CS229T/STAT231: Statistical Learning Theory (Winter 2013)

Percy Liang

Last updated Mon Mar 11 2013 00:02

These lecture notes are still rough, but they will be updated periodically as the course goes on. Please let us know when you find typos. Section A.1 describes the basic notation and definitions; Section A.2 describes some basic theorems.

Lectures The notes are organized according to topic. However, the lectures themselves will often spill across topics, so the material actually covered in each lecture is explicitly delimited and linked below.

- Lecture 1/7: Course overview
- Lecture 1/9: Online learning
- Lecture 1/14: Online learning
- Lecture 1/16: Online learning
- Lecture 1/21: No class (Martin Luther King Day)
- Lecture 1/23: Online learning
- Lecture 1/28: Online learning
- Lecture 1/30: Kernel methods
- Lecture 2/4: Kernel methods
- Lecture 2/6: Kernel methods
- Lecture 2/11: Uniform convergence
- Lecture 2/13: Uniform convergence
- Lecture 2/18: No class (Presidents Day)
- Lecture 2/20: Uniform convergence
- Lecture 2/25: Uniform convergence
- Lecture 2/27: Uniform convergence
- Lecture 3/4: Asymptotic analysis

- Lecture 3/6: Asymptotic analysis
- Lecture 3/11: Method of moments
- Lecture 3/13: Final project presentations

[begin lecture 1/7] (1/7)

1 Overview

1.1 What is this course about?

- Machine learning has become an indispensible part of many application areas, ranging from science (biology, neuroscience, psychology, astronomy, etc.) to engineering (natural language processing, computer vision, robotics, etc.).
- But machine learning is not one coherent approach, but rather consists of a dazzling array of seemingly disparate frameworks and paradigms spanning classification, regression, clustering, matrix factorization, graphical models, probabilistic programming, etc.
- This course tries to uncover the common **statistical principles** underlying this diverse array of techniques and applications, using theoretical analysis as the main tool. We will try to answer two questions:
  - When do learning algorithms work (or fail)? Anyone with hands-on machine learning experience will probably know that learning algorithms rarely work the first time. When they don't, a deeper theoretical understanding offers a new perspective and can aid in troubleshooting.
  - How do we make learning algorithms better? Theoretical analyses often point to weaknesses in current algorithms, and suggest new algorithms to address the limitations of existing ones. For example, is it possible to combine weak classifiers into a strong classifier? This theoretical challenge eventually led to the development of AdaBoost, a beautifully simple and practical algorithm with strong theoretical guarantees.
- Example: text classification
  - Goal: build a classifier to predict the topic of a document (e.g., sports, politics, technology, etc.).
  - We train an SVM on bag-of-words features and obtain 8% training error on 1000 training documents, test error is 13% on 1000 documents. Now what? Here are several questions which could be useful in figuring out what's going on.
    * How reliable are these numbers (error bars)?
    * 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?
    - Should we change the learning algorithm altogether?
  - In this class, we develop tools to tackle some of these questions. The 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, number of training examples), and remark on how they relate to error rate.

- In this course, we will assume knowledge of the following (you don't need to remember the deep theorems, just be able to work proficiently with the concepts):

  - Linear algebra: eigendecomposition, pseudoinverses, matrix derivatives, etc.
  - Probability/statistics: conditional expectation, variance, conditional independence, common distributions (Gaussian, Dirichlet, etc.)
  - Convex optimization: convex functions, duality
  - Machine learning: Perceptron algorithm, SVM, Naive Bayes, logistic regression, least squares regression, HMM, EM

  But perhaps most importantly, having mathematical maturity and being able to do proofs is key.

- In the next sections, we give a brief overview of the various topics in this course.

