CS229T/STAT231: Statistical Learning Theory (Winter 2016)

Percy Liang

Last updated Wed Apr 20 2016 01:36

These lecture notes will be updated periodically as the course goes on. The Appendix describes the basic notation, definitions, and theorems.

Contents

[begin lecture 1] (1)

1 Overview

1.1 What is this course about? (Lecture 1)

- Machine learning has become an indispensible part of many application areas, in both science (biology, neuroscience, psychology, astronomy, etc.) and engineering (natural language processing, computer vision, robotics, etc.). But machine learning is not a single approach; rather, it consists of a dazzling array of seemingly disparate frameworks and paradigms spanning classification, regression, clustering, matrix factorization, Bayesian networks, Markov random fields, etc. This course aims to uncover the common **statistical principles** underlying this diverse array of techniques.

- This class is about the theoretical analysis of learning algorithms. Many of the analysis techniques introduced in this class—which involve a beautiful blend of probability, linear algebra, and optimization—are worth studying in their own right and are useful outside machine learning. For example, we will provide generic tools to bound the supremum of stochastic processes. We will show how to optimize an arbitrary sequence of convex functions and do as well on average compared to an expert that sees all the functions in advance.

- Meanwhile, the practitioner of machine learning is hunkered down trying to get things to work. Suppose we want to build a classifier to predict the topic of a document (e.g., sports, politics, technology, etc.). We train a logistic regression with bag-of-words features and obtain 8% training error on 1000 training documents, test error is 13% on 1000 documents. There are many questions we could ask that could help us move forward.

  - How reliable are these numbers? If we reshuffled the data, would we get the same answer?
  - How much should we expect the test error to change if we double the number of examples?
  - What if we double the number of features? What if our features or parameters are sparse?
  - What if we double the regularization? Maybe use L_1 regularization?
  - Should we change the model and use an SVM with a polynomial kernel or a neural network?

  In this class, we develop tools to tackle some of these questions. Our goal isn't to give precise quantitative answers (just like analyses of algorithms doesn't tell you how exactly many hours a particular algorithm will run). Rather, the analyses will reveal the relevant quantities (e.g., dimension, regularization strength, number of training examples), and reveal how they influence the final test error.

- While a deeper theoretical understanding can offer a new perspective and can aid in troubleshooting existing algorithms, it can also suggest new algorithms which might have been non-obvious without the conceptual scaffolding that theory provides.
  - A famous example is boosting. The following question was posed by Kearns and Valiant in the late 1980s: is it possible to combine weak classifiers (that get 51% accuracy) into a strong classifier (that get 99% accuracy)? This theoretical challenge eventually led to the development of AdaBoost in the mid 1990s, a simple and practical algorithm with strong theoretical guarantees.
  - In a more recent example, Google's latest 22-layer convolutional neural network that won the 2014 ImageNet Visual Recognition Challenge was initially inspired by a theoretically-motivated algorithm for learning deep neural networks with sparsity structure.

  There is obviously a large gap between theory and practice; theory relies on assumptions can be simultaneously too strong (e.g., data are i.i.d.) and too weak (e.g., any distribution). The philosophy of this class is that the the purpose of theory here not to churn out formulas that you simply plug numbers into. Rather, theory should *change the way you think*.

- This class is structured into four sections: asymptotics, uniform convergence, kernel methods, and online learning. We will move from very strong assumptions (assuming the data are Gaussian, in asymptotics) to very weak assumptions (assuming the data can be generated by an adversary, in online learning). Kernel methods is a bit of an outlier in this regard; it is more about representational power rather than statistical learning.

1.2 Asymptotics (Lecture 1)

- The setting is thus: Given data drawn based on some unknown parameter vector θ^*, we compute a parameter estimate $\hat{\theta}$ from the data. How close is $\hat{\theta}$ to θ^*?
- For simple models such as Gaussian mean estimation and fixed design linear regression, we can compute $\hat{\theta} - \theta^*$ in closed form.
- For most models, e.g., logistic regression, we can't. But we can appeal to asymptotics, a common tool used in statistics (van der Vaart, 1998). The basic idea is to take Taylor expansions and show **asymptotic normality**: that is, the distribution of $\sqrt{n}(\hat{\theta} - \theta^*)$ tends to a Gaussian as the number of examples n grows. The appeal of asymptotics is that we can get this simple result even if $\hat{\theta}$ is very complicated!

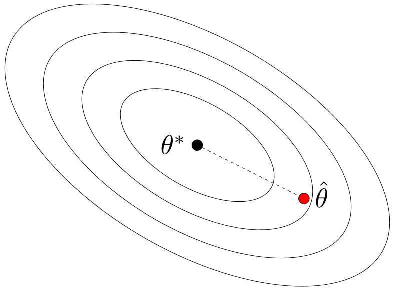

Figure 1: In asymptotic analysis, we study how a parameter estimate $\hat{\theta}$ behaves when it is close to the true parameters θ^*.

- Most of our analyses will be done with maximum likelihood estimators, which have nice statistical properties (they have the smallest possible asymptotic variance out of all estimators). But maximum likelihood is computationally intractable for most latent-variable models and requires non-convex optimization. These optimization problems are typically solved by EM, which is only guaranteed to converge to local optima. We will show how the **method of moments**, an old approach to parameter estimation dating back to Pearson (1894) can be brought to bear on this problem, yielding efficient algorithms that produce globally optimal solutions (Anandkumar et al., 2012b).

1.3 Uniform convergence (Lecture 1)

- Asymptotics provides a nice first cut analysis, and is adequate for many scenarios, but it has two drawbacks:
  - It requires a smoothness assumption, which means you can't analyze the hinge loss or L_1 regularization.
  - It doesn't tell you how large n has to be before the asymptotics kick in.
- Uniform convergence provides another perspective on the problem. To start, consider standard supervised learning: given a training set of input-output (x, y) pairs, the learning algorithm chooses a predictor $h : \mathcal{X} \to \mathcal{Y}$ from a hypothesis class $\mathcal{H}$ and we evaluate it based on unseen test data. Here's a simple question: how do the training error $\hat{L}(h)$ and test error $L(h)$ relate to each other?
- For a fixed $h \in \mathcal{H}$, the training error $\hat{L}(h)$ is an average of i.i.d. random variables (loss on each example), which converges to test error $L(h)$ at a rate governed by Hoeffding's inequality or the central limit theorem.
- But the point in learning is that we're supposed to choose a hypothesis based on the training data, not simply used a fixed h. Specifically, consider the **empirical risk**

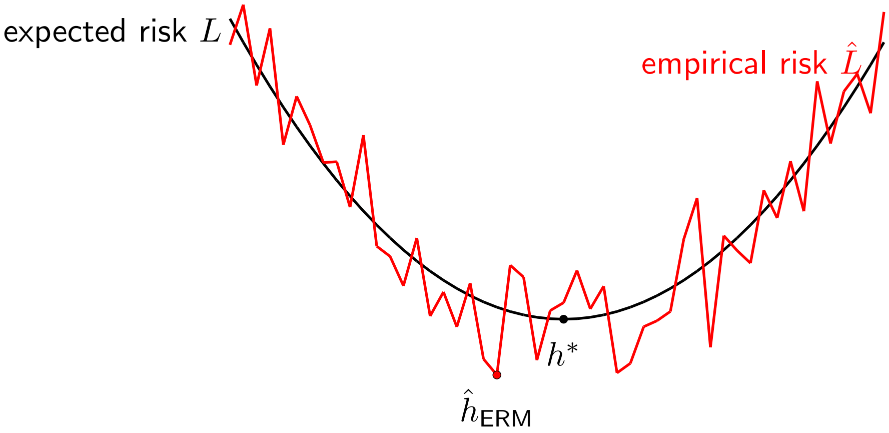

Figure 2: We want to minimize the expected risk L to get h^*, but we can only minimize the empirical risk $\hat{L}$ to get $\hat{h}$.

minimizer (ERM), which minimizes the training error:

$$\hat{h}_{\text{ERM}} \in \arg\min_{h \in \mathcal{H}} \hat{L}(h). \tag{1}$$

Now what can we say about the relationship between $\hat{L}(\hat{h}_{\text{ERM}})$ and $L(\hat{h}_{\text{ERM}})$? The key point is that $\hat{h}_{\text{ERM}}$ depends on $\hat{L}$, so $\hat{L}(\hat{h}_{\text{ERM}})$ is no longer an average of i.i.d. random variables; in fact, it's quite a narly beast. Intuitively, the training error should be smaller than the test error and hence less reliable, but how do we formalize this? Using **uniform convergence**, we will show that:

$$\underbrace{L(\hat{h}_{\text{ERM}})}_{\text{test error}} \le \underbrace{\hat{L}(\hat{h}_{\text{ERM}})}_{\text{training error}} + O_p\left(\sqrt{\frac{\text{Complexity}(\mathcal{H})}{n}}\right). \tag{2}$$

- The rate of convergence is governed by the complexity of $\mathcal{H}$. We will devote a good deal of time computing the complexity of various function classes $\mathcal{H}$. VC dimension, covering numbers, and Rademacher complexity are different ways of measuring how big a set of functions is. For example, we will see that if $\mathcal{H}$ contains d-dimensional linear classifiers that are s-sparse, then Complexity$(\mathcal{H}) = O(s \log d)$, which means we can tolerate exponentially more irrelevant features! All of these results are **distribution-free**, makes no assumptions on the underlying data-generating distribution!

- These generalization bounds are in some sense the heart of statistical learning theory. But along the way, we will develop generally useful **concentration inequalities** whose applicability extends beyond machine learning (e.g., showing convergence of eigenvalues in random matrix theory).

1.4 Kernel methods (Lecture 1)

- Let us take a detour away from analyzing the error of learning algorithms and thinking more about what models we should be learning. Real-world data is complex, so we need expressive models. Kernels provide a rigorous mathematical framework to build complex, non-linear models based on the machinery of linear models.

- For concreteness, suppose we're trying to solve a regression task: predicting $y \in \mathbb{R}$ from $x \in \mathcal{X}$.

- The usual way of approaching machine learning is to define functions via a linear combination of features: $f(x) = w \cdot \phi(x)$. Richer features yield richer functions:
  - $\phi(x) = [1, x]$ yields linear functions
  - $\phi(x) = [1, x, x^2]$ yields quadratic functions

- Kernels provide another way to define functions. We define a **positive semidefinite kernel** $k(x, x')$, which captures the "similarity" between x and x', and the define a function by comparing with a set of exemplars: $f(x) = \sum_{i=1}^n \alpha_i k(x_i, x)$.
  - Kernels allow us to construct complex non-linear functions (e.g., Gaussian kernel $k(x, x') = \exp(\frac{-\|x-x'\|^2}{2\sigma^2})$) that are **universal**, in that it can approximate *any* continuous function.
  - For strings, trees, graphs, etc., can define kernels that exploit dynamic programming for efficient computation.

- Finally, we operate on the functions themselves, which is at the end of the day all that matters. We will define a space of functions called an **reproducing kernel Hilbert space** (RKHS), which allows us to treat the functions like vectors and perform linear algebra.

- It turns out that all three perspectives are getting at the same thing:

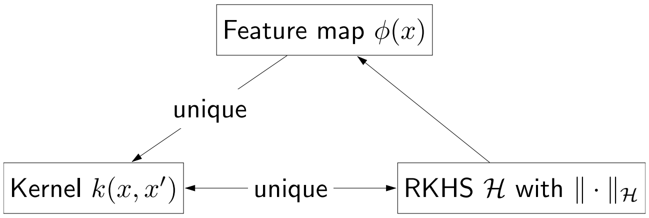

Figure 3: The three key mathematical concepts in kernel methods.

- There are in fact many feature maps that yield the same kernel (and thus RKHS), so one can choose the one based on computational convenience. Kernel methods generally require $O(n^2)$ time (n is the number of training points) to even compute the kernel matrix. Approximations based on sampling offer efficient solutions. For example, by generating lots of random features of the form $\cos(\omega \cdot x + b)$, we can approximate any shift-invariant kernel. Random features also provides some intuition into why neural networks work.

1.5 Online learning (Lecture 1)

- The world is a dynamic place, something that's not captured well by our earlier analyses based on asymptotics and uniform convergence. Online learning tries to do address this in two ways:
  - So far, in order to analyze the error of a learning algorithm, we had to assume that the training examples were i.i.d. However in practice, data points might be dependent, and worse, they might be generated by an adversary (think spam classification).
  - In addition, we have so far considered the batch learning setting, where we get a training set, learn a model, and then deploy that model. But in practice, data might be arriving in a stream, and we might want to interleave learning and prediction.
- The online learning setting is a game between a learner and nature:

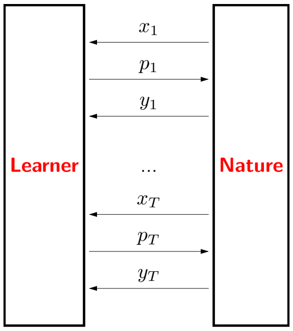

Figure 4: Online learning game.

  - Iterate $t = 1, \dots, T$:
    - ∗ Learner receives input $x_t \in \mathcal{X}$ (e.g., system gets email)
    - ∗ Learner outputs prediction $p_t \in \mathcal{Y}$ (e.g., system labels email)
    - ∗ Learner receives true label $y_t \in \mathcal{Y}$ (e.g., user relabels email if necessary)
    - ∗ (Learner updates model parameters)

- How do we evaluate?
  - Loss function: $\ell(y_t, p_t) \in \mathbb{R}$ (e.g., 1 if wrong, 0 if correct)
  - Let $\mathcal{H}$ be a set of *fixed* expert predictors (e.g., heuristic rules)
  - **Regret**: cumulative difference between learner's loss and best fixed expert's loss: in other words, how well did the learner do compared to if we had seen all the data and chose the best single expert in advance?
- Our primary aim is to upper bound the regret. For example, for finite $\mathcal{H}$, we will show:

$$\text{Regret} \le O(\sqrt{T \log |\mathcal{H}|}). \tag{3}$$

  Things to note:

  - The average regret $\frac{\text{Regret}}{T}$ goes to zero (in the long run, we're doing as well as the best fixed expert).
  - The bound only depends logarithmically on the number of experts, which means that the number of experts $|\mathcal{H}|$ can be exponentially larger than the number of data points T.
  - There is no probability here; the data is generated adversarially. What we rely on is the (possibly hidden) **convexity** in the problem.
- Online learning naturally leads to the **multi-armed bandit setting**, where the learner only receives partial feedback. Suppose you're trying to assign treatments p_t to patients x_t. In this case, you only observe the outcome associated with treatment p_t, not the other treatments you could have chosen (in the standard online learning setting, you receive the true label and hence feedback about every possible output). Online learning techniques can be adapted to work in the multi-armed bandit setting.
- Overall, online learning has many things going for it: it leads to many simple algorithms (online gradient descent) which are used heavily in practice. At the same time, the theory is remarkably clean and results are quite tight.

2 Asymptotics

2.1 Overview (Lecture 1)

- Here's the basic question: Suppose we have data points drawn from an unknown distribution with parameters θ^*:

$$x^{(1)}, \dots, x^{(n)} \sim p_{\theta^*}. \tag{4}$$

  Can we come up with an estimate $\hat{\theta}$ (some function of the data) that gets close to θ^*?

- We begin with perhaps the simplest statistical problem: estimating the mean of a Gaussian distribution (Section 2.2). This allows us to introduce the concept of parameter error, and provides intuition for how this error scales with dimensionality and number of examples. We then consider estimating multinomial distributions (Section 2.3), where we get our first taste of asymptotics in order to cope with the non-Gaussianity, appealing to the central limit theorem and the continuous mapping theorem. Finally, we generalize to exponential families (Section 2.4), a hugely important class of models. Here, our parameter estimate is no longer an empirical average, so we lean on the delta method.

- We then introduce the method of moments, and show an equivalence with exponential families via the maximum entropy principle (Section 2.5). But the real benefit of method of moments is in latent-variable models (Section 2.6). Here, we show that we can obtain an efficient estimator for three-view mixture models when maximum likelihood would otherwise require non-convex optimization.

- Next, we turn from parameter estimation to prediction, starting with fixed design linear regression (Section 2.7). The analysis is largely similar to mean estimation, and we get exact closed form expressions. For the random design setting and general loss functions (to handle logistic regression), we need to again appeal to asymptotics (Section 2.8).

- So far, we've only considered unregularized estimators, which make sense when the number of data points is large relative to the model complexity. Otherwise, regularization is necessary to prevent overfitting. As a first attempt to study regularization, we analyze the regularized least squares estimator (ridge regression) for fixed design linear regression, and show how to tune the regularization strength to balance the bias-variance tradeoff (Section 2.9).

2.2 Gaussian mean estimation (Lecture 1)

- We warmup with a simple classic example from statistics. The goal is to estimate the mean of a Gaussian distribution. Suppose we have n i.i.d. samples from a Gaussian distribution with unknown mean $\theta^* \in \mathbb{R}^d$ and known variance $\sigma^2 I$ (so that each of the d components are independent):

$$x^{(1)}, \dots, x^{(n)} \sim \mathcal{N}(\theta^*, \sigma^2 I) \tag{5}$$

- Define $\hat{\theta}$ to be parameter estimate equal to the sample mean:

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}. \tag{6}$$

- Let us study how this estimate $\hat{\theta}$ deviates from the true parameter θ:

$$\hat{\theta} - \theta^* \in \mathbb{R}^d. \tag{7}$$

This deviation is a random vector which should depend on the number of points n and the variance σ^2. What do you think the dependence will be?

- **Lemma 1 (parameter deviation for Gaussian mean)**
  - The sample mean estimate $\hat{\theta}$ deviates from the true parameter θ^* according to a Gaussian:

$$\hat{\theta} - \theta^* \sim \mathcal{N}\left(0, \frac{\sigma^2 I}{n}\right). \tag{8}$$

  - FIGURE: [$\hat{\theta}$ in a ball around θ^* for $d = 2$]

- Proof of Lemma 1
  - We need the following basic properties of Gaussians:
    - ∗ Fact 1 (sums of independent Gaussians): if $v_1 \sim \mathcal{N}(0, \Sigma_1)$ and $v_2 \sim \mathcal{N}(0, \Sigma_2)$ are independent, then $v_1 + v_2 \sim \mathcal{N}(0, \Sigma_1 + \Sigma_2)$.
    - ∗ Fact 2 (constant times Gaussian): if $v \sim \mathcal{N}(0, \Sigma)$, then $Av \sim \mathcal{N}(0, A\Sigma A^\top)$ for any matrix A.
  - The rest is just algebra:
    - ∗ First, note that $x^{(i)} - \theta^* \sim \mathcal{N}(0, \sigma^2 I)$.
    - ∗ Next, we have that $S_n \stackrel{\text{def}}{=} \sum_{i=1}^n x^{(i)} \sim \mathcal{N}(0, n\sigma^2 I)$ by Fact 1.
    - ∗ Finally, we have $\hat{\theta} - \theta^* = \frac{S_n}{n} \sim \mathcal{N}\left(0, \frac{\sigma^2 I}{n}\right)$ by Fact 2.

- Lemma 1 tells us how the entire estimated vector behaves, but sometimes it's convenient to just get one number that tells us how we're doing. Let's define the **mean-squared parameter error**, which is the squared distance between the estimate $\hat{\theta}$ and the true parameter θ^*:

$$\|\hat{\theta} - \theta^*\|_2^2. \tag{9}$$

How do you think the parameter error behaves?

- **Lemma 2 (parameter error for Gaussian mean)**
  - The mean-squared parameter error is:

$$\|\hat{\theta} - \theta^*\|_2^2 \sim \frac{\sigma^2}{n}\chi_d^2, \tag{10}$$

  and has expected value:

$$\boxed{\mathbb{E}\left[\|\hat{\theta} - \theta\|_2^2\right] = \frac{d\sigma^2}{n}.} \tag{11}$$

- Proof of Lemma 2
  - Standard properties of chi-squared distributions:
    - * If $v_1, \dots, v_d \sim \mathcal{N}(0,1)$, then $\sum_{j=1}^d v_j^2 \sim \chi_d^2$.
    - * If $z \sim \chi_d^2$, then $\mathbb{E}[z] = d$ and $\text{Var}[z] = 2d$.
  - By Lemma 1, we know that $v = \sqrt{\frac{n}{\sigma^2}}(\hat{\theta} - \theta^*) \sim \mathcal{N}(0, I)$.
  - The squared 2-norm of this vector ($\|v\|_2^2 = \sum_{j=1}^d v_j^2$) is therefore distributed as χ_d^2.
  - Multiplying both sides by $\frac{\sigma^2}{n}$ yields (10).
  - Taking expectations on both sides yields (11).

- A Gaussian random vector has fluctuations on the order of its standard deviation, we can think of $\hat{\theta}$ having typical deviations on the order of $\sqrt{\frac{d\sigma^2}{n}}$.

- FIGURE: [relate to previous figure, density of a chi-squared for $\|\hat{\theta} - \theta^*\|_2^2$]

- Takeaways
  - Life is easy when everything's Gaussian, since we can compute things in closed form.
  - Think geometrically about shrinking balls around θ^*.

2.3 Multinomial estimation (Lecture 1)

- In the above, things worked out nicely because everything was Gaussian. What if are data are not Gaussian?

- Suppose we have an unknown multinomial distribution over d choices: $\theta^* \in \Delta_d$ (that is, $\theta = [\theta_1, \dots, \theta_d]$, $\theta_j \geq 0$ for each j and $\sum_{j=1}^d \theta_j = 1$). Suppose we have n i.i.d. samples from this unknown multinomial distribution

$$x^{(1)}, \dots, x^{(n)} \sim \text{Multinomial}(\theta^*), \tag{12}$$

  where each $x^{(i)} \in \{e_1, \dots, e_d\}$ and $e_j \in \{0,1\}^d$ is an indicator vector that is 1 at position j and 0 elsewhere.

- Consider the empirical distribution as the estimator (same form as the sample mean):

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^n x^{(i)}. \tag{13}$$

- Example:

  - $\theta^* = [0.2, 0.5, 0.3]$
  - $x^{(1)} = [0, 1, 0], x^{(2)} = [1, 0, 0], x^{(3)} = [0, 1, 0]$
  - $\hat{\theta} = [\frac{1}{3}, \frac{2}{3}, 0]$

- We can now ask the same question as we did for Gaussian estimation: what is the parameter error $\|\hat{\theta} - \theta^*\|_2^2$? Before we had gotten a chi-squared distribution because we had Gaussians and could work out everything in closed form. But now, we have multinomial distributions, which makes things more difficult to analyze. How do you think the parameter error will behave?

- Fortunately, not all hope is lost. Our strategy will be to study the **asymptotic behavior** of the estimators. By the **central limit theorem**, we have

$$\sqrt{n}(\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}(0, V), \tag{14}$$

  where $V \stackrel{\text{def}}{=} \operatorname{diag}(\theta^*) - \theta^*(\theta^*)^\top$ is the the asymptotic variance of a single multinomial draw. Written out, the elements of V are:

$$V_{jk} = \begin{cases} \theta_j^*(1 - \theta_j^*) & \text{if } j = k \\ -\theta_j^*\theta_k^* & \text{if } j \neq k. \end{cases} \tag{15}$$

  It is easy to check this by noticing that $\mathbb{E}[x_j] = \mathbb{E}[x_j^2] = \theta_j^*$, $\mathbb{E}[x_j x_k] = 0$ for $j \neq k$. Further sanity check:

  - For a single component j, when θ_j^* is close to 0 or 1, the variance is small.
  - The covariance between components j and k is negative since the probabilities sum to 1 (there is competition among components).

- Next, we can apply the **continuous mapping theorem** on the function $\|\cdot\|_2^2$ (see Appendix), which lets us transfer convergence results through continuous functions:

$$n\|\hat{\theta} - \theta^*\|_2^2 \xrightarrow{d} \operatorname{tr}(\mathcal{W}(V, 1)), \tag{16}$$

  where $\mathcal{W}(V, k)$ is the Wishart distribution (multivariate generalization of the chi-squared) with mean matrix V and k degrees of freedom. This follows because $z \sim \mathcal{N}(0, V)$, then $zz^\top \sim \mathcal{W}(V, 1)$, and $\|z\|_2^2 = \operatorname{tr}(zz^\top)$.

- Taking expectations[1] and dividing by n on both sides yields:

$$\mathbb{E}\left[\|\hat{\theta} - \theta^*\|_2^2\right] = \left(\sum_{j=1}^{d} \theta_j^*(1 - \theta_j^*)\right) \frac{1}{n} + o\left(\frac{1}{n}\right) \tag{17}$$

$$\leq \boxed{\frac{1}{n} + o\left(\frac{1}{n}\right).} \tag{18}$$

- Note that the $\frac{1}{n}$ term is independent of d, unlike in the Gaussian case, where the result was $\frac{d\sigma^2}{n}$. The difference is that in the Gaussian case, each dimension had standard deviation σ, whereas in the multinomial case here, it is about $\sqrt{\frac{1}{d}}$ for the uniform distribution.

[1] This step actually requires additional justification. In particular, if we have $Y_n \xrightarrow{d} Y$, in order to conclude $\mathbb{E}[Y_n] \to \mathbb{E}[Y]$, we need (Y_n) to be uniformly integrable. A sufficient condition is that Y_n has finite second-order moments, which means that a data point needs to have finite fourth-order moments. This is clearly the case for the multinomial distribution (which is bounded), and will be true for exponential families (for which all moments exist), discussed in the next section. Showing this result formally is non-trivial and is out of the scope of this class.

$$\textbf{[begin lecture 2]} \tag{2}$$

2.4 Exponential families (Lecture 2)

- So far, we've analyzed the Gaussian and the multinomial. Now let us delve into exponential families, a much more general family of distributions which include the Gaussian, multinomial, gamma, beta, etc., but not things like the uniform distribution or mixture models.

- Exponential families are also the basis of undirected graphical models. You probably first saw exponential families in a univariate continuous setting, but we will introduce them here in the multivariable discrete setting in order to build some different intuitions (and to simply the math).

- **Definition 1 (exponential family)**

  - Let $\mathcal{X}$ be a discrete set.
    - Example: $\mathcal{X} = \{-1, +1\}^3$
  - Let $\phi : \mathcal{X} \to \mathbb{R}^d$ be a feature function.
    - Example (unary and binary factors):

$$\phi(x) = [x_1, x_2, x_3, x_1 x_2, x_2 x_3] \in \mathbb{R}^5. \tag{19}$$

  - Define a family of distributions $\mathcal{P}$, where each $p_\theta \in \mathcal{P}$ assigns each $x \in \mathcal{X}$ a probability that depends on x only through $\phi(x)$:

$$\boxed{\mathcal{P} \stackrel{\text{def}}{=} \{p_\theta : \theta \in \mathbb{R}^d\}, \quad p_\theta(x) \stackrel{\text{def}}{=} \exp\{\theta \cdot \phi(x) - A(\theta)\},} \tag{20}$$

  where the **log-partition function**

$$A(\theta) \stackrel{\text{def}}{=} \log \sum_{x \in \mathcal{X}} \exp\{\theta \cdot \phi(x)\} \tag{21}$$

  ensures the distribution is normalized.

  - Example: if $\theta = [0, 0, 0, 0, 0]$, then p_θ is the uniform distribution.
  - Example: if $\theta = [0, 0, 0, \log 2, 0]$ (which favors x_1 and x_2 having the same sign), then p_θ assigns probability:

x_1	x_2	x_3	$\exp\{\theta \cdot \phi(x)\}$	$p_\theta(x)$
-1	-1	-1	2	0.2
-1	-1	$+1$	2	0.2
-1	$+1$	-1	0.5	0.05
-1	$+1$	$+1$	0.5	0.05
$+1$	-1	-1	0.5	0.05
$+1$	-1	$+1$	0.5	0.05
$+1$	$+1$	-1	2	0.2
$+1$	$+1$	$+1$	2	0.2

- Note that unlike multinomial distributions, $\mathcal{P}$ does not contain all possible distributions. Varying one parameter θ_j has the effect of moving all points x with the same feature $\phi(x)_j$ the same way.
- FIGURE: [Venn diagram over outcome x, indicator features correspond to regions]

- A very useful property of exponential families is that derivatives of the log-partition function correspond to moments of the distribution (a short algebra calculation that you should do once, but we're going to skip):
  - The mean is the gradient of the log-partition function:

$$\nabla A(\theta) = \mathbb{E}_\theta[\phi(x)] \overset{\text{def}}{=} \sum_{x \in \mathcal{X}} p_\theta(x)\phi(x). \tag{22}$$

  - The covariance is the Hessian of the log-partition function:

$$\nabla^2 A(\theta) = \text{Cov}_\theta[\phi(x)] \overset{\text{def}}{=} \mathbb{E}_\theta[(\phi(x) - \mathbb{E}_\theta[\phi(x)])(\phi(x) - \mathbb{E}_\theta[\phi(x)])^\top]. \tag{23}$$

  - Note that since $\nabla^2 A(\theta)$ is a covariance matrix, it is necessarily positive semidefinite, which means that A is convex.
  - If $\nabla^2 A(\theta) \succ 0$, then A is strongly convex, and ∇A is invertible (for intuition, consider the 1D case where ∇A would be an increasing function). In this case, the exponential family is said to be **minimal**. For simplicity, we will assume we are working with minimal exponential families henceforth.
  - One important consequence of minimality is that there is a one-to-one mapping between the **canonical parameters** θ and the **mean parameters** μ:

$$\theta = (\nabla A)^{-1}(\mu) \qquad \mu = \nabla A(\theta). \tag{24}$$

  - FIGURE: [moment mapping diagram from θ to μ]
  - An example of a non-minimal exponential family is the standard representation of a multinomial distribution: $\mathcal{X} = \{\text{A}, \text{B}, \text{C}\}$ and $\phi(x) = [\mathbb{I}[x = \text{A}], \mathbb{I}[x = \text{B}], \mathbb{I}[x = \text{C}]]$. There are three parameters but only two actual degrees of freedom in the model family $\mathcal{P}$. To make this exponential family minimal, just remove one of the features.

- Now let us turn to parameter estimation. Assume as usual that we get n i.i.d. points drawn from some member of the exponential family with parameters θ^*:

$$x^{(1)}, \dots, x^{(n)} \sim p_{\theta^*}. \tag{25}$$

The classic way to estimate this distribution is **maximum likelihood**:

$$\hat{p} = p_{\hat{\theta}}, \quad \hat{\theta} = \arg\max_{\theta \in \mathbb{R}^d} \sum_{i=1}^n \log p_\theta(x^{(i)}). \tag{26}$$

For exponential families, we can write this as succinctly as:

$$\hat{\theta} = \arg\max_{\theta \in \mathbb{R}^d} \{\hat{\mu} \cdot \theta - A(\theta)\}, \tag{27}$$

where

$$\hat{\mu} \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \phi(x^{(i)}) \tag{28}$$

are the sufficient statistics of the data. At the end of the day, estimation in exponential families is still about forming empirical averages. Let's now try to get a closed form expression for $\hat{\theta}$ as a function of $\hat{\mu}$. Since A is convex and differentiable, we can differentiate the expression in (27) to get the necessary and sufficient conditions for optimality:

$$\hat{\mu} - \nabla A(\hat{\theta}) = 0. \tag{29}$$

Inverting, we get that $\hat{\theta}$ and $\hat{\mu}$ are canonical and mean parameters:

$$\boxed{\hat{\theta} = (\nabla A)^{-1}(\hat{\mu}).} \tag{30}$$

In other words, the maximum likelihood estimator $\hat{\theta}$ for exponential families is just a non-linear transformation of the sufficient statistics $\hat{\mu}$.

- Asymptotic analysis of maximum likelihood in exponential families
  - By the definition of $\hat{\mu}$ (28) and the central limit theorem, we have:

  $$\sqrt{n}(\hat{\mu} - \mu^*) \xrightarrow{d} \mathcal{N}(0, \mathrm{Cov}_\theta(\phi(x))), \tag{31}$$

  where $\mu^* = \mathbb{E}_{\theta^*}[\phi(x)]$.
  - By the connection between $\hat{\theta}$ and $\hat{\mu}$ (30) and redefining $f = (\nabla A)^{-1}$, we can rewrite the quantity of interest as:

  $$\sqrt{n}(\hat{\theta} - \theta^*) = \sqrt{n}(f(\hat{\mu}) - f(\mu^*)). \tag{32}$$

What do we do with the RHS? The **delta method** allows us to transfer asymptotic normality results on one quantity ($\hat{\mu}$ in this case) to a non-linear transformation of another quantity ($f(\hat{\mu})$ in this case). The asymptotic variance is just multiplied by a linearization of the non-linear transformation. Specifically:

$$\sqrt{n}(f(\hat{\mu}) - f(\mu^*)) \xrightarrow{d} \mathcal{N}(0, \nabla f(\mu^*)^\top \mathrm{Cov}_\theta(\phi(x)) \nabla f(\mu^*)). \tag{33}$$

Conveniently, $\nabla f(\mu) = \nabla^2 A(\mu)^{-1} = \mathrm{Cov}_\theta(\phi(x))^{-1}$ for $\theta = f(\mu)$, so we get that:

$$\boxed{\sqrt{n}(\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \mathrm{Cov}_\theta(\phi(x))^{-1}).} \tag{34}$$

  - Notice that the asymptotic variance is the inverse of the covariance of the features $\phi(x)$. Intuitively, if the features vary more, we can estimate the parameters better (smaller asymptotic variance).
  - The parameter error $\|\hat{\theta} - \theta\|_2^2$ can be computed exactly the same way as for multinomial mean estimation, so we will skip this.

- Method of moments
  - We have introduced $\hat{\theta}$ as a maximum likelihood estimator. But we can also interpret it as a **method of moments** estimator. If we rewrite the optimality conditions for $\hat{\theta}$ using the fact that $\nabla A(\theta) = \mathbb{E}_\theta[\phi(x)]$, we see that

$$\mathbb{E}_{\hat{\theta}}[\phi(x)] = \hat{\mu}. \tag{35}$$

    In other words, the empirical moments $\hat{\mu}$ are identical to those defined by the model $p_{\hat{\theta}}$.
  - The method of moments principle is: Choose model parameters to make the moments under the model match moments of the data.
  - The maximum likelihood principle is: Choose model parameters to maximize the likelihood of the entire data.
  - The method of moments dates back to Karl Pearson from 1894, which predates the full development of maximum likelihood by Fisher in the 1920s. Since then, maximum likelihood has been the dominant paradigm for parameter estimation, mainly due to its statistical efficiency and naturalness. There is more ad-hocness in the method of moments, but it is exactly this ad-hocness that allows us to get computationally efficient algorithms at the price of reduced statistical efficiency, as we'll see later for latent-variable models.

2.5 Maximum entropy principle (Lecture 2)

- Let us scrap exponential families for now and embrace the method of moments perspective for a moment. Can we develop an estimator from first principles?

- Setup
  - We are given n data points $x^{(1)}, \dots, x^{(n)}$.
  - We have a feature function $\phi : \mathcal{X} \to \mathbb{R}^d$.
  - Define the **empirical moments** to be a vector aggregating the feature function over all the n examples:

$$\hat{\mu} \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \phi(x^{(i)}). \tag{36}$$

- Define $\mathcal{Q}$ to be the set of distributions with these empirical moments:

$$\boxed{\mathcal{Q} \overset{\text{def}}{=} \{q \in \Delta_{|\mathcal{X}|} : \mathbb{E}_q[\phi(x)] = \hat{\mu}\},} \tag{37}$$

  where $\mathbb{E}_q[\phi(x)] \overset{\text{def}}{=} \sum_{x \in \mathcal{X}} q(x)\phi(x)$.

- Since $|\mathcal{X}|$ can be extremely large (2^m if $\mathcal{X} = \{-1, +1\}^m$), there are many possible distributions q but only d constraints, so $\mathcal{Q}$ contains tons of candidates, and the problem is clearly underconstrained. How do we break ties? Edwin Jaynes introduced the principle of maximum entropy which provides one answer:

- **Definition 2 (maximum entropy principle (Jaynes, 1957))**
  - Choose the distribution with the empirical moments that has the highest entropy:

$$\boxed{\hat{q} \overset{\text{def}}{=} \arg\max_{q \in \mathcal{Q}} H(q),} \tag{38}$$

    where $H(q) \overset{\text{def}}{=} \mathbb{E}_q[-\log q(x)]$ is the entropy of the distribution q. In other words: respect the data (by choosing $q \in \mathcal{Q}$) but don't commit to more than that (by maximizing the entropy).

- We've now seen two approaches to estimate a distribution $\hat{p}$ given n data points:
  - Maximum likelihood in exponential families
  - Maximum entropy subject to moment constraints

  Which one should we choose? It turns out that we don't need to because they are equivalent, as the following theorem shows:

- **Theorem 1 (maximum entropy duality)**
  - Assume $\mathcal{Q}$ is non-empty.

  - Then maximum entropy subject to moment constraints is equivalent to maximum likelihood in exponential families:

$$\boxed{\arg\max_{q\in\mathcal{Q}} H(q) = \arg\max_{p\in\mathcal{P}} \sum_{i=1}^{n} \log p(x^{(i)}).} \tag{39}$$

- Proof of Theorem 1:

  - This result is a straightforward application of Langrangian duality.

$$\max_{q\in\mathcal{Q}} H(q) = \max_{q\in\Delta_{|\mathcal{X}|}} \min_{\theta\in\mathbb{R}^d} H(q) - \theta\cdot(\hat{\mu} - \mathbb{E}_q[\phi(x)]). \tag{40}$$

  Since $\mathcal{Q}$ is non-empty (Slater's condition), we can switch the min and max. Expanding:

$$\min_{\theta\in\mathbb{R}^d} \max_{q\in\Delta_{|\mathcal{X}|}} -\sum_{x\in\mathcal{X}} q(x)\log q(x) - \theta\cdot\left(\hat{\mu} - \sum_{x\in\mathcal{X}} q(x)\phi(x)\right). \tag{41}$$

  Next, differentiate with respect to q and set it to some constant c (because of the sum-to-one constraint): for each $x \in \mathcal{X}$,

$$c = -(1 + \log q(x)) + \theta\cdot\phi(x). \tag{42}$$

  Solving for q (which depends on θ, so we write q_θ), we see that $q_\theta \in \mathcal{P}$ is a member of the exponential family:

$$q_\theta(x) \propto \exp(\theta\cdot\phi(x)). \tag{43}$$

  Plugging this choice of q_θ into the original problem, we get:

$$\min_{\theta\in\mathbb{R}^d} -(\theta\cdot\mathbb{E}_\theta[\phi(x)] - A(\theta)) - \theta\cdot(\hat{\mu} - \mathbb{E}_\theta[\phi(x)]), \tag{44}$$

  which is equivalent to the maximum likelihood objective:

$$\max_{\theta\in\mathbb{R}^d} \{\theta\cdot\hat{\mu} - A(\theta)\}. \tag{45}$$

  This completes the proof. One sanity check we can optionally perform is to differentiate the maximum likelihood objective. Using the fact that $\nabla A(\theta) = \mathbb{E}_\theta[\phi(x)]$, we see that the moment constraints indeed hold at the optimal θ:

$$\hat{\mu} - \mathbb{E}_\theta[\phi(x)] = 0, \tag{46}$$

  so that the solution is also in $\mathcal{Q}$.

- Information geometry (digression)

- We can understand understand maximum entropy duality more fully by looking at it from an information geometry point of view, which studies the geometry of probability distributions.
- Recall the definition of the KL divergence between two distributions q and p:

$$\mathrm{KL}\left(q \| p\right) \overset{\text{def}}{=} \mathbb{E}_q[\log q(x) - \log p(x)]. \tag{47}$$

  Although KL is not a distance metric (it's not even symmetric), we can still talk about the notion of projections with respect to KL.
- Remark: if $\mathrm{KL}\left(q \| p\right)$ is finite, then $p(x) = 0$ implies $q(x) = 0$ (p must place positive probability mass in at least as many places as q does).
- Just for intuition: First, observe that $\mathcal{Q}$, the set of distributions consistent with a set of moments is closed under convex combinations of the distributions:

$$q_1, q_2 \in \mathcal{Q} \quad \Rightarrow \quad \alpha q_1 + (1-\alpha) q_2 \in \mathcal{Q} \quad \text{for all } \alpha \in [0,1] \tag{48}$$

  Second, observe that $\mathcal{P}$, the set of distributions in the exponential family is closed under convex combinations of the parameters:

$$p_{\theta_1}, p_{\theta_2} \in \mathcal{P} \quad \Rightarrow \quad p_{\alpha\theta_1 + (1-\alpha)\theta_2} \in \mathcal{P} \quad \text{for all } \alpha \in [0,1] \tag{49}$$

  So both $\mathcal{P}$ and $\mathcal{Q}$ are in some sense convex when viewed appropriately.
- We can think of a "projection" of an arbitrary distribution onto either $\mathcal{P}$ or $\mathcal{Q}$ based on the KL divergence:
  - **M-projection** (moment projection) of some q onto $\mathcal{P}$: $\arg\min_{p \in \mathcal{P}} \mathrm{KL}\left(q \| p\right)$
    - Example: maximum likelihood in an exponential family $\mathcal{P}$ is an M-projection of the empirical distribution $\frac{1}{n}\sum_{i=1}^n \delta_{x^{(i)}}$ onto $\mathcal{P}$.
  - **I-projection** (information projection) of some p onto $\mathcal{Q}$: $\arg\min_{q \in \mathcal{Q}} \mathrm{KL}\left(q \| p\right)$
    - Example: maximum entropy with respect to $\mathcal{Q}$ satisfying empirical moments is an I-projection of the uniform distribution onto $\mathcal{Q}$.
- **Theorem 2 (Pythagorean equality for exponential families)**
  - Let $\hat{p}$ be the solution to (39) (maximum entropy / maximum likelihood solution).
  - Then a Pythagorean identity holds:

$$\boxed{\mathrm{KL}\left(q \| p\right) = \mathrm{KL}\left(q \| \hat{p}\right) + \mathrm{KL}\left(\hat{p} \| p\right) \quad \text{for all } q \in \mathcal{Q}, p \in \mathcal{P}.} \tag{50}$$

This theorem says that some of the intuitions of Euclidean geometry carry over to information geometry when restricted to certain nice families of distributions. See Figure 5 for a visualization.

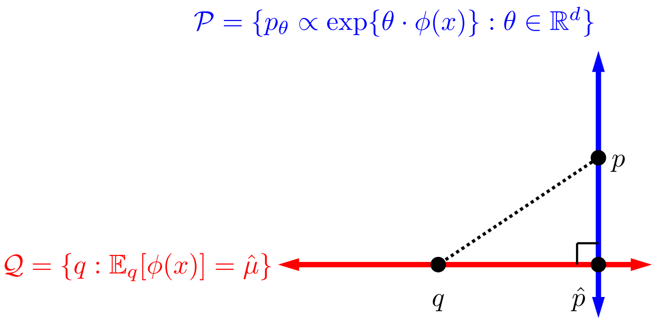
$\mathcal{P} = \{p_\theta \propto \exp\{\theta \cdot \phi(x)\} : \theta \in \mathbb{R}^d\}$
p
$\mathcal{Q} = \{q : \mathbb{E}_q[\phi(x)] = \hat\mu\}$
q
$\hat p$

Figure 5: The Pythagorean identity holds between $\mathcal{Q}$ (distributions satisfying moment constraints) and $\mathcal{P}$ (distributions in the corresponding exponential family).

[begin lecture 3] (3)

2.6 Method of moments for latent-variable models (Lecture 3)

- Motivation
  - For exponential families, we saw that maximum likelihood (choosing $\hat\theta$ to maximize the likelihood of the data) and method of moments (choosing $\hat\theta$ to match the moments of the data) resulted in the same estimator. Furthermore, this estimator can be solved efficiently by convex optimization. Case closed.
  - In this section, we consider parameter estimation in **latent-variable models**, which include examples such as Gaussian mixture models (GMMs), Hidden Markov Models (HMMs), Latent Dirichlet Allocation (LDA), etc. These models are useful in practice because our observed data is often incomplete. For example, in a mixture model, we assume that each data point belongs to a latent cluster (mixture component).
  - While latent-variable models are powerful, parameter estimation in latent-variable models turns out rather tricky. Intuitively, it is because we need to both estimate the parameters and infer the latent variables. We can adapt maximum likelihood to the latent-variable setting by maximizing the marginal likelihood (integrating out the latent variable). However, this is in general a **non-convex** optimization problem.
  - In practice, people often use Expectation Maximization (EM) to (approximately) optimize these objective functions, but EM is only guaranteed to converge to a local optimum, which can be arbitrarily far away from the global optimum. The same problem afflicts other optimization methods such as gradient descent. To get around this problem, one typically optimizes from different random initializations, but this is a heuristic without any theoretical guarantees.

  - In this section, we will explore an alternative technique for parameter estimation based on **method of moments**. We will see that breaking free of the likelihood allows us to get an efficient algorithm with strong formal guarantees.

- Setup
  - We will develop a method of moments estimator for the following mixture model:
  - **Example 1 (Naive Bayes mixture model)**
    - Let k be the number of possible document clusters (e.g., sports, politics, etc.).
    - Let b be the number of possible words in the vocabulary (e.g., "game", "election").
    - let L be the length of a document.
    - Model parameters $\theta = (\pi, B)$:
      - $\pi \in \Delta_k$: prior distribution over clusters.
      - $B = (\beta_1, \dots, \beta_k) \in (\Delta_b)^k$: for each cluster h, $\beta_h \in \Delta_b$ is a distribution over words for cluster h. These are the columns of B, which is a $b \times k$ matrix.

      Let Θ denote the set of all valid parameters.
    - The probability model $p_\theta(h, x)$ is defined as follows:
      - Sample the cluster: $h \sim \text{Multinomial}(\pi)$
      - Sample the words in the document independently: $x = (x_1, \dots, x_L) \mid h \sim \text{Multinomial}(\beta_h)$
  - The parameter estimation problem is as follows: Given n documents $x^{(1)}, \dots, x^{(n)}$ drawn i.i.d. from p_{θ^*}, return an estimate $\hat{\theta} = (\hat{\pi}, \hat{B})$ of $\theta^* = (\pi^*, B^*)$.
  - One minor comment is that we can only hope to recover B^* up to permutation of its columns, since the number of the clusters is arbitrary and any permutation yields the same distribution over the observed variables.
  - Maximum (marginal) likelihood is the standard approach to parameter estimation. For completeness, we will review the procedure, although we will not need it in the sequel, so this part can be skipped. The maximum likelihood estimator is:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} - \log \sum_{h=1}^{k} p_\theta(h, x^{(i)}). \tag{51}$$

    A popular optimization algorithm is to use the EM algorithm, which can be shown to be either a bound optimization algorithm (which repeatedly constructs a lower bound and optimizes) or a coordinate-wise ascent on a related objective:
    - E-step: for each example i, compute the posterior

$$q_i(h) = p_\theta(h^{(i)} = h \mid x^{(i)}). \tag{52}$$

    - M-step: optimize the expected log-likelihood:

$$\max_{\theta} \sum_{i=1}^{n} \sum_{h=1}^{k} q_i(h) \log p_\theta(h, x^{(i)}). \tag{53}$$

  The EM algorithm is widely used and can get excellent empirical results, although there are no theoretical guarantees in general that EM will converge to a global optimum, and in practice, it can get stuck in bad local optima.

- Method of moments
  - Step 1: define a **moment mapping** M relating the model parameters θ to moments m of the data distribution specified by those parameters.
  - Step 2: **plug in** the empirical moments $\hat{m}$ and invert the mapping to get parameter estimates $\hat{\theta}$.

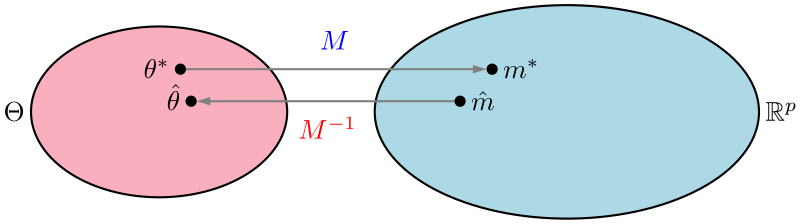

Figure 6: Schema for method of moments: We define a moment mapping M from parameters to moments. We estimate the moments (always easy) and invert the mapping to recover parameter estimates (sometimes easy).

- Moment mapping
  - Let $\phi(x) \in \mathbb{R}^p$ be an observation function which only depends on the observed variables x.
    - Example: $\phi(x) = (x, x^2)$.
  - Define the **moment mapping**

$$\boxed{M(\theta) \stackrel{\text{def}}{=} \mathbb{E}_{x\sim p_\theta}[\phi(x)],} \tag{54}$$

  which maps each parameter vector $\theta \in \mathbb{R}^d$ to the expected value of $\phi(x)$ with respect to $p_\theta(x)$. This mapping is the key that links moments (which are simple functions of the observed data) with the parameters (which are quantities that we want to estimate).
  - Example (Gaussian distribution):

  - $*$ Suppose our model is a univariate Gaussian with parameters $\theta = (\mu, \sigma^2)$.
  - $*$ Then for M defined above, the moment equations are as follows:

$$M((\mu, \sigma^2)) = \mathbb{E}_{x\sim\mathcal{N}(\mu,\sigma^2)}[(x, x^2)] = (\mu, \sigma^2 + \mu^2). \tag{55}$$

  - Let's see how moment mappings are useful. Suppose that someone told us some moments m^* (where $m^* = M(\theta^*)$). Then assuming M were invertible, we could solve for $\theta^* = M^{-1}(m^*)$. Existence of the inverse is known as identifiability: we say that the parameters of a model family Θ are **identifiable** from the moments given by observation function ϕ if M^{-1} exists. Note that for mixture models, strict identifiability never holds, because we can always permute the k cluster labels. So we say that a mixture model is identifiable if $|M^{-1}(m)| = k!$ for all $m \in M(\Theta)$.
  - Example (Gaussian distribution): We can recover the parameters $\theta^* = (\mu^*, \sigma^{2*})$ from $m^* = (m_1^*, m_2^*)$ as follows:
    - $*$ $\mu^* = m_1^*$
    - $*$ $\sigma^{2*} = m_2^* - m_1^{2*}$

    Thus, the parameters are identifiable given the first two moments.

- Plug-in
  - In practice, of course, we don't have access to the true moments m^*. However, the key behind the method of moments is that we can estimate it extremely easily using a sample average over the data points. These are the **empirical moments**:

$$\boxed{\hat{m} \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \phi(x^{(i)}).} \tag{56}$$

  - Given these empirical moments, we can simply **plug in** $\hat{m}$ for m^* to yield the method of moments estimator:

$$\boxed{\hat{\theta} \overset{\text{def}}{=} M^{-1}(\hat{m}).} \tag{57}$$

- Asymptotic analysis
  - How well does this procedure work? It is relatively easy to study how fast $\hat{m}$ converges to m^*, at least in an asymptotic sense. We can show that (i) $\hat{m}$ is close to m^*, and then use that tho show that (ii) $\hat{\theta}$ is close to θ^*. The analysis reflects the conceptual simplicity of the method of moments.
  - First, since $\hat{m}$ is just an average of i.i.d. variables, we can apply the central limit theorem:

$$\sqrt{n}(\hat{m} - m^*) \xrightarrow{d} \mathcal{N}(0, \mathrm{Cov}_{x\sim p^*}[\phi(x)]). \tag{58}$$

  - Second, assuming that M^{-1} is continuous around m^*, we can use the delta method to argue that $\hat{\theta}$ converges θ^*:

$$\boxed{\sqrt{n}(\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \nabla M^{-1}(m^*)\mathrm{Cov}_{x \sim p^*}[\phi(x)]\nabla M^{-1}(m^*)^\top),} \tag{59}$$

    where $\nabla M^{-1}(m^*) \in \mathbb{R}^{d \times p}$ is the Jacobian matrix for the inverse moment mapping M^{-1}. Therefore, the parameter error depends on how sensitive M^{-1} is around m^*. It is also useful to note that $\nabla M^{-1}(m^*) = \nabla M(\theta^*)^\dagger$ (where † denotes pseudoinverse).

  - These asymptotics are rough calculations that shed some light onto when we would expect the method of moments to work well: $\nabla M(\theta) \in \mathbb{R}^{p \times d}$ must have full column rank and moreover, the first d singular values should be far away from zero. Intuitively, if we perturb θ by a little bit, we want m to move a lot, which coincides with our intuitions about the asymptotic error of parameter estimation.

- In general, computing the inverse moment mapping M^{-1} is not easier than the maximum likelihood estimate. Method of moments is only useful if we can find an appropriate observation function ϕ such that:

  - The moment mapping M is invertible, and hopefully has singular values that are bounded below (M provides enough information about the parameters θ).

  - The inverse moment mapping M^{-1} is computationally tractable.

- Computing $\hat{\theta}$ for Example 1

  - Now that we have established the principles behind method of moments, let us tackle the Naive Bayes clustering model (Example 1). The strategy is to just start writing some moments and see what kind of equations we get (they turn out to be product of matrices) (Anandkumar et al., 2012b).

  - Preliminaries

    * Assume each document has $L \geq 3$ words.

    * Assume $b \geq k$ (at least as many words as clusters).

    * We represent each word $x_j \in \mathbb{R}^b$ as an indicator vector which is equal to one in the entry of that word and zero elsewhere.

  - Let's start with the first-order moments:

$$M_1 \overset{\text{def}}{=} \mathbb{E}[x_1] = \sum_{h=1}^{k} \pi_h \beta_h = B\pi. \tag{60}$$

    * Note that the moments require marginalizing out the latent variables, and marginalization corresponds to matrix products.

    * Interpretation: $M_1 \in \mathbb{R}^b$ is a vector of marginal word probabilities.

    - Clearly this is not enough information to identify the parameters.

- We can write the second-order moments:

$$M_2 \overset{\text{def}}{=} \mathbb{E}[x_1 x_2^\top] = \sum_{h=1}^{k} \pi_h \beta_h \beta_h^\top = B\text{diag}(\pi)B^\top. \tag{61}$$

    - Interpretation: $M_2 \in \mathbb{R}^{d \times d}$ is a matrix of co-occurrence word probabilities. Specifically, $M_2(u, v)$ is the probability of seeing word u with word v, again marginalizing out the latent variables.
    - Are the parameters identifiable given M_1 and M_2? It might be tempting to conclude yes, since there are $O(kb)$ parameters and M_2 already specifies $O(b^2)$ equations. But it turns out that the parameters are still non-identifiable from second-order moments (so one has to be careful with parameter counting!) The intuition is that we are trying to decompose M_2 into $AA^\top$, where $A = B\,\text{diag}(\pi)^{1/2}$. However, M_2 also equals $(AQ)(AQ)^\top$ for any orthogonal matrix $Q \in \mathbb{R}^{k \times k}$, so we can only identify A up to rotation. In our setting, A has non-negativity constraints, but there's only $O(k)$ of them as opposed to the $O(k^2)$ degrees of freedom in Q. So we cannot identify the parameters θ from just M_1 and M_2.

- Let us proceed to third-order moments to get more information.

$$M_3(\eta) \overset{\text{def}}{=} \mathbb{E}[x_1 x_2^\top (x_3^\top \eta)] = \sum_{h=1}^{k} \pi_h \beta_h \beta_h^\top (\beta_h^\top \eta) = B\text{diag}(\pi)\,\text{diag}(B^\top \eta)B^\top. \tag{62}$$

  Here, $M_3(\eta)$ is a projection of a rank-3 tensor onto $\eta \in \mathbb{R}^p$. Think of $M_3(\eta)$ as M_2 (they have the same row and column space), but someone has come in and tweaked the diagonal entries. It turns out that this is enough to pin down the parameters. But first, a useful lemma which captures the core computation:

- Lemma
    - Suppose we observe matrices $X = BDB^\top$ and $Y = BEB^\top$ where
        - D, E are diagonal matrices such that the ratios $\{D_{ii}/E_{ii}\}_{i=1}^k$ are all non-zero and distinct.
        - $B \in \mathbb{R}^{b \times k}$ has full column rank (this is a reasonable condition which intuitively says that all the clusters are different in a linear algebriac sense). Note this automatically implies $b \geq k$.
    - Then we can recover B (up to permutation/scaling of its columns).

- Proof:
    - Simple case
        - Assume B is invertible ($b = k$).
        - Then X and Y are also invertible.

        · Compute

$$YX^{-1} = BEB^\top B^{-\top} D^{-1} B^{-1} = B \underbrace{ED^{-1}}_{\text{diagonal}} B^{-1}. \tag{63}$$

        · The RHS has the form of an eigendecomposition, so the eigenvectors of YX^{-1} are be exactly the columns of B up to permutation and scaling. We actually know the scaling since the columns of B are probability distributions and must sum to 1.
        · Since the diagonal elements of ED^{-1} are distinct and non-zero by assumption, the eigenvectors are distinct.
    * Now suppose X and Y are not invertible. Note that X and Y have the same column space, so we can basically project them down into that column space where they will be invertible.
    * Let $U \in \mathbb{R}^{b\times k}$ be any orthonormal basis of the column space of B (taking the SVD of M_2 suffices: $M_2 = USU^\top$). Note that U has full column rank by assumption.
    * Important: although $B \in \mathbb{R}^{b\times k}$ is not invertible, $\tilde{B} \overset{\text{def}}{=} U^\top B \in \mathbb{R}^{k\times k}$ is invertible.
    * So let us project X and Y onto U by both left multiplying by $U^\top$ and right multiplying by U.
    * Then we have the following decomposition:
        · $U^\top XU = \tilde{B}D\tilde{B}^\top$
        · $U^\top YU = \tilde{B}E\tilde{B}^\top$
    * Now we are back in the simple case, which allows us to recover $\tilde{B}$. We can obtain $B = U\tilde{B}$.
  – We apply the lemma with $X = M_2$ and $Y = M_3(\eta)$. Since η is chosen at random, $D = \operatorname{diag}(\pi)$ and $E = \operatorname{diag}(\pi)\operatorname{diag}(B^\top\eta)$ will have distinct ratios with probability 1.
  – Once we have recovered B, then we can recover π easily by setting $\pi = B^\dagger M_1$.
  – We have therefore shown that with infinite data, the method of moments approach recovers the true parameters. For finite data, we have to use a matrix perturbation argument, detailed in Anandkumar et al. (2012b).

- We have so far that we can solve the moment equations for the Naive Bayes clustering model and recover the parameters given the moments. These techniques can be further extended to obtain consistent parameter estimates for spherical Gaussian mixture models (Hsu and Kakade, 2013), hidden Markov models (Anandkumar et al., 2012b), Latent Dirichlet allocation (Anandkumar et al., 2012a), stochastic block models for community detection (Anandkumar et al., 2013), noisy-or Bayesian networks (Halpern and Sontag, 2013), mixture of linear regressions (Chaganty and Liang, 2014), general graphical models (Chaganty and Liang, 2014), neural networks (Janzamin et al., 2015), and others. An active area of research is to extend these techniques to yet other model families.

