here, one-sided bounds imply two-sided ones, but (a) one-sided is what we really use, and (b) in more general settings the distinction can matter.

6. $\text{VCdim}(\mathcal{H}) < \infty$.

Proof. 1 implying 2 is our usual argument:

$$L_{\mathcal{D}}(\hat{h}_S) \leq L_S(\hat{h}_S) + \sup_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_S(h) \leq L_S(h^*) + \varepsilon \leq L_{\mathcal{D}}(h^*) + [L_S(h^*) - L_{\mathcal{D}}(h^*)] + \varepsilon,$$

plus Hoeffding on $L_S(h^*) - L_{\mathcal{D}}(h^*)$.

2 implying 3, and 4 implying 5, are immediate.

2 implying 4, and 3 implying 5, is also straightforward from the definitions.

Corollary 8.2 shows that 5 implies 6.

6 implying 1 is shown by Theorem 6.11. □

Theorem 6.8 of [SSBD14] gives a quantitative version, bounding the sample complexities in terms of the VC dimension, by collecting lower bounds like Theorem 8.4 and (8.1) and upper bounds like Theorem 6.11 and the realizable equivalent that we didn't prove.

9 Nonuniform Learning

Recall the decomposition of error we made back in Section 1.4:

$$\underbrace{L_{\mathcal{D}}(\hat{h}_S) - L_{bayes}}_{\text{excess error}} = \underbrace{L_{\mathcal{D}}(\hat{h}_S) - \inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h)}_{\text{estimation error}} + \underbrace{\inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_{bayes}}_{\text{approximation error}}.$$

We've talked a lot about the estimation error of ERM, bounding it in terms of Rademacher complexity or (when applicable) VC dimension. What we haven't really talked about yet is the approximation error. We drew some examples with polynomials in Figure 1.1, but if we don't know what the optimal predictor looks like... what should we do?

There are some particular cases where we can analyze this approximation error gap mathematically, if we assume things about the form of \mathcal{D} . But those assumptions usually rely on constants that are hard to know for any specific problem, and there's not usually a clear way to estimate them (or the Bayes error) from data, either.

The practical solution is generally to just try a bunch of different \mathcal{H} and/or a bunch of different learning algorithms, then pick the best based on a validation set V . This is a good idea in practice, and we can make some theoretical guarantees on its generalization based on L_V being close to $L_{\mathcal{D}}$. But it's still hard to use that approach to say anything with confidence about the approximation error.

9.1 STRUCTURAL RISK MINIMIZATION

SRM says: let's use a *huge* \mathcal{H} , one where the approximation error is going to be small, maybe even *zero* if \mathcal{H} is what's called *universal* (coming up soon!). This will probably mean \mathcal{H} has infinite VC dimension, large Rademacher complexity, etc. But let's decompose

$$\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \dots = \bigcup_{k \in \mathbb{N}} \mathcal{H}_k.$$

For instance, we might have \mathcal{H}_k the set of decision trees of depth k , the set of degree- k polynomials, or the set of linear classifiers with $\|w\| \leq 2^k$. We're going to assume that *each* \mathcal{H}_k has uniform convergence:

$$\forall k \in \mathbb{N}. \quad \Pr_{S \sim \mathcal{D}^m} \left(\sup_{h \in \mathcal{H}_k} L_{\mathcal{D}}(h) - L_S(h) \leq \varepsilon_k(m, \delta) \right) \geq 1 - \delta \quad (9.1)$$

for functions ε_k satisfying that for all k and all $\delta \in (0, 1)$, $\lim_{m \rightarrow \infty} \varepsilon_k(m, \delta) = 0$.

We'll also need a set of weights $w_k \geq 0$ such that $\sum_{k=1}^{\infty} w_k \leq 1$; a typical choice is

This is the problem that made Euler famous. $6/(\pi^2 k^2) \approx 0.61/k^2$, since $\sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$.

PROPOSITION 9.1. Let $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \dots$ satisfy (9.1), and let $w_k \geq 0$ have $\sum_{k=1}^{\infty} w_k \leq 1$. Then for any \mathcal{D} , with probability at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$, we have

$$\forall h \in \mathcal{H}. \quad L_{\mathcal{D}}(h) \leq L_S(h) + \min_{k: h \in \mathcal{H}_k} \varepsilon_k(m, \delta w_k).$$

Proof. We do a union bound over the \mathcal{H}_k , allocating δw_1 probability that anything in \mathcal{H}_1 violates the bound, δw_2 that anything in \mathcal{H}_2 does, and so on. Thus the total probability anything in \mathcal{H} violates it is at most $\sum_k \delta w_k \leq \delta$. \square

SRM is then the algorithm that minimizes this upper bound on $L_{\mathcal{D}}(h)$:

DEFINITION 9.2. Given bounds on a decomposition of \mathcal{H} as in (9.1), and weights $w_k \geq 0$ with $\sum w_k \leq 1$ and $\bigcup_{k: w_k > 0} \mathcal{H}_k = \mathcal{H}$, structural risk minimization is given by

$$\text{SRM}_{\mathcal{H}, \delta}(S) \in \arg \min_{h \in \mathcal{H}} \left[L_S(h) + \varepsilon_{k_h}(m, \delta w_{k_h}) \right] \quad \text{where } k_h \in \arg \min_{k: h \in \mathcal{H}_k} \varepsilon_k(m, w_k \delta).$$

Typically, $k_h = \min\{k : h \in \mathcal{H}_k\}$.

We can implement this minimization by a finite number of calls to an ‘‘ERM oracle’’, as long as our loss is lower-bounded by $a \leq \ell(h, z)$, e.g. $a = 0$:

```

function SRMℋ, δ(S)
    best ← ∞
    for k = 1, 2, ... do
        hk ← ERMℋk(S)
        cand_loss ← LS(hk) + εk(m, wk δ)
        if cand < best then
            ĥ ← hk
            best ← cand
        if mink' > k a + εk'(m, wk' δ) > best then
            break
    return ĥ
    
```

Note that if we ‘‘decompose’’ as $\mathcal{H}_1 = \mathcal{H}$, then SRM becomes just $\text{ERM}_{\mathcal{H}}$.

THEOREM 9.3. Let $h^* \in \mathcal{H}$ be any fixed hypothesis in the setup of Definition 9.2, and let $a \leq \ell(h, z) \leq b$ for all $h \in \mathcal{H}$, $z \in \mathcal{Z}$. Then, with probability at least $1 - \delta - \delta'$ over the choice of random samples $S \sim \mathcal{D}^m$, SRM satisfies

$$L_{\mathcal{D}}(\text{SRM}_{\mathcal{H}, \delta}(S)) \leq L_{\mathcal{D}}(h^*) + \varepsilon_{k_{h^*}}(m, w_{k_{h^*}} \delta) + (b - a) \sqrt{\frac{1}{2m} \log \frac{1}{\delta'}}.$$

Proof. Let $\hat{h}_S = \text{SRM}_{\mathcal{H}}(S)$. We have that

$$\begin{aligned} L_{\mathcal{D}}(\hat{h}_S) &\leq L_S(\hat{h}_S) + \varepsilon_{k_{\hat{h}_S}}(m, w_{k_{\hat{h}_S}} \delta) && \text{by Proposition 9.1, prob } \geq 1 - \delta \\ &\leq L_S(h^*) + \varepsilon_{k_{h^*}}(m, w_{k_{h^*}} \delta) && \text{by def of SRM;} \end{aligned}$$

the conclusion follows by applying Hoeffding’s inequality with probability δ' to upper-bound $L_S(h^*)$. \square

Compare this to ERM that just knows in advance which $\mathcal{H}_{k_{h^*}}$ to pick; with probability at least $1 - 2\delta$, that would have performance

$$L_{\mathcal{D}}(\text{ERM}_{\mathcal{H}}(S)) \leq L_{\mathcal{D}}(h^*) + \varepsilon_{k_{h^*}}(m, \delta) + (b - a) \sqrt{\frac{1}{2m} \log \frac{1}{\delta}}.$$

How much worse this is depends on how much worse $\varepsilon_{k_{h^*}}(m, w_{k_{h^*}} \delta)$ is than $\varepsilon_{k_{h^*}}(m, \delta)$.

9.1.1 With Rademacher bounds

Since this is a little abstract, let's see what happens if we plug in the Rademacher bound of Theorem 5.7: let $\mathcal{R}_{k,m} = \mathbb{E}_{S \sim \mathcal{D}^m} \text{Rad}((\ell \circ \mathcal{H}_k)|_S)$, assume $a \leq \ell(h, z) \leq b$, and for simplicity assume that $\mathcal{R}_{k+1,m} \geq \mathcal{R}_{k,m}$ for all k . Then

$$\varepsilon_k(m, \delta) = 2\mathcal{R}_{k,m} + (b - a) \sqrt{\frac{1}{2m} \log \frac{1}{\delta}}.$$

Let's also plug in $w_k = 6/(\pi^2 k^2)$. Then Proposition 9.1 becomes that

$$\Pr \left(\forall h \in \mathcal{H}. \quad L_{\mathcal{D}}(h) \leq L_S(h) + 2\mathcal{R}_{k_h,m} + (b - a) \sqrt{\frac{1}{2m} \log \frac{\pi^2 k_h^2}{6\delta}} \right) \geq 1 - \delta, \quad (9.2)$$

where $k_h = \min\{k : h \in \mathcal{H}_k\}$. Using this bound to define an SRM algorithm gives

$$\text{SRM}_{\mathcal{H},\delta}(S) \in \arg \min_{h \in \mathcal{H}} \left[L_S(h) + 2\mathcal{R}_{k_h,m} + (b - a) \sqrt{\frac{1}{2m} \log \frac{\pi^2 k_h^2}{6\delta}} \right]. \quad (9.3)$$

Theorem 9.3 gives that with probability at least $1 - (1 + \frac{6}{\pi^2})\delta$,

$$\begin{aligned} L_{\mathcal{D}}(\text{SRM}_{\mathcal{H},\delta}(S)) &\leq L_{\mathcal{D}}(h^*) + 2\mathcal{R}_{k_{h^*},m} + \frac{b-a}{\sqrt{m}} \left[\sqrt{\log k_h + \frac{1}{2} \log \frac{\pi^2}{6\delta}} + \sqrt{\frac{1}{2} \log \frac{\pi^2}{6\delta}} \right] \\ &\leq L_{\mathcal{D}}(h^*) + 2\mathcal{R}_{k_{h^*},m} + \frac{b-a}{\sqrt{m}} \left[\sqrt{\log k_h} + \sqrt{2 \log \frac{\pi^2}{6\delta}} \right]. \end{aligned}$$

Letting $\delta' = \frac{\pi^2+6}{\pi^2} \delta$ so that $\frac{\pi^2}{6\delta} = \frac{\pi^2}{6} \frac{\pi^2}{\pi^2+6} \frac{1}{\delta} < \frac{1.03}{\delta}$, this means that with probability at least $1 - \delta'$ we have

$$L_{\mathcal{D}} \left(\text{SRM}_{\mathcal{H}, \frac{\pi^2}{\pi^2+6} \delta'}(S) \right) \leq L_{\mathcal{D}}(h^*) + 2\mathcal{R}_{k_{h^*},m} + \frac{b-a}{\sqrt{m}} \left[\sqrt{\log k_{h^*}} + \sqrt{2 \log \frac{1.03}{\delta'}} \right]. \quad (9.4)$$

Compare to ERM with $\mathcal{H}_{k_{h^*}}$: with probability at least $1 - \delta'$,

$$L_{\mathcal{D}}(\text{ERM}_{\mathcal{H}_{k_{h^*}}}(S)) \leq L_{\mathcal{D}}(h^*) + 2\mathcal{R}_{k_{h^*},m} + \frac{b-a}{\sqrt{m}} \sqrt{2 \log \frac{2}{\delta'}}.$$

So, as long as we have a reasonable number of samples compared to the complexity of h^* – that is, $m \gg \log k_{h^*}$ – we pay essentially no penalty for not knowing the correct \mathcal{H}_k in advance!

9.1.2 Problems with bound minimization

Concentration inequalities are usually pretty conservative, since they hold for *all* distributions subject to some mild constraints (e.g. sub-Gaussianity). Symmetrization

is also often a bit loose; it introduces a factor of 2 that might not be needed, e.g. in equation (11) / Appendix E.4 of [Zho+22] we established that this 2 can (basically) be a 1 for Gaussian-data ℓ_1 -loss regression.

So, if we minimize a potentially loose bound, then we might get bad results: because our bound is too conservative, we'll have too much bias towards a simple solution. (If the problem turns out to be realizable, but we didn't assume that from the outset, then we can't adapt to the fast $1/m$ rate; we'll operate assuming the slow $1/\sqrt{m}$ rate.) Fundamentally, this means the performance of our algorithm is based on how good at theoretical analysis we are; we'd usually rather have an algorithm that works well whether we're smart or not.

It's also kind of weird for us to have to pre-commit to a certain failure probability δ ; that's not usually how we think about things. That in particular, though, we'll be able to avoid.

9.1.3 Aside: Avoiding the δ dependence

It's pretty annoying that the algorithm depends on a specific choice of δ ; that "feels like" an analysis parameter, not an algorithm one. We can do this by defining a slight variant of the algorithm; notice that (9.2) implies

$$\Pr \left(\forall h \in \mathcal{H}. L_{\mathcal{D}}(h) \leq L_S(h) + 2\mathcal{R}_{k_h, m} + (b-a)\sqrt{\frac{1}{m} \log k_h} + (b-a)\sqrt{\frac{1}{2m} \log \frac{\pi^2}{6\delta}} \right) \geq 1 - \delta,$$

since we only made the upper bound looser with $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for nonnegative a, b . But when minimizing *this* upper bound, the $(b-a)\sqrt{\frac{1}{2m} \log \frac{\pi^2}{6\delta}}$ term doesn't depend on h at all, and so we can just ignore it;

$$\text{SRM}_{\mathcal{H}}(S) \in \arg \min_{h \in \mathcal{H}} \left[L_S(h) + 2\mathcal{R}_{k_h, m} + (b-a)\sqrt{\frac{1}{m} \log k_h} \right].$$

A slight variant of Theorem 9.3 still applies; we just have to use the ε_k that splits the two square root terms up, giving for this variant that with probability at least $1 - \delta$,

$$\begin{aligned} \text{w/ prob at least } 1 - \frac{\pi^2}{6+\pi^2}\delta \quad L_{\mathcal{D}}(\text{SRM}_{\mathcal{H}}(S)) &\leq L_S(\hat{h}_S) + 2\mathcal{R}_{k_{\hat{h}_S}, m} + \frac{b-a}{\sqrt{m}} \left[\sqrt{\log k_{\hat{h}_S}} + \sqrt{\frac{1}{2} \log \frac{6+\pi^2}{6\delta}} \right] \\ \text{by def of SRM} \quad &\leq L_S(h^*) + 2\mathcal{R}_{k_{h^*}, m} + \frac{b-a}{\sqrt{m}} \left[\sqrt{\log k_{h^*}} + \sqrt{\frac{1}{2} \log \frac{6+\pi^2}{6\delta}} \right] \\ \text{w/ prob at least } 1 - \frac{6}{6+\pi^2}\delta \quad &\leq L_{\mathcal{D}}(h^*) + 2\mathcal{R}_{k_{h^*}, m} + \frac{b-a}{\sqrt{m}} \left[\sqrt{\log k_{h^*}} + \sqrt{\frac{1}{2} \log \frac{6+\pi^2}{6\delta}} + \sqrt{\frac{1}{2} \log \frac{6+\pi^2}{6\delta}} \right] \\ &= L_{\mathcal{D}}(h^*) + 2\mathcal{R}_{k_{h^*}, m} + \frac{b-a}{\sqrt{m}} \left[\sqrt{\log k_{h^*}} + \sqrt{2 \log \frac{1+\pi^2/6}{\delta}} \right]. \end{aligned}$$

Note that $1 + \pi^2/6 < 2.7$. This gets essentially the same result as (9.4), without requiring committing to a δ in the algorithm.

Note that this was only possible because $[a, b]$ didn't depend on \mathcal{H}_k . This isn't always true; for example, our analysis of logistic regression with $\|x\| \leq C$ and $\mathcal{H}_B = \{x \mapsto w \cdot x : \|w\| \leq B\}$ used (4.4) to get that $b - a = BC$. In these cases, if we

want to use our exact SRM analysis, as far as I know we have to incorporate δ into the algorithm itself.

9.1.4 Relationship to regularization

Think about using SRM with $\mathcal{H} = \{x \mapsto w \cdot x : w \in \mathbb{R}^d\}$ with $\mathcal{H}_k = \{x \mapsto w \cdot x : \|w\| \leq r2^{k-1}\}$ for some $r > 0$; this should be chosen in advance of seeing the data, e.g. just picking $r = 1$. Consider logistic loss, and assume $\|x\| \leq C$ almost surely.

Suppose that h corresponds to a vector w . If $\|w\| \geq r$, we have

$$B_{k_h-1} = \frac{1}{2}B_{k_h} = r2^{k_h-2} < \|w\| \leq r2^{k_h-1} = B_{k_h},$$

implying $B_{k_h} < 2\|w\|$ and $k_h < 2 + \log_2 \frac{\|w\|}{r}$. Thus, in general, $B_{k_h} < \max(2\|w\|, r)$ and $k_h < 2 + \max\left(0, \log_2 \frac{\|w\|}{r}\right) = \max\left(2, \log_2 \frac{4\|w\|}{r}\right)$. Thus, recalling Section 5.2.2 and Equation (4.4), we can use (9.3) to construct an instance of SRM as

$$\arg \min_{w \in \mathbb{R}^d} L_S(x \mapsto w \cdot x) + \frac{C \max(2\|w\|, r)}{\sqrt{m}} \left[2 + \sqrt{\log \left(\max \left(2, \log_2 \frac{4\|w\|}{r} \right) \right) + \frac{1}{2} \log \frac{\pi^2}{6\delta}} \right].$$

Now, let's squint a bit, and assume that we chose an r such that the w with $\|w\|$ significantly smaller than r aren't relevant to the optimization – they're not confident enough to achieve a small L_S – but that getting a low L_S doesn't require a $\|w\|$ so big that $\log \log_2 \frac{4\|w\|}{r}$ is meaningfully more than “constant.” Then, this optimization problem looks a lot like

$$\arg \min_{w \in \mathbb{R}^d} L_S(x \mapsto w \cdot x) + \frac{\lambda}{\sqrt{m}} \|w\|$$

for some $\lambda > 0$. This is pretty close to the “default” regularized logistic regression, which would use $\|w\|^2$. (It also probably wouldn't have an explicit m in the equation, but if you're tuning λ for a fixed particular problem, that doesn't matter, and indeed the total amount of regularization should often scale with m according to \sqrt{m} , as we'll see a little later in the course.)

In fact, the optimization problems with $\|w\|$ and with $\|w\|^2$ are themselves equivalent: if you consider the curve of possible solutions as you vary λ (the “regularization path”), you would get the exact same set of solutions. So, SRM can be seen as motivation for standard regularization techniques.

9.2 NONUNIFORM LEARNABILITY

The sample complexity for SRM to learn a hypothesis h^* depends on the particular h^* , not just on \mathcal{H} . This motivates a weaker definition of learning than PAC learning, called *nonuniform learning*.