1.2 Online learning

- We will start with online learning, as it leads to many practical algorithms and offers a relatively clean theory and sharp results.
- Prediction task: given input $x \in \mathcal{X}$ (e.g., $\{0,1\}^d$), predict output $y \in \mathcal{Y}$ (e.g., $\{0,1\}$ or $\mathbb{R}$)
- Iterate $t = 1, \ldots, T$:
  - Learner receives input $x_t \in \mathcal{X}$
  - Learner chooses prediction $p_t \in \mathcal{Y}$
  - Nature chooses true label $y_t \in \mathcal{Y}$
  - Suffer **loss** $\ell(y_t, p_t) \in \mathbb{R}$
- How do we evaluate?
  - Let $\mathcal{H}$ be a set of expert predictors.
  - **Regret**: cumulative difference between learner's loss and best expert's loss (low is good).

- We will prove statements of the following form:

$$\text{Regret} \leq \text{SomeFunction}(\mathcal{H}, T). \tag{1}$$

  For example, for finite $\mathcal{H}$, we have:

$$\text{Regret} \leq O(\sqrt{T \log |\mathcal{H}|}), \tag{2}$$

  so the average regret goes to zero (in the long run, we're doing as well as the best expert). And this is with zero assumptions about the examples—could be generated by an **adversary**!

- Ideas

  - **Online convex optimization**: The theoretical foundation for online learning is built on **convexity**. We will consider a more general framework known as online convex optimization. In this framework, learners correspond to simple and scalable (sub)gradient descent algorithms of a weight vector $w_t \in \mathbb{R}^d$, for example:

$$w_{t+1} \leftarrow w_t - \eta \nabla \ell(y_t, w_t \cdot x_t), \tag{3}$$

    where η is the step-size and $\nabla \ell$ is the gradient of the loss with respect to w_t.

  - **Online-to-batch conversion**: in batch learning, we care about generalization ability of an algorithm.

    * Learner takes in n i.i.d. training examples (x, y) and outputs a predictor h.
    * Evaluate h on test examples drawn from the same distribution.

    While the online and batch scenarios on the surface look different, they are closely connected. In fact, we can take an online learner with **low regret** and convert it into a batch learner that has **low generalization error**.

  - **Multi-armed bandits**: learner only gets partial feedback about the best predictions to make.

    * Bandits have many applications such as advertisement placement and packet routing.
    * Need to deal with exploration/exploitation, but not with state (so easier than reinforcement learning).

1.3 Kernel methods

- Real-world data is complex, so we need expressive models. Kernels provide a rigorous mathematical framework to build complex, non-linear models.

- Feature view: define functions via features $\phi(x)$.

  - $\phi(x) = x$ yields linear functions

- $\phi(x) = x^2$ yields quadratic functions

- Kernel view: define **positive semidefinite kernel** $k(x, x')$, which captures similarity between x and x'.
  - Allows 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 represent *any* continuous function (and with enough data, it will learn it).
  - For strings, trees, graphs, etc., can define kernels that exploit dynamic programming for efficient computation.

- Two views are dual to each other; in some sense, which one to use boils down to computation and convenience. Note that kernels are not necessary for building non-linear models.

- You might have been exposed to kernels through SVMs, but essentially all learning algorithms that require only a dot product can be kernelized: includes regression, classification, ranking, PCA, CCA, etc.

- Kernel methods generally require $O(n^2)$ time (n is the number of training points) to even compute the kernel matrix. Low-rank approximations such as column sampling offer efficient solutions.

1.4 Uniform convergence: VC, covering numbers, Rademacher complexity

- Next, we will analyze the generalization error of learning algorithms in the batch setting. Instead of designing new algorithms, we will assume we have one that computes the **empirical risk minimizer** (M-estimator in statistics):

$$\hat{h}_{\text{ERM}} \in \arg\min_{h \in \mathcal{H}} \hat{R}(h), \tag{4}$$

  where $\hat{R}(h)$ is the empirical risk of h, the average loss over n i.i.d. training examples. This includes maximum likelihood.