2.7 Fixed design linear regression (Lecture 3)

- So far, we have considered parameter estimation of θ^* given data $x^{(1)}, \dots, x^{(n)}$ drawn i.i.d. from p_{θ^*}. We will now move towards the **prediction** setting, where we are trying to learn a function from an input x to an output y. Of course, we could still study parameter estimation in this context, but we will use the opportunity to segue into prediction.

- Setup

  - Our goal is to predict a real-valued output (response) $y \in \mathbb{R}$ given an input (covariates/features) $x \in \mathbb{R}^d$.
  - The **fixed design** setting means that we have a fixed set of n inputs $x_1, \dots, x_n$, which are treated as constants. As an example, imagine that $x_1, \dots, x_n$ as the 50 states in America, and $y_1, \dots, y_n$ represent their demographics, weather, etc. Fixed design makes sense here because the states isn't random or growing–they're just fixed. In general, one can think of the fixed design setting as performing signal reconstruction, where the x_i's correspond to fixed locations, and y_i is a noisy sensor reading.
  - We assume that there is a true underlying parameter vector $\theta^* \in \mathbb{R}^d$, which governs the outputs. For each $i = 1, \dots, n$:

    $$y_i = x_i \cdot \theta^* + \epsilon_i, \tag{64}$$

    where we assume that the noise terms ϵ_i are i.i.d. with mean $\mathbb{E}[\epsilon_i] = 0$ and variance $\mathrm{Var}[\epsilon_i] = \sigma^2$. Note that $\epsilon_1, \dots, \epsilon_n$ are the only source of randomness in the problem.
  - FIGURE: [linear regression line over $x_1, \dots, x_n$]
  - At training time, we observe one realization of $y_1, \dots, y_n$. For convenience, let's put the data into matrices:
    - $X = [x_1, \dots, x_n]^\top \in \mathbb{R}^{n \times d}$
    - $\epsilon = [\epsilon_1, \dots, \epsilon_n]^\top \in \mathbb{R}^d$
    - $Y = [y_1, \dots, y_n]^\top \in \mathbb{R}^d$
    - $\Sigma = \frac{1}{n} X^\top X \in \mathbb{R}^{d \times d}$ (second moment matrix)
  - FIGURE: [matrices X, θ, Y]

also called prediction error in statistics

- Our goal is to minimize the **expected risk**[2] as defined by the squared loss:

$$L(\theta) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[(x_i \cdot \theta - y_i)^2] = \frac{1}{n} \mathbb{E}[\|X\theta - Y\|_2^2]. \tag{65}$$

  Note that the expectation is over the randomness in Y (recall that X is fixed).

- The least squares estimator is as follows:

$$\hat{\theta} \overset{\text{def}}{=} \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \|X\theta - Y\|_2^2. \tag{66}$$

  By taking derivatives and setting the result to zero, we obtained the following closed form solution:

  these Y's are to be distinguished from that in (65)

$$\hat{\theta} = X^\top X^{-1} X^\top Y = \frac{1}{n} \Sigma^{-1} X^\top Y, \tag{67}$$

  where we assume that the covariance matrix $\Sigma \overset{\text{def}}{=} \frac{1}{n} X^\top X \in \mathbb{R}^{d \times d}$ is invertible.

- Our goal is to study the expected risk of the least squares estimator:

  r.v. because of the randomness in \hat \theta

$$L(\hat{\theta}). \tag{68}$$

  For simplicity, we will study the expectation $\mathbb{E}[L(\hat{\theta})]$, where the expectation is taken over the training data.

- Let us first understand the expected risk $L(\theta)$ of some fixed θ.

  - The goal will be to understand this quantity geometrically. The basic idea is to expand $Y = X\theta^* + \epsilon$; the rest is just algebra. We have:

$$\begin{aligned}
L(\theta) &= \frac{1}{n} \mathbb{E}[\|X\theta - Y\|_2^2] && (69)\\
&= \frac{1}{n} \mathbb{E}[\|X\theta - X\theta^* + \epsilon\|_2^2] \quad \text{[by definition of } Y \text{ (64)]} && (70)\\
&= \frac{1}{n} \mathbb{E}[\|X\theta - X\theta^*\|_2^2 + \|\epsilon\|_2^2] \quad \text{[cross terms vanish]} && (71)\\
&= \frac{1}{n} (\theta - \theta^*)^\top (X^\top X)(\theta - \theta^*) + \sigma^2 \quad \text{[algebra, definition of } \epsilon\text{]} && (72)\\
&= \|\theta - \theta^*\|_\Sigma^2 + \sigma^2. && (73)
\end{aligned}$$

  weighted by the shape of the features

  - Intuitively, the first term of the expected risk is the squared distance between the estimate θ and the true parameters θ^* as measured by the shape of the data. If the data does not vary much in one direction, then the discrepancy between θ and θ^* will be downweighted in that direction, because that direction doesn't matter for prediction, which depends on $x \cdot \theta$.

[2] We are using expected risk in the machine learning sense, which is a quality metric on parameters, not expected risk in statistics, which is a quality metric on decision procedures.

- The second term is the unavoidable variance term coming from the noise, which is present even with the optimal parameters θ^*. Note that $L(\theta^*) = \sigma^2$.
- In conclusion, the **excess risk**—how far we are from optimal—is:

should distinguish from estimation error

$$\boxed{L(\theta) - L(\theta^*) = \|\theta - \theta^*\|_\Sigma^2.} \tag{74}$$

is a function of theta (and features)

- Let us now analyze the excess risk $L(\hat{\theta}) - L(\theta^*)$ of the least squares estimate $\hat{\theta}$.
  - Assume that $X^\top X \succ 0$ (which means necessarily that $n \geq d$). The key is to expand $\hat{\theta}$ and Y based on their definitions and perform algebra. The expectation is over the test data now. Rewriting the excess risk:

$$\begin{aligned}
L(\hat{\theta}) - L(\theta^*) &= \|\hat{\theta} - \theta^*\|_\Sigma^2 \quad \text{[by (74)]} && (75)\\
&= \frac{1}{n}\|X\hat{\theta} - X\theta^*\|_2^2 && (76)\\
&= \frac{1}{n}\|X(X^\top X)^{-1}X^\top(X\theta^* + \epsilon) - X\theta^*\|_2^2 && (77)\\
&= \frac{1}{n}\|\Pi X\theta^* + \Pi\epsilon - X\theta^*\|_2^2 \quad [\text{projection } \Pi \overset{\text{def}}{=} X(X^\top X)^{-1}X^\top] && (78)\\
&= \frac{1}{n}\|\Pi\epsilon\|_2^2 \quad [\text{projection doesn't change } X\theta^*\text{, cancel}] && (79)\\
&= \frac{1}{n}\operatorname{tr}(\Pi\epsilon\epsilon^\top) \quad [\text{projection is idempotent and symmetric}]. && (80)
\end{aligned}$$

  - Taking expectations (over the training data), and using the fact that $\mathbb{E}[\epsilon\epsilon^\top] = \sigma^2 I$, we get:

$$\boxed{\mathbb{E}[L(\hat{\theta}) - L(\theta^*)] = \frac{d\sigma^2}{n}.} \tag{81}$$

    Note that the complexity of the problem is completely determined by the dimensionality d and the variance of the noise σ^2.
  - Intuitively, the noise ϵ is an n-dimensional vector gets projected onto d dimensions by virtue of having to fit the data using a linear function with d degrees of freedom.

[begin lecture 4] (4)

2.8 General loss functions and random design (Lecture 4)

- Motivation
  - In the previous section, the simplicity of the least squares problem in the fixed design setting allowed us to compute everything exactly. What happens if we are in the random design setting or if we wanted to handle other loss functions (e.g., logistic loss)? We can't hope to compute the excess risk $L(\hat{\theta}) - L(\theta^*)$ exactly.
  - In this unit, we will return to using asymptotics to get a handle on this quantity. In particular, we will show that the excess risk approaches a simple quantity as the number of data points n goes to infinity by performing **Taylor expansions** around θ^*. In brief, we will see that $\hat{\theta} - \theta^*$ is approximately Gaussian with some variance that is $O\left(\frac{1}{n}\right)$, and assuming that L is continuous, we can convert this result into one about the expected risk.
    - ∗ FIGURE: [$\hat{\theta}$ in a ellipsoid Gaussian ball around θ^*]
  - We will carry out the asymptotic analysis in a fairly general setting:
    - ∗ We assume a general (twice-differentiable) loss function.
    - ∗ We do not make any assumptions about the data generating distribution.
- Setup
  - $z = (x, y)$ is an example
  - $\ell(z, \theta)$ is a loss function on example z with parameters $\theta \in \mathbb{R}^d$
    - ∗ Example: $\ell((x, y), \theta) = \frac{1}{2}(\theta \cdot x - y)^2$
  - Let p^* be the true probability distribution over examples $z \in \mathcal{Z}$, not assumed to be related to our loss function in any way.
  - Let $\theta^* \in \mathbb{R}^d$ be the minimizer of the expected risk:

$$\theta^* \stackrel{\text{def}}{=} \arg\min_{\theta \in \mathbb{R}^d} L(\theta), \quad L(\theta) \stackrel{\text{def}}{=} \mathbb{E}_{z \sim p^*}[\ell(z, \theta)] \tag{82}$$

  - Let $\hat{\theta} \in \mathbb{R}^d$ be the minimizer of the empirical risk:

$$\hat{\theta} \stackrel{\text{def}}{=} \arg\min_{\theta \in \mathbb{R}^d} \hat{L}(\theta), \quad \hat{L}(\theta) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell(z^{(i)}, \theta), \tag{83}$$

    where $z^{(1)}, \dots, z^{(n)}$ are drawn i.i.d. from p^*.
  - Recall that we are interested in studying the excess risk $L(\hat{\theta}) - L(\theta^*)$.

- Assumptions on the loss
  - Loss function $\ell(z, \theta)$ is twice differentiable in θ (works for squared and logistic loss, but not hinge)
  - Let $\nabla \ell(z, \theta) \in \mathbb{R}^d$ be the gradient of the loss at θ.
    - Example: $\nabla \ell(z, \theta) = (\theta \cdot x - y)x$
  - Let $\nabla^2 \ell(z, \theta) \in \mathbb{R}^{d \times d}$ be the Hessian of the loss at θ.
    - Example: $\nabla^2 \ell(z, \theta) = xx^\top$
  - Assume that the expected loss Hessian $\mathbb{E}_{z \sim p^*}[\nabla^2 \ell(z, \theta)] \succ 0$ is positive definite for all $\theta \in \mathbb{R}^d$. This assumption is actually not needed, but it will make the math simpler.[3]
- Outline of analysis
  - Step 0 (**consistency**): show that $\hat{\theta} \xrightarrow{P} \theta^*$. This is obtained by a uniform convergence argument to show that $\hat{L}$ approximates L well. Then, since the Hessian $\mathbb{E}[\nabla^2 \ell(z, \theta)]$ is positive definite, minimizing $\hat{L}$ will eventually minimize L. Uniform convergence will be discussed later, so we will not dwell on this point.
  - Step 1: obtain an asymptotic expression for the **parameter error** by Taylor expanding the gradient of the empirical risk.

$$\boxed{\hat{\theta} - \theta^*.} \tag{84}$$

  - Step 2: obtain an asymptotic expression for the **excess risk** by Taylor expanding the expected risk.

$$\boxed{L(\hat{\theta}) - L(\theta^*).} \tag{85}$$

  While we haven't made any assumptions about the relationship between p^* and ℓ, results become a lot nicer if there is. This relationship is captured by the following definition:

- **Definition 3 (well-specified model)**
  - Assume that the loss function corresponds to the log-likelihood under a probabilistic model p_θ:

$$\ell((x, y); \theta) = -\log p_\theta(y \mid x), \tag{86}$$

    so that $\hat{\theta}$ is the (conditional) maximum likelihood estimate under this model.

[3] In fact, if the expected loss is rank deficient, things are actually even better, since the complexity will depend on the rank of the Hessian rather than the dimensionality.

  - We say that this model family $\{p_\theta\}_{\theta \in \mathbb{R}^d}$ is **conditionally well-specified** if $p^*(x, y) = p^*(x) p_{\theta^*}(y \mid x)$ for some parameter $\theta^* \in \mathbb{R}^d$.
  - Suppose each model θ actually specifies a joint distribution over both x and y: $p_\theta(x, y)$. We say that this model family $\{p_\theta\}_{\theta \in \mathbb{R}^d}$ is **jointly well-specified** if $p^*(x, y) = p_{\theta^*}(x, y)$ for some parameter $\theta^* \in \mathbb{R}^d$. This places a much stronger assumption on the data generating distribution. If x is an image and y is a single binary label, this is much harder to satisfy.
  - Of course, jointly well-specified implies conditionally well-specified.

- In the conditionally well-specified case, the Bartlett identity allows us to equate the variance of the risk gradient with the risk Hessian. This quantity is the **Fisher information** matrix (or rather, a generalization of it).

- **Theorem 3 (Bartlett identity)**

  - In the well-specified case (conditionally, and thus also jointly), the following identity holds:

$$\boxed{\nabla^2 L(\theta^*) = \mathrm{Cov}[\nabla \ell(z, \theta^*)].} \tag{87}$$

- Proof of Theorem 3:

  - Recall that $z = (x, y)$.
  - Using the fact that probability densities integrate to one:

$$\int p^*(x) \underbrace{e^{-\ell(z, \theta^*)}}_{p_{\theta^*}(y|x)} dz = 1. \tag{88}$$

  - Assuming regularity conditions, differentiate with respect to θ^*:

$$\int p^*(x) e^{-\ell(z, \theta^*)} (-\nabla \ell(z, \theta^*)) dz = 0. \tag{89}$$

    Note that this implies $\mathbb{E}[\nabla \ell(z, \theta^*)] = 0$, which shouldn't be surprising since θ^* minimizes $L(\theta) = \mathbb{E}[\ell(z, \theta^*)]$.

  - Differentiating again, using the product rule:

$$\int p^*(x) [-e^{-\ell(z, \theta^*)} \nabla^2 \ell(z, \theta^*) + e^{-\ell(z, \theta^*)} \nabla \ell(z, \theta^*) \nabla \ell(z, \theta^*)^\top] dz = 0. \tag{90}$$

  - Re-arranging:

$$\mathbb{E}[\nabla^2 \ell(z, \theta^*)] = \mathbb{E}[\nabla \ell(z, \theta^*) \nabla \ell(z, \theta^*)^\top]. \tag{91}$$

  - Using the fact that $\mathbb{E}[\nabla \ell(z, \theta^*)] = 0$ and the definition of $L(\theta)$ yields the result.

- Remark: our general analysis does not assume the model is well-specified. We will only make this assumption for examples to get simple expressions for intuition.

- **Example 2 (well-specified random design linear regression)**

  - Assume that the conditional model is as follows:
    - $x \sim p^*(x)$ for some arbitrary $p^*(x)$
    - $y = \theta^* \cdot x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$
  - Loss function: $\ell((x, y), \theta) = \frac{1}{2}(\theta \cdot x - y)^2$ (the log-likelihood up to constants)
  - Check that the following two are equal:
    - Hessian: $\nabla^2 L(\theta) = \mathbb{E}[xx^\top]$ (covariance matrix of data)
    - Variance of gradient: $\mathrm{Cov}[\nabla \ell(z, \theta)] = \mathbb{E}[\epsilon x x^\top \epsilon] - \mathbb{E}[\epsilon x]\mathbb{E}[\epsilon x]^\top = \mathbb{E}[xx^\top]$, where the last equality follows from independence of x and ϵ, and the fact that $\mathbb{E}[\epsilon x] = 0$.

Now let us analyze the parameter error (step 1) and expected risk (step 2) in the general (not necessarily well-specified) case. In the following, pay attention how fast each of the terms is going to zero.

- Step 1: analysis of **parameter error**

  - Since ℓ is twice differentiable, we can perform a Taylor expansion of the gradient of the empirical risk ($\nabla \hat{L}$) around θ^*:

$$\nabla \hat{L}(\hat{\theta}) = \nabla \hat{L}(\theta^*) + \nabla^2 \hat{L}(\theta^*)(\hat{\theta} - \theta^*) + O_p(\|\hat{\theta} - \theta^*\|_2^2). \tag{92}$$

  - Using the fact that the LHS $\nabla \hat{L}(\hat{\theta}) = 0$ (by optimality conditions of the empirical risk minimizer) and rearranging:

$$\hat{\theta} - \theta^* = -\nabla^2 \hat{L}(\theta^*)^{-1} \left(\nabla \hat{L}(\theta^*) + O_p(\|\hat{\theta} - \theta^*\|_2^2) \right). \tag{93}$$

  - As $n \to \infty$:
    - By the weak law of large numbers, we have

$$\nabla^2 \hat{L}(\theta^*) \xrightarrow{P} \nabla^2 L(\theta^*). \tag{94}$$

      Since the inverse is a smooth function around θ^* (we assumed $\nabla^2 L(\theta^*) \succ 0$), we can apply the continuous mapping theorem:

$$\nabla^2 \hat{L}(\theta^*)^{-1} \xrightarrow{P} \nabla^2 L(\theta^*)^{-1}. \tag{95}$$

    * $\hat{L}(\theta^*)$ is a sum of mean zero i.i.d. variables, so by the central limit theorem, $\sqrt{n} \cdot \nabla \hat{L}(\theta^*)$ converges in distribution:

$$\sqrt{n} \cdot \nabla \hat{L}(\theta^*) \xrightarrow{d} \mathcal{N}(0, \mathrm{Cov}[\nabla \ell(z, \theta^*)]). \tag{96}$$

      An intuitive implication of this result is that $\nabla \hat{L}(\theta^*) = O_p\left(\frac{1}{\sqrt{n}}\right)$.

    * Suppose $\hat{\theta} - \theta^* = O_p(f(n))$. By (93), $f(n)$ goes to zero at a rate which is the maximum of $\frac{1}{\sqrt{n}}$ and $f(n)^2$. This implies that $f(n) = \frac{1}{\sqrt{n}}$, so we have:

$$\sqrt{n} \cdot O_p(\|\hat{\theta} - \theta^*\|_2^2) \xrightarrow{P} 0. \tag{97}$$

  - By Slutsky's theorem (see Appendix), we can substitute the limits into (93) to obtain:

$$\boxed{\sqrt{n} \cdot \underbrace{(\hat{\theta} - \theta^*)}_{\text{parameter error}} \xrightarrow{d} \mathcal{N}\left(0, \nabla^2 L(\theta^*)^{-1} \mathrm{Cov}[\nabla \ell(z, \theta^*)] \nabla^2 L(\theta^*)^{-1}\right),} \tag{98}$$

  where we used the fact that if $x_n \xrightarrow{d} \mathcal{N}(0, \Sigma)$, then $A x_n \xrightarrow{d} \mathcal{N}(0, A \Sigma A^\top)$. This also establishes that the parameter error behaves $\hat{\theta} - \theta^* = O_p\left(\frac{1}{\sqrt{n}}\right)$ as expected.

    * $\nabla^2 L(\theta^*)$: measures the amount of **curvature** in the loss function at θ^*. The more there is, the more stable the parameter estimates are. This quantity is analogous to the Jacobian of the moment mapping for method of moments.
    * $\mathrm{Cov}[\nabla \ell(z, \theta^*)]$: measures the **variance** in the loss gradient. The less there is, the better. This quantity is analogous to variance of the observation function in the method of moments.
    * When $\sqrt{n}(\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \Sigma)$, Σ is known as the **asymptotic variance** of the estimator $\hat{\theta}$.

  - **Example 3 (well-specified linear regression)**

    * In this case, due to (87), we have $\mathrm{Cov}[\nabla \ell(z, \theta^*)] = \mathbb{E}[\nabla^2 \ell(z, \theta^*)] = \mathbb{E}[x x^\top]$, so the variance factor is canceled out by one of the curvature factors.

$$\boxed{\sqrt{n} \cdot (\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}\left(0, \mathbb{E}[x x^\top]^{-1}\right).} \tag{99}$$

  Intuition: the larger x is, the more stable the parameter estimates; think about wiggling a pencil by either holding it with two hands out at the ends (large x) or near the center (small x).

- Step 2: analysis of excess risk

- Perform a Taylor expansion of the expected risk around θ^*:

  $$L(\hat{\theta}) = L(\theta^*) + \nabla L(\theta^*)^\top (\hat{\theta} - \theta^*) + \frac{1}{2}(\hat{\theta} - \theta^*)^\top \nabla^2 L(\theta^*)(\hat{\theta} - \theta^*) + O_p(\|\hat{\theta} - \theta^*\|_2^3). \tag{100}$$

- By optimality conditions of the expected risk minimizer θ^*, we have $\nabla L(\theta^*) = 0$. This the key to getting $O\left(\frac{1}{n}\right)$ rates of convergence.
- Multiplying by n and rearranging:

  $$n(L(\hat{\theta}) - L(\theta^*)) = \frac{1}{2}\sqrt{n}(\hat{\theta} - \theta^*)^\top \nabla^2 L(\theta^*)\sqrt{n}(\hat{\theta} - \theta^*) + O_p(n\|\hat{\theta} - \theta^*\|_2^3). \tag{101}$$

- Substituting in the parameter error:
  - Facts
    - If $x_n \xrightarrow{d} \mathcal{N}(0, \Sigma)$, then $x_n x_n^\top \xrightarrow{d} \mathcal{W}(\Sigma, 1)$, where $\mathcal{W}(\Sigma, 1)$ is a Wishart distribution with mean Σ and 1 degree of freedom.
    - Taking the trace of both sides, we have that $x_n^\top x_n = \operatorname{tr}(x_n x_n^\top) \xrightarrow{d} \operatorname{tr}(\mathcal{W}(\Sigma, 1))$.[4]
    - The distribution on the RHS is a weighted sum of d chi-squared distributed variables, whose distribution is the same as $\sum_{j=1}^d \Sigma_{jj} v_j^2$, where $v_j \sim \mathcal{N}(0, 1)$ is a standard Gaussian and $v_j^2 \sim \chi_1^2$ is a chi-squared.
  - In our context, let us define

    $$x_n = \sqrt{n}(\nabla^2 L(\theta^*))^{\frac{1}{2}}(\hat{\theta} - \theta^*). \tag{102}$$

    Then

    $$\Sigma = \nabla^2 L(\theta^*)^{-\frac{1}{2}} \operatorname{Cov}[\nabla \ell(z, \theta^*)] \nabla^2 L(\theta^*)^{-\frac{1}{2}}. \tag{103}$$

    Therefore:

    $$\boxed{n(L(\hat{\theta}) - L(\theta^*)) \xrightarrow{d} \frac{1}{2} \operatorname{tr} \mathcal{W}\left(\nabla^2 L(\theta^*)^{-\frac{1}{2}} \operatorname{Cov}[\nabla \ell(z, \theta^*)] \nabla^2 L(\theta^*)^{-\frac{1}{2}}, 1\right),} \tag{104}$$

    where $\mathcal{W}(V, n)$ is the Wishart distribution with scale matrix V and n degrees of freedom.
- **Example 4 (well-specified models)**
  - Since the model is well-specified, everything cancels nicely, resulting in:

    $$\boxed{n(L(\hat{\theta}) - L(\theta^*)) \xrightarrow{d} \frac{1}{2} \operatorname{tr} \mathcal{W}(I_{d \times d}, 1).} \tag{105}$$

[4] We are abusing notation slightly by writing the trace of a distribution D to mean the distribution of the trace of $x \sim D$.

    * The limiting distribution is half times a χ_d^2 distributed random variable, which has mean $\frac{d}{2}$ and variance d.
    * To get an idea of their behavior, we can compute the mean and variance:
        - Mean: $\mathbb{E}[n(L(\hat{\theta}) - L(\theta^*))] \to \frac{d}{2}$.
        - Variance: $\mathrm{Var}[n(L(\hat{\theta}) - L(\theta^*))] \to d$.

      In short,

$$\boxed{L(\hat{\theta}) - L(\theta^*) \sim \frac{d}{2n}.} \tag{106}$$

      Note that this recovers the result from fixed-design linear regression (81) (the factor of $\frac{1}{2\sigma^2}$ is due to the different definition of the loss function).
    * Interestingly, in the well-specified case, the expected risk does not depend asymptotically on any properties of x.
        - For parameter estimation, the more x varies, the more accurate the parameter estimates.
        - For prediction, the more x varies, the harder the prediction problem.
        - The two forces cancel each other out exactly (asymptotically).

- Remarks
    - For this brief section, suppose the model is well-specified.
    - We have shown that excess risk $L(\hat{\theta}) - L(\theta^*)$ is exactly $\frac{d}{2n}$ asymptotically; we emphasize that there are no hidden constants and this is equality, not just a bound.
    - Lower-order terms
        * Of course there could be more error lurking in the lower order $(\frac{1}{n^2})$ terms.
        * For linear regression, the low-order terms $O_p(\|\hat{\theta} - \theta^*\|_2^2)$ in the Taylor expansion are actually zero, and the only approximation comes from estimating the second moment matrix $\mathbb{E}[xx^\top]$.
        * For the fixed design linear regression setting, $\nabla^2 \hat{L}(\theta^*) = \nabla^2 L(\theta^*)$, so all lower-order terms are identically zero, and so our asymptotic expressions are exact.
    - Norm/regularization
        * We only obtained results in the unregularized case. Asymptotically as $n \to \infty$ and the dimension d is held constant, the optimal thing to do (up to first order) is not use regularization.
        * So these results are only meaningful when n is large compared to the complexity (dimensionality) of the problem. This is consistent with the fact that the type of analyses we do are fundamental local around the optimum.

2.9 Regularized fixed design linear regression (Lecture 4)

- So far, we have considered asymptotic analyses of estimators, which holds when the number of examples $n \to \infty$ holding the dimension d of the problem fixed. In this regime, maximum likelihood estimators are the best option (if we can compute them efficiently). However, when d is comparable to n, it turns out that **regularization** can improve the accuracy of our estimates.

- James-Stein estimator (digression)

  - Before diving in to linear regression, let us visit the simpler problem of Gaussian mean estimation, which we started with: given $x^{(1)}, \dots, x^{(n)}$ drawn i.i.d. from $\mathcal{N}(\theta^*, \sigma^2 I)$, what estimator $\hat{\theta}$ should we use?
  - We defined the sample mean estimator $\hat{\theta} = \frac{1}{n} \sum_{i=1}^n x^{(i)}$ (6) and computed its expected MSE to be $\frac{d\sigma^2}{n}$. Can we do better?
  - The answer is yes, surprisingly, as demonstrated by Charles Stein in 1955 in what is famously known as Stein's paradox. The James-Stein estimator (1961) is one popular estimator that improves over the sample mean estimator:

$$\hat{\theta}_{\text{JS}} \overset{\text{def}}{=} \left(1 - \frac{(d-2)\sigma^2}{n\|\hat{\theta}\|_2^2}\right) \hat{\theta}. \tag{107}$$

    In words, this estimator shrinks the sample mean $\hat{\theta}$ by some data-dependent factor towards 0 (0 is not special—any fixed point would work). The amount of shrinkage is governed by the size of the initial estimate $\|\hat{\theta}\|_2^2$; the smaller it is, the more we shrink.
  - For intuition, suppose $\theta^* = 0$. Then $\|\hat{\theta}\|_2^2$ is approximately $\frac{d\sigma^2}{n}$, which provides a massive shrinkage factor of $\frac{2}{d}$. When $\theta^* \neq 0$, then the shrinkage factor goes to 0 as we get more data, which is desired.
  - There is much more to say about the Stein's paradox, but the point of this digression is to show that the standard estimators are often not optimal even when the intuitively feel like they should be.

- Recall the fixed design linear regression setup:

  - We have n examples, inputs are d-dimensional.
  - Design matrix: $X \in \mathbb{R}^{n \times d}$
  - Responses: $Y \in \mathbb{R}$
  - Noise: $\epsilon \in \mathbb{R}$, where components are independent and $\mathbb{E}[\epsilon_i] = 0$ and $\text{Var}[\epsilon_i] = \sigma^2$
  - Assume data satisfies:

$$Y = X\theta^* + \epsilon. \tag{108}$$

- Now consider the **regularized least squares** estimator (ridge regression):

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{n}\|X\theta - Y\|_2^2 + \lambda\|\theta\|_2^2 \tag{109}$$

$$= \frac{1}{n}\Sigma_\lambda^{-1} X^\top Y. \tag{110}$$

  where

$$\Sigma_\lambda \overset{\text{def}}{=} \frac{1}{n} X^\top X + \lambda I. \tag{111}$$

  Here, $\lambda \geq 0$ is the regularization strength that controls how much we want to shrink the estimate towards 0 to guard against overfitting. Intuitively, the more data points we have (larger n), the less we should regularize (smaller λ), and vice-versa. But what precisely should λ be and what is the resulting expected risk? The subsequent analysis will provide answers to these questions.

- The first main insight is the **bias-variance tradeoff**, whose balance is determined by λ. Let us decompose the excess risk:

$$\mathbb{E}[\|\hat{\theta} - \theta^*\|_\Sigma^2] = \mathbb{E}[\|\hat{\theta} - \mathbb{E}[\hat{\theta}] + \mathbb{E}[\hat{\theta}] - \theta^*\|_\Sigma^2] \tag{112}$$

$$= \underbrace{\mathbb{E}[\|\hat{\theta} - \mathbb{E}[\hat{\theta}]\|_\Sigma^2]}_{\overset{\text{def}}{=}\text{Var}} + \underbrace{\|\mathbb{E}[\hat{\theta}] - \theta^*\|_\Sigma^2}_{\overset{\text{def}}{=}\text{Bias}^2}, \tag{113}$$

  where the cross terms are designed to cancel out. Note that in the unregularized case ($\lambda = 0$), the bias is zero (provided $\Sigma \succ 0$) since $\mathbb{E}[\hat{\theta}] = (X^\top X)^{-1} X^\top (X\theta^* + \mathbb{E}[\epsilon]) = \theta^*$, but when $\lambda > 0$, the bias will be non-zero.

- The second main insight is that the risk of the regularized least squares estimate on the original problem is the same as the risk of an equivalent problem which has been **rotated** into a basis that is easier to analyze.

  - Suppose we rotate each data point x_i by an orthogonal matrix $R \in \mathbb{R}^{d\times d}$ ($x_i \mapsto R^\top x_i$), so that $X \mapsto XR$ for some orthogonal matrix R. Correspondingly, we must rotate the parameters $\theta^* \mapsto R^\top \theta^*$. Then the claim is that the excess risk does not change.
  - The excess risk of the modified problem is:

$$\mathbb{E}[\|XR(R^\top X^\top XR + n\lambda I)^{-1} R^\top X^\top (XRR^\top \theta^* + \epsilon) - XRR^\top \theta^*\|_2^2]. \tag{114}$$

    Simplification reveals that we get back exactly the original excess risk:

$$\mathbb{E}[\|X(X^\top X + n\lambda I)^{-1} X^\top (X\theta^* + \epsilon) - X\theta^*\|_2^2]. \tag{115}$$

    This equivalence holds for any orthogonal matrix R.

  - If we take the SVD $X = USV^\top$ and rotate by $R = V$, then we can see that $X^\top X \mapsto (V^\top VSU^\top)(USV^\top V) = S^2$, which is diagonal. Therefore, for the purposes of analysis, we can assume that Σ is diagonal without loss of generality:

$$\Sigma = \operatorname{diag}(\tau_1, \dots, \tau_d). \tag{116}$$

    Diagonal covariances have the advantage that we can analyze each component separately, turning matrix computations into independent scalar computations.

- Let us compute the mean of the estimator:

$$\begin{aligned}
\bar{\theta}_j &\overset{\text{def}}{=} \mathbb{E}[\hat{\theta}_j] && (117)\\
&= \mathbb{E}[\Sigma_\lambda^{-1} n^{-1} X^\top (X\theta^* + \epsilon)]_j \quad [\text{expand } Y] && (118)\\
&= \mathbb{E}[\Sigma_\lambda^{-1}\Sigma\theta^* + \Sigma_\lambda^{-1} n^{-1} X^\top \epsilon]_j \quad [\text{algebra}] && (119)\\
&= \frac{\tau_j}{\tau_j + \lambda}\theta_j^* \quad [\text{since } \mathbb{E}[\epsilon] = 0]. && (120)
\end{aligned}$$

  Thus, the expected value of the estimator is the true parameter value θ^* shrunk towards zero by a strength that depends on λ.

- Compute the squared bias term of the expected risk:

$$\begin{aligned}
\text{Bias}^2 &= \|\bar{\theta} - \theta^*\|_\Sigma^2 && (121)\\
&= \sum_{j=1}^d \tau_j \left(\frac{\tau_j}{\tau_j + \lambda}\theta_j^* - \theta_j^*\right)^2 && (122)\\
&= \sum_{j=1}^d \frac{\tau_j \lambda^2 (\theta_j^*)^2}{(\tau_j + \lambda)^2}. && (123)
\end{aligned}$$

  If $\lambda = 0$, then the squared bias is zero as expected. As $\lambda \to \infty$, the squared bias tends to $\|\theta^*\|_2^2$, reflecting the fact that the estimator $\hat{\theta} \to 0$.

- Compute the variance term of the expected risk:

$$\begin{aligned}
\text{Var} &= \mathbb{E}[\|\hat{\theta} - \bar{\theta}\|_\Sigma^2] && (124)\\
&= \mathbb{E}[\|\Sigma_\lambda^{-1} n^{-1} X^\top \epsilon\|_\Sigma^2] \quad [\text{definition of } \hat{\theta}] && (125)\\
&= \frac{1}{n^2}\mathbb{E}[\epsilon^\top X \Sigma_\lambda^{-1} \Sigma \Sigma_\lambda^{-1} X^\top \epsilon] \quad [\text{expand}] && (126)\\
&= \frac{1}{n^2}\operatorname{tr}(\Sigma_\lambda^{-1} \Sigma \Sigma_\lambda^{-1} X^\top \mathbb{E}[\epsilon\epsilon^\top] X) \quad [\text{cyclic trace}] && (127)\\
&= \frac{\sigma^2}{n}\operatorname{tr}(\Sigma_\lambda^{-1} \Sigma \Sigma_\lambda^{-1} \Sigma) \quad [\text{since } \mathbb{E}[\epsilon\epsilon^\top] = \sigma^2 I] && (128)\\
&= \frac{\sigma^2}{n}\sum_{j=1}^d \frac{\tau_j^2}{(\tau_j + \lambda)^2} \quad [\text{matrices are diagonal}]. && (129)
\end{aligned}$$

If we didn't regularize, the variance would be $\frac{d\sigma^2}{n}$. Regularization reduces the variance since $\frac{\tau_j^2}{(\tau_j+\lambda)^2} \le 1$.

- Balancing squared bias and variance
  - Having computed the squared bias and variance terms, we can now try to choose λ to minimize their sum, the expected risk.
  - Rather than minimize their sum as a function as λ, which would be annoying, let us upper bound both terms to get simpler expressions.
  - Fact: $(a+b)^2 \ge 2ab$.
  - Upper bound the squared bias and variance by replacing the denominators with a smaller quantity:

$$\mathrm{Bias}^2 \le \sum_{j=1}^d \frac{\lambda(\theta_j^*)^2}{2} = \frac{\lambda\|\theta^*\|_2^2}{2} \tag{130}$$

$$\mathrm{Var} \le \sum_{j=1}^d \frac{\tau_j \frac{\sigma^2}{n}}{2\lambda} = \frac{\mathrm{tr}(\Sigma)\sigma^2}{2n\lambda}. \tag{131}$$

  - Now we can minimize the sum over the upper bounds with respect to λ.
  - This pattern shows up commonly:
    - ∗ Suppose we want to minimize $a\lambda + \frac{b}{\lambda}$.
    - ∗ Then the optimal $\lambda = \sqrt{\frac{b}{a}}$ yields $2\sqrt{ab}$.
  - Optimizing yields an bound on the excess risk:

$$\boxed{\mathbb{E}[L(\hat{\theta}) - L(\theta^*)] \le \sqrt{\frac{\|\theta^*\|_2^2\,\mathrm{tr}(\Sigma)\sigma^2}{n}},} \tag{132}$$

  with the regularization strength set to:

$$\lambda = \sqrt{\frac{\mathrm{tr}(\Sigma)\sigma^2}{\|\theta^*\|_2^2 n}}. \tag{133}$$

- Remarks
  - FIGURE: [spectra]
  - Let us compare the risk bound for regularized least squares (133) with that of ordinary least squares (81), which is $\frac{d\sigma^2}{n}$.

- The main observation is that the bound no longer depends on the dimensionality d. Instead, we have a more nuanced dependency in terms of $\mathrm{tr}(\Sigma)$, which can be thought of as the "true dimensionality" of the problem. Note that $\mathrm{tr}(\Sigma)$ is the sum of the eigenvalues. If the data can be described (say, via PCA) by a few dimensions, then the excess risk is governed by this effective dimension. The upshot is that we can be in very high dimensions, but so long as we regularize properly, then our risk will be well-behaved.
- On the other hand, what we pay is that the excess risk is now $O(\sqrt{\frac{1}{n}})$, which is slower than the previous rate of $O(\frac{1}{n})$.
- In the random design where X is a random variable, things are more complicated. In that setting, $X^\top X$ would be random and different from the expected covariance $\mathbb{E}[X^\top X]$. We would need to argue that the two converge. See the Hsu/Kakade/Zhang paper for more details.

2.10 Summary (Lecture 4)

- This concludes our unit on asymptotic analysis. The central problem is this: given data, $x^{(1)}, \dots, x^{(n)}$, we get an estimator $\hat{\theta}$. How well does this estimator do, either in terms of parameter error for estimation ($\|\hat{\theta} - \theta^*\|_2^2$) or expected risk for prediction ($L(\theta)$)? We were able to get quantities that depend on the key quantities of interest: number of examples n, dimensionality d, noise level σ^2.

- At a high-level, what makes everything work is the central limit theorem.
  - This is most obvious for estimating the means of Gaussians and multinomial distributions.
  - For exponential families, where the canonical parameters are a non-linear function of the moments, we can apply the delta method to get asymptotic normality of the canonical parameters.
  - For general loss functions, we need to apply Slutsky's theorem because the non-linear mapping (governed by $\nabla^2 L(\theta^*)$) is also being estimated.

- We considered estimators based on the method of moments as an alternative to maximum likelihood. For exponential families, these two estimators are the same, but for more complex models such as mixture models, it is possible to get method of moments estimators that are computationally efficient to compute, whereas the maximum likelihood estimator would involve solving a non-convex optimization problem. The high-level point is that the design space of estimators is quite large, and for many problem settings, it's an open question as to what the right estimator to use is. We can evaluate these estimators based on computational efficiency, and asymptotics provides a way to evaluate them based on statistical efficiency.

- Finally, one weakness of the classical asymptotics that we've studied here is that it assumes that the number of examples n is substantially larger than the dimensionality d of the problem. If this is not true, then we need to employ regularization to guard against overfitting. We gave one simple example (fixed design linear regression) where we could obtain a meaningful expected risk even when $d \gg n$, as long as norms were bounded. In the next unit, we will develop the theory of uniform convergence, which will allow us to analyzed regularized estimators much more generally.

2.11 References

- Sham Kakade's statistical learning theory course

- Hsu/Kakade/Zhang, 2014: Random design analysis of ridge regression

- van der Vaart, 2000: Asymptotic Statistics

- Csiszár/Shields, 2004: Information Theory and Statistics: A Tutorial

- Anandkumar/Hsu/Kakade, 2012: A Method of Moments for Mixture Models and Hidden Markov Models

- Anandkumar/Ge/Hsu/Kakade/Telgarsky, 2012: Tensor Decompositions for Learning Latent Variable Models

3 Uniform convergence

[begin lecture 5] (5)

3.1 Overview (Lecture 5)

- The central question is:

  Why does minimizing training error reduce test error?

  The answer is not obvious for the training error and test error are two separate quantities which can in general be arbitrarily far apart. This deep question is at the core of statistical learning theory, and answering it reveals what it means to learn.

- In the previous unit on asymptotics, we analyzed the performance of learning algorithms (estimators) in the regime where the number of parameters d is fixed and the number of examples n grows. There are two deficiencies of this analysis:

  - It doesn't tell you how large n has to be before the "asymptotics kick in." This is problematic in the high-dimensional settings of modern statistics and machine learning, where d is actually fairly large compared to n.
  - It applies only to unregularized estimators operating on smooth loss functions. However, we want to analyze things like the zero-one loss for classification. We also want to consider different forms of regularization like L_1 regularization for performing feature selection.

  We did analyze regularized least squares at the end of last unit, but that analysis was very specific to linear regression. How can we generalize to other problems and estimators?

- In this unit, we develop a new suite of tools to answer this question. We will shift our focus from asymptotic variance of an estimator to thinking about estimators that choose a predictor from a **hypothesis class**. We then study how the **empirical risk** (a.k.a. training error, which our estimator is based on) relates to the **expected risk** (a.k.a. test error, which we're evaluated on) on this hypothesis class using **uniform convergence**. In the process, we will develop some fairly general machinery from probability theory; these tools are more broadly applicable outside machine learning. These techniques are mathematically quite involved, so make sure you have a good understanding of probability!

3.2 Formal setup (Lecture 5)

- In this section, we formalize the (batch) supervised learning setting. Much of what we will do also works for unsupervised learning, but we will describe it in the context of supervised learning to provide intuition.

- Consider the problem of predicting an output $y \in \mathcal{Y}$ given an input $x \in \mathcal{X}$. Example: $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1, +1\}$.

- Let $\mathcal{H}$ be a set of **hypotheses**. Usually, each $h \in \mathcal{H}$ maps $\mathcal{X}$ to $\mathcal{Y}$. Example: $\mathcal{H} = \{x \mapsto \text{sign}(w \cdot x) : w \in \mathbb{R}^d\}$ is all thresholded linear functions.

- Let $\ell : (\mathcal{X} \times \mathcal{Y}) \times \mathcal{H} \to \mathbb{R}$ be a **loss function**. Example: $\ell((x, y), h) = \mathbb{I}[y \neq h(x)]$ is the zero-one loss.

- Let p^* denote the true underlying data-generating distribution over input-output pairs $\mathcal{X} \times \mathcal{Y}$.

- **Definition 4 (expected risk)**

  - Let $L(h)$ be the **expected risk** (test error) of a hypothesis $h \in \mathcal{H}$, which is the loss that h incurs on a new test example (x, y) in expectation:

$$\boxed{L(h) \overset{\text{def}}{=} \mathbb{E}_{(x,y) \sim p^*}[\ell((x, y), h)].} \tag{134}$$

    Getting low expected risk is in some sense the definition of successful learning.

  - Define an **expected risk minimizer** h^* to be any hypothesis that minimizes the expected risk:

$$h^* \in \arg\min_{h \in \mathcal{H}} L(h). \tag{135}$$

    This is the thing that we can only aspire to. $L(h^*)$ is the lowest possible expected risk (which might be large if the learning problem is noisy or your hypothesis class is too small).

- To do learning, we are given n **training examples**, which are a set of input-output pairs:

$$(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)}), \tag{136}$$

  where each $(x^{(i)}, y^{(i)})$ is drawn **i.i.d.** from p^*.

  - Note: the training and test distributions are the same. While this assumption often doesn't hold exactly in practice, the training and test distributions morally have to be related somehow, or else there's no hope that what we learned from training would be useful at test time.[5]

  - Note: the independence assumption, which also doesn't hold exactly in practice, ensures that more training data gives us more information (or else we would get the same training example over and over again, which would be useless).[6]

[5] If the two distributions are different but still related somehow, not all hope is lost; dealing with this discrepancy is called domain adaptation.

[6] Pure i.i.d. is not necessary, but certainly some amount of independence is necessary.

- **Definition 5 (empirical risk)**
  - Let $\hat{L}(h)$ be the **empirical risk** (training error) of a hypothesis $h \in \mathcal{H}$ as the average loss over the training examples:

$$\boxed{\hat{L}(h) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell((x^{(i)}, y^{(i)}), h).} \tag{137}$$

    Note that for a fixed h, $\hat{L}(h)$ is just an empirical average with mean $L(h)$. This is key.

  - Define an **empirical risk minimizer** (ERM) be any hypothesis that minimizes the empirical risk:

$$\hat{h} \in \arg\min_{h \in \mathcal{H}} \hat{L}(h). \tag{138}$$

- Let us pause a moment to remember what are the random variables and what the independence assumptions are: $\hat{h}$ is a random variable that depends the training examples (in a rather complicated way), but h^* is non-random. The empirical risk $\hat{L}(h)$ is a random variable for each h, but the expected risk $L(h)$ is non-random.

- Recall that we are interested in the expected risk of the ERM:

$$L(\hat{h}). \tag{139}$$

  We will not study this quantity directly, but rather look at its difference with a baseline. There are two questions we can ask:

  - How does the expected and empirical risks compare for the ERM?

$$\underbrace{L(\hat{h})}_{\text{expected risk of ERM}} - \underbrace{\hat{L}(\hat{h})}_{\text{empirical risk of ERM}}. \tag{140}$$

  - How well is ERM doing with respect to the best in the hypothesis class?

$$\underbrace{L(\hat{h})}_{\text{expected risk of ERM}} - \underbrace{L(h^*)}_{\text{lowest expected risk}}. \tag{141}$$

    This is know as as the **excess risk**.

- How do we analyze the excess risk? The excess risk is a random quantity that depends on the training examples (through $\hat{h}$). It is possible that the excess risk is high even for large n (for instance, if we just happened to see the same example over and over again). So we can't deterministically bound the excess risk. Note that in asymptotics, we dealt with this via the central limit theorem, but now n is an arbitrarily small finite number.

- Fortunately, we can show that bad outcomes are not too likely. We will prove bounds of the following flavor: With probability at least $1-\delta$, the excess risk is upper bounded by some ϵ ($L(\hat{h}) - L(h^*) \le \epsilon$), where ϵ is generally a function that depends on δ (and other aspects of the learning problem). More formally, the types of statements we'd like to show can be written compactly as:

$$\boxed{\mathbb{P}[L(\hat{h}) - L(h^*) > \epsilon] \le \delta.} \tag{142}$$

  - FIGURE: [distribution over $L(\hat{h})$, δ is area of tail, ϵ is difference on x-axis from $L(h^*)$]

  Note that the randomness in the probability is over draws of the n training examples: $(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)}) \sim p^*$.

- It is important to note that there are two sources of randomness at play here:
  - Expected risk (upper bounded by ϵ) is defined with respect to randomness over the test example.
  - Confidence (lower bounded by $1-\delta$) is defined with respect to randomness over the training examples.

- Probably Approximately Correct (PAC) framework [Leslie Valiant, 1984]
  - A learning algorithm $\mathcal{A}$ PAC learns $\mathcal{H}$ if for any distribution p^* over $\mathcal{X} \times \mathcal{Y}$, $\epsilon > 0$, $\delta > 0$, $\mathcal{A}$ (which takes as input n training examples along with ϵ and δ), returns $\hat{h} \in \mathcal{H}$ such that with probability at least $1-\delta$, $L(\hat{h}) - L(h^*) \le \epsilon$, and $\mathcal{A}$ runs in $\text{poly}(n, \text{size}(x), 1/\epsilon, 1/\delta)$ time.
  - Remark: time complexity upper bounds sample complexity, because you have to go through all the data points. In this class, we will not focus so much on the computational aspect in our presentation, but just work with the ERM, and assume that it can be optimized efficiently (even though that's not true for general non-convex losses).
  - The ideas in PAC learning actually predate Valiant and were studied in the field of empirical process theory, but Valiant and others focused more on more combinatorial problems with computation in mind.

3.3 Realizable finite hypothesis classes (Lecture 5)

- So far, we have presented a framework that is very general, and in fact, so general, that we can't say anything. So we need to make some assumptions. Ideally, the assumptions would be both realistic (not too strong) but still allow us to prove something interesting (not too weak).

- In this section, we will start with an easy case, where we have a finite number of hypotheses, at least one of which has zero expected risk. These assumptions are as formalized as follows:

- **Assumption 1 (finite hypothesis space)**

  - Assume $\mathcal{H}$ is finite.

- **Assumption 2 (realizable)**

  - Assume there exists a hypothesis $h^* \in \mathcal{H}$ that obtains zero expected risk, that is:

$$L(h^*) = \mathbb{E}_{(x,y)\sim p^*}[\ell((x,y),h^*)] = 0. \tag{143}$$

- **Theorem 4 (realizable finite hypothesis class)**

  - Let $\mathcal{H}$ be a hypothesis class, where each hypothesis $h \in \mathcal{H}$ maps some $\mathcal{X}$ to $\mathcal{Y}$.
  - Let ℓ be the zero-one loss: $\ell((x,y),h) = \mathbb{I}[y \neq h(x)]$.
  - Let p^* be any distribution over $\mathcal{X} \times \mathcal{Y}$.
  - Assume Assumptions 1 and 2 hold.
  - Let $\hat{h}$ be the empirical risk minimizer.
  - Then the following two equivalent statements hold (each has a different interpretation depending on what we're interested in):
    - * Interpretation 1: what is the error after training on n examples (**expected risk**)? Answer: with probability at least $1-\delta$,

$$\boxed{L(\hat{h}) \leq \frac{\log|\mathcal{H}| + \log(1/\delta)}{n}.} \tag{144}$$

      Usually, think of $\log(1/\delta)$ as a constant (e.g., $\delta = 0.01$, then $\log(1/\delta) \approxeq 4.6$), so the

$$\underbrace{L(\hat{h})}_{\text{expected risk}} = O\left(\frac{\overbrace{\log|\mathcal{H}|}^{\text{complexity}}}{\underbrace{n}_{\text{number of training examples}}}\right) \tag{145}$$

    - * Interpretation 2: how many examples n (**sample complexity**) do I need to obtain expected risk at most ϵ with confidence at least $1-\delta$? Answer: With probability at least $1-\delta$:

$$\boxed{n \geq \frac{\log|\mathcal{H}| + \log(1/\delta)}{\epsilon} \quad \Rightarrow \quad L(\hat{h}) \leq \epsilon.} \tag{146}$$

- Remarks
  - Statisticians generally talk about error, and computer scienists like to talk about sample complexity. But they're just two sides of the same coin.
  - In this case, the excess risk behaves as $O(1/n)$, which is known as a "fast" rate. This is because we've assumed realizability: as soon as h makes even a single mistake on a training example, we can throw it away.
  - The excess risk only grows logarithmically with $|\mathcal{H}|$, so we can use pretty big hypothesis classes.
  - Note that our result is independent of $p^*(x, y)$. This is known as a **distribution-free** result, which is great, because typically we don't know what p^* is.
- Proof of Theorem 4
  - FIGURE: [a row of hypotheses, sorted by increasing $L(h)$]
  - We'd like to upper bound the probability of the bad event that $L(\hat{h}) > \epsilon$.
  - Let $B \subseteq \mathcal{H}$ be the set of bad hypotheses h: $B = \{h \in \mathcal{H} : L(h) > \epsilon\}$. We can rewrite our goal as upper bounding the probability of selecting a bad hypothesis:

$$\mathbb{P}[L(\hat{h}) > \epsilon] = \mathbb{P}[\hat{h} \in B]. \tag{147}$$

  - Recall that the empirical risk of the ERM is always zero ($\hat{L}(\hat{h}) = 0$) because at least $\hat{L}(h^*) = L(h^*) = 0$. So if we selected a bad hypothesis ($\hat{h} \in B$), then some bad hypothesis must have zero empirical risk:

$$\mathbb{P}[\hat{h} \in B] \le \mathbb{P}[\exists h \in B : \hat{L}(h) = 0]. \tag{148}$$

  - We now get to the heart of the argument, which consists of two steps.
  - Step 1: bound $\mathbb{P}[\hat{L}(h) = 0]$ for a **fixed** $h \in B$.
    - On each example, hypothesis h does not err with probability $1 - L(h)$.
    - Since the training examples are i.i.d. and the fact that $L(h) > \epsilon$ for $h \in B$:

$$\mathbb{P}[\hat{L}(h) = 0] = (1 - L(h))^n \le (1 - \epsilon)^n \le e^{-\epsilon n}, \tag{149}$$

      where the last step follows since $1 - a \le e^{-a}$.
    - Remark: this probability **decreases exponentially** with n, which is important.
  - Step 2: show that step 1 holds simultaneously for all $h \in B$:
    - We apply the **union bound** to bound the probability of the event for any $h \in B$:

$$\mathbb{P}\Big[\exists h \in B : \hat{L}(h) = 0\Big] \le \sum_{h \in B} \mathbb{P}[\hat{L}(h) = 0]. \tag{150}$$

- The rest is straightforward:

$$\begin{aligned}\sum_{h\in B}\mathbb{P}[\hat{L}(h)=0] &\le |B|e^{-\epsilon n} && (151)\\ &\le |\mathcal{H}|e^{-\epsilon n} && (152)\\ &\stackrel{\text{def}}{=} \delta. && (153)\end{aligned}$$

- Taking logs of the last equality and rearranging:

$$\epsilon = \frac{\log|\mathcal{H}| + \log(1/\delta)}{n}. \quad (154)$$

The theorem follows by substuting this expression for δ.

3.4 Generalization bounds via uniform convergence (Lecture 5)

- The proof of Theorem 4 is elementary but illustrates an important pattern that will recur again in more complex scenarios. At a high level, we are interested in expected risk L, but only have access to empirical risk $\hat{L}$ to choose the ERM $\hat{h}$. In the proof, we saw two steps:

  - Step 1 (convergence): For a **fixed** h, show that $\hat{L}(h)$ is close to $L(h)$ with high probability. In the above proof, this meant showing that if $L(h) > \epsilon$, then $\hat{L}(h) = 0$ is unlikely.
  - Step 2 (uniform convergence): Show that the above holds simultaneously for **all** hypotheses $h \in \mathcal{H}$. In the above proof, this meant using a union bound.

  The difference between convergence and uniform convergence is absolutely crucial. It is important to note that $\hat{h}$ is a random variable that depends on the training examples, so $\hat{L}(\hat{h})$ is not just a sum of i.i.d. variables, so step 1 does not apply directly.

- Theorem 4 also made two restrictive assumptions.

  - First, there exists a perfect hypothesis (realizability). What happens when the problem is not realizable (all hypotheses make some error)? To answer this, we consider the general problem of convergence of random variables using **concentration inequalities**.
  - Second, the hypothesis class is finite. What happens when the number of hypotheses is infinite? We can't just apply a union bound any more. To answer this, we need to have more suitable ways of measuring the "size" of a set other than cardinality. This leads to **Rademacher complexity**, **VC dimension**, and **covering numbers** as ways for measuring the "size" of an infinite set from the point of view of the loss function.

- Breaking free of these restrictive assumptions, we will show how bounding expected risk can be reduced to one of uniform convergence. Recall that our goal is to bound the excess risk, the amount by which ERM's expected risk exceeds the lowest possible expected risk:

$$\mathbb{P}[L(\hat{h}) - L(h^*) \geq \epsilon] \leq \delta. \tag{155}$$

  Note that the difference between $\geq$ and $>$ isn't important here, and it will be more convenient to use $\geq$.

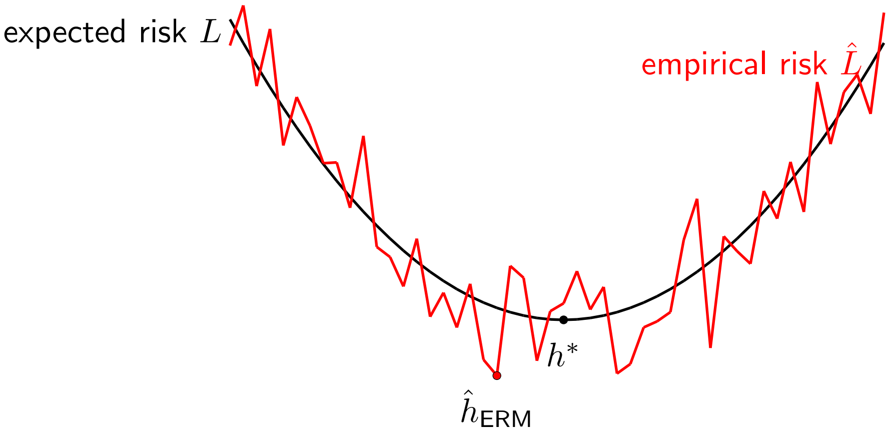

Figure 7: Uniform convergence.

- Let us first relate expected risk to empirical risk, since that's what the ERM is defined in terms of:

$$L(\hat{h}) - L(h^*) = \underbrace{[L(\hat{h}) - \hat{L}(\hat{h})]}_{\text{concentration}} + \underbrace{[\hat{L}(\hat{h}) - \hat{L}(h^*)]}_{\leq 0} + \underbrace{[\hat{L}(h^*) - L(h^*)]}_{\text{concentration}}. \tag{156}$$

- The second term is non-positive by definition of the empirical risk minimizer.

- The third term involves a comparison of $\hat{L}(h^*)$ and $L(h^*)$. If we expand things, we realize that this is just a question of the difference between an average of n i.i.d. random variables and its mean:

$$\hat{L}(h^*) = \frac{1}{n}\sum_{i=1}^{n} \ell((x^{(i)}, y^{(i)}), h^*), \quad L(h^*) = \mathbb{E}_{(x,y)\sim p^*}[\ell((x,y), h^*)]. \tag{157}$$

  We'll see how concentration inequalities can be used to control this difference.

- What about the first term? The same reasoning doesn't apply because $\hat{h}$ depends on the training examples, and so $\hat{L}(\hat{h})$ is not a sum of i.i.d. random variables! We'll need

~~something~~ something more sophisticated: uniform convergence. Suppose we could ensure that $\hat{L}(h)$ and $L(h)$ were close (say within $\frac{\epsilon}{2}$) *for all* $h \in \mathcal{H}$. Then, we could ~~be~~ ensure that $\hat{L}(\hat{h})$ and $L(\hat{h})$ were within $\frac{\epsilon}{2}$, as well as $\hat{L}(h^*)$ and $L(h^*)$.

- The contrapositive can be written formally as:

$$\boxed{\mathbb{P}[L(\hat{h}) - L(h^*) \geq \epsilon] \leq \mathbb{P}\left[\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h)| \geq \frac{\epsilon}{2}\right].} \tag{158}$$

  On the LHS is a statement about excess risk, and on the RHS is a statement about uniform convergence. The RHS is the probability of the event that the largest difference between the empirical and expected risk is at least $\frac{\epsilon}{2}$, or equivalently, the event that this difference exceeds $\frac{\epsilon}{2}$ for at least one $h \in \mathcal{H}$.