DEFINITION 9.4. An algorithm $\mathcal{A}(S)$ (ϵ, δ)-competes with a hypothesis h if it satisfies $\Pr_{S \sim \mathcal{D}^m}(L_{\mathcal{D}}(\mathcal{A}(S)) \leq L_{\mathcal{D}}(h) + \epsilon) \geq 1 - \delta$.

DEFINITION 9.5. An algorithm \mathcal{A} nonuniformly learns \mathcal{H} there is a finite sample complexity function $m(\epsilon, \delta, h)$ such that for all $\epsilon, \delta \in (0, 1)$ and $h \in \mathcal{H}$, given $m \geq m(\epsilon, \delta, h)$ iid samples from any \mathcal{D} , $\mathcal{A}(S)$ (ϵ, δ)-competes with h .

DEFINITION 9.6. A hypothesis class \mathcal{H} is nonuniformly learnable if there exists an

algorithm \mathcal{A} which nonuniformly learns \mathcal{H} .

Theorem 9.3 establishes that SRM nonuniformly learns any \mathcal{H} which we can decompose into a countable union of \mathcal{H}_k which each allow for uniform convergence.

In fact, for binary classifiers with 0-1 loss, SRM nonuniformly learns any \mathcal{H} which is nonuniformly learnable:

PROPOSITION 9.7. *If \mathcal{H} of binary classifiers is nonuniformly learnable under the 0-1 loss, it can be written as a countable union of \mathcal{H}_k with finite VC dimension.*

Proof. Define

$$\mathcal{H}_k = \left\{ h \in \mathcal{H} : m\left(\frac{1}{8}, \frac{1}{7}, h\right) \leq k \right\},$$

where $m(\varepsilon, \delta, h)$ is the sample complexity function of an algorithm \mathcal{A} that nonuniformly learns \mathcal{H} . Then $\mathcal{H} = \bigcup_{k \geq 1} \mathcal{H}_k$.

For any k , consider \mathcal{H}_k . Let \mathcal{D} be any distribution realizable by \mathcal{H}_k , i.e. there is some $h^* \in \mathcal{H}_k$ with $L_{\mathcal{D}}(h^*) = 0$. Since $\mathcal{A}(S)$ competes with that h^* , $\Pr_{S \sim \mathcal{D}^m}(L_{\mathcal{D}}(\mathcal{A}(S)) \leq \frac{1}{8}) \geq \frac{6}{7}$. This means that we can (roughly) learn *any* realizable distribution. But our No Free Lunch theorem, specifically Corollary 8.2, implied that, if $\text{VCdim}(\mathcal{H}_k) = \infty$, then there would be some realizable \mathcal{D} that we can't learn to this (ε, δ) . Thus $\text{VCdim}(\mathcal{H}_k)$ can't be infinite. \square

9.3 MINIMUM DESCRIPTION LENGTH

9.3.1 Singleton Classes

Suppose we have a countable $\mathcal{H} = \{h_1, h_2, \dots\}$. Then we could partition it into *singleton* sub-classes, $\mathcal{H}_k = \{h_k\}$. Denoting the weight for the class $\{h\}$ by w_h , each of these \mathcal{H}_k have “uniform convergence” via a simple Hoeffding bound with

$$\varepsilon_k(m, w_h \delta) \leq (b-a) \sqrt{\frac{1}{2m} \log \frac{1}{w_h \delta}} \leq (b-a) \sqrt{\frac{1}{2m} \log \frac{1}{w_h}} + (b-a) \sqrt{\frac{1}{2m} \log \frac{1}{\delta}},$$

splitting out the dependence on δ for simplicity as in Section 9.1.3. SRM then becomes

$$\text{SRM}_{\mathcal{H}}(S) \in \arg \min_{h \in \mathcal{H}} L_S(h) + \sqrt{\frac{1}{2m} \log \frac{1}{w_h}},$$

and this has the guarantee by Theorem 9.3 that

$$L_{\mathcal{D}}(\text{SRM}_{\mathcal{H}}(S)) \leq L_{\mathcal{D}}(h^*) + (b-a) \sqrt{\frac{1}{2m} \log \frac{1}{w_{h^*}}} + (b-a) \sqrt{\frac{2}{m} \log \frac{2}{\delta}}.$$

But... how should we set w_h ? There's no “smaller” h ; what order should we use?

9.3.2 Minimum Description Length

One popular way to decide on weights is based on choosing some *prefix-free binary language* to determine the hypotheses: for example, the binary representation of a gzipped Python program implementing that hypothesis. Then we can choose a weight according to the following result:

PROPOSITION 9.8 (Kraft’s inequality). *If $\mathcal{S} \subseteq \{0, 1\}^*$ is prefix-free (there are no $s \neq s' \in \mathcal{S}$ such that s is a prefix of s'), then*

$$\sum_{s \in \mathcal{S}} 2^{-|s|} \leq 1.$$

Proof. Define the following random process: starting with the empty string, add either a 0 or a 1 with equal probability. If the current string is in \mathcal{S} , terminate; if no element of \mathcal{S} begins with the current string, also terminate; otherwise, repeat. Since \mathcal{S} is prefix-free, this process hits any string $s \in \mathcal{S}$ with probability $2^{-|s|}$; these probabilities must sum to at most one. \square

Thus, we can choose a representation for \mathcal{H} so that h has description length $|h|$, and assign $w_h = 2^{-|h|}$. This gives

$$\begin{aligned} \text{MDL}_{\mathcal{H}}(\mathcal{S}) &\in \arg \min_{h \in \mathcal{H}} L_{\mathcal{S}}(h) + \sqrt{\frac{\log 2}{2m}} |h| \\ L_{\mathcal{D}}(\text{MDL}_{\mathcal{H}}(\mathcal{S})) &\leq L_{\mathcal{D}}(h^*) + (b-a) \sqrt{\frac{\log 2}{2m}} |h^*| + (b-a) \sqrt{\frac{2}{m} \log \frac{2}{\delta}}. \end{aligned}$$

This is one formalization of Occam’s razor: if there are multiple explanations of the data ($L_{\mathcal{S}}(h_1) = 0 = L_{\mathcal{S}}(h_2)$), prefer the simplest one (the one with shortest explanation).

But we need to *pre-commit* to a notion of description length before seeing the data. A nice analogy: `codegolf.stackexchange.com`, a site where people compete to find the shortest implementation of a program doing some task, prohibits by default any language written **after the contest was started**.

If we choose $|h|$ to be the length of shortest possible implementation of h in some programming language, this is known as the *Kolmogorov complexity*. This version of the MDL principle is then to regularize by the Kolmogorov complexity. If you’re familiar with Bayesian learning, this would be something like *maximum a posteriori* (MAP) inference with a Kolmogorov complexity prior. The “free lunch” algorithm outlined by Nakkiran [Nak21] is closely related to this where \mathcal{H} is just the set of all Turing machines. The fully-Bayesian analogue is (basically) something called *Solomonoff induction*. For fuller introductions to these concepts, there are various relevant textbooks [LV19; Hut05; HQC24].

It’s not quite the same; MAP wouldn’t have the square root.

10 Universal Approximation

In our motivation of SRM in Chapter 9, we talked about wanting to use an \mathcal{H} so big that the approximation error $\inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - L_{\text{bayes}}$ is zero. What kinds of \mathcal{H} satisfy that?

To keep things simple, we'll think about $\mathcal{Y} \subseteq \mathbb{R}$ today.

One example would be the set of all functions $\mathcal{X} \rightarrow \mathcal{Y}$. This way leads a million mathematical counterexamples of being able to do even super basic things like computing expectations, let alone being able to learn.

A milder set to target is the set of all continuous functions. If there's a continuous function achieving the Bayes error, then this immediately guarantees that the approximation error would be zero.

DEFINITION 10.1. For a metric space \mathcal{X} , $C(\mathcal{X})$ denotes the Banach space of continuous functions $\mathcal{X} \rightarrow \mathbb{R}$, with norm given by $\|f\|_{\infty} = \sup_{x \in \mathcal{X}} |f(x)|$.

Recall that if f and g are elements of a function space and $a \in \mathbb{R}$, we have that $af + g$ is the function mapping x to $af(x) + g(x)$. So, $\|f - g\|_{\infty} = \sup_{x \in \mathcal{X}} |f(x) - g(x)|$ is one possible distance metric on functions.

The following result suggests that this is a reasonable (if strict) way to calculate distances between functions.

PROPOSITION 10.2. Suppose that $\ell(h, (x, y)) = l_y(h(x))$ for $l_y : \mathbb{R} \rightarrow \mathbb{R}$. Then $L_{\mathcal{D}}$ is $\left(\mathbb{E}_{(x,y) \sim \mathcal{D}} \|l_y\|_{\text{Lip}}\right)$ -Lipschitz with respect to $\|h - g\|_{\infty}$.

Proof. We have that

$$\begin{aligned} |L_{\mathcal{D}}(h) - L_{\mathcal{D}}(g)| &= \left| \mathbb{E}_{(x,y) \sim \mathcal{D}} l_y(h(x)) - \mathbb{E}_{(x,y) \sim \mathcal{D}} l_y(g(x)) \right| \leq \mathbb{E}_{(x,y) \sim \mathcal{D}} |l_y(h(x)) - l_y(g(x))| \\ &\leq \mathbb{E}_{(x,y) \sim \mathcal{D}} \|l_y\|_{\text{Lip}} |h(x) - g(x)| \leq \left(\mathbb{E}_{(x,y) \sim \mathcal{D}} \|l_y\|_{\text{Lip}} \right) \|h - g\|_{\infty}. \quad \square \end{aligned}$$

10.1 DENSENESS

Even if the “target function” isn't continuous, the approximation error could still be zero.

EXAMPLE 10.3. Consider $\mathcal{X} = \mathbb{R}$ and the true labels being determined by the discontinuous function $y = \mathbf{1}(x > 0)$. Although this function isn't in $C(\mathcal{X})$, you can get

arbitrarily close to it, e.g. by taking the continuous functions

$$f_\sigma(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x/\sigma & \text{if } 0 \leq x \leq \sigma \\ 1 & \text{if } x \geq \sigma. \end{cases}$$

The 0-1 loss here is

$$L_{\mathcal{D}}(f_\sigma) = \Pr(x \in (0, \sigma)) \mathbb{E}\left[1 - \frac{x}{\sigma} \mid x \in (0, \sigma)\right] < \Pr(x \in (0, \sigma)).$$

As $\sigma \rightarrow 0$, we have $L_{\mathcal{D}}(f_\sigma) \rightarrow 0$ regardless of \mathcal{D} . Thus, $\inf_{h \in C(\mathcal{X})} L_{\mathcal{D}}(h) = 0$, even though there is no $h \in C(\mathcal{X})$ with $L_{\mathcal{D}}(h) = 0$. Therefore the approximation error, in this case, is zero.

$C(\mathcal{X})$ can approximate many interesting function classes. We can frame this with the following definition from metric topology:

DEFINITION 10.4. Let $\mathcal{G} \subseteq \mathcal{F}$ for some metric space \mathcal{F} . We say that \mathcal{G} is *dense* in \mathcal{F} with respect to the metric ρ if, for every $f \in \mathcal{F}$, $\inf_{g \in \mathcal{G}} \rho(g, f) = 0$.

That is, for every point in $f \in \mathcal{F}$ that isn't in \mathcal{G} , you need to be able to get *arbitrarily close* to f with points in \mathcal{G} .

A canonical example is that the set of rational numbers is dense in the set of real numbers.

PROPOSITION 10.5. Suppose that \mathcal{H} is dense in \mathcal{F} with respect to $\|\cdot\|_\infty$, and use loss $\ell(h, (x, y)) = l_y(h(x))$ with finite $\mathbb{E}_{(x,y) \sim \mathcal{D}} \|l_y\|_{\text{Lip}}$. Then $\inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) = \inf_{f \in \mathcal{F}} L_{\mathcal{D}}(f)$.

Proof. Let $M = \mathbb{E}_{(x,y) \sim \mathcal{D}} \|l_y\|_{\text{Lip}} < \infty$. Choose (f_1, f_2, \dots) to be a sequence in \mathcal{F} such that $L_{\mathcal{D}}(f_i) \rightarrow \inf_{f \in \mathcal{F}} L_{\mathcal{D}}(f)$. For each f_i , choose a $g_i \in \mathcal{G}$ such that $\|f_i - g_i\|_\infty \leq \frac{1}{i}$, which is possible because \mathcal{G} is dense in \mathcal{F} . Then, by Proposition 10.2, $|L_{\mathcal{D}}(g_i) - L_{\mathcal{D}}(f_i)| \leq M \|g_i - f_i\|_\infty \leq \frac{M}{i} \rightarrow 0$, and thus $(L_{\mathcal{D}}(g_i))$ converges to the same point as $(L_{\mathcal{D}}(f_i))$. \square

10.2 UNIVERSAL APPROXIMATORS

There are many variants of universality [see e.g. SFL10]; this is a reasonable baseline.

DEFINITION 10.6. We call a hypothesis class \mathcal{H} of functions $\mathcal{X} \rightarrow \mathbb{R}$ *universal* if $\mathcal{H} \cap C(\mathcal{X})$ is dense in $C(\mathcal{X})$ with respect to $\|\cdot\|_\infty$.

The following property is known as *separating compact sets*. It establishes that thresholding functions in a universal hypothesis class can shatter any set, so that $\text{VCdim}(\text{sgn} \circ \mathcal{H}) = \infty$. It also implies that the Rademacher complexity is infinite.

Finite sets are compact. **PROPOSITION 10.7.** Let $V, W \subset \mathcal{X}$ be disjoint compact sets, and let \mathcal{H} be universal. Choose any $a \geq 0$. Then there exists an $h \in \mathcal{H}$ such that $h(x) > a$ for all $x \in V$, and $h(x) < -a$ for all $x \in W$.

Proof. Define $\rho_V(x) = \min_{v \in V} \|x - v\|$, and likewise ρ_W . Since the sets are compact, we can use just min instead of inf, and they'll still be well-defined continuous functions in $C(\mathcal{X})$. Since the sets are compact and disjoint, if $\rho_V(x) = 0$ then

$\rho_W(x) > 0$, and vice versa. Thus the following g is well-defined and continuous:

$$g(x) = 2a \frac{\rho_V(x) - \rho_W(x)}{\rho_V(x) + \rho_W(x)}.$$

If $x \in V$, then $D_V(x) = 0$, and so $g(x) = -2a$ for $x \in V$. Likewise, $g(x) = 2a$ for $x \in W$. Thus, any $h \in \mathcal{H}$ with $\|h - g\|_\infty < a$ will satisfy the property we want. Since g is continuous and \mathcal{H} is dense in $C(\mathcal{X})$, such an h must exist. \square

This result implies that, at least for binary classifiers, it's impossible to PAC-learn a universal \mathcal{H} . Depending on the \mathcal{H} , though, we may be able to nonuniformly learn it with SRM or similar algorithms. If we use a decomposition $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2 \cup \dots$ for $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \dots$, then even though the approximation error in all of \mathcal{H} is zero, the approximation error in \mathcal{H}_k might not be. As we consider \mathcal{H}_k for increasing k , we trade off higher estimation error for lower approximation error. When \mathcal{H} is universal, there might be some \mathcal{H}_k where we can achieve zero approximation error (if there's some $h \in \mathcal{H}$ achieving the minimal loss, also called the *well-specified* setting). We might, though, only have the approximation error of \mathcal{H}_k going to zero as k increases, called a *misspecified setting*; this would be true e.g. in Example 10.3 with $\mathcal{H}_k = \{f : \|f\|_{\text{Lip}} \leq k\}$.

Well-specified doesn't imply realizable; you might have $\inf_{h \in \mathcal{H}} L_{\mathcal{D}}(h) > 0$.

10.3 UNIVERSAL APPROXIMATION OF NEURAL NETWORKS

As you may have heard before (probably invoked in somewhat mystical ways), classes of neural networks are universal.

A *feedforward neural network* (or *multilayer perceptron*, MLP) is a function defined hierarchically as

$$f(x) = f^{(D)}(x) \quad f^{(k)}(x) = \sigma_k(W_k f^{(k-1)}(x) + b_k) \quad f^{(0)}(x) = x,$$

where $W_k \in \mathbb{R}^{d'_k \times d_{k-1}}$, $b_k \in \mathbb{R}^{d'_k}$, and $\sigma_k : \mathbb{R}^{d'_k} \rightarrow \mathbb{R}^{d_k}$; usually, $d_k = d'_k$. Typically $\sigma_D(z) = z$, while intermediate *hidden layers* use nonlinear activations. Many common choices are componentwise, such as $\text{ReLU}(z) = \max\{z, 0\}$, \tanh , or $\text{sigmoid}(z) = \frac{1}{1 + \exp(-z)}$. Other choices include $\text{softmax}(z) = (\exp(z_j))_j / \sum_j \exp(z_j)$, max pooling, attention operators, and so on.

On A3 Q3, you bounded the Rademacher complexity for some such networks, with some assumptions on σ_k , \mathcal{D} , bounds on W_k , and that $b_k = 0$. (There are [slightly] better bounds than this one; we'll talk about this a bit soon.) Your bound didn't explicitly depend on the number of parameters, just on their norms.

It's worth noting now that neural networks are usually trained via stochastic gradient descent, but this non-convex optimization can be difficult: in general, it's NP-hard, even to optimize a single ReLU unit with square loss [GKMR21]. We'll talk more about optimization soon.

10.3.1 Constructive proofs

The following result is easy to understand, and extremely simple, but is indicative of universal approximation results in general.

THEOREM 10.8. *Let $g : [0, 1] \rightarrow \mathbb{R}$ be M -Lipschitz. For any $\varepsilon > 0$, there exists a network f such that $\|f - g\|_\infty \leq \varepsilon$, where the network has one hidden layer of width $N = \lceil M/\varepsilon \rceil$*

using threshold activations $\sigma(t) = \mathbb{1}(t \geq 0)$, and a linear output unit.

Proof. We're going to construct a piecewise-constant approximation to g . For $i \in \{0, \dots, N-1\}$, let $b_i = \frac{i\varepsilon}{M}$, i.e.

$$b_0 = 0, \quad b_1 = \frac{\varepsilon}{M}, \quad \dots, \quad b_{N-1} = \left(\left\lceil \frac{M}{\varepsilon} \right\rceil - 1\right) \frac{\varepsilon}{M} < \frac{M}{\varepsilon} \cdot \frac{\varepsilon}{M} = 1.$$

We're going to construct

$$f(x) = \begin{cases} g(0) & \text{if } 0 \leq x < b_1 \\ g(b_1) & \text{if } b_1 \leq x < b_2 \\ \vdots & \\ g(b_{N-1}) & \text{if } b_{N-1} \leq x \leq 1 \end{cases}$$

as a two-layer network. To do this, let $a_0 = g(0)$, and for $i \geq 1$ let $a_i = g(b_i) - a_{i-1}$, so that

$$\sum_{i=0}^k a_i = g(0) + (g(b_1) - g(0)) + (g(b_2) - (g(b_1) - g(0))) + \dots = g(b_k).$$

Thus the desired f is just

$$f(x) = \sum_{i=0}^{N-1} a_i \mathbb{1}(x \geq b_i),$$

which is a network of the desired form: the first layer has a weight matrix of all ones, and a bias vector collecting the negatives of the thresholds b_i , while the second layer has weights collecting the a_i and no offset.

You could use a narrower network by depending on the total variation of g , how much it "wiggles" up and down: if g is pretty flat in some region, there's no need to keep putting points there, you only need a new one when g changes more than ε .

Now, consider any input x , and let $k = \max\{k : b_k \leq x\}$. Then, since g is M -Lipschitz,

$$|g(x) - f(x)| \leq \underbrace{|g(x) - g(b_k)|}_{\leq M|x-b_k|} + \underbrace{|g(b_k) - f(b_k)|}_0 + \underbrace{|f(b_k) - f(x)|}_0 \leq M \frac{\varepsilon}{M} = \varepsilon. \quad \square$$

We could do a similar thing with ReLU networks, using piecewise-linear approximations rather than piecewise-constant.

Here's a similar result in \mathbb{R}^d :

δ exists for any ε , since continuous functions on compact domains are uniformly continuous, and $\|\cdot\|_2$ and $\|\cdot\|_\infty$ are equivalent.

THEOREM 10.9. *Let $g : [0, 1]^d \rightarrow \mathbb{R}$ be continuous. For any $\varepsilon > 0$, choose $\delta > 0$ such that $\|x - x'\|_\infty \leq \delta$ implies $|g(x) - g(x')| \leq \varepsilon$. Then there is a three-layer ReLU network f with $\Omega\left(\frac{1}{\delta^d}\right)$ ReLU nodes satisfying $\int_{[0,1]^d} |f(x) - g(x)| dx \leq 2\varepsilon$.*

Proof (sketch). Approximate the continuous g by a piecewise-constant h , with pieces given by hyper-rectangles. Construct a two-layer ReLU net to check whether the input x is in each hyper-rectangle. Put those networks side-by-side as the first two layers of f , so that the second hidden layer is just an indicator vector of which hyper-rectangle x is in. Use a linear readout layer to set any value on those pieces.

For details, see Theorem 2.1 of Telgarsky [Tel21]. □

Notice the *curse of dimensionality*: the size of the network depends exponentially on the dimension, which for deep learning is typically *at least* hundreds, perhaps millions or more. This isn't just a proof artifact; it's necessary to approximate

arbitrary continuous functions. The construction also needs really large weights, and has a really bad Lipschitz constant; it also only gives an L_1 approximation bound, not sup-norm like before.

10.3.2 Non-constructive bound via Stone-Weierstrass

We can actually get a sup-norm bound with only one hidden layer a different way, using the celebrated Stone-Weierstrass approximation theorem from analysis.

THEOREM 10.10 (Stone-Weierstrass, special case). *Let \mathcal{X} be a compact metric space. Suppose \mathcal{F} is a set of functions from $\mathcal{X} \rightarrow \mathbb{R}$ such that:*

- Each $f \in \mathcal{F}$ is continuous: $\mathcal{F} \subseteq C(\mathcal{X})$.
- For each $x \in \mathcal{X}$, there is at least one $f \in \mathcal{F}$ with $f(x) \neq 0$.
- For all $f, g \in \mathcal{F}$ and $\alpha \in \mathbb{R}$, we have $\alpha f + g \in \mathcal{F}$ and $f g = (x \mapsto f(x)g(x)) \in \mathcal{F}$. \mathcal{F} is an algebra.
- For each $x \neq x' \in \mathcal{X}$, there is at least one $f \in \mathcal{F}$ with $f(x) \neq f(x')$. \mathcal{F} separates points.

Then \mathcal{F} is dense in $C(\mathcal{X})$ with respect to $\|\cdot\|_\infty$.

You may have heard of the Weierstrass theorem, which shows that polynomial functions are dense in $C(\mathcal{X})$; this is a generalization.

PROPOSITION 10.11. *The set of functions \mathcal{F}_{exp} is dense in $C(\mathcal{X})$, where*

$$\mathcal{F}_{\text{exp}} = \left\{ x \mapsto \sum_{i=1}^m a_i \exp(w_i \cdot x) : m \geq 1; w_1, \dots, w_m \in \mathbb{R}^d; a_1, \dots, a_m \in \mathbb{R} \right\}.$$

Notice that \mathcal{F}_{exp} is a set of one-hidden-layer neural networks with exponential hidden activations and *unbounded width*.

Proof. We just need to show that it satisfies the conditions of Stone-Weierstrass. The first two are clear. For $f(x) = \sum_{i=1}^m a_i \exp(w_i \cdot x)$ and $g(x) = \sum_{i=1}^{m'} a'_i \exp(w'_i \cdot x)$, we have

$$\alpha f + g = \left(x \mapsto \sum_{i=1}^m (\alpha a_i) \exp(w_i \cdot x) + \sum_{i=1}^{m'} a'_i \exp(w'_i \cdot x) \right) \in \mathcal{F}_{\text{exp}}$$

$$f g = \left(x \mapsto \sum_{i=1}^m \sum_{j=1}^{m'} a_i a'_j \exp((w_i + w'_j) \cdot x) \right) \in \mathcal{F}_{\text{exp}}.$$

To show \mathcal{F}_{exp} separates x_1 and x_2 , consider $f(x) = \exp((x_1 - x_2) \cdot x)$, so that

$$\frac{f(x_1)}{f(x_2)} = \frac{\exp(\|x_1\|^2 - x_2 \cdot x_1)}{\exp(x_1 \cdot x_2 - \|x_2\|^2)} = \exp(\|x_1\|^2 - 2x_1 \cdot x_2 + \|x_2\|^2) = \exp(\|x_1 - x_2\|^2),$$

which is one iff $x_1 = x_2$. □

PROPOSITION 10.12 ([HSW89]). *Let $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be continuous with $\lim_{z \rightarrow -\infty} \sigma(z) = 0$, $\lim_{z \rightarrow \infty} \sigma(z) = 1$. Then \mathcal{F}_σ is dense in $C(\mathcal{X})$, where \mathcal{F}_σ is defined as*

$$\mathcal{F}_\sigma = \left\{ x \mapsto \sum_{i=1}^m a_i \sigma(w_i \cdot x) : m \geq 1; w_1, \dots, w_m \in \mathbb{R}^d; a_1, \dots, a_m \in \mathbb{R} \right\}.$$

Proof (sketch). For any continuous target g , first find an $f_0 \in \mathcal{F}_{\text{exp}}$ such that $\|f_0 - g\|_\infty \leq \varepsilon/2$. Now, find some coefficients such that

$$\exp(z) \approx \sum_j c_j \sigma(t_j z)$$

is sufficiently accurate so that when we replace each $\exp(w_i \cdot x)$ in f_0 by $\sum_i c_i \sigma(t_i w_i \cdot x)$, we find an $f \in \mathcal{F}_\sigma$ such that $\|f - f_0\|_\infty \leq \varepsilon/2$. \square

More generally, this works if σ is anything that's not a polynomial [LLPS93]. (A shallow network with fixed-degree polynomial activations is itself a polynomial of fixed degree.) These are for shallow, wide networks, but if you use a deep, narrow network you can get away even with polynomial activations [KL20].

There are also a variety of other results. Maybe most important is an infinite-width construction of Barron [Bar93]; also see Section 3 of [Tel21] or Section 9.3 of [Bach24].

10.4 CIRCUIT COMPLEXITY

We won't go into depth on this perspective, but it's definitely worth knowing it exists. Shalev-Shwartz and Ben-David [SSBD14, Chapter 20] overview the general basic results, but the standard classic text seems to be Parberry [Par94]. There's also recent work, particularly on Transformers.

The short version:

- Two-layer networks with threshold activations can represent all functions from $\{\pm 1\}^d \rightarrow \{\pm 1\}$. Since computers always represent things as binary strings, that's pretty powerful.
- But, it takes exponential width to do that.
- But, for any Boolean function that can be computed with maximal runtime T , there exists a network of size $\mathcal{O}(T^2)$ that implements that function.

10.5 INTERPRETATION

“Neural networks can do anything!!”

(You don't hear “decision trees can do anything!!” as often, but it's just as true....)

These results mean that, for any (continuous) function (on a bounded domain) that we'd like to approximate, there *is* some neural net that can closely approximate that behaviour. Continuous functions also aren't a huge limit, as in Example 10.3. So, there is *some* neural network that can approximate “what's the next bit in the response of a very smart human to a Unicode string of length at most 128,000 bytes.” But that network is going to be *very* large (in parameter count and also weight norm). There's also a *really really big* decision tree that can do that.

So, does ERM in a large enough hypothesis class, or SRM, or whatever other learning algorithm, necessarily generalize? Maybe not.

Also, for neural networks ERM is NP-hard; does gradient descent approximate it well? Maybe not.

But, are these constructions with *enormous* norms indicative of the actual norm required for functions we care about? Maybe not.

One way to help answer these questions is to characterize what kinds of functions have large norms. This is mostly beyond the scope of this course, but the typical traditional scheme is based on functions in Sobolev classes; [Bach24] has a bunch of material on this. There's also recent work on, say, constructing Transformers to do some particular task, as an existence proof of approximation for *that* task (rather than universally).

11 Kernels

We've mentioned a couple times the idea of implementing a polynomial classifier as a special case of a linear one: in \mathbb{R} , a cubic classifier might look like

$$h(x) = w_0 + w_1x + w_2x^2 + w_3x^3$$

where we have four parameters in w . Notice that we can also write this as

$$h(x) = w \cdot \phi(x), \quad w \in \mathbb{R}^4, \quad \phi(x) = (1, x, x^2, x^3).$$

Now, consider the set of all cubic functions

$$\mathcal{F} = \{x \mapsto w \cdot \phi(x) = w_0 + w_1x + w_2x^2 + w_3x^3 : w \in \mathbb{R}^4\}.$$

We're going to introduce some machinery to think about \mathcal{F} as a function space, along the lines of the space $C(\mathcal{X})$ from Definition 10.1. This will lead to *kernel methods* that allow us to optimize over \mathcal{F} using basically the same techniques as optimizing over linear spaces.

"Kernel" is a super-overloaded word. This is not the same thing as in kernel density estimation, the kernel of a convolution, the kernel of a probability density, the kernel of a linear map, a CUDA kernel, an operating system kernel...

11.1 DEFINING FUNCTION SPACES

To think of \mathcal{F} as a vector space of functions, let $f, f' \in \mathcal{F}$ correspond to weight vectors w, w' . Then we can let $f + f'$ be the function with weight vector $w + w'$, and af that with weight vector aw . This definition makes it a valid **vector space**:

DEFINITION 11.1. A real *vector space* is a non-empty set V along with the operations of *vector addition*, denoted $v + w \in V$ for any $v, w \in V$, and *scalar multiplication*, denoted $av \in V$ for any $v \in V$ and $a \in \mathbb{R}$, satisfying the following requirements:

- Vector addition is associative: for all $u, v, w \in V$, $u + (v + w) = (u + v) + w$.
- Vector addition is commutative: for all $v, w \in V$, $v + w = w + v$.
- Vector addition has an identity: there is some *zero vector* $0 \in V$ such that for all $v \in V$, $v + 0 = v$.
- Vector addition has inverses: for each $v \in V$, there is some $-v \in V$ such that $v + (-v) = 0$.
- Compatibility of scalar multiplication: for all $a, b \in \mathbb{R}$ and $v \in V$, $a(bv) = (ab)v$.
- Identity of scalar multiplication: for all $v \in V$, $(1)v = v$.
- Distributive property I: for all $a \in \mathbb{R}$ and $v, w \in V$, $a(v + w) = av + aw$.
- Distributive property II: for all $a, b \in \mathbb{R}$ and $v \in V$, $(a + b)v = av + bv$.

A lot of the familiar linear algebra stuff you know and love from \mathbb{R}^d applies to any vector space as well.

DEFINITION 11.2. A real *normed vector space* is a real vector space V with a *norm*: a function $V \rightarrow \mathbb{R}$, written $\|v\|$, such that:

- Non-negativity: for all $v \in V$, $\|v\| \geq 0$.
- Positive definiteness: for every $v \in V$, $\|v\| = 0$ if and only if $v = 0$.
- Absolute homogeneity: for every $a \in \mathbb{R}$ and $v \in V$, $\|av\| = |a|\|v\|$.
- Sub-additivity / triangle inequality: for every $v, w \in V$, $\|v + w\| \leq \|v\| + \|w\|$.

The norm of a normed vector space induces the metric $\rho(x, y) = \|x - y\|$, which we can check satisfies the formal definition of a metric space:

DEFINITION 11.3. A *metric space* is a set \mathcal{X} along with a function $\rho : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, called the *metric*, satisfying the following properties:

- Non-negativity: for all $x, y \in \mathcal{X}$, $\rho(x, y) \geq 0$.
- Positive definiteness for all $x, y \in \mathcal{X}$, $\rho(x, y) = 0$ if and only if $x = y$.
- Symmetry: for all $x, y \in \mathcal{X}$, $\rho(x, y) = \rho(y, x)$.
- Triangle inequality: for all $x, y, z \in \mathcal{X}$, $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$.

DEFINITION 11.4. Consider a sequence x_1, x_2, \dots in a metric space \mathcal{X} .

This sequence has a *limit* x_∞ if for every $\varepsilon > 0$, there exists a positive integer N such that for all $n > N$, $\rho(x_n, x_\infty) < \varepsilon$.

This sequence is called *Cauchy* if, for every $\varepsilon > 0$, there exists a positive integer N such that for all $m, n > N$, $\rho(x_m, x_n) < \varepsilon$.

The metric space \mathcal{X} is called *complete* if all Cauchy sequences in \mathcal{X} have limits in \mathcal{X} .

DEFINITION 11.5. A real *Banach space* is a real normed vector space whose norm induces a complete vector space.

You can check that $C(\mathcal{X})$ is a Banach space.

There's one other major structure in \mathbb{R}^d that we don't have yet: dot products.

DEFINITION 11.6. A real *inner product space* is a real vector space V together with an inner product, a function $V \times V \rightarrow \mathbb{R}$ written $\langle v, w \rangle$ satisfying

- Symmetry: for all $v, w \in V$, $\langle v, w \rangle = \langle w, v \rangle$.
- Linearity: for all $u, v, w \in V$ and $a, b \in \mathbb{R}$, $\langle au + bv, w \rangle = a\langle u, w \rangle + b\langle v, w \rangle$.
- Positive-definiteness: if $v \neq 0$, then $\langle v, v \rangle > 0$.

An inner product space is also a normed vector space with $\|v\| = \sqrt{\langle v, v \rangle}$, and hence a metric space with $\rho(v, w) = \|v - w\| = \sqrt{\langle v - w, v - w \rangle}$.

DEFINITION 11.7. A real *Hilbert space* is a real inner product space whose induced metric space is complete.

11.2 POLYNOMIAL FUNCTIONS

Now, recall our function space

$$\mathcal{F} = \{x \mapsto w \cdot \phi(x) = w_0 + w_1x + w_2x^2 + w_3x^3 : w \in \mathbb{R}^4\}$$

with addition defined by adding weight vectors, and scalar multiplication by scaling the weight vectors. We can also define an inner product $\langle f, f' \rangle_{\mathcal{F}}$ by $w \cdot w'$, also giving the norm $\|f\|_{\mathcal{F}} = \|w\|$. We can check that this satisfies all the conditions we need, including completeness, for \mathcal{F} to define a Hilbert space.

Now, let's think about a different function class. Choose any $c > 0$ and define

$$\mathcal{F}_c = \{x \mapsto w \cdot \phi(x) = w_0\sqrt{c^3} + w_1\sqrt{3c^2}x + w_2\sqrt{3c}x^2 + w_3x^3 : w \in \mathbb{R}^4\},$$

then again define addition / scalar multiplication / inner products in terms of *these* weight vectors w . The reason for this reparameterization is that we get

$$\phi(x) \cdot \phi(x') = c^3 + 3c^2xx' + 3c(xx')^2 + (xx')^3 = (xx' + c)^3,$$

which makes $\phi(x) \cdot \phi(x')$ much easier to compute. The same thing happens in higher dimensions or with higher polynomial degrees; for degree- ℓ polynomials in d dimensions, there are $\mathcal{O}(d^\ell)$ parameters, but we can compute this inner product $\phi(x) \cdot \phi(x')$ still in $\mathcal{O}(d)$ time.

We call this function $\phi(x) \cdot \phi(x')$ the *kernel function*:

$$k(x, x') = \phi(x) \cdot \phi(x').$$

We'll see soon that it's a very fundamental object.

The set of functions in \mathcal{F} and \mathcal{F}_c for any c are the same, as functions; addition and scalar multiplication also agree between all of them. But the inner product doesn't! So $\|w\|$, and hence $\|f\|_{\mathcal{F}_c}$, is different depending on your choice of c . (Larger c will mean the lower-order coefficients can be smaller in order to express the same function, and so means that $\|f\|_{\mathcal{F}}$ is more determined by the coefficient on x^3 .) This will be important when we use algorithms that depend on $\|f\|_{\mathcal{F}}$.

Now, let's do something slightly weird. Recall that

$$\phi(x) = (\sqrt{c^3}, \sqrt{c^2}x, \sqrt{c}x^2, x^3) \in \mathbb{R}^4.$$

Elements of \mathcal{F}_c are functions corresponding to any $w \in \mathbb{R}^4$. So what happens if we think of the element of $\phi(x)$ as a weight vector for an element in \mathcal{F}_c ? This would give us a function of the form

$$\begin{aligned} x' \mapsto & \sqrt{c^3}\sqrt{c^3} + \sqrt{3c^2}x\sqrt{3c^2}x' + \sqrt{3c}x\sqrt{3c}(x')^2 + x^3(x')^3 \\ & = c^3 + 3c^2xx' + 3c(xx')^2 + (xx')^3 \\ & = (xx' + c)^3 = \phi(x) \cdot \phi(x'). \end{aligned}$$

That is, if we evaluate the function with weights $\phi(x)$ at a point x' , we just get the kernel back. There isn't any magic here; we defined \mathcal{F} that way in the first place! Letting $f_w \in \mathcal{F}$ denote the function with weight vector w , this means that

$$\langle f_{\phi(x)}, f_{\phi(x')} \rangle_{\mathcal{F}} = k(x, x').$$

Now, because it's a vector space, we know that $\sum_{i=1}^n \alpha_i f_{\phi(x_i)} \in \mathcal{F}$ for any n , $\alpha_i \in \mathbb{R}$, and choice of x_i . By the linearity properties of inner product spaces,

$$\left\langle \sum_{i=1}^n \alpha_i f_{\phi(x_i)}, f_{\phi(x)} \right\rangle_{\mathcal{F}} = \sum_{i=1}^n \alpha_i \langle f_{\phi(x_i)}, f_{\phi(x)} \rangle_{\mathcal{F}} = \sum_{i=1}^n \alpha_i k(x_i, x).$$

Since $f_{\phi(x_i)} \in \mathcal{F}$ is a function from \mathcal{X} to \mathbb{R} , this is the same as taking a linear combination of the functions, in terms of their pointwise evaluations.

So, we can think of \mathcal{F} as having a vector space structure without direct reference to w , where $af + f'$ is defined as the function $x \mapsto af(x) + f'(x)$, and where $f(x) = \langle f, f_{\phi(x)} \rangle_{\mathcal{F}}$ (also known as the **reproducing property**) – at least for any f that's a linear combination of $f_{\phi(x_i)}$ for some x_i . This will be the basis for our construction of a *reproducing kernel Hilbert space* (RKHS) for a generic kernel.