- Note: the classic example of uniform convergence is the Glivenko-Cantelli theorem (also called the uniform law of large numbers), for estimating distribution functions. Given $x_1, \dots, x_n$ drawn i.i.d. from some distribution with cumulative distribution function (CDF) $F(x)$, we can form the empirical CDF $F_n(x) = \frac{1}{n}\sum_{i=1}^n \mathbb{I}[x \leq x_i]$. One can ask for the convergence of the CDF function in the uniform norm:

$$\|F_n - F\|_\infty \overset{\text{def}}{=} \sup_{x \in \mathbb{R}} |F_n(x) - F(x)| \xrightarrow{P} 0. \tag{159}$$

  What we will be studying is a generalization of (159), where we have arbitrary hypotheses $h \in \mathcal{H}$ rather than $x \in \mathbb{R}$.

- Note: if we look at the difference between $\hat{L}$ and L, we can construct something called an **empirical process**:

$$\{G_n(h)\}_{h \in \mathcal{H}}, \quad G_n \overset{\text{def}}{=} \sqrt{n}(\hat{L}(h) - L(h)), \tag{160}$$

  which is a stochastic process (collection of random variables indexed by $h \in \mathcal{H}$). Empirical process theory focuses on studying empirical processes. We know that for a given $h \in \mathcal{H}$, $G_n(h)$ converges to a normal distribution by the central limit theorem. We can think about G_n converging to a Gaussian process G with covariance function

$$\mathrm{Cov}[G(h), G(h')] = \mathrm{Cov}[\ell(z,h), \ell(z,h')]. \tag{161}$$

  This stochastic process viewpoint allows one to talk not just about the supremum $\sup_{h \in \mathcal{H}} G_n(h)$, but also get distributional results, which is useful for computing confidence intervals. This is outside the scope of the class; for additional information, Pollard has an excellent book on this.

3.5 Concentration inequalities (Lecture 5)

- Concentration inequalities are a very powerful set of techniques from probability theory that shows that an appropriate combination of independent random variables will *concentrate* around its expectation. From the point of view of learning theory, the random variables of interest are the losses of hypotheses on training examples.

- Mean estimation
  - Let $X_1, \dots, X_n$ be i.i.d. real-valued random variables with mean $\mu \overset{\text{def}}{=} \mathbb{E}[X_1]$
  - Define the empirical mean as follows:

$$\hat{\mu}_n \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} X_i \tag{162}$$

  - Question: how does $\hat{\mu}_n$ relate to μ?
  - Examples
    - ∗ X_i is the height of the i-th person sampled from some population.
    - ∗ X_i is the loss of a *fixed* hypothesis $h \in \mathcal{H}$ on the i-th example.

- FIGURE: [μ with distribution over $\hat{\mu}_n$]

- Types of statements
  - **Consistency**: by the law of large numbers,

$$\hat{\mu}_n - \mu \xrightarrow{P} 0, \tag{163}$$

    where $\xrightarrow{P}$ denotes convergence in probability.[7] Consistency assures us that as we get more data ($n \to \infty$), we will approach the correct answer, but it doesn't tell us how quickly.
  - **Asymptotic normality**: Letting $\mathrm{Var}[X_1] = \sigma^2$, by the central limit theorem,

$$\sqrt{n}(\hat{\mu}_n - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \tag{164}$$

    where $\xrightarrow{d}$ denotes convergence in distribution.[8] Asymptotic normality says that if n is large enough, then $\hat{\mu}_n - \mu$ behaves as $\frac{\sigma}{\sqrt{n}}$), where the variance is decreasing at a rate of $1/n$. But this result is only asymptotic, it doesn't tell us anything precise for a particular value of n (say, $n = 10$).

[7] Convergence in probability: For each $\epsilon > 0$, $\mathbb{P}[|\hat{\mu}_n - \mu| \ge \epsilon] \to 0$ as $n \to \infty$.

[8] Convergence in distribution: For each t, $\mathbb{P}[\frac{\sqrt{n}(\hat{\mu}_n - \mu)}{\sigma} \le t] \to \Phi(t)$ as $n \to \infty$, where Φ is the cumulative distribution of the standard Gaussian distribution.

- **Tail bounds**: Ideally, we want a statement of the following form:

$$\mathbb{P}[|\hat{\mu}_n - \mu| \geq \epsilon] \leq \mathrm{SomeFunction}(n, \epsilon) = \delta. \tag{165}$$

  Based on the Gaussian approximation, we expect that the bounding function on the RHS would decay double exponentially in ϵ and exponentially in n. We shall see shortly that this intuition is indeed true. In the context of learning theory, ϵ would be the bound on the difference between empirical and expected risks, and $1 - \delta$ would be the confidence.

- Note: of course, as we sweep ϵ from 0 to ∞ and look at how much probability mass is past ϵ, we get a complete picture of the full distribution. However, typically tail bounds are simple upper bounds which are often loose and only become more reliable for small ϵ.

- Our starting point is Markov's inequality, a very simple tool that allows us to control the deviation of a non-negative random variable from its mean using the expectation of that random variable. In short, it turns expectations (which are easier to work with) into tail probabilities (what we want).

- **Theorem 5 (Markov's inequality)**
  - Let $Z \geq 0$ be a random variable.
  - Then

$$\mathbb{P}[Z \geq t] \leq \frac{\mathbb{E}[Z]}{t}. \tag{166}$$

- Proof:
  - Since Z is non-negative, we have $t\mathbb{I}[Z \geq t] \leq Z$.
  - Taking expectations on both sides and rearranging completes the proof.

- Remarks
  - We can apply $Z = (X - \mu)^2$ (second moment) and $t = \epsilon^2$ to obtain **Chebyshev's inequality**:

$$\mathbb{P}[|X - \mu| \geq \epsilon] \leq \frac{\mathrm{Var}[X]}{\epsilon^2}. \tag{167}$$

    Applying the inequality to the average over i.i.d. variables ($\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n X_i$), then $\mathrm{Var}[\hat{\mu}_n] = \frac{\mathrm{Var}[X_1]}{n}$. This is a very weak result, because the tail probability is decaying only at at polynomial rate $(1/n)$.

  - To get stronger bounds, we need to apply Markov's inequality on higher order moments. In particular, we will look at all moments by considering $Z = e^{tX}$, where t is a free parameter we will use later to optimize the bound. "All the moments" is captured by the moment generating function:

- **Definition 6 (moment generating function)**
  - For a random variable X, the moment generating function (MGF) of X is:

$$\boxed{M_X(t) \overset{\text{def}}{=} \mathbb{E}[e^{tX}].} \tag{168}$$

- One useful way to think about the MGF is in terms of its Taylor expansion:

$$M_X(t) = 1 + t\,\mathbb{E}[X] + \frac{t^2}{2}\mathbb{E}[X^2] + \frac{t^3}{6}\mathbb{E}[X^3] + \cdots . \tag{169}$$

- The moment generating function receives its name because the k-th derivative evaluated at $t = 0$ yield the k-th moment (assuming we can swap integration and differentiation):

$$\frac{d^k M_X(t)}{dt^k}\Big|_{t=0} = \mathbb{E}[X^k]. \tag{170}$$

- One important property is that the MGF of a sum of independent random variables is simply the product of the MGFs. If X_1 and X_2 are independent random variables, then we have:

$$M_{X_1+X_2}(t) = M_{X_1}(t)M_{X_2}(t). \tag{171}$$

  The distribution over $X_1 + X_2$ can be computed using a convolution, which is typically cumbersome, but MGFs (like Fourier transforms) turn convolutions into products.

- Applying Markov's inequality to $Z = e^{tX}$, we get that

$$\mathbb{P}[X \geq \epsilon] \leq \frac{M_X(t)}{e^{t\epsilon}} \text{ for all } t > 0. \tag{172}$$

  We can apply this to the case of sample means ($X = \hat{\mu}_n$) by computing $\mathbb{P}[\hat{\mu}_n \geq \epsilon] = \mathbb{P}[X_1 + \cdots + X_n \geq n\epsilon]$. We get that all $t > 0$:

$$\mathbb{P}[\hat{\mu}_n \geq \epsilon] \leq \left(\frac{M_{X_1}(t)}{e^{t\epsilon}}\right)^n. \tag{173}$$

  - Provided that $\frac{M_{X_1}(t)}{e^{t\epsilon}} < 1$ for some t, getting n independent samples means that our tail probability will decrease exponentially. This is key.
  - Why should we expect this to happen? Assume If $\mathbb{E}[X_1] = 0$. The dominant term in the Taylor expansion of $M_{X_1}(t)$, the numerator of (173), is $1 + O(t^2)$. The dominant term in the Taylor expansion of $e^{t\epsilon}$, the denominator of (173), is $1 + O(t)$. We can always choose t such that the ratio is strictly less than 1 since $t^2 \ll t$ for small enough t.

  - Note that $M_{X_1}(t)$ could be infinite for some t, in which case the bounds are vacuous. In this class, we will work with distributions X such that $M_{X_1}(t) < \infty$ for all $t > 0$.

- Now let's actually compute the MGF of some probability distributions of interest. The Gaussian distribution is a natural place to start, as we will see.

- **Example 5 (MGF of Gaussian variables)**

  - Let $X \sim \mathcal{N}(0, \sigma^2)$.
  - Then $\boxed{M_X(t) = e^{\sigma^2 t^2/2}}$.
  - Derivation (by completing the square):

$$M_X(t) = \mathbb{E}[e^{tX}] = \int (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(\frac{x^2 - 2\sigma^2 tx}{-2\sigma^2}\right) dx \tag{174}$$

$$= \int (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(\frac{(x-\sigma^2 t)^2 - \sigma^4 t^2}{-2\sigma^2}\right) dx \tag{175}$$

$$= \exp\left(\frac{\sigma^2 t^2}{2}\right). \tag{176}$$

- **Lemma 3 (Tail bound for Gaussian variables)**

  - Having control on the Gaussian MGF, we can now derive a tail bound by plugging the form of the MGF into (172). This yields:

$$\mathbb{P}[X \geq \epsilon] \leq \inf_t \exp\left(\frac{\sigma^2 t^2}{2} - t\epsilon\right). \tag{177}$$

  The infimum on the RHS is attained by setting $t = \epsilon/\sigma^2$, yielding:

$$\boxed{\mathbb{P}[X \geq \epsilon] \leq \exp\left(\frac{-\epsilon^2}{2\sigma^2}\right).} \tag{178}$$

- What about non-Gaussian variables? Note that the bounds would still hold if we replaced $M_X(t)$ with an upper bound. This motivates the following definition:

- **Definition 7 (sub-Gaussian)**

  - A mean-zero random variable X is **sub-Gaussian** with parameter σ^2 if its moment generating function is bounded as follows:

$$\boxed{M_X(t) \leq \exp\left(\frac{\sigma^2 t^2}{2}\right).} \tag{179}$$

- It follows immediately by analogy with (178) that:

$$\boxed{\mathbb{P}[X \geq \epsilon] \leq \exp\left(\frac{-\epsilon^2}{2\sigma^2}\right).} \tag{180}$$

- Examples
  - Gaussian random variables: If $X \sim \mathcal{N}(0, \sigma^2)$, then X is sub-Gaussian with parameter σ^2 (trivial). Note that the sub-Gaussian parameter and the variance coincide in this case, but this is not true in general.
  - Bounded random variables (**Hoeffding's lemma**): If $a \leq X \leq b$ with probability 1 and $\mathbb{E}[X] = 0$, then X is sub-Gaussian with parameter $(b-a)^2/4$.
    - * Intuition (not a proof): in the Gaussian case, the sub-Gaussian parameter was the variance. Suppose $a = -b$. Then the variance of X is $b^2 = (b-a)^2/4$, the sub-Gaussian parameter given by the lemma. In general, the variance is at most the sub-Gaussian parameter.
  - Non-examples: exponential and Gamma variables, since these distributions have tails which are too fat, decaying exponentially rather than double exponentially.[9]

- Properties
  - Sum: If X_1 and X_2 are independent sub-Gaussian variables with parameters σ_1^2 and σ_2^2, respectively, then $X_1 + X_2$ is sub-Gaussian with parameter $\sigma_1^2 + \sigma_2^2$.
  - Multiplication by a constant: if X is sub-Gaussian with parameter σ^2, then for any $c > 0$, cX is sub-Gaussian with parameter $c^2\sigma^2$.
  - Not surprisingly, these properties coincide with those of Gaussians.
  - Note that the product of two sub-Gaussian variables is in general not sub-Gaussian.

- Given the machinery thus far, we can easily obtain the following classic tail bound for bounded random variables:

- **Theorem 6 (Hoeffding's inequality)**
  - Let $X_1, \dots, X_n$ be independent random variables.
  - Assume each X_i is bounded: $a_i \leq X_i \leq b_i$.
  - Let $\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n X_i$ be the sample mean.
  - Then

$$\mathbb{P}[\hat{\mu}_n \geq \mathbb{E}[\hat{\mu}_n] + \epsilon] \leq \exp\left(\frac{-2n^2\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right). \tag{181}$$

[9] These variables have moment generating functions which are defined not for all t but only small t. These are known as sub-exponential variables. This is consistent with the central limit theorem, which states that eventually (large enough n, small enough t) things behave Gaussian.

  - Special case ($a_i = -B, b_i = +B$):

$$\mathbb{P}[\hat{\mu}_n \geq \mathbb{E}[\hat{\mu}_n] + \epsilon] \leq \exp\left(\frac{-n\epsilon^2}{2B^2}\right). \tag{182}$$

- Proof of Theorem 6:
  - Using compositional properties of sub-Gaussian, $\hat{\mu}_n - \mathbb{E}[\hat{\mu}_n]$ is sub-Gaussian with parameter $\frac{1}{n^2}\sum_{i=1}^n \frac{(b_i-a_i)^2}{4}$.
  - Apply (180).
- Summary so far
  - We've shown that Gaussian and bounded random variables are sub-Gaussian and enjoy sharp tail bounds.
  - Furthermore, if we have an average of n independent sub-Gaussian variables, then the bound simply gets powered up by n.

[begin lecture 6] (6)

3.6 Finite hypothesis classes (Lecture 6)

- Having a good understanding of basic concentration inequalities, let's put them to use by analyzing the excess risk of the ERM for a finite hypothesis class, where now we do not assume realizability; that is, it is possible that every hypothesis $h \in \mathcal{H}$ has non-zero loss. The analysis will proceed via the two-step concentration + union bound format.

- **Theorem 7 (finite hypothesis class)**

  - Let $\mathcal{H}$ be a hypothesis class, where each hypothesis $h \in \mathcal{H}$ maps $\mathcal{X}$ to $\mathcal{Y}$.
  - Let ℓ be the zero-one loss: $\ell((x, y), h) = \mathbb{I}[y \neq h(x)]$. bounded
  - Assume $\mathcal{H}$ is finite (Assumption 1 holds).
  - Let $\hat{h}$ be the empirical risk minimizer.
  - Then with probability at least $1 - \delta$, the excess risk is bounded as follows:

$$L(\hat{h}) - L(h^*) \leq \sqrt{\frac{2(\log|\mathcal{H}| + \log(2/\delta))}{n}} = O\left(\sqrt{\frac{\log|\mathcal{H}|}{n}}\right). \tag{183}$$

- Proof of Theorem 7:

  - Recall by (158), the excess risk $L(\hat{h}) - L(h^*)$ is upper bounded using uniform convergence where we control $\sup_{h\in\mathcal{H}} |L(h) - \hat{L}(h)|$.
  - We adopt the same high-level strategy as for the realizable case:
    - Show convergence of a fixed hypothesis $h \in \mathcal{H}$ (using Hoeffding's inequality),
    - Show uniform convergence using a union bound.
  - Step 1 (convergence)
    - For a fixed $h \in \mathcal{H}$, note that $\hat{L}(h)$ is an empirical average over n i.i.d. loss terms (bounded in $[0, 1]$) with expectation $L(h)$. Therefore, by Hoeffding's inequality,

      realizable case: exp(-2n \varepsilon)

$$\mathbb{P}[\hat{L}(h) - L(h) \geq \epsilon] \leq \exp\left(-2n\epsilon^2\right). \tag{184}$$

    - To bound the absolute value, we apply the bound again on the negative loss and combine using the union bound:

$$\mathbb{P}[|\hat{L}(h) - L(h)| \geq \epsilon] \leq 2\exp\left(-2n\epsilon^2\right). \tag{185}$$

  - Step 2 (uniform convergence)

- * Since $\mathcal{H}$ is finite, we can apply the union bound over $|\mathcal{H}|$ hypotheses, obtaining:

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |\hat{L}(h) - L(h)| \geq \frac{\epsilon}{2}\right] \leq |\mathcal{H}| \cdot 2\exp\left(-2n\left(\frac{\epsilon}{2}\right)^2\right) \overset{\text{def}}{=} \delta. \tag{186}$$

  Note that we substituted $\frac{\epsilon}{2}$ for ϵ, which is needed by the generalization bound (158).

- * Rearranging completes the proof.

- Comparison with the realizable case (144):
  - We still have logarthmic dependence on $|\mathcal{H}|$ and $1/\delta$.
  - The main difference is that the realizable bound had a $\frac{1}{n}$ rate, whereas when there is noise, we now get a $\frac{1}{\sqrt{n}}$ rate, which is due to the use of Hoeffding's inequality. This gap is real and is due to the fact that learning is much easier / faster when there is no noise. really? what if empirical loss has Gaussian tail. Is Hoeffding sharp?
  - Note that when we performed asymptotic analysis, we also got a $\frac{1}{n}$ rate and did not assume realizability. However, there were two differences there: we assumed the loss was twice differentiable (in contrast to the zero-one loss here), and we were analyzing the risk near θ^* as opposed to globally across $\mathcal{H}$.

3.7 Concentration inequalities (continued) (Lecture 6)

- Now that we have relaxed the realizability assumption, let us now try to relax the finiteness one too. To do this, we will develop the theory of Rademacher complexity, but first, we will take a detour to McDiarmid's inequality, which will help us convert the tail bound (RHS of (158)) into an expectation.

- McDiarmid's inequality is a generalization of Hoeffding's inequality, where we want to bound not the average of random variables $X_1, \dots, X_n$, but any function on $X_1, \dots, X_n$ satisfying an appropriate bounded differences condition.

- **Theorem 8 (McDiarmid's inequality (bounded differences inequality))**
  - Let f be a function satisfying the following bounded differences condition:

$$|f(x_1, \dots, x_i, \dots, x_n) - f(x_1, \dots, x_i', \dots, x_n)| \leq c_i \tag{187}$$

    for all $i = 1, \dots, n$ and all $x_1, \dots, x_n, x_i'$. Intuition: modifying one coordinate doesn't change the function value too much.
  - Let $X_1, \dots, X_n$ be independent random variables.

  - Then

$$\mathbb{P}[f(X_1, \dots, X_n) - \mathbb{E}[f(X_1, \dots, X_n)] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right). \tag{188}$$

- Remarks
  - McDiarmid's inequality generalizes Hoeffding's inequality as follows. Define the function $f(x_1, \dots, x_n) = \frac{1}{n}\sum_{i=1}^n x_i$, where each x_i satisfies $a_i \leq x_i \leq b_i$. Then f satisfies the bounded differences condition with $c_i = \frac{1}{n}(b_i - a_i)$. Substituting this value of c_i recovers Hoeffding's inequality exactly.
  - This result is quite powerful, as it holds for any independent random variables, and f could be quite complex (e.g., fitting a neural network). As long as the function isn't too sensitive to perturbations in one of its arguments, we get good concentration.
  - The proof is not difficult, but it requires introducing martingales, which generalize partial sums of independent random variables.

- **Definition 8 (martingale)**
  - A sequence of random variables $Z_0, Z_1, \dots, Z_n$ is a **martingale sequence** with respect to another sequence of random variables $X_1, \dots, X_n$ iff Z_i is a function of $X_{1:i}$, $\mathbb{E}[|Z_i|] < \infty$, and

$$\mathbb{E}[Z_i \mid X_{1:i-1}] = Z_{i-1}. \tag{189}$$

  - Define $D_i \overset{\text{def}}{=} Z_i - Z_{i-1}$. We call $D_{1:n}$ a **martingale difference sequence** with respect to $X_{1:n}$. Another way to write (189) is

$$\mathbb{E}[D_i \mid X_{1:i-1}] = 0. \tag{190}$$

- Examples
  - Random walk: $Z_0 = 0$, $Z_i = Z_{i-1} + 1$ with probability $\frac{1}{2}$ and $Z_i = Z_{i-1} - 1$ with probability $\frac{1}{2}$. This is just a sum of i.i.d. random variables.
  - Random walk with absorbing state: same as above but $Z_i = Z_{i-1}$ if $Z_{i-1} = 42$.

- Intuitions
  - Given $X_{1:i-1}$ (the past), the expected value of Z_i is the same as Z_{i-1} (i.e., can't predict the future).
  - Think of $D_{1:n}$ as a generalization of an i.i.d. sequence.

- Recall that we showed that sums of independent sub-Gaussian variables are sub-Gaussian. The exact same result holds for martingales, as shown in the following lemma:
- **Lemma 4 (sub-Gaussian martingales)**
  - Let $Z_0, Z_1, \dots, Z_n$ be a martingale with respect to $X_1, \dots, X_n$.
  - Suppose that each difference $D_i = Z_i - Z_{i-1}$ is *conditionally* sub-Gaussian with parameter σ_i^2, that is:

$$\mathbb{E}[e^{tD_i} \mid X_{1:i-1}] \le \exp(\sigma_i^2 t^2/2). \tag{191}$$

  - Then $Z_n - Z_0 = \sum_{i=1}^n D_i$ is sub-Gaussian with parameter $\sigma^2 \stackrel{\text{def}}{=} \sum_{i=1}^n \sigma_i^2$.
- Proof of Lemma 4
  - This proof is straightforward by induction on n:

$$\begin{aligned}
\mathbb{E}[\exp(t(Z_n - Z_0))] &= \mathbb{E}[\exp(tD_n)\exp(t(Z_{n-1} - Z_0))] && (192)\\
&= \mathbb{E}[\mathbb{E}[\exp(tD_n)\exp(t(Z_{n-1} - Z_0)) \mid X_{1:n-1}]] && (193)\\
&\le \exp(\sigma_n^2 t^2/2)\mathbb{E}[\exp(t(Z_{n-1} - Z_0))] && (194)\\
&\le \exp\left(\sum_{i=1}^n \sigma_i^2 t^2/2\right). && (195)
\end{aligned}$$

  - The key is that conditioned on $X_{1:i-1}$, D_i is just a sub-Gaussian variable and $Z_{i-1} - Z_0$ is just a constant. The last inequality follows by repeating the argument recursively.
- Proof of Theorem 8 (McDiarmid's inequality)
  - We construct a particular type of martingle called **Doob martingale**:

$$Z_i = \mathbb{E}[f(X_1, \dots, X_n) \mid X_{1:i}]. \tag{196}$$

  Note the extremes: $Z_0 = \mathbb{E}[f(X_1, \dots, X_n)]$ and $Z_n = f(X_1, \dots, X_n)$. We can think of this martingale as exposing more information about $f(X_1, \dots, X_n)$ over time. Using this notation, we are interested in bounding $\mathbb{P}[Z_n - Z_0 \ge \epsilon]$.
  - To show that we have a sub-Gaussian martingale, let's study $D_i = Z_i - Z_{i-1}$: Both Z_i and Z_{i-1} condition on $X_{1:i-1}$ and take expectations over $X_{i+1:n}$. The only difference is in the treatment of X_i:

$$\begin{aligned}
Z_{i-1} &= \mathbb{E}[f(\overbrace{X_1, \dots, X_{i-1},}^{\text{condition}} \overbrace{X_i, X_{i+1}, \dots, X_n}^{\text{marginalize}}) \mid X_{1:i-1}] && (197)\\
Z_i &= \mathbb{E}[f(\underbrace{X_1, \dots, X_{i-1}, X_i}_{\text{condition}}, \underbrace{X_{i+1}, \dots, X_n}_{\text{marginalize}}) \mid X_{1:i}] && (198)
\end{aligned}$$

Therefore we would expect that D_i is contained in some interval of length c_i. Of course, we need to handle the marginalization over $X_{i+1:n}$ properly.

- To make this precise, define the lower and upper bounds which measure how large Z_i could get:

$$L_i = \inf_x \mathbb{E}[f(X_{1:n}) \mid X_{1:i-1}, X_i = x] - \mathbb{E}[f(X_{1:n}) \mid X_{1:i-1}], \tag{199}$$

$$U_i = \sup_x \mathbb{E}[f(X_{1:n}) \mid X_{1:i-1}, X_i = x] - \mathbb{E}[f(X_{1:n}) \mid X_{1:i-1}]. \tag{200}$$

  Note that $L_i \le D_i \le U_i$.

- Let x_L and x_U correspond to the x's achieving L_i and U_i, respectively. By the bounded differences assumption,

$$f(X_{1:i-1}, x_U, X_{i+1}) - f(X_{1:i-1}, x_L, X_{i+1}) \le c_i. \tag{201}$$

- Now here's the key step: By independence of the X_i's, the distribution over $X_{i+1:n}$ is the same no matter on whether we condition on $X_i = x_L$ or $X_i = x_U$. Therefore, we can take an expectation over $X_{i+1:n}$ to get that

$$U_i - L_i = \mathbb{E}[f(X_{1:i-1}, x_U, X_{i+1}) \mid X_{1:i-1}, X_i = x_U] \tag{202}$$

$$- \mathbb{E}[f(X_{1:i-1}, x_L, X_{i+1}) \mid X_{1:i-1}, X_i = x_L] \le c_i. \tag{203}$$

  This means that $D_i \in [L_i, U_i]$ is sub-Gaussian with parameter $c_i^2/4$ conditioned on $X_{1:i-1}$.

- Applying Lemma 4, we have that $Z_n - Z_0$ is sub-Gaussian with parameter $\sum_{i=1}^n c_i^2/4$. Finally, apply the sub-Gaussian tail inequality (180).

3.8 Rademacher complexity (Lecture 6)

- So far, we've relied on concentration inequalities (Hoeffding's inequality) along with the union bound to derive generalization bounds via uniform convergence. However, we can't directly apply the union bound to infinite hypothesis classes (set of all linear classifiers). We need a more sophisticated way to measure the complexity of a hypothesis class. In this section, we will develop such a framework known as Rademacher complexity, which has many nice properties and connects to other measures of complexity such as VC dimension and covering numbers.

- We will arrive at Rademacher complexity by deriving generalization bounds. Specifically, we carry out the following steps:

  - Step 1: Define a random variable G_n (maximum difference between expected and empirical risk).
  - Step 2: Show that it concentrates to $\mathbb{E}[G_n]$ using McDiarmid's inequality.

  - Step 3: Use a technique called **symmetrization** to bound the expectation using a quantity known as the Rademacher complexity.

- Step 1 (setup)

  - Consider the largest difference between the expected and the empirical risk over all possible hypotheses:

$$\boxed{G_n \overset{\text{def}}{=} \sup_{h \in \mathcal{H}} L(h) - \hat{L}(h).} \tag{204}$$

    Here, G_n is a random variable that depends on the data points $Z_1, \dots, Z_n$. An upper bound on G_n would ensure that if you observed empirical risk $\hat{L}(\hat{h})$, the expected risk $L(\hat{h})$ will not be much higher.

  - In order to bound the excess risk (158), we actually need to also bound G'_n defined analogously for the negative loss $\ell'(z, h) = -\ell(z, h)$, so that

$$\mathbb{P}[L(\hat{h}) - L(h^*) \geq \epsilon] \leq \mathbb{P}\left[\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h)| \geq \frac{\epsilon}{2}\right] \tag{205}$$

$$\leq \mathbb{P}\left[G_n \geq \frac{\epsilon}{2}\right] + \mathbb{P}\left[G'_n \geq \frac{\epsilon}{2}\right]. \tag{206}$$

    Usually, the tail bound for G_n is the same as for G'_n, as we'll see later.

- Step 2 (concentration): convert tail bound into an expectation

  - Let g be the deterministic function such that $G_n = g(Z_1, \dots, Z_n)$.
  - Then g satisfies the following bounded differences condition:

$$|g(Z_1, \dots, Z_i, \dots, Z_n) - g(Z_1, \dots, Z'_i, \dots, Z_n)| \leq \frac{1}{n}. \tag{207}$$

  - Proof:
    * Recall $\hat{L}(h) = \frac{1}{n} \sum_{i=1}^n \ell(Z_i, h)$.
    * We have:

$$\left| \underbrace{\sup_{h \in \mathcal{H}} [L(h) - \hat{L}(h)]}_{g(Z_1, \dots, Z_i, \dots, Z_n)} - \underbrace{\sup_{h \in \mathcal{H}} [L(h) - \hat{L}(h) + \frac{1}{n}(\ell(Z_i, h) - \ell(Z'_i, h))]}_{g(Z_1, \dots, Z'_i, \dots, Z_n)} \right| \leq \frac{1}{n}. \tag{208}$$

    * For each $h \in \mathcal{H}$, the difference between the G_n and the perturbed G_n is at most $\frac{1}{n}$ since the loss is bounded: $\ell(z, h) \in [0, 1]$. Taking the supremum (which is a contraction) cannot increase the difference.

  - Now we can apply McDiarmid's inequality (Theorem 8) to get that:

$$\boxed{\mathbb{P}\left[G_n \geq \mathbb{E}[G_n] + \epsilon\right] \leq \exp(-2n\epsilon^2).} \tag{209}$$

- Remark: Note that g is quite a non-trivial function (involving a sup over a huge hypothesis class), but because it satisfies the bounded differences condition, we can simply apply McDiarmid's inequality.

- Step 3 (symmetrization)
  - Now we need to bound $\mathbb{E}[G_n]$. This quantity is quite difficult since it depends on $L(h)$, an expectation over the unknown distribution p^*. The goal of symmetrization is to remove this strong dependence on p^* and replace it with a quantity that *only depends on* p^* *through data points* $Z_1, \dots, Z_n$.
  - The key idea of symmetrization is to introduce a ghost dataset $Z'_1, \dots, Z'_n$, drawn i.i.d. from p^*. Let $\hat{L}'(h) = \frac{1}{n}\sum_{i=1}^n \ell(Z'_i, h)$ be the empirical risk with respect to this ghost dataset.
  - Rewriting $L(h)$ in terms of the ghost dataset:

$$\mathbb{E}[G_n] = \mathbb{E}[\sup_{h \in \mathcal{H}} \mathbb{E}[\hat{L}'(h)] - \hat{L}(h)]. \tag{210}$$

  - Bring $\hat{L}(h)$ into the inner expectation by conditioning on the original dataset $Z_{1:n}$ (the two datasets are independent):

$$\mathbb{E}[G_n] = \mathbb{E}[\sup_{h \in \mathcal{H}} \mathbb{E}[\hat{L}'(h) - \hat{L}(h) \mid Z_{1:n}]]. \tag{211}$$

  - Pushing the sup inside the expectation can only increase:

$$\mathbb{E}[G_n] \le \mathbb{E}[\mathbb{E}[\sup_{h \in \mathcal{H}} \hat{L}'(h) - \hat{L}(h) \mid Z_{1:n}]]. \tag{212}$$

  - Apply law of iterated conditional expectation:

$$\mathbb{E}[G_n] \le \mathbb{E}[\sup_{h \in \mathcal{H}} \hat{L}'(h) - \hat{L}(h)]. \tag{213}$$

  - Expanding, we see that we have successfully gotten an expression that depends on p^* through $Z_{1:n}$, but also the ghost dataset $Z'_{1:n}$.

$$\mathbb{E}[G_n] \le \mathbb{E}\left[\sup_{h \in \mathcal{H}} \frac{1}{n}\sum_{i=1}^n [\ell(Z'_i, h) - \ell(Z_i, h)]\right]. \tag{214}$$

  - Let's try to remove the dependence on the ghost dataset $Z'_{1:n}$ now. To that end, introduce i.i.d. **Rademacher variables** $\sigma_1, \dots, \sigma_n$ independent of $Z_{1:n}, Z'_{1:n}$, where σ_i is uniform over $\{-1, +1\}$. Since $\ell(Z'_i, h) - \ell(Z_i, h)$ is symmetric around 0, multiplying by σ_i doesn't change its distribution. Now expanding the definition of the empirical risk: for any fixed h

$$\mathbb{E}[G_n] \le \mathbb{E}\left[\sup_{h \in \mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i[\ell(Z'_i, h) - \ell(Z_i, h)]\right]. \tag{215}$$

    In fact, this holds for every $\sigma_{1:n}$ and doesn't depend on its distribution.

- Pushing the sup inside $\sup_h[a_h - b_h] \le \sup_h[a_h] + \sup_h[-b_h]$:

$$\mathbb{E}[G_n] \le \mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i \ell(Z_i', h) + \sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n (-\sigma_i)\ell(Z_i, h)\right]. \tag{216}$$

- By linearity of expectation, the fact that Z_i' has the same distribution as Z_i, and the fact that σ_i and $-\sigma_i$ have the same distribution, we get:

large rademacher —> more flexible of l over the sample

$$\boxed{\mathbb{E}[G_n] \le 2\mathbb{E}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i \ell(Z_i, h)\right]}. \tag{217}$$

The RHS motivates the following general definition of the Rademacher complexity:

- **Definition 9 (Rademacher complexity)**

  loss, as a function of data and learning algorithm

  - Let $\mathcal{F}$ be a class of real-valued functions $f : \mathcal{Z} \to \mathbb{R}$. Example: $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ consists of input-output pairs.
  - Define the **Rademacher complexity** (or Rademacher average) of $\mathcal{F}$ to be

$$\boxed{R_n(\mathcal{F}) \overset{\text{def}}{=} \mathbb{E}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n \sigma_i f(Z_i)\right],} \tag{218}$$

  where

    - $Z_1, \dots, Z_n$ are drawn i.i.d. from p^* (data points); and
    - $\sigma_1, \dots, \sigma_n$ are drawn i.i.d. from the uniform distribution over $\{-1, +1\}$ (Rademacher variables).

  - Interpretation of Rademacher complexity:
    - Consider a binary classification problem with inputs $Z_1, \dots, Z_n$ and *random labels* $\sigma_1, \dots, \sigma_n$. Clearly, this is a meaningless learning problem.
    - The Rademacher complexity captures (in expectation) how well the best function from the function class $\mathcal{F}$ can align with these random labels. In other words, how well $\mathcal{F}$ can fit noise? A large $\mathcal{F}$ will be able to fit noise better and thus have a larger Rademacher complexity.
    - As we'll see later, we'd like $R_n(\mathcal{F})$ to go to zero as n increases.
  - Define the **empirical Rademacher complexity** of $\mathcal{F}$ to be:

$$\boxed{\hat{R}_n(\mathcal{F}) \overset{\text{def}}{=} \mathbb{E}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n \sigma_i f(Z_i) \mid Z_{1:n}\right],} \tag{219}$$

  which is a random variable depending on the data. Note that $R_n(\mathcal{F}) = \mathbb{E}[\hat{R}_n(\mathcal{F})]$, where the expectation is taken over the n training examples.

- **Theorem 9 (generalization bound based on Rademacher complexity)**
  - Define $\mathcal{A} = \{z \mapsto \ell(z,h) : h \in \mathcal{H}\}$ to be the **loss class**, the composition of the loss function with each of the hypotheses. With probability at least $1-\delta$,

$$\boxed{L(\hat{h}) - L(h^*) \le 4R_n(\mathcal{A}) + \sqrt{\frac{2\log(2/\delta)}{n}}.} \tag{220}$$

- Proof of Theorem 9:
  - Note that $\mathbb{E}[G_n] \le 2R_n(\mathcal{A})$ by definition of Rademacher complexity.
  - Since negation doesn't change the Rademacher complexity, $R_n(\mathcal{A}) = R_n(-\mathcal{A})$, so $\mathbb{E}[G_n'] \le 2R_n(-\mathcal{A})$.
  - Let's apply the tail bound (209) on both G_n and G_n':

$$\begin{aligned} \mathbb{P}\left[G_n \ge \frac{\epsilon}{2}\right] &\le \exp\left(-2n\left(\frac{\epsilon}{2} - \mathbb{E}[G_n]\right)^2\right) && (221) \\ &\le \exp\left(-2n\left(\frac{\epsilon}{2} - 2R_n(\mathcal{A})\right)^2\right) \quad [\text{for } \epsilon \ge 4R_n(\mathcal{A})] && (222) \\ &\overset{\text{def}}{=} \frac{\delta}{2}. && (223) \end{aligned}$$

    We have an analogous inequality for G_n':

$$\mathbb{P}\left[G_n' \ge \frac{\epsilon}{2}\right] \le \frac{\delta}{2}. \tag{224}$$

    Adding them and solving for ϵ yields the result.
  - Remark: We see that the essence of generalization is captured in the Rademacher complexity of the loss class $R_n(\mathcal{A})$.

- Summary
  - We have reduced the problem of bounding G_n, which involved a complex comparison between $L(h)$ and $\hat{L}(h)$, to something that only depends on samples. In fact, the empirical Rademacher complexity can even be computed from data. We'll see that in the analysis to follow, we will often condition on $Z_{1:n}$; the points take a backseat as it is the randomness of σ_i and the supremum over $\mathcal{F}$ that really determine the Rademacher complexity.
  - We will study the Rademacher complexity $R_n(\mathcal{F})$ of various function classes $\mathcal{F}$. First, let us discuss some basic compositional properties that Rademacher complexity enjoys, mostly due to linearity of expectation:

- **Basic properties of Rademacher complexity**

- Boundedness
  - $R_n(\mathcal{F}) \le \max_{f \in \mathcal{F}} \max_z f(z)$.
  - This is a trivial bound. It's not very impressive to show that the Rademacher complexity is a constant; rather, we'd ideally like it to go to zero as $n \to \infty$.
- Singleton
  - $R_n(\{f\}) = 0$
  - Proof: σ_i has zero mean and is independent of everything else, so $\mathbb{E}[\sigma_i f(Z_i)] = 0$
- Monotonicity
  - $R_n(\mathcal{F}_1) \le R_n(\mathcal{F}_2)$ if $\mathcal{F}_1 \subseteq \mathcal{F}_2$
  - Proof: $\sup_{f \in \mathcal{F}_2}$ ranges over at least as many functions as $\sup_{f \in \mathcal{F}_1}$, which makes it at least as large.
- Linear combination
  - $R_n(\mathcal{F}_1 + \mathcal{F}_2) = R_n(\mathcal{F}_1) + R_n(\mathcal{F}_2)$ for $\mathcal{F}_1 + \mathcal{F}_2 = \{f_1 + f_2 : f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2\}$
  - Proof: linearity of expectation
- Scaling
  - $R_n(c\mathcal{F}) = |c| R_n(\mathcal{F})$
  - Proof: for $c > 0$, this is trivial; if $c < 0$, then we can absorbe the negation into σ_i's, which are symmetric around zero.
- Lipschitz composition (a generalization of scaling):
  - $R_n(\phi \circ \mathcal{F}) \le c_\phi R_n(\mathcal{F})$, where $\phi \circ \mathcal{F} = \{z \mapsto \phi(f(z)) : f \in \mathcal{F}\}$ and c_ϕ is the Lipschitz constant of ϕ: $|\phi(z) - \phi(z')| \le c_\phi \|z - z'\|_2$.
  - Proof: see Corollary 3.17 of Ledoux and Talagrand 1991
  - If ϕ is differentiable, then c_ϕ is just a bound on the L_2 norm of the gradient.
  - This property is useful because, as we will see later, we can analyze the complexity of our hypothesis class (e.g., linear functions), and compose with ϕ to get the loss class (e.g., 0-1 loss of linear functions).
- Convex hull
  - $R_n(\text{convex-hull}(\mathcal{F})) = R_n(\mathcal{F})$ for finite $\mathcal{F}$
  - Proof: supremum over the convex hull is attained at one of its vertices
  - This property is useful because if we want to compute the Rademacher of a polytope (an infinite set), it suffices to compute the Rademacher complexity of its vertices (a finite set). We will use this property when we look at L_1 regularization.

[begin lecture 7] (7)

3.9 Finite hypothesis classes (Lecture 7)

- So far, we've set up Rademacher complexity as a means of bounding the complexity of a function class (specifically, the loss class associated with a hypothesis class). Now, let's instantiate Rademacher complexity for the case where the function class is finite. This may initially seem like overkill (since we already analyzed the finite hypothesis case), but doing this analysis in the Rademacher complexity is a good sanity check, and it will reveal an unexpected but important aspect of finiteness, which we will exploit when we talk about shattering coefficients in the next section.

- As a motivating example of a finite hypothesis class, consider the set of functions that take conjunctions (products) of ~~k~~ s of d boolean features:

$$\mathcal{F} = \left\{ z \mapsto \prod_{j \in J} z_j : J \subseteq \{1, \dots, d\}, |J| = s \right\}. \tag{225}$$

  Question: what should the Rademacher complexity of $\mathcal{F}$ be? Well, $|\mathcal{F}| = \binom{d}{s}$, and the complexity should depend on $\log |\mathcal{F}| = O(s \log d)$, and also go down as $1/\sqrt{n}$.

- **Lemma 5 (Massart's finite lemma)**

  - Throughout this lemma, condition on $Z_1, \dots, Z_n$ (treat them as constants).
  - Let $\mathcal{F}$ be a finite set of functions.
  - Let M^2 be a bound on the second moment:

$$\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n f(Z_i)^2 \le M^2. \tag{226}$$

  - Then the empirical Rademacher complexity is upper bounded:

$$\boxed{\hat{R}_n(\mathcal{F}) \le \sqrt{\frac{2M^2 \log |\mathcal{F}|}{n}}.} \tag{227}$$

- Proof of Lemma 5:

  - The key idea of the proof is to leverage the moment generating function of an average of i.i.d. variables.
  - Let $W_f = \frac{1}{n} \sum_{i=1}^n \sigma_i f(Z_i)$.
  - We are interested in bounding $\hat{R}_n(\mathcal{F}) = \mathbb{E}[\sup_{f \in \mathcal{F}} W_f \mid Z_{1:n}]$.

- Exponentiate and use convexity of $x \mapsto \exp(tx)$ for $t \geq 0$ to push the exp inside the expectation, which turns it into the moment generating function of $\sup_{f \in \mathcal{F}} W_f$:

$$\exp(t\mathbb{E}[\sup_{f \in \mathcal{F}} W_f \mid Z_{1:n}]) \leq \mathbb{E}[\exp(t \sup_{f \in \mathcal{F}} W_f) \mid Z_{1:n}]. \tag{228}$$

- By monotonicity, we can pull the sup out of the exponent:

$$= \mathbb{E}[\sup_{f \in \mathcal{F}} \exp(tW_f) \mid Z_{1:n}]. \tag{229}$$

- The sup over non-negative terms can be upper bounded by the sum:

$$\leq \sum_{f \in \mathcal{F}} \mathbb{E}[\exp(tW_f) \mid Z_{1:n}]. \tag{230}$$

- Now we have to bound $\mathbb{E}[\exp(tW_f)]$, the moment generating function of a single W_f.
  * By Hoeffding's lemma (after Lemma 3.5), σ_i is a bounded random variable with sub-Gaussian parameter $2^2/4 = 1$.
  * By scaling and independence of σ_i, we get that W_f is sub-Gaussian with parameter $\frac{1}{n^2} \sum_{i=1}^n f(Z_i)^2 \leq \frac{M^2}{n}$ (remember that we're conditioning on $Z_{1:n}$, which are just treated like constants).
  * By the definition of sub-Gaussian, we have

  $$\mathbb{E}[exp(tW_f)] \leq \exp\left(\frac{t^2 M^2}{2n}\right). \tag{231}$$

  The important thing is that W_f is an average of i.i.d. variables, so its moment generating function decays exponentially fast.

- Plugging this in to the overall bound and the rest is algebra:

$$\exp(t\hat{R}_n(\mathcal{F})) \leq \sum_{f \in \mathcal{F}} \mathbb{E}[\exp(tW_f) \mid Z_{1:n}] \leq |\mathcal{F}| \exp\left(\frac{t^2 M^2}{2n}\right). \tag{232}$$

- Taking logs and dividing by t:

$$\hat{R}_n(\mathcal{F}) \leq \frac{\log |\mathcal{F}|}{t} + \frac{tM^2}{2n}. \tag{233}$$

- Optimizing over t yields the result as desired (by just taking the twice the geometric average of $\log |\mathcal{F}|$ and $M^2/(2n)$):

$$\hat{R}_n(\mathcal{F}) \leq \sqrt{\frac{2M^2 \log |\mathcal{F}|}{n}}. \tag{234}$$

- We can immediately apply Massart's finite lemma to any finite loss class $\mathcal{A}$. For example, for the zero-one loss, we have each $f(Z_i) = \ell(Z_i, h) \in [0, 1]$ (for $h \in \mathcal{H}$ corresponding to $f \in \mathcal{F}$) so we can take $M = 1$. Combined with (220), this gives us

$$L(\hat{h}) - L(h^*) \le \sqrt{\frac{32 \log |\mathcal{H}|}{n}} + \sqrt{\frac{2 \log(2/\delta)}{n}} = O\left(\sqrt{\frac{\log |\mathcal{H}|}{n}}\right), \tag{235}$$

which gives the same result as Theorem 7 (just with worse constants).

3.10 Shattering coefficient (Lecture 7)

- What happens if we want to bound a class of half-spaces passing through the origin:

$$\mathcal{F} = \{z \mapsto \mathbb{I}[w \cdot z \ge 0] : w \in \mathbb{R}^d\}. \tag{236}$$

What should the complexity of $\mathcal{F}$ be? Since $\mathcal{F}$ is infinite, we can't just apply Massart's finite lemma directly. Here's some really sloppy intuition though: consider a 32-bit precision representation of the weight vector w; then there are $(2^{32})^d$ possible representations, so we might expect the Rademacher complexity to depend on d.

- Before we give up on Massart's finite lemma, let's take a closer look. The lemma places a bound on the empirical Rademacher complexity $\hat{R}_n$, which only depends on the n data points. If we had an infinite function class $\mathcal{F}$ that acts the same on the n points as a finite function class $\mathcal{F}'$ (which is allowed to depend on $Z_{1:n}$, which are as constants), then the two function classes have the same empirical Rademacher complexity.

- For example, consider two points in 2D: $Z_1 = [3, 0], Z_2 = [-3, 0]$. Then

$$\mathcal{F}' = \{z \mapsto \mathbb{I}[z \ge 0], z \mapsto \mathbb{I}[-z \ge 0]\} \tag{237}$$

has the same behaviors as $\mathcal{F}$, namely $[1, 0]$ and $[0, 1]$.

  - FIGURE: [draw points with lines]

- More formally, if

$$\{[f(Z_1), \dots, f(Z_n)] : f \in \mathcal{F}\} = \{[f'(Z_1), \dots, f'(Z_n)] : f' \in \mathcal{F}'\}, \tag{238}$$

then $\hat{R}_n(\mathcal{F}) = \hat{R}_n(\mathcal{F}')$. Therefore, all that matters as far as empirical Rademacher complexity is concerned is **the behavior of a function class on n data points**. This is the key observation!

- This observation is useful when we are trying to analyze **boolean functions**, those that return either 0 or 1 (or any function class with a small finite range). An important example is the loss class $\mathcal{A} = \{z \mapsto \ell(z, h) : h \in \mathcal{H}\}$ where ℓ is the zero-one loss. In this case, the number of behaviors on n points is certainly finite (and perhaps even small).

- The behavior of $\mathcal{F}$ on n data points is captured by the shattering coefficient:
- **Definition 10 (shattering coefficient (growth function))**
  - Let $\mathcal{F}$ be a family of functions that map $\mathcal{Z}$ to a finite set (usually $\{0, 1\}$).
  - The shattering coefficient of $\mathcal{F}$ is the maximum number of behaviors over n points:

$$s(\mathcal{F}, n) \overset{\text{def}}{=} \max_{z_1,\dots,z_n \in \mathcal{Z}} |\{[f(z_1), \dots, f(z_n)] : f \in \mathcal{F}\}|. \tag{239}$$

- We can use Massart's finite lemma to upper bound the empirical Rademacher complexity by the shattering coefficient. If $\mathcal{F}$ contains boolean functions, we have the bound $M = 1$, so that:

$$\boxed{\hat{R}_n(\mathcal{F}) \le \sqrt{\frac{2 \log s(\mathcal{F}, n)}{n}}.} \tag{240}$$

  The significance is that we can apply Massart's finite lemma to **infinite function classes with finite shattering coefficient**. In order to have applied this bound, it is important to condition on the data (note that the distribution over the data has disappeared completely!) After applying Massart's finite lemma, we can take expectations over the data without changing the bound. We have turned an annoying expectation over p^* into a purely combinatorial notion of complexity.

- Intuition:
  - For boolean functions, if $s(\mathcal{F}, n) = 2^n$ (we obtain all possible labelings), then we say $\mathcal{F}$ shatters any n points $z_1, \dots, z_n$ that achieve the maximum of (239).
  - To get meaningful bounds, we want $s(\mathcal{F}, n)$ to grow sub-exponentially with n; otherwise the Rademacher complexity will not go to zero, and we will not obtain uniform convergence. This is expected since if $\mathcal{F}$ can really hit all labelings for all n, then we would be able to fit any labeling of the data, leading to massive overfitting.
  - Example: consider n points in 1D. The class $\mathcal{F} = \{z \mapsto \mathbb{I}[z \ge t] : t \in \mathbb{R}\}$ has shattering coefficient $s(\mathcal{F}, n) = n + 1$.
- Shattering coefficient of hypotheses class and loss class
  - The bounds we derive depend on the shattering coefficient $s(\mathcal{A}, n)$ of the loss class $\mathcal{A}$ (via (240) and (220)). However, it will be more intuitive to talk about the shattering coefficient hypothesis class $\mathcal{H}$. We now show that the two are the same for zero-one loss and binary classifiers.
  - Let $\mathcal{H}$ be a set of binary classifiers $h : \mathcal{X} \to \{0, 1\}$ and zero-one loss $\ell((x, y), h) = \mathbb{I}[y \ne h(x)]$.

- Note that the hypothesis class $\mathcal{H}$ contains functions on $\mathcal{X}$, and the loss class $\mathcal{A}$ contains functions on $\mathcal{X} \times \{0, 1\}$: $\mathcal{A} = \{(x, y) \mapsto \mathbb{I}[y \neq h(x)] : h \in \mathcal{H}\}$.
- The key point is that

$$\boxed{s(\mathcal{H}, n) = s(\mathcal{A}, n).} \tag{241}$$

- The reason is as follows: for any $(x_1, y_1), \dots, (x_n, y_n)$, there is a bijection between the behavior of the losses and the hypotheses:

$$[\ell((x_1, y_1), h), \dots, \ell((x_n, y_n), h)] \quad \Leftrightarrow \quad [h(x_1), \dots, h(x_n)], \tag{242}$$

  where the translation between the two sets of vectors is obtained by taking the XOR with $[y_1, \dots, y_n]$. Therefore, as we range over $h \in \mathcal{H}$, we obtain the same number of behaviors from ℓ as we do from h.

3.11 VC dimension (Lecture 7)

- In the previous section, we saw that the shattering coefficient can be used to upper bound the Rademacher complexity of a family of boolean functions (240), which in turn can be used to upper bound the excess risk (Theorem 9). Although the shattering coefficient nicely captures the behavior of an infinite $\mathcal{H}$, it is not necessarily the most convenient quantity to get a handle on. In this section, we will use a concept called VC dimension to gain more intuition about the shattering coefficient.
- **Definition 11 (VC dimension)**
  - The **VC dimension** of a family of functions $\mathcal{H}$ with boolean outputs is the maximum number of points that can be shattered by $\mathcal{H}$:

$$\boxed{\mathrm{VC}(\mathcal{H}) = \sup\{n : s(\mathcal{H}, n) = 2^n\}.} \tag{243}$$

  - Intuition: the VC dimension of $\mathcal{H}$ is the maximum number of points whose labels can be memorized perfectly by choosing some $h \in \mathcal{H}$.
- Example (intervals)
  - Let $\mathcal{H} = \{z \mapsto \mathbb{I}[z \in [a, b]] : a, b \in \mathbb{R}\}$.
  - $s(\mathcal{H}, 1) = 2 = 2^1$ (can shatter)
  - $s(\mathcal{H}, 2) = 4 = 2^2$ (can shatter)
  - $s(\mathcal{H}, 3) = 7 < 2^3$ (can't shatter, because can't isolate the middle point)
  - $s(\mathcal{H}, n) = \binom{n+1}{2} + 1$
  - Therefore, $\mathrm{VC}(\mathcal{H}) = 2$.

- Important note: to show that a class $\mathcal{H}$ has VC dimension d, one needs to
  - Derive an upper bound: show that *no* $d+1$ points can be shattered. This is done by showing that for *any* $d+1$ points, there is some labeling that cannot be achieved by $\mathcal{H}$.
  - Derive a lower bound: show that there exists d points can be shattered. This is done by showing that there *exists* d points such that *all* 2^d labelings can be achieved by $\mathcal{H}$.
- One can verify that that the VC dimension of rectangles in two dimensions is 4. Based on this, one might be tempted to conclude that the VC dimension of a hypothesis class with d parameters is d, but the following example shows that this is in general not the case (there are pathological examples).
- Example (infinite VC dimension with one parameter)
  - Let $\mathcal{H} = \{x \mapsto \mathbb{I}[\sin(\theta x) \ge 0] : \theta \in \mathbb{R}\}$.
  - We have that $\mathrm{VC}(\mathcal{H}) = \infty$.
- It turns out that it's not the number of parameters that matter, but the dimension of the function space:
- **Theorem 10 (finite-dimensional function class)**
  - Let $\mathcal{F}$ be a function class containing functions $f : \mathcal{X} \to \mathbb{R}$.
  - Recall that the dimension of $\mathcal{F}$ is the number of elements in a basis of $\mathcal{F}$. For example, if $\mathcal{F} = \{x \mapsto w \cdot x : w \in \mathbb{R}^d\}$, then $\dim(\mathcal{F}) = d$.
  - Let $\mathcal{H} = \{x \mapsto \mathbb{I}[f(x) \ge 0] : f \in \mathcal{F}\}$ (for example, $\mathcal{F}$ could be linear functions, but not necessarily).
  - Then we have

$$\boxed{\mathrm{VC}(\mathcal{H}) \le \dim(\mathcal{F}).} \tag{244}$$

  - Remark: this allows us to connect the linear algebraic properties of $\mathcal{F}$ (dimension) with the combinatorial properties of $\mathcal{H}$ (VC dimension).
- Proof of Theorem 10:
  - Take any n points $x_1, \dots, x_n$ with $n > \dim(\mathcal{F})$. We will show that these n points cannot be shattered. The key is to find some direction that $\mathcal{F}$ can't cover.
  - Consider the linear map $M(f) \overset{\text{def}}{=} [f(x_1), \dots, f(x_n)] \in \mathbb{R}^n$ which maps each function $f \in \mathcal{F}$ to the function values on the n points (function evaluation is linear).

  - The vector space $\{M(f) : f \in \mathcal{F}\} \in \mathbb{R}^n$ has dimension at most $\dim(\mathcal{F})$ (applying linear maps can't increase dimension).
  - Since $n > \dim(\mathcal{F})$, there is some non-zero vector $c \in \mathbb{R}^n$ such that $M(f) \cdot c = 0$ for all $f \in \mathcal{F}$.
  - Without loss of generality, some component of c is negative (otherwise, just take $-c$, which satisfies $M(f) \cdot c = 0$ too).
  - So we have for all $f \in \mathcal{F}$:

$$\sum_{i:c_i \ge 0} c_i f(x_i) + \sum_{i:c_i < 0} c_i f(x_i) = 0. \tag{245}$$

  - For the purposes of contradiction, suppose $\mathcal{H}$ shatters $\{x_1, \dots, x_n\}$.
    * Then we could find an $h = (x \mapsto \mathbb{I}[f(x) \ge 0]) \in \mathcal{H}$ (equivalently some $f \in \mathcal{F}$) such that
      · $h(x_i) = 1$ ($f(x_i) \ge 0$) whenever $c_i \ge 0$ (so the first term of (245) is non-negative)
      · $h(x_i) = 0$ ($f(x_i) < 0$) whenever $c_i < 0$ (so the second term of (245) is strictly positive).
    * The sum of the two terms couldn't equal zero, which contradicts (245).
  - Therefore, $\mathcal{H}$ can't shatter $\{x_1, \dots, x_n\}$ for any choice of $x_1, \dots, x_n$, so $\mathrm{VC}(\mathcal{H}) \le \dim(\mathcal{F})$.

Here is the classic application of Theorem 10:

- **Example 6 (Half-spaces passing through the origin)**
  - Let $\mathcal{H} = \{x \mapsto \mathbb{I}[w \cdot x \ge 0] : w \in \mathbb{R}^d\}$ be the set of half-spaces in d dimensions.
  - By Theorem 10, the VC dimension of $\mathcal{H}$ is at most d ($\mathcal{H}$ is backed by a linear function class of dimension d).
  - Upper bounds suffice for obtaining generalization bounds, but just for fun, let's get a lower bound on the VC dimension.
  - Showing that the VC dimension of some $\mathcal{H}$ is at least d is easier because we just have to construct some set of d points that can be shattered rather than showing no $d + 1$ points can be shattered.
  - Create d points which are the basis vectors:

$$x_1 = [1, 0, \dots, 0, 0] \tag{246}$$
$$\cdots \tag{247}$$
$$x_d = [0, 0, \dots, 0, 1] \tag{248}$$

- Given any subset $I \subseteq \{1, \dots, d\}$ (which defines a labeling of $x_1, \dots, x_d$), we can construct a w as follows to obtain that labeling:
  - Set $w_i = 1$ for $i \in I$
  - Set $w_i = -1$ for $i \notin I$
- This establishes that $\mathrm{VC}(\mathcal{H}) \ge d$.
- Putting the upper and lower bounds, we get that $\boxed{\mathrm{VC}(\mathcal{H}) = d}$.

- We have defined VC dimension in terms of shattering coefficient (244), and have given some examples of VC dimension. Let us now upper bound the shattering coefficient in terms of the VC dimension.

- **Lemma 6 (Sauer's lemma)**
  - For a class $\mathcal{H}$ be a class with VC dimension d.
  - Then

$$\boxed{s(\mathcal{H}, n) \le \sum_{i=0}^{d} \binom{n}{i} \le \begin{cases} 2^n & \text{if } n \le d \\ \left(\frac{en}{d}\right)^d & \text{if } n > d. \end{cases}} \tag{249}$$

- Intuition
  - For $n \le d$, the shattering coefficient grows exponentially in n.
  - For $n > d$, the shattering coefficient grows only polynomially in n.
  - In some sense, $\mathcal{H}$ can only represent any subset of up to d of the n points.