The notation $f_{\phi(x)}$ is a little bit cumbersome. Kernels people often use $k(x, \cdot)$ to denote this. This notation is justified because $k(x, \cdot)$ would normally mean the function $t \mapsto k(x, t)$; but that's exactly what you get when you do $f_{\phi(x)}(t) = \phi(x) \cdot \phi(t) = k(x, t)$.

11.3 REPRODUCING KERNELS

Not every function can be a kernel: it needs to be possible to write as an inner product. So:

DEFINITION 11.8. A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a *positive definite kernel* if and only if there exists some Hilbert space \mathcal{G} and feature map $\phi : \mathcal{X} \rightarrow \mathcal{G}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{G}}$.

Notice that the space, and the map, don't need to be unique (e.g. you could always use $-\phi$ instead of ϕ). Sometimes it's clear what such a map is: for the cubic kernel we considered above, we used $\mathcal{G} = \mathbb{R}^4$ and $\phi(x) = (\sqrt{c^3}, \sqrt{3c^2}x, \sqrt{3c}x^2, x^3)$. Sometimes, though, it's not obvious for a given k whether there is such a map or not.

The definition implies that we need $k(x, x') = k(x', x)$, and that $k(x, x) \geq 0$. But those are only necessary, not sufficient.

Unfortunately people are very inconsistent about terminology around positive definiteness. For matrices, "positive semi-definite" unambiguously means the eigenvalues are nonnegative, and "strictly positive definite" unambiguously means eigenvalues are all positive, but "positive definite" might mean either. Some people get annoyed if you try to say "positive semi-definite kernel function," though.

THEOREM 11.9 ([Aro50]). A function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a positive definite kernel if and

only if for all $m \geq 1$ and $x_1, \dots, x_m \in \mathcal{X}$, the kernel matrix
$$\begin{bmatrix} k(x_1, x_1) & \dots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \dots & k(x_m, x_m) \end{bmatrix} \in \mathbb{R}^{m \times m}$$
 is positive semi-definite.

Recall that a positive semi-definite matrix can be equivalently characterized as:

- For all $\alpha \in \mathbb{R}^m$, $\alpha^T K \alpha \geq 0$.
- All eigenvalues of K are nonnegative.
- $K = LL^T$ for some $L \in \mathbb{R}^{m \times m}$.

Proof (sketch). One direction is easy: if $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{G}}$, then

$$\alpha^T K \alpha = \sum_{i=1}^m \sum_{j=1}^m \alpha_i \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{G}} \alpha_j = \left\| \sum_{i=1}^m \alpha_i \phi(x_i) \right\|_{\mathcal{G}}^2 \geq 0.$$

To show the other direction, given a k satisfying this property, we'll construct a space \mathcal{F} : the reproducing kernel Hilbert space.

We'll start by building a "pre-Hilbert space" \mathcal{F}_0 , containing functions $\mathcal{X} \rightarrow \mathbb{R}$. Start by defining the functions $\varphi(x) = [x' \mapsto k(x, x')]$ for all x . Then, let \mathcal{F}_0 be the set of all linear combinations of these functions, $\sum_{i=1}^m \alpha_i \varphi(x_i)$ for any $m \geq 0$, $x_1, \dots, x_m \in \mathcal{X}$, $\alpha_1, \dots, \alpha_m \in \mathbb{R}$. Define an inner product by

$$\left\langle \sum_{i=1}^m \alpha_i \varphi(x_i), \sum_{j=1}^n \beta_j \varphi(x'_j) \right\rangle_{\mathcal{F}_0} = \sum_{i=1}^m \sum_{j=1}^n \alpha_i \beta_j k(x_i, x'_j).$$

This satisfies the required linearity and nonnegativity properties to be an inner product. It also has the reproducing properties that we expect:

$$\langle \varphi(x), \varphi(x') \rangle_{\mathcal{F}_0} = k(x, x') \quad \langle f, \varphi(x) \rangle_{\mathcal{F}_0} = f(x).$$

Notice also that this is well-defined in the sense that it's representation-independent:

$$\left\langle \sum_{i=1}^m \alpha_i \varphi(x_i), f' \right\rangle_{\mathcal{F}_0} = \sum_{i=1}^m \alpha_i \langle \varphi(x_i), f' \rangle_{\mathcal{F}_0} = \sum_{i=1}^m \alpha_i f'(x_i),$$

which doesn't depend on how we wrote f' as a linear combination, just on its values.

The only thing left is that we need \mathcal{F}_0 to be complete: it's conceivable that not all Cauchy sequences have limits in this space. So, we construct the RKHS as the completion of \mathcal{F}_0 : just add the limits in, defining their inner products as limits of the inner products of the sequence (which is guaranteed to exist since the sequence is Cauchy and \mathbb{R} is complete). So, not all $f \in \mathcal{F}$ can be written as $\sum_{i=1}^n \alpha_i \varphi(x_i)$, but you can always get arbitrarily close (in the distance defined by $\|\cdot\|_{\mathcal{F}}$) to f with things of that form.

After checking all the details work out, we've constructed a Hilbert space and a feature map for any k . \square

(There are also other ways to define an RKHS; it turns out each RKHS has a unique kernel, and each kernel has a unique RKHS, though there could be more than Hilbert space aligning with the definition.)

11.3.1 Special case: linear kernel

If we use $k(x, x') = x \cdot x'$ for $x \in \mathbb{R}^d$, then $\varphi(x) = [x' \mapsto x' \cdot x]$ is just a linear function with weight x . Also,

$$\|\varphi(x)\|_{\mathcal{F}} = \sqrt{\langle \varphi(x), \varphi(x) \rangle_{\mathcal{F}}} = \sqrt{k(x, x)} = \|x\|.$$

So everything we've done with linear predictors can be thought of as operating in the RKHS corresponding to a linear kernel. This is often a useful thing to think about if you're looking at some complicated kernel expression: see what it'd be with a linear kernel.

11.4 OPTIMIZING IN THE RKHS

THEOREM 11.10 (Representer theorem). *If \mathcal{F} is an RKHS with feature map φ , then for any function $L : \mathbb{R}^m \rightarrow \mathbb{R}$ and any nondecreasing function $R : \mathbb{R} \rightarrow \mathbb{R} \cup \{\infty\}$,*

$$\arg \min_{f \in \mathcal{F}} L(f(x_1), \dots, f(x_m)) + R(\|f\|)$$

contains a solution of the form $f = \sum_{i=1}^m \alpha_i \varphi(x_i)$, where $S = (x_1, \dots, x_m)$. If R is strictly increasing, all solutions are of this form.

Notice that $\arg \min_{f: \|f\|_{\mathcal{F}} \leq B} L_S(f)$ fits this form: use $R(t) = \begin{cases} 0 & t \leq B \\ \infty & t > B \end{cases}$.

Proof. Let \mathcal{F}_{\parallel} be the subspace of \mathcal{F} spanned by $\{\varphi(x_i)\}_{i=1}^m$, and \mathcal{F}_{\perp} its orthogonal complement. Then any element of \mathcal{F} can be uniquely decomposed into $f_{\parallel} + f_{\perp}$, where $f_{\parallel} \in \mathcal{F}_{\parallel}$, $f_{\perp} \in \mathcal{F}_{\perp}$, and $\langle f_{\parallel}, f_{\perp} \rangle_{\mathcal{F}} = 0$. Now, since

$$f(x_i) = \langle f, \varphi(x_i) \rangle_{\mathcal{F}} = \langle f_{\parallel} + f_{\perp}, \varphi(x_i) \rangle_{\mathcal{F}} = \langle f_{\parallel}, \varphi(x_i) \rangle_{\mathcal{F}} + \underbrace{\langle f_{\perp}, \varphi(x_i) \rangle_{\mathcal{F}}}_0,$$

the L component only depends on f_{\parallel} . Also,

$$\|f\|_{\mathcal{F}}^2 = \|f_{\parallel}\|_{\mathcal{F}}^2 + \|f_{\perp}\|_{\mathcal{F}}^2 + 2 \underbrace{\langle f_{\parallel}, f_{\perp} \rangle_{\mathcal{F}}}_0 = \|f_{\parallel}\|_{\mathcal{F}}^2 + \|f_{\perp}\|_{\mathcal{F}}^2.$$

Thus, having a nonzero value of f_{\perp} does not change L , and cannot help R . If R is strictly increasing, it can only hurt the overall objective. \square

That is, *any* problem will have a solution of the form $w = \sum_i \alpha_i \varphi(x_i)$. This allows us to reduce optimization in \mathcal{F} – potentially infinite-dimensional – to optimization over $\alpha \in \mathbb{R}^m$.

11.4.1 Example: kernel ridge regression

Consider the problem

$$\min_{h \in \mathcal{F}} L_S^{sq}(h) + \lambda \|h\|_{\mathcal{F}}^2 \quad (11.1)$$

for $\lambda > 0$. First off, with a linear kernel, this becomes just plain ridge regression $\min_w L_S^{sq}(x \mapsto w \cdot x) + \lambda \|w\|^2$.

We know that all solutions will be of the form $\sum_{i=1}^m \alpha_i \varphi(x_i)$, so (11.1) is equivalent to

$$\min_{\alpha \in \mathbb{R}^m} L_S^{sq} \left(\sum_i \alpha_i \varphi(x_i) \right) + \lambda \left\| \sum_i \alpha_i \varphi(x_i) \right\|_{\mathcal{F}}^2. \quad (11.2)$$

The second term here is just

$$\left\| \sum_i \alpha_i \varphi(x_i) \right\|_{\mathcal{F}}^2 = \sum_{i,j} \alpha_i k(x_i, x_j) \alpha_j = \alpha^{\top} \mathbf{K}_{|S_x} \alpha,$$

where $K|_{S_x} \in \mathbb{R}^{m \times m}$ is the kernel matrix on S_x , as in Theorem 11.9. For the first term, notice that

$$\sum_i \alpha_i k(x_i, x_j) = \alpha^\top K|_{S_x} e_j$$

where $e_j \in \mathbb{R}^m$ is the j th standard basis vector. Then

$$L_S^{sq} \left(\sum_i \alpha_i \varphi(x_i) \right) = \frac{1}{m} \sum_i (\alpha^\top K|_{S_x} e_i - y_i)^2 = \frac{1}{m} \|K\alpha - y\|_{\mathbb{R}^m}^2.$$

Thus the overall problem is

$$\begin{aligned} \hat{\alpha} &\in \arg \min_{\alpha} \frac{1}{m} \alpha^\top K|_{S_x} K|_{S_x} \alpha - \frac{2}{m} y^\top K|_{S_x} \alpha + \frac{1}{m} y^\top y + \lambda \alpha^\top K|_{S_x} \alpha \\ &= \arg \min_{\alpha} \alpha^\top K|_{S_x} (K|_{S_x} + m\lambda I) \alpha - 2y^\top K|_{S_x} \alpha. \end{aligned}$$

Setting the gradient to zero gives that we want

$$K|_{S_x} (K|_{S_x} + m\lambda I) \alpha = K|_{S_x} y,$$

which is achieved by

$$\hat{\alpha} = (K|_{S_x} + m\lambda I)^{-1} y.$$

When $\lambda > 0$ this inverse is guaranteed to exist, since $K|_{S_x}$ is positive semidefinite, so $K|_{S_x} + m\lambda$ has all eigenvalues at least $m\lambda$.

We can also make predictions on an arbitrary test point with

$$\left\langle \sum_i \hat{\alpha}_i \varphi(x_i), \varphi(x) \right\rangle_{\mathcal{F}} = \sum_i \hat{\alpha}_i k(x_i, x) = \hat{\alpha} \cdot \begin{bmatrix} k(x_1, x) \\ \vdots \\ k(x_m, x) \end{bmatrix}.$$

It's worth checking for yourself that this agrees with standard ridge regression. (You might have to use the [Woodbury matrix identity](#) to line them up, since usual expressions for ridge regression invert a $d \times d$ matrix instead of an $m \times m$ one. In 340, we called this version the “other normal equations.”)

People sometimes call this transformed version a dual form, especially e.g. for kernel ridge regression. While “dual” isn't necessarily a strictly defined term, note that it's not a Lagrange dual.

We often won't be able to solve things in closed form like we can for kernel ridge regression. But the representer theorem will still be helpful for any problem of the right form; we just still might have to run an optimization algorithm like gradient descent on the α variables.

11.5 OTHER KERNELS

The most common kernel people use is the Gaussian kernel, also called the “square exponential” or “exponentiated quadratic” by some communities:

$$k(x, x') = \exp\left(-\frac{1}{2\sigma^2} \|x - x'\|^2\right).$$

My preferred way to prove this is a kernel goes through the following construction:

PROPOSITION 11.11. *Let k, k_1, k_2, \dots be positive definite kernels on \mathcal{X} . Then the following are all also positive definite kernels:*

1. $\gamma k = (x, x') \mapsto \gamma k(x, x')$ for any $\gamma > 0$.

2. $k_1 + k_2 = (x, x') \mapsto k_1(x, x') + k_2(x, x')$.
3. $k_1 k_2 = (x, x') \mapsto k_1(x, x') k_2(x, x')$.
4. $k^n = (x, x') \mapsto k(x, x')^n$ for any nonnegative integer n .
5. $k_\infty = (x, x') \mapsto \lim_{n \rightarrow \infty} k_n(x, x')$, when the limit always exists.
6. $e^k = (x, x') \mapsto \exp(k(x, x'))$.
7. $(x, x') \mapsto f(x)k(x, x')f(x')$ for any function $f : \mathcal{X} \rightarrow \mathbb{R}$.
8. $(x, x') \mapsto k'(f(x), f(x'))$ for any function $f : \mathcal{X} \rightarrow \mathcal{X}'$ and k' a kernel on \mathcal{X}' .

Proof. Let $\varphi, \varphi_1, \varphi_2, \dots$ be the feature maps for these kernels, and K, K_1, K_2, \dots the kernel matrices for arbitrary $(x_1, \dots, x_m) \in \mathcal{X}^m$.

1. Use the feature map $x \mapsto \sqrt{\gamma} \phi$.
2. $\alpha^\top (K_1 + K_2) \alpha = \alpha^\top K_1 \alpha + \alpha^\top K_2 \alpha \geq 0$.
3. This is called the **Schur product theorem**. Define independent multivariate normal random vectors $V \sim \mathcal{N}(0, K_1)$ and $W \sim \mathcal{N}(0, K_2)$. Let $V \odot W$ be the elementwise product of V and W ; this has covariance matrix $K_1 \odot K_2$, and covariances must be psd.
4. Iteratively apply the previous property; also, k^0 has feature map $x \mapsto 1$.
5. $\alpha^\top K_\infty \alpha = \alpha^\top [\lim_{n \rightarrow \infty} K_n] \alpha = \lim_{n \rightarrow \infty} \alpha^\top K_n \alpha \geq 0$.
6. Use $\exp(k(x, x')) = \lim_{N \rightarrow \infty} \sum_{n=0}^N \frac{1}{n!} k(x, x')^n$ and the previous properties.
7. Use the feature map $x \mapsto f(x) \varphi(x)$.
8. Use the feature map $x \mapsto \varphi'(f(x))$. □

To get the Gaussian kernel, notice that

$$\exp\left(-\frac{1}{2\sigma^2} \|x - x'\|^2\right) = \exp\left(-\frac{1}{2\sigma^2} \|x\|^2\right) \exp\left(\frac{1}{\sigma^2} x \cdot x'\right) \exp\left(-\frac{1}{2\sigma^2} \|x'\|^2\right)$$

and apply the properties above.

The Gaussian kernel is universal; you can prove this fairly immediately via Stone-Weierstrass (Theorem 10.10).

The Gaussian is *not* always the best kernel, particularly in high dimensions. Functions in \mathcal{F} for a Gaussian kernel are very smooth; the Matérn kernel is preferred in some settings where rougher functions are expected. Another good general-purpose kernel is the *distance kernel* [SSGF13]

$$k(x, x') = \rho(x, O) + \rho(x', O) - \rho(x, x')$$

where ρ is a (semi)metric, and $O \in \mathcal{X}$ is some arbitrary center point, perhaps 0. This kernel isn't actually universal [SSGF13, Proposition 35], but it is "almost universal" and works well in various settings.

If you have a good (e.g. deep) feature extractor ψ , using a kernel of the form $k(\psi(x), \psi(x'))$ can often be a good idea. This usually won't be universal, but that usually doesn't matter for the particular problem you're looking at.

11.5.1 Some properties

PROPOSITION 11.12. *Consider a kernel k with RKHS \mathcal{F} . Then*

$$\text{Rad}\left(\left\{f \in \mathcal{F} : \|f\|_{\mathcal{F}} \leq B\right\}\Big|_{S_x}\right) \leq \frac{B}{\sqrt{m}} \sqrt{\frac{1}{m} \sum_{i=1}^m k(x_i, x_i)}.$$

Proof. The analysis in Section 5.2.2 carries through exactly when replacing x_i with $k(x_i, \cdot) \in \mathcal{F}$, in which case $\|\phi(x_i)\|^2 = \langle k(x_i, \cdot), k(x_i, \cdot) \rangle_{\mathcal{F}} = k(x_i, x_i)$. \square

For many kernels, such as the Gaussian, $k(x, x) = 1$ no matter the choice of x . This makes it even simpler to handle than for the linear case, since we don't care about the data distribution.

This is a case where Rademacher analyses are *much* better than straightforward uses of covering numbers, since for infinite-dimensional kernels like the Gaussian the covering number of the sphere is infinite [Isr15].

PROPOSITION 11.13. *Let $f \in \mathcal{F}$, the RKHS with kernel k . Then*

$$|f(x)| \leq \|f\|_{\mathcal{F}} \sqrt{k(x, x)} \quad |f(x) - f(x')| \leq \|f\|_{\mathcal{F}} \sqrt{k(x, x) + k(x', x') - 2k(x, x')}.$$

Proof. We have by the representer property and Cauchy-Schwartz that

$$|f(x)| = |\langle f, \varphi(x) \rangle_{\mathcal{F}}| \leq \|f\|_{\mathcal{F}} \|\varphi(x)\|_{\mathcal{F}}.$$

Similarly,

$$|f(x) - f(x')| = |\langle f, \varphi(x) - \varphi(x') \rangle_{\mathcal{F}}| \leq \|f\|_{\mathcal{F}} \sqrt{k(x, x) + k(x', x') - 2k(x, x')}. \quad \square$$

Many more properties of this kind are available. For *shift-invariant* kernels, $k(x, x') = \kappa(x - x')$, a lot is available via Fourier properties of κ .

We've only scratched the surface here. We'll touch on kernels again through the rest of the course, but if you want more, Chapter 7 of [Bach24] goes in some more depth, and [SC08] is a classic very deep/mathematically thorough reference. Bayesian-oriented people might also want to see connections to Gaussian Processes [RW06; KHSS18], which are very much "almost the same thing" from a slightly different point of view.

12 Is ERM enough?

See slides at <https://djsutherland.ml/slides/532d-24w1-erm-not-enough.pdf>.

13 Optimization

We haven't yet really talked in this course about any optimization algorithms to actually *implement* our learning algorithms ERM, RLM, or SRM.