- The ramification of Sauer's lemma is that now we have a new upper bound on the Rademacher complexity (and thus on uniform convergence):

$$\hat{R}_n(\mathcal{A}) \le \sqrt{\frac{2 \log s(\mathcal{A}, n)}{n}} = \sqrt{\frac{2 \log s(\mathcal{H}, n)}{n}} \le \boxed{\sqrt{\frac{2\mathrm{VC}(\mathcal{H})(\log n + 1)}{n}}}, \tag{250}$$

  where the first inequality follows from (240), the second equality follows from (241), and the final inequality follows from (249) and holds for $n \ge d$.

- Proof of Lemma 6 (somewhat technical and specific, skipped in class)
  - The idea is to take $\mathcal{H}$ and transform it into $\mathcal{H}'$ with the same shattering coefficient ($s(\mathcal{H}, n) = s(\mathcal{H}', n)$) but where $s(\mathcal{H}', n)$ will be easier to bound.
  - Key: all that matters is the action of $\mathcal{H}$ and $\mathcal{H}'$ on a finite set of n points, which we will represent as a table.

- Draw a table whose columns are the n points and rows are the $s(\mathcal{H}, n)$ possible labelings, and each entry is either a 0 or 1. The question is how many rows there are.
- Here's an example table T:

x_1	x_2	x_3	x_4
0	1	0	1
0	0	0	1
1	1	1	0
1	0	1	0

- We will transform the table to a canonical form as follows:
  - Pick a column j.
  - For each row r with $r_j = 1$, set $r_j = 0$ if the resulting r doesn't exist in the table.
  - Repeat until no more changes are possible.
- Here is the resulting table T' (corresponding to some $\mathcal{H}'$):

x_1	x_2	x_3	x_4
0	1	0	1
0	0	0	1
0	1	0	0
0	0	0	0

- Step 1: Note that the number of rows is still the same and all the rows are all distinct, so $s(\mathcal{H}, n) = s(\mathcal{H}', n)$. So we just have to compute $s(\mathcal{H}', n)$, which should be easier.
- Step 2: We show that the VC dimension doesn't increase by transformation ($\text{VC}(\mathcal{H}') \le \text{VC}(\mathcal{H})$)
  - The transformations proceed one column at a time:

$$T \to T_1 \to \cdots T_k \overset{\text{transform column } j}{\to} T_{k+1} \to \cdot \to T'. \tag{251}$$

  - Claim: After transforming any column j, if some subset $S \subseteq \{1, \dots, n\}$ of points is shattered (all $2^{|S|}$ labelings exist on those columns) after transformation (in T_{k+1}), then S was also shattered before transformation (in T_k).
  - Case 1: trivially true for all subsets S that don't contain j.
  - Case 2: take any subset S that contains j.
    - For any row i with 1 in column j, there is a row i' with 0 in column j and agrees with r on all columns except j: $T_{k+1}(i, j') = T_{k+1}(i', j')$ for all $j' \in \{1, \dots, n\} \backslash \{j\}$, but $T_{k+1}(i, j) = 1$ and $T_{k+1}(i', j) = 0$.
    - Note that $T_k(i, j) = 1$ (because we never turn zeros into ones).
    - Note that $T_k(i', j) = 0$ because if it had been a 1, then rows i and i' would have been identical, and we maintain the invariant that there are no duplicate rows.
  - So all $2^{|S|}$ labelings on S existed before transformation in T_k.

- Step 3: Each row of T' must contain at most d ones.
  - Suppose if T' has a row with k ones in columns $S \subseteq \{1, \dots, n\}$.
  - Then for each $j \in S$, there must be another row with a labeling that assigns ones to exactly $S \subseteq \{j\}$ (otherwise we would have been able to transform column j by changing the 1 to a 0).
  - Reasoning recursively, all 2^k subsets must exist.
  - Since T' has VC dimension at most d, $k \le d$.
  - Based on simple counting, we find that number of rows (remember, they're all distinct!) is upper bounded by $\sum_{i=0}^{d} \binom{n}{i}$, completing the first inequality of (249).
- Finishing the second part of the inequality of (249) is just algebra. Observe that for $n \ge d$,

$$\sum_{i=0}^{d} \binom{n}{i} \le \left(\frac{n}{d}\right)^d \sum_{i=0}^{d} \binom{n}{i} \left(\frac{d}{n}\right)^i \tag{252}$$

$$\le \left(\frac{n}{d}\right)^d \sum_{i=0}^{n} \binom{n}{i} \left(\frac{d}{n}\right)^i \tag{253}$$

$$= \left(\frac{n}{d}\right)^d \left(1 + \frac{d}{n}\right)^n \tag{254}$$

$$\le \left(\frac{n}{d}\right)^d e^d. \tag{255}$$

3.12 Norm-constrained hypothesis classes (Lecture 7)

- We started by establishing Rademacher complexity $R_n(\mathcal{F})$ as a measure of complexity of a function class that yielded generalization bounds when applied to the loss class $\mathcal{A}$. Then we specialized to boolean functions and zero-one losses, which led to notions of shattering coefficients and VC dimension as combinatorial means of bounding the Rademacher complexity.

- However, combinatorial notions are not the most appropriate way to think about complexity. For example, it is common in machine learning to have a huge number of features (large d) in a linear classifier, but where we regularize or constrain the norm of the weights, as in an SVM:

$$\hat{w} = \arg \min_{w \in \mathbb{R}^d : \|w\|_2 \le B_2} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y^{(i)} w \cdot x^{(i)}\}. \tag{256}$$

  In this case, d is the VC dimension doesn't really capture the true complexity of the hypothesis class. Somehow B_2 should play a role, no?

- Deriving Rademacher complexity will actually be easier for the norm-constrained setting. We will study the Rademacher complexity of three such examples:
  - Linear functions with L_2 norm constraints (suitable for kernel methods)
  - Linear functions with L_1 norm constraints (suitable for sparsity)
- Rademacher complexity of linear functions with weights bounded in an L_2 ball
  - **Theorem 11 (Rademacher complexity of L_2 ball)**
    - ∗ Let $\mathcal{F} = \{z \mapsto w \cdot z : \|w\|_2 \le B_2\}$ (bound on weight vectors).
    - ∗ Assume $\mathbb{E}_{Z\sim p^*}[\|Z\|_2^2] \le C_2^2$ (bound on spread of data points).
    - ∗ Then

$$\boxed{R_n(\mathcal{F}) \le \frac{B_2 C_2}{\sqrt{n}}.} \tag{257}$$

  - Proof of Theorem 11:
    - ∗ The key idea is to exploit the linear algebraic structure of Rademacher complexity.
    - ∗ Expand the definition:

$$R_n(\mathcal{F}) = \frac{1}{n}\mathbb{E}\left[\sup_{\|w\|_2\le B_2} \sum_{i=1}^n \sigma_i(w \cdot Z_i)\right]. \tag{258}$$

    - ∗ By Cauchy-Schwartz applied to w and $\sum_{i=1}^n \sigma_i Z_i$:

$$\le \frac{B_2}{n}\mathbb{E}\left[\|\sum_{i=1}^n \sigma_i Z_i\|_2\right]. \tag{259}$$

    - ∗ By concavity of $\sqrt{\cdot}$, we can push it outside the expectation:

$$\le \frac{B_2}{n}\sqrt{\mathbb{E}\left[\|\sum_{i=1}^n \sigma_i Z_i\|_2^2\right]}. \tag{260}$$

    - ∗ Distribute the sum; **expectation of cross terms is zero** by independence of σ_i (this is the key point, which turns n^2 terms into n terms):

$$= \frac{B_2}{n}\sqrt{\mathbb{E}\left[\sum_{i=1}^n \|\sigma_i Z_i\|_2^2\right]}. \tag{261}$$

- We can drop σ_i because it changes sign, not magnitude:

$$= \frac{B_2}{n}\sqrt{\mathbb{E}\left[\sum_{i=1}^{n} \|Z_i\|_2^2\right]}. \tag{262}$$

- Use the bound on Z_i:

$$\leq \frac{B_2}{n}\sqrt{nC_2^2}. \tag{263}$$

Simple algebra completes the proof.

- Rademacher complexity of linear functions with weights bounded in an L_1 ball
  - Motivation
    - Working with L_2 regularization has the advantage that we can use kernel methods, and the dimensionality could be infinite as long the norm is bounded.
    - In some applications, we have a finite but large set of features, and we believe that there are a relatively small subset that are relevant to our task (i.e., we believe in **parameter sparsity**). It is common to use L_1 regularization, or similarily, assume that the weights satisfy $\|w\|_1 \leq B_1$.

  Let us compute the Rademacher complexity of $\mathcal{F} = \{z \mapsto w \cdot z : \|w\|_1 \leq B_1\}$.
  - **Theorem 12 (Rademacher complexity of L_1 ball)**
    - Assume that the coordinates are bounded: $\|Z_i\|_\infty \leq C_\infty$ with probability 1 for all data points $i = 1, \dots, n$.
    - Then

$$\boxed{R_n(\mathcal{F}) \leq \frac{B_1 C_\infty \sqrt{2\log(2d)}}{\sqrt{n}}.} \tag{264}$$

  - Proof of Theorem 12
    - The key step is to realize that the L_1 ball ($\{w : \|w\|_1 \leq B_1\}$) is the convex hull of the following $2d$ weight vectors:

$$W = \cup_{j=1}^{d}\{B_1 e_j, -B_1 e_j\}. \tag{265}$$

Since the Rademacher complexity of a class is the same as the Rademacher complexity of its convex hull, we just need to look at the finite class:

$$R_n(\mathcal{F}) = \mathbb{E}\left[\sup_{w \in W} \frac{1}{n}\sum_{i=1}^{n} \sigma_i(w \cdot Z_i)\right]. \tag{266}$$

    - ∗ Apply Hölder's inequality, we have $w \cdot Z_i \le \|w\|_1 \|Z_i\|_\infty \le B_1 C_\infty$.
    - ∗ Applying Massart's finite lemma (Lemma 5) to the function class specified by W (of size $2d$) with $M^2 = B_1^2 C_\infty^2$, we get that

$$R_n(\mathcal{F}) \le \sqrt{\frac{2 B_1^2 C_\infty^2 \log(2d)}{n}}. \tag{267}$$

- – Remarks
    - ∗ It is useful to recall that as p increases,
        - · p-norms decrease: $\|w\|_p \ge \|w\|_q$
        - · Size of balls increase: $\{w : \|w\|_p \le B\} \subseteq \{w : \|w\|_q \le B\}$
    - ∗ Note that a L_1 bound on the parameters is placing a much stronger constraint than the L_2 norm, which allows us to measure the L_∞ norm of the data (which is much better) rather than the L_2 norm at the expense of a logarithmic dependence on d.
- – Ramifications under **sparsity**
    - ∗ FIGURE: [w and z as arrays]
    - ∗ L_1 regularization is often used when we believe that most features are irrelevant; formally, that the desired weight vector has $s \ll d$ non-zero entries.
    - ∗ You might have seen the intuition that L_1 regularization has sharp corners which encourage weights to be identically zero, but that doesn't directly tell us anything about generalization. We would like a stronger justification.
    - ∗ For convenience, assume that all entries of w and x have magnitude at most 1 ($\|w\|_\infty \le 1$, $\|x\|_\infty \le 1$).
    - ∗ It suffices to consider the hypothesis class $\|w\|_1 \le B_1 = s$.
    - ∗ Then the Rademacher complexity (and thus the expected risk) is $O\left(\frac{s\sqrt{\log d}}{\sqrt{n}}\right)$.
    - ∗ Interpretation: essentially the number of relevant features (s) controls the complexity, and we can have a ton of irrelevant features (an exponentially large number).
    - ∗ In contrast, if we use L_2 regularization, we would have $B_2 = \sqrt{s}$ and $C_2 = \sqrt{d}$. The Rademacher complexity of $\{w : \|w\|_2 \le B_2\}$ is $O\left(\frac{s\sqrt{d/s}}{\sqrt{n}}\right)$.
    - ∗ When $s \ll d$, then L_1 regularization is desirable, but if $s = d$, then L_1 regularization is worse by a factor of $\sqrt{\log d}$.
    - ∗ Note that these are heuristic arguments since we are comparing upper bounds. In this case, the heuristics are accurate, but in general, one should exercise caution.
- – In general, these norm-constrained bounds either do not depend (L_2 constrained) or only depend weakly (L_1 constrained) on the dimensionality d. This supports

the prevailing wisdom that it doesn't matter how many features you have (how big d is). As long as you regularize properly (constrain the norm of the weights), then you will still have good generalization.

- From hypothesis class to loss class (for binary classification)
  - Composition
    - So far, we have bounded the Rademacher complexity of various hypothesis classes $\mathcal{F}$. We still need to turn that into a bound on the loss class $\mathcal{A}$. For boolean functions with finite shattering coefficients, we showed that $\mathcal{F}$ and $\mathcal{A}$ had the same complexity (242). But now $\mathcal{F}$ contains real-valued functions, so the previous argument for shattering coefficients doesn't hold.
    - Consider a loss function ϕ for binary linear classification that only depends on the margin $m = yx \cdot w$. For example,
      - Zero-one loss: $\phi(m) = \mathbb{I}[m \le 0]$.
      - Hinge loss: $\phi(m) = \max\{0, 1 - m\}$.
      - FIGURE: [plot these loss functions]

      Let $w \in W$ be a set of weight vectors (for example, $\{w : \|w\|_2 \le B_2\}$). The loss class corresponding to these weight vectors is then:

      $$\mathcal{A} = \{(x, y) \mapsto \phi(yx \cdot w) : w \in W\}. \tag{268}$$

    - Since the loss function only depends on (x, y) through their product, we can equivalently think about our data points as simply being $z = xy$. Recall the function class that we've bounded the Rademacher complexity for:

      $$\mathcal{F} = \{z \mapsto w \cdot z : w \in W\}. \tag{269}$$

      So therefore, we can rewrite the loss class as a composition:

      $$\mathcal{A} = \phi \circ \mathcal{F}. \tag{270}$$

    - Note that we can only apply the composition rule for Rademacher complexity to ϕ which are Lipschitz. The hinge loss is 1-Lipschitz, which means the Rademacher bounds for the function class $\mathcal{F}$ directly carry over to the loss class: $R_n(\mathcal{A}) = R_n(\mathcal{F})$.
  - The zero-one loss is not Lipschitz, so we cannot cannot directly control the zero-one loss. Indeed, the zero-one loss is not sensitive to the the norm of w, so one should not expect a norm-based bound to work. Working with norms allows us to go to infinite dimensions, but this is the price we pay.
  - There is still a weaker statement that we can make, however. Let us define a margin-sensitive version of the zero-one loss which penalizes us whenever we don't predict correctly by at least a margin of γ:

    $$\phi_\gamma(m) = \mathbb{I}[m \le \gamma] \tag{271}$$

and

$$L_\gamma(w) = \mathbb{E}_{z \sim p^*}[\phi_\gamma(w \cdot z)] \tag{272}$$

be the associated expected risk.

- **Theorem 13 (margin-sensitive zero-one loss for linear classifiers)**
  - Let $\mathcal{F}$ be a set of linear functions.
  - Let L_γ be the expected risk under the margin-sensitive zero-one loss.
  - Let $\hat{w}$ and w^* be the weight vectors associated with the empirical and expected risk minimizers with respect to the loss $\tilde{\phi}_\gamma$ (to be defined later).
  - With probability at least $1 - \delta$,

$$L_0(\hat{w}) \le L_\gamma(w^*) + \frac{4R_n(\mathcal{F})}{\gamma} + \sqrt{\frac{2\log(2/\delta)}{n}}. \tag{273}$$

    Important: The LHS is one the usual zero-one loss.
  - Remarks
    - The moral of this theorem is that if it is possible to get low risk under the more stringent L_γ, then we should also be able to do well on zero-one loss.
    - This also in a way supports the use of SVMs, which try to maximize the margin; a larger margin means we have better generalization.
    - Also note that as margin γ decreases, the bound on the RHS gets worse, blowing up as $1/\gamma$.
- Proof
  - The problem is that ϕ_γ is not Lipschitz for any value of γ. So the key trick is to introduce a Lipschitz version $\tilde{\phi}_\gamma$ that interpolates smoothly between the two:

$$\tilde{\phi}_\gamma(m) = \min\left\{1, \max\left\{0, 1 - \frac{m}{\gamma}\right\}\right\}. \tag{274}$$

    - FIGURE: [plot $\tilde{\phi}_\gamma$]
  - Then the Rademacher complexity of this intermediate loss class can be bounded:

$$R_n(\tilde{\phi}_\gamma \circ \mathcal{F}) = \frac{R_n(\mathcal{F})}{\gamma}, \tag{275}$$

    because $\tilde{\phi}_\gamma$ is $\frac{1}{\gamma}$-Lipschitz. Applying Theorem 9, we get that:

$$\tilde{L}_\gamma(\hat{w}) \le \tilde{L}_\gamma(w^*) + \frac{4R_n(\mathcal{F})}{\gamma} + \sqrt{\frac{2\log(2/\delta)}{n}}. \tag{276}$$

- Note that $\phi_0 \leq \tilde{\phi}_\gamma \leq \phi_\gamma$, so the same relation holds on the expected risks for all $w \in W$:

$$L_0(w) \leq \tilde{L}_\gamma(w), \quad \tilde{L}_\gamma(w) \leq L_\gamma(w). \tag{277}$$

The final result follows from applying these two inequalities to (276).

[begin lecture 8] (8)

3.13 Covering numbers (metric entropy) (Lecture 8)

- Motivation
  - We can measure the complexity of a finite hypothesis class simply by computing its cardinality. For infinite hypothesis classes, we observed that shattering coefficient was an appropriate measure since (thanks to symmetrization) all that mattered was the behavior of a function class on a finite set of points. However, shattering coefficient only works for functions that return a finite number of values. Can we retain the combinatorial nature of shattering coefficients, but allow for real-valued functions (for example, to handle regression problems)?
  - The key measure we will explore in this section is covering numbers, which counts the number of balls of size ϵ one needs to cover the hypothesis class.
  - We can use the Massart's finite lemma to control the representatives; the behavior of the other functions is controlled by virtue of being close to some representative. In essence, covering numbers allows us to discretize the problem.

To talk about closeness of functions, we introduce the general notion of a metric space:

- **Definition 12 (metric space)**
  - A metric space $(\mathcal{X}, \rho)$ is defined over a set $\mathcal{F}$ with equipped with a metric ρ.
  - A metric $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ must be non-negative, symmetric, satisfy the triangle inequality, and evaluate to 0 iff its arguments are equal.
  - If $\rho(f, f') = 0$ is possible for $f \neq f'$, then we say ρ is a pseudometric. In this section, technically we will be working with the pseudometric.

- Examples of metrics ρ
  - Euclidean distance
    - For real vectors $\mathcal{X} = \mathbb{R}^d$.
    - The metric is $\rho(f, f') = \|f - f'\|_2$.
  - $L_2(P_n)$
    - This is the L_2 distance with respect to the empirical distribution over n data points: $P_n = \frac{1}{n}\sum_{i=1}^n \delta_{z_i}$.
    - Let $\mathcal{X}$ be a family of functions mapping $\mathcal{Z}$ to $\mathbb{R}$.
    - The metric is $\rho(f, f') = \left(\frac{1}{n}\sum_{i=1}^n (f(z_i) - f'(z_i))^2\right)^{\frac{1}{2}}$.

  - ∗ Remark: this metric can be thought of as computing an empirical standard deviation over differences between f and f'.
  - ∗ Remark: since we are restricting the functions to evaluations at n points, we can really think of these functions as n-dimensional vectors, with a metric which is Euclidean distance scaled down by $\sqrt{n}$.
  - ∗ FIGURE: [two functions and n points]

- **Definition 13 (ball)**
  - Let $(\mathcal{X}, \rho)$ be a metric space.
  - Then the ball with radius $\epsilon > 0$ centered at $f \in \mathcal{X}$ is defined as:

$$B_\epsilon(f) \stackrel{\text{def}}{=} \{f' \in \mathcal{X} : \rho(f, f') \le \epsilon\}. \tag{278}$$

- **Definition 14 (covering number)**
  - FIGURE: [covering a set with balls]
  - An **ϵ-cover** of a set $\mathcal{F} \subseteq \mathcal{X}$ with respect to a metric ρ is a finite subset $C = \{f_1, \dots, f_m\} \subseteq \mathcal{X}$[10] such that if we put a ball of radius ϵ at each f_j, the resulting union is a superset of $\mathcal{F}$; that is $\mathcal{F} \subseteq \cup_{j=1}^m B_\epsilon(f_j)$.
  - Equivalently, every point in $\mathcal{F}$ is at most distance ϵ away (under the metric ρ) from some point f_j in the cover C.
  - Define the ϵ-**covering number** of $\mathcal{F}$ with respect to ρ to be the size of the smallest cover:

$$N(\epsilon, \mathcal{F}, \rho) \stackrel{\text{def}}{=} \min\{m : \exists\{f_1, \dots, f_m\} \subseteq \mathcal{X}, \mathcal{F} \subseteq \cup_{j=1}^m B_\epsilon(f_j)\}. \tag{279}$$

  - The **metric entropy** of $\mathcal{F}$ is $\log N(\epsilon, \mathcal{F}, \rho)$.

- As ϵ decreases, the points f_j in the cover are a better approximation of the points in $B_\epsilon(f_j)$, but $N(\epsilon, \mathcal{F}, \rho)$ also increases. What is this tradeoff?

- **Example 7 (all functions)**
  - FIGURE: [plot a function, discretize range, put down $z_1, \dots, z_n$]
  - Let $\mathcal{F} = \mathcal{X}$ be all functions from $\mathbb{R}$ to $[0, 1]$.
  - Recall that under the metric $\rho = L_2(P_n)$, only function evaluations on the points $z_1, \dots, z_n$ matter.

[10] Note: there are two variants of covers, internal (where C needs to be in the set $\mathcal{F}$) and external (where the cover C need not be). We will work with external covers since it gives us some flexibility, and all we will care about is the cardinality of C.

  - In order to cover $\mathcal{F}$, fix any $f \in \mathcal{F}$. Our strategy is to construct some $g \in \mathcal{F}$ such that $\rho(f, g) \le \epsilon$ and count the number of g's we obtain as we sweep f across $\mathcal{F}$.
  - For each point z_i, we cover the range $[0, 1]$ with the set of discrete points $Y = \{2\epsilon, 4\epsilon, \dots, 1\}$. For any value of $f(z_i) \in [0, 1]$, we can pick $g(z_i) \in Y$ so that $|f(z_i) - g(z_i)| \le \epsilon$. The value of $g(z)$ for z not equal to any of the n points can be chosen arbitrarily. Averaging over all z_i, we get that $\rho(f, g) \le \epsilon$.
  - Furthermore, $|Y| = \frac{1}{2\epsilon}$, so

$$N(\epsilon, \mathcal{F}, L_2(P_n)) \le \left(\frac{1}{2\epsilon}\right)^n. \tag{280}$$

    The metric entropy is thus $O(n \log(1/\epsilon))$, which intuitively is too large. To see this, even if we ignored the discretization error ϵ for the moment, we would see that the empirical Rademacher complexity of the cover, by Massart's finite lemma is $O(\sqrt{\frac{n \log(1/\epsilon)}{n}}) = O(1)$, which does not go to zero. Clearly this class of functions is too large.
  - Let's consider a smaller function class which is still infinite-dimensional and pretty rich:

- **Example 8 (non-decreasing functions)**
  - FIGURE: [plot an increasing function, discretize range, put down $z_1, \dots, z_n$]
  - Let $\mathcal{F}$ be all non-decreasing functions from $\mathbb{R}$ to $[0, 1]$.
  - Recall that $z_1, \dots, z_n$ are the n fixed points. WLOG, assume they are sorted in increasing order.
  - Similar to Example 7, we break up the range $[0, 1]$ into $\frac{1}{\epsilon}$ levels $Y = \{\epsilon, 2\epsilon, \dots 1\}$.
  - Fix any function $f \in \mathcal{F}$. We will construct a piecewise constant approximation g for which $\rho(f, g) \le \epsilon$.
  - For each level $y \in Y$, consider the points z_i for which $f(z_i) \in [y - \epsilon, y]$. Set $g(z_i) = y$ for these points. Note that g is a non-decreasing function across $z_1, \dots, z_n$. By construction, $\rho(f, g) \le \epsilon$.
  - Now let's count the number of possible g's there are. The key observation is that because each g is non-decreasing we can associate each level $y \in Y$ with a leftmost the leftmost point z_i for which $g(z_i) = y$; the choice of leftmost points (n possible values) for each level ($|Y| = 1/\epsilon$ values) uniquely defines g. Therefore

$$N(\epsilon, \mathcal{F}, L_2(P_n)) = O(n^{1/\epsilon}), \tag{281}$$

    and the metric entropy is $O((\log n)/\epsilon)$. This is much better than that of all functions. The main difference is that we're choosing a point for each level rather than a level for each point.

Having set up covering numbers with some examples, we are ready to state our main theorems. Specifically, we will show that the metric entropy leads to an upper bound on the Rademacher complexity. This can be accomplished two ways:

- First, we will show a simple discretization argument that applies covering numbers at a single resolution ϵ.
- Next, we provide a more sophisticated argument called **chaining** that adaptively chooses the resolution, yielding tighter bounds.

- **Theorem 14 (simple discretization)**
  - Let $\mathcal{F}$ be a family of functions mapping $\mathcal{Z}$ to $[-1, 1]$.
  - The empirical Rademacher complexity of a function class $\mathcal{F}$ can be upper bounded using its covering number:

$$\hat{R}_n(\mathcal{F}) \le \inf_{\epsilon > 0} \left(\sqrt{\frac{2 \log N(\epsilon, \mathcal{F}, L_2(P_n))}{n}} + \epsilon \right). \tag{282}$$

  - Remark: the RHS involves two terms:
    - ∗ The first is the covering number, which increases as ϵ decreases. This comes from Massart's finite lemma.
    - ∗ The second is ϵ (which decreases as ϵ decreases). This is the penalty we pay for having a discretization.

- Preparation
  - We will also assume $Z_{1:n}$ are constant and suppress the explicit conditioning, writing $\mathbb{E}[A]$ instead of $\mathbb{E}[A \mid Z_{1:n}]$.
  - To simplify notation, we write $\|f\|$ for $\|f\|_{L_2(P_n)}$ and $\langle f, g \rangle$ for $\langle f, g \rangle_{L_2(P_n)}$.
  - Overloading notation, let $\sigma : \mathcal{Z} \to \{-1, +1\}$ be defined as a function which when evaluated on z_i returns the random sign σ_i. This allows us to write the empirical Rademacher complexity succinctly as:

$$\hat{R}_n(\mathcal{F}) = \mathbb{E}\left[\sup_{f \in \mathcal{F}} \langle \sigma, f \rangle\right]. \tag{283}$$

  - Note that $\|\sigma\| = 1$ since it's just a vector of $+1$s and -1s.
  - In summary, think about each function $f \in \mathcal{F}$ as an n-dimensional vector

$$\frac{1}{\sqrt{n}}[f(z_1), \dots, f(z_n)] \tag{284}$$

and the Rademacher variables $\sigma_1, \dots, \sigma_n$ also as an n-dimensional vector:

$$\frac{1}{\sqrt{n}}[\sigma_1, \dots, \sigma_n]. \tag{285}$$

We then are just using the usual Euclidean distance on them.

- Proof of Theorem 14:
  - Fix $\epsilon > 0$ and let C be an ϵ-cover of $\mathcal{F}$.
  - The proof follows from manipulating the empirical Rademacher complexity:

$$\hat{R}_n(\mathcal{F}) = \mathbb{E}\left[\sup_{f \in \mathcal{F}} \langle \sigma, f \rangle\right] \tag{286}$$

$$= \mathbb{E}\left[\sup_{g \in C} \sup_{f \in \mathcal{F} \cap B_\epsilon(g)} \langle \sigma, g \rangle + \langle \sigma, f - g \rangle\right] \tag{287}$$

$$= \mathbb{E}\left[\sup_{g \in C} \frac{1}{n} \langle \sigma, g \rangle + \epsilon\right] \quad \text{[Cauchy-Schwartz]} \tag{288}$$

$$= \sqrt{\frac{2 \log N(\epsilon, \mathcal{F}, L_2(P_n))}{n}} + \epsilon \quad \text{[Massart's finite lemma]}. \tag{289}$$

- **Example 9 (non-decreasing functions (with simple discretization))**
  - Let $\mathcal{F}$ be all non-decreasing functions from $\mathcal{Z} = \mathbb{R}$ to $[0, 1]$.
  - Plugging the covering number of $\mathcal{F}$ into Theorem 14:

$$\hat{R}_n(\mathcal{F}) \le \inf_{\epsilon > 0}\left(\sqrt{\frac{2 \cdot O(\frac{\log n}{\epsilon})}{n}} + \epsilon\right). \tag{290}$$

The RHS is minimized when the two terms are equal. Solving for ϵ and substituting it back yields:

$$\hat{R}_n(\mathcal{F}) = O\left(\left(\frac{\log n}{n}\right)^{\frac{1}{3}}\right). \tag{291}$$

  - Note that this bound provides a rate which is worse than the usual $\frac{1}{\sqrt{n}}$ rates. Is it because non-decreasing functions are just more complex than parametric function classes? Not in this case. We'll see shortly that this is just an artifact of the analysis being too weak, because it is only able to work at one level of resolution ϵ.

- Where is the looseness? The problem with simple discretization is that we are covering at resolution ϵ, which inevitably requires many functions ($N(\epsilon, \mathcal{F}, L_2(P_n))$ of them, in fact), but our application of Massart's finite lemma just assumes we have an arbitrary set of functions with magitude up to 1. But these functions aren't arbitrary—they live in a metric space, and the nearby ones are actually quite close (within ϵ). Rather than paying a 1 (the maximum deviation of a function $f \in \mathcal{F}$), wouldn't it be nice if we could pay something like ϵ (the maximum difference in nearby functions in the ϵ-cover)? Of course, certain pairs of functions in $\mathcal{F}$ are quite far.

- We will now introduce a clever technique called chaining gets at the intuition by constructing covers at multiple levels of resolution, and using functions in the coarser covers as waypoints. The idea is that the fewer functions in the coarse cover can have larger deviation, and the many functions in the finer covers has smaller deviation.

- **Theorem 15 (chaining (Dudley's theorem))**

  - Let $\mathcal{F}$ be a family of functions mapping $\mathcal{Z}$ to $\mathbb{R}$.
  - The empirical Rademacher complexity of a function class $\mathcal{F}$ can be upper bounded using an integral over its covering number:

$$\hat{R}_n(\mathcal{F}) \le 12 \int_0^\infty \sqrt{\frac{\log N(\epsilon, \mathcal{F}, L_2(P_n))}{n}} d\epsilon. \tag{292}$$

  - Remark: note that compared with Theorem 14, this bound involves an integral that sweeps across different resolutions ϵ, and importantly removes the additive ϵ penalty.

- Proof of Theorem 15

  - FIGURE: [lines corresponding to levels, where each level is discretized more finely than the previous]
  - Let $\epsilon_0 = \sup_{f \in \mathcal{F}} \|f\|$ be the maximum norm of a function $f \in \mathcal{F}$, which will serve the coarsest resolution. Let $\epsilon_j = 2^{-j}\epsilon_0$ for $j = 1, \dots, m$ be successively finer resolutions.
  - For each $j = 0, \dots, m$, let C_j be an ϵ_j-cover of $\mathcal{F}$.
  - Fix any $f \in \mathcal{F}$.
  - Let $g_j \in C_j$ be such that $\|f - g_j\| \le \epsilon_j$; take $g_0 = 0$. Note that g_j's depend on f.
  - Let us decompose f as follows:

$$f = f - g_m + \underbrace{g_0}_{=0} + \sum_{j=1}^m (g_j - g_{j-1}). \tag{293}$$

- Restating Massart's finite lemma, for a class of functions $\mathcal{B}$, the empirical Rademacher complexity is:

$$\hat{R}_n(\mathcal{B}) \le \left(\sup_{b\in\mathcal{B}} \|b\|\right)\sqrt{\frac{2\log|\mathcal{B}|}{n}}. \tag{294}$$

- Let us bound some norms:
  * $\|f - g_m\| \le \epsilon_m$
  * $\|g_j - g_{j-1}\| \le \|g_j - f\| + \|f - g_{j-1}\| \le \epsilon_j + \epsilon_{j-1} = 3\epsilon_j$ (since $2\epsilon_j = \epsilon_{j-1}$)
- Now compute the empirical Rademacher complexity:

$$\begin{aligned}
\hat{R}_n(\mathcal{F}) &= \mathbb{E}\left[\sup_{f\in\mathcal{F}}\langle\sigma, f\rangle\right] \quad \text{[definition]} && (295)\\
&= \mathbb{E}\left[\sup_{f\in\mathcal{F}}\langle\sigma, f - g_m\rangle + \sum_{j=1}^{m}\langle\sigma, g_j - g_{j-1}\rangle\right] \quad \text{[decompose } f\text{]} && (296)\\
&\le \epsilon_m + \mathbb{E}\left[\sup_{f\in\mathcal{F}}\sum_{j=1}^{m}\langle\sigma, g_j - g_{j-1}\rangle\right] \quad \text{[Cauchy-Schwartz]} && (297)\\
&\le \epsilon_m + \sum_{j=1}^{m}\mathbb{E}\left[\sup_{f\in\mathcal{F}}\langle\sigma, g_j - g_{j-1}\rangle\right] \quad \text{[push sup inside]} && (298)\\
&\le \epsilon_m + \sum_{j=1}^{m}\mathbb{E}\left[\sup_{g_j\in C_j, g_{j-1}\in C_{j-1}}\langle\sigma, g_j - g_{j-1}\rangle\right] \quad \text{[refine dependence]} && (299)\\
&\le \epsilon_m + \sum_{j=1}^{m}(3\epsilon_j)\sqrt{\frac{2\log(|C_j||C_{j-1}|)}{n}} \quad \text{[Massart's finite lemma]} && (300)\\
&\le \epsilon_m + \sum_{j=1}^{m}(6\epsilon_j)\sqrt{\frac{\log|C_j|}{n}} \quad \text{[since } |C_j| \ge |C_{j-1}|\text{]} && (301)\\
&= \epsilon_m + \sum_{j=1}^{m}12(\epsilon_j - \epsilon_{j+1})\sqrt{\frac{\log|C_j|}{n}} \quad \text{[since } \epsilon_j = 2(\epsilon_j - \epsilon_{j+1})\text{]} && (302)\\
&\le 12\int_0^\infty \sqrt{\frac{\log N(\epsilon, \mathcal{F}, L_2(P_n))}{n}}d\epsilon \quad \text{[bound sum with integral]} && (303)
\end{aligned}$$

  In the last step, we took $m \to \infty$, which makes the additive penalty $\epsilon_m \to 0$.
- FIGURE: [plot showing $N(\epsilon, \mathcal{F}, \rho)$ as a function of ϵ, with locations of $\epsilon_0, \epsilon_1, \dots$]
- Remark: The reason why chaining provides better results is the following. In simple discretization, we apply Massart's finite lemma on functions whose magitude is $\sup_{f\in\mathcal{F}}\|f\|$, whereas in chaining, we apply Massart's finite lemma on differences between functions in C_j and C_{j-1} whose range is $3\epsilon_j$.

- **Example 10 (non-decreasing functions (with chaining))**
  - Let $\mathcal{F}$ be all non-decreasing functions from $\mathcal{Z}$ to $[0, 1]$.
  - Note that $\|f\| \le 1$ for all $f \in \mathcal{F}$, so the coarsest resolution is $\epsilon_0 = 1$, so we only have to integrate ϵ up to 1, not ∞, since $\log N(\epsilon, \mathcal{F}, L_2(P_n)) = 0$ for $\epsilon \ge \epsilon_0$.
  - Plugging the covering number of $\mathcal{F}$ into Theorem 15:

$$\hat{R}_n(\mathcal{F}) \le 12 \int_0^1 \left(\sqrt{\frac{O(\frac{\log n}{\epsilon})}{n}} \right) d\epsilon \tag{304}$$

$$= O\left(\sqrt{\frac{\log n}{n}} \right) \int_0^1 \sqrt{\frac{1}{\epsilon}} d\epsilon \tag{305}$$

$$= O\left(\sqrt{\frac{\log n}{n}} \right). \tag{306}$$

  - Remark: compared with the bound using the simple discretization, we get a better bound ($n^{-\frac{1}{2}}$ versus $n^{-\frac{1}{3}}$).

- Summary
  - Covering numbers is a powerful technique that allows us to handle rich function classes such as all non-decreasing functions.
  - The essence is discretization of a function class, so that we can use Massart's finite lemma.
  - Chaining allows us to leverage multiple discretization resolutions ϵ to obtain tight bounds.
  - Covering numbers give you a lot of freedom for customization: choice of metric and choice of discretization.
  - Covering numbers can sometimes be more convenient; for example, the covering number of $\mathcal{F}_1 \cup \mathcal{F}_2$ is at most that of $\mathcal{F}_1$ plus that of $\mathcal{F}_2$, a property not shared by Rademacher complexity.

[begin lecture 9] (9)

3.14 Algorithmic stability (Lecture 9)

- Motivation
  - Let us shelve excess risk $L(\hat{h}) - L(h^*)$ for the moment, and instead consider the gap between the expected and empirical risk. It is easy to see that this quantity can be upper bounded using uniform convergence:

$$\mathbb{P}[L(\hat{h}) - \hat{L}(\hat{h}) \geq \epsilon] \leq \mathbb{P}[\sup_{h \in \mathcal{H}} L(h) - \hat{L}(h) \geq \epsilon]. \tag{307}$$

    Colloquially, this answers the following question: if I get 10% error on my training set, what should I expect my test error to be? (The answer is no more than $10\% + \epsilon$.) Analyzing the excess risk has relied on $\hat{h}$ being the ERM, but (307) actually holds for *any* estimator $\hat{h}$. It is useful to think about $\hat{h}$ as a function of the training examples: $\hat{h} = A(z_1, \dots, z_n)$, where A is an *algorithm*.
  - Uniform convergence hones in on studying the "size" of $\mathcal{H}$ (using Rademacher complexity, VC dimension, etc.). However, what if a learning algorithm A does not "use" all of $\mathcal{H}$? For example, what if you use naive Bayes, regularization via a penalty term, early stopping, or dropout training? All of these could in principle can return any hypothesis from $\mathcal{H}$, but are somehow constrained by their objective function or algorithm, so we might expect better generalization than simply looking at the complexity of $\mathcal{H}$ in an algorithm-independent way.
  - To be more concrete, let us analyze $\mathcal{H} = \{x \mapsto w \cdot x : w \in \mathbb{R}^d\}$, the class of linear functions (with no bounds on the norm). Define the norm $\|h\|_{\mathcal{H}}$ to be $\|w\|_2^2$ of the associated weight vector. Define the **regularized empirical risk minimizer** as follows:

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{L}(h) + \frac{\lambda}{2}\|h\|_{\mathcal{H}}^2. \tag{308}$$

  - Of course, for a given realization of the data $z_{1:n}$, the solution to (308) can be gotten by finding the ERM on the constrained set $\{h \in \mathcal{H} : \|h\|_{\mathcal{H}} \leq B\}$ for some bound B. But this connection is weak in that B would need to depend on the data to obtain an exact equivalence. How can we analyze (308) directly?
  - In this section, we will introduce bounds based on a notion of **algorithmic stability**, where we view a learning algorithm as a function A that take as input the data $z_{1:n}$ and outputs some hypothesis $\hat{h}$. We will study the generalization properties of $\hat{h}$ as a function of how stable the algorithm is. Notice the focus on algorithms rather than on the hypothesis class.

- Stability will be measured with respect to the difference in behavior of an algorithm on a training set S and a perturbed version S^i:
  - $S = (z_1, \dots, z_n)$: training data, drawn i.i.d. from p^*
  - $S^i = (z_1, \dots, z'_i, \dots, z_n)$: training data with an i.i.d. copy of the i-th example
  - z_0 is a new test example

We start with a definition of stability:

- **Definition 15 (uniform stability)**
  - We say that an algorithm $A : \mathcal{Z}^n \to \mathcal{H}$ has **uniform stability** β (or in the context of these notes, simply β-stable) with respect to a loss function ℓ if for all training sets $S \in \mathcal{Z}^n$, perturbations $S^i \in \mathcal{Z}^n$, and test example $z_0 \in \mathcal{Z}$,

$$|\ell(z_0, A(S)) - \ell(z_0, A(S^i))| \le \beta. \tag{309}$$

  - Note that this is a very strong condition in that the bound must hold uniformly for all z_0, S, S^i and is not reliant on the probability distribution. There are weaker notions detailed in Bosquet/Elisseeff (2002).
- **Example 11 (stability of mean estimation)**
  - Assume all points $z \in \mathbb{R}^d$ are bounded $\|z\|_2 \le B$.
  - Define the squared loss: $\ell(z, h) = \frac{1}{2}\|z - h\|_2^2$.
  - Define an algorithm that computes the regularized empirical risk minimizer:

$$A(S) \overset{\text{def}}{=} \arg\min_{h \in \mathbb{R}^d} \hat{L}(h) + \frac{\lambda}{2}\|h\|_2^2 \tag{310}$$

$$= \frac{1}{(1+\lambda)n} \sum_{i=1}^n z_i. \tag{311}$$

  - Then A has uniform stability $\boxed{\beta = \frac{6B^2}{(1+\lambda)n}}$.
  - Intuitively, averaging in a new point should contribute $O(1/n)$ and the $1/(1+\lambda)$ is just due to the shrinkage towards zero induced by the regularizer.
  - To derive this formally, define $v_1 = A(S) - z_0$ and $v_2 = \frac{1}{(1+\lambda)n}[z'_i - z_i]$.

$$|\ell(z_0, A(S)) - \ell(z_0, A(S^i))| \le \left|\frac{1}{2}\|v_1\|_2^2 - \frac{1}{2}\|v_1 + v_2\|_2^2\right| \tag{312}$$

$$\le \|v_2\|_2(\|v_1\|_2 + \frac{1}{2}\|v_2\|_2) \tag{313}$$

$$\le \frac{2B}{(1+\lambda)n}\left(2B + \frac{B}{(1+\lambda)n}\right) \tag{314}$$

$$\le \frac{2B}{(1+\lambda)n}\left(2B + B\right). \tag{315}$$

- We can see that as we increase regularization (larger λ) or amount of data (larger n), we have more stability (smaller β).

- **Example 12 (stability of linear predictors)**
  - Let $\mathcal{H} = \{x \mapsto w \cdot x : w \in \mathbb{R}^d\}$.
  - Assume that $\|x\|_2 \le C_2$ with probability 1 (according to the data-generating distribution p^*).
  - Assume the loss ℓ is 1−Lipschitz: for all $z_0 \in \mathcal{Z}$ and $h, h' \in \mathcal{H}$:

$$|\ell(z_0, h) - \ell(z_0, h')| \le \|h - h'\|_\infty \overset{\text{def}}{=} \sup_{x \in \mathbb{R}^d} |h(x) - h'(x)|. \tag{316}$$

    For example, for classification ($y \in \{-1, +1\}$), this holds for the hinge loss $\ell((x, y), h) = \max\{1 - yh(x), 0\}$.
  - Define the regularized empirical risk minimizer:

$$A(S) \overset{\text{def}}{=} \arg\min_{h \in \mathcal{H}} \hat{L}(h) + \frac{\lambda}{2}\|h\|_{\mathcal{H}}^2 \tag{317}$$

  - Then A has uniform stability $\boxed{\beta = \frac{C_2^2}{\lambda n}}$ with respect to ℓ. Again, the stability is similar to the case of mean estimation; the difference is that $\lambda = 0$ is very bad in regularized ERM while it's fine for mean estimation; this is because mean estimation is naturally stabilizing, whereas the ERM (with no control on the norm) is not stable. See the Bousquet/Elisseeff paper on stability for the proof.

Now that we have computed the stability for two examples and have a better intuition for what stability is measuring, let us state the main theorem, which relates stability to generalization:

- **Theorem 16 (generalization under uniform stability)**
  - Let A be an algorithm with uniform stability β.
  - Assume the loss is bounded: $\sup_{z,h} |\ell(z, h)| \le M$.
  - Then with probability at least $1 - \delta$,

$$\boxed{L(A(S)) \le \hat{L}(A(S)) + \beta + (\beta n + M)\sqrt{\frac{2\log(1/\delta)}{n}}.} \tag{318}$$

  - Remark: Due to the presence of βn, for this bound to be not vacuous, we must have $\beta = o\left(\frac{1}{\sqrt{n}}\right)$. Generally, we will have $\beta = O(\frac{1}{n})$, which guarantees that $L(A(S)) - \hat{L}(A(S)) = O(\frac{1}{\sqrt{n}})$.
- Proof of Theorem 16:

- Our goal is to bound the difference between the expected and empirical risks:

$$D(S) \overset{\text{def}}{=} L(A(S)) - \hat{L}(A(S)). \tag{319}$$

  Stability tells us about $\ell(z_0, A(S))$, which suggests using McDiarmid's inequality. That is the heart of the proof.

- Step 1: Bound the expectation of $D(S)$.
  - Recall that $S = (z_1, \dots, z_n), (z'_1, \dots, z'_n), z_0$ are all independent and therefore exchangeable. The basic trick is just to rename variables that preserve the dependency structure and therefore the expectation $\mathbb{E}[D(S)]$, and get it into a form where we can apply the definition of uniform stability:

$$\begin{aligned}
\mathbb{E}[D(S)] &= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}[\ell(z_0, A(S)) - \ell(z_i, A(S))]\right] \quad \text{[definition]} && (320)\\
&= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}[\ell(z'_i, A(S)) - \ell(z'_i, A(S^i))]\right] \quad \text{[renaming]} && (321)\\
&\leq \beta \quad \text{[definition of uniform stability]} && (322)
\end{aligned}$$

    The point is that in the first term z_0 is not in S and z_i is in S. This logic is preserved: z'_i is not in S and z'_i is in S^i.

- Step 2: show that $D(S)$ satisfies the bounded differences property.
  - We should expect such a property to hold given the definition of uniform stability. But we are not directly applying the bounded differences to the loss $\ell(z_0, A(S))$, but rather to $D(S)$. So there is slightly more work. The trick here is to break down differences using the triangle inequality into a chain of comparisons, each involving a single perturbation.
  - Let $\hat{L}^i$ denote the empirical expectation with respect to S^i.
  - We have

$$\begin{aligned}
&|D(S) - D(S^i)| && (323)\\
&= |L(A(S)) - \hat{L}(A(S)) - L(A(S^i)) + \hat{L}^i(A(S^i))| && (324)\\
&\leq |L(A(S)) - L(A(S^i))| + |\hat{L}(A(S)) - \hat{L}^i(A(S^i))| \quad \text{[triangle inequality]} && (325)\\
&\leq \underbrace{|L(A(S)) - L(A(S^i))|}_{\leq \beta} + \underbrace{|\hat{L}(A(S)) - \hat{L}(A(S^i))|}_{\leq \beta} + \underbrace{|\hat{L}(A(S^i)) - \hat{L}^i(A(S^i))|}_{\leq \frac{2M}{n}} && (326)\\
&\leq 2\beta + \frac{2M}{n}. && (327)
\end{aligned}$$

In the first two cases, we just used the fact that A is β-stable; recall that $\hat{L}(h) = \frac{1}{n}\sum_{i=1}^n \ell(z_i, h)$ and $L(h) = \mathbb{E}_{z_0\sim p^*}[\ell(z_0, h)]$; here it's important that β-stability holds uniformly for any first argument of ℓ, so that we can upper bound by β regardless of the dependence structure between the two arguments. In the third case, $\hat{L}$ and $\hat{L}^i$ just differ by one term which can differ by at most $2M$ and is scaled by $\frac{1}{n}$.

- ∗ Step 3: apply McDiarmid's inequality to bound $D(S)$ in high probability.
  - · This is a straightforward application. We have

$$\mathbb{P}[D(S)) - \mathbb{E}[D(S)] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{n(2\beta + \frac{2M}{n})^2}\right) \tag{328}$$

$$= \exp\left(\frac{-n\epsilon^2}{2(\beta n + M)^2}\right) \overset{\text{def}}{=} \delta. \tag{329}$$

Rewriting the bound and using the fact that $\mathbb{E}[D(S)] \leq \beta$, we have that with probability at least $1-\delta$,

$$D(S) \leq \beta + (\beta n + M)\sqrt{\frac{2\log(1/\delta)}{n}}. \tag{330}$$

- – Application to linear functions
  - ∗ Recall that for regularized ERM on linear functions with 1-Lipschitz losses (like hinge loss), $\beta = \frac{C_2^2}{\lambda n}$, where λ is the regularization strength and $\|x\|_2 \leq C_2^2$. Plugging this value of β into Theorem 16, we get that with probability at least $1-\delta$,

$$L(A(S)) \leq \hat{L}(A(S)) + O\left(\frac{C_2^2 + M}{\lambda\sqrt{n}}\right). \tag{331}$$

Note that this bound has the right dependence on n, but has a worse dependence on C_2 compared to the Rademacher bounds.

3.15 PAC-Bayesian bounds (Lecture 9)

- Bayesians and frequentists
  - – A Bayesian estimation procedure starts with a **prior distribution** $P(h)$ over hypotheses, observe the training data $z_1, \dots, z_n$, and produces a **posterior distribution** $Q(h) \propto P(h)\prod_{i=1}^n F(z_i \mid h)$, where F is the likelihood function. Bayesian procedures are optimal (see the work of Edwin Jaynes) assuming the prior and the likelihood functions are correct. But what happens if either is wrong (which is all the time)? We still want some guarantees.

  - From the cold frequentist perspective, this Bayesian procedure is just another estimator, an algorithm that takes some data $z_1, \dots, z_n$ and returns a single hypothesis $\hat{h}$ or a posterior $Q(h)$ over hypotheses. In this light, we can still ask for frequentist guarantees: how does the expected and empirical risks compare when data is generated form any distribution?
  - Philosophically, this hybrid view is quite satisfying: Bayesian procedures are often very sensible, and now we can still hold them up to the scrutiny of frequentist analysis.

- Bounds that depend on the prior
  - In fact, we already have frequentist guarantees that relate the expected and empirical risks for all $h \in \mathcal{H}$, and thus for any estimator $\hat{h}$.
  - Recall that for finite hypothesis classes with loss bounded in $[0, 1]$, using Hoeffding's inequality plus the union bound, we have that with probability at least $1 - \delta$,

$$\forall h \in \mathcal{H} : L(h) \le \hat{L}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log(1/\delta)}{2n}}. \tag{332}$$

    (This is similar to Theorem 7, but the bound is on $L - \hat{L}$ rather than excess risk.)
  - But in this bound, each $h \in \mathcal{H}$ is treated the same, whereas the prior $P(h)$ explicitly says that some are more likely than others. Can we *incorporate the prior $P(h)$ into the analysis without assuming that the prior is "correct"*?
  - PAC-Bayesian bounds provide exactly this. In this class, we will consider two types:
    * Occam bound: assume a countable hypothesis class, algorithm outputs a single hypothesis
    * PAC-Bayesian theorem: assume an infinite hypothesis class, algorithm outputs a posterior

    Let us start with Occam bound, which captures the key intuitions:

- **Theorem 17 (Occam bound)**
  - Let $\mathcal{H}$ be a countable hypothesis class.
  - Let the loss function be bounded: $\ell(z, h) \in [0, 1]$.
  - Let P be any "prior" distribution over $\mathcal{H}$.
  - Then with probability at least $1 - \delta$,

$$\boxed{\forall h \in \mathcal{H} : L(h) \le \hat{L}(h) + \sqrt{\frac{\log(1/P(h)) + \log(1/\delta)}{2n}}.} \tag{333}$$

- Let's try to understanding this bound with a few special cases.
  - If the prior puts all its mass on one hypothesis h_0 ($P(h) = \mathbb{I}[h = h_0]$), then the bound is just the standard Hoeffding bound you would get for a single hypothesis.
  - If we have a uniform distribution over some subset of hypotheses $S \subseteq \mathcal{H}$ ($P(h) = \mathbb{I}[h \in S]/|S|$), then we recover the standard result for finite hypotheses (similar to Theorem 7, which is for excess risk).
  - This reveals how PAC-Bayes is a generalization: rather than commiting to prior distributions which are uniform over some support $S \subseteq \mathcal{H}$, we can have prior distributions which place different probability mass on different hypotheses. We can let $\mathcal{H}$ be as large as we want as long as we still have probabilities ($\sum_{h \in \mathcal{H}} P(h) = 1$). In some sense, $P(h)$ defines a fuzzy hypothesis class.

- Proof of Theorem 17:
  - The proof is very simple. The key idea is to allocate our confidence parameter δ across different hypotheses proportionally based on the prior $P(h)$.
  - By Hoeffding's inequality, we have that for any fixed $h \in \mathcal{H}$:

$$\mathbb{P}[L(h) \geq \hat{L}(h) + \epsilon] \leq \exp(-2n\epsilon^2). \tag{334}$$

  - If we set the RHS to $\delta P(h)$, then we get that with probability at most $\delta P(h)$,

$$L(h) \geq \hat{L}(h) + \sqrt{\frac{\log(1/(\delta P(h)))}{2n}}. \tag{335}$$

  - Applying the union bound across all $h \in \mathcal{H}$, we have that with probability at most δ,

$$\exists h \in \mathcal{H} : L(h) \geq \hat{L}(h) + \sqrt{\frac{\log(1/(\delta P(h)))}{2n}}. \tag{336}$$

  - Take the contrapositive of this statement to get the statement in the theorem.

- The bound also suggests an algorithm:
  - The ultimate goal is to finte h that minimizes $L(h)$. Of course, we can't evaluate $L(h)$. But we do have an alleged upper bound on $L(h)$!
  - The bound is interesting because the penalty term $\log(1/P(h))$ actually depends on the hypothesis h, whereas all our previous bounds depended only on the complexity of $\mathcal{H}$. Let's treating the RHS bound as an objective to be minimized by an algorithm. Recall that this bound holds simultaneously for all $h \in \mathcal{H}$, which means that it will hold for the output of any algorithm $A(S)$.

  - Motivated by the bound, we can define an *algorithm that actually uses the bound* by minimizing the RHS:

$$A(S) \overset{\text{def}}{=} \arg\min_{h \in \mathcal{H}} \hat{L}(h) + R(h), \quad R(h) = \sqrt{\frac{\log(1/P(h)) + \log(1/\delta)}{2n}}. \tag{337}$$

    From this perspective, the bound provides a regularizer $R(h)$ which penalizes h more if it has small prior probability $P(h)$. The algorithm $A(S)$ thus balances the empirical risk $\hat{L}(h)$ with the regularizer $R(h)$.

  - As we get more data ($n \to \infty$), the regularizer also goes to zero, meaning that we will trust the empirical risk more, allowing us to consider more complex hypotheses in $\mathcal{H}$. This is a pretty nice behavior to have.
  - Note this is not a Bayesian procedure. The closest thing is if we let the loss function be the negative log-likelihood $\ell(z, h) = -\log F(z \mid h)$, then the MAP estimate would be $\arg\min_{h \in \mathcal{H}} \hat{L}(h) + \frac{\log(1/P(h))}{n}$, so the only difference is the square root. A partial explanation is that in order to obtain frequentist guarantees without any smoothness conditions, we need to regularize more (with the square root).

- There are two things lacking about the Occam bound:
  - It only applies to countable $\mathcal{H}$, which does not include the set of all weight vectors, for example.
  - It only embraces half of the Bayesian story: while we have a prior $P(h)$, only a single $h \in \mathcal{H}$ is returned rather than a full posterior $Q(h)$.

The following theorem generalizes the Occam bound:

- **Theorem 18 (PAC-Bayesian theorem (McAllester))**
  - Let the loss function be bounded: $\ell(z, h) \in [0, 1]$.
  - Let P be any "prior" distribution over $\mathcal{H}$.
  - Let Q_S be any "posterior" distribution over $\mathcal{H}$, which is a function of the training data S.
  - Then with probability at least $1 - \delta$,

$$\boxed{\mathbb{E}_{h \sim Q_S}[L(h)] \le \mathbb{E}_{h \sim Q_S}[\hat{L}(h)] + \sqrt{\frac{\mathrm{KL}\left(Q_S \| P\right) + \log(4n/\delta)}{2n - 1}}.} \tag{338}$$

- Proof: see the McAllester paper.
- To recover the Occam bound (up to constants), we simply set Q_S to be a point mass at some h.

3.16 Interpretation of bounds (Lecture 9)

- Now that we've derived a whole host of generalization bounds, let us take a step back and ask the question: how should we think about these bounds?
- Properties
  - One could evaluate these bounds numerically, but they will probably be too loose to use directly. The primary purpose of these bounds is to instead formalize the relevant properties of a learning problem and characterize their relationship to the expected risk, the quantity of interest.
  - The relationships solidify intuitions about learning. Here are some examples:
    - If we have d features, $n \sim d$ training examples suffices to learn. If the number of features increase by 2, we need to increase n by 2 as well to maintain the same estimation error.
    - We can actually have as many features d as we want (even infinite), so long as we regularize properly using L_2 regularization: bounds depend on norm B not dimension d.
    - If there are many irrelevant features use L_1 regularization: the L_1 ball is just much smaller than the L_2 ball. Here, exploiting the structure of the problem leads to better bounds (and algorithms).
    - If there is low noise in the problem (in the realizable setting, some predictor obtains zero expected risk), then estimation error is smaller ($O(1/n)$ versus $O(1/\sqrt{n})$ convergence).
- Excess risk is only part of the story.
  - It is important to note that generalization bounds focus on addressing the (excess risk $L(\hat{h}) - L(h^*)$, not the approximation error $L(h^*)$.
  - For example, if d is the number of features, then $L(\hat{h}) - L(h^*) = O(\sqrt{\frac{d}{n}})$ shows that by adding more features, the estimation error will worsen. However, we hope that $L(h^*)$ will improve. Therefore, in practice, one can still hope to reduce $L(\hat{h})$ by adding additional features.
  - The technical core of this section is **concentration of measure**: as you aggregate over increasing amounts of **independent** data, many of the relevant quantities convergence. Insight is obtained by closely observing how fast these quantities converge.
- What loss function?
  - The bounds derived using finite hypothesis classes or finite VC dimension operated on the zero-one loss (supposing for the moment that's our desired loss function). However, the empirical risk minimizer in this case is NP-hard to compute.

  - However, the norm-based bounds using Rademacher complexity required a Lipschitz loss function such as the hinge loss, which is a surrogate loss. This gives us results on an empirical risk minimizer which we can actually evaluate in practice. One can say is that the zero-one loss is upper bounded by the hinge loss, but this is relatively weak, and in particular, minimizing the hinge loss even in the limit of infinite data will not give you something that minimizes the zero-one loss. A special case is that for universal kernels, minimizing the hinge loss (among others) does correspond to minimizing the zero-one loss in the limit of infinite data (Bartlett/Jordan/McAuliffe, 2005).

3.17 Summary (Lecture 9)

- The main focus of this section was to study the **excess risk** $L(\hat{h}) - L(h^*)$ of the **empirical risk minimizer** $\hat{h}$. In particular, we wanted that with probability at least $1-\delta$, the excess risk $L(\hat{h}) - L(h^*)$ is upper bounded by something that depends on the complexity of the learning problem and n, the number of i.i.d. training examples. (Another goal is to bound the difference between expected and empirical risks $L(\hat{h}) - \hat{L}(\hat{h})$.)

- The excess risk is often within a factor of two of the difference between empirical and expected risk: $\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h)|$, which has a form which suggests using uniform convergence tools to bound it.

- Results on excess risk $L(\hat{h}) - L(h^*)$:
  - Realizable, finite hypothesis class: $O\left(\frac{\log |\mathcal{H}|}{n}\right)$
  - Finite hypothesis class: $O\left(\sqrt{\frac{\log |\mathcal{H}|}{n}}\right)$
  - Shattering coefficient / VC dimension: $O\left(\sqrt{\frac{\log s(\mathcal{H},n)}{n}}\right) = O\left(\sqrt{\frac{\mathrm{VC}(\mathcal{H}) \log n}{n}}\right)$
  - L_2 norm constrained—kernels ($\|w\|_2 \le B_2, \|x\|_2 \le C_2$): $O\left(\frac{B_2 C_2}{\sqrt{n}}\right)$
  - L_1 norm constrained—sparsity ($\|w\|_1 \le B_1, \|x\|_\infty \le C_\infty$): $O\left(\frac{B_1 C_\infty \sqrt{\log d}}{\sqrt{n}}\right)$
  - Non-decreasing functions (via covering numbers and chaining): $O\left(\int_0^\infty \sqrt{\frac{\log N(\epsilon, \mathcal{H}, L_2(P_n))}{n}} d\epsilon\right) = O\left(\sqrt{\frac{\log n}{n}}\right)$

- Technical tools
  - Tail bounds

    - How much do random variables deviate from their mean? We generally look for sharp concentration bounds, which means that the probability of deviation by a constant decays **exponentially** fast as a function of n: $\mathbb{P}[G_n - \mathbb{E}[G_n] \geq \epsilon] \leq c^{-n\epsilon^2}$.
    - When G_n is an average of i.i.d. **sub-Gaussian** variables Z_i we can bound the moment generating function and get the desired bound (**Hoeffding's inequality** for bounded random variables). Sub-Gaussian random variables include Gaussian and bounded random variables (it is convenient to assume our loss or data is bounded), but not Laplace random variables, which has heavier tails (there, we need sub-exponential bounds).
    - When G_n is the result of applying a function with bounded differences to i.i.d. variables, then **McDiarmid's inequality** gives us the same bound.
  - Complexity control
    - For a single hypothesis, we can directly apply the tail bound to control the difference $L(h) - \hat{L}(h)$. However, we seek **uniform convergence** over all $h \in \mathcal{H}$.
    - Finite hypothesis classes: $\log |\mathcal{H}|$ (use simple union bound)
    - For infinite hypothesis classes, the key intuition is that the complexity of $\mathcal{H}$ is described by **how it acts on n data points**. Formally, **symmetrization** (introduce ghost dataset and Rademacher variables) reveals this.
    - The complexity is the **shattering coefficient** $s(\mathcal{H}, n)$ (technically of the loss class $\mathcal{A}$). By **Sauer's lemma**, the shattering coefficient can be upper bounded using the **VC dimension** $\mathrm{VC}(\mathcal{H})$.
    - Rademacher complexity $R_n(\mathcal{H})$ measures how well $\mathcal{H}$ can fit random binary labelings (noise). Rademacher complexity is nice because of the numerous compositional properties (convex hull, Lipschitz composition, etc.)
      - By **Massart's finite lemma**, we can relate $R_n(\mathcal{H})$ to the shattering coefficient.
      - For L_2 norm constrained linear functions, we used linearity and Cauchy-Schwartz, which enables us to analyze kernels.
      - For L_1 norm constrained linear functions, we used the fact that the L_1 polytope is really simple as it only has $2d$ vertices.

- Other paradigms
  - We also studied two alternative pardigms for analyzing learning, both of which were motivated by the need for greater nuance. Let A be any learning algorithm that maps training data S to a hypothesis $\hat{h} \in \mathcal{H}$. The algorithm A could be the ERM, but it need not be.
  - Typical uniform convergence bounds yield results that depend on the complexity

of the entire hypothesis class $\mathcal{H}$:

$$L(A(S)) - \hat{L}(A(S)) \leq \text{SomeFunction}(\mathcal{H}). \tag{339}$$

- Algorithmic stability allows us to obtain a bound that depends on properties of the algorithm A (i.e., its stability β) rather than $\mathcal{H}$. We obtained bounds of the form:

$$L(A(S)) - \hat{L}(A(S)) \leq \text{SomeFunction}(A). \tag{340}$$

- PAC-Bayesian bounds allow us to incorporate the prior into the analysis without sacrificing objective rigor. We obtained bounds of the form:

$$L(A(S)) - \hat{L}(A(S)) \leq \text{SomeFunction}(A(S)). \tag{341}$$

  Note that the bound depends on the output of the algorithm.

Of course, algorithmic stability and PAC-Bayes bounds only bound the expected and empirical risks, not the excess risk, which requires the heavyweight uniform convergence. Intuitively, one expects this: to bound excess risk $A(S)$ has to be related to h^* somehow, not just be stable or be based on a possibly incorrect prior.

3.18 References

- Bousquet/Boucheron/Lugosi, 2008: Introduction to Statistical Learning Theory
- Martin Wainwright's lecture notes
- Peter Bartlett's lecture notes
- Sham Kakade's lecture notes
- Tomaso Poggio and Lorenzo Rosasco's lecture notes
- Bartlett/Mendelson, 2002: Rademacher and Gaussian Complexities: Risk Bounds and Structural Results
- Kakade/Sridharan/Tewari, 2008: On the Complexity of Linear Prediction: Risk Bounds, Margin Bounds, and Regularization
- Bosquet/Elisseeff, 2002: Stability and Generalization
- David McAllester's notes on PAC-Bayesian bounds

4 Kernel methods

[begin lecture 10] (10)

4.1 Motivation (Lecture 10)

- So far in this class, we have focused on studying the excess risk $L(\hat{h}) - L(h^*)$, which measures how far our estimated predictor $\hat{h}$ is away from the best possible predictor $h^* \in \mathcal{H}$ in terms of expected risk. But this is only half of the story, since the expected risk that we care about is the sum of the excess risk and the best expected risk of the hypothesis class:

  $$L(\hat{h}) = L(\hat{h}) - L(h^*) + L(h^*). \tag{342}$$

  The latter term has to do with how expressive $\mathcal{H}$ is.

- We have mainly focused on linear models, where the prediction is a function of the inner product $\langle w, x \rangle$ between a weight vector $w \in \mathbb{R}^d$ and an input $x \in \mathbb{R}^d$ (e.g., for regression, the prediction function is just $f(x) = \langle w, x \rangle$). However, real data often exhibit highly non-linear relationships which are important to model.

- But we can sneakily replace $\langle w, x \rangle$ with $\langle w, \phi(x) \rangle$, where $\phi : \mathcal{X} \to \mathbb{R}^d$ is an arbitrary feature map:

  - Example: $\phi(x) = (1, x, x^2)$ for $x \in \mathbb{R}$
  - Example: $\phi(x) = $ (count of a appearing in $x, \dots$) for a string x.

  Note that x does not even need to be a real vector. In general, we assume that $x \in \mathcal{X}$ for some set $\mathcal{X}$ of all possible inputs (we won't assume any further structure on $\mathcal{X}$ for now).

- Therefore, we can represent very expressive **non-linear** functions by making $\phi(x)$ complex, but the problem is that $\phi(x)$ might have to be very high-dimensional in order to attain the desired degree of expressiveness, resulting in computationally expensive algorithms.

- But perhaps we can cut down on computation.

  - Suppose we are trying to minimize the average squared loss over n training examples:

    $$\hat{L}(w) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y^{(i)} - \langle w, \phi(x^{(i)}) \rangle)^2. \tag{343}$$

Optimization algorithms typically access the function only through the gradient, which is:

$$\nabla \hat{L}(w) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \langle w, \phi(x^{(i)})\rangle)\phi(x^{(i)}). \tag{344}$$

An algorithm that only accumulates (scaled) gradients will ultimately produce a weight vector w that can be written as a linear combination of the feature vectors of the data points (we will revisit this property in much greater generality when we study the representer theorem):

$$w = \sum_{i=1}^{n}\alpha_i \phi(x^{(i)}). \tag{345}$$

Given a weight vector of this form, we can make predictions as follows:

$$\langle w, \phi(x)\rangle = \sum_{i=1}^{n}\alpha_i \langle \phi(x^{(i)}), \phi(x)\rangle. \tag{346}$$

- The key is that predictions only depend on the **inner product** between the feature vectors. This suggests we can work with in high (even infinite!) dimensions as long as we can compute this inner product efficiently. The computational tradeoff is as follows: we store the α_i's (n numbers) rather than w (d numbers). If the number of features d is much larger than the number of training examples n, then we win on space. On the other hand, if we have a lot of data (n), we must resort to approximations, as we'll see later.
- **Example 13 (computing with quadratic features)**
  * Let the raw input be $x \in \mathbb{R}^b$.
  * Feature map with all quadratic terms:

    $$\phi(x) = [x_1^2, \dots, x_b^2, \sqrt{2}x_1x_2, \dots, \sqrt{2}x_1x_b, \sqrt{2}x_2x_3, \dots, \sqrt{2}x_2x_b, \dots, \sqrt{2}x_{b-1}x_b], \tag{347}$$

    There are $O(b^2)$ dimensions.
  * Explicit computation: $\langle w, \phi(x)\rangle$ takes $O(b^2)$ time.
  * Implicit computation: $\langle \phi(x), \phi(x')\rangle = \langle x, x'\rangle^2$, which takes $O(b)$ time. $\langle w, \phi(x)\rangle$ requires doing this for each of n data points, which takes $O(bn)$ time.
- It's important to realize that mathematically, we're still computing the same function values as before. All we've done is perform a computational sleight of hand, known as the **kernel trick**.
- Aside: sometimes, we can compute dot products efficiently in high (even infinite) dimensions without using the kernel trick. If $\phi(x)$ is sparse (as is often the case in natural language processing), then $\langle \phi(x), \phi(x')\rangle$ can be computed in $O(s)$ time rather than $O(d)$ time, where s is the number of nonzero entries in $\phi(x)$.

- Since these algorithms only depend on the inner product, maybe we can just cut to the chase and directly write down functions k that correspond to inner products: $k(x, x') = \langle \phi(x), \phi(x') \rangle$ for some feature map ϕ.

- This is a key conceptual change: it shifts our perspective from thinking in terms of features of single inputs to thinking about notions of "similarity" $k(x, x')$ between two inputs x and x'. Sometimes, similarities might be more convenient from a modeling point of view.

- Therefore, kernels offers two things:
  - A *computationally* efficient way of working with high (and even infinite) dimensional $\phi(x)$ **implicitly**.
  - A different perspective on features, which can be more natural from a *modeling* perspective for certain applications, and help us understand our model family, even if we computationally end up working with features.

4.2 Kernels: definition and examples (Lecture 10)

- We now define kernels formally. While we have suggested that kernels are related to feature maps, we hold back on establishing their formal connection until later to avoid confusion.

- **Definition 16 (kernel)**
  - A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive semidefinite kernel (or more simply, a kernel) iff for every finite set of points $x_1, \dots, x_n \in \mathcal{X}$, the **kernel matrix** $K \in \mathbb{R}^{n \times n}$ defined by $K_{ij} = k(x_i, x_j)$ is positive semidefinite.

- Let us now give some examples of kernels, and then prove that they are indeed valid according to Definition 16. Assume in the following that the input space is $\mathcal{X} = \mathbb{R}^b$. Note that we can visualize a kernel for $\mathcal{X} = \mathbb{R}$ by fixing x' to some value (say 1) and plotting $k(x, 1)$.
  - Linear kernel:

$$k(x, x') = \langle x, x' \rangle . \tag{348}$$

  - Polynomial kernel:

$$k(x, x') = (1 + \langle x, x' \rangle)^p. \tag{349}$$

    * Intuition: for boolean features ($x \in \{0, 1\}^b$), this corresponds to forming conjunctions of the original features.

    - ∗ Here, we can check that the corresponding dimensionality (number of features) is $O(b^p)$, which is exponential in p.

  - Gaussian kernel:

$$k(x, x') = \exp\left(\frac{-\|x - x'\|_2^2}{2\sigma^2}\right). \tag{350}$$

    - ∗ A Gaussian kernel puts a smooth bump at x.
    - ∗ The bandwidth parameter σ^2 governs how smooth the functions should be: larger σ^2 yields more smoothness.
    - ∗ The corresponding dimensionality is infinite (as we'll see later), so computationally, we really have no choice but to work with kernels.
    - ∗ The Gaussian is in some sense the "go to" kernel in machine learning, because it defines a very expressive hypothesis class, as we will see.

  - Sequence mis-match kernel:
    - ∗ The above kernels have been defined for continuous input spaces $\mathcal{X} \subseteq \mathbb{R}^b$, but we can define kernels on any type of object.
    - ∗ Suppose $\mathcal{X} = \Sigma^*$ is the set of all possible sequences (strings) over some alphabet Σ. We want to define a kernel between two strings which measures their similarity, a problem that arises in NLP and computational biology. For example, consider the strings *format* and *fmt*.
    - ∗ A string $u \in \mathcal{X}$ is a subsequence of $x \in \mathcal{X}$ if there exists a sequence of indices $\mathbf{i} = (i_1, \dots, i_{|u|})$ such that $1 \le i_1 < \cdots < i_{|u|} \le |x|$ such that $u_j = x_{i_j}$. In this case, we write $u = x(\mathbf{i})$.
    - ∗ Note that x has exponentially many subsequences in general.
    - ∗ Now define a kernel between two sequences x, x' to be a weighted number of common subsequences:

$$k(x, x') = \sum_{u \in \Sigma^*} \sum_{(\mathbf{i}, \mathbf{j}): x(\mathbf{i}) = x'(\mathbf{j}) = u} \lambda^{|\mathbf{i}| + |\mathbf{j}|}, \tag{351}$$

      for some decay parameter $0 \le \lambda \le 1$. Smaller values of λ discount longer subsequences more.

- Non-example: $k(x, x') = \mathbb{I}[\|x - x'\|_2 \le 1]$
  - Exercise: show that k is not a kernel function.

- Now we show the above kernels (linear, polynomial, Gaussian) are actually valid according to Definition 16. Let $x_1, \dots, x_n \in \mathcal{X}$ be any points. We have to check that the kernel matrix K formed by $K_{ij} = k(x_i, x_j)$ is positive semidefinite. But we first start with some general principles that will allow us to easily check whether a given function k is a kernel easily.

- General principles for checking kernels
  - Base case: for any function $f : \mathcal{X} \to \mathbb{R}$, $k(x, x') = f(x)f(x')$ is positive semidefinite.
    - Proof: the kernel matrix can be written as $K = uu^\top \succeq 0$, where $u = (f(x_1), \cdots, f(x_n))$.
  - Recursive case: given two kernels k_1, k_2, we can create new kernels k. Note that to check that k is a kernel, it suffices to check that $K_1, K_2 \succeq 0 \Rightarrow K \succeq 0$, where K_1, K_2, K are the corresponding kernel matrices of k_1, k_2, k.
  - Sum (recursive case): $k(x, x') = k_1(x, x') + k_2(x, x')$
    - Since positive semidefiniteness is closed under addition, $K = K_1 + K_2 \succeq 0$.
  - Product (recursive case): $k(x, x') = k_1(x, x')k_2(x, x')$
    - $K = K_1 \circ K_2$ corresponds to elementwise product.
    - Since K_1, K_2 are positive semidefinite, we can take their eigendecompositions:
      - $K_1 = \sum_{i=1}^n \lambda_i u_i u_i^\top$
      - $K_2 = \sum_{j=1}^n \tau_j z_j z_j^\top$
    - Taking the elementwise product yields the following eigendecomposition, showing that K is also positive semidefinite:
      - $K = \sum_{i=1}^n \sum_{j=1}^n \lambda_i \tau_j (u_i \circ z_j)(u_i \circ z_j)^\top$, where $\circ$ denotes elementwise products.

- Using these three principles, we can show that the linear, polynomial, and Gaussian kernels are valid.
  - Linear kernel: sum over kernels defined by functions of the form $f(x) = x_i$.
  - Polynomial kernel: given the linear kernel $\langle x, x' \rangle$, add 1 (which is a kernel); this shows that the sum $\langle x, x' \rangle + 1$ is a kernel. By the product property, $(\langle x, x' \rangle + 1)^2$ is also a kernel. Repeat $p - 1$ times to show that $(\langle x, x' \rangle + 1)^p$ is a kernel.
  - Gaussian kernel:
    - Rewrite

$$k(x, x') = \exp\left(\frac{-\|x\|_2^2}{2\sigma^2}\right) \exp\left(\frac{-\|x'\|_2^2}{2\sigma^2}\right) \exp\left(\frac{\langle x, x' \rangle}{\sigma^2}\right). \tag{352}$$

    - The first two factors are handled by the base case.
    - For the third factor, take the Taylor expansion:

$$\exp\left(\frac{\langle x, x' \rangle}{\sigma^2}\right) = 1 + \frac{\langle x, x' \rangle}{\sigma^2} + \frac{1}{2}\frac{\langle x, x' \rangle^2}{\sigma^4} + \frac{1}{6}\frac{\langle x, x' \rangle^3}{\sigma^6} + \cdots \tag{353}$$

      Each term is just a homogenous polynomial kernel. Summing a finite number of terms yields a kernel. Kernels are closed under taking limits (since the set of positive semidefinite matrices is closed).

4.3 Three views of kernel methods (Lecture 10)

- We now start laying the mathematical foundation for kernel methods. Specifically, we will develop three views of kernel methods, as illustrated in Figure 8.

  - Feature map ϕ: maps from a data point $x \in \mathcal{X}$ to an element of an inner product space (the feature vector). This allows us to think about properties of single data points.
  - Kernel k: takes two data points $x, x' \in \mathcal{X}$ and returns a real number. This allows us to think about similarity between two data points.
  - RKHS $\mathcal{H}$: a set of functions $f : \mathcal{X} \to \mathbb{R}$ equipped a norm $\| \cdot \|_{\mathcal{H}}$ for measuring the complexity of functions. This allows us to think about the prediction function f itself.

  We will define each of the three views separately, but eventually show that they are all in a sense equivalent.

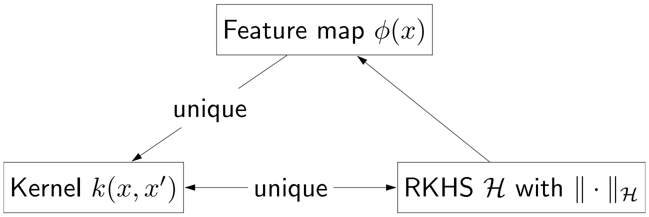

Figure 8: The three key mathematical concepts in kernel methods.

- First, we need a formal notion of infinite feature vectors (the range of a feature map ϕ) that generalizes $\mathbb{R}^d$.

- **Definition 17 (Hilbert space)**

  - A Hilbert space $\mathcal{H}$ is an complete[11] vector space with an inner product $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ that satisfies the following properties:
    - ∗ Symmetry: $\langle f, g \rangle = \langle g, f \rangle$
    - ∗ Linearity: $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle = \alpha_1 \langle f_1, g \rangle + \alpha_2 \langle f_2, g \rangle$
    - ∗ Positive definiteness: $\langle f, f \rangle \ge 0$ with equality only if $f = 0$

    The inner product gives us a norm: $\|f\|_{\mathcal{H}} \stackrel{\text{def}}{=} \sqrt{\langle f, f \rangle}$.
  - Examples

[11] Completeness means that all Cauchy sequences (in which elements get closer and closer to each other) converge to some element in the space (see Definition 31). Examples: set of real numbers is complete, set of rational numbers is not.

- $*$ Euclidean space: $\mathbb{R}^d$, with $\langle u, v\rangle = \sum_{i=1}^d u_i v_i$
- $*$ Square summable sequences: $\ell^2 = \{(u_i)_{i\geq 1} : \sum_{i=1}^\infty u_i^2 < \infty\}$, with $\langle u, v\rangle = \sum_{i=1}^\infty u_i v_i$.
- $*$ Square integrable functions on $[0,1]$: $L^2([0,1]) = \{f : \int_0^1 f(x)^2 dx < \infty\}$, with $\langle f, g\rangle = \int_0^1 f(x)g(x)dx$.[12]

- **Definition 18 (feature map)**
  - Given a Hilbert space $\mathcal{H}$, a **feature map** $\phi : \mathcal{X} \to \mathcal{H}$ takes inputs $x \in \mathcal{X}$ to infinite feature vectors $\phi(x) \in \mathcal{H}$.

- **Theorem 19 (feature map defines a kernel)**
  - Let $\phi : \mathcal{X} \to \mathcal{H}$ be a feature mapping some input space $\mathcal{X}$ to a Hilbert space $\mathcal{H}$.
  - Then $k(x, x') \stackrel{\text{def}}{=} \langle \phi(x), \phi(x')\rangle$ is a kernel.

- Proof:
  - The key idea is that the definition of the kernel only needs to look at n points, which reduces everything to a finite problem.
  - Let $x_1, \dots, x_n$ be a set of points, and let K be the kernel matrix where $K_{ij} = \langle \phi(x_i), \phi(x_j)\rangle$.
  - To show that K is positive semidefinite, take any $\alpha \in \mathbb{R}^n$. We have

$$\alpha^\top K \alpha = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j)\rangle \tag{354}$$

$$= \left\langle \sum_{i=1}^n \alpha_i \phi(x_i), \sum_{i=j}^n \alpha_j \phi(x_j) \right\rangle \tag{355}$$

$$\geq 0, \tag{356}$$

  where we use linearity of the inner product.

- **Theorem 20 (kernel defines feature maps)**
  - For every kernel k (Definition 16), there exists a Hilbert space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x')\rangle$.

[12] Technically, $L^2([0,1])$ is not a vector space since a function f which is non-zero on a measure zero set will still have $\|f\|_{\mathcal{H}} = 0$. But the quotient space (with respect to functions f with $\|f\|_{\mathcal{H}} = 0$) is a vector space.

- We will prove Theorem 20 later since it requires more sophisticated machinery (RKHSes). But to get some intuition, let's prove it for the case where the number of inputs $\mathcal{X}$ is finite.
  - Let $\mathcal{X} = \{x_1, \dots, x_n\}$ and define the kernel matrix K (which defines the entire kernel).
  - Since $K \in \mathbb{R}^{n \times n}$ is positive semidefinite, we can write it as $K = \Phi\Phi^\top$. For example we can take an eigendecomposition $K = UDU^\top$ and let $\Phi = UD^{1/2}$ or the Cholesky decomposition $K = LL^\top$.
  - Let the feature vector $\phi(x_i) \in \mathbb{R}^n$ be the i-th row of Φ. We can verify that $K = \Phi\Phi^\top$ (equivalently, $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$).
  - Note: the feature map is not unique, since we can also define an alternate feature matrix $\Phi' = \Phi Q$ for any orthogonal matrix Q. In this light, kernels are a more "pure" way of defining models, because feature vectors have this rotational indeterminancy.
  - If the input space $\mathcal{X}$ infinite, then we need to generalize our notion of feature vector from $\mathbb{R}^n$ to an infinite dimensional space. What is that space?

4.4 Reproducing kernel Hilbert spaces (RKHS) (Lecture 10)

- We will now introduce RKHSes (a type of Hilbert space), the third view on kernel methods. RKHSes allow us to work directly on the prediction *function*. Concretely, in linear regression, we fit a weight vector w, constraining or regularizing the norm $\|w\|_2$, and we use w to predict on a new input x via $x \mapsto \langle w, \phi(x) \rangle$. But can we get a handle on this prediction function $x \mapsto f(x)$ directly as well as a norm $\|f\|_{\mathcal{H}}$ measuring the complexity of f?

- Our first reaction might be to consider a Hilbert space over functions $f : \mathcal{X} \to \mathbb{R}$. But not all Hilbert spaces are not suitable for machine learning.
  - For example, consider $\mathcal{H} = L^2([0,1])$. Recall that every $f \in \mathcal{H}$ is actually an equivalence class over functions which differ on a measure zero set, which means pointwise evaluations $f(x)$ at individual x's is not even defined.
  - This is highly distressing given that the whole point is to learn an f for the purpose of doing pointwise evaluations (a.k.a. prediction)!
  - RKHSes remedy this problem by making pointwise evaluations really nice, as we'll see.

- **Definition 19 (bounded functional)**
  - Given a Hilbert space $\mathcal{H}$, a functional $L : \mathcal{H} \to \mathbb{R}$ is bounded iff there exists an $M < \infty$ such that

$$|L(f)| \leq M\|f\|_{\mathcal{H}} \text{ for all } f \in \mathcal{H}. \tag{357}$$

  - Example: $\mathcal{H} = \mathbb{R}^d$ with the usual inner product, $L(f) = \langle c, f \rangle$ is bounded (with $M = \|c\|_2$ by Cauchy-Schwartz)

- **Definition 20 (evaluation functional)**
  - Let $\mathcal{H}$ be a Hilbert space consisting of functions $f : \mathcal{X} \to \mathbb{R}$.
  - For each $x \in \mathcal{X}$, define the evaluation functional $L_x : \mathcal{H} \to \mathbb{R}$ as

$$\boxed{L_x(f) \stackrel{\text{def}}{=} f(x).} \tag{358}$$

  - Example: for $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{H} = \{f_c : c \in \mathbb{R}^d\}$ where $f_c(x) = \langle c, x \rangle$ be linear functions, then the evaluation functional is $L_x(f_c) = \langle c, x \rangle$.
  - Intuitively, the evaluation functional is a projection operator that turns a function f into one component $f(x)$. Hence, evaluation functionals are linear. This will be important later.

- **Definition 21 (reproducing kernel Hilbert space)**
  - A reproducing kernel Hilbert space $\mathcal{H}$ is a Hilbert space over functions $f : \mathcal{X} \to \mathbb{R}$ such that for each $x \in \mathcal{X}$, the **evaluation functional** L_x is bounded.

- Non-example
  - Let $\mathcal{H}$ be the set of all square integrable continuous functions from $[0, 1]$ to $\mathbb{R}$.
  - Consider $f_\epsilon(x) = \max(0, 1 - \frac{|x - \frac{1}{2}|}{\epsilon})$, which is zero except for a small spike at $x = \frac{1}{2}$ up to $f(x) = 1$. Note that $\|f_\epsilon\|_{\mathcal{H}} \to 0$ as $\epsilon \to 0$.
  - Consider the evaluation functional $L_{\frac{1}{2}}$. Note that $L_{\frac{1}{2}}(f_\epsilon) = f_\epsilon(\frac{1}{2}) = 1$. So there cannot exist an M such that $L_{\frac{1}{2}}(f_\epsilon) \le M \|f_\epsilon\|_{\mathcal{H}}$ for all $\epsilon > 0$.
  - So this $\mathcal{H}$ is not a RKHS. Note that continuity ensures that functions are actually defined pointwise, not just up to a measure zero set. The problem is that this $\mathcal{H}$ is just too big of a set.

Having defined what an RKHS is, we now show the connection with kernels. In particular:

  - Each RKHS $\mathcal{H}$ is associated with a unique kernel k (Theorem 21)
  - Each kernel k is associated with a unique RKHS $\mathcal{H}$ (Theorem 22)

- **Theorem 21 (RKHS defines a kernel)**
  - Every RKHS $\mathcal{H}$ over functions $f : \mathcal{X} \to \mathbb{R}$ defines a unique kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called the reproducing kernel of $\mathcal{H}$.

- Proof (construction of the kernel)
  - Note that L_x is a *linear* functional: $L_x(cf) = cL_x(f)$ and $L_x(f+g) = L_x(f) + L_x(g)$.
  - The **Riesz representation theorem** states that all *bounded linear functionals* L on a Hilbert space can be expressed as an inner product $L(f) = \langle R, f \rangle$ for a *unique* $R \in \mathcal{H}$.
  - Applying this theorem to the evaluation functionals L_x, we can conclude that for each $x \in \mathcal{X}$, there exists a unique **representer** $R_x \in \mathcal{H}$ such that $L_x(f) = \langle R_x, f \rangle$. Recall that we also have $L_x(f) = f(x)$ by definition. Combining yields the **reproducing property**:

    $$\boxed{f(x) = \langle R_x, f \rangle \text{ for all } f \in \mathcal{H}.} \tag{359}$$

    This is the key property: *function evaluations can be expressed as inner products.*
  - Now let's define a function k:

    $$\boxed{k(x, x') \overset{\text{def}}{=} R_x(x').} \tag{360}$$

    Applying the reproducing property one more time with $f = R_x$ yields

    $$k(x, x') = R_x(x') = \langle R_x, R_{x'} \rangle. \tag{361}$$

    If we define a feature map $\phi(x) \overset{\text{def}}{=} R_x$, we can invoke Theorem 19 to conclude that $k(x, x')$ is a valid kernel.
  - In summary, any RKHS $\mathcal{H}$ gives rise to a kernel k called the **reproducing kernel** of $\mathcal{H}$. The key is the Riesz representation, which turns function evaluations into inner products.

To complete the connection between RKHSes and kernels, we need to show the converse, namely that a kernel defines an unique RKHS:

- **Theorem 22 (Moore-Aronszajn theorem)**
  - For every kernel k, there exists a unique RKHS $\mathcal{H}$ with reproducing kernel k.

- Proof sketch:
  - Let k be a kernel. We will construct a RKHS $\mathcal{H}$ from the functions $\{k(x, \cdot) : x \in \mathcal{X}\}$.

- First, define $\mathcal{H}_0$ to contain all finite linear combinations of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x), \tag{362}$$

  for all $n, \alpha_{1:n}, x_{1:n}$. By construction, $\mathcal{H}_0$ is a vector space (not necessarily complete though). This is a very natural class of functions: just a linear combination of basis functions centered around the points $x_1, \dots, x_n$.

- Second, define the inner product between $f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$ and $g(x) = \sum_{j=1}^{n'} \beta_j k(x'_j, x)$ as follows:

$$\langle f, g \rangle \overset{\text{def}}{=} \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(x_i, x'_j). \tag{363}$$

  Let's check that our definition of $\langle \cdot, \cdot \rangle$ is an actual inner product:

  * Symmetry ($\langle f, g \rangle = \langle g, f \rangle$): by symmetry of k
  * Linearity ($\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle = \alpha_1 \langle f_1, g \rangle + \alpha_2 \langle f_2, g \rangle$): by definition of f and g (they are just a linear sum of terms).
  * Positive definiteness ($\langle f, f \rangle \geq 0$ with equality only if $f = 0$):
    · For any $f \in \mathcal{H}_0$, we have $\langle f, f \rangle = \alpha^\top K \alpha \geq 0$ by the positive semidefinite property of kernels.
    · Now we will show that $\langle f, f \rangle = 0$ implies $f = 0$. Let $f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$. Take any $x \in \mathcal{X}$ and define $c = [k(x_1, x), \dots, k(x_n, x)]^\top$. Since

$$\begin{pmatrix} K & c \\ c^\top & k(x, x) \end{pmatrix} \succeq 0, \tag{364}$$

    we must have that

$$\alpha^\top K \alpha + 2bc^\top \alpha + b^2 k(x, x) \geq 0 \tag{365}$$

    for all b. Note that $\langle f, f \rangle = \alpha^\top K \alpha = 0$. We now argue that $c^\top \alpha = 0$. If $c^\top \alpha > 0$, then as b approaches 0 from the negative side, we have that the LHS of (365) is strictly negative (since the b term will dominate the b^2 term), which is a contradiction. If $c^\top \alpha < 0$, then as b approaches 0 from the positive side, we get a contradiction as well. Therefore, $f(x) = c^\top \alpha = 0$.

- So far we have a valid Hilbert space, but we need to still check that all evaluation functionals L_x are bounded to get an RKHS. Also, we should check that $R_x \overset{\text{def}}{=} k(x, \cdot)$ is indeed a representer of function evaluationals. Take any $f \in \mathcal{H}_0$. Then:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n} \alpha_i k(x_i, x) \quad \text{[definition of } f\text{]} && (366) \\ &= \langle f, k(x, \cdot) \rangle \quad \text{[definition of inner product]} && (367) \\ &= \langle R_x, f \rangle \quad \text{[definition of } R_x\text{]}. && (368) \end{aligned}$$

Boundedness of L_x follows from $|L_x(f)| = |\langle f, k(x,\cdot)\rangle| \le \|f\|_{\mathcal{H}}\|k(x,\cdot)\| = \|f\|_{\mathcal{H}} k(x,x)$ by Cauchy-Schwartz.

  - Finally, let $\mathcal{H}$ be the completion of $\mathcal{H}_0$ (by including all limit points of sequences in $\mathcal{H}_0$). This is the only part of the proof that we're punting on. For details, see the references at the end of this section.
  - This proves Theorem 20 because the RKHS $\mathcal{H}$ is an inner product space by construction.

- Summary
  - A feature map $\phi : \mathcal{X} \to \mathcal{H}$: maps points in $\mathcal{X}$ to some inner product space $\mathcal{H}$.
  - A (positive semidefinite) kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ has the property that every derived kernel matrix K is positive semidefinite.
  - A reproducing kernel Hilbert Space (RKHS) $\mathcal{H}$ containing functions $f : \mathcal{X} \to \mathbb{R}$ such that function evaluations are bounded linear operators.
  - Equivalences
    * $f(x) = \sum_{i=1}^{\infty} \alpha_i k(x_i, x)$, $R_x = k(x,\cdot)$: Moore-Aronszajn establishes connection between kernels and RKHSes
    * $\phi(x) = R_x$: can set feature map (not unique) to map x to the representer of x in the RKHS
    * $k(x, x') = \langle \phi(x), \phi(x')\rangle$: every kernel k corresponds to some inner product (via RKHS) and vice-versa (easy)

[begin lecture 11] (11)

4.5 Learning using kernels (Lecture 11)

- We have established that kernels k provide a space of functions $\mathcal{H}$; this is the hypothesis class.[13] Now let's talk about learning, which is about combining the hypothesis class $\mathcal{H}$ with actual data.

- Let's start with kernelized ridge regression, where we obtain examples $\{(x_i, y_i)\}_{i=1}^n$, and want to find a function $f \in \mathcal{H}$ that fits the data, where $\mathcal{H}$ is an RKHS. A natural objective function is to penalize the squared loss plus a penalty for the complexity of f, where the complexity is the RKHS norm:

$$f^* \in \arg\min_{f \in \mathcal{H}} \sum_{i=1}^n \frac{1}{2}(f(x_i) - y_i)^2 + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2. \tag{369}$$

- More generally, a learning problem can be posed as the following optimization problem:

$$f^* \in \arg\min_{f \in \mathcal{H}} L(\{(x_i, y_i, f(x_i))\}_{i=1}^n) + Q(\|f\|_{\mathcal{H}}^2), \tag{370}$$

  where

  - $L : (\mathcal{X} \times \mathcal{Y} \times \mathbb{R})^n \to \mathbb{R}$ is an arbitrary loss function on n examples.
    - ∗ Example (regression): $L(\{(x_i, y_i, f(x_i))\}_{i=1}^n) = \sum_{i=1}^n \frac{1}{2}(f(x_i) - y_i)^2$.
  - $Q : [0, \infty) \to \mathbb{R}$ is a strictly increasing function (regularizer).
    - ∗ Example (quadratic): $Q(\|f\|_{\mathcal{H}}^2) = \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2$.

  This optimization problem may seem daunting since it is optimizing over a potentially very large function space $\mathcal{H}$. But the following representer theorem reassures us that all minimizers can be written as a linear combination of the kernel functions evaluated at the training points.

- **Theorem 23 (representer theorem)**
  - Let V denote the span of the representers of the training points:

$$V \overset{\text{def}}{=} \text{span}(\{k(x_i, \cdot) : i = 1, \ldots, n\}) = \left\{ \sum_{i=1}^n \alpha_i k(x_i, \cdot) : \alpha \in \mathbb{R}^n \right\}. \tag{371}$$

  - Then all minimizers f^* of (370) satisfy $f^* \in V$.

[13] For regression, our prediction at x is simply $f(x)$, where $f \in \mathcal{H}$. For classification and ranking problems, we need to pass the function values through some non-linear transformation.

- Proof
  - FIGURE: [projection of f^* onto V]
  - The key is to use the fact that an RKHS has an inner product structure, which allows us to use linear algebra.
  - Define the orthogonal complement:

$$V_\perp = \{g \in \mathcal{H} : \langle f, g \rangle = 0 \text{ for all } f \in V\}. \tag{372}$$

  - Any $f \in \mathcal{H}$ can be decomposed in to a part in the span of the examples and an orthogonal part:

$$f^* = f + f_\perp, \tag{373}$$

    where $f \in V$ and $f_\perp \in V_\perp$.
  - The idea is that the loss is unchanged by $f_\perp$ but the regularizer grows with non-zero $f_\perp$, so we must have $f_\perp = 0$.
  - The loss depends on f^* only through $\{f^*(x_j) : j = 1, \dots, n\}$, which can be written as:

$$f^*(x_j) = f(x_j) + \langle f_\perp, k(x_j, \cdot) \rangle . \tag{374}$$

    The second term is zero, so the loss doesn't depend on $f_\perp$.
  - The regularizer:

$$Q(\|f^*\|_{\mathcal{H}}^2) = Q(\|f\|_{\mathcal{H}}^2 + \|f_\perp\|_{\mathcal{H}}^2). \tag{375}$$

    Since Q is strictly monotonic and f^* is a minimizer, we must have $f_\perp = 0$.
  - Therefore, $f^* \in V$.
- Remark: the representer theorem does not require the loss function L to be convex.
- The representer theorem tells us α's exist, but how to find the α's depends on the actual loss function and regularizer. Let's now look at some examples.
- **Example 14 (Kernelized ridge regression)**
  - Recall the optimization problem for regression:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^n \frac{1}{2}(f(x_i) - y_i)^2 + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2. \tag{376}$$

    By the representer theorem, we have the equivalent optimization problem:

$$\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^n \frac{1}{2}\left(\sum_{j=1}^n \alpha_j k(x_i, x_j) - y_i\right)^2 + \frac{\lambda}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j k(x_i, x_j). \tag{377}$$

- Letting $K \in \mathbb{R}^{n \times n}$ be the kernel matrix and $Y \in \mathbb{R}^n$ denote the vector of outputs, we have:

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2}\|K\alpha - Y\|_2^2 + \frac{\lambda}{2}\alpha^\top K\alpha. \tag{378}$$

  Differentiating with respect to α and setting to zero:

$$K(K\alpha - Y) + \lambda K\alpha = 0. \tag{379}$$

  Rearranging:

$$K(K + \lambda I)\alpha = KY. \tag{380}$$

  Solving yields a solution:

$$\boxed{\alpha = (K + \lambda I)^{-1}Y.} \tag{381}$$

  Note that the solution is not necessarily unique, since we could add any vector in the null space of K, but there's no reason to consider them.

- To predict on a new example x, we form kernel evaluations $c \in \mathbb{R}^n$ where $c_i = k(x_i, x)$, and then predict

$$y = c^\top \alpha. \tag{382}$$

- **Example 15 (SVM classification)**

  - This was done in CS229, so we won't go through it again.
  - Primal ($y_i \in \{-1, +1\}$):

$$\min_{f \in \mathcal{H}} \sum_{i=1}^n \max\{0, 1 - y_i f(x_i)\} + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2. \tag{383}$$

  - Dual (define $\tilde{K}_{ij} = y_i y_j K_{ij}$):

$$\min_{\alpha \in \mathbb{R}^n} -\mathbf{1}^\top \alpha + \alpha^\top \tilde{K}\alpha \quad \text{subject to} \quad 0 \le \alpha_i \le \frac{1}{\lambda}, Y^\top \alpha = 0. \tag{384}$$

    The dual is computed by taking the Lagrange dual of the primal optimization problem.

- **Example 16 (Kernel PCA)**

  - Review of PCA

- ∗ Recall in (featurized) PCA, we want to find directions in our data with highest variance.
- ∗ Suppose we have data points $x_1, \dots, x_n$ and a feature map $\phi : \mathcal{X} \to \mathbb{R}^d$.
- ∗ Assume that the data points are centered at zero: $\sum_{i=1}^n \phi(x_i) = 0$.
- ∗ Define the empirical covariance matrix as follows:

$$C \overset{\text{def}}{=} \frac{1}{n} \Phi^\top \Phi, \tag{385}$$

  where the i-th row of Φ is $\phi(x_i)$.

- ∗ Then PCA seeks to find the principal eigenvector of C:

$$Cv = \lambda v. \tag{386}$$

  In practice, take the first r eigenvectors $v_1, \dots, v_r$ (principal components) as an approximation of the entire feature space. Note that the v_i's form an orthonormal basis (have unit norm and are orthogonal).

- ∗ The squared reconstruction error of a new point x is:

$$\left\| \sum_{i=1}^r \langle \phi(x), v_i \rangle v_i - \phi(x) \right\|_2^2. \tag{387}$$

  For example, if we were doing anomaly detection, if a data point x has a large reconstruction error, then x is an anomaly.

– Heuristic derivation of kernel PCA

- ∗ By the representer theorem (or even more simply, by inspecting the form of the covariance matrix), we have $v = \sum_{i=1}^n \alpha_i \phi(x_i) = \Phi^\top \alpha$, so an equivalent characterization is to project the vectors on the data points:

$$\Phi C v = \lambda \Phi v. \tag{388}$$

- ∗ Using the definition of C and the fact that $\Phi v = K\alpha$, we have

$$\frac{1}{n} K^2 \alpha = \lambda K \alpha. \tag{389}$$

  Again, any solution to the following is a valid solution to the above (but we can always add spurious vectors in the null space of K):

$$\boxed{\frac{1}{n} K \alpha = \lambda \alpha.} \tag{390}$$

- ∗ In practice, we compute the eigendecomposition of K and take the first r eigenvectors as the approximation. For simplicity, let's assume we just take one principal component $v \in \mathcal{H}$.

- Computing in infinite dimensions
  * Though the derivation assumed $\phi(x) \in \mathbb{R}^d$, the result is the same for $\phi(x) = k(x, \cdot) \in \mathcal{H}$ in general.
  * We won't go through the details, but the idea is to define a covariance operator (rather than a matrix) $C : \mathcal{H} \to \mathcal{H}$. The intuition is the same.
  * Recall the principal component is now a function $v \in \mathcal{H}$ with $\|v\|_{\mathcal{H}} = 1$,[14] where $v(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$.
  * The squared reconstruction error of a new point x is:

$$\|v(x)v - \phi(x)\|_{\mathcal{H}}^2 = v(x)^2 - 2v(x)^2 + k(x,x) = k(x,x) - v(x)^2. \tag{391}$$

    As expected, all we need are kernel evaluations.
- Interpretation
  * The point of PCA is to reduce the dimensionality of the data, so it might seem strange at first we would want to use kernel PCA to first increase the number of dimensions (possibly to infinity).
  * This is actually okay, because the point of kernels is to reshape the data (in non-linear ways). In doing so, we can expose better directions than those present in the original data.
  * For example, if we use a quadratic kernel, we are saying that we believe the data lies close to a quadratic surface. Of course, with more dimensions, statistical error in the directions could increase.

4.6 Fourier properties of shift-invariant kernels (Lecture 11)

- Having explored how kernels can be used in practice, let us turn back to studying their theoretical properties. Specifically, we will use Fourier analysis to get a handle on what information about the data a kernel is capturing. We will also see that this leads to a new basis representation of kernels. In this section, we will focus on shift-invariant kernels, which are kernels that don't depend on the absolute position of the data points.
- **Definition 22 (shift-invariant kernel)**
  - A kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subseteq \mathbb{R}^b$ is **shift invariant** (a.k.a. stationary, translation invariant) iff k can be written as $k(x, x') = h(x - x')$ for some function h.
  - Example: Gaussian kernel
  - Non-example: linear kernel $((x + a)(x' + a) \neq xx')$

[14] To make v a unit vector, we just rescale v by $\|v(x)\|_{\mathcal{H}}^{-1}$.

- Our goal is to understand shift-invariant kernels in the frequency domain. In particular, it will shed insight into the smoothness properties of the kernel.
- First, let's warmup with some basic facts from Fourier analysis:
  - FIGURE: [complex unit circle]
  - $e^{i\omega t} = \cos(\omega t) + i \sin(\omega t)$
  - $e^{-i\omega t} = \cos(\omega t) - i \sin(\omega t)$
  - $\cos(\omega t) = \frac{1}{2}e^{i\omega t} + \frac{1}{2}e^{-i\omega t}$
  - Here $\omega \in \mathbb{R}$ is the frequency of the sinusoid.
  - Note that everything thus far generalizes to $\omega \in \mathbb{R}^b$ and $t \in \mathbb{R}^b$; just replace ωt with $\langle \omega, t \rangle$.
- Now let's try to construct a kernel using the Fourier basis:
  - First define a feature map for a fixed ω: $\phi(x) = e^{-i\langle \omega, x \rangle}$ (note that this is complex-valued, but that's okay).
  - Using this feature map, let's define a kernel:

$$k(x, x') = \phi(x)\overline{\phi(x')} = e^{-i\langle \omega, x - x' \rangle}, \tag{392}$$

    where $\bar{a}$ denotes the complex conjugate of a. This kernel deems two points to be similar if they are close modulo $2\pi/\omega$ (for some fixed scalar frequency ω); clearly this is a bit silly.
  - To get more realistic kernels, we need to incorporate multiple frequencies. We can do this by simply averaging over multiple kernels (recall that the sum of kernels is a kernel). Specifically, let $\mu(\cdot)$ be a finite non-negative measure over frequencies. Then

$$k(x, x') = \int e^{-i\langle \omega, x - x' \rangle} \mu(d\omega). \tag{393}$$

    is also a valid kernel. Or in terms of h (recall that $t = x - x'$):

$$h(t) = \int e^{-i\langle \omega, t \rangle} \mu(d\omega). \tag{394}$$

  - The corresponding feature map consists of the following basis functions: $\{x \mapsto e^{-i\langle \omega, x \rangle} : \omega \in \mathbb{R}^b\}$
  - Intuitively μ tells us how much focus to put on various frequencies.
- **Example 17 (constant)**
  - Let the spectral measure $\mu = \delta_0$ place all its mass at 0.

- Then $k(x, x') = h(t) = 1$ is the constant kernel.

- **Example 18 (single frequency)**
  - Suppose μ places mass only at $-\omega$ and ω:

$$\mu = \frac{1}{2}(\delta_{-\omega} + \delta_{\omega}). \tag{395}$$

  - Then the resulting kernel is:

$$k(x, x') = h(t) = \cos(\omega t). \tag{396}$$

  - In general, $h(t)$ defined via μ might be complex-valued, but if μ is **symmetric** (that is, $\mu(A) = \mu(-A)$ for all measurable sets A), then $h(t)$ will be **real-valued**.

- **Example 19 (sinc)**
  - Let the spectral density s be the 1 over $[-a, a]$ (that is, we only keep frequencies below a):

$$s(\omega) = \mathbb{I}[-a \le \omega \le a]. \tag{397}$$

  - Then the resulting kernel is:

$$h(t) = \frac{2\sin(at)}{t}. \tag{398}$$

  - Proof: just integrate:

$$h(t) = \int_{-a}^{a} e^{-i\omega t} d\omega = \frac{1}{-it}(e^{-iat} - e^{iat}) = \frac{1}{-it}(-2i\sin(at)). \tag{399}$$

  - Note: as a increases, we cover more frequencies. If $a \to \infty$, then $h(t)$ converges to a delta function at 0, which corresponds to the degenerate kernel $k(x, x) = \mathbb{I}[x = x']$.

- At first, it might seem that this is a funny way of defining certain types of kernels, but what's remarkable is that *all* shift-invariant kernels can be written in the form (393) for some appropriate choice of μ. This statement is precisely Bochner's theorem:

- **Theorem 24 (Bochner's theorem)**
  - Let $k(x, x') = h(x - x')$ be a continuous shift-invariant kernel ($x \in \mathbb{R}^b$).

- Then there exists a unique finite non-negative measure μ (called the **spectral measure**) on $\mathbb{R}^b$ such that

$$\boxed{h(t) = \int e^{-i\langle t,\omega\rangle}\mu(d\omega).} \tag{400}$$

- Furthermore, if μ has a density s,

$$\mu(d\omega) = s(\omega)d\omega, \tag{401}$$

then we call s the **spectral density**, and h is the Fourier transform of s.

- So far, we've defined kernels through various spectral measures μ. But we can also take a given kernel and compute its spectral measure to study the properties of the kernel. Given a candidate kernel function $k(x, x') = h(x - x')$, take the Fourier transform of h (which for symmetric functions is the inverse Fourier transform times $1/(2\pi)$) to get the spectral density s.

- **Example 20 (box is not a kernel)**

  - Consider the following function:

$$h(t) = \mathbb{I}[-1 \le t \le 1]. \tag{402}$$

  - The inverse Fourier transform times $1/(2\pi)$ is

$$s(\omega) = \frac{\sin(\omega)}{\pi\omega}, \tag{403}$$

reusing the result from above.

  - But notice that $s(\omega)$ is negative in some places, which means, by Bochner's theorem, that $h(t)$ is not a valid (positive semidefinite) kernel! Now we have another way to check whether a shift-invariant function specifies a kernel—simply take the inverse Fourier transform and see whether it's non-negative everywhere.

- **Example 21 (Gaussian kernel)**

  - Let the spectral density s be the density of the multivariate Gaussian distribution with variance $1/\sigma^2$:

$$s(\omega) = \left(\frac{2\pi}{\sigma^2}\right)^{-d/2} \exp\left(\frac{-\sigma^2\|\omega\|_2^2}{2}\right). \tag{404}$$

- Then the resulting kernel is the Gaussian kernel with variance σ^2 (note the inverting of the variance):

$$h(t) = \exp\left(\frac{-\|t\|_2^2}{2\sigma^2}\right). \tag{405}$$

- Proof:

$$h(t) = \int \left(\frac{2\pi}{\sigma^2}\right)^{-d/2} \exp\left(\frac{(-\sigma^2\|\omega\|_2^2 - 2i\langle\omega, t\rangle - \sigma^{-2}i^2\|t\|_2^2) + \sigma^{-2}i^2\|t\|_2^2}{2}\right) d\omega. \tag{406}$$

  Complete the square and note that the Gaussian distribution (with mean it/σ) integrates to 1.

- Intuition: the larger σ^2 is, one can see from $s(\omega)$ that high frequency components are dampened. Consequently, the smoother the kernel $h(t)$ is.

- **Example 22 (rational quadratic kernel)**

  - Motivation: with Gaussian kernels, how do we set the variance σ^2?
  - Putting on our Bayesian hats, let's define a prior over $\tau = \sigma^{-2}$.
  - Let $k_\tau(x, x')$ be the Gaussian kernel with hyperparameter τ.
  - Recalling that the sum of kernels is a kernel, we have that

$$\int k_\tau(x, x')p(\tau)d\tau \tag{407}$$

    is also a kernel for any $p(\tau)$.
  - For mathematical convenience, let's put a Gamma(α, β) prior on τ.
  - By conjugacy, we can integrate a Gamma distribution against a Gaussian, which yields a student-t distribution.
  - Ignoring normalization constants, the kernel is the **rational quadratic kernel** (derivation omitted):

$$h(t) = \left(1 + \frac{\beta t^2}{2\alpha}\right)^{-\alpha}. \tag{408}$$

  - When $\alpha = 1$ and $\beta = 2$, we have:

$$h(t) = \frac{1}{1 + t^2}. \tag{409}$$

  - Compared with the Gaussian kernel:

    - The area near zero is steeper, so the function can change rapidly.
    - The tails decay slower, function values can have longer range dependencies.

  This flexbility comes from integrating over values of σ^2.

- Note that as $\alpha, \beta \to \infty$ with $\alpha = \beta$, the rational quadratic approaches the Gaussian kernel with $\sigma^2 = 1$.

[begin lecture 12] (12)

4.7 Efficient computation (Lecture 12)

- We saw how kernel methods could be used for learning: Given n points $x_1, \dots, x_n$, we form the $n \times n$ kernel matrix K, and optimize an objective function whose variables are $\alpha \in \mathbb{R}^n$. For example, in kernel ridge regression, we have $\alpha = (K + \lambda I)^{-1}Y$, which requires $O(n^3)$ time. When we have large datasets (e.g., $n = 10^6$), this is prohibitively expensive. On the other hand, when the feature vector $\phi(x)$ is high-dimensional (especially infinite-dimensional), then scaling with n is the lesser of two evils. But can we do better? Note that even merely computing the full kernel matrix K takes $O(n^2)$ time, which is already too large, so we will probably have to cut some corners.

- We will introduce two types of kernel approximations:

  - Random features: We will write the kernel function as an integral, and using Monte Carlo approximations of this integral. These approximations are of the kernel function and are data-independent.
  - Nyström method: We will sample a subset of the n points and use these points to approximate the kernel matrix. These approximations are of the kernel matrix and are data-dependent.

- Gaussian kernel intuitions

  - Let's get some intuition about when approximations might be a sensible thing to do on a concrete scenario. Recall the Gaussian kernel:

$$k(x, x') = \exp\left(\frac{-\|x - x'\|_2^2}{2\sigma^2}\right). \tag{410}$$

  - If the points $x_1, \dots, x_n$ are far apart (relative to the bandwidth of σ^2), then the kernel matrix K will be roughly the identity matrix. In this case, we can't possibly hope for a low-rank approximation to capture everything.
  - On the other hand, if the points are tightly clustered into m clusters, then the kernel matrix (with sorted columns/rows) looks like a block diagonal matrix with m blocks, where each block is a rank 1 all-ones matrix. Here, you would expect a rank m approximation to be effective.
  - In reality, the situation is somewhere in between. Of course, kernel methods are exactly useful when the data are fairly complex, so we shouldn't expect these approximations to provide magical savings, unless the data is very redundant.

- Random Fourier features (Rahimi/Recht, 2008)

- Our starting point is Bochner's theorem (Theorem 24), which allows us to write shift-invariant kernels in terms of an integral over some spectral measure μ:

$$k(x, x') = \int \phi_\omega(x)\overline{\phi_\omega(x')}\mu(d\omega), \tag{411}$$

  where $\phi_\omega(x) = e^{-i\langle \omega, x\rangle}$ is a single (Fourier) feature.

- The key idea is to replace the integral with a finite sum over m elements. For simplicity, assume that μ is a probability distribution. If it is not, then we can normalize it and then multiply the result by $\mu(\mathbb{C}^b)$. Let $\omega_1, \dots, \omega_m$ be drawn i.i.d. from μ. Then, define the approximate kernel as:

$$\hat{k}(x, x') = \frac{1}{m}\sum_{i=1}^m \phi_{\omega_i}(x)\overline{\phi_{\omega_i}(x)}. \tag{412}$$

  This kernel corresponds to having the following random feature map:

$$\hat{\phi}(x) = \frac{1}{\sqrt{m}}[\phi_{\omega_1}(x), \dots, \phi_{\omega_m}(x)] \in \mathbb{C}^m. \tag{413}$$

- As a concrete example, consider the Gaussian kernel, which has a Gaussian spectral density (recall $\mu(d\omega) = s(\omega)d\omega$) with the inverse variance:

$$k(x, x') = \exp\left(\frac{-\|x - x'\|_2^2}{2\sigma^2}\right), \tag{414}$$

$$s(\omega) = \left(\frac{2\pi}{\sigma^2}\right)^{-b/2} \exp\left(\frac{-\sigma^2\|\omega\|_2^2}{2}\right). \tag{415}$$

  This means that each $\omega_i \sim \mathcal{N}(0, \sigma^{-2}I)$ is drawn from a Gaussian.

- Algorithm
  - ∗ The practical upshot of random Fourier features on the Gaussian kernel is that it is dirt simple.
  - ∗ Before you get data, draw $\omega_1, \dots, \omega_m \sim \mathcal{N}(0, \sigma^{-2}I)$, which defines the random feature map $\hat{\phi}$. This feature map is fixed once and for all.
  - ∗ In training/test, given a new data point x, we can apply the feature map $\hat{\phi}(x)$, which simply involves m Gaussian projections.

- Note that the approximate kernel is unbiased ($\mathbb{E}[\hat{k}(x, x')] = k(x, x')$), so as $m \to \infty$, we have that $\hat{k}(x, x')$ converges to $k(x, x')$ for a fixed x, x'. We want this to work well on average for all the data that we're going to see, which smells almost like uniform convergence. The following theorem quantifies this:

- **Theorem 25 (random features (Rahimi/Recht, 2008))**
  - ∗ Let k be a shift-invariant kernel on $x \in \mathbb{R}^b$.