By far the most common optimization algorithm used in machine learning is (stochastic) gradient descent and its variants. Due to time this year, we're only going to talk about full-batch gradient descent, and point to papers that discuss stochastic variants. (This is a major area of research in the intersection between learning theory and optimization, which these days are becoming more integrated.) For much much more about optimization, some good resources are graduate courses by Michael Friedlander (CPSC 536M) and Mark Schmidt (CPSC "5XX"), the books of Boyd and Vandenberghe [BV04], Nocedal and Wright [NW06], and Bubeck [Bub15], and the recent survey of Garrigos and Gower [GG23]. Chapter 14 of Shalev-Shwartz and Ben-David [SSBD14] also gives an approachable account of projected stochastic subgradient descent, which generalizes what we're talking about here.

13.1 GRADIENT DESCENT

Gradient descent tries to find $\min_w f(w)$ for some function f , such as $L_S(f_w)$. Here w should be some finite-dimensional parameter; in kernel methods, we'd typically use the representer theorem, though there's also something called "kernel gradient descent."

We start at some initial point w_1 , often either 0 or a sample from, say, $\mathcal{N}(0, \sigma^2 I)$. We then update according to the rule

$$w_{t+1} = w_t - \eta_t \nabla f(w_t);$$

$\eta_t > 0$ is known as either the "learning rate" or the "step size," although note that it's not actually the size of the step since $\|w_{t+1} - w_t\| = \eta_t \|\nabla f(w_t)\|$.

One way to motivate this is to say that we should only "trust" the gradient direction locally, and then should re-check it regularly. Another way is to notice that this update actually minimizes the local quadratic approximation given by

$$g(w) = f(w_t) + \langle \nabla f(w_t), w - w_t \rangle + \frac{1}{2\eta} \|w - w_t\|^2;$$

if f is $\frac{1}{\eta}$ -strongly convex, then g will be a global lower bound for f . Even if not, though, it'll be an okay approximation locally.

We repeat this until we decide to stop, after T steps, and then return a result: this might be w_T (the "last iterate"), $\bar{w} = \frac{1}{T} \sum_{t=1}^T w_t$ (the "average iterate"), $w_{\hat{t}}$ for $\hat{t} \in \arg \min_{t \in [T]} f(w_t)$ (the "best iterate"), the best iterate according to a validation set, or some other scheme.

If instead of $\frac{1}{2\eta} \|w - w_t\|^2$ we use $\frac{1}{2}(w - w_t)\nabla^2 f(w_t)(w - w_t)$, i.e. the second-order Taylor expansion, this is called Newton's method. Each step of Newton's method often improves your loss much more than gradient descent, but each step is also much more computationally expensive.

We'll usually assume that η_t is some constant η , independent of the data, and that we optimize for a fixed number of steps T , also chosen independently of the data. In practice, other schemes are probably better; for instance, it's often better to use a *backtracking* scheme to adaptively choose η_t , or to otherwise have some kind of learning rate schedule that decreases over time.

13.2 β -SMOOTH FUNCTIONS

A common assumption in optimization is that the target function is β -smooth:

Note that this is not what analysts mean when they say a "smooth function" (i.e. infinitely differentiable).

DEFINITION 13.1. We say a function f is β -smooth if it is differentiable everywhere, and its gradient ∇f is β -Lipschitz.

PROPOSITION 13.2. If f is twice-differentiable, it is β -smooth iff for all w in the interior of its domain, all eigenvalues of the Hessian of f at x have absolute value at most β : $-\beta I \preceq \nabla^2 f(w) \preceq \beta I$.

The notation $A \succeq 0$ means "is positive-semi-definite"; $A \succeq B$ means that $A - B$ is positive-semi-definite.

Proof. When f is twice-differentiable and β -smooth, we have by Taylor's theorem that for any vector δ ,

$$\nabla f(w + \delta) = \nabla f(w) + \nabla^2 f(w)\delta + \mathcal{O}(\|\delta\|^2).$$

Thus by the triangle inequality,

$$\|\nabla^2 f(w)\delta\| \leq \|\nabla f(w + \delta) - \nabla f(w)\| + \mathcal{O}(\|\delta\|^2).$$

Divide through by $\|\delta\|$ and apply that ∇f is β -Lipschitz:

$$\frac{\|\nabla^2 f(w)\delta\|}{\|\delta\|} \leq \frac{\|\nabla f(w + \delta) - \nabla f(w)\|}{\|\delta\|} + \mathcal{O}(\|\delta\|) \leq \beta + \mathcal{O}(\|\delta\|).$$

Now suppose v is a (unit-norm) eigenvector of $\nabla^2 f(w)$ with eigenvalue λ , and plug in $\delta = tv$ for a scalar t , so that $\|\delta\| = |t|$. Then $\|\nabla^2 f(w)tv\| = \|\lambda tv\| = |\lambda| |t|$. This gives us that $|\lambda| \leq \beta + \mathcal{O}(t)$. Taking $t \rightarrow 0$ gives that $|\lambda| \leq \beta$.

The other direction is a vector-valued version of Lemma 4.8. □

PROPOSITION 13.3. Suppose f is β -smooth. Then for any w and w' in its domain,

$$|f(w') - f(w) - \langle \nabla f(w), w' - w \rangle| \leq \frac{1}{2} \beta \|w - w'\|^2 :$$

its deviation from its tangent planes is upper-bounded by a quadratic.

Proof. Use x_0 for w and x_1 for w' . Then for any x_0, x_1 , let $x_\alpha = (1 - \alpha)x_0 + \alpha x_1$ for all $\alpha \in (0, 1)$, and define $g : [0, 1] \rightarrow \mathbb{R}$ by $g(\alpha) = f(x_\alpha)$. Notice that $g'(\alpha) =$

$\langle \nabla f(x_\alpha), x_1 - x_0 \rangle$, and so by the fundamental theorem of calculus we have

$$\begin{aligned} f(x_1) - f(x_0) &= g(1) - g(0) = \int_0^1 g'(\alpha) d\alpha \\ &= \int_0^1 \langle \nabla f(x_\alpha) - \underbrace{\nabla f(x_0) + \nabla f(x_0)}_0, x_1 - x_0 \rangle d\alpha \\ &= \langle \nabla f(x_0), x_1 - x_0 \rangle + \int_0^1 \langle \nabla f(x_\alpha) - \nabla f(x_0), x_1 - x_0 \rangle d\alpha. \end{aligned}$$

Thus

$$\begin{aligned} |f(x_1) - f(x_0) - \langle \nabla f(x_0), x_1 - x_0 \rangle| &= \left| \int_0^1 \langle \nabla f(x_\alpha) - \nabla f(x_0), x_1 - x_0 \rangle d\alpha \right| \\ &\leq \int_0^1 |\langle \nabla f(x_\alpha) - \nabla f(x_0), x_1 - x_0 \rangle| d\alpha \\ &\leq \int_0^1 \|\nabla f(x_\alpha) - \nabla f(x_0)\| \|x_1 - x_0\| d\alpha \\ &\leq \int_0^1 \beta \|x_\alpha - x_0\| \|x_1 - x_0\| d\alpha; \end{aligned}$$

since $x_\alpha - x_0 = (1 - \alpha)x_0 + \alpha x_1 - x_0 = \alpha(x_1 - x_0)$, this is

$$|f(x_1) - f(x_0) - \langle \nabla f(x_0), x_1 - x_0 \rangle| \leq \beta \|x_1 - x_0\|^2 \int_0^1 \alpha d\alpha = \frac{1}{2} \beta \|x_1 - x_0\|^2. \quad \square$$

LEMMA 13.4 (Descent lemma). *Let $w^+ = w - \eta \nabla f(w)$ for a β -smooth function f , where $\eta < 2/\beta$. Then $f(w) - f(w^+) \geq \eta(1 - \frac{1}{2}\eta\beta) \|\nabla f(w)\|^2$, and hence either $\nabla f(w) = 0$ or $f(w^+) < f(w)$.*

Proof. By Proposition 13.3, we have

$$\begin{aligned} f(w^+) &\leq f(w) + \langle \nabla f(w), w^+ - w \rangle + \frac{1}{2} \beta \|w^+ - w\|^2 \\ &= f(w) - \eta \langle \nabla f(w), \nabla f(w) \rangle + \frac{1}{2} \beta \|-\eta \nabla f(w)\|^2 \\ &= f(w) - \eta \left(1 - \frac{1}{2} \eta \beta\right) \|\nabla f(w)\|^2. \end{aligned}$$

Since we assumed $\eta < 2/\beta$, $1 - \eta\beta/2 > 0$. The claim follows. \square

So, this means that gradient descent with a small-enough learning rate is a “descent method”: each step decreases the objective.

For convex functions, a point with $\nabla f(w) = 0$ is a global min. But for nonconvex functions, we can only say that it's a stationary point: it might be a local but non-global minimizer, or a saddle point. (A local max could only happen if we happened to initialize exactly on it.)

13.3 ASIDE: CONVEX FUNCTIONS

For convex functions in particular (with a slightly smaller learning rate), we can use the following lemma to help in a proof of overall convergence: this lemma relates the improvement of the descent lemma to how much closer a step gets us to some “target point” (presumably the minimizer) w^* :

LEMMA 13.5. *Let f be a convex, β -smooth function, and suppose that $\eta \leq 1/\beta$. Let w, w^* be arbitrary points in the interior of the domain of f . Then*

$$f(w^+) - f(w) \leq \frac{1}{2\eta} \left[\|w - w^*\| - \|w - \eta \nabla f(w) - w^*\| \right].$$

Proof. The first-order characterization of convexity implies that $f(w^*) \geq f(w) - \langle \nabla f(w), w^* - w \rangle$, or equivalently $f(w) \leq f(w^*) + \langle \nabla f(w), w - w^* \rangle$. Thus, starting from Lemma 13.4 and using $\eta \leq 1/\beta$,

$$\begin{aligned} f(w^+) &\leq f(w) - \eta \left(1 - \frac{1}{2} \beta \eta \right) \|\nabla f(w)\|^2 \\ &\leq f(w) - \frac{1}{2} \eta \|\nabla f(w)\|^2 \\ &\leq f(w^*) + \langle \nabla f(w), w - w^* \rangle - \frac{1}{2} \eta \|\nabla f(w)\|^2 \\ &= f(w^*) + \frac{1}{2\eta} \left[2\eta \langle \nabla f(w), w - w^* \rangle - \eta^2 \|\nabla f(w)\|^2 \right] \\ &= f(w^*) + \frac{1}{2\eta} \left[\|w - w^*\|^2 - \|w - w^*\|^2 + 2\eta \langle \nabla f(w), w - w^* \rangle - \eta^2 \|\nabla f(w)\|^2 \right] \\ &= f(w^*) + \frac{1}{2\eta} \|w - w^*\|^2 - \frac{1}{2\eta} \left[\|w - w^*\|^2 - 2\eta \langle \nabla f(w), w - w^* \rangle + \eta^2 \|\nabla f(w)\|^2 \right] \\ &= f(w^*) + \frac{1}{2\eta} \|w - w^*\|^2 - \frac{1}{2\eta} \left\| (w - w^*) - \eta \nabla f(w) \right\|^2 \\ &= f(w^*) + \frac{1}{2\eta} \left(\|w - w^*\|^2 - \|w^+ - w^*\|^2 \right). \quad \square \end{aligned}$$

PROPOSITION 13.6. *Let f be convex and β -smooth, with $\eta \leq 1/\beta$. Then the procedure that initializes at w_0 and then sets $w_t = w_{t-1} - \eta \nabla f(w_t)$ satisfies for all $T \geq 1$ that*

$$f(w_T) - f(w^*) \leq \frac{1}{2\eta T} \|w_0 - w^*\|^2,$$

and also that

$$f\left(\frac{1}{T} \sum_{t=1}^T w_t\right) - f(w^*) \leq \frac{1}{2\eta T} \|w_0 - w^*\|^2.$$

Proof. For each step t ,

$$f(w_t) - f(w^*) \leq \frac{1}{2\eta} (\|w_{t-1} - w^*\|^2 - \|w_t - w^*\|^2).$$

Using the descent lemma and then Lemma 13.5, we know that

$$\begin{aligned} f(w_T) - f(w^*) &\leq \frac{1}{T} \sum_{t=1}^T (f(w_t) - f(w^*)) \\ &\leq \frac{1}{2\eta T} \sum_{t=1}^T (\|w_{t-1} - w^*\|^2 - \|w_t - w^*\|^2) \\ &= \frac{1}{2\eta T} [\|w_0 - w^*\|^2 - \|w_1 - w^*\|^2 + \|w_1 - w^*\|^2 - \dots - \|w_T - w^*\|^2] \\ &= \frac{1}{2\eta T} (\|w_0 - w^*\|^2 - \|w_T - w^*\|^2) \\ &\leq \frac{1}{2\eta T} \|w_0 - w^*\|^2. \end{aligned}$$

By Jensen's inequality, $f\left(\frac{1}{T} \sum_{t=1}^T w_t\right) \leq \frac{1}{T} \sum_{t=1}^T f(w_t)$, and so the same bound applies. \square

13.3.1 Aside: SGD non-convex convergence

The analysis above can be pretty-easily extended to SGD; see e.g. Chapter 14 of Shalev-Shwartz and Ben-David [SSBD14] or the recent survey of Garrigos and Gower [GG23]. It can be generalized further, though more complicatedly, to show that even SGD eventually reaches a stationary point, even for non-convex functions:

PROPOSITION 13.7 (Corollary 1 of [KR23]). *Let $\inf_x f(x) \geq f^{\text{inf}} \in \mathbb{R}$ be β -smooth. Let $\hat{g}_t \mid x_t$ be independent such that $\mathbb{E}[\hat{g}_t \mid x_t] = \nabla f(x_t)$ and*

$$\mathbb{E}[\|\hat{g}_t\|^2 \mid x_t] \leq 2A(f(x_t) - f^{\text{inf}}) + B\|\nabla f(x_t)\|^2 + C$$

for some $A, B, C \geq 0$. Fix $\varepsilon > 0$, and pick $\eta = \min\left\{\frac{1}{\sqrt{\beta AT}}, \frac{1}{\beta B}, \frac{\varepsilon}{2\beta C}\right\}$. Initialize stochastic gradient descent at x_1 , with $\delta_1 = f(x_1) - f^{\text{inf}}$, and $x_{t+1} = x_t - \eta\hat{g}_t$. As long as $T \geq \frac{12\delta_1\beta}{\varepsilon^2} \max\left\{B, \frac{12\delta_1A}{\varepsilon^2}, \frac{2C}{\varepsilon^2}\right\}$, it holds that $\min_{1 \leq t \leq T} \mathbb{E}[\|\nabla f(x_t)\|] \leq \varepsilon$.

That is, the *best iterate* achieves ε suboptimality (in expectation) with $\mathcal{O}(1/\varepsilon^4)$ steps. The assumption on \hat{g}_t is satisfied for example if the \hat{g}_t have a bounded variance, or if we use subsampling for a Lipschitz loss, or various other settings.

13.4 ARE DEEP NETWORKS β -SMOOTH?

Is $f(w) = L_S(h_w)$ for h_w a class of deep networks β -smooth?

Consider the very simple network

$$h_{W,v}(x) = v \cdot \sigma(Wx),$$

where σ is itself β -smooth. Then the square loss for a single data point is

$$f(W, v) = (v^\top \sigma(Wx) - y)^2 = v^\top \sigma(Wx) \sigma(Wx)^\top v - 2y \sigma(Wx)^\top v + y^2,$$

and we have

$$\nabla_v f(W, v) = 2(\sigma(Wx)^\top v - y) \sigma(Wx)$$

$$\nabla_v^2 f(W, v) = 2\sigma(Wx) \sigma(Wx)^\top.$$

If this is unfamiliar, try looking at individual partial derivatives to see that they line up.

Autodiff is nice... The Jacobian with W is more annoying, since we'd have to flatten W and reshape and stuff. But the overall Hessian of f with respect to its input parameters will have $\nabla_v^2 f$ as a block in it, and so its largest eigenvalue will depend on W : if σ is the ReLU or something similar, then large values of W will result in much larger Hessians. Thus the loss is only going to be fully β -smooth if you bound the set of possible W s, but for any particular parameters it's going to be "locally" smooth.

Notice that the descent lemma doesn't actually need a global upper bound on the smoothness, just along the path from x_t to x_{t+1} . So, intuitively, we should roughly expect (stochastic) gradient descent to reach a stationary point of the loss as long as $\nabla^2 f$ doesn't blow up, i.e. in typical situations as long as none of the parameters blows up.

Aside: edge of stability

So, if we're optimizing a deep network with a fixed learning rate η , whether the descent lemma applies or not – whether gradient descent is "stable" or not – depends on whether $\eta < \frac{2}{\beta}$, or more relevantly $\beta < \frac{2}{\eta}$, for the "local" value of β . We can roughly get this local value of β by just checking the largest eigenvalue of $\nabla^2 f(x_t)$, and see whether it stays in a "stable" regime or not.

Note that the "local β " might be larger than $\max(\nabla^2 f(x_t), \nabla^2 f(x_{t+1}))$: you might go through a sharper point on the way. For instance, consider $f(x) = |x|$ on the reals: $f''(x) = 0$ for all $x \neq 0$, but the descent lemma might not apply when you switch signs, since you go through 0 which has "infinite second derivative."

Cohen et al. [Coh+21] demonstrated that in fact, optimization typically exhibits "progressive sharpening" where β increases up to $2/\eta$, then hovers around there on the "edge of stability" [also see Fox23]. Damian, Nichani, and Lee [DNL23] have recently proposed a mechanism for how this happens, based on Taylor expansions of the training process.

13.5 IS A STATIONARY POINT ENOUGH?

One model we can look at is deep linear nets, $f(x) = w_d W_{d-1} \cdots W_2 W_1 x$. These are just linear models, but they're nonconvex and hierarchical and so exhibit some of the same behaviour as regular deep nets. It's reasonable to expect that, generally speaking, if something doesn't work on deep linear nets, it won't work on deep nonlinear nets either.

To see that they're nonconvex: consider just a depth two model on scalars, $f(x) = vwx$ for $v, w \in \mathbb{R}$. Consider square loss with the training set $S = ((1, 1))$. Then $L_S(f) = (vw - 1)^2$, whose minimizers are

$$\{(v, w) : vw = 1\} = \{(v, 1/v) : v \neq 0\}.$$

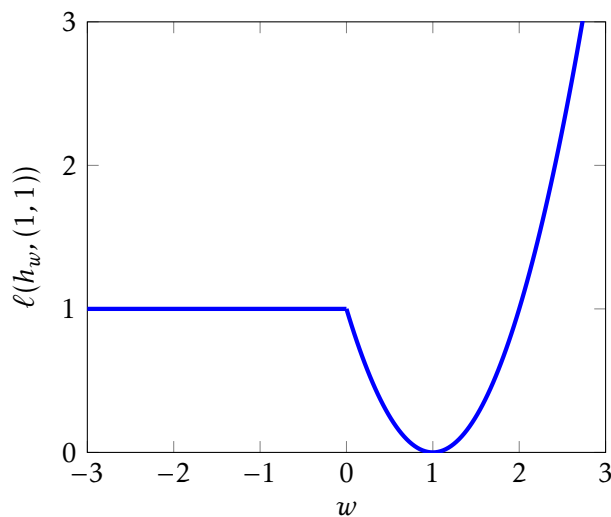
But this is *not* a convex set: it's a line in \mathbb{R}^2 with the point $(0, 0)$ cut out of it. The set of minimizers of convex functions must be convex, so therefore L_S is not convex.