  - ∗ Let

    $$\mathcal{F} \overset{\text{def}}{=} \left\{ x \mapsto \int \alpha(\omega)\phi_\omega(x)\mu(d\omega) : \forall\omega, |\alpha(\omega)| \le C \right\} \tag{416}$$

    be the subset of functions in the RKHS $\mathcal{H}$ with bounded Fourier coefficients $\alpha(\omega)$.
  - ∗ Let

    $$\hat{\mathcal{F}} \overset{\text{def}}{=} \left\{ x \mapsto \frac{1}{m}\sum_{i=1}^{m} \alpha(\omega_i)\phi_{\omega_i}(x) : \forall\omega, |\alpha(\omega)| \le C \right\} \tag{417}$$

    be the subset that is spanned by the random feature functions, where $\omega_{1:k}$ be drawn i.i.d. from μ.
  - ∗ Let p^* be any distribution over $\mathcal{X} = \mathbb{R}^b$.
  - ∗ Define the inner product with respect to the data-generating distribution (this is not the RKHS inner product):

    $$\langle f, g \rangle \overset{\text{def}}{=} \mathbb{E}_{x \sim p^*}[f(x)g(x)]. \tag{418}$$

  - ∗ Let $f^* \in \mathcal{F}$ be any true function.
  - ∗ Then with probability at least $1 - \delta$, there exists $\hat{f} \in \hat{\mathcal{F}}$ that

    $$\|\hat{f} - f^*\| \le \frac{C}{\sqrt{m}}\left(1 + \sqrt{2\log(1/\delta)}\right). \tag{419}$$

- Proof of Theorem 25:
  - ∗ This proof uses fairly standard tools: McDiarmid's inequality and Jensen's inequality. The function we're applying involves taking a norm of a function, but we just need the bounded differences condition to hold.
  - ∗ Fix $f^* \in \mathcal{F}$ with coefficients $\alpha(\omega)$.
  - ∗ Construct $\hat{f}$ with the same coefficients, and note that $\hat{f} \in \hat{\mathcal{F}}$ and $\mathbb{E}[\hat{f}] = f^*$.
  - ∗ Define

    $$D(\omega_{1:m}) = \|\hat{f} - f^*\|. \tag{420}$$

    Note that D satisfies the bounded differences inequality: letting $\omega^i_{1:m} = \omega_{1:m}$ except on the i-th component, where it is ω'_i:

    $$\begin{aligned} D(\omega_{1:m}) - D(\omega^i_{1:m}) &\le \|\hat{f} - f^*\| - \|\hat{f}^i - f^*\| && (421) \\ &\le \|\hat{f} - \hat{f}^i\| \quad \text{[triangle inequality]} && (422) \\ &\le \frac{1}{m}\|\alpha(\omega_i)\phi_{\omega_i} - \alpha(\omega'_i)\phi_{\omega'_i}\| && (423) \\ &\le \frac{2C}{m}. && (424) \end{aligned}$$

    Note that the last line follows because $|\alpha(\omega_i)| \le C$ and $\phi_{\omega_i}(x) = e^{-i\langle\omega_i, x\rangle}$ and $|e^{-ia}| = 1$ for all a.

- We can bound the mean by passing to the variance:

$$\mathbb{E}[D(\omega_{1:m})] \le \sqrt{\mathbb{E}[D(\omega_{1:m})^2]} \quad \text{[Jensen's inequality]} \tag{425}$$

$$= \sqrt{\mathbb{E}\left[\left\|\frac{1}{m}\sum_{i=1}^{m}(\alpha(\omega_i)\phi_{\omega_i} - f^*)\right\|^2\right]} \quad \text{[expand]} \tag{426}$$

$$= \sqrt{\frac{1}{m^2}\sum_{i=1}^{m}\mathbb{E}\left[\|\alpha(\omega_i)\phi_{\omega_i} - f^*\|^2\right]} \quad \text{[variance of i.i.d. sum]} \tag{427}$$

$$\le \frac{C}{\sqrt{m}} \quad \text{[use } |\alpha(\omega_i)| \le C\text{]}. \tag{428}$$

- Applying McDiarmid's inequality (Theorem 8), we get that

$$\mathbb{P}\left[D(\omega_{1:m}) \ge \frac{C}{\sqrt{m}} + \epsilon\right] \le \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{m}(2C/m)^2}\right). \tag{429}$$

Rearranging yields the theorem.

- Remark: the definition of α here differs from the Rahimi/Recht paper.
- Corollary:
  - Suppose we had a loss function $\ell(y, v)$ which is 1-Lipschitz in the second argument. (e.g., the hinge loss). Define the expected risk in the usual way:

$$L(f) \stackrel{\text{def}}{=} \mathbb{E}_{(x,y)\sim p^*}[\ell(y, f(x))]. \tag{430}$$

Then the approximation ratio is bounded:

$$L(\hat{f}) - L(f^*) \le \mathbb{E}[|\ell(y, \hat{f}(x)) - \ell(y, f^*(x))|] \quad \text{[definition, add } |\cdot|\text{]} \tag{431}$$

$$\le \mathbb{E}[|\hat{f}(x) - f^*(x)|] \quad \text{[fix } y,\ \ell \text{ is Lipschitz]} \tag{432}$$

$$\le \|\hat{f} - f^*\| \quad \text{[concavity of } \sqrt{\cdot}\text{]}. \tag{433}$$

- So far, we have analyzed approximation error due to having a finite m, but assuming an infinite amount of data. Separately, there is the estimation error due to having n data points:

$$L(\hat{f}_{\text{ERM}}) - L(\hat{f}) \le O_p\left(\frac{C}{\sqrt{n}}\right), \tag{434}$$

where $\hat{f}_{\text{ERM}}$ minimzes the empirical risk over the random hypothesis class $\hat{\mathcal{F}}$. So, the total error, which includes approximation error and estimation error is

$$L(\hat{f}_{\text{ERM}}) - L(f^*) = O_p\left(\frac{C}{\sqrt{n}} + \frac{C}{\sqrt{m}}\right). \tag{435}$$

This bound suggests that the approximation and estimation errors are balanced when m and n are on the same order. One takeaway is that we shouldn't over-optimize one without the other. But one might also strongly object and say that if $m \approxeq n$, then we aren't really getting any savings! This is indeed a valid complaint, and in order to get stronger results, we would need to impose more structure on the problem.

- Dot product kernels (Kar/Karnick, 2012)
  - We have seen that shift-invariant kernels admit an integral representation, which allows us to use Monte Carlo to approximate it. What about non-shift invariant kernels such as polynomial kernels, such as the following?

    $$k(x, x') = \langle x, x' \rangle^p. \tag{436}$$

  - Although random Fourier features will not work, we can still try to write the kernel as an expectation. The key is that if we draw a Rademacher variable $\omega \in \{-1, +1\}^b$ (uniform), randomly projecting x onto ω yields an unbiased estimate of the inner product:

    $$\langle x, x' \rangle = \mathbb{E}[\langle \omega, x \rangle \langle \omega, x' \rangle]. \tag{437}$$

    Of course, this isn't useful by itself, but it does reduce x to a scalar $\langle \omega, x \rangle$, which is useful.

  - To generalize to polynomial kernels, we simply do the above construction p times and multiply it all together. For the quadratic kernel, let ω_1 and ω_2 be two independent Rademacher vectors. Then:

    $$\langle x, x' \rangle^2 = \mathbb{E}[\langle \omega_1, x \rangle \langle \omega_1, x' \rangle]\mathbb{E}[\langle \omega_2, x \rangle \langle \omega_2, x' \rangle] \tag{438}$$

    $$= \mathbb{E}[\langle \omega_1, x \rangle \langle \omega_2, x \rangle \langle \omega_1, x' \rangle \langle \omega_2, x' \rangle], \tag{439}$$

    where the first line follows from the earlier calculation, and the second line follows from independence of ω_1 and ω_2. Note that $\langle \omega_1, x \rangle \langle \omega_2, x \rangle$ is still just a number.

  - More generally, if the kernel is a analytic function of the dot product, then it admits the following Taylor expansion around 0:

    $$k(x, x') = f(\langle x, x' \rangle), \quad f(z) = \sum_{j=0}^{\infty} a_j z^j. \tag{440}$$

  - To construct a random feature,
    - ∗ Choose J with probability proportional to a_j (if we can't sample from a_j exactly, then we can use importance weighting).

    - Choose $\omega_1, \dots, \omega_J$ Rademacher vectors, and let

$$\phi_{\omega_{1:J}}(x) = \prod_{j=1}^{J} \langle \omega_j, x \rangle . \tag{441}$$

  - If we do this m times to form a m-dimensional feature vector, then we have a Monte Carlo estimate of the kernel k. Note that in the process, we have to draw an expected $mb\mathbb{E}[J]$ Rademacher variables.
  - At this point, we have only showed that we have an unbiased estimate of the kernel. We still need to show that the variance isn't too large. See the paper in the references below for that.

- Arc-cosine kernel (Cho/Saul, 2009)
  - The random features perspective is very suggestive of the computation in neural networks. A one layer neural network computes a function

$$f(x) = \sum_{j=1}^{m} \alpha_j \sigma(\omega_j \cdot x), \tag{442}$$

    where the parameters $\alpha_{1:m}$ and $\omega_{1:m}$ are optimized via (stochastic) gradient descent (backpropagation), and σ is a non-linear function such as a hard-threshold ($\sigma(z) = \mathbb{I}[z \geq 0]$) or rectified linear unit ($\sigma(z) = \mathbb{I}[z \geq 0]z$).
  - For comparison, the random features view of kernels defines a function class where $\omega_{1:m}$ is chosen randomly rather than optimized, while $\alpha_{1:m}$ are optimized. Typically, neural network weights are initialized randomly, sometimes according to a Gaussian with the appropriate variance. The fact that random features approximates a kernel suggests that even the random initialization is quite sensible starting point (provided the final layer $\alpha_{1:m}$ are optimized).
  - What if we take σ to be a rectified linear unit and let the number of hidden units $m \to \infty$? Does this limiting quantity have a nice form? To answer this question, let us define a more general family.
  - Define the random basis function:

$$\phi_\omega(x) = \mathbb{I}[\omega \cdot x \geq 0](\omega \cdot x)^q. \tag{443}$$

    - For $q = 0$, we obtain the threshold function.
    - For $q = 1$, we obtain the rectified linear unit (ReLU).
  - As $m \to \infty$, we obtain the following kernel:

$$k(x, x') = 2 \int \phi_\omega(x) \phi_\omega(x') p(\omega) \omega. \tag{444}$$

- This kernel can be shown to have the following closed form solution:

$$k(x, x') = \frac{1}{\pi} \|x\|^q \|x'\|^q J_q(\theta), \tag{445}$$

where

$$\theta = \arccos\left(\frac{x \cdot x'}{\|x\| \|x'\|} \right) \tag{446}$$

is the angle between x and x' and $J_q(\theta)$ captures the angular dependence. In other words, this *arc-cosine kernel* decouples the magnitude from the angle.

- In general J_q is a complex function, but we can consider two simple cases:

$$J_0(\theta) = \pi - \theta \tag{447}$$

$$J_1(\theta) = \sin(\theta) + (\pi - \theta)\cos(\theta). \tag{448}$$

- The punchline is that even if we are using neural networks, we can use kernel methods to better understand them from a representational point of view.

- Nyström method (Williams/Seeger, 2000)

  - In the above, we have constructed random features, which were independent of the data. A technique that predates these, which can work better when the spectrum of the kernel matrix is to form a low-rank approximation. This method applies more generically to approximating large PSD matrices.

  - Given a kernel matrix $K \in \mathbb{R}^{n \times n}$, we will sample a subset of the indices $I \subseteq \{1, \dots, n\}$ with $|I| = m$, and let $J = \{1, \dots, n\} \backslash I$ be the other indices. We then evaluate the kernel on points in I paired with all other points, for a total of $O(|I|n)$ evaluations. Then we can define the approximate kernel matrix:

$$K = \begin{pmatrix} K_{II} & K_{IJ} \\ K_{JI} & K_{JJ} \end{pmatrix} \approxeq \begin{pmatrix} K_{II} & K_{IJ} \\ K_{JI} & K_{JI} K_{II}^{\dagger} K_{IJ}, \end{pmatrix} = \tilde{K} \tag{449}$$

  or more compactly:

$$\tilde{K} \stackrel{\text{def}}{=} K_{\cdot I} K_{II}^{\dagger} K_{I \cdot}. \tag{450}$$

  Note that the difference $K_{JJ} - K_{JI} K_{II}^{\dagger} K_{IJ}$ is the Schur complement of K_{JJ}. If we interpret K as a covariance matrix of a Gaussian Z, then this is the conditional variance $\mathrm{Var}[Z_J \mid Z_I]$.

  - Note that if K is rank m and and K_{II} also contains linearly independent columns (so that it captures the subspace of K), then the Schur complement is zero, and the Nyström method is exact. If not, then the error stems from simply not being able to capture the low rank solution by having a rank m matrix plus an error from doing column sampling (which doesn't yield the eigenvectors). We can think of this as projecting the kernel matrix K on to the subspace of the data points in I.

- How do we choose I? Two popular choices are uniform sampling and sampling proportional to $K_{ii} = k(x_i, x_i)$, which corresponds to the squared magnitude of x_i. Intuitively, the weighted sampling focuses more energy on points which are more important.
- The following theorem formalizes the error bound:
- **Theorem 26 (Nyström with non-uniform sampling (Drineas/Mahoney, 2005))**
  - Suppose we choose I by sampling (with replacement), choosing $i \in \{1, \dots, n\}$ with probability $K_{ii} / \sum_{j=1}^n K_{jj}$.
  - Let $\tilde{K}_m$ be the best rank m approximation of K.
  - Let $\tilde{K}$ be defined as in (449), but where we replace K_{II} with the best rank m approximation of K_{II}.
  - Then with probability at least $1 - \delta$,

$$\|K - \tilde{K}\|_F^2 \le \|K - \tilde{K}_m\|_F^2 + 4(1 + \sqrt{8 \log(1/\delta)}) \operatorname{tr}(K)^2 \sqrt{\frac{m}{|I|}}. \tag{451}$$

- Proof: follows from algebraic manipulation and concentration. Note that the theorem statement is a correct version of Drineas and Mahoney's Theorem 3, where we just combined equation 31 with Lemma 9.
- The theorem suggests that we should take $|I| > m$, which gives us more opportunities to cover the column space of $\tilde{K}_m$.

4.8 Universality (skipped in class)

- We have explored several different kernels, and we can (and should) certainly choose one based on domain knowledge.

- But one can ask: is there a general purpose kernel k, in the sense that k can be used to solve *any* learning problem given sufficient data? The notion of general purpose is defined as follows.

- **Definition 23 (universal kernel)**
  - Let $\mathcal{X}$ be a locally compact Hausdorff space (e.g., $\mathbb{R}^b$ or any discrete set, but not infinite-dimensional spaces in general).
  - Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel.
  - We say that k is a **universal kernel** (specifically, a c_0-universal kernel) iff the RKHS $\mathcal{H}$ with reproducing kernel k is dense in $C_0(\mathcal{X})$, the set of all continuous bounded functions on $\mathcal{X}$ (with respect to the uniform norm). In other words, for any function $f \in C_0(\mathcal{X})$ and $\epsilon > 0$, there exists some $g \in \mathcal{H}$ such that $\sup_{x \in \mathcal{X}} \|f - g\| \le \epsilon$.

- The premise is that the target function we want to learn is in $C_0(\mathcal{X})$, so by using a universal kernel, we are defining an RKHS which can approximate any function in $C_0(\mathcal{X})$ as well as we want.

- The following theorem characterizes universal kernels in terms of their Fourier properties:

- **Theorem 27 (Carmeli, 2010)**

  - Let k be a shift-invariant kernel with spectral measure μ on $\mathcal{X} = \mathbb{R}^d$.
  - If the support of μ is all of $\mathbb{R}^b$, then k is a universal kernel.

  Intuition: in order to represent any $C_0(\mathcal{X})$ function, we must not have any gaps in our spectrum.

- Example: the Gaussian kernel is universal; the sinc kernel is not universal.

- The final piece of the puzzle is **universal consistency**, which means that that a learning algorithm will actually achieve the best possible error as the number of training examples tends to infinity. Steinwart showed that using SVMs with a universal kernel guarantees universal consistency. Of course, universality is only about how well we can represent the target function; it says nothing about readily we can actually estimate that function based on finite data.

4.9 RKHS embedding of probability distributions (skipped in class)

- So far, we've showed that kernels can be used for estimating functions for regression, classification, dimensionality reduction (PCA), etc. Now we will show how kernels can be used to represent and answer questions about probability distributions without having to explicitly estimate them.

- As a motivating example, consider the problem of testing whether two probability distributions P and Q are the same by only observing expectations under the distributions.

- Given a distribution P, we can look at various **moments** of the distribution $\mathbb{E}_{x \sim P}[f(x)]$ for various functions f. For example, if $f(x) = x$, then we get the mean. Such a f only gives us partial information about P: if two distributions differ in their mean, then we know they are different, but if they have the same mean, we cannot conclude that they are same.

- More generally, assume P and Q are defined on some locally compact Hausdorff space $\mathcal{X}$ (e.g., $\mathbb{R}^b$). Define the **maximum mean discrepancy** (MMD) as follows:

$$\boxed{D(P, Q, \mathcal{F}) \stackrel{\text{def}}{=} \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{x \sim Q}[f(x)]\right),} \tag{452}$$

for some set of functions $\mathcal{F}$. Shorthand: $\mathbb{E}_P[f]$ means $\mathbb{E}_{x\sim P}[f(x)]$.

- Can we find $\mathcal{F}$ so that

$$D(P, Q, \mathcal{F}) = 0 \Leftrightarrow P = Q? \tag{453}$$

  Note that $P = Q$ always implies $D(P, Q, \mathcal{F})$, but the other direction requires some work.

- If we knew P and Q were Gaussian, then it suffices to take $\mathcal{F} = \{x \mapsto x, x \mapsto x^2\}$, since the first two moments define a Gaussian distribution. However, what about general P and Q? We need a much larger class of functions $\mathcal{F}$:

- **Theorem 28 (Dudley, 1984)**

  - If $\mathcal{F} = C_0(\mathcal{X})$ (all continuous bounded functions), then $D(P, Q, \mathcal{F}) = 0$ implies $P = Q$.

- However, $C_0(\mathcal{X})$ is a large and difficult set to work with. Fortunately, it suffices to take $\mathcal{F}$ to be any set that is dense in $C_0(\mathcal{X})$, in particular an RKHS:

- **Theorem 29 (Steinwart, 2001)**

  - Let $\mathcal{F} = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \le 1\}$, where $\mathcal{H}$ is the RKHS defined by a universal kernel k.
  - Then $D(P, Q, \mathcal{F}) = 0$ implies $P = Q$.

- Proof:

  - Let $P \neq Q$ be two distinct distributions.
  - Then there exists $f \in C_0(\mathcal{X})$ such that $|\mathbb{E}_P[f] - \mathbb{E}_Q[f]| = \epsilon > 0$.
  - Since $\mathcal{H}$ is universal (i.e., $\mathcal{H}$ is dense in $\mathcal{C}_0(\mathcal{X})$ with respect to the uniform norm), there exists $g \in \mathcal{H}$ with $g \neq 0$ such that

$$\|f - g\|_\infty \overset{\text{def}}{=} \sup_{x\in\mathcal{X}} |f(x) - g(x)| \le \epsilon/3. \tag{454}$$

  - This means $|\mathbb{E}_P[f] - \mathbb{E}_P[g]| \le \epsilon/3$ and $|\mathbb{E}_Q[f] - \mathbb{E}_Q[g]| \le \epsilon/3$.
  - By the triangle inequality, $|\mathbb{E}_P[g] - \mathbb{E}_Q[g]| \ge \epsilon/3 > 0$.
  - Rescale g: let $u = g/\|g\|_{\mathcal{H}} \in \mathcal{F}$.
  - We still have $D(P, Q, \mathcal{F}) \ge |\mathbb{E}_P[u] - \mathbb{E}_Q[u]| > 0$.

- Computing $D(P, Q, \mathcal{F})$

- We've established that $D(P, Q, \mathcal{F})$ contains sufficient information for testing whether two distributions are equal. But how do we actually compute the max over $\mathcal{F}$? This seems daunting at first sight. Fortunately, we can compute $D(P, Q, \mathcal{F})$ in closed form by exploiting properties of the RKHS.
- First, a general statement. By the reproducing property and linearity of the inner product, we can express expected function value as an inner product:

$$\mathbb{E}_{x\sim P}[f(x)] = \mathbb{E}_{x\sim P}[\langle k(\cdot, x), f\rangle] = \left\langle \underbrace{\mathbb{E}_{x\sim P}[k(x,\cdot)]}_{\overset{\text{def}}{=}\mu_P}, f \right\rangle. \tag{455}$$

  Here, $\mu_P \in \mathcal{H}$ is the **RKHS embedding** of the probability distribution P.
- We can now write the MMD solution as follows:

$$D(P, Q, \mathcal{F}) = \sup_{f\in\mathcal{F}} \langle \mu_P - \mu_Q, f\rangle = \|\mu_P - \mu_Q\|_{\mathcal{H}}, \tag{456}$$

  where the sup is obtained by setting f to be a unit vector in the direction of $\mu_P - \mu_Q$.
- Unpacking the square of the last expression and rewriting in terms of kernel evaluations:

$$\|\mu_P - \mu_Q\|_{\mathcal{H}}^2 = \mathbb{E}_{P\times P}[k(x, x')] - \mathbb{E}_{P\times Q}[k(x, y)] - \mathbb{E}_{Q\times P}[k(y, x)] + \mathbb{E}_{Q\times Q}[k(y, y')]. \tag{457}$$

- Of course, in practice, we only have samples from P, Q: let $x_1, \dots, x_n \sim P$ and $y_1, \dots, y_n \sim Q$ all be drawn independently.

- We can obtain an empirical estimate of $D(P, Q, \mathcal{F})$ as a U-statistic (a U-statistic is a function which is an average over some function applied to all pairs of points):

$$\hat{D}_n(P, Q, \mathcal{F}) = \frac{1}{\binom{n}{2}} \sum_{i<j} [k(x_i, x_j) - k(x_i, y_j) - k(y_i, x_j) + k(y_i, y_j)]. \tag{458}$$

  This estimate is unbiased, since the expectation of each term is $D(P, Q, \mathcal{F})$.

- Let the null hypothesis be that $P = Q$. Under the null hypothesis, as $n \to \infty$, we know that $\hat{D}_n(P, Q, \mathcal{F}) \xrightarrow{P} 0$, but in order to use $\hat{D}_n$ as a test statistic for hypothesis testing, we need to know its (approximate) distribution. (Recall that $\hat{D}_n(P, Q, \mathcal{F})$ is a random variable that is a function of the data points.)

- There are two ways to go about this:
  - We can derive finite sample complexity bounds to bound the deviation of $\hat{D}(P, Q, \mathcal{F})$ from its mean $D(P, Q, \mathcal{F})$.
  - We can show that $\hat{D}(P, Q, \mathcal{F})$ is asymptotically normal with some variance, and use the normal as an approximation of the distribution.

- In the next section, we will develop the tools to analyze random variables such as these.

4.10 Summary (Lecture 12)

- We began by noting that some algorithms (e.g., gradient descent) do not require arbitrary inner products between weight vectors and feature vectors, but only **inner products between feature vectors**.
- This motivated the use of **kernels** (defined to be positive semidefinite functions), which can provide both computational (ability to implicitly compute inner products between infinite-dimensional feature vectors) and modeling advantages (thinking in terms of similarities between two inputs).
- Taking a step back, we saw that all that matters at the end of the day are functions evaluated at various inputs. This motivated the definition of **reproducing kernel Hilbert spaces** (RKHS), in which two important properties hold: (i) function evaluations were bounded (meaningful), and (ii) there is a nice inner product structure.
- We showed that the three distinct viewpoints above (features, kernels, functions) are actually all equivalent (due to the Moore-Aronszajn theorem).
- The **representer theorem** shows that the optimum over an appropriately regularized function space $\mathcal{H}$ is attained by a function in the span of the training data. This allows us to derive kernelized SVMs, kernelized regression, kernel PCA, RKHS embeddings of probability distributions.
- **Bochner's theorem**, allows us to study the Fourier properties of shift-invariant kernels, relating universality and smoothness properties of a kernel to the frequencies that the kernel passes through.
- Bochner's theorem allowed us to obtain computationally efficient ways to approximate kernel methods by writing kernels as an integral over dot products of **random features**. This leads to efficient algorithms. but have uncertainty estimates over function values. Uncertainty estimates are critical for active learning and Bayesian optimization.

4.11 References

- Hofmann/Scholkopf/Smola, 2008: Kernel Methods in Machine Learning
- Drineas/Mahoney, 2005: On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning
- Rahimi/Recht, 2008: Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning
- Yang/Li/Mahdavi/Jin/Zhou, 2012: Nystrom Method vs Random Fourier Features: A Theoretical and Empirical Comparison
- Kar/Karnick, 2012: Random Feature Maps for Dot Product Kernels

[begin lecture 13] (13)

5 Online learning

5.1 Introduction (Lecture 13)

- Thus far, we have analyzed algorithms (maximum likelihood, ERM) in the *statistical setting*, where we assume the the training and test data are both drawn i.i.d. from some distribution p^*. We even boasted that we need not make any assumption about what p^* is. In this unit, we will weaken the assumptions even more and assume that data can be generated completely *adversarily*. In addition, we will move to the online setting where training and test are interleaved. Thus we make two shifts to the learning setup:

  – Batch to online

  – Statistical to adversarial

- We will first discuss the online learning framework, focusing on prediction. Then, we will cast online learning as online convex optimization and develop several algorithms and prove regret bounds for these algorithms. Finally, we will look at multi-armed bandit problems, where the learner obtains partial feedback. Throughout this section, there will be very little probability (since we will be working in the adversarial setting), but we will draw quite a bit from convex analysis.

- Framework

  – Prediction task: we want to map inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$.

  – The online learning setting can be thought of as the following game between a learner and nature:

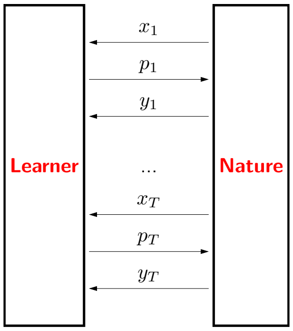

Figure 9: Online learning game.

∗ Iterate $t = 1, \dots, T$:

    - · Learner receives input $x_t \in \mathcal{X}$
    - · Learner outputs prediction $p_t \in \mathcal{Y}$
    - · Learner receives true label $y_t \in \mathcal{Y}$
    - · (Learner suffers loss $\ell(y_t, p_t)$)
    - · (Learner updates model parameters)
  - More formally, the learner is a function $\mathcal{A}$ that returns the current prediction given the full history:[15]

$$p_{t+1} = \mathcal{A}(x_{1:t}, p_{1:t}, y_{1:t}, x_{t+1}). \tag{459}$$

  Nature can be defined similarly.

- **Example 23 (online binary classification for spam filtering)**
  - Inputs: $\mathcal{X} = \{0, 1\}^d$ are boolean feature vectors (presence or absence of a word).
  - Outputs: $\mathcal{Y} = \{+1, -1\}$: whether a document is spam or not spam.
  - Zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$ is whether the prediction was incorrect.

- Remarks
  - The typical training phase (setting model parameters) and testing phase (making predictions for evaluation) are **interleaved** in online learning.
  - Note that the the online learning setting leaves completely open the time and memory usage of the algorithms that operate in this setting. Technically, we could just train an SVM on all the data that we've seen so far, and predict on the next example. However, the spirit of online learning suggests that the amount of work an algorithm does per iteration should not grow with t. In practice, online learning algorithms update parameters after each example, and hence tend to be **faster** than traditional batch optimization algorithms such as Newton's method.
  - The real world is complex and constantly-changing, but online learning algorithms have the potential to **adapt** (although we will not analyze their adapative capabilities in this course).
  - In some applications such as spam filtering, the inputs could be generated by an **adversary**. In our analyses, we will make no assumptions about the input/output sequence.

- Evaluation
  - Now comes the most important part: How we measure the quality of an online learner $\mathcal{A}$? While seemingly simple and obvious, how we answer this question has a *defining impact* on the behavior of $\mathcal{A}$.

[15] For now, assume deterministic algorithms. Later, we'll consider stochastic algorithms, which will be important against adversaries.

- The first attempt is to just write down the cumulative loss of the learner:

  $$\sum_{t=1}^{T} \ell(y_t, p_t). \tag{460}$$

  However, if we are in the adversarial setting, no algorithm can do better than the maximum regret T (for the zero-one loss) since the adversary can always choose $y_t \neq p_t$. What do you do when your performance is awful? You show that others are doing even worse than you.

- Less flippantly, let $\mathcal{H}$ be a set of experts, where each **expert** is a function $h : \mathcal{X} \to \mathcal{Y}$ that predicts $h(x)$ on input $x \in \mathcal{X}$. Note that the learner can adapt (change over time) whereas the experts are fixed.

- **Definition 24 (regret)**

  * The regret of a learner with respect to an expert h is the cumulative difference between the loss incurred by the learner and the loss incurred by expert h:

    $$\text{Regret}(h) \overset{\text{def}}{=} \sum_{t=1}^{T} [\ell(y_t, p_t) - \ell(y_t, h(x_t))]. \tag{461}$$

    Note that Regret(h) depends on $\mathcal{H}$, T, the sequences $x_{1:T}, y_{1:T}$, and of course the algorithm $\mathcal{A}$ itself; but we're eliding the dependence on all of these things in the notation.

  * The regret with respect to a class of experts $\mathcal{H}$ is the maximum regret

    $$\text{Regret} \overset{\text{def}}{=} \max_{h \in \mathcal{H}} \text{Regret}(h). \tag{462}$$

    Equivalently, we can write:

    $$\text{Regret} = \underbrace{\sum_{t=1}^{T} \ell(y_t, p_t)}_{\text{learner}} - \underbrace{\min_{h \in \mathcal{H}} \sum_{t=1}^{T} \ell(y_t, h(x_t))}_{\text{best expert}}. \tag{463}$$

- The best expert is a role model for the learner. While it is technically possible for the learner to do better than all the experts since it can adapt over time (leading to negative regret), this generally won't happen, and therefore, we will be content with trying to achieve small positive regret.

- We are interested in particular in the maximum possible regret over all sequences $x_{1:T}, y_{1:T}$; in more colorful language, an adversary chooses the inputs and outputs as to maximize regret.

- Having a comparison set of experts gives us some hope to compete against an adversary. Intuitively, if the adversary tries to screw over the learner, it will probably screw over the experts too.

- In the next few lectures, we will prove results of the following form for various online learning algorithms $\mathcal{A}$: for all $\mathcal{H}, T, x_{1:T}, y_{1:T}$, we have:

$$\text{Regret} \le \text{SomeFunction}(\mathcal{H}, T). \tag{464}$$

  Usually, we want the regret to be sublinear in T, which means that the average regret per example goes to zero.

5.2 Warm-up (Lecture 13)

We will give two examples, one where online learning is impossible and one where it is possible.

- **Example 24 (negative result)**
  - Assume binary classification: $y \in \{-1, +1\}$
  - Assume zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$
  - Assume the learner is fully **deterministic**.
  - Claim: for all learners $\mathcal{A}$, there exists an $\mathcal{H}$ and input/output sequence $x_{1:T}, y_{1:T}$ such that:

$$\boxed{\text{Regret} \ge T/2} \quad \text{[awful!]} \tag{465}$$

  - **Key point**: adversary (having full knowledge of learner) can choose y_t to be always different from p_t.
  - Learner's cumulative loss: T (make mistake on every example).
  - We're not done yet, because remember regret is the difference between learner's cumulative loss and the best expert's, so we have to check how well the experts do in this case.
  - Consider two experts, $\mathcal{H} = \{h_{-1}, h_{+1}\}$, where h_y always predicts y.
  - The sum of the cumulative losses of the experts equals T because $\ell(y_t, h_{-1}(x_t)) + \ell(y_t, h_{+1}(x_t)) = 1$, so one of the experts must achieve loss $\le T/2$.
  - Therefore, the difference between T and $\le T/2$ is $\ge T/2$.
  - It is perhaps not surprising that no learner can do well because nature is too powerful here (it can choose any sequence with full knowledge of what the learner will do). Not even measuring regret with respect to only two experts is enough. So we will need assumptions to get positive results.
- **Example 25 (positive result (learning with expert advice))**
  - Assume zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$.

- Clearly, we need to make *some* assumptions to make progress. Here, we will make a fairly strong assumption just to get some intuition.
- **Assumption 3 (realizable)**

  Assume the best expert $h^* \in \mathcal{H}$ obtains zero cumulative loss ($\ell(y_t, h^*(x_t)) = 0$ for all $t = 1, \dots, T$); or equivalently, since we're dealing with zero-one loss, $y_t = h^*(x_t)$ for all t. This assumption places a joint restriction on $\mathcal{H}, x_{1:T}, y_{1:T}$.
- We will design an algorithm that queries the experts on each example x_t, and try to combine their predictions in some way; this setting is called **learning with expert advice**.
- **Algorithm 1 (majority algorithm)**
  - Maintain a set $V_t \subseteq \mathcal{H}$ of valid experts (those compatible with the first $t-1$ examples).
  - On each iteration t, predict p_t to be the majority vote over predictions of valid experts $\{h(x_t) : h \in V_t\}$.
  - Keep experts which were correct: $V_{t+1} = \{h \in V_t : y_t = h(x_t)\}$.
- Analysis:
  - On each mistake, at least half of the experts are eliminated.
  - So $1 \le |V_{T+1}| \le |\mathcal{H}|2^{-M}$, where M is the number of mistakes (also equal to Regret since the best expert has zero loss).
  - Note that the lower bound is due to the realizability assumption.
  - M is the exactly the regret here.
  - Take logs and rearrange:

$$\boxed{\text{Regret} \le \log_2 |\mathcal{H}|.} \tag{466}$$

- Notes:
  - This is a really strong result: note that the regret is constant; after a finite number of iterations, we cease to make any more mistakes forever.
  - However, realizability (that some expert is perfect) is too strong of an assumption.
  - Also, the regret bound is useless here if there are an infinite number of experts ($|\mathcal{H}| = \infty$).

5.3 Online convex optimization (Lecture 13)

- In order to obtain more general results, we move to a framework called online convex optimization. **Convexity** will give us considerable leverage. Afterwards, we'll connect online convex optimization with online learning.

- Let $S \subseteq \mathbb{R}^d$ be a convex set (e.g., representing the set of allowed weight vectors).

  - Example: $S = \{u : \|u\|_2 \le B\}$.

- FIGURE: [convex function with subgradients]

- **Definition 25 (convexity)**

  A function $f : S \to \mathbb{R}$ is convex iff for all points $w \in S$, there is some vector $z \in \mathbb{R}^d$ such that

$$\boxed{f(u) \ge f(w) + z \cdot (u - w) \quad \text{for all } u \in S.} \tag{467}$$

  This says that at any point $w \in S$, we can find a linear approximation (RHS of (467)) that lower bounds the function f (LHS of (467)).

- **Definition 26 (subgradient)**

  For each $w \in S$, the set of all z satisfying (467) are known as the subgradients at w:

$$\boxed{\partial f(w) \overset{\text{def}}{=} \{z : f(u) \ge f(w) + z \cdot (u - w) \text{ for all } u \in S\}.} \tag{468}$$

  If f is differentiable at w, then there is one subgradient equal to the gradient: $\partial f(w) = \{\nabla f(w)\}$.

- Convexity is remarkable because it allows you to say something (in particular, lower bound) the global behavior (for all $u \in S$) of a function f by something local and simple (a linear function). An important consequence of f being convex is that if $0 \in \partial f(w)$, then w is a global minimum of f.

- Checking convexity

  - How do you know if a function is convex? You can try to work it out from the definition or if the function is twice-differentiable, compute the Hessian and show that it's positive semidefinite.
  - But often, we can check convexity by decomposing the function using a few rules. This is by no means an exhaustive list, but these are essentially all the important cases that we'll need in this class.
    * Linear functions: $f(w) = w \cdot z$ for any $z \in \mathbb{R}^d$
    * Quadratic functions: $f(w) = w^\top A w$ for positive semidefinite $A \in \mathbb{R}^{d \times d}$
    * Negative entropy: $f(w) = \sum_{i=1}^d w_i \log w_i$ on $w \in \Delta_d$.
    * Sum: $f + g$ if f and g are convex
    * Scale: cf where $c \ge 0$ and f is convex
    * Supremum: $\sup_{f \in \mathcal{F}} f$, where $\mathcal{F}$ is a family of convex functions
  - Example: hinge loss (skipped in class)
    * Function: $f(w) = \max\{0, 1 - y(w \cdot x)\}$.

    - ∗ Subgradient:
        - · $\partial f(w) = \{0\}$ if $y(w \cdot x) > 1$ (w achieves margin at least 1)
        - · $\partial f(w) = \{-yx\}$ if $y(w \cdot x) < 1$
        - · $\partial f(w) = \{-yx\alpha : \alpha \in [0, 1]\}$ if $y(w \cdot x) = 1$

- The setup for online convex optimization is similar to online learning:

  - Iterate $t = 1, \dots, T$:
    - ∗ Learner chooses $w_t \in S$
    - ∗ Nature chooses convex loss function $f_t : S \to \mathbb{R}$

    Formally, the learner $\mathcal{A}$ chooses w_{t+1} depending on the past:

    $$w_{t+1} = \mathcal{A}(w_{1:t}, f_{1:t}). \tag{469}$$

  - Regret is defined in the way you would expect (cumulative difference of losses):

    $$\text{Regret}(u) \overset{\text{def}}{=} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)]. \tag{470}$$

    $$\text{Regret} \overset{\text{def}}{=} \max_{u \in S} \text{Regret}(u). \tag{471}$$

  - The set S plays two roles: it is the set of experts with which we define our regret, and it is also the set of parameters that our learner is going to consider. For simplicity, we assume these two are the same, although in general, they do not have to be.

- We now turn to our original goal of doing online learning. We will show some examples of reducing online learning to online convex optimization. In particular, given an input/output sequence input $x_{1:T}, y_{1:T}$, we will construct a sequence of convex functions $f_{1:T}$ such that the low regret incurred by a learner on the online convex optimization problem implies low regret on the online learning problem. Let OL be the online learner who has access to OCO, a online convex optimizer.

  - **Example 26 (linear regression)**
    - ∗ FIGURE: [two boxes, OL and OCO with arrows between]
    - ∗ Assume we are doing linear regression with the squared loss, $\ell(y_t, p_t) = (p_t - y_t)^2$.
    - ∗ On each iteration $t = 1, \dots, T$:
      - · OL receives an input $x_t \in \mathbb{R}^d$ from nature.
      - · OL asks OCO for a weight vector $w_t \in \mathbb{R}^d$.
      - · OL sends the prediction $p_t = w_t \cdot x_t$ to nature.
      - · OL receives the true output y_t from nature.

  - OL relays the feedback to OCO via the loss function

$$f_t(w) = (w \cdot x_t - y_t)^2. \tag{472}$$

Note that (x_t, y_t) is baked into f_t, which changes each iteration. Since the squared loss is convex, f_t is convex. One can check easily based on matching definitions that the regret of OCO is exactly the same as regret of OL.

- **Example 27 (learning with expert advice)**
  - Now let's turn to a problem where the convexity structure isn't as apparent.
  - Assume we have a finite number (this is important) of experts $\mathcal{H}$, which the learner can query. Let

$$\mathcal{H} = \{h_1, \dots, h_d\}. \tag{473}$$

  - Assume we are doing binary classification with the zero-one loss (this is not important—any bounded loss function will do), $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$.
  - Note that neither the loss function nor the set of experts $\mathcal{H}$ is convex; in fact, this doesn't really make sense, since the domains are discrete.
  - Nonetheless, we can convexify the problem using **randomization**. Specifically, we allow the learner to produce a probability distribution $w_t \in \Delta_d$ ($\Delta_d \subseteq \mathbb{R}^d$ is the $(d-1)$-dimensional simplex)[16] over the d experts $\mathcal{H}$, sample an expert from this distribution, and predict according to that expert. The expected zero-one loss is then a convex (in fact, linear) function of w_t.
  - Formally, the reduction is as follows:
    - OL receives an input x_t from nature.
    - OL asks OCO for a weight vector $w_t \in \Delta_d$, which represents a distribution over d experts.
    - OL samples an expert $j_t \sim w_t$ and send prediction $p_t = h_{j_t}(x_t)$ to nature.
    - OL receives the true output y_t from nature.
    - OL relays the feedback to OCO via the loss function

$$f_t(w) = w \cdot z_t, \tag{474}$$

where z_t is the vector of losses incurred by each expert:

$$z_t = [\ell(y_t, h_1(x_t)), \dots, \ell(y_t, h_d(x_t))] \in \{0, 1\}^d. \tag{475}$$

Again, f_t bakes the loss and the data into the same function, and one can check that the expected regret of OL is the same as the regret of OCO. Note that we assume no structure on x_t and y_t—they could be arbitrarily complicated; we are only exposed to them via the the experts and the loss function.

[16] Formally, think of a probability distribution over a random variable $J \in \{1, \dots, d\}$, with $\mathbb{P}[J = j] = w_j$.

- ∗ This convexify-by-randomization trick applies more generally to any loss function and any output space $\mathcal{Y}$. The key is that the set of experts is finite, and the learner is just choosing a convex combination of those experts.
- ∗ Yet another way to convexify non-convex losses without randomization is to use an upper bound (e.g., hinge loss or logistic loss upper bounds the zero-one loss). However, minimizing the convex upper bound does not guarantee minimizing the zero-one loss.

5.4 Follow the leader (FTL) (Lecture 13)

We first start out with a natural algorithm called follow the leader (FTL), which in some sense is the analog of the majority algorithm for online learning. We'll see that it works for quadratic functions but fails for linear functions. This will give us intuition about how to fix up our algorithm.

- **Algorithm 2 (follow the leader (FTL))**
  - Let $f_1, \dots, f_T$ be the sequence of loss functions played by nature.
  - The learner chooses the weight vector $w_t \in S$ that minimizes the cumulative loss so far on the previous $t-1$ iterations:

$$\boxed{w_t \in \arg\min_{w \in S} \sum_{i=1}^{t-1} f_i(w).} \tag{476}$$

    (If there are multiple minima, choose any one of them. This is not important.)
  - Aside: solving this optimization problem at each iteration is expensive in general (which would seem to destroy the spirit of online learning), but we'll consider special cases with analytic solutions.
  - Note: We can think of FTL as an empirical risk minimizer where the training set is the first $t-1$ examples.

- We want to now study the regret of FTL. Regret compares the learner (FTL) with any expert $u \in S$, but this difference can be hard to reason about. So to get started, we will use the following result (Lemma 7) to replace u in the bound by (i) something easier to compare w_t to and (ii) at least as good as any $u \in S$.

- **Lemma 7 (compare FTL with one-step lookahead cheater)**
  - Let $f_1, \dots, f_T$ be any sequence of loss functions.
  - Let $w_1, \dots, w_T$ be produced by FTL according to (476). For any $u \in S$:

$$\boxed{\text{Regret}(u) \overset{\text{def}}{=} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)] \le \sum_{t=1}^{T} [f_t(w_t) - f_t(w_{t+1})].} \tag{477}$$

  - Note: This is saying our regret against the best fixed u is no worse than comparing against the one-step lookahead w_{t+1} that peeks at the current function f_t (which is cheating!).
  - Note: The RHS terms $f_t(w_t) - f_t(w_{t+1})$ measure how *stable* the algorithm is; smaller is better. Stability is an important intuition to develop.

- Proof of Lemma 7:
  - Subtracting $\sum_t f_t(w_t)$ from both sides, it suffices to show

$$\sum_{t=1}^{T} f_t(w_{t+1}) \le \sum_{t=1}^{T} f_t(u) \tag{478}$$

    for all $u \in S$. Intuitively, this says that w_{t+1} (which takes the minimum over the first t functions) is better than using a fixed u for all time.
  - Proof by induction:
    * Assume the inductive hypothesis on $T - 1$:

$$\sum_{t=1}^{T-1} f_t(w_{t+1}) \le \sum_{t=1}^{T-1} f_t(u) \quad \text{for all } u \in S. \tag{479}$$

    * Add $f_T(w_{T+1})$ to both sides:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \le \sum_{t=1}^{T-1} f_t(u) + f_T(w_{T+1}) \text{ for all } u \in S. \tag{480}$$

    * In particular, this holds for $u = w_{T+1}$, so we have:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \le \sum_{t=1}^{T} f_t(w_{T+1}). \tag{481}$$

    * Since $w_{T+1} \in \arg\min_u \sum_{t=1}^{T} f_t(u)$ by definition of FTL, we have:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \le \sum_{t=1}^{T} f_t(u) \text{ for all } u \in S, \tag{482}$$

      which is the inductive hypothesis for T.

- Note that Lemma 7 doesn't actually require on convexity of f_t, but rather only stability of the iterates $\{w_t\}$ as measured by f_t. As we'll see later, strong convexity is the main way we will achieve this stability. For now, let's consider two examples to gain some intuition: one where the $\{w_t\}$ are stable (Example 28) and one where they are not (Example 29).

- **Example 28 (quadratic optimization: FTL works)**
  - Assume nature always chooses quadratic functions:

$$f_t(w) = \frac{1}{2}\|w - z_t\|_2^2, \tag{483}$$

    where the points are bounded: $\|z_t\|_2 \le L$ for all $t = 1, \dots, T$.

  - FTL (minimizing over $S = \mathbb{R}^d$) has a closed form solution, which is just the average of the previous points:

$$w_t = \frac{1}{t-1}\sum_{i=1}^{t-1} z_i. \tag{484}$$

  - Bound one term of the RHS of Lemma 7 (intuitively the difference is only one term):

$$\begin{aligned} f_t(w_t) - f_t(w_{t+1}) &= \frac{1}{2}\|w_t - z_t\|_2^2 - \frac{1}{2}\|(1-1/t)w_t + (1/t)z_t - z_t\|_2^2 && (485)\\ &= \frac{1}{2}(1 - (1-1/t)^2)\|w_t - z_t\|_2^2 && (486)\\ &\le (1/t)\|w_t - z_t\|_2^2 && (487)\\ &\le (1/t)4L^2. && (488) \end{aligned}$$

  - Side calculation: summing $1/t$ yields $\log T$:

$$\sum_{t=1}^{T}(1/t) \le 1 + \int_1^T (1/t)dt = \log(T) + 1. \tag{489}$$

  - Summing over t yields:

$$\boxed{\text{Regret} \le \sum_{t=1}^{T}[f_t(w_t) - f_t(w_{t+1})] \le 4L^2(\log(T) + 1).} \tag{490}$$

  - The important thing here is that the difference between w_t and w_{t+1} (measured in terms of loss) is really small (only $1/t$), which means that FTL for quadratic functions is really stable. This makes sense because averages are stable: adding an extra data point should only affect the running average by $O(1/t)$.

- **Example 29 (linear optimization: FTL fails)**
  - We will construct an evil example to make FTL fail.

- Let $S = [-1, 1]$ be FTL's possible predictions. This is a nice bounded one-dimensional convex set, so we're not even trying hard to be pathological.
- Consider linear functions $f_t(w) = wz_t$ in $d = 1$ dimension, where

$$(z_1, z_2, \dots) = (-0.5, 1, -1, 1, -1, 1, -1, \dots). \tag{491}$$

- The minimizer computed by FTL will be attained at an extreme point, causing oscillating behavior.

$$(w_1, w_2, \dots) = (0, 1, -1, 1, -1, 1, -1, 1, \dots). \tag{492}$$

- FTL obtains $T - 1$ cumulative loss (get loss 1 on every single example except the first).
- Expert $u = 0$ obtains 0 cumulative loss (not necessarily even the best).
- Therefore, the regret is pretty depressing:

$$\boxed{\text{Regret} \ge T - 1}. \tag{493}$$

- What's the lesson?
  - For these quadratic functions, w_t and w_{t+1} must get closer (low regret).
  - For these linear functions, w_t and w_{t+1} do not get closer (high regret).
  - It seems then that FTL works when functions f_t offer "stability" (e.g., quadratic) but fail when they do not (e.g., linear).
  - We will reveal the more general principle (strong convexity) later work.

[begin lecture 14] (14)

5.5 Follow the regularized leader (FTRL) (Lecture 14)

- It would be nice if nature just handed us quadratic-looking f_t's, but in general, we're not the ones calling the shots there. But we do control the learner, so the key idea is to *add some regularization* of our own to stablize the learner.

- **Algorithm 3 (follow the regularized leader (FTRL))**
  - Let $\psi : S \to \mathbb{R}$ be a function called a **regularizer** (this defines the learner).
  - Let $f_1, \dots, f_T$ be the sequence of loss functions played by nature.
  - On iteration t, the learner chooses the weight vector that minimizes the regularizer plus the losses on the first $t-1$ examples:

$$\boxed{w_t \in \arg\min_{w \in S} \psi(w) + \sum_{i=1}^{t-1} f_i(w).} \tag{494}$$

  - Note: FTL is just FTRL with $\psi = 0$.

- Quadratic ψ, linear f_t
  - For the remainder of the section, just to build the right intuition in a transparent way, let's specialize to quadratic regularizers ψ and linear loss functions f_t:

$$\psi(w) = \frac{1}{2\eta}\|w\|_2^2, \qquad f_t(w) = w \cdot z_t. \tag{495}$$

  - Then the FTRL optimization problem (494) is:

$$w_t = \arg\min_{w \in S} \left\{ \frac{1}{2\eta}\|w\|_2^2 - w \cdot \theta_t \right\}, \tag{496}$$

    where

$$\theta_t = -\sum_{i=1}^{t-1} z_i \tag{497}$$

    is the negative sum of the gradients z_t. Interpret θ_t as the direction that we want to move in to reduce the loss, but now, unlike in FTL, we're held back by the quadratic regularizer.

- If $S = \mathbb{R}^d$, then FTRL has a closed form solution:

$$w_t = \eta\theta_t, \tag{498}$$

  a scaled down version of θ_t. We can write $w_t = -\eta z_1 - \eta z_2 - \cdots - \eta z_{t-1}$ and equivalently think of the weights as being updated incrementally according to the following recurrence:

$$\boxed{w_{t+1} = w_t - \eta z_t.} \tag{499}$$

  From this perspective, the recurrence in (499) looks awfully like an online subgradient update where z_t is the gradient and η is the step size.

- If $S \neq \mathbb{R}^d$, then FTRL requires a projection onto S. We rewrite (496) by completing the square (add $\frac{\eta}{2}\|\theta_t\|_2^2$):

$$\boxed{w_t \in \arg\min_{w \in S} \frac{1}{2\eta}\|w - \eta\theta_t\|_2^2 = \Pi_S(\eta\theta_t),} \tag{500}$$

  which is a Euclidean projection of $\eta\theta_t$ onto set S. This is called a **lazy projection** since θ_t still accumulates unprojected gradients, and we only project when we need to obtain a weight vector w_t for prediction. This is also known as Nesterov's **dual averaging** algorithm.

- Regularizers in online and batch learning

  - It's worth pausing to examine the difference in the way regularization enters online learning versus batch learning, with which you're probably more familiar.
  - In batch learning (e.g., in training an SVM or ridge regression), one typically seeks to optimize a function which is the sum of the training loss plus the regularizer. Notably, the regularizer is part of the objective function.
  - In online learning here, our objective in some sense is the regret, which makes no mention of the regularizer. The regularizer lies purely inside the learner's head, and is used to defines the updates. In our example so far, the regularizer (in the context of FTRL) gives birth to the online gradient algorithm in (499).

- Now we will analyze the regret FTRL for quadratic regularizers and linear losses.

- **Theorem 30 (regret of FTRL)**

  - Let $S \subseteq \mathbb{R}^d$ be a convex set (of weight vectors).
  - Let $f_1, \dots, f_T$ be any sequence of linear loss functions: $f_t(w) = w \cdot z_t$ for some $z_t \in \mathbb{R}^d$.
  - Let $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ be a quadratic regularizer for any $\eta > 0$.

- Then the regret of FTRL (as defined in Algorithm 3) with respect to any $u \in S$ is as follows:

$$\boxed{\text{Regret}(u) \le \frac{1}{2\eta}\|u\|_2^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|z_t\|_2^2.} \tag{501}$$

- Interpretation: the step size η allows us to trade off two terms:
  - First term: "squared bias" due to regularization. To compete with u, the learner's iterates w_t must somehow get close to u. Smaller η means more regularization (remember $w_t = \eta\theta_t$), which means it's harder to get there.
  - Second term: "variance" due to changing z_t. A smaller η means that successive weight vectors are closer and thus stabler (recall $w_{t+1} - w_t = -\eta z_t$). With a small η, we can hope to avoid the bad scenario in Example 29.
- FIGURE: [ellipse with u at edge, draw iterates w_t trying to reach u]
- Corollary:
  - To make the bound look cleaner:
    - * Let $\|z_t\|_2 \le L$.
    - * Let $\|u\|_2 \le B$ for all experts $u \in S$.

    Now (501) can be rewritten as:

$$\text{Regret}(u) \le \frac{B^2}{2\eta} + \frac{\eta T L^2}{2}. \tag{502}$$

  - Side calculation:
    - * Suppose we want to minimize some function with the following form: $C(\eta) = a/\eta + b\eta$.
    - * Take the derivative: $-a/\eta^2 + b = 0$, resulting in $\eta = \sqrt{a/b}$ and $C(\eta) = 2\sqrt{ab}$, which is just twice the geometric average of a and b.
  - Letting $a = B^2/2$ and $b = TL^2/2$, we get $\eta = \frac{B}{L\sqrt{T}}$ and

$$\boxed{\text{Regret} \le BL\sqrt{T}.} \tag{503}$$

    Note that the average regret goes to zero as desired, even though not as fast as for quadratic functions ($\log T$).
- Proof of weakened version of Theorem 30

- We will prove Theorem 30 later using Bregman divergences, but just to give some intuition without requiring too much technical machinery, we will instead prove a slightly weaker result (note that the second term is looser by a factor of 2):

$$\boxed{\text{Regret}(u) \le \frac{1}{2\eta}\|u\|_2^2 + \eta \sum_{t=1}^{T} \|z_t\|_2^2.} \tag{504}$$

- The key idea is to reduce FTRL to FTL. Observe that FTRL is the same as FTL where the first function is the regularizer.

- Let us then apply Lemma 7 to the sequence of functions $\psi, f_1, \dots, f_T$ (when applying the theorem, note that the indices are shifted by 1). This results in the bound:

$$[\psi(w_0) - \psi(u)] + \sum_{t=1}^{T}[f_t(w_t) - f_t(u)] \le [\psi(w_0) - \psi(w_1)] + \sum_{t=1}^{T}[f_t(w_t) - f_t(w_{t+1})]. \tag{505}$$

- Canceling $\psi(w_0)$, noting that $\psi(w_1) \ge 0$, and rearranging, we get:

$$\text{Regret}(u) \overset{\text{def}}{=} \sum_{t=1}^{T}[f_t(w_t) - f_t(u)] \le \frac{1}{2\eta}\|u\|_2^2 + \sum_{t=1}^{T}[f_t(w_t) - f_t(w_{t+1})]. \tag{506}$$

- Now let's bound one term of the RHS sum:

$$\begin{aligned}
f_t(w_t) - f_t(w_{t+1}) &= z_t \cdot (w_t - w_{t+1}) \quad [\text{since } f_t(w) = w \cdot z_t] && (507)\\
&\le \|z_t\|_2 \|w_t - w_{t+1}\|_2 \quad [\text{Cauchy-Schwartz}] && (508)\\
&= \|z_t\|_2 \|\Pi_S(\eta\theta_t) - \Pi_S(\eta\theta_{t+1})\|_2 \quad [\text{since } w_t = \Pi_S(\eta\theta_t)] && (509)\\
&\le \|z_t\|_2 \|\eta\theta_t - \eta\theta_{t+1}\|_2 \quad [\text{projection decreases distance}] && (510)\\
&= \eta\|z_t\|_2^2 \quad [\text{since } \theta_{t+1} = \theta_t - z_t]. && (511)
\end{aligned}$$

  Plugging this bound back into (506) completes the proof.

5.6 Online subgradient descent (OGD) (Lecture 14)

- So far, we have proven regret bounds for FTRL with linear loss functions. However, many loss functions we care about in practice (e.g., squared loss, hinge loss) are not linear.

- Even if did derive a regret for FTRL with more general losses, there would still be a computational problem: FTRL (see (494)) requires minimizing over all the loss functions seen so far, which in general is computationally impractical, especially for an online learning algorithm. For linear loss functions, on the other hand, we could optimize w_t easily by maintaining θ_t as a "sufficient statistics" (by linearity).

- Our strategy to handling general losses efficiently is by appealing to the linear machinery we already have. The key idea is to *run FTRL on a linear approximation of* f_t.

- What linear approximation $w \mapsto w \cdot z_t$ should we use? Let's use the subgradient of f_t at the current weights w_t: take any $z_t \in \partial f_t(w_t)$. Just to highlight the simplicity of the algorithm, here it is:

- **Algorithm 4 (Online subgradient descent (OGD))**

  - Let $w_1 = 0$.
  - For iteration $t = 1, \dots, T$:
    - ∗ Predict w_t and receive f_t.
    - ∗ Take any subgradient $z_t \in \partial f_t(w_t)$.
    - ∗ If $S = \mathbb{R}^d$, perform the update:

      $$w_{t+1} = w_t - \eta z_t. \tag{512}$$

    - ∗ If $S \subseteq \mathbb{R}^d$, project cumulative gradients onto S:

      $$w_{t+1} = \Pi_S(\eta\theta_{t+1}), \quad \theta_{t+1} = \theta_t - z_t. \tag{513}$$

- To emphasize: OGD on f_t is nothing more than FTRL on quadratic regularizers and linear subgradient approximations of f_t.

- Analyzing regret

  - From our earlier analysis of FTRL (Theorem 30), we already have a bound on the regret on the linearized losses:

    $$\sum_{t=1}^{T} [w_t \cdot z_t - u \cdot z_t]. \tag{514}$$

  - We are interested on controlling the actual regret with respect to f_t:

    $$\sum_{t=1}^{T} [f_t(w_t) - f(u)]. \tag{515}$$

  - How do we relate these two? Here's where *convexity* comes in a crucial way.
  - Since $z_t \in \partial f_t(w_t)$ is a subgradient, we have by the definition of subgradient:

    $$f_t(u) \geq f_t(w_t) + z_t \cdot (u - w_t). \tag{516}$$

    Rearranging, we get a direct comparison for each term of the regret:

    $$\boxed{f_t(w_t) - f_t(u) \leq (w_t \cdot z_t) - (u \cdot z_t).} \tag{517}$$

  - The upshot is that the bounds we got for linear losses apply without modification to general losses! Intuitively, linear functions are the hardest to optimize using online convex optimization.
  - FIGURE: [draw convex f_t with linearized]

- Remarks:
  - OGD works for any convex loss function f_t (so does FTRL, but we only analyzed it for linear losses).
  - OGD is in some sense the first practical, non-toy algorithm we've developed.
  - Gradient-based methods are most commonly thought of as a procedure for optimizing a global objective, but this analysis provides a different perspective: that of doing full minimization of linearized losses with quadratic regularization.
  - The minimization viewpoint opens way to many other algorithms, all of which can be thought of as using different regularizers or using better approximations of the loss functions, while maintaining efficient parameter updates.

- **Example 30 (Online SVM)**
  - Let us use our result on OGD to derive a regret bound for learning SVMs in an online manner.
  - We will just apply OGD on the hinge loss for classification ($x_t \in \mathbb{R}^d, y_t \in \{+1, -1\}$):

$$f_t(w) = \max\{0, 1 - y_t(w \cdot x_t)\}. \tag{518}$$

    The algorithm (assume $S = \mathbb{R}^d$, so we don't need to project):
    * If $y_t(w_t \cdot x_t) \geq 1$ (classify correctly with margin 1): do nothing.[17]
    * Else: $w_{t+1} = w_t + \eta y_t x_t$.
  - Analysis:
    * Assume the data points are bounded: $\|x_t\|_2 \leq L$. Then $z_t \in \partial f_t(w_t)$ also satisfies that bound $\|z_t\|_2 \leq L$.
    * Assume that expert weights are bounded: $\|u\|_2 \leq B$.
    * The regret bound from Theorem 30 is as follows:

$$\boxed{\text{Regret} \leq BL\sqrt{T}.} \tag{519}$$

- **Example 31 (Learning with expert advice)**

[17] This algorithm is very similar to the Perceptron algorithm; the only difference is that Perceptron just requires any positive margin, not necessarily 1.

- Now let us consider learning with expert advice.
  - We maintain a distribution $w_t \in \Delta_d$ over d experts and predict by sampling an expert from that distribution.
  - Assume the zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$.
  - The loss function is linear: $f_t(w_t) = w_t \cdot z_t$, where

$$z_t = [\ell(y_t, h_1(x_t)), \cdots, \ell(y_t, h_d(x_t))] \in \{0,1\}^d \tag{520}$$

    is the loss vector.
- Bound on set of experts (B): the experts live in the simplex $S = \Delta_d$, which has its 2-norm bounded by $B = 1$ (attained at a corner).
- Bound on loss gradient (L):
  - The Lipschitz constant is bounded by the norm of the gradient z_t, which is at most $\sqrt{d}$.
  - Therefore, the regret bound we get is

$$\boxed{\text{Regret} \le BL\sqrt{T} = \sqrt{dT}.} \tag{521}$$

- Note that we are depending on the square root of the number of experts d rather than the log of the number of experts in our first online learning bound for learning with expert advice. Can we obtain a $\log d$ dependence without assuming realizability? We will find out in the next section.

5.7 Online mirror descent (OMD) (Lecture 14)

So far, we have analyzed FTRL for quadratic regularizers, which leads to (lazy projected) gradient-based algorithms. Quadratic regularization is imposing a certain prior knowledge, namely that there is a good parameter vector w in a small L_2 ball. But perhaps we know some dimensions to be more important than others. Then we might want to use a non-spherical regularizer. Or in the case of learning with expert advice, we know that $w \in \Delta_d$ (a probability distribution), so negative entropy might be more appropriate. In this section, we will develop a general way of obtaining regret bounds for general regularizers, and make explicit the glorious role that strong convexity plays. We make make extensive use of **Fenchel duality** and **Bregman divergences**.

- The goal for this lecture is to analyze FTRL (Algorithm 3) for arbitrary convex loss functions and regularizers. The resulting algorithm is this:
- **Algorithm 5** (online mirror descent (OMD))
  - Let $\psi : \mathbb{R}^d \to \mathbb{R}$ be the regularizer (this defines the learner).
  - Let $f_1, \dots, f_T$ be the sequence of loss functions played by nature.

  - On each iteration $t = 1, \dots, T$, the learner chooses weights w_t to minimize the regularized (linearized) loss:

$$w_t \in \arg\min_{w \in \mathbb{R}^d} \{\psi(w) - w \cdot \theta_t\}, \tag{522}$$

    where $z_t \in \partial f_t(w_t)$ is the t-th subgradient, and $\theta_t = -\sum_{i=1}^{t-1} z_i$ is the negative sum of the first $t - 1$ subgradients.

- Technical note: we will let the domain of weight vectors be unconstrained ($S = \mathbb{R}^d$). We can always fold a constraint into the regularizer by setting $\psi(w) = \infty$ if w violates the constraint.

- To recap the terminology:

  - OMD on f_t is equivalent to FTRL on the linearizations $w \mapsto w \cdot z_t$.
  - OGD is OMD with the quadratic regularizer $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$

- Examples of regularizers:

  - Quadratic regularizer: $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$.
  - Non-spherical quadratic regularizer: $\psi(w) = \frac{1}{2\eta} w^\top A w$ for $A \succeq 0$.
  - Entropic regularizer: $\psi(w) = \frac{1}{\eta}\sum_{j=1}^d w_j \log w_j$ if $w \in \Delta_d$ (this is the negative entropy defined on the probability simplex), and ∞ otherwise.
  - Note: the difference between the two regularizers is that the entropic regularizer slopes up violently when w approaches the boundary of the simplex Δ_d (the function value is finite but the gradient goes to infinity).

- We now need to build up some tools to help us analyze OMD. First, we introduce Fenchel duality, an important tool in optimization:

  - **Definition 27 (Fenchel conjugate)**

    * The **Fenchel conjugate**[18] of a function (not necessarily convex) $\psi : \mathbb{R}^d \to \mathbb{R}$ is

$$\boxed{\psi^*(\theta) \overset{\text{def}}{=} \sup_{w \in \mathbb{R}^d} \{w \cdot \theta - \psi(w)\}.} \tag{523}$$

  - Intuition

    * For scalars $w, \theta \in \mathbb{R}$, given a fixed θ (interpreted as a slope), $-\psi^*(\theta)$ is the position where the supporting hyperplane of ψ with slope θ hits the vertical axis.

[18] Also known as the convex conjugate or Legendre-Fenchel transformation.

  - · FIGURE: [draw ψ]
  - ∗ One can think of sweeping θ and reading out $\psi^*(\theta)$; this information in some sense encodes the epigraph of ψ if ψ is convex.

- – Useful facts:
  - ∗ ψ^* is always convex (even if ψ is not), since it's just a supremum over a collection of linear functions $\theta \mapsto w \cdot \theta - \psi(w)$.
  - ∗ $\psi^*(\theta) \ge w \cdot \theta - \psi(w)$ for all $w \in \mathbb{R}^d$ (**Fenchel-Young inequality**). This follows directly from the definition of ψ^*.
  - ∗ If $r(w) = a\psi(w)$ with $a > 0$, then $r^*(\theta) = a\psi^*(\theta/a)$. This is useful because once we've computed the Fenchel conjugate for one function, we know it for different scalings. In fact, Fenchel duality has a very nice algebra that makes computing Fenchel conjugates modular.
  - ∗ $\psi^{**} = \psi$ iff ψ is convex (and lower semi-continuous). In this case, we have

$$\psi(w) = \psi^{**}(w) = \sup_{\theta \in \mathbb{R}^d} \{w \cdot \theta - \psi^*(\theta)\}. \tag{524}$$

  - ∗ If ψ is differentiable, then

$$\nabla \psi^*(\theta) = \arg\max_{w \in \mathbb{R}^d} \{w \cdot \theta - \psi(w)\}. \tag{525}$$

    This is because the gradient of the supremum of a collection of functions at θ is the gradient of the function that attains the max at θ.

- – Mirroring
  - ∗ Comparing this with the the OMD update (522), we see that $w_t = \nabla\psi^*(\theta_t)$, and $-\psi^*(\theta_t)$ is the corresponding value of the regularized loss.
  - ∗ Since w_t attains the supremum of the Fenchel conjugate ψ^*, the optimality conditions (differentiate with respect to w) tell us that $\theta_t = \nabla\psi(w_t)$.
  - ∗ We have a one-to-one mapping between weights w and negative cumulative subgradients θ, linked via the gradients of ψ and ψ^*, which are inverses of one another:

$$\boxed{w_t = \nabla\psi^*(\theta_t), \quad \theta_t = \nabla\psi(w_t).} \tag{526}$$

  - ∗ This provides a very elegant view of what online mirror descent is doing. OMD takes a series of gradient updates $\theta_{t+1} = \theta_t - z_t$, generating $\theta_1, \theta_2, \dots, \theta_T$. These steps are *mirrored* via Fenchel duality in the sequence $w_1, w_2, \dots, w_T$.
  - ∗ FIGURE: [mapping]

- – **Example 32 (Quadratic regularizer)**
  - ∗ Let $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$.
  - ∗ Then $\psi^*(\theta) = \frac{\eta}{2}\|\theta\|_2^2$, attained by $w = \nabla\psi^*(\theta) = \eta\theta$.

  - In this case, w and θ are simple rescalings of each other.

- **Example 33 (Entropic regularizer)**
  - Let $\psi(w) = \frac{1}{\eta} \sum_{j=1}^{d} w_j \log w_j$ for $w \in \Delta_d$ (negative entropy).
  - Then $\psi^*(\theta) = \frac{1}{\eta} \log \left(\sum_{j=1}^{d} e^{\eta \theta_j} \right)$ (log partition function), attained by $w_j = \nabla \psi^*(\theta) = \frac{e^{\eta \theta_j}}{\sum_{k=1}^{d} e^{\eta \theta_k}}$.
  - Aside: this is maximum entropy duality in exponential families, where w and θ represent the mean and canonical parametrization of a multinomial distribution.

[begin lecture 15] (15)

5.8 Regret bounds with Bregman divergences (Lecture 15)

- Motivation
  - Having reinterpreted OMD as a mapping using a conjugate function, let's turn to proving a regret bound. Recall that in order to prove bounds, we needed to ensure that w_t and w_{t+1} don't change too much according to some criteria.
  - For quadratic regularization, this criteria was based on Euclidean distance.
  - We now generalize this by using Bregman divergences, which generalizes the notion of distance based on the regularizer.
  - **Definition 28 (Bregman divergence)**
    - Let f be a continuously-differentiable convex function.
    - The **Bregman divergence** between w and u is the difference at w between f and a linear approximation around u:

$$\boxed{D_f(w\|u) \stackrel{\text{def}}{=} f(w) - f(u) - \nabla f(u) \cdot (w - u).} \tag{527}$$

    - Intuitively, the divergence captures the the error of the linear approximation of f based on the gradient $\nabla f(u)$.
    - FIGURE: [show gap between f and its linear approximation]
  - Property: $D_f(w\|u) \geq 0$ (by definition of convexity).
  - Note: Bregman divergences are not symmetric and therefore not a distance metric.
  - **Example 34 (Quadratic regularizer)**
    - Let $f(w) = \frac{1}{2}\|w\|_2^2$.
    - Then $D_f(w\|u) = \frac{1}{2}\|w - u\|_2^2$ (squared Euclidean distance).
  - **Example 35 (Entropic regularizer)**
    - Let $f(w) = \sum_{j=1}^d w_j \log w_j$ for $w \in \Delta_d$ (negative entropy).
    - Then $D_f(w\|u) = \mathrm{KL}\,(w\|u) = \sum_j w_j \log(w_j/u_j)$ for $w \in \Delta_d$ (KL divergence).
  - Property (scaling): $D_{af}(w\|u) = aD_f(w\|u)$.
- **Theorem 31 (regret of OMD using Bregman divergences)**
  - OMD (Algorithm 5) obtains the following regret bound:

$$\boxed{\text{Regret}(u) \leq [\psi(u) - \psi(w_1)] + \sum_{t=1}^T D_{\psi^*}(\theta_{t+1}\|\theta_t).} \tag{528}$$

  - Furthermore, if all f_t's are linear, the inequality is actually an equality if u is the best expert. So this bound is pretty air tight.

- Proof of Theorem 31:
  - We assume all the loss functions are linear; recall in our analysis of OGD that linear functions are the worst case.
  - The key step is to find a potential function which allows us to monitor progress. The pivoting quantity is $\psi^*(\theta_{T+1})$, the negative regularized loss of the best fixed expert. This will allow us to relate the learner (w_t) to the expert (u).
  - Recall:
    * Learner's loss is $\sum_{t=1}^T w_t \cdot z_t$.
    * Expert's loss is $\sum_{t=1}^T u \cdot z_t$.
    * The regret is the difference between the two.
  - The expert:

$$\psi^*(\theta_{T+1}) \geq u \cdot \theta_{T+1} - \psi(u) \tag{529}$$

$$= \sum_{t=1}^T [-u \cdot z_t] - \psi(u) \tag{530}$$

    by the Fenchel-Young inequality. Note that we have equality if u is the best expert, by definition of ψ^*.
  - The learner:

$$\psi^*(\theta_{T+1}) = \psi^*(\theta_1) + \sum_{t=1}^T [\psi^*(\theta_{t+1}) - \psi^*(\theta_t)] \quad \text{[telescoping sums]} \tag{531}$$

$$= \psi^*(\theta_1) + \sum_{t=1}^T [\nabla\psi^*(\theta_t) \cdot (\theta_{t+1} - \theta_t) + D_{\psi^*}(\theta_{t+1} \| \theta_t)] \quad \text{[Bregman definition]} \tag{532}$$

$$= \psi^*(\theta_1) + \sum_{t=1}^T [-w_t \cdot z_t + D_{\psi^*}(\theta_{t+1} \| \theta_t)] \quad \text{[definition of OMD (526) and } \theta_t\text{].} \tag{533}$$

    Note that $\psi^*(\theta_1) = -\psi(w_1)$ since $\theta_1 = 0$.
  - Combining last equations for the expert and learner and rearranging yields the result.
  - The regret bound (528) is a generalization of (501), where $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ is quadratic regularizer, and $D_{\psi^*}(\theta_{t+1} \| \theta_t) = \frac{\eta}{2}\|z_t\|_2^2$.
  - However, a general Bregman divergence usually doesn't have such a nice form, and thus it is useful to bound it using a nicer quantity: a norm of some kind.

5.9 Strong convexity and smoothness (Lecture 15)

- First, let's review some intuition behind norms.
  - L_p norms decrease as p increases:

$$\|w\|_1 \ge \|w\|_2 \ge \|w\|_\infty. \tag{534}$$

  - The difference between the norms can be huge. For example, take $w = (1, \dots, 1) \in \mathbb{R}^d$. Then $\|w\|_1 = d$, $\|w\|_2 = \sqrt{d}$, $\|w\|_\infty = 1$.
  - Recall that the dual norm of a norm $\|\cdot\|$ is

$$\|x\|_* = \sup_{\|y\| \le 1} (x \cdot y). \tag{535}$$

  Thinking of y as a linear operator on x, the dual norm measures the gain when we measure perturbations in x using $\|\cdot\|$.

  - The norms $\|\cdot\|_p$ and $\|\cdot\|_q$ are dual to each other when $\frac{1}{p} + \frac{1}{q} = 1$. Two common examples are:

$$\|\cdot\|_1 \quad \text{is dual to} \quad \|\cdot\|_\infty. \tag{536}$$

$$\|\cdot\|_2 \quad \text{is dual to} \quad \|\cdot\|_2. \tag{537}$$

- We will now use these squared norms to control Bregman divergences provided the Bregman divergences have an important special structure called *strong convexity*:
- **Definition 29 (strong convexity/smoothness)**
  - A function f is α-strongly convex with respect to a norm $\|\cdot\|$ iff for all w, u:

$$D_f(w\|u) \ge \frac{\alpha}{2}\|w - u\|^2. \tag{538}$$

  - A function f is α-strongly smooth with respect to a norm $\|\cdot\|$ iff for all w, u:

$$D_f(w\|u) \le \frac{\alpha}{2}\|w - u\|^2. \tag{539}$$

- Intuitions
  - Strong convexity means that f must be growing at least quadratically.
  - Strong smoothness means that f must be growing slower than some quadratic function.
  - Example: the quadratic regularizer $\psi(w) = \frac{1}{2}\|w\|_2^2$ is both 1-strongly convex and 1-strongly smooth with respect to the L_2 norm, since $D_\psi(w\|u) = \frac{1}{2}\|w - u\|_2^2$.

- Duality links strong convexity and strong smoothness, as the following result shows.
- **Lemma 8 (strong convexity and strong smoothness)**
  - The following two statements are equivalent:
    - $\psi(w)$ is $1/\eta$-strongly convex with respect to a norm $\|\cdot\|$.
    - $\psi^*(\theta)$ is η-strongly smooth with respect to the dual norm $\|\cdot\|_*$.
  - Sanity check the quadratic regularizer: $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ and $\psi^*(\theta) = \frac{\eta}{2}\|\theta\|_2^2$.
- With these tools, we can finally make some progress on our regret bound from Theorem 31, rewriting the Bregman divergences in terms of norms.
- **Theorem 32 (regret of OMD using norms)**
  - Suppose ψ is a $\frac{1}{\eta}$-strongly convex regularizer.
  - Then the regret of online mirror descent is

$$\boxed{\text{Regret}(u) \le [\psi(u) - \psi(w_1)] + \frac{\eta}{2}\sum_{t=1}^{T} \|z_t\|_*^2.} \tag{540}$$

- Proof of Theorem 32
  - By Lemma 8, since ψ is $1/\eta$-strongly convex, the Fenchel conjugate ψ^* is η-strongly smooth.
  - By definition of strong smoothness and the fact that $\theta_{t+1} = \theta_t - z_t$,

$$\boxed{D_{\psi^*}(\theta_{t+1}\|\theta_t) \le \frac{\eta}{2}\|z_t\|_*^2.} \tag{541}$$

  - Plugging this bound into (528) gives us the result.
- Remarks:
  - If we plug in the quadratic regularizer $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$, then we get back the original result (501).
  - However, (540) generalizes to other regularizers.
  - We get to measure the size of z_t using the dual norm $\|\cdot\|_*$ rather than the default L_2 norm, which will help us get tighter bounds for the learning from expert advice problem.
- Learning form expert advice
  - Let's now use our tools to improve the regret bound that we got for learning from expert advice.

- Recall that when we used the quadratic regularizer, we got a regret bound of $\sqrt{dT}$ because $\|z_t\|_2$ could be as big as $\sqrt{d}$.
- To reduce this, let's just use another norm: $\|z_t\|_\infty \le 1$; no dependence on d!
- But this means that the regularizer ψ has to be strongly convex with respect to the L_1 norm, but this is harder because $\|\cdot\|_1 \ge \|\cdot\|_2$.
- Failure: the quadratic regularizer $\psi(w) = \frac{1}{2\eta}\|w\|_2^2$ is only $\frac{1}{\eta d}$-strongly convex with respect to the L_1 norm:

$$D_\psi(w \| u) \ge \frac{1}{2\eta}\|w-u\|_2^2 \ge \frac{1}{2\eta d}\|w-u\|_1^2. \tag{542}$$

  Sanity check: $\|(1,\dots,1)\|_2^2 = d$ but $\|(1,\dots,1)\|_1^2 = d^2$. So if we try to use this in the regret bound, we get $\frac{1}{2\eta} + \frac{T\eta d}{2}$, which is still $\sqrt{dT}$ (for optimal η).
- Failure: the L_1 regularizer $\psi(w) = \|w\|_1$ is neither strongly convex nor strongly smooth with respect to the any norm for any α: it doesn't grow fast enough for large w and grows too fast for small w.
- Success: entropic regularizer $\psi(w) = \frac{1}{\eta}\sum_j w_j \log w_j$ for $w \in \Delta_d$ is $1/\eta$-strongly convex with respect to the L_1 norm. This requires some algebra and an application of Cauchy-Schwartz (see Example 2.5 in Shai Shalev-Shwartz's online learning tutorial).
- FIGURE: [compare quadratic and entropic regularizer for $d = 2$]

- **Example 36 (exponentiated gradient (EG))**
  - Entropic regularizer: $\psi(w) = \frac{1}{\eta}\sum_{j=1}^d w_j \log w_j$ for $w \in \Delta_d$.
  - Recall $\psi^*(\theta) = \frac{1}{\eta}\log\sum_{j=1}^d e^{\eta\theta_j}$ and $\nabla\psi_j^*(\theta) = \frac{e^{\eta\theta_j}}{\sum_{k=1}^d e^{\eta\theta_k}}$.
  - OMD updates:

$$\boxed{w_{t,j} \propto e^{\eta\theta_{t,j}}.} \tag{543}$$

    The equivalent recursive formula:

$$\boxed{w_{t+1,j} \propto w_{t,j} e^{-\eta z_{t,j}}.} \tag{544}$$

    Interpretation: we maintain a distribution over the d experts. We use the gradient z_t to reweight the experts, normalizing afterwards to ensure a proper distribution.
  - Recap: the exponentiated gradient (EG) algorithm is just online mirror descent using the entropic regularizer.

- **Example 37 (EG for learning with expert advice)**

- Consider learning with expert advice (Example 27).
- We use the expected zero-one loss: $f_t(w) = w \cdot z_t$, where z_t is the loss vector.
- We have that the dual norm of the gradients are bounded $\|z_t\|_\infty \le 1$.
- The minimum and maximum values of the regularizer:
  - $\max_w \psi(w) = 0$ (minimum entropy)
  - $\min_w \psi(w) = \frac{\log(1/d)}{\eta} = \frac{-\log d}{\eta}$ (maximum entropy)
- Then

$$\text{Regret} = \frac{\log(d)}{\eta} + \frac{\eta T}{2}. \tag{545}$$

- Setting $\eta = \sqrt{\frac{2 \log d}{T}}$ yields:

$$\boxed{\text{Regret} = \sqrt{2 \log(d) T}.} \tag{546}$$

- To compare with quadratic regularization (OGD):
  - Quadratic: $[\max_u \psi(u) - \min_u \psi(u)] \le \frac{1}{2\eta}$, $\|z_t\|_2 \le \sqrt{d}$
  - Entropic: $[\max_u \psi(u) - \min_u \psi(u)] \le \frac{\log d}{\eta}$, $\|z_t\|_\infty \le 1$
- Discussion
  - Online mirror descent (OMD) allows us to use different regularizers based on our prior knowledge about the expert vector u and the data z_t. As we see with EG, tailoring the regularizer can lead to better bounds.
  - Using the L_2 norm means that we use the bound $\|z_t\|_2 \le \sqrt{d}$. To get rid of the dependence on d here, we use the L_∞ norm with $\|z_t\|_\infty \le 1$.
  - However, this means that ψ must be strongly convex with respect to the L_1 norm. The quadratic regularizer isn't strong enough (only $\frac{1}{\eta d}$-strongly convex with respect to L_2), so we need the entropic regularizer, which is 1-strongly convex with respect to L_1.
  - Curvature hurts us because $[\psi(u) - \psi(w_1)]$ is now larger, but the simplex is small, so $[\psi(u) - \psi(w_1)]$ only grows from 1 (with the quadratic regularizer) to $\log d$ (with the entropic regularizer).
  - So the tradeoff was definitely to our advantage.

5.10 Local norms (Lecture 15)

- Recall that that the general online mirror descent (OMD) analysis (Theorem 31) yields:

$$\text{Regret}(u) \le [\psi(u) - \psi(w_1)] + \sum_{t=1}^{T} D_{\psi^*}(\theta_{t+1} \| \theta_t). \tag{547}$$

  Using the fact that ψ is $1/\eta$-strongly convex with respect to some norm $\|\cdot\|$, we can upper bound the Bregman divergence by the following (540):

$$D_{\psi^*}(\theta_{t+1} \| \theta_t) \le \frac{\eta}{2} \|z_t\|_*^2. \tag{548}$$

- If we use the entropic regularizer for ψ with norm $\|\cdot\| = \|\cdot\|_1$ and dual norm $\|\cdot\|_* = \|\cdot\|_\infty$, we get our key $\sqrt{2\log(d)T}$ regret bound for EG.

- In this section, we will use the local norms technique to improve (548). This will allow us to do two things:

  - Recover the strong $O(\log d)$ bound for the realizable setting.
  - Allow us to analyze the multi-armed bandit setting.

- Why we should do better:

  - Consider an example where there are two experts: one which is perfect ($z_{t,1} = 0$ for all t) and one which is horrible ($z_{t,2} = 1$ for all t).
  - The EG algorithm will quickly downweight the bad expert exponentially fast:

$$w_{t,1} \propto 1 \qquad w_{t,2} \propto e^{-\eta(t-1)}. \tag{549}$$

  - So as $t \to \infty$, we have basically put all our weight on the first expert, and should be suffering no loss.
  - But $\|z_t\|_\infty^2 = 1$, so in the regret bound we still pay $\frac{\eta T}{2}$, which is simply a travesty.
  - We would hope that $\|z_t\|_\infty^2 = \max_{1 \le j \le d} z_{t,j}^2$ should be replaced with something that is sensitive to how much weight we're placing on it, which is $w_{t,j}$. Indeed the following theorem fulfills this dream:

- **Theorem 33 (exponentiated gradient (analysis using local norms))**

  - Assume nature plays a sequence of linear loss functions $f_t(w) = w \cdot z_t$, where $z_{t,j} \ge 0$ for all $t = 1, \dots, T$ and $j = 1, \dots, d$.
  - Then the exponentiated gradient (EG) algorithm (Example 37) achieves the following regret bound:

$$\boxed{\text{Regret}(u) \le [\psi(u) - \psi(w_1)] + \eta \sum_{t=1}^{T} \sum_{j=1}^{d} w_{t,j} z_{t,j}^2.} \tag{550}$$

- This bound (550) is better than the bound in Theorem 32 because we are taking an average (with respect to the distribution $w_t \in \Delta_d$) instead of a max:

$$\sum_{j=1}^{d} w_{t,j} z_{t,j}^2 \le \max_{1 \le j \le d} z_{t,j}^2 \overset{\text{def}}{=} \|z_t\|_\infty^2. \tag{551}$$

  This allows some components of the loss subgradients $z_{t,j}$ to be large provided that the corresponding weights $w_{t,j}$ are small.

- **Example 38 (exponentiated gradient in the realizable setting)**

  - Assume the loss vector is bounded: $z_t \in [0,1]^d$.
  - Assume there exists an expert $u \in \Delta_d$ with zero cumulative loss (realizability).
  - Recall the regret bound from using local norms:

$$\text{Regret}(u) \overset{\text{def}}{=} \sum_{t=1}^{T}(w_t \cdot z_t) - \underbrace{\sum_{t=1}^{T}(u \cdot z_t)}_{=0} \le \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \underbrace{\sum_{j=1}^{d} w_{t,j} z_{t,j}^2}_{\le w_t \cdot z_t} \tag{552}$$

    .

  - Note that $\sum_{j=1}^{d} w_{t,j} z_{t,j}^2 \le w_t \cdot z_t$, because all quantities are non-negative and $a^2 \le a$ for $a \in [0,1]$.
  - Rearranging, we get:

$$\text{Regret}(u) \le \frac{\log d}{\eta(1-\eta)}. \tag{553}$$

  - Minimize the bound with $\eta = \frac{1}{2}$ to obtain:

$$\boxed{\text{Regret}(u) \le 4 \log d.} \tag{554}$$

  - Recall that the majority algorithm (which aggressively zeros out the weights of components as soon as they err) also obtained the very low $O(\log d)$ regret (see Example 25), so it's really nice to see that EG obtains the same regret guarantee.
  - If the problem isn't realizable, the majority algorithm isn't even correct (it will eliminate all the experts), whereas EG will gracefully fall back to $O(\sqrt{\log(d)T})$ regret.

- Now that you are hopefully somewhat convinced that the theorem is useful, let's prove it.

- Proof of Theorem 33:

- This proof mostly involves starting with the Bregman divergence, and performing some low-level algebraic manipulation. Perhaps the most useful high-level take-away is whenever we're trying to massage some expression with log's and exp's, it's useful to try to approximate the functions using linear and quadratic approximations (think Taylor approximations).
- Specifically, we will use the following two facts:
  - Fact 1: $e^{-a} \le 1 - a + a^2$ for $a \ge 0$ (this actually holds for smaller a, but let's keep it simple)
  - Fact 2: $\log(1-a) \le -a$
- Recall the Fenchel conjugate of the entropic regularizer is the log-partition function:

$$\psi^*(\theta) = \frac{1}{\eta} \log \sum_{j=1}^{d} e^{\eta \theta_j}. \tag{555}$$

- By the definition of Bregman divergences (this was used in the proof of Theorem 31), we have:

$$D_{\psi^*}(\theta_{t+1} \| \theta_t) = \psi^*(\theta_{t+1}) - \psi^*(\theta_t) - \underbrace{\nabla \psi^*(\theta_t)}_{w_t} \cdot \underbrace{(\theta_{t+1} - \theta_t)}_{-z_t}. \tag{556}$$

- The rest is just applying the two facts and watching stuff cancel:

$$\begin{aligned}
D_{\psi^*}(\theta_{t+1} \| \theta_t) &= \psi^*(\theta_{t+1}) - \psi^*(\theta_t) + w_t \cdot z_t && (557)\\
&= \frac{1}{\eta} \log \left(\frac{\sum_{j=1}^d e^{\eta \theta_{t+1,j}}}{\sum_{j=1}^d e^{\eta \theta_{t,j}}} \right) + w_t \cdot z_t \quad [\text{definition of } \psi^*] && (558)\\
&= \frac{1}{\eta} \log \left(\sum_{j=1}^d w_{t,j} e^{-\eta z_{t,j}} \right) + w_t \cdot z_t \quad [\text{definition of } \theta_t] && (559)\\
&\le \frac{1}{\eta} \log \left(\sum_{j=1}^d w_{t,j} [1 - (\eta z_{t,j} - \eta^2 z_{t,j}^2)] \right) + w_t \cdot z_t \quad [\text{fact 1}] && (560)\\
&= \frac{1}{\eta} \log \left(1 - \sum_{j=1}^d w_{t,j} (\eta z_{t,j} - \eta^2 z_{t,j}^2) \right) + w_t \cdot z_t \quad [w_t \in \Delta_d] && (561)\\
&\le \frac{1}{\eta} \sum_{j=1}^d w_{t,j} (-\eta z_{t,j} + \eta^2 z_{t,j}^2) + w_t \cdot z_t \quad [\text{fact 2}] && (562)\\
&= \eta \sum_{j=1}^d w_{t,j} z_{t,j}^2 \quad [\text{algebra}]. && (563)
\end{aligned}$$

[begin lecture 16] (16)

5.11 Adaptive optimistic mirror descent (Lecture 16)

- Motivation
  - Recall that the local norm bound technique for exponentiated gradient (EG) yields the following bound (550):

$$\mathrm{Regret}(u) \le \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \sum_{j=1}^{d} w_{t,j} z_{t,j}^2, \tag{564}$$

    improving the $\|z_t\|_\infty^2$ from the standard mirror descent analysis (Example 37) to the local norm $\|z_t\|_{\mathrm{diag}(w_t)}^2$. (Note that this was just an improvement in the analysis, not the algorithm.)
  - But there are two weaknesses of this regret bound. To see the first weakness, consider the following example.
  - **Example 39 (perfect and confused expert)**
    - We have two experts:
      - Expert 1 is perfect ($z_{t,1} = 0$).
      - Expert 2 is just confused and alternates between loss of -1 and $+1$ ($z_{t,2} = (-1)^{t-1}$).
    - Playing either expert 1 or expert 2 alone is optimal, yielding a cumulative loss of 0 (for expert 2, for even T). But expert 2 has higher variance and is thus intuitively riskier, so maybe we should avoid it.
    - However, the EG algorithm does not:

$$w_t \propto \begin{cases} [1,1] & \text{if } t \text{ is odd} \\ [1,\exp(-\eta)] & \text{if } t \text{ is even.} \end{cases} \tag{565}$$

      EG doesn't penalize expert 2 on the odd rounds at all and barely penalizes it on the even rounds, because expert 2 redeems itself on the even rounds.
    - On this example, assuming the step size $\eta \le 1$, EG incurs a regret (not just

a regret bound) of:

$$\text{EG regret} = \frac{T}{2}\left(\frac{1}{2}(1) + \frac{\exp(-\eta)}{1+\exp(-\eta)}(-1)\right) \tag{566}$$

$$= \frac{T}{2}\left(\frac{1}{2} - \frac{1}{1+\exp(\eta)}\right) \tag{567}$$

$$\geq \frac{T}{2}\left(\frac{1}{2} - \frac{1}{2+2\eta}\right) \quad [\exp(\eta) \leq 1+2\eta \text{ for } \eta \leq 1] \tag{568}$$

$$\geq \frac{T}{2}\eta. \tag{569}$$

  - Typically, $\eta = \Theta(\frac{1}{\sqrt{T}})$, in which case, the regret is $\Omega(T)$.
  - If η is too large, then EG ends up swinging back and forth between the two experts, and incurring a lot of regret, just as in the FTL example with linear loss functions (Example 29). On the other hand, η can't be too small either, or else we don't end up being able to switch to a good expert quickly enough (consider the example where expert 2 gets a loss of 1 always).

- The second weakness is that if we add 5 to the losses of all the experts on all T iterations, then the regret bound suffers something like an extra $25\eta T$. But morally, the problem hasn't changed!
- In what follows, we will produce two improvements to the EG algorithm. There are two key ideas:
  - **Adaptivity**: instead of using a fixed regularizer ψ, we will use a regularizer ψ_t that depends on the current iteration. This way, we can adapt the regularizer to the loss functions that we've seen so far.
  - **Optimism**: we will incorporate hints $m_1, \dots, m_T$, which are a guess of the actual loss sequence $z_1, \dots, z_T$. By incorporating these hints, our algorithm can do much better when the hints are correct, but still be robust enough to perform well when the hints are not correct. As a concrete example, think of $m_t = z_{t-1}$ (a recency bias), which hints that the subgradients aren't changing very much.

Using these two ideas, we define an algorithm called **adaptive optimistic mirror descent (AOMD)**, which obtain the following regret bound for learning with expert advice (for $\eta \leq \frac{1}{4}$):

$$\text{Regret}(u) \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \sum_{j=1}^{d} u_j (z_{t,j} - m_{t,j})^2. \tag{570}$$

Compared to (564), (570) depends on the expert u_j rather than the learner $w_{t,j}$; and it also depends on the squared errors $(z_{t,j} - m_{t,j})^2$ rather than $z_{t,j}^2$. Unlike local norms, this requires an actual change to the algorithm—a change which is inspired by the theory.

- Recall that the normal mirror descent updates are given as follows:

$$w_t = \arg\min_{w \in \mathbb{R}^d} \left\{ \psi(w) + \sum_{i=1}^{t-1} w \cdot z_i \right\} = \nabla \psi^*(\theta_t). \tag{571}$$

- Define the following **adaptive** mirror descent updates:

$$w_t = \arg\min_{w \in \mathbb{R}^d} \left\{ \psi_t(w) + \sum_{i=1}^{t-1} w \cdot z_i \right\} = \nabla \psi_t^*(\theta_t), \tag{572}$$

  where all we've done is replaced ψ with ψ_t.

- To put **optimism** in, suppose we have a sequence of **hints**

$$m_1, \dots, m_T, \tag{573}$$

  which are guesses of $z_1, \dots, z_T$. These have the property that m_t only depends on $z_{1:t-1}$. For example, the simplest one could be $m_t = z_{t-1}$ (recency bias), which means that we believe the subgradients do not change very much. Note that if $m_t = z_t$, then we would be cheating and using the one-step lookahead. The **adaptive optimistic mirror descent** updates are as follows:

$$\boxed{w_t = \arg\min_{w \in \mathbb{R}^d} \left\{ \psi(w) + \sum_{i=1}^{t-1} w \cdot z_i + w \cdot m_t \right\} = \nabla \psi^*(\tilde{\theta}_t),} \tag{574}$$

  where $\tilde{\theta}_t \overset{\text{def}}{=} \theta_t - m_t$ is the guess of θ_{t+1}.

- **Theorem 34 (regret of adaptive optimistic mirror descent)**
  - Let $m_1, \dots, m_T$ be any sequence of hints. Assume $m_1 = 0$.
  - Suppose we have a sequence of regularizers $\psi_1, \dots, \psi_T$ satisfying the following loss-bounding property:

$$\psi_{t+1}^*(\theta_{t+1}) \le \psi_t^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t). \tag{575}$$

  - Then

$$\boxed{\text{Regret}(u) \le \psi_{T+1}(u) - \psi(w_1).} \tag{576}$$

- Note: in our previous mirror descent analysis, we had two terms, one coming from the regularizer, and one coming from the stability of our estimates. In contrast, (575) "pushes the loss into the regularizer", which gives us a bit more flexibility.

- Proof of Theorem 34:

$$u \cdot \theta_{T+1} - \psi_{T+1}(u) \le \psi^*_{T+1}(\theta_{T+1}) \quad \text{[Fenchel-Young]} \tag{577}$$

$$= \psi^*_1(\theta_1) + \sum_{t=1}^{T} [\psi^*_{t+1}(\theta_{t+1}) - \psi^*_t(\theta_t)] \quad \text{[telescoping]} \tag{578}$$

$$= \psi^*_1(\theta_1) + \sum_{t=1}^{T} [\psi^*_{t+1}(\theta_{t+1}) - \psi^*_t(\tilde{\theta}_t) - \underbrace{\nabla\psi^*(\tilde{\theta}_t)}_{w_t} \cdot m_t] \quad \text{[convexity of } \psi^*_t\text{]} \tag{579}$$

$$= \psi^*_1(\theta_1) + \sum_{t=1}^{T} -w_t \cdot z_t \quad \text{[loss-bounding property]} \tag{580}$$

$$= -\psi_1(w_1) + \sum_{t=1}^{T} -w_t \cdot z_t \quad \text{[since } m_1 = 0\text{].} \tag{581}$$

  Recall that $\theta_{T+1} = -\sum_{t=1}^{T} z_t$. Algebra completes the proof.

- Now what kind of regularizer should we use to satisfy the loss-bounding property?
  - First attempt: let's try just using a fixed regularizer $\psi_t(w) = \psi(w)$. This means that we need

$$\psi^*(\theta_{t+1}) \le \psi^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t). \tag{582}$$

    But in fact, since ψ^* is convex, and recalling $w_t = \nabla\psi^*(\tilde{\theta}_t)$ and $\theta_{t+1} - \tilde{\theta}_t = -(z_t - m_t)$ we actually have:

$$\psi^*(\theta_{t+1}) \ge \psi^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t), \tag{583}$$

    which is the opposite of what we want!
  - FIGURE: [graph of ψ^* $\theta_t, \tilde{\theta}_t, \theta_{t+1}$]
  - Second attempt: let's try to provide a second-order correction:

$$\psi^*(\theta_{t+1} - \eta(z_t - m_t)^2) \le \psi^*(\tilde{\theta}_t) - w_t \cdot (z_t - m_t). \tag{584}$$

    The $\eta(z_t - m_t)^2$ term makes the LHS smaller. However, condition doesn't look like the loss-bounding precondition we need for Theorem 34. To make the condition match, the key is to *fold the second-order term into the regularizer.*
  - To that end, define the vector a_t to be the squared magnitude deviations between z_t and m_t:

$$a_t \overset{\text{def}}{=} (a_{t,1}, \dots, a_{t,d}), \quad \boxed{a_{t,j} \overset{\text{def}}{=} (z_{t,j} - m_{t,j})^2.} \tag{585}$$

This motivates us to consider the following adaptive regularizer:

$$\boxed{\psi_t(w) \overset{\text{def}}{=} \psi(w) + \eta \sum_{i=1}^{t-1} w \cdot a_i.} \tag{586}$$

In the expert advice setting where w is a distribution over experts, we are penalizing experts that have large deviations from the hints, which is in line with our intuitions that we want experts which match our hints (kind of like our prior).

- Algorithmically, AOMD with this choice of ψ_t is:

$$w_t = \arg\min_w \left\{ \psi(w) + \eta \sum_{i=1}^{t-1} w \cdot a_i - w \cdot \tilde{\theta}_t \right\} \tag{587}$$

$$= \arg\min_w \left\{ \psi(w) - w \cdot \tilde{\beta}_t \right\} \tag{588}$$

$$= \nabla \psi^*(\tilde{\beta}_t), \tag{589}$$

where we define the second-corrected vectors (without and with m_t-lookahead):

$$\beta_t \overset{\text{def}}{=} \theta_t - \eta \sum_{i=1}^{t-1} a_i, \quad \tilde{\beta}_t \overset{\text{def}}{=} \beta_t - m_t. \tag{590}$$

Recall that $\tilde{\theta}_t = \theta_t - m_t$.

- **Theorem 35 (second-order corrections)**
  - Consider running AOMD using the adaptive regularizer ψ_t defined in (586).
  - Assume the second-order-corrected loss-bounding property holds:

$$\psi^*(\beta_{t+1}) \le \psi^*(\tilde{\beta}_t) - w_t \cdot (z_t - m_t). \tag{591}$$

  - Then we have the following regret bound:

$$\boxed{\text{Regret}(u) \le \psi(u) - \psi(w_1) + \eta \sum_{t=1}^{T} u \cdot a_t.} \tag{592}$$

- Proof of Theorem 35
  - We need to translate the second-order-corrected loss-bounding property on ψ the loss-bounding property on ψ_t.
    - Since $\psi_t(w) = \psi(w) + w \cdot (\eta \sum_{i=1}^{t-1} a_i)$ by definition, Fenchel conjugacy yields:

$$\psi_t^*(x) = \psi^* \left(x - \eta \sum_{i=1}^{t-1} a_i \right). \tag{593}$$

In other words, ψ_t^* is just ψ^* shifted by the second-order corrections.

- * Therefore,

$$\psi^*_{t+1}(\theta_{t+1}) = \psi^*(\beta_{t+1}), \quad , \psi^*_t(\tilde{\theta}_t) = \psi^*(\tilde{\beta}_t). \tag{594}$$

- Note that $\beta_1 = 0$, so $\psi^*(\beta_1) = -\psi(w_1)$.
- Then:

$$\begin{aligned} \text{Regret}(u) &\le \psi_{T+1}(u) - \psi(w_1) \quad \text{[from Theorem 34]} \quad (595) \\ &\le \psi(u) + \eta \sum_{t=1}^{T} u \cdot a_t - \psi(w_1) \quad \text{[definition of } \psi_t]. \quad (596) \end{aligned}$$

- Now let us apply this result to the expert advice setting.

- **Theorem 36 (adaptive optimistic gradient descent (expert advice))**

  - Consider $\|z_t\|_\infty \le 1$ and $\|m_t\|_\infty \le 1$.
  - Let $\psi(w) = \frac{1}{\eta} \sum_{j=1}^{d} w_j \log w_j$ be the standard entropic regularizer, which corresponds to the following algorithm:

$$w_{t,j} \propto \exp(\tilde{\beta}_{t,j}) = \exp\left(\eta\theta_{t,j} - \underbrace{\eta m_{t,j}}_{\text{optimism}} \underbrace{- \eta^2 \sum_{i=1}^{t-1} (z_{i,j} - m_{i,j})^2}_{\text{adaptivity}} \right). \tag{597}$$

  - Assume $0 < \eta \le \frac{1}{4}$.
  - Then adaptive optimistic mirror descent (AOMD) obtains the following regret bound:

$$\boxed{\text{Regret}(u) \le \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \sum_{j=1}^{d} u_j (z_{t,j} - m_{t,j})^2.} \tag{598}$$

- Proof sketch: do algebra on $\psi^*(\beta_{t+1})$ to show the condition of Theorem 35; see Steinhardt and Liang (2014).

- **Example 40 (path length bound)**

  - Set $m_t = z_{t-1}$ be the average subgradients in the past.
  - Algorithmically, this corresponds to counting the last z_{t-1} twice in addition to penalizing deviations.
  - FIGURE: [plot expert paths over time]

- We obtain

$$\text{Regret} \le \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} (z_{t,j^*} - z_{t-1,j^*})^2, \tag{599}$$

where $j^* \in [d]$ is best expert (the one with the lowest cumulative loss).

- On Example 39, expert 1 is a best expert (obtains cumulative loss 0). We have that the path length for expert 1 is $(z_{t,1} - z_{t-1,1})^2 = 0$, so we obtain:

$$\text{Regret} \le \frac{\log d}{\eta}. \tag{600}$$

The bound only holds for $\eta \le \frac{1}{4}$. By setting $\eta = \frac{1}{4}$, we that the regret is a mere constant:

$$\text{Regret} \le 4 \log d. \tag{601}$$

- Note that unlike EG (which had $\Omega(\sqrt{T})$ regret), we don't get distracted by expert 2, who has very low loss as well, but has longer path length and is thus discounted.

- At a high-level, what we're doing with the hints m_t is penalizing experts whose losses $z_{t,j}$ don't agree with our "prior" $m_{t,j}$. Of course, the hints have only a secondary $\Theta(\eta)$ effect, whereas the actual losses have a primary $\Theta(1)$ effect.

- This section is mostly based on Steinhardt and Liang (2014), who combines optimistic mirror descent (Rakhlin and Sridharan, 2013) and adaptive mirror descent (Orabona et al., 2015).

5.12 Online-to-batch conversion (Lecture 16)

- So far, we have been focusing on the **online** setting, where the learner receives one example at a time and is asked to make a prediction on each one. Good online learners have low **regret**.

- Sometimes it is more natural to operate in the **batch** setting, where the learner gets a set of training examples, learns a model, and then is asked to predict on new test examples. Good batch learners have low **expected risk**.

- In this section, we will show that low regret implies low expected risk by explicitly reducing the online setting to the batch setting.

- Batch setting

  - Assume we have a unknown data-generating distribution $p^*(z)$, where we use $z = (x, y) \in \mathcal{Z}$ to denote an input-output pair.

- Assume our hypothesis class is a convex set of weight vectors $S \subseteq \mathbb{R}^d$.
- As in online learning, we define a convex loss function $\ell(z, w) \in \mathbb{R}$; for example, the squared loss for linear regression would be $\ell((x, y), w) = (w \cdot x - y)^2$. Assume $\ell(z, w)$ is convex in w for each $z \in \mathcal{Z}$.
- The **expected risk** of a weight vector $w \in S$ is defined as follows:

$$\boxed{L(w) = \mathbb{E}_{z \sim p^*}[\ell(z, w)].} \tag{602}$$

- Define the weight vector that minimizes the expected risk:

$$w^* \in \arg\min_{w \in S} L(w). \tag{603}$$

- The batch learner gets T i.i.d. training examples $(z_1, \dots, z_T)$, where each $z_t \sim p^*$. The goal is to output some estimate $w \in S$ that hopefully has low $L(w)$.

Assuming we have an online learner as a black box, we will construct a batch learner as follows:

- **Algorithm 6 (online-to-batch conversion)**
  - Input: T training examples $z_1, \dots, z_T$.
  - Iterate $t = 1, \dots, T$:
    - Receive w_t from the online learner.
    - Send loss function $f_t(w) = \ell(z_t, w)$ to the online learner.
  - Return the average of the weight vectors: $\bar{w} = \frac{1}{T} \sum_{t=1}^T w_t$.
- Remarks about randomness
  - In contrast to the online learning (adversarial) setting, many of the quantities we are working with now have distributions associated with them.
  - For example, f_t is a random function that depends on z_t.
  - Each w_t is a random variable which depends on (i.e., is in the sigma-algebra of) $z_{1:t-1}$.
  - Note that $L(w^*)$ is not random.
- **Theorem 37 (Online-to-batch conversion)**
  - Recall the usual definition of regret (which depends on $z_{1:T}$):

$$\text{Regret}(u) = \sum_{t=1}^T [f_t(w_t) - f_t(u)]. \tag{604}$$

  - Online-to-batch conversion obtains the following expected expected risk:

$$\boxed{\mathbb{E}[L(\bar{w})] \le L(w^*) + \frac{\mathbb{E}[\text{Regret}(w^*)]}{T}.} \tag{605}$$

- Interpretation: the expected expected risk of the online-to-batch conversion $L(\bar{w})$ is bounded by the best possible expected risk (attained by $w^* \in S$) plus the online regret.

- Note that $\mathbb{E}[L(\bar{w})]$ has two expectations here: the $\mathbb{E}[\cdot]$ is an expectation over possible training datasets, and L contains an expectation over test examples.

- Proof:
  - The first key insight is that $f_t(w_t)$ provides an unbiased estimate of the expected risk of w_t.

$$\mathbb{E}[f_t(w_t) \mid w_t] = L(w_t). \tag{606}$$

  This is true because all the examples are independent, so w_t (deterministic function of $z_{1:t-1}$) is independent of f_t (deterministic function of z_t).

  - The second key insight is that averaging can only reduce loss. Since $\ell(z, \cdot)$ is convex, and an average of convex functions is convex, L is convex. By Jensen's inequality:

$$L(\bar{w}) \le \frac{1}{T}\sum_{t=1}^{T} L(w_t). \tag{607}$$

  - Now, the rest is just putting the pieces together. Putting (607) and (606) together:

$$L(\bar{w}) \le \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[f_t(w_t) \mid w_t]. \tag{608}$$

  Adding and subtracting $L(w^*)$ to the RHS, noting that $L(w^*) = \mathbb{E}[f_t(w^*)]$:

$$L(\bar{w}) \le L(w^*) + \frac{1}{T}\sum_{t=1}^{T} (\mathbb{E}[f_t(w_t) \mid w_t] - \mathbb{E}[f_t(w^*)]). \tag{609}$$

  - Taking expectations on both sides:

$$\mathbb{E}[L(\bar{w})] \le L(w^*) + \frac{1}{T}\sum_{t=1}^{T} (\mathbb{E}[f_t(w_t) - f_t(w^*)]). \tag{610}$$

  - The second term of the RHS is upper bounded by the regret, so we have:

$$\mathbb{E}[L(\bar{w})] \le L(w^*) + \frac{\mathbb{E}[\text{Regret}(w^*)]}{T}. \tag{611}$$

- Remarks
  - If you run online subgradient descent once over your training data,[19] you should expect $O(\sqrt{T})$ regret by our previous analyses.
  - The resulting average weight vector will, *in expectation*,[20] have an expected risk which is within $O(1/\sqrt{T})$ of the best weight vector.
  - If the batch learner returns the last weight vector w_{T+1}, then our analysis above doesn't apply.
  - In general, averaging is a useful concept for stabilizing learning algorithms.

5.13 Adversarial bandits: expert advice (Lecture 16)

- In online learning with expert advice, on each iteration, after we choose one of the d experts/actions, nature reveals the loss vector z_t for every single action.

- However, in many applications such as clinical trials or advertisement placement, packet routing, we only get to observe the loss of the action we took, not the losses of the actions we didn't take.

- This setting is known as the multi-armed bandit problem, which is a type of online learning problem with **partial feedback**.

- This problem is much more difficult. Intuitively, the learner should choose actions that we expect to yield low loss, but it must choose which actions to explore to get more information about the losses. Thus, the multi-armed bandit problem exposes the challenges of the classic exploration/exploitation tradeoff.[21]

- Setup
  - FIGURE: [matrix with loss functions]
  - There are d possible actions (corresponding to the arms of a row of slot machines).
  - Let $z_t \in [0,1]^d$ denote the vector of losses of the d actions at iteration t.
  - For each iteration $t = 1, \dots, T$:
    - ∗ Learner chooses a distribution $w_t \in \Delta_d$ over actions, and samples an action $a_t \sim w_t$.
    - ∗ Learner observes the loss of *only that particular action* z_{t,a_t} and no others.

[19] In practice, it helps to run the online learning algorithm multiple times over the training data.

[20] You can get high probability bounds too using martingales, but we won't discuss those here.

[21] Reinforcement learning requires managing the exploration/exploitation tradeoff, but is even more difficult because actions take the learner between different states in a way that is unknown to the learner.

  - The expected regret with respect to an expert $u \in \Delta_d$ is defined in the same way as it was for online learning with expert advice:

$$\mathbb{E}[\text{Regret}(u)] = \sum_{t=1}^{T} [w_t \cdot z_t - u \cdot z_t]. \tag{612}$$

    Note that taking the max over randomized experts $u \in \Delta_d$ is equivalent to taking the max over single actions $a \in [d]$, since the maximum over a linear function is attained at one of the vertices.

- Estimating the loss vector

  - Recall that in online learning, we would observe the entire loss vector z_t. In that case, we could use the exponentiated gradient (EG) algorithm ($w_{t+1,j} \propto w_{t,j} e^{-\eta z_{t,j}}$) to obtain a regret bound of Regret $\le \sqrt{2 \log(d) T}$ (see Example 37).
  - In the bandit setting, we don't observe z_t, so what can we do? Let's try to estimate it with some $\hat{z}_t$ that (i) we can observe and (ii) is equal to z_t in expectation (unbiased) in that for all $a = 1, \dots, d$:

$$\mathbb{E}_{a_t \sim w_t}[\hat{z}_{t,a} \mid w_t] = z_{t,a}. \tag{613}$$

  - Given these two constraints, it's not hard to see that the only choice for $\hat{z}_t$ is:

$$\boxed{\hat{z}_{t,a} = \begin{cases} \frac{z_{t,a}}{w_{t,a}} & \text{if } a = a_t \\ 0 & \text{otherwise.} \end{cases}} \tag{614}$$

    Note that dividing $w_{t,a}$ compensates for sampling $a_t \sim w_t$.

- **Algorithm 7 (Bandit-EG)**

  - Run EG with the unbiased esimate $\hat{z}_t$ rather than z_t. Really, that's it.

- **Theorem 38 (Bandit-EG analysis)**

  - Bandit-EG obtains expected regret (expectation taken over learner's randomness) of

$$\boxed{\mathbb{E}[\text{Regret}(u)] \le 2\sqrt{d \log(d) T}.} \tag{615}$$

- Comparison of Bandit-EG (bandit setting) and EG (online learning setting):

  - Compared with the bound for EG in the full information online learning setting (546), we see that this bound (615) is worse by a factor of $\sqrt{d}$.

- In other words, the number of iterations T for Bandit-EG needs to be d times larger in order to obtain the same average regret as EG.
- This is intuitive since in the bandit setting, each iteration reveals $(1/d)$-th the amount of information compared with the online learning setting.

- Proof of Theorem 38:
  - If EG is run with the unbiased estimate $\hat{z}_t$ ($z_t = \mathbb{E}[\hat{z}_t \mid w_t]$), then the expected regret is simply:

$$\mathbb{E}[\text{Regret}(u)] \le \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \mathbb{E}_{a_t \sim w_t}\left[\sum_{a=1}^{d} w_{t,a}\hat{z}_{t,a}^2 \mid w_t\right]. \tag{616}$$

    Note that the random variable $\hat{z}_t$ only depends on the previous actions through the random variable w_t.
  - So now, we just have to compute the expected local norm:

$$\mathbb{E}_{a_t \sim w_t}\left[\sum_{a=1}^{d} w_{t,a}\hat{z}_{t,a}^2 \mid w_t\right] = \sum_{a=1}^{d} w_{t,a}^2\left(\frac{z_{t,a}^2}{w_{t,a}^2}\right) \le d, \tag{617}$$

    since $z_t \in [0,1]^d$.
  - Note that it is crucial that we use local norms here; the generic bound (540) of $\|z_t\|_\infty^2$ is not good enough because $\|z_t\|_\infty$ is unbounded.
  - Minimizing the regret bound with respect to η, we get $2\sqrt{d\log(d)T}$ with $\eta = \sqrt{\log(d)/(dT)}$.

- Note that we have only gotten results in expectation (over randomness of the learner $a_t \sim w_t$). To get high probability results, we also need to control the variance of $\hat{z}_t$. To do this, we modify the EG algorithm by smoothing w_t with the uniform distribution over actions. This results in the standard Exp3 algorithm.

5.14 Adversarial bandits: online gradient descent (Lecture 16)

- In the online convex optimization setting, we are presented with a sequence of functions $f_1, \dots, f_T$. Importantly, we could compute subgradients $z_t \in \partial f_t(w_t)$, which allowed us to update our weight vector using online gradient descent (OGD). In the bandit setting, suppose we can only query function values of f_t. Can we still minimize regret?

- Intuitively, something should be possible. In one dimension, the gradient of f_t at a point w can be approximated by $\frac{f_t(w+\delta)-f_t(w-\delta)}{2\delta}$. But this requires two function evaluations, which is not permitted in the bandit setting: you only get to administer one treatment to a patient, not two! Can we do it with one function evaluation? Can we generalize to high dimensions? The answer to both questions is yes. The original idea is from Flaxman et al. (2005); we follow the exposition in Shalev-Shwartz (2011).

- For a function f, define a smoothed version of f as follows:

$$\tilde{f}(w) = \mathbb{E}_{v \sim U_{\le 1}}[f(w + \delta v)], \tag{618}$$

where $U_{\le 1}$ is the uniform distribution over the unit ball (the set of vectors v with $\|v\|_2 = 1$). There are two important properties:

  - We can compute the gradient of $\tilde{f}$ (this is the key step):

$$\boxed{\nabla \tilde{f}(w) = \mathbb{E}_{v \sim U_1}\left[\frac{d}{\delta} f(w + \delta v) v\right].} \tag{619}$$

    This is due to Stokes' theorem, but the intuition can be seen in 1D due to the Fundamental Theorem of calculus:

$$\tilde{f}(w) = \mathbb{E}[f(w + \delta v)] = \frac{\int_{-\delta}^{\delta} f(w+t)dt}{2\delta} = \frac{F(w+\delta) - F(w-\delta)}{2\delta}, \tag{620}$$

    where F is the antiderivative of f and thus

$$\tilde{f}'(w) = \frac{f(w+\delta) - f(w-\delta)}{2\delta}. \tag{621}$$

    In the general setting, if we draw $v \sim U_1$ from the unit sphere, then $\frac{d}{\delta} f(w + \delta v) v$ is an unbiased estimate of $\nabla \tilde{f}(w)$.

  - We have that $\tilde{f}$ well approximates f:

$$|\tilde{f}(w) - f(w)| \le |\mathbb{E}[f(w + \delta v) - f(w)]| \le L\delta, \tag{622}$$

    where L is the Lipschitz constant of f.

Now we are ready to define the algorithm:

- **Algorithm 8 (Bandit-OGD)**

  - For $t = 1, \dots, T$:
    - Set $w_t = \arg\min_{w \in S} \|w - \eta\theta_t\|_2$ (same as OGD)
    - Pick random noise $v_t \sim U_1$
    - Predict $\tilde{w}_t = w_t + \delta v_t$
    - Compute $z_t = \frac{d}{\delta} f_t(\tilde{w}_t) v_t$
    - Update $\theta_{t+1} = \theta_t - z_t$

- **Theorem 39 (regret bound bandit online gradient descent)**

  - Let $f_1, \dots, f_T$ be a sequence of convex functions.

  - Let $B = \max_{u \in S} \|u\|_2$ be the magnitude of an expert.
  - Let $F = \max_{u \in S, t \in [T]} f_t(u)$ be the maximum function value.
  - Then Bandit-OGD obtains the following regret bound:

$$\mathbb{E}[\text{Regret}(u)] = \mathbb{E}\left[\sum_{t=1}^{T} [f_t(\tilde{w}_t) - f_t(u)]\right] \tag{623}$$

$$\leq 3L\delta T + \frac{B^2}{2\eta} + \frac{\eta}{2} T d^2 (F/\delta + L)^2, \tag{624}$$

    where the expectation is over the learner's random choice (nature is still adversarial).

- Let us set δ and η to optimize the bound.
  - First, we optimize the step size η: setting $\eta = \frac{B}{d(F/\delta + L)\sqrt{T}}$ yields a regret bound of

$$3L\delta T + Bd(F/\delta + L)\sqrt{T}. \tag{625}$$

  - Second, we optimize the amount of smoothing δ: setting $\delta = \sqrt{\frac{BdF}{3L}} T^{-1/4}$ yields a final regret bound of

$$\mathbb{E}[\text{Regret}] \leq \sqrt{12BdFLT^{3/4} + BdL\sqrt{T}} \tag{626}$$

$$= \boxed{O(\sqrt{BdFL}T^{3/4}),} \tag{627}$$

    where in the last equality, we're thinking of the dependence on T as primary.
  - Note that our regret bound of $O(T^{3/4})$ has a worse dependence than the $O(T^{1/4})$ that we're used to seeing. This comes from the fact that we're only getting an unbiased estimate of the gradient of the a *smoothed function* $\hat{f}$, not f. And as usual, we have a $\sqrt{d}$ dependence on the dimension since we are in the bandit setting and only get $1/d$-the the information of a full gradient.