It turns out that for deep linear nets:

- Fortunately, all local minima in deep linear nets are global minima [Kaw16; LvB18].
- Unfortunately, stationary points can also be saddle points – including potentially “bad” saddles with $\lambda_{\min}(\nabla^2 f) = 0$ even though they’re not local minima. (For example, x^3 has a saddle point like this at $x = 0$; they can be even worse in high dimensions.)
- Fortunately, in general, gradient descent almost surely converges to local minimizers, not saddles (or local maxes) [LSJR16].
- Unfortunately, doing so can take exponential time [Du+17].
- Fortunately, this doesn’t happen for deep linear networks, under some conditions [ACGH19].

Unfortunately, there *are* bad local minima in nonlinear networks. For a very simple example, consider the network $h : \mathbb{R} \rightarrow \mathbb{R}$ given by $h(x) = \text{ReLU}(wx)$, where $w \in \mathbb{R}$; use square loss with a single example, $(1, 1)$. Then the loss is

$$\ell(h_w, (1, 1)) = \begin{cases} (w - 1)^2 & w \geq 0 \\ 1 & w \leq 0 \end{cases}.$$



Any negative input is a (non-strict) local min (since $f(w) \geq f(v)$ for all v in a neighbourhood of w), but it’s not a global min (since $f(1) = 0$). Thus, if you start gradient descent with a negative w , it’s just stuck. In fact, bad (strict) local minima can appear for almost any activation function [DLS20], and with more units, the loss landscape has such points almost all the time.

But, do bad local minima exist for realistic networks, with realistic data? Even if they do, does SGD find them?

This is very much still an active topic of research, but we’ll see next that, in one unrealistic (but not *too* ridiculous) setting, gradient descent always finds a local minimum.

14 Neural Tangent Kernels

Continuing our study of nonconvex optimization (Chapter 13), we're going to see one setting today where we can prove global optimization for certain deep networks.

Initially, there were a series of papers that proved this in some special cases; later, it was realized that they all shared a common structure, which amounts to the following: if you initialize a very wide network appropriately and minimize the square loss with a small learning rate, your model becomes equivalent to kernel regression, with a particular kernel defined by the architecture.

14.1 SIMPLE CASE

To motivate things, let's begin with a very simple network. (We're mostly following the presentation of Telgarsky [Tel21] here.) Consider

$$h(x; W) = \frac{1}{\sqrt{N}} \sum_{j=1}^N a_j \sigma(w_j \cdot x),$$

where the a_j are fixed and the $w_j \in \mathbb{R}^d$ are the rows of a weight matrix $W \in \mathbb{R}^{N \times d}$, viewed as column vectors. We're writing W as a parameter to the function because we want to emphasize that we're changing W , not x .

We could of course absorb the \sqrt{N} into the a_j . Writing it like this scales the gradients w.r.t. a_j differently, and is called the NTK parameterization; it makes some things simpler.

Now, let's see what happens if we take a first-order Taylor expansion (a linearization) of h with respect to W , around some point \tilde{W} . (We're going to pick \tilde{W} to be our starting point for gradient descent.) This function will be linear in W but nonlinear in x ; it should be a reasonable approximation if $W \approx \tilde{W}$.

$$\begin{aligned} h_{\tilde{W}}(x; W) &= h(x; \tilde{W}) + \langle \nabla_W h(x; \tilde{W}), W - \tilde{W} \rangle \\ &= \frac{1}{\sqrt{N}} \sum_{j=1}^N a_j \left[\sigma(\tilde{w}_j \cdot x) + \langle \sigma'(\tilde{w}_j \cdot x) x, w_j - \tilde{w}_j \rangle \right] \\ &= \frac{1}{\sqrt{N}} \sum_{j=1}^N a_j \left[\underbrace{\sigma(\tilde{w}_j \cdot x)}_{\text{constant in } W} - \sigma'(\tilde{w}_j \cdot x) \tilde{w}_j \cdot x + \underbrace{\sigma'(\tilde{w}_j \cdot x) x \cdot w_j}_{\text{linear in } W} \right]. \end{aligned} \tag{14.1}$$

We can either think of this as a matrix with the Frobenius inner product $\langle A, B \rangle = \sum_{ij} A_{ij} B_{ij}$, or of flattening the parameters into a vector and using regular gradients.

Notice that we have $\text{ReLU}(t) = \text{ReLU}'(t)t$ for all t (even zero), and so for ReLU activations the term that is constant in W vanishes. In that case, we see that

$$h_{\tilde{W}}(x; W) = \langle \nabla_W h(x; \tilde{W}), W \rangle.$$

But, if we write $\phi_{\tilde{W}}(x) = \nabla_W h(x; \tilde{W})$, the function $h_{\tilde{W}}$ is of the form $\langle W, \phi_{\tilde{W}}(x) \rangle$, and

in particular the set of achievable functions

$$\{x \mapsto h_{\tilde{W}}(x; W) : W \in \mathbb{R}^{N \times d}\}$$

is exactly the RKHS of the kernel

$$k_{\tilde{W}}(x, x') = \langle \phi_{\tilde{W}}(x), \phi_{\tilde{W}}(x') \rangle = \langle \nabla_W h(x; \tilde{W}), \nabla_W h(x'; \tilde{W}) \rangle. \quad (14.2)$$

Even if we don't have ReLU activations, recalling (14.1) we can see that the *residual* $h_{\tilde{W}}(x; W) - h(x; \tilde{W})$ is an element of that same RKHS.

But, so what? This linearization is only going to be accurate when $W \approx \tilde{W}$, it doesn't tell us anything about actually training a network. . . unless?

14.1.1 Approximation quality

PROPOSITION 14.1. *For the h and $h_{\tilde{W}}$ defined above, assume σ is β -smooth. Then*

$$|h(x; W) - h_{\tilde{W}}(x; W)| \leq \frac{\beta}{2\sqrt{N}} (\max_j |a_j|) \|x\|^2 \|W - \tilde{W}\|_F^2.$$

Proof. Recall Proposition 13.3, that β -smooth functions deviate from their linearizations by at most a quadratic. Thus

$$\begin{aligned} |h(x; W) - h_{\tilde{W}}(x; W)| &\leq \frac{1}{\sqrt{N}} \sum_{j=1}^N |a_j| |\sigma(w_j \cdot x) - \sigma(\tilde{w}_j \cdot x) - \sigma'(\tilde{w}_j \cdot x)(w_j \cdot x - \tilde{w}_j \cdot x)| \\ &\leq \frac{1}{\sqrt{N}} \sum_{j=1}^N |a_j| \frac{1}{2} \beta (w_j \cdot x - \tilde{w}_j \cdot x)^2 \\ &\leq \frac{\beta}{2\sqrt{N}} (\max_j |a_j|) \sum_{j=1}^N \|w_j - \tilde{w}_j\|^2 \|x\|^2. \quad \square \end{aligned}$$

So, if N is large enough, the linearization is good even for a large radius of parameters around \tilde{W} , no matter what the data looks like (as long as it's bounded) and no matter what \tilde{W} looks like.

One thing to keep in mind, though, is that $\|W - \tilde{W}\|_F$ is also changing meaning with N : in fact, $\mathbb{E} \|\tilde{W}\|_F^2 = \mathbb{E} \sum_{j=1}^N \sum_{k=1}^d \tilde{W}_{jk}^2 = Nd$, so with standard Gaussian initializations the linearization potentially even gets worse at $W = 0$. It's thus not really obvious yet that this approximation is a "good idea." It'll turn out, though, that in the high-width regime, the optimization process doesn't have to move very much, and everything works out.

The result being *so* general in \tilde{W} is special for this very simple network; even for the same architecture but with ReLU activations, the analysis is a lot nastier.

PROPOSITION 14.2 (Lemma 4.1 of [Tel21]). *For any radius $B \geq 0$, for any fixed $x \in \mathbb{R}^d$ with $\|x\| \leq 1$, suppose we pick \tilde{W} with entries i.i.d. standard normal. Then, with probability at least $1 - \delta$ over the draw of \tilde{W} , it holds that for all $W \in \mathbb{R}^{N \times d}$ with*

$\|W - \tilde{W}\|_F \leq B$ that

$$|h(x; W) - h_{\tilde{W}}(x; W)| \leq \frac{2B^{4/3} + B \log(1/\delta)^{1/4}}{N^{1/6}}.$$

For deeper networks, the result is even more complicated and degrades with depth, but it's still going to be true that with appropriate Gaussian initializations the linearization works for an increasing radius.

14.1.2 The neural tangent kernel

What does the kernel (14.2) look like? For this shallow network, it's

$$\begin{aligned} k_{\tilde{W}}(x, x') &= \langle \phi_{\tilde{W}}(x), \phi_{\tilde{W}}(x') \rangle = \langle \nabla_W h(x; \tilde{W}), \nabla_W h(x'; \tilde{W}) \rangle \\ &= \left\langle \begin{bmatrix} a_1 x^\top \sigma'(\tilde{w}_1 \cdot x) / \sqrt{N} \\ \vdots \\ a_N x^\top \sigma'(\tilde{w}_N \cdot x) / \sqrt{N} \end{bmatrix}, \begin{bmatrix} a_1 (x')^\top \sigma'(\tilde{w}_1 \cdot x') / \sqrt{N} \\ \vdots \\ a_N (x')^\top \sigma'(\tilde{w}_N \cdot x') / \sqrt{N} \end{bmatrix} \right\rangle \\ &= \frac{1}{N} \sum_{j=1}^N a_j^2 \langle x \sigma'(\tilde{w}_j \cdot x), x' \sigma'(\tilde{w}_j \cdot x') \rangle \\ &= x \cdot x' \left[\frac{1}{N} \sum_{j=1}^N a_j^2 \sigma'(\tilde{w}_j \cdot x) \sigma'(\tilde{w}_j \cdot x') \right]. \end{aligned}$$

The $1/\sqrt{N}$ scaling hopefully makes sense now: if we also use the common setup $a_j \in \{\pm 1\}$ so that $a_j^2 = 1$, as $N \rightarrow \infty$, this converges almost surely to the kernel

$$k_\infty(x, x') = x \cdot x' \mathbb{E}_{\tilde{w} \sim \mathcal{N}(0, I)} [\sigma'(\tilde{w} \cdot x) \sigma'(\tilde{w} \cdot x')].$$

We'll call $k_{\tilde{W}}$ the *empirical neural tangent kernel at \tilde{W}* and k_∞ the *infinite neural tangent kernel*. Terminology here isn't quite standardized, though.

For the ReLU, $\sigma'(t) = \mathbf{1}(t > 0)$, and so k_∞ is $x \cdot x'$ times

$$\begin{aligned} \mathbb{E}_{\tilde{w} \sim \mathcal{N}(0, I)} [\sigma'(\tilde{w} \cdot x) \sigma'(\tilde{w} \cdot x')] &= \Pr_{\tilde{w} \sim \mathcal{N}(0, I)} (\tilde{w} \cdot x > 0 \text{ and } \tilde{w} \cdot x' > 0) \\ &= \Pr_{v \sim \text{Unif}(\text{unit sphere})} \left(v \cdot \frac{x}{\|x\|} > 0 \text{ and } v \cdot \frac{x'}{\|x'\|} > 0 \right) \quad \text{Draw a picture for this...} \\ &= \frac{\pi - \arccos\left(\frac{x \cdot x'}{\|x\| \|x'\|}\right)}{2\pi}. \end{aligned}$$

In fact, a slight tweak to this kernel is even universal [Tel21, Theorem 4.1].

14.1.3 More general networks

For more general architectures, we still use the linearization

$$h_{\tilde{w}}(x; w) = h(x; \tilde{w}) + \langle \nabla_w h(x; \tilde{w}), w - \tilde{w} \rangle,$$

where now w represents *all* the parameters in the network collected into a vector. This gives us the same kind of empirical neural tangent kernel

$$k_{\tilde{w}}(x, x') = \langle \nabla_w h(x; \tilde{w}), \nabla_w h(x'; \tilde{w}) \rangle. \quad (14.3)$$

Even for complex deep architectures (including convolutions, pooling, even attention), if we scale things according to *fan-out* “NTK parameterizations,” $k_{\tilde{w}}$ still converges to some k_∞ almost surely as it becomes infinitely wide [Yan19]. It’s also often possible (though computationally expensive) to exactly compute this kernel; the standard software is the `neural-tangents` library [Nov+20].

14.2 OPTIMIZATION

So, we know that the set of “nearby” updates to the initialization \tilde{w} is described by the set of functions in an RKHS. But maybe we break out of the “nearby” set immediately, and this is all pointless.

In some regimes, it’s true that we break the NTK approximation immediately, in practice; e.g. we can check numerically whether k_{w_0} and $k_{w_{100}}$ are very different. But, if we set things up in the right way, this doesn’t happen, and the whole gradient descent process stays near enough to w_0 that the RKHS is a good approximation.

To see this, we’ll consider square loss and take the limit of gradient descent as the step size η goes to zero, which is known as *gradient flow*. A lot of the time, the continuous limit is much easier to handle, and then you check that the discretization into actual gradient descent doesn’t take you too far away from the continuous version.

In this case, we have

$$\frac{dw_t}{dt} = -\eta \nabla_w L_S(h(\cdot; w_t)) = -\frac{2\eta}{m} \sum_{i=1}^m (h(x_i; w_t) - y_i) \nabla_w h(x_i; w_t), \quad (14.4)$$

and so if we track the prediction on a point x we get

$$\begin{aligned} \frac{d}{dt} h(x; w_t) &= \left\langle \nabla h(x; w_t), \frac{dw_t}{dt} \right\rangle \\ &= -\frac{2\eta}{m} \sum_{i=1}^m (h(x_i; w_t) - y_i) \langle \nabla_w h(x_i; w_t), \nabla_w h(x; w_t) \rangle \\ &= -\frac{2\eta}{m} \sum_{i=1}^m (h(x_i; w_t) - y_i) k_{w_t}(x_i, x). \end{aligned} \quad (14.5)$$

Suppose that k_{w_t} is constant in t . Then this would be *exactly the same dynamics* as “kernel gradient descent” for kernel regression, which we’ll define now. This is

gradient descent directly “in kernel space”:

$$\begin{aligned}
L_S(h) &= \frac{1}{m} \sum_{i=1}^m (h(x_i) - y_i)^2 \\
&= \frac{1}{m} \sum_{i=1}^m [\langle h, \varphi(x_i) \rangle \langle \varphi(x_i), h \rangle - 2y_i \langle \varphi(x_i), h \rangle + y_i^2] \\
\frac{dh}{dt} &= -\eta \nabla_h L_S(h) \\
&= -\frac{\eta}{m} \sum_{i=1}^m [k(x_i, \cdot) \langle k(x_i, \cdot), h \rangle + \langle k(x_i, \cdot), h \rangle k(x_i, \cdot) - 2y_i k(x_i, \cdot)] \\
&= -\frac{2\eta}{m} \sum_{i=1}^m (h(x_i) - y_i) k(x_i, \cdot) \tag{14.6}
\end{aligned}$$

$$\frac{d}{dt} h(x) = \left\langle \frac{dh}{dt}, k(x, \cdot) \right\rangle = -\frac{2\eta}{m} \sum_{i=1}^m (h(x_i) - y_i) k(x_i, x). \tag{14.7}$$

Note that this is not the same as using the representer theorem and doing gradient descent on the coefficients α ! That’s a different parameterization, giving different gradients.

Indeed, (14.5) and (14.7) agree if k_{w_t} is constant throughout training; so do (14.4) and (14.6), recalling the definition of the kernel (14.3). In fact, though, we can solve this ordinary differential equation (14.7) analytically [JGH18]; we get for an arbitrary prediction point that

$$h_t(x) = h_0(x) + [k(x, x_i)]_{i=1}^m (k|_{S_x})^{-1} \left(I - e^{-\eta t k|_{S_x}} \right) (S_y - h_0|_{S_x}), \tag{14.8}$$

where the matrix exponential $e^A = I + A + \frac{1}{2}A^2 + \dots$ can be found by exponentiating the singular values of A . In the limit as $t \rightarrow \infty$, $e^{-\eta t k|_{S_x}} \rightarrow 0$ as long as $k|_{S_x}$ is nonsingular (which it is for universal kernels). We can also see then that $h_t|_{S_x} \rightarrow S_y$.

This becomes equivalent to $\lim_{\lambda \rightarrow 0} \arg \min_{h \in \mathcal{F}} L_S(h) + \lambda \|h - h_0\|_{\mathcal{F}}^2$, sometimes called kernel “ridgeless” regression; it’s also like GP regression with a prior mean of h_0 .

In the limit as the network becomes infinitely wide, for appropriate initializations, $k_{w_t}|_{S_x}$ does in fact stay constant through training [Lee+19; Aro+19; COB19; YL21]. For example, Theorem 3.2 of Arora et al. [Aro+19] shows a closeness guarantee with finite width. The proof basically amounts to showing that parameters don’t change much, so the empirical NTK doesn’t change much, and so gradients throughout the network training are close to gradients from the kernel process.

This behaviour is very much a function of having the right scaling, though. For “fan-in” parameterizations that people usually actually use in practice, the kernel blows up, but it does so in a way where regression still makes sense [Lee+19; MBS23]. If you choose other scalings, it doesn’t happen at all; Chizat, Oyallon, and Bach [COB19] cover this point of view, arguing that the parameter scaling amounts to “zooming in” on the Taylor expansion so the linearization works well. Yang and Hu [YH22] also study scaling regimes other than just the kernel one.

Dealing with the random initialization $h_0(x)$ can be a problem, sometimes; it takes a while for gradient descent to counteract the noise there. Sometimes people scale the network output by a small $\varepsilon > 0$ (which also scales the gradient, but that’s good too), or use special symmetric initializations to their network parameters which guarantee $h_0(x) = 0$.

14.3 DOES THIS MEAN DEEP LEARNING IS SOLVED?

So, in these regimes, you can avoid training a network at all and just use a special kernel. That kernel is indeed often a good kernel for particular problems [Aro+20].

But, there are problems that deep learning can solve provably faster than *any* kernel method. Essentially, the NTK regime doesn't allow for feature learning, since the kernel is fixed.

One concrete example is learning a single ReLU: Yehudai and Shamir [YS19] show that kernel methods cannot learn a single ReLU unit $\text{ReLU}(w \cdot x + b)$ in square loss without RKHS norm exponential in the dimension, which implies (roughly) exponential sample complexity – but gradient descent can get it with polynomially many samples.

Malach et al. [MKAS21] give a nice characterization on when these gaps are and aren't possible.

Overall, it's definitely *not* the case that the practical success of deep learning can be explained by infinite NTK behaviour. Even so, the infinite NTK is still useful to know about:

- As far as I know, they're the only proofs that gradient descent works in nonlinear deep nets.
- It's actually a reasonable approximation in certain regimes.
- Infinite NTKs are sometimes practically useful [Aro+20; JNWB21].

Also, I think the *empirical* NTK is even more useful: I've given a talk called “[A Defense of \(Empirical\) Neural Tangent Kernels](#)”. Basically, the empirical NTK can be a really useful tool for understanding what's happening “locally” during training, even when it doesn't stay constant over the whole course of training. This can be theoretical [RGS22; RGS23], for “empirical science” understanding [For+20; MBS23], or even purely practical [WHS22; MBS22].