- Proof of Theorem 39
  - Our analysis of plain OGD Theorem 30 yields the following regret bound when applied to w_t and linear functions $x \mapsto x \cdot z_t$:

$$\sum_{t=1}^{T} [w_t \cdot z_t - u \cdot z_t] \leq \frac{1}{2\eta} \|u\|_2^2 + \frac{\eta}{2} \sum_{t=1}^{T} \|z_t\|_2^2. \tag{628}$$

  - First, taking expectations on both sides, using the fact that $\tilde{f}_t$ is convex with $\nabla \tilde{f}_t(w_t) = \mathbb{E}[z_t]$ for the LHS, expanding the definition of z_t on the RHS:

$$\sum_{t=1}^{T} \mathbb{E}[\tilde{f}_t(w_t) - \tilde{f}_t(u)] \leq \frac{1}{2\eta} \|u\|_2^2 + \frac{\eta}{2} \sum_{t=1}^{T} \mathbb{E}[\|(d/\delta) f_t(\tilde{w}_t) v_t\|_2^2]. \tag{629}$$

  - Second, let's handle the gradient: Since f_t is L-Lipschitz, we have that

$$f_t(\tilde{w}_t) \le f_t(w_t) + L\delta \le F + L\delta. \tag{630}$$

    Plugging this inequality into (629):

$$\sum_{t=1}^{T} \mathbb{E}[\tilde{f}_t(w_t) - \tilde{f}_t(u)] \le \frac{B^2}{2\eta} + \frac{\eta}{2}\frac{d^2}{\delta^2}(F + L\delta)^2. \tag{631}$$

  - Finally, we need to relate $\tilde{f}_t(w_t)$ to $f_t(\tilde{w}_t)$. Using the Lipschitz property of f_t, we have:

$$f_t(\tilde{w}_t) - f_t(u) \le f_t(w_t) - f_t(u) + L\delta \le \tilde{f}_t(w_t) - \tilde{f}_t(u) + 3L\delta. \tag{632}$$

    Plugging this into (631) completes the proof.

5.15 Stochastic bandits: upper confidence bound (UCB) (Lecture 16)

- Let us return to the learning from expert advice setting where at each iteration, we are choosing an arm $a_t \sim w_t$ and suffering loss z_{t,a_t}. So far, we have assumed that the losses z_t were adversarial. Now, let us now consider the **stochastic setting**, where z_t's are drawn from an (unknown) distribution. By leveraging the probabilistic structure of the problem, we'll show in this section that we can improve the regret bounds from $O(\sqrt{T})$ to $O(\log T)$.

- We will show an algorithm called upper confidence bound (UCB) (Lai and Robbins, 1985; Auer et al., 2002) that is based on the principle of *optimism in the face of uncertainty*: we maintain upper bounds on the reward of each action and choose the action that maximizes the best possible payoff. The idea is that we either get very low loss (which is great), or we learn something.

- Let us define the notation for the stochastic bandits setting:

  - Let $r_1, \dots, r_T$ be an i.i.d. sequence of reward vectors,[22] where each $r_t \in [0,1]^d$
  - Define $\mu \overset{\text{def}}{=} \mathbb{E}[r_t]$ to be the average rewards (μ doesn't depend on t).
  - Define $j^* \overset{\text{def}}{=} \arg\max_{j=1}^{d} \mu_j$ be the best action.
  - Define $\Delta_j \overset{\text{def}}{=} \mu_{j^*} - \mu_j > 0$ be the gap between action j and the best action j^*. Let $\Delta \overset{\text{def}}{=} \min_{j \ne j^*} \Delta_j$ the smallest gap.

[22] In the stochastic bandits literature, we typically maximize reward rather than minimize loss, so think of $r_t = -z_t$.

- Let $a_1, \dots, a_T \in [d]$ be the sequence of actions taken by the learner.
- Let $S_j(t) \overset{\text{def}}{=} \{i \in [t] : a_i = j\}$ be the set of time steps i up to time t where we took action j.
- Let $N_j(t) = |S_j(t)|$ be the number of times we took action j.
- We are interested in minimizing the expected regret:

$$\mathbb{E}[\text{Regret}] = \sum_{t=1}^{T} \mathbb{E}[r_{t,j^*} - r_{t,a_t}] \tag{633}$$

$$= \sum_{j=1}^{d} \mathbb{E}[N_j(T)]\Delta_j, \tag{634}$$

  where in the last inequality, we've rewritten the regret to sum over all possible actions; each of the $N_j(T)$ times we take action j, we add Δ_j to the regret. Note that at each time step t, r_t is independent of a_t, which is a function of the past.

- Now let's design an algorithm for solving the stochastic bandits problem. The way to think about it is that we're just estimating the mean μ in an online manner. At each time step t, we have $N_j(t)$ i.i.d. draws of action j, from which we can form the empirical estimate

$$\hat{\mu}_j(t) = \frac{1}{|S_j(t)|} \sum_{i \in S_j(t)} r_{i,j}. \tag{635}$$

  Provided that $S_j(t)$ grows to infinity, we can estimate the means: $\hat{\mu}_j(t) \xrightarrow{P} \mu_j$.

- The difference from mean estimation is that we are also trying to *exploit* and will choose actions j that are more likely to have higher reward.

- By Hoeffding's inequality,

$$\mathbb{P}[\hat{\mu}_j(t) \le \mu_j - \epsilon] \le \exp\left(-2N_j(t)\epsilon^2\right). \tag{636}$$

  In other words, with probability at least $1 - \delta$,

$$\mu_j \le \hat{\mu}_j(t) + \sqrt{\frac{\log(1/\delta)}{2N_j(t)}}. \tag{637}$$

  This motivates the following algorithm:

- **Algorithm 9 (upper confidence bound (UCB))**

  - Play each of the d actions once.

- For each subsequent time step $t = d+1, \dots, T$:
  - ∗ Choose action

$$a_t = \arg\max_{1 \le j \le d} \left\{ \hat{\mu}_j(t-1) + \sqrt{\frac{2\log t}{N_j(t-1)}} \right\}, \tag{638}$$

- **Theorem 40 (regret of upper confidence bound (UCB))**
  - The UCB algorithm obtains the following expected regret bound:

$$\boxed{\mathbb{E}[\text{Regret}] \le \frac{8d\log T}{\Delta} + 5d.} \tag{639}$$

- Proof sketch:
  - The main idea is to upper bound the number of times an suboptimal action $j \neq j^*$ was taken. This happens when the following event A happens:

$$\hat{\mu}_j + \epsilon_j \ge \hat{\mu}_{j^*} + \epsilon_{j^*}. \tag{640}$$

  - FIGURE: [$\hat{\mu}_j, \mu_j, \hat{\mu}_{j^*}, \mu_{j^*}$ on a line with Δ_j gap]
  - Recall that the separation between μ_j and μ_{j^*} is Δ_j. If the actual values $\hat{\mu}_j$ and $\hat{\mu}_{j^*}$ are within $O(\Delta_j)$ of their means, then we're all set.
  - The probability of that not happening after m rounds is $e^{-O(m\Delta^2)}$ as given by Hoeffding's inequality. If we wait m rounds and look at the expected number of times j was chosen:

$$m + \sum_{t=1}^{T} t^2 e^{-O(m\Delta_j^2)}, \tag{641}$$

    where the t^2 comes from taking a sloppy union bound over pairs of time steps (because we don't know which time steps we're comparing so just try all pairs). If we set $m = O\left(\frac{\log T}{\Delta_j^2}\right)$, then we can make the second term be a constant. Summing the first term over j and multiplying Δ_j yields the result.
  - Note that we're being pretty sloppy here. For the full rigorous proof, see Auer et al. (2002).

- Interpretation: that we have a logarithmic dependence on T, and a necessary linear dependence on the number of actions d. There is also a dependence on Δ^{-1}; a smaller Δ means that it's harder to distinguish the best action from the others.

5.16 Stochastic bandits: Thompson sampling (Lecture 16)

- In UCB, our principle was optimism in the face of uncertainty. We now present an alternative method called Thompson sampling, which dates back to Thompson (1933). The principle is *choose an action with probability proportional to it being optimal*, according to a Bayesian model of how the world works.

- Setup
  - Prior $p(\theta)$ over parameters θ
  - Probability over rewards given actions: $p(r \mid \theta, a)$

  The Thompson sampling principle leads to a simple, efficient algorithm:

- Algorithm
  - For each time $t = 1, \dots, T$:
    - Sample a model from the posterior: $\theta_t \sim p(\theta \mid a_{1:t-1}, r_{1:t-1})$
    - Choose the best action under that model: $a_t = \arg\max_a p(r \mid \theta_t, a)$.

- Example for Bernoulli rewards
  - Assume each reward $r_j \sim \text{Bernoulli}(\mu_j)$ (here the parameters θ are the means μ).
  - For each action $j = 1, \dots, d$
    - Let the prior be $\theta_j \sim \text{Beta}(\alpha, \beta)$
    - Keep track of the number of successes s_j and failures f_j.
  - For each time $t = 1, \dots, T$:
    - Sample $\theta_j \sim \text{Beta}(\alpha + s_j, \beta + f_j)$ for each $j = 1, \dots, d$
    - Choose $a_t = \arg\max_j \theta_j$ and observe reward r_t.
    - If $r_t = 1$ then increment s_{a_t}; else increment f_{a_t}.

  Despite its long history, Thompson sampling was not very popular until recently, where rigorous regret bounds have appeared and people realize that it outperforms UCB in many settings. Agrawal and Goyal (2012) prove the following:

- **Theorem 41 (regret for Thompson sampling)**
  - For any $\epsilon > 0$, the expected regret of Thompson sampling us upper bounded by:

$$\mathbb{E}[\text{Regret}] \le (1+\epsilon) \sum_{j:\Delta_j > 0} \frac{\Delta_j \log T}{\text{KL}\left(\mu_j \| \mu^*\right)} + O\left(\frac{d}{\epsilon^2}\right). \tag{642}$$

- One strength of Thompson sampling is that it's a principle that allows one to easily generalize the algorithm to structured settings such as reinforcement learning.

- One weakness of Thompson sampling is that it does not choose actions that takes into account the value of information gained (especially important for structured settings). As an example (from Ian Osband), consider $\theta \in \{e_1, \dots, e_d\}$ be an unknown indicator vector, and suppose $a \in \{0, 1\}^d$ and the reward is is $a \cdot \theta - 0.0001\|a\|_1$. The optimal strategy would be to do binary search to find θ in $O(\log d)$ time, but Thompson sampling would just choose one a at a time, which would take $O(d)$ time. More expensive methods such as Gittins index performs the actual dynamic programming and have better guarantees but are computationally more expensive.

5.17 Summary (Lecture 16)

- This concludes our tour of online learning and its extensions to the bandit setting.
- Our measure of success is getting low **regret**, the difference between the learner's cumulative losses and the best *fixed* expert's cumulative losses. In particular, we hope for sublinear regret: Regret $= o(T)$.
- Without no additional assumptions (even with two experts), any deterministic algorithm must have Regret $= \Omega(T)$ (fail).
- In the realizable setting (some expert is perfect), the majority algorithm achieves $O(\log d)$ regret (constant in T).
- We started with the **follow the leader (FTL)**, and saw that it worked for quadratic loss functions (Regret $= O(\log T)$), but failed for linear loss functions (Regret $= \Omega(T)$).
- Inspecting the regret bounds reveals that in order to get low regret, we need to have a *stable* learner, in the sense that w_t and w_{t+1} should be close (according to some notion of proximity).
- This motivated us to look at **follow the regularized leader (FTRL)**, where we add a quadratic regularizer, which gave us a regret bound with two terms: (i) a bias-like term (value of regularizer applied to the best expert) and (ii) a variance-like term (sum of the norm of the subgradients). Balancing the two by controlling the amount of regularization (inverse step size) yields Regret $= O(\sqrt{T})$.
- If our loss functions were non-linear, we could linearize using the subgradient. Coupled with a quadratic regularizer, we get the **online subgradient descent (OGD)** algorithm. We also showed that it suffices to analyze linear functions, since they result in the most regret.
- We looked at learning with expert advice, and got regret bounds of $O(\sqrt{dT})$, where d is the number of experts. Inspired by the logarithmic dependence on d in the majority algorithm, this motivated us to consider different regularizers.

- The general algorithm that deals with different regularizers is **online mirror descent (OMD)**. We analyzed this algorithm using Fenchel duality and Bregman divergences, which allowed us to look at arbitrary regularizers through the lens of the mapping between w_t and θ_t. Specializing to learning with expert advice, we get a $O(\sqrt{\log(d)T})$ regret bound with the entropic regularizer, resulting in the **exponentiated gradient (EG)** algorithm.

- By exploiting properties of exponentiated gradient, we can perform a refined analysis again using **local norms** to achieve stronger results, matching the $O(\log d)$ regret in the realizable setting.

- In the bandit setting with partial feedback, we get $O(\sqrt{d\log(d)T})$ regret again using local norms to control the size of the now unbounded subgradients. The general principle is to find an estimate of the loss vector z_t and run existing online learning algorithms (which are quite tolerant of noise).

- Finally, we showed that learners with low regret in the online setting directly lead to learners with low **expected risk** in the batch setting via an online-to-batch conversion.

5.18 References

- Shalev-Shwartz, 2012: Online Learning and Online Convex Optimization (survey paper)

  - This is a nice introduction to online learning, on which much of these online learning notes are based.

6 Neural networks (skipped in class)

6.1 Motivation (Lecture 16)

- One major motivating factor for studying neural networks is that they have had a lot of empirical success and attention in recent years. Here is the general recipe:
  - Train on large amounts of data (Krizhevsky, 2012: 1 million examples).
  - Use a very large neural network (Krizhevsky, 2012: 60 million parameters).
  - Running simple stochastic gradient descent (with some (important) tweaks such as step size control, sometimes momentum, dropout), and wait a moderately long period of time (a week).
  - Get state-of-the-art results across multiple domains.
    - Object recognition (ImageNet): reduce error from 25.7% to 17.0% (Krizhevsky, 2012)
    - Speech recognition (Switchboard): reduce error from 23% to 13% (Microsoft, 2009–2013)

  Error reductions on these tasks/datasets are significant, since these are large realistic datasets (unlike MNIST) on which many serious people have tried to improve accuracy.

- However, in theory, neural networks are poorly understood:
  - The objective function is non-convex. SGD has no reason to work.
  - What about the particular hypothesis class of neural networks makes them perform well across so many tasks? What types of functions are they good at representing?

  This makes neural networks an important but challenging area to do new theoretical research.

- Compared to the theory of online learning, uniform convergence, or kernel methods, the theory of neural networks ("why/when do they work?") is spotty at best. In this lecture, we will attempt lay out the high-level important questions and provide preliminary thrusts towards answering them, which in turn produces a series of more concrete open questions.

- A caveat: there are many theorems that one could prove about neural networks. We will write some of these down. However, most of these theorems only nibble off the corners of the problem, and do not really answer the hard questions. But that's where we are. We will try to point out the interpretation of these results, which is especially important to recognize in this prehistoric stage of understanding.

6.2 Setup (Lecture 16)

- Definition
  - Let $x \in \mathbb{R}^d$ be an input.
  - A neural network defines a non-linear function on x. For concreteness, let us focus on two-layer neural networks (which means it has one hidden layer).
  - Let $\sigma : \mathbb{R} \mapsto \mathbb{R}$ be a non-linear function (called an activation or transfer function). Examples:
    - * Logistic: $z \mapsto \frac{1}{1+e^{-z}}$ (maps real numbers monotonically to $[0,1]$)
    - * Hyperbolic tangent (tanh): $z \mapsto \frac{e^z - e^{-z}}{e^z + e^{-z}}$ (maps real numbers monotonically to $[-1,1]$)
    - * Rectified linear: $z \mapsto \max(0,z)$ (truncates negative numbers to zero)

    We will extend σ to operate on vectors elementwise:

    $$\sigma(x) \stackrel{\text{def}}{=} [\sigma(x_1), \dots, \sigma(x_d)]. \tag{643}$$

  - An artificial **neural network** (not to be confused with and completely different from the thing that's in your brain) can be described by a function $f : \mathbb{R}^d \mapsto \mathbb{R}$ which has the following form:

    $$f_\theta(x) = \sum_{i=1}^{m} \alpha_i \sigma(w_i \cdot x + b_i). \tag{644}$$

    Here, the parameters are $\theta = (\alpha_{1:m}, w_{1:m}, b_{1:m})$. In matrix notation:

    $$f_\theta(x) = \alpha \cdot \sigma(Wx + b), \tag{645}$$

    where the i-th row of W is $w_i \in \mathbb{R}^d$.
  - A useful way to think about a neural network is that the first layer $\sigma(Wx+b)$ computes some non-linear features of the input and the second layer simply performs a linear combination. So you can think of a neural network as parametrizing the features as well as the weights α.
  - FIGURE: [draw neural network]
- In practice, people use many layers (for example, in speech recognition, seven is not uncommon). A three-layer neural network looks like this:

  $$f_\theta(x) = \sigma(W_2 \sigma(W_1 x + b_1) + b_2). \tag{646}$$

  People also use convolutional neural networks in which W has additional low-dimensional structure. Recurrent neural networks are good for representing time series. We will not discuss these here and focus on the two-layer neural networks in this lecture.

- Let $\mathcal{F} = \{f_\theta\}$ be the class of all neural networks as we range over values of θ.
- We can use $\mathcal{F}$ for regression or classification in the usual way by minimizing the loss on the training set. For regression:

$$\hat{f}_{\mathrm{ERM}} = \arg\min_{f_\theta} \hat{L}(\theta), \quad \hat{L}(\theta) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n}(f_\theta(x^{(i)}) - y^{(i)})^2. \tag{647}$$

- The surprising thing about neural networks is that people use stochastic gradient descent (online gradient descent where we choose $i \in \{1, \dots, n\}$ at each step randomly), which converges to $\hat{f}_{\mathrm{SGD}}$. Since the objective function $\hat{L}$ is non-convex, $\hat{f}_{\mathrm{SGD}}$ will be different from $\hat{f}_{\mathrm{ERM}}$. so there is an additional **optimization error** in addition to the usual approximation and estimation errors. The decomposition must be taken with a grain of salt, because having suboptimal optimization effectively restricts the hypothesis class, which actually improves estimation error (a simple example is early stopping). So approximation and optimization error are perhaps difficult to tease apart in methods whose success is tied to an algorithm not just an optimization problem.

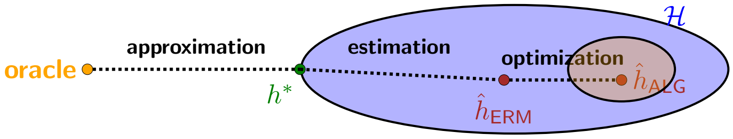

Figure 10: Cartoon showing error decomposition into approximation, estimation, and optimization errors.

6.3 Approximation error (universality) (Lecture 16)

- Question: which functions can be represented by two-layer neural networks? Answer: basically all of them.
- We saw that Gaussian kernels were universal (Theorem 27) in the sense that they are dense in $C_0(\mathcal{X})$, the set of all continuous bounded functions on $\mathcal{X}$.
- Functions defined by (Gaussain) kernels can be represented in terms of the partial kernel evaluations

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x). \tag{648}$$

or using Fourier features:

$$f(x) = \sum_{i=1}^{m} \alpha_i e^{-i\langle w_i, x\rangle}. \tag{649}$$

Note that these expressions resemble neural networks (644). All of these are linear combinations of some non-linear basis functions. So we might expect that neural networks would be universal. The following theorem answers affirmatively, which is not hard to show:

- **Theorem 42 (neural networks are universal)**
  - Consider $\mathcal{X} = [0,1]^d$.
  - Then the class of two-layer neural networks $\mathcal{F}$ is dense in $C_0(\mathcal{X})$ in the uniform metric: for every $f^* \in C_0(\mathcal{X})$, and $\epsilon > 0$, there exists an $f_\theta \in \mathcal{F}$ such that $\max_{x\in\mathcal{X}} |f^*(x) - f_\theta(x)| \le \epsilon$.

- Proof of Theorem 42:
  - Fix $f^* \in \mathcal{F}$. Because f^* is continuous over a compact set ($\mathcal{X}$), it is uniformly continuous, which means we can find a width δ such that $|x_1 - x_2| \le \delta$ implies $|f(x_1) - f(x_2)| \le \epsilon$.
  - Grid $\mathcal{X}$ into regions R_j which are of the form: $R_j = \{x : |x - x_j|_\infty \le \delta\}$.
  - FIGURE: [1D function]
  - Note that $z \mapsto \sigma(Cz)$ converges to the step function as $C \to \infty$.
  - Let us construct $3d$ hidden units for each R_j. For $d = 1$, let $R_j = [x_j - \delta, x_j + \delta] = [a_j, b_j]$.

    $$f^*(x_j)\sigma(C(x - a_j)) + f^*(x_j)\sigma(C(b_j - x)) - f^*(x_j)\sigma(0). \tag{650}$$

    This function is approximately $f^*(x_j)$ on R_j and zero elsewhere, So we we do this for all regions, then we can approximate f^* uniformly well.

- Note that the number of hidden units is exponential in d, since the number of regions is $O((\frac{1}{\delta})^d)$. This is just a glorified nearest neighbors. So while this theorem holds, it doesn't really provide much insight into why neural networks work well and generalize.

- Depth: One argument made in favor of deep neural networks is that they more compactly represent the data compared to a shallower one. To get intuition, let us think about networks that perform arithmetic operations on the raw inputs via addition and multiplication. Such networks define polynomial functions in a compact way, since internal nodes represent factors. For example, the polynomial

$$f(x) = x_1x_2^2x_3 + x_1x_2x_3x_4 + x_2^2x_3^3 + x_2x_3^3x_4 \tag{651}$$

$$= (a + b)(b + c), \tag{652}$$

where $a = x_1x_2$, $b = x_2x_3$, and $c = x_3x_4$ correspond to internal nodes.

6.4 Generalization bounds (Lecture 16)

- The excess risk $L(\hat{h}) - L(h^*)$ captures the estimation error (the generalization ability) of an estimator $\hat{h}$. Recall that the excess risk is controlled by the Rademacher complexity (218) of the function class, so it suffices to study the Rademacher complexity. We will perform the analysis for two-layer neural networks.

- **Theorem 43 (Rademacher complexity of neural networks)**

  - Let $x \in \mathbb{R}^d$ be the input vector with $\|x\|_2 \le C_2$.
  - Let $w_j \in \mathbb{R}^d$ be the weights connecting to the j-th hidden unit, $j = 1, \dots, m$
  - Let $\alpha \in \mathbb{R}^m$ be the weights connecting the hidden units to the output
  - Let $h : \mathbb{R} \to \mathbb{R}$ be a non-linear activation function with Lipschitz constant 1 such that $h(0) = 0$; examples include
    - ∗ Hyperbolic tangent: $h(z) = \tanh(z)$
    - ∗ Rectified linear: $h(z) = \max\{0, z\}$
  - For each set of weights (w, α), define the predictor:

$$f_{w,\alpha}(x) = \sum_{j=1}^{m} v_j h(w_j \cdot x). \tag{653}$$

  - Let $\mathcal{F}$ be the class of prediction functions where the weights are bounded:

$$\mathcal{F} = \{f_{w,\alpha} : \|\alpha\|_2 \le B_2', \|w_j\|_2 \le B_2 \text{ for } j = 1, \dots, m\}. \tag{654}$$

  - Then

$$R_n(\mathcal{F}) \le \frac{2B_2 B_2' C_2 \sqrt{m}}{\sqrt{n}}. \tag{655}$$

- Proof of Theorem 43:

  - The key is that the composition properties of Rademacher complexity aligns very nicely with the layer-by-layer compositionality of neural networks.
  - The function mapping the input layer to a hidden unit is just a linear function:

$$R_n(\{x \mapsto w \cdot x : \|w\|_2 \le B_2\}) \le \frac{B_2 C_2}{\sqrt{n}}. \tag{656}$$

  - Sending each of these through the 1-Lipschitz non-linearity does not change the upper bound on the complexity:

$$R_n(\{x \mapsto h(w \cdot x) : \|w\|_2 \le B_2\}) \le \frac{B_2 C_2}{\sqrt{n}}. \tag{657}$$

For convenience, define $\mathcal{G}$ as the set of vector-valued functions mapping the input to the hidden activations, as w ranges over different values:

$$g(x) = [h(w_1 \cdot x), \dots, h(w_m \cdot x)]. \tag{658}$$

– Now let us handle the second layer:

$$R_n(\mathcal{F}) \le \mathbb{E}\left[\sup_{\|\alpha\|_2 \le B_2', g \in \mathcal{G}} \frac{1}{n}\sum_{i=1}^n \sigma_i(\alpha \cdot g(Z_i)).\right] \tag{659}$$

– Applying Cauchy-Schwartz:

$$R_n(\mathcal{F}) \le B_2' \mathbb{E}\left[\sup_{g \in \mathcal{G}} \left\| \frac{1}{n}\sum_{i=1}^n \sigma_i g(Z_i)\right\|_2\right]. \tag{660}$$

– Using the fact that if $a \in \mathbb{R}^m$ then $\|a\|_2 \le \sqrt{m}\max_{1 \le j \le m} |a_j|$:

$$R_n(\mathcal{F}) \le B_2' \sqrt{m}\, \mathbb{E}\left[\max_{1 \le j \le m} \sup_{\|w_j\|_2 \le B_2} \left| \frac{1}{n}\sum_{i=1}^n \sigma_i h(w_j \cdot Z_i)\right|\right]. \tag{661}$$

– This is the same as taking the sup over a single generic w subject to $\|w\|_2 \le B_2$:

$$R_n(\mathcal{F}) \le B_2' \sqrt{m}\, \mathbb{E}\left[\sup_{\|w\|_2 \le B_2} \left| \frac{1}{n}\sum_{i=1}^n \sigma_i h(w \cdot Z_i)\right|\right]. \tag{662}$$

Note that the expectation on the RHS is almost the Rademacher complexity of a single unit (657), but with absolute values. In some parts of the literature, Rademacher complexity is actually defined with absolute values (including the original Bartlett/Mendelson (2002) paper), but the absolute value version is a bit harder to work with in general. For that definition, the Lipschitz composition property still holds with an additional factor of 2 but requires that $h(0) = 0$ (see point 4 of Theorem 12 of Bartlett/Mendelson (2002)). So therefore we can adapt (657) and plug it into (662) to obtain:

$$R_n(\mathcal{F}) \le B_2' \sqrt{m}\left(\frac{2B_2 C_2}{\sqrt{n}}\right). \tag{663}$$

- Interpretation: the bottom line of this theorem is that neural networks with non-linearities which are smooth (Lipschitz) will generalize regardless of convexity.

6.5 Approximation error for polynomials (Lecture 16)

- There are two problems with the previous analyses:
  - First, we defined approximation error with respect to the set of all bounded continuous functions, which is clearly much too rich of a function class, and as a result we get exponential dependence on the dimension d.
  - Second, we have not yet said anything about optimization error.
- We will now present part of an ICML paper (Andoni/Panigrahy/Valiant/Zhang), which makes some progress on both points. We will focus on using neural networks to approximate bounded-degree polynomials, which is a reasonable class of functions which itself approximates Lipschitz functions. Further, we will show that choosing weights W *randomly* and optimizing α (which is convex) is sufficient.
- Setup
  - Monomial $x^J = x^{J_1} \cdots x^{J_d}$ with degrees $J = (J_1, \dots, J_d)$
    - ∗ Example: $J = (1, 0, 7)$, $x^J = x_1 x_3^7$
  - A polynomial is $f(x) = \sum_J b_J x^J$
    - ∗ Example: $b_{(1,0,7)} = 3$ and $b_{(0,1,0)} = -5$, $f(x) = 3x_1x_3^7 - 5x_2$
  - Degree of polynomial: $\deg(f) = \max_{b_J \neq 0} |J|$, $|J| \overset{\text{def}}{=} \sum_{i=1}^d J_i$
- Inner product structure
  - Let $p^*(x)$ is the uniform distribution over $\mathbb{C}(R)^d$, where $\mathbb{C}(R)$ is set of complex numbers c with norm at most R ($|c| = \sqrt{c\bar{c}} \leq R$).
  - Define the inner product over functions and the associated norm:

$$\langle f, g \rangle_{p^*} \overset{\text{def}}{=} \mathbb{E}_{x \sim p^*}[f(x)\overline{g(x)}] \tag{664}$$

$$\|f\|_{p^*} \overset{\text{def}}{=} \sqrt{\langle f, f \rangle_{p^*}}. \tag{665}$$

- Neural network as an infinite polynomial
  - Let $\sigma(z) = \sum_{j \geq 0} a_j z^j$
    - ∗ Example: for $\sigma(z) = e^z$, $a_j = \frac{1}{j!}$
  - For $w \in \mathbb{C}^d$, we define the basis function, which can be written as a weighted combination of monomials x^J:

$$\phi^w(x) \overset{\text{def}}{=} \sigma(w \cdot x) = \sum_{j \geq 0} a_j \left(\sum_{k=1}^d w_k x_k \right)^j \overset{\text{def}}{=} \sum_J a_J w^J x^J. \tag{666}$$

  Here, $w^J = \prod_{k=1}^d w^{J_k}$ is the product of the base weights.

- The neural network is

$$f(x) = \sum_{i=1}^{m} \alpha_i \phi^{w_i}(x). \tag{667}$$

- Note that weights $\{w^J\}$ are orthogonal in the following sense:

$$\mathbb{E}_{w \sim \mathbb{C}(R)^d}[w^J \overline{w^{J'}}] = \begin{cases} R^{2|J|} & \text{if } J = J' \\ 0 & \text{otherwise.} \end{cases} \tag{668}$$

  This stems from the fact that each $w_j \sim \mathbb{C}(R)$ means $w_j = e^{it}$ where $t \sim \text{Uniform}([0, 2\pi])$; and $\mathbb{E}_{t \sim \text{Uniform}([0,2\pi])}[e^{iat}\overline{e^{ibt}}] = \mathbb{I}[a = b]$.

- **Theorem 44 (neural networks approximate bounded-degree polynomials)**
  - Choose $w_1, \dots, w_m$ i.i.d. from $\mathbb{C}(1/\sqrt{d})^d$.
  - For any polynomial f^* of degree q and norm $\|f^*\| = 1$, exists $\alpha_{1:m}$ with $\|\alpha_{1:m}\|_2 = \sum_{i=1}^{m} |\alpha_i|^2 = O(d^{2q}/m)$ such that

$$\left\| \sum_{i=1}^{m} \alpha_i \phi^{w_i} - f^* \right\|_{p^*} = O_p\left(\sqrt{\frac{d^{2q}}{m}}\right). \tag{669}$$

- Proof of Theorem 44:
  - Let the target function be:

$$f^*(x) = \sum_J b_J x^J. \tag{670}$$

  - Construct an unbiased estimate of f^*:
    - For any $x \in \mathbb{C}^d$ and $J \in \mathbb{N}^d$,

$$\mathbb{E}_{w \sim \mathbb{C}(R)^d}[\phi^w(x)\overline{w^J}] = a_J R^{2|J|} x^J, \tag{671}$$

      which follows from applying (668) to the expansion (666) and noting that all random coefficients except w^J drop out.
    - Define the coefficients

$$T(w) = \sum_J \frac{b_J \overline{w^J}}{a_J R^{2|J|}}. \tag{672}$$

    - Then by construction, we have an unbiased estimate of the target polynomial:

$$\mathbb{E}_{w \sim \mathbb{C}(R)^d}[T(w)\phi^w(x)] = f^*(x). \tag{673}$$

- Now we would like to control the variance of the estimator.
  - First, define the error:

$$\eta(x) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^{m} T(w_i)\phi^{w_i}(x) - f^*(x). \tag{674}$$

  - We can show that the variance falls as $1/m$ due to independence of $w_1, \dots, w_m$:

$$\mathbb{E}_{w_1,\dots,w_m \sim \mathbb{C}(R)^d}[\|\eta\|_{p^*}^2] \le \frac{1}{m}\mathbb{E}_{w\sim\mathbb{C}(R)^d}[|T(w)|^2\|\phi^w\|_{p^*}^2]. \tag{675}$$

  - To bound the coefficients:
    - Let $a(q) = \min_{|J|\le q} |a_J|$ (property of the non-linear activation function)
    - Let $\|f^*\|_1 = \sum_J |b_J|$ (property of the target function)
    - Let $q = \deg(f^*)$.
    - Assume $R \le 1$.
    - Then

$$\begin{aligned} |T(w)| &\le \sum_J \left| \frac{b_J \overline{w^J}}{a_J R^{2|J|}} \right| & (676)\\ &\le \sum_J \left| \frac{b_J}{a_J R^{|J|}} \right| \quad [\text{since } |\overline{w^J}| \le R^{|J|}] & (677)\\ &\le \frac{\sum_J |b_J|}{a(q)R^q} \quad [\text{since } |J| \le q, a(q) \le |a_J|] & (678)\\ &= \frac{\|f^*\|_1}{a(q)R^q}. & (679)\end{aligned}$$

  - Define the following bounds on the variance of the basis functions ϕ^w and the 1-norm of the target function f^*, respectively:

$$\begin{aligned} \beta(q,R) &\stackrel{\text{def}}{=} \mathbb{E}_{w\sim\mathbb{C}(R)^d}\left[\frac{\|\phi^w\|_{p^*}^2}{R^{2q}}\right], & (680)\\ \gamma(q) &\stackrel{\text{def}}{=} \max_{\|f^*\|_{p^*}=1} \|f^*\|_1. & (681)\end{aligned}$$

    Then we have the following result (by plugging quantities in and applying concavity of $\sqrt{\cdot}$):

$$\mathbb{E}_{w_1,\dots,w_m\sim\mathbb{C}(R)^d}[\|\eta\|_{p^*}] \le \sqrt{\frac{\gamma(q)^2\beta(q,R)}{a(q)^2 m}}. \tag{682}$$

- Finally, we specialize to $\sigma(z) = e^z$ and $p^* = \mathbb{C}(1)^d$.

- Claim: $a(q) \geq 1/q!$
  - We have $a_j = 1/j!$ and a_J is just a_j times a positive integer stemming from the multiplicities in J (for example, if $J = (2, 0, 3)$, then $a_J = 2!3!a_j \geq a_j$.
- Claim: $\beta(q, R) = O(d^q)$ if $R = O(1/\sqrt{d})$.
  - We have that

$$\mathbb{E}_{w \sim \mathbb{C}(R)^d, x \sim \mathbb{C}(1)^d}[\phi^w(x)\overline{\phi^w(x)}] = \sum_J a_J^2 R^{2|J|} \tag{683}$$

$$= O(e^{2\sqrt{d}R}). \tag{684}$$

  - Finally, $\beta(q, R) = O(e^{2\sqrt{d}R}/R^{2q}) = O(d^q)$ by definition of R. Note that we need R to be small to fight the explosion stemming from $e^{2\sqrt{d}}$.
- Claim: $\gamma(q) = O(\sqrt{d^q})$
- Claim: the coefficients are bounded:

$$\sum_{i=1}^m |\alpha_i|^2 = \frac{1}{m^2} \sum_{i=1}^m |T(w_i)|^2 \leq \frac{\|f^*\|_1^2}{ma(q)^2 R^{2q}} = O(d^{2q}/m). \tag{685}$$

See the paper for more details.

6.6 References

- Telgarsky, 2012: Representation Power of Feedforward Neural Networks (slides)
- Andoni/Panigrahy/Valiant/Zhang, 2014: Learning Polynomials with Neural Networks
- Livni/Shalev-Shwartz/Shamir, 2014: On the Computational Efficiency of Training Neural Networks
- Livni/Shalev-Shwartz/Shamir, 2014: An Algorithm for Training Polynomial Networks
- Schmidhuber, 2014: Deep Learning in Neural Networks: An Overview

[begin lecture 18] (18)

7 Conclusions and outlook

7.1 Review (Lecture 18)

- The goal of this course is to provide a theoretical understanding of why machine learning algorithms work. To undertake this endeavor, we have developed many powerful tools (e.g., convex optimization, uniform convergence) and applied them to the classic machine learning setting: learning a predictor given a training set and applying to unseen test examples.

- To obtain more clarity, it is useful to decompose the error into approximation, estimation, and optimization error:

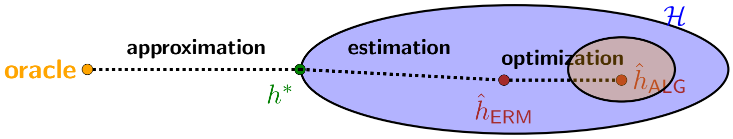

Figure 11: Cartoon showing error decomposition into approximation, estimation, and optimization errors.

  - Approximation error has to do with how much potential the hypothesis class has (e.g., Gaussian kernel versus linear kernel, large norm versus smaller norm)
  - Estimation error has to do with how well you're able to learn, which depends on the complexity of the hypothesis and the amount of data you have.
  - Optimization error is how well your algorithm is able to approximate the perfect optimizer (empirical risk minimizer).

  Sometimes approximation and optimization error are hard to distinguish. For example, in kernel methods, random Fourier features can be viewed as defining a smaller hypothesis space or doing an approximate job optimizing the original kernel-based objective. An algorithm implicity defines a class of hypotheses which are accessible by the algorithm.

- In the batch setting, we wish to study the generalization ability of learning algorithms. The key quantity was excess risk

$$L(\hat{h}) - L(h^*). \tag{686}$$

- We could use uniform convergence, which studies

$$\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h)|. \tag{687}$$

  - FIGURE: [empirical and expected risk, convergence]
  - Key ideas:
    - Concentration: moment generating function of sub-Gaussian random variables
    - Measures of complexity of hypothesis class: Rademacher complexity, covering numbers, VC dimension

- We could also use asymptotic statistics to get

$$L(\hat{\theta}) = L(\theta^*) + (\hat{\theta} - \theta^*)^\top \nabla L^2(\theta^*)(\hat{\theta} - \theta^*) + \cdots \tag{688}$$

$$\hat{\theta} - \theta^* = -\nabla^2 \hat{L}(\theta^*) \nabla \hat{L}(\theta^*) + \cdots, \tag{689}$$

  which vastly simplifies the calculations needed.

- In online learning, nature adversarily chooses convex loss functions $f_1, \dots, f_T$. The online mirror descent learner produces

$$w_t = \arg\min_{w \in S} \left\{ \psi(w) + \sum_{i=1}^{t-1} w \cdot z_i \right\}, \tag{690}$$

  We analyze the regret:

$$\text{Regret}(u) = \sum_{t=1}^{T} [f_t(w_t) - f_t(u)]. \tag{691}$$

  The decomposition into approximation, estimation, and optimization errors is not perfect here.

  - Key ideas:
    - Convexity allows us to linearize losses to produce an upper bound on regret
    - Strongly convex regularizer: tradeoff stability with fitting the data
    - ψ allows us to adapt to the geometry of the problem and arrive at algorithsm such as EG

- Results
  - Our results for estimation error depend on both the hypothesis class $\mathcal{H}$ and the number of examples. The exact dependence depends on the structure of the problem, which we summarize here (there are analogues both in the online and the batch settings):

    - Classic parametric rate: $1/\sqrt{n}$
    - If we have realizability or strong convexity: $1/n$
    - Default dependence on dimension: d
    - If we assume sparsity (k nonzeros): $k \log d$
    - In infinite dimensions (RKHS), we depend on the norm of the weight vector: $\|w\|_2$

- But machine learning isn't all about prediction where training and test examples are drawn i.i.d. from the same distribution. In the following, we will point out a few other directions.

7.2 Changes at test time (Lecture 18)

- Suppose we train a predictor, and deploy it in real-life. Theory says that as long as the test distribution doesn't deviate from training, we have guarantees on the predictor's behavior. But we do not live in a stationary world.

- Example: Google Flu Trends
  - Trained on 2003–2008 data, predict flu based on (50M) common queries, got 97% accuracy.
  - In 2009, vastly underestimated (due to swine flu, people searched differently).
  - In 2011–2013, overpredicted.

- Online learning doesn't solve this problem
  - One might think that online learning doesn't place distribution over the data, so it's robust.
  - But the analyses are with respect to an expert which is static, and if the world is changing under you, being static is a pretty lame thing to do. So the bounds do hold, but the statements are quite weak.

- Covariate shift
  - Training distribution: $x \sim p^*$
  - Test distribution: $x \sim q^*$
    - Example: object recognition in different lighting conditions / camera
  - Assume $p^*(y \mid x)$ is fixed across both training and test
  - Assume lots of unlabeled examples drawn from both p^* and q^* which can be used to estimate the marginal distributions.
  - Instance weighting (simplest approach)
    - Idea: upweight examples that are underrepresented at training time.

        - Estimator:

$$\hat{\theta} = \arg\min_{\theta} \hat{L}(\theta), \quad \hat{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \hat{w}(x)\ell((x,y);\theta). \tag{692}$$

        - If we had infinite unlabeled data, we could use the weights $\hat{w}(x) = p^*(x)/q^*(x)$, from which it's easy to check that $\hat{\theta}$ is an unbiased estimator of the expected risk at test time.
        - There are two problems with this:
            - First, in practice, we don't have infinite unlabeled data, so we would need to estimate $\hat{w}(x)$.
            - Second, we need to assume that q^* is absolutely continuous with respect to p^* (must see all test examples at training time). Otherwise, we have no hope of doing well.
        - There are several procedures that address these problems as well as analyses. For example, one can use asymptotic statistics:
            - Shimodaira: Improving predictive inference under covariate shift by weighting the log-likelihood function

- Domain adaptation / multi-task learning
    - In domain adaptation, even $p^*(y \mid x)$ might be different between train and test.
        - Example: my email and your email
    - Techniques
        - The general idea is to solve joint ERM problem where assume that weight vectors are close. Let $W = [w_1, \dots, w_T] \in \mathbb{R}^{d \times T}$ be the matrix of weights for T tasks.
        - We can think about performing joint learning where we regularize W using one of the following:
            - Assume weight vectors are similar in Euclidean distance: $\sum_{i \neq j} \|w_i - w_j\|^2$
            - Assume weight vectors lie in the same low-dimensional subspace: use trace norm $\|W\|_*$
            - Assume there exists a sparse set of features that is shared by all tasks: use block norm $\|W\|_{2,1}$, which is the sum of the L2 norms of the rows.
        - Neural networks provide a natural and compelling solution: just have all the tasks share the same hidden layer.
    - As far as theoretical analyses, the key intuition is that the regularizer reduces the effective hypothesis space from T independent weight vectors to T highly-related weight vectors.
        - Maurer, 2006: Bounds for Linear Multi-Task Learning

- Example: deep neural networks are non-robust
  - Szegedy/Zaremba/Sutskever/Bruna/Erhan/Goodfellow/Fergus, 2014: Intriguing properties of neural networks
  - This paper shows that one can take a high-accuracy neural network for object recognition, perturb the input by a very small adversarial amount to make the predictor incorrect. The perturbation is inperceptible to the naked eye and is obtained via an optimization problem like this:

    $$\min_{r \in \mathbb{R}^d} (f(x+r) - y_{\text{wrong}})^2 + \lambda \|r\|^2, \tag{693}$$

    where r is the perturbation, f is the trained neural network, and x is the input.
  - Note that if we had a classifier based on a Gaussian kernel, we couldn't do this, because the Gaussian kernel is smooth.
- Robustness at test time
  - At test time, suppose that up to K features can be zeroed out adversarily.
  - We can optimize for this using robust optimization. The following paper shows that for classification, the robust optimization version results in a quadratic program.
  - Globerson/Roweis, 2006: Nightmare at Test Time: Robust Learning by Feature Deletion

7.3 Alternative forms of supervision (Lecture 18)

- Active learning (learner chooses examples non i.i.d.)
  - Example: learning binary classifiers in 1D
    - Suppose data is generated according to $y = \mathbb{I}[x \geq \theta]$ for some unknown θ.
    - If we sample i.i.d. from some distribution over x, our expected risk falls as $O(1/n)$.
    - However, if we actively choose points (using binary search), our expected risk falls as $O(e^{-cn})$, which is exponentially faster.
  - Intuition: query examples that we are most uncertain about. In reality, we have noise, and sampling random i.i.d. is not terrible in the beginning.
  - Dasgupta, 2006: Coarse sample complexity bounds for active learning
  - Bach, 2007: Active learning for misspecified generalized linear models
- Semi-supervised learning
  - Motivation: labeled data is expensive, unlabeled data is cheap

  - But how does knowing $p^*(x)$ (from unlabeled data) help you with $p^*(y \mid x)$ (prediction problem). There is no free lunch, so we need to make some assumptions about the relationship between $p^*(x)$ and $p^*(y \mid x)$. For example, in classification, we can assume that the decision boundary defined by $[p^*(y \mid x) \geq \frac{1}{2}]$ doesn't cut through high-density regions of p^*.
  - Theory
    - $p^*(x)$ + assumptions about relationship defines a compatibility function $\chi(h)$
    - Reduced hypothesis class $\mathcal{H}$ to ones which are compatible $\{h \in \mathcal{H} : \chi(h) = 1\}$
    - Unlabeled data helps because we're reducing the hypothesis class
  - Balcan/Blum, 2009: A Discriminative Model for Semi-Supervised Learning

7.4 Interaction between computation and statistics (Lecture 18)

- This class is mostly about the statistical properties of learning algorithms, for which there is by now quite solid theory for. One of the most interesting avenues for future research is how computation plays a role in this story. The situation is rather complex.

- Favorable relationship: large optimization error results in smaller effective hypothesis space and actually controls estimation error.

  - Early stopping: relate to L2 regularization
  - Low-rank kernel approximations

- Unfavorable relationship: good estimators are hard to compute

  - Graphical models (partition function): pseudolikelihood
    - Suppose $x \in \{0,1\}^d$.
    - $p_\theta(x) = W_\theta(x)/Z(\theta)$
    - The partition function $Z(\theta) = \sum_x W_\theta(x)$ is computationally expensive, taking in the worst case $O(2^d)$ time.
    - However, we can maximize the pseudolikelihood $\prod_{i=1}^d p_\theta(x_i \mid x_{-i})$, which takes $O(d)$ time.
    - Pseudolikelihood is computationally more efficient, but statistically less efficient.
  - Learning latent variable models (e.g., Gaussian mixture models):
    - Maximum likelihood is non-convex and hard to compute.
    - The method of moments estimator (which works under some assumptions) is easy to compute but statistically less efficient.
  - Other constraints on learning: communication, memory, privacy

- Neural networks

- Here, the story is a bit more complex.
- On one hand, neural networks are difficult to train. People often talk about the vanishing gradient problem in the context of gradient-based optimization, where we are nowhere near a good solution, and yet the gradient is near zero. This happens when the weights w_j are too large, which saturates the sigmoid. To avoid this, careful initialization and step size selection are important.
- On the other hand, if we fully optimized existing neural network models, there is some chance they would just overfit, and the fact that we're not fully optimizing is actually central to their ability to generalize.

A Appendix

A.1 Notation

In general, we will not be religious about using uppercase letters to denote random variables or bold letters to denote vectors or matrices. The type of the variable should hopefully be clear from context.

- Basic definitions
  - $[n] = \{1, \dots, n\}$
  - For a sequence $v_1, \dots, v_n$:
    - Let $v_{i:j} = (v_i, v_{i+1}, \dots, v_{j-1}, v_j)$ be the subsequence from i to j inclusive.
    - Let $v_{<i} = v_{1:i-1}$.
  - ∇f: gradient of a differentiable function f
  - $\partial f(v)$: set of subgradients of a convex function f
  - Indicator (one-zero) function:

$$\mathbb{I}[\text{condition}] \overset{\text{def}}{=} \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{otherwise.} \end{cases} \tag{694}$$

  - Probability simplex:

$$\Delta_d \overset{\text{def}}{=} \left\{ w \in \mathbb{R}^d : w \succeq 0 \text{ and } \sum_{i=1}^{d} w_i = 1 \right\}. \tag{695}$$

  - Euclidean projection:

$$\Pi_S(w) \overset{\text{def}}{=} \arg\min_{u \in S} \|u - w\|_2 \tag{696}$$

    is the closest point (measured using Euclidean distance) to w that's in S.

- We will try to stick with the following conventions:
  - x: input
  - y: output
  - z: input-output pair
  - d: dimensionality
  - n: number of examples
  - t: iteration number

- T: total number of iterations
- f: (convex) function
- w: weight vector
- θ: parameters
- L: Lipschitz constant
- λ: amount of regularization
- η: step size
- $p^*(x, y)$: true distribution of data
- In general:
  * v^* denotes the optimal value of some variable v.
  * $\hat{v}$ denotes an empirical estimate of some variable v based on training data.

A.2 Linear algebra

- Inner (dot) product: given two vectors $u, v \in \mathbb{R}^n$, the dot product is $u^\top v = u \cdot v = \langle u, v \rangle = \sum_{i=1}^n u_i v_i$.

- Outer product: for a vector $v \in \mathbb{R}^n$, let $v^\otimes = vv^\top$

- Positive semidefinite (PSD) matrix: a symmetric square matrix $A \in \mathbb{R}^{n \times n}$ is PSD iff $v^\top A v \geq 0$ for all vectors $v \in \mathbb{R}^n$. Note: in this class, whenever we say a matrix is PSD, we implicitly assume that the matrix is symmetric.

- Eigenvalues: for a PSD matrix $A \in \mathbb{R}^{n \times n}$, let $\lambda_1(A), \dots, \lambda_n(A)$ be the eigenvalues of A, sorted in non-increasing order.

- $\mathrm{tr}(A)$: for a square matrix $A \in \mathbb{R}^{n \times n}$, $\mathrm{tr}(A)$ denotes the trace of A, the sum of the diagonal entries ($\mathrm{tr}(A) = \sum_{i=1}^n A_{ii}$).
  - $\mathrm{tr}(ABC) = \mathrm{tr}(BCA)$, but $\mathrm{tr}(ABC) \neq \mathrm{tr}(BAC)$ in general
  - $\mathrm{tr}(A) = \sum_{i=1}^n \lambda_i(A)$

- $\mathrm{diag}(v)$: for a vector $v \in \mathbb{R}^d$, a matrix whose diagonal entries are the components of v and off-diagonal entries are zero.

- Hölder's inequality
  - For any real vectors $u, v \in \mathbb{R}^d$,

$$|u \cdot v| \leq \|u\|_p \|v\|_q, \tag{697}$$

    for $1/p + 1/q = 1$, where $\|u\|_p = (\sum_{j=1}^d |u_j|^p)^{1/p}$ is the p-norm.

  - For $p = q = 2$, this is the Cauchy-Schwartz inequality.
  - Another important case is $p = 1$ and $q = \infty$.
  - Note that this result holds in greater generality (for L_p spaces).

A.3 Probability

- In general, we define a background space over which all the random variables are defined and use $\mathbb{P}[\cdot]$ and $\mathbb{E}[\cdot]$ to reference that space.

- Conditional expectation: If we have two random variables X and Y, we will write $\mathbb{E}[F(X, Y) \mid Y]$ to denote the random variable (that is a function of Y), rather than $\mathbb{E}_X[F(X, Y)]$ to denote that the expectation is taken over X. In general, we condition on the variables that we're not taking an expectation over rather than using subscripts to denote the variables that we are taking an expectation over.

- In the following, assume the following:
  - Let $X_1, \dots, X_n$ be real vectors drawn i.i.d. from some distribution with mean μ and covariance matrix Σ.
  - Let $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i$.

- Convergence in probability
  - We say that a sequence of random variables (Y_n) converges in probability to a random variable Y (written $Y_n \xrightarrow{P} Y$) if for all $\epsilon > 0$, we have $\lim_{n\to\infty} \mathbb{P}[|Y_n - Y| \geq \epsilon] = 0$.
  - Example: $\hat{\mu} \xrightarrow{P} \mu$ (weak law of large numbers)

- Convergence in distribution
  - We say that a sequence of distributions (P_n) converges in distribution (weak convergence) to a distribution P (written $P_n \xrightarrow{d} P$) if for all bounded continuous functions f, $\lim_{n\to\infty} \int f(x) P_n(dx) = \int f(x) P(dx)$.
  - When we write $Y_n \xrightarrow{d} P$ for a sequence of random variables (Y_n), we mean that the sequence of distribution of those random variables converges in distribution.
  - Example: $\sqrt{n}(\hat{\mu} - \mu) \xrightarrow{d} \mathcal{N}(0, \Sigma)$ (central limit theorem)

- Continuous mapping theorem
  - This theorem allows us to transfer convergence in probability and convergence in distribution results through continuous transformations.
  - If $f : \mathbb{R}^d \to \mathbb{R}^k$ is continuous and $Y_n \xrightarrow{P} Y$, then $f(Y_n) \xrightarrow{P} f(Y)$.

  - If $f : \mathbb{R}^d \to \mathbb{R}^k$ is continuous and $Y_n \xrightarrow{d} Y$, then $f(Y_n) \xrightarrow{d} f(Y)$.
  - Example: $\|\hat{\mu}\|_2 \xrightarrow{P} \|\mu\|_2$

- Delta method

  - This theorem allows us to transfer asymptotic normality results through smooth transformations.
  - If $\nabla f : \mathbb{R}^d \to (\mathbb{R}^d \times \mathbb{R}^k)$ is continuous and $\sqrt{n}(\hat{\mu} - \mu) \xrightarrow{d} \mathcal{N}(0, \Sigma)$, then

$$\sqrt{n}(f(\hat{\mu}) - f(\mu)) \xrightarrow{d} \mathcal{N}(0, \nabla f(\mu)^\top \Sigma \nabla f(\mu)). \tag{698}$$

  - The key intuition here is a Taylor approximation (valid because $\hat{\mu}$ is converging to μ):

$$f(\hat{\mu}) = f(\mu) + \nabla f(\mu)^\top (\hat{\mu} - \mu) + \cdots . \tag{699}$$

    Then apply the continuous mapping theorem.

- Slutsky's theorem

  - This result allows us to compose convergence results.
  - If $Y_n \xrightarrow{P} c$ and $Z_n \xrightarrow{d} Z$ then $Y_n Z_n \xrightarrow{d} cZ$.
  - Example: $\sqrt{n}\hat{\mu} \cdot (\hat{\mu} - \mu) \xrightarrow{d} \mathcal{N}(0, \mu^\top \Sigma \mu)$.

- Notation: $X_n = O_p(f(n))$ if $X_n / f(n)$ is bounded in probability, that is, for every $\epsilon > 0$, there exists M_ϵ such that for all n, $\mathbb{P}[|X_n / f(n)| > M_\epsilon] < \epsilon$.

  - Example: $\mathcal{N}(0, \Sigma) = O_p(1)$
  - Example: $\hat{\mu} - \mu = O_p\left(\frac{1}{\sqrt{n}}\right)$

- These results turn complicated things on the LHS into simple things on the RHS. Use your intuitions from real analysis.

- Jensen's inequality

  - For any convex function $f : \mathbb{R}^d \to \mathbb{R}$ and random variable $X \in \mathbb{R}^d$,

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]. \tag{700}$$

A.4 Functional analysis

- **Definition 30 (Cauchy sequence)**

  A Cauchy sequence in a metric space (X, ρ) is $(x_i)_{i \geq 1}$ such that for any $\epsilon > 0$, there exists an integer n such that all $i, j > n$, $\rho(x_i, x_j) < \epsilon$.

- **Definition 31 (complete metric space)**

  A complete metric space (X, ρ) is one where every Cauchy sequence $(x_i)_{i \geq 1}$ converges to a limit point $x^* \in X$. Intuition: if the elements of the sequence are getting arbitrarily close to each other, they are getting close to some particular point in the space.

- $L^p(\mathcal{X})$ is the space of all measurable functions f with finite p-norm:

$$\|f\|_p \stackrel{\text{def}}{=} \left(\int |f(x)|^p dx \right)^{1/p} < \infty. \tag{701}$$

- Example: if $\mathcal{X} = \mathbb{R}$, $f(x) = 1$ is not in L^p for any $p < \infty$.
- $L^2(\mathcal{X})$ is the set of all square integrable functions.
- $L^\infty(\mathcal{X})$ is the set of all bounded functions.

References

Agrawal, S. and Goyal, N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory (COLT)*.

Anandkumar, A., Foster, D. P., Hsu, D., Kakade, S. M., and Liu, Y. (2012a). Two SVDs suffice: Spectral decompositions for probabilistic topic modeling and latent Dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*.

Anandkumar, A., Ge, R., Hsu, D., and Kakade, S. (2013). A tensor spectral approach to learning mixed membership community models. In *Conference on Learning Theory (COLT)*, pages 867–881.

Anandkumar, A., Hsu, D., and Kakade, S. M. (2012b). A method of moments for mixture models and hidden Markov models. In *Conference on Learning Theory (COLT)*.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256.

Chaganty, A. and Liang, P. (2014). Estimating latent-variable graphical models using moments and likelihoods. In *International Conference on Machine Learning (ICML)*.

Flaxman, A. D., Kalai, A. T., and McMahan, H. B. (2005). Online convex optimization in the bandit setting: gradient descent without a gradient. In *Symposium on Discrete Algorithms (SODA)*, pages 385–394.

Halpern, Y. and Sontag, D. (2013). Unsupervised learning of noisy-or Bayesian networks. In *Uncertainty in Artificial Intelligence (UAI)*.

Hsu, D. and Kakade, S. M. (2013). Learning mixtures of spherical Gaussians: Moment methods and spectral decompositions. In *Innovations in Theoretical Computer Science (ITCS)*.

Janzamin, M., Sedghi, H., and Anandkumar, A. (2015). Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*.

Lai, T. L. and Robbins, H. (1985). Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Orabona, F., Crammer, K., and Cesa-Bianchi, N. (2015). A generalized online mirror descent with applications to classification and regression. *Machine Learning*, 99(3):411–435.

Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110.

Rakhlin, A. and Sridharan, K. (2013). Online learning with predictable sequences. In *Conference on Learning Theory (COLT)*, pages 993–1019.

Shalev-Shwartz, S. (2011). Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194.

Steinhardt, J. and Liang, P. (2014). Adaptivity and optimism: An improved exponentiated gradient algorithm. In *International Conference on Machine Learning (ICML).*

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3):285–294.

van der Vaart, A. W. (1998). *Asymptotic statistics.* Cambridge University Press.