15 Implicit Regularization and Margins

We've seen some examples so far of settings where there's more than one empirical risk minimizer; this often happens with *interpolation*, when you can achieve $L_S(h) = 0$ in more than one way, some of which are awful, but \mathcal{A} often picks decent ones. In particular, we saw some explicit examples with polynomial regression.

One way to choose between ERMs (or near-ERMs) is regularized loss minimization, where we prefer solutions with e.g. a small norm. But often we don't do that, and we just run gradient descent to minimize $L_S(h)$. Doing this doesn't just get us *any* arbitrary ERM; it gets us a particular one, decided on by our choice of algorithm. The idea that our optimization algorithm or other such "implementation details" can actually choose for us which of the "equally valid solutions" we end up with is called the *implicit regularization* of the algorithm: we don't explicitly write down a regularizer, but the choice of algorithm has a similar effect.

It's also sometimes called the implicit bias of the algorithm, in the sense that the algorithm has a certain inductive bias towards certain kinds of solutions. That can sometimes cause confusion with the concept of the same name from social science, though, and just generally kind of imply that it's "bad" when actually often the presence of this implicit regularization is "good."

In our discussion of neural tangent kernels, we mentioned that we could solve the ODE for gradient flow to say *which* ERM we end up at in (14.8). We didn't prove this, though, and it only applied to "kernel gradient flow" which is not really the algorithm we usually use. What happens for actual problems, with finite learning rates?

15.1 GRADIENT DESCENT FOR LINEAR REGRESSION

Let's think about optimizing the function

$$f(w) = L_S^{\text{sq}}(x \mapsto w \cdot x) = \frac{1}{m} \|Xw - y\|^2,$$

where $X \in \mathbb{R}^{m \times d}$ is the matrix stacking up S_x and $y \in \mathbb{R}^m$ is the vector form of S_y .

It's possible to use this form to handle kernels, too. If there's a finite-dimensional embedding ϕ , we could just collect $\phi(x_i)$ in rows of X and find w . If we instead write $f_\alpha(x) = \sum_i \alpha_i k(x_i, x)$ and do gradient descent on α , notice the training set loss

This agrees with "kernel gradient descent" as in Chapter 14 for finite-dimensional kernels.

becomes $L_S(f_\alpha) = \frac{1}{m} \|K\alpha - y\|^2$ and so the rest of the analysis will apply with $X = K$ – which will potentially give a *different* solution than the kernel gradient descent version. Implicit regularization is highly algorithm-specific.

In any case, we have

$$\nabla f(w) = \frac{2}{m} X^T(Xw - y),$$

which notice is $\frac{2}{m} \|X^T X\|$ -smooth, so f is convex and β -smooth, thus small-learning-rate gradient descent finds a global optimum (Proposition 13.6). In the traditional $m > d$ case when X is full-rank, there's a unique solution to this problem, typically with $Xw \neq y$ but always having $X^T(Xw - y) = 0$. In high-dimensional settings $d > m$,

though, it's possible to achieve $Xw = y$ (interpolation) in infinitely many ways.

Which one does gradient descent find?

There's a more explicit (but longer) analysis for least squares, which gives some more details without relying on any general gradient descent analyses, in last year's notes.

PROPOSITION 15.1. *Let $X \in \mathbb{R}^{m \times d}$ be of rank m (implying $d \geq m$), and $y \in \mathbb{R}^m$. Suppose that $l(h, (x, y)) = l_y(h(x))$ for a differentiable function l_y such that $l_y(\hat{y}) \rightarrow 0$ implies $\hat{y} \rightarrow y$.*

Consider any iterative optimization method which begins at a point w_0 and then has updates of the form $w_{t+1} - w_t \in \text{span}\{\nabla L_S(x \mapsto w_k \cdot x) : 0 \leq k \leq t\}$. If this method converges to a global minimizer w_∞ of $L_S(x \mapsto w \cdot x)$, then

$$w_\infty = X^\top (XX^\top)^{-1} y + (I - X^\top (XX^\top)^{-1} X) w_0 = \arg \min_{w: Xw=y} \|w - w_0\|.$$

Proof. XX^\top is $m \times m$ of rank m , and so $(XX^\top)^{-1}$ exists; then $X(X^\top (XX^\top)^{-1} y) = y$. The matrix $X^\top (XX^\top)^{-1}$ is the **pseudoinverse** of X , written X^\dagger ; this then implies that $L_S(x \mapsto (X^\dagger y) \cdot x) = 0$. Since w_∞ is optimal, we must have $Xw_\infty = y$.

Now, for any w we have that

$$\nabla_w L_S(h_w) = \frac{1}{m} \sum_{i=1}^m l'_{y_i}(w \cdot x_i) x_i \in \text{span}\{x_i : i \in [m]\}.$$

This is true for each step, no matter the learning rate; it is also true e.g. if we do stochastic gradient descent based on choosing a subset of the data at each step. Thus the iterates of gradient descent must all be of the form

$$w_t = w_0 + \sum_i \alpha_i^{(t)} x_i = w_0 + X^\top \alpha^{(t)};$$

they can only ever move in the subspace spanned by the data, and otherwise stay where they started. Thus, this must also be true for the limiting point: $w_\infty = w_0 + X^\top \alpha$ for some $\alpha \in \mathbb{R}^m$.

Thus, we know that

$$X(w_0 + X^\top \alpha) = y$$

or equivalently

$$XX^\top \alpha = y - Xw_0.$$

Since we know already that XX^\top is invertible, we have that

$$\begin{aligned} \alpha &= (XX^\top)^{-1} (y - Xw_0) \\ w_\infty &= w_0 + X^\top (XX^\top)^{-1} (y - Xw_0) \\ &= X^\top (XX^\top)^{-1} y + (I - X^\top (XX^\top)^{-1} X) w_0. \end{aligned}$$

This second matrix is the orthogonal projection onto the null space of X , which can be seen e.g. by considering the SVD. The result follows by Lemma 15.2. \square

Aside: closest interpolator

LEMMA 15.2. *Let $X \in \mathbb{R}^{m \times d}$, $y \in \mathbb{R}^m$, and let Π_\perp be the orthogonal projection onto the null space of X . Then*

$$\arg \min_{w: Xw=y} \|w - w_0\| = X^\dagger y + \Pi_\perp w_0.$$

Proof. First, the set of possible interpolators must all have $y = Xw$, hence $X^\dagger y = X^\dagger Xw$. $X^\dagger X$ is exactly the orthogonal projection onto the row space of X : letting the compact SVD of X be $U\Sigma V^\top$, $X^\dagger = V\Sigma^{-1}U^\top$, and $X^\dagger X = V\Sigma^{-1}U^\top U\Sigma V^\top = V\Sigma^{-1}\Sigma V^\top = VV^\top$, which has $(VV^\top)^\top = VV^\top$ and $(VV^\top)^2 = VV^\top VV^\top = VV^\top$. Notice also that $\Pi_\perp = I - VV^\top$. Thus, $X^\dagger y = VV^\top w$ for any interpolator, and so the set of interpolators is the set $\{X^\dagger y + q : VV^\top q = 0\}$. For any such solution,

$$\begin{aligned} \|X^\dagger y + q - w_0\|^2 &= \|VV^\top(X^\dagger y + q - w_0)\|^2 + \|(I - VV^\top)(X^\dagger y + q - w_0)\|^2 \\ &= \|X^\dagger y - VV^\top w_0\|^2 + \|q - (I - VV^\top)w_0\|^2. \end{aligned}$$

The choice of q does not affect the first term, while the second is uniquely minimized by $q = (I - VV^\top)w_0$. \square

15.2 SEPARABLE LOGISTIC REGRESSION

There's another major class of loss functions not satisfying the requirement of Proposition 15.1: for instance, with logistic loss $l_y(\hat{y}) = \log(1 + \exp(-y\hat{y}))$, $l_y(\hat{y}) \rightarrow 0$ implies $\hat{y} \rightarrow y\infty$, not y .

So, let's consider logistic regression in particular: for $y_i \in \{-1, 1\}$,

$$f(w) = \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-y_i x_i^\top w)).$$

We're also going to assume that the data is *linearly separable*: there is some w^* such that $y_i x_i^\top w^* > 0$ for all i . Then, it's possible to drive $f(w)$ arbitrarily close to zero, but never to actually reach it: we only get $\log(1 + \exp(-t)) \rightarrow 0$ for $t \rightarrow \infty$, so we need $\|w\| \rightarrow \infty$. A solution of the form cw^* for $c \rightarrow \infty$ would work, but potentially so would many other solutions, since there are probably many possible perfect linear separators on this dataset. Which one does gradient descent find?

We're going to approach this informally, for time and simplicity. Soudry et al. [Sou+18] and Gunasekar et al. [GLSS18] handle it in full, and Ji and Telgarsky [JT19] approach the non-separable case; Bach [Bach24, Section 11.1.2] gives an overview including a few things we aren't covering here.

Notice that

$$\nabla f(w) = -\frac{1}{m} \sum_{i=1}^m \frac{\exp(-y_i x_i^\top w)}{1 + \exp(-y_i x_i^\top w)} y_i x_i.$$

We know that we'll get $\|w_t\| \rightarrow \infty$ from the argument above; it's reasonable to expect, then, that we'll have $\frac{w_t}{\|w_t\|} \rightarrow v$ for some $\|v\| = 1$, and $y_i x_i^\top v > 0$ for all i since otherwise we wouldn't approach a minimizer. This gives us, roughly speaking,

$$\nabla f(\|w_t\| v) \sim -\frac{1}{m} \sum_{i=1}^m \frac{\exp(-y_i \|w_t\| x_i^\top v)}{1 + \exp(-y_i \|w_t\| x_i^\top v)} y_i x_i \sim -\frac{1}{m} \sum_{i=1}^m \exp(-y_i \|w_t\| x_i^\top v) y_i x_i,$$

since $\frac{t}{1+t} = t + \mathcal{O}(t^2)$ and we'll eventually have $\exp(-y_i \|w_t\| x_i^\top v) \ll 1$.

So, eventually each gradient term gets small. Which ones are bigger than the others? The asymptotic ratio between the size of the gradient contributions from x_i and x_j is

$$\frac{\exp(-y_i \|w_t\| x_i^\top v) |y_i| \|x_i\|}{\exp(-y_j \|w_t\| x_j^\top v) |y_j| \|x_j\|} = \frac{\|x_i\|}{\|x_j\|} \exp(-\|w_t\| (y_i x_i^\top v - y_j x_j^\top v)).$$

As $\|w_t\| \rightarrow \infty$, this ratio goes to 0 if $y_i x_i^\top v > y_j x_j^\top v$, or ∞ if the order is reversed; it is $\|x_i\|/\|x_j\| \in (0, \infty)$ if and only if $y_i x_i^\top v = y_j x_j^\top v$. So, for whatever v we have, let \mathcal{I}_v be the set of indices such that $y_i x_i^\top v$ is minimized. Only these terms really matter:

$$\nabla f(\|w_t\| v) \sim -\frac{1}{m} \sum_{i \in \mathcal{I}_v} \exp(-y_i \|w_t\| x_i^\top v) y_i x_i.$$

So, if gradient descent diverges in a direction v , the dominant direction in which w_t moves is a (positive) linear combination of the points $\{x_i : i \in \mathcal{I}_v\}$. Let's define $\rho = \min_i y_i x_i^\top v$; then, summarizing,

$$v = \sum_{i=1}^m \alpha_i y_i x_i \quad \text{with } \forall i, (\alpha_i \geq 0 \text{ and } y_i x_i^\top v = \rho) \text{ or } (\alpha_i = 0 \text{ and } y_i x_i^\top v > \rho). \quad (15.1)$$

In fact, ρ is a quantity known as the *geometric margin* of the linear separator v ; it is exactly the smallest distance from any of the x_i to the hyperplane $\{x : v^\top x = 0\}$, the decision boundary of the linear classifier with weights v .

15.2.1 Margin maximization

The equations (15.1) turn out to be equivalent to the **KKT conditions** of the problem of finding the *max-margin separator*, also known as a hard support vector machine (SVM). This problem is given by

$$\arg \max_{v: \|v\|=1} \min_{i \in [m]} y_i x_i \cdot v \quad \text{s.t. } \forall i \in [m], y_i x_i \cdot v > 0$$

Change so that $v = w/\|w\|$ for any w :

$$\begin{aligned} &= \arg \max_{w \in \mathbb{R}^d} \min_{i \in [m]} \frac{y_i x_i \cdot w}{\|w\|} \quad \text{s.t. } \forall i \in [m], y_i x_i \cdot w > 0 \\ &= \arg \max_{w \in \mathbb{R}^d} \frac{1}{\|w\|} \min_{i \in [m]} y_i x_i \cdot w \quad \text{s.t. } \forall i \in [m], y_i x_i \cdot w > 0 \end{aligned}$$

The objective is the same for any $w' = cw$ for $c > 0$, so we might as well limit ourselves to solutions where $\min_i y_i x_i \cdot w = 1$:

$$\begin{aligned} &\supseteq \arg \max_{w \in \mathbb{R}^d} \frac{1}{\|w\|} \quad \text{s.t. } \forall i \in [m], y_i x_i \cdot w \geq 1 \\ &= \arg \min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|^2 \quad \text{s.t. } \forall i \in [m], y_i x_i \cdot w \geq 1. \end{aligned} \quad (15.2)$$

Using the definition of the KKT conditions on this problem and rearranging a bit yields (15.1). But, here's a direct argument without appealing to the KKT conditions.

First, the solution to (15.2) is unique: the objective is strictly convex, and the constraints are affine and by assumption feasible.

These constraints will be "active" exactly for the indices \mathcal{I}_v ; other points will have larger values of $y_i x_i \cdot w$. But if w included some component w_\perp such that $x_i \cdot w_\perp = 0$ for all $i \in \mathcal{I}_v$, then this wouldn't affect the active constraints at all, and (similarly to the representer theorem and Lemma 15.2) would only make the objective $\|w\|^2$ bigger. So solutions to (15.2) must have $w = \sum_{i \in \mathcal{I}_v} \alpha_i y_i x_i$ for some coefficients α .

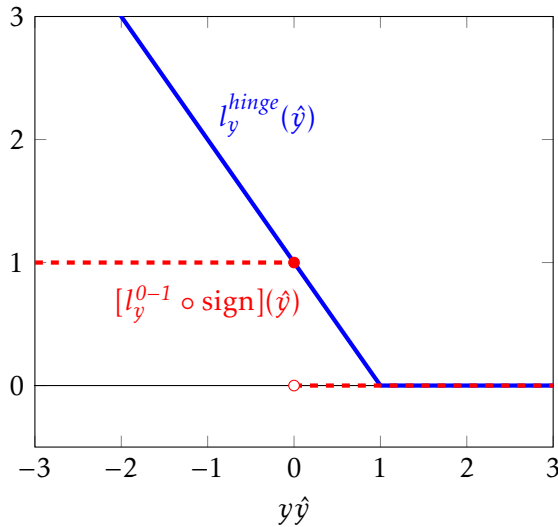
The solution also have that each $\alpha_i \geq 0$: suppose WLOG that $\alpha_1 < 0$; we'll see that setting $\alpha_1 = 0$ to zero would strictly improve the optimization. We have $w \cdot y_j x_j = \alpha_1 y_1 x_1 \cdot y_j x_j + \sum_{i>1} \alpha_i y_i x_i \cdot y_j x_j$, so the only way the margin can be improved by a negative α_1 for any point x_j is if $y_1 y_j x_1 \cdot x_j < 0$. But if the data is linearly separable, this is impossible: $\text{sign}(w \cdot x_i) = y_i$ for all i , but if $x_1 \cdot x_j < 0$ then there is no w such that $\text{sign}(x_1 \cdot w) = \text{sign}(x_j \cdot w) \neq 0$. Thus $\alpha_1 < 0$ does not help satisfy any of the constraints. It also does not help with the objective; $\|\alpha_1 y_1 x_1 - v\|^2 = \alpha_1^2 \|x_1\|^2 + \|v\|^2 - 2\alpha_1 y_1 x_1 \cdot v$, but we just established that $\text{sign}(x_1 \cdot v) = y_1$, so $\alpha_1 < 0$ only hurts the objective.

This establishes that the solution to (15.2) must satisfy (15.1). In the other direction: I don't know a concise formal proof without appealing to optimization theory we haven't covered, but geometrically, for suboptimal values of ρ there could be many solutions to (15.1). For the maximal value, however, for points in general position there will only be a single v satisfying these properties.

15.2.2 Hinge loss interpolation

The *hinge loss* is given by

$$l_y^{\text{hinge}}(\hat{y}) = \begin{cases} 1 - y\hat{y} & \text{if } y\hat{y} \leq 1 \\ 0 & \text{if } y\hat{y} \geq 1. \end{cases}$$



Notice that if $L_S^{\text{hinge}}(x \mapsto w \cdot x) = 0$, then for all $i \in [m]$, $y_i x_i \cdot w \geq 1$. Thus (15.2) is equivalent to

$$\arg \min_{w: L_S^{\text{hinge}}(x \mapsto w \cdot x) = 0} \|w\|,$$

the *minimum-norm hinge loss interpolator*. This is kind of a nice analogy to how gradient descent for least squares or similar losses (starting at $w_0 = 0$) finds the minimum-norm interpolator for that loss! But, interestingly, explicitly minimizing logistic loss (with gradient descent) implicitly minimizes hinge loss.

Transforming the hard constraint into a soft one gives us a soft support vector machine,

$$\arg \min_h L_S^{\text{hinge}}(h) + \lambda \|h\|^2.$$

15.2.3 Margin analysis

How can we think about the 0-1 generalization error of the max-margin predictor?

In dimension d , one option is to use that the VC dimension is either d or $d + 1$, depending on if we put an intercept in. But when d is high, e.g. $d > m$, this doesn't really tell us anything.

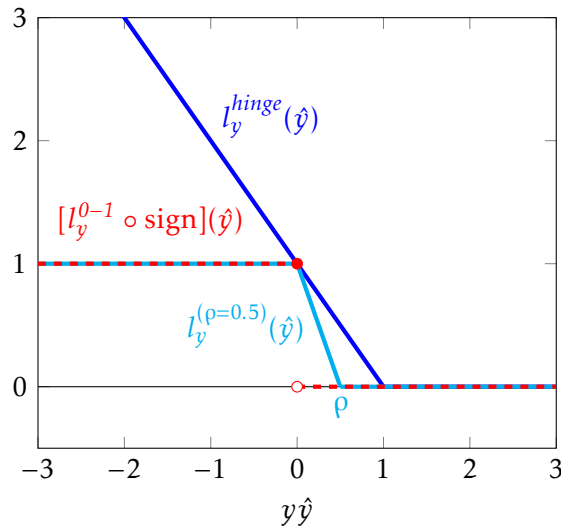
We're finding the minimum-norm interpolator, though, so maybe we can use a Rademacher bound that exploits that the norm isn't too big. So, let's think about $\mathcal{H}_B = \{h \in \mathcal{F} : \|h\| \leq B\}$ for some RKHS \mathcal{F} , potentially the linear kernel in dimension d but potentially not. We know that $\mathbb{E}_S \text{Rad}(\mathcal{H}_B|_{S_x}) \leq \frac{B}{\sqrt{m}} \sqrt{\mathbb{E} k(x, x)}$. To use this for a generalization bound on the 0-1 loss, though, we need to convert these soft predictions into hard ones with the sign function, so that the estimation error is bounded in terms of $\text{Rad}((\ell_{0-1} \circ \text{sign} \circ \mathcal{H}_B)|_S)$. But $\ell_{0-1} \circ \text{sign}$ isn't Lipschitz; it jumps suddenly from 0 to 1 as the sign of the predictor changes. So we can't use Talagrand's lemma to peel it off at all.

(When deriving VC dimension, we pretended the 0-1 loss was Lipschitz, but that only worked because we were working with a hypothesis class mapping to ± 1 . There's no similar trick we can play with continuous-output \mathcal{H} .)

We can work around this problem with *surrogate losses*. The hinge loss, above, is a good example: $\ell_{0-1}(h, z) \leq \ell_{\text{hinge}}(h, z)$ for any inputs, so necessarily $L_{\mathcal{D}}^{0-1}(h) \leq L_{\mathcal{D}}^{\text{hinge}}(h)$, and any generalization bound that applies to hinge loss also applies to 0-1 loss.

We can also use a tighter surrogate, though. One choice is *margin loss*:

$$l_y^\rho(\hat{y}) = \begin{cases} 1 & \text{if } y\hat{y} \leq 0 \\ 1 - \frac{1}{\rho}y\hat{y} & \text{if } 0 \leq y\hat{y} \leq \rho \\ 0 & \text{if } y\hat{y} \geq \rho \end{cases}$$



This is $1/\rho$ -Lipschitz, bounded in $[0, 1]$, and always an upper bound to the 0-1 loss. If $\min_i y_i h(x_i) \geq \rho$, then $L_S^\rho(h) = 0$. We get an immediate result:

$$L_{\mathcal{D}}^{0-1}(\text{sign} \circ h) \leq L_{\mathcal{D}}^\rho(h) \leq L_S^\rho(h) + \frac{2}{\rho} \mathbb{E}_S \text{Rad}(\mathcal{H}|_{S_x}) + \sqrt{\frac{1}{2m} \log \frac{1}{\delta}} \quad (15.3)$$

if $h \in \mathcal{H}$ and we picked ρ independently of S and h .

We can also do a nonuniform analysis to avoid committing in advance to a particular margin ρ , exactly like what we did for SRM:

PROPOSITION 15.3. *Let \mathcal{H} contain functions mapping to \mathbb{R} , and fix some $r > 0$. Then for any $\delta \in (0, 1)$, we have with probability at least $1 - \delta$ over the choice of $S \sim \mathcal{D}^m$ that it holds for all $h \in \mathcal{H}$ and $\rho \in (0, r]$ that*

$$L_{\mathcal{D}}^{0-1}(\text{sign} \circ h) \leq L_S^\rho(h) + \frac{4}{\rho} \mathbb{E}_{S' \sim \mathcal{D}^m} \text{Rad}(\mathcal{H}|_{S'}) + \sqrt{\frac{1}{m} \log \log_2 \frac{2r}{\rho}} + \sqrt{\frac{1}{2m} \log \frac{2}{\delta}}.$$

Proof. Let $\rho_k = r2^{-k}$ for all $k \geq 0$, and $\delta_k = \frac{6\delta}{\pi^2 k^2}$ for $k \geq 1$; note that $\sum_{k=1}^{\infty} \delta_k = \delta$. By (15.3), it holds with probability at least $1 - \delta_k$ for each ρ_k that

$$\forall h \in \mathcal{H}, \quad L_{\mathcal{D}}^{0-1}(\text{sign} \circ h) \leq L_S^{\rho_k}(\text{margin}(h)) + \frac{2}{\rho_k} \mathbb{E}_{S' \sim \mathcal{D}^m} \text{Rad}(\mathcal{H}|_{S'}) + \sqrt{\frac{1}{2m} \log \frac{1}{\delta_k}}.$$

For any $\rho \in (0, r]$, the smallest k such that $\rho_k \leq \rho$ is given by $k = \lceil \log_2 \frac{r}{\rho} \rceil$.

We have $\ell_{\rho'} \leq \ell_\rho$ for any $\rho' \leq \rho$, so $L_S^{\rho_k}(h) \leq L_S^\rho(h)$.

We also know that $\rho \leq \rho_{k-1} = 2\rho_k$, so $\frac{1}{\rho_k} \leq \frac{2}{\rho}$.

Finally, from $\log \frac{1}{\delta_k} = \log \frac{\pi^2}{6\delta} + 2 \log \log_2 \lceil \log_2 \frac{r}{\rho} \rceil$ we use that $\pi^2/6 < 2$ and $\lceil \log_2 a \rceil < \log_2(a) + 1 = \log_2(2a)$. \square

We do have to commit to some predefined upper bound on the margin r , but the resulting bound only depends on it through $\sqrt{\log \log_2 r}$, so we can pick something big.

15.3 OTHER MODELS/ALGORITHMS

Lyu and Li [LL20] and Ji and Telgarsky [JT20] study small-learning-rate gradient descent on L-homogeneous networks, those satisfying $h(x; \alpha w) = \alpha^L h(x; w)$ for $\alpha > 0$; this is true e.g. for (leaky)-ReLU networks. (We'll describe the [LL20] results.) Their analysis is in terms of the *normalized margin*

$$\bar{\gamma}(w) = \frac{\min_{i \in [m]} y_i h(x_i; w)}{\|w\|_2^L}.$$

This normalization is exactly the one that makes $\bar{\gamma}(\alpha w) = \bar{\gamma}(w)$. They show, using an approach like that of Section 15.2, that gradient flow or small-learning-rate gradient descent (under some additional regularity conditions) monotonically increase the log-sum-exp version of normalized margin, which means they approximately monotonically increase the normalized margin, which roughly means that it finds a local maximum (ish) of the normalized margin.

This is a kind of margin maximization, and Proposition 15.3 applies, but in general it's *not* margin maximization in an RKHS. Compare this to training a very wide network with square loss, in which case the implicit regularization prefers solutions with minimal NTK norm distance from the initialization. Knowing these results, you can ask questions like what this margin maximization actually does on particular models [e.g. Fre+23].

They talk about convergence to a “KKT point”; this is using the version of the KKT conditions where stationarity is defined by gradients, not subgradients, and hence isn't sufficient for optimality in nonconvex problems.

There's been a bunch of recent work trying to figure out the implicit regularization of Adam, rather than SGD, on homogeneous networks; some recent papers are [WMCL21; Wan+22; CKS23; XL24].

There's also a *ton* more work in this area; Vardi [Var22] gives a (now kind of outdated) survey.

Bibliography

- [ACGH19] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. “A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks”. *ICLR*. 2019. arXiv: [1810.02281](#).
- [AO10] Peter Auer and Ronald Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica* 61.1-2 (2010), pages 55–65.
- [Aro+19] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. “On Exact Computation with an Infinitely Wide Neural Net”. *NeurIPS*. 2019. arXiv: [1904.11955](#).
- [Aro+20] Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. “Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks”. *ICLR*. 2020. arXiv: [1910.01663](#).
- [Aro50] Nachman Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society* 68.3 (May 1950), pages 337–404.
- [Bach24] Francis Bach. *Learning Theory from First Principles*. Draft version. August 2024.
- [Bar93] Andrew R. Barron. Universal Approximation Bounds for Superpositions of a Sigmoidal Function. *IEEE Transactions on Information Theory* 39 (3 1993), pages 930–45.
- [BC12] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* 5.1 (2012), pages 1–122. arXiv: [1204.5721](#).
- [BLM13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.
- [BM02] Peter L. Bartlett and Shahar Mendelson. Rademacher and Gaussian Complexities: Risk Bounds and Structural Results. *Journal of Machine Learning Research* 3 (2002), pages 463–482.
- [Bub15] Sébastien Bubeck. *Convex Optimization: Algorithms and Complexity*. *Foundations and Trends in Machine Learning* 8.3-4 (2015). arXiv: [1405.4980](#).
- [BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
- [CKS23] Matias D. Cattaneo, Jason M. Klusowski, and Boris Shigida. *On the Implicit Bias of Adam*. 2023. arXiv: [2309.00079](#).
- [COB19] Lenaic Chizat, Edouard Oyallon, and Francis Bach. “On Lazy Training in Differentiable Programming”. *NeurIPS*. 2019. arXiv: [1812.07956](#).
- [Coh+21] Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. “Gradient Descent on Neural Networks Typically Occurs at the Edge of Stability”. *ICLR*. 2021. arXiv: [2103.00065](#).

- [CS02] Felipe Cucker and Steve Smale. [On the mathematical foundations of learning](#). *Bulletin of the American Mathematical Society* 39.1 (2002), pages 1–49.
- [DLS20] Tian Ding, Dawei Li, and Ruoyu Sun. [Sub-Optimal Local Minima Exist for Neural Networks with Almost All Non-Linear Activations](#). 2020. arXiv: 1911.01413.
- [DNL23] Alex Damian, Eshaan Nichani, and Jason D. Lee. [“Self-Stabilization: The Implicit Bias of Gradient Descent at the Edge of Stability”](#). *ICLR*. 2023. arXiv: 2209.15594.
- [Du+17] Simon S. Du, Chi Jin, Jason D. Lee, Michael I. Jordan, Barnabás Póczos, and Aarti Singh. [“Gradient Descent Can Take Exponential Time to Escape Saddle Points”](#). *NeurIPS*. 2017. arXiv: 1705.10412.
- [For+20] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. [“Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the Neural Tangent Kernel”](#). *NeurIPS*. 2020. arXiv: 2010.15110.
- [Fox23] Curtis Fox. [“A study of the edge of stability in deep learning”](#). MSc. Thesis. University of British Columbia, 2023.
- [Fre+23] Spencer Frei, Gal Vardi, Peter L. Bartlett, Nathan Srebro, and Wei Hu. [“Implicit Bias in Leaky ReLU Networks Trained on High-Dimensional Data”](#). *ICLR*. 2023. arXiv: 2210.07082.
- [GG23] Guillaume Garrigos and Robert M. Gower. [Handbook of Convergence Theorems for \(Stochastic\) Gradient Methods](#). 2023. arXiv: 2301.11235.
- [GKMR21] Surbhi Goel, Adam Klivans, Pasin Manurangsi, and Daniel Reichman. [“Tight Hardness Results for Training Depth-2 ReLU Networks”](#). *ITCS*. 2021. arXiv: 2011.13550.
- [GLSS18] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. [“Characterizing Implicit Bias in Terms of Optimization Geometry”](#). *ICML*. 2018. arXiv: 1802.08246.
- [HQC24] Marcus Hutter, David Quarel, and Elliot Catt. [An Introduction to Universal Artificial Intelligence](#). 2024.
- [HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halber White. [Multilayer Feedforward Networks are Universal Approximators](#). *Neural Networks* 2 (1989), pages 359–366.
- [Hut05] Marcus Hutter. [Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability](#). 2005.
- [Isr15] Robert Israel. [Can the ball \$B\(0, r_0\)\$ be covered with a finite number of balls of radius \$< r_0\$](#) . Mathematics Stack Exchange. April 1, 2015.
- [JGH18] Arthur Jacot, Franck Gabriel, and Clément Hongler. [“Neural Tangent Kernel: Convergence and Generalization in Neural Networks”](#). *NeurIPS*. 2018. arXiv: 1806.07572.
- [JNWB21] Sheng Jia, Ehsan Nezhadarya, Yuhuai Wu, and Jimmy Ba. [“Efficient Statistical Tests: A Neural Tangent Kernel Approach”](#). *ICML*. 2021.
- [JT19] Ziwei Ji and Matus Telgarsky. [“The implicit bias of gradient descent on nonseparable data”](#). *COLT*. 2019. arXiv: 1803.07300.
- [JT20] Ziwei Ji and Matus Telgarsky. [“Directional convergence and alignment in deep learning”](#). *NeurIPS*. 2020. arXiv: 2006.06657.
- [Kaw16] Kenji Kawaguchi. [“Deep Learning without Poor Local Minima”](#). *NeurIPS*. 2016. arXiv: 1605.07110.

- [KHSS18] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. *Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences*. 2018. arXiv: 1807.02582.
- [KL20] Patrick Kidger and Terry Lyons. “Universal Approximation with Deep Narrow Networks”. *COLT*. 2020. arXiv: 1905.08539.
- [KR23] Ahmed Khaled and Peter Richtárik. *Better Theory for SGD in the Nonconvex World*. *TMLR* (2023).
- [Lee+19] Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. “Wide neural networks of any depth evolve as linear models under gradient descent”. *NeurIPS*. 2019. arXiv: 1902.06720.
- [LL20] Kaifeng Lyu and Jian Li. “Gradient Descent Maximizes the Margin of Homogeneous Neural Networks”. *ICLR*. 2020. arXiv: 1906.05890.
- [LLPS93] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*. *Neural Networks* 6.6 (1993), pages 861–867.
- [LSJR16] Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. “Gradient Descent Only Converges to Minimizers”. *COLT*. 2016.
- [LV19] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. 4th edition. 2019.
- [LvB18] Thomas Laurent and James von Brecht. “Deep Linear Networks with Arbitrary Loss: All Local Minima Are Global”. *ICML*. 2018.
- [Ma22] Tengyu Ma. *Lecture Notes for Machine Learning Theory (CS229M/STATS214)*. June 2022.
- [MBS22] Mohamad Amin Mohamadi, Wonho Bae, and Danica J. Sutherland. “Making Look-Ahead Active Learning Strategies Feasible with Neural Tangent Kernels”. *NeurIPS*. 2022. arXiv: 2206.12569.
- [MBS23] Mohamad Amin Mohamadi, Wonho Bae, and Danica J. Sutherland. “A Fast, Well-Founded Approximation to the Empirical Neural Tangent Kernel”. *ICML*. 2023. arXiv: 2206.12543.
- [McD89] Colin McDiarmid. *On the method of bounded differences*. *Surveys in Combinatorics, 1989: Invited Papers at the Twelfth British Combinatorial Conference*. London Mathematical Society Lecture Note Series. Cambridge University Press, 1989, pages 148–188.
- [MKAS21] Eran Malach, Pritish Kamath, Emmanuel Abbe, and Nathan Srebro. “Quantifying the Benefit of Using Differentiable Learning over Tangent Kernels”. *ICML*. 2021. arXiv: 2103.01210.
- [MRT18] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. 2nd edition. MIT Press, 2018.
- [Nak21] Preetum Nakkiran. *Turing-Universal Learners with Optimal Scaling Laws*. 2021. arXiv: 2111.05321.
- [Nov+20] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. “Neural Tangents: Fast and Easy Infinite Neural Networks in Python”. *ICLR*. 2020. arXiv: 1912.02803.
- [NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.
- [Par94] Ian Parberry. *Circuit complexity and neural networks*. MIT Press, 1994.
- [Rag14] Maxim Raginsky. *Concentration inequalities*. September 2014.

- [RGS23] Yi Ren, Shangmin Guo, Wonho Bae, and Danica J. Sutherland. “How to prepare your task head for finetuning”. *ICLR*. 2023. arXiv: 2302.05779.
- [RGS22] Yi Ren, Shangmin Guo, and Danica J. Sutherland. “Better Supervisory Signals by Observing Learning Paths”. *ICLR*. 2022. arXiv: 2203.02485.
- [Rom21] Marc Romani. *A short proof of Hoeffding’s lemma*. May 1, 2021.
- [RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.
- [SC08] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, 2008.
- [SFL10] Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R. G. Lanckriet. “On the relation between universality, characteristic kernels and RKHS embedding of measures”. *AISTATS*. 2010. arXiv: 1003.0887.
- [Sou+18] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. *The Implicit Bias of Gradient Descent on Separable Data*. *JMLR* (2018). arXiv: 1710.10345.
- [SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
- [SSGF13] Dino Sejdinovic, Bharath K. Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. *Equivalence of distance-based and RKHS-based statistics in hypothesis testing*. *Annals of Statistics* 41.5 (October 2013), pages 2263–2291.
- [Tel21] Matus Telgarsky. *Deep learning theory lecture notes*. October 2021.
- [Val84] Leslie G. Valiant. *A Theory of the Learnable*. *Communications of the ACM* 27.11 (1984), pages 1134–1142.
- [Var22] Gal Vardi. *On the Implicit Bias in Deep-Learning Algorithms*. 2022. arXiv: 2208.12591.
- [VC71] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*. *Theory of Probability & Its Applications* 16.2 (1971), pages 264–280.
- [vdV98] Aad W. van der Vaart. *Asymptotic Statistics*. Cambridge University Press, 1998.
- [vRoo+24] Iris van Rooij, Olivia Guest, Federico Adolphi, Ronald de Haan, Antonina Kolokolova, and Patricia Rich. *Reclaiming AI as a Theoretical Tool for Cognitive Science*. *Computational Brain & Behavior* (2024).
- [Wai19] Martin Wainwright. *High-dimensional statistics: a non-asymptotic viewpoint*. Cambridge University Press, 2019.
- [Wan+22] Bohan Wang, Qi Meng, Huishuai Zhang, Ruoyu Sun, Wei Chen, Zhi-Ming Ma, and Tie-Yan Liu. “Does Momentum Change the Implicit Regularization on Separable Data?” *NeurIPS*. 2022. arXiv: 2110.03891 [cs.LG].
- [WHS22] Alexander Wei, Wei Hu, and Jacob Steinhardt. “More Than a Toy: Random Matrix Models Predict How Real-World Neural Representations Generalize”. *ICML*. 2022. arXiv: 2203.06176 [cs.LG].
- [WMCL21] Bohan Wang, Qi Meng, Wei Chen, and Tie-Yan Liu. “The Implicit Bias for Adaptive Optimization Algorithms on Homogeneous Neural Networks”. *ICML*. 2021. arXiv: 2012.06244.
- [Wol96] David H. Wolpert. *The Lack of A Priori Distinctions Between Learning Algorithms*. *Neural Computation* 8.7 (October 1996), pages 1341–1390.

-
- [XL24] Shuo Xie and Zhiyuan Li. “Implicit Bias of AdamW: ℓ_∞ Norm Constrained Optimization”. *ICML*. 2024. arXiv: 2404.04454.
- [Yan19] Greg Yang. “Tensor Programs I: Wide Feedforward or Recurrent Neural Networks of Any Architecture are Gaussian Processes”. *NeurIPS*. 2019. arXiv: 1910.12478.
- [YH22] Greg Yang and Edward J. Hu. “Feature Learning in Infinite-Width Neural Networks”. *ICML*. 2022. arXiv: 2011.14522.
- [YL21] Greg Yang and Etai Littwin. “Tensor Programs IIb: Architectural Universality of Neural Tangent Kernel Training Dynamics”. *ICML*. 2021. arXiv: 2105.03703.
- [YS19] Gilad Yehudai and Ohad Shamir. “On the Power and Limitations of Random Features for Understanding Neural Networks”. *NeurIPS*. 2019. arXiv: 1904.00687.
- [Zhang23] Tong Zhang. *Mathematical Analysis of Machine Learning Algorithms*. Pre-publication version. 2023.
- [Zho+22] Lijia Zhou, Frederic Koehler, Pragya Sur, Danica J. Sutherland, and Nathan Srebro. “A Non-Asymptotic Moreau Envelope Theory for High-Dimensional Generalized Linear Models”. *NeurIPS*. 2022. arXiv: 2210.12082.