- We will derive **generalization bounds** of the following form:

$$\underbrace{R(\hat{h}_{\text{ERM}})}_{\text{generalization error}} \leq \underbrace{\hat{R}(\hat{h}_{\text{ERM}})}_{\text{training error}} + \text{SomeFunction}(\text{Complexity}(\mathcal{H}), n). \tag{5}$$

  This directly translates to a bound relating the generalization error of the empirical risk minimizer to the generalization error of the best $h^* \in \mathcal{H}$:

$$R(\hat{h}_{\text{ERM}}) \leq R(h^*) + \text{SomeFunction}(\text{Complexity}(\mathcal{H}), n). \tag{6}$$

- The main idea is that for a fixed $h \in \mathcal{H}$, the training error $\hat{R}(h)$ converges to generalization error $R(h)$ by Hoeffding's inequality or central limit theorem.
- However, $\hat{h}_{\text{ERM}}$ is random, so we need to get **uniform convergence** over all functions $h \in \mathcal{H}$, which ensures convergence for $R(\hat{h}_{\text{ERM}})$ as well.
- The rate of convergence is governed by the complexity of $\mathcal{H}$, so 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, for finite $\mathcal{H}$, the right measure of complexity is $\log |\mathcal{H}|$, which agrees with online learning bounds.
- In these analyses, we assume all examples are drawn i.i.d. from some unknown distribution $p^*(x, y)$ but make no further assumptions about p^*.

1.5 Direct analysis

- Uniform convergence only gives us upper bounds so we can't directly compare the generalization error of two algorithms. Also, it is worst case over all distributions p^*, so it isn't sensitive to the problem structure.
- In simplified settings, we can actually do a more direct analysis to get sharper results which are exact. Here, the goal will not be to obtain the most general results, but to obtain intuition about problem structure for certain cases.
- First, we consider linear regression, which will allow us to really exploit the underlying geometry of the problem. We will see that statistical estimation is just a Euclidean projection.
- We will also derive bias-variance decompositions, showing the tradeoffs between the different types of approximation and estimation error.
- For non-linear models, we can't compute the error directly. But we can appeal to classical asymptotics, a common tool in statistics. Assuming twice differentiability, we can perform a **Taylor expansion** of the expected risk to obtain:

$$R(\hat{h}_{\text{ERM}}) = R(h^*) + \frac{\text{SomeFunction}(p^*, h^*)}{n} + \cdots \tag{7}$$

  The asymptotics approach is fairly lightweight and gives equality up to the first-order, which allows us to compare different estimators (e.g., generative estimators versus discriminative estimators) at least in an asymptotic sense.

1.6 Dimensionality reduction using spectral methods

- So far, we have focused mostly on prediction with fixed data representations (fixed features, kernels). This only works well when the representations are good. The natural question to ask is whether these representations can be learned automatically. We will consider several types of methods that learn representations by leveraging the eigenstructure of the data.

- First, we will look at matrix factorization, a critical piece in the famous Netflix competition for recommender systems. We will see how to obtain a convex formulation using the trace norm.

- Next, we consider spectral clustering, a paradigm for grouping data points, in which clusters take a shape suggested by the data. In contrast, clusters produced by K-means must be spherical.

- Finally, we will consider recent work on learning probabilistic latent-variable models such as mixtures of Gaussians, hidden Markov models, and Latent Dirichlet Allocation.

  - Traditional methods rely either on local optimization (via EM or variational methods) of the likelihood or sampling of the posterior, both of which fall prey to local optima.
  - Recent work shows that we can actually completely sidestep these issues by considering the spectral properties of these models in the context of **method of moments**, leading to simple algorithms based on eigendecomposition which (i) do not have problems with local optima and (ii) reveal insight into the model.

1.7 Miscellaneous

- In the unlikely event that there is extra time left, we will briefly touch upon a subset of the following:

  - Sparsity, regularization with mixed norms
  - Submodularity
  - Random forests, bagging, boosting
  - Active learning
  - Domain adaptation
  - Multi-view learning, multi-task learning

[begin lecture 1/9] (1/9)

2 Online learning

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 mostly in the adversarial setting), but we will draw quite a bit from convex analysis.

2.1 Introduction

- Framework
  - Prediction task: we want to map inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$.
  - We can think of online learning as a game between a learner and Nature.
  - Iterate $t = 1, \ldots, T$:
    - Learner receives input $x_t \in \mathcal{X}$
    - Learner makes prediction $p_t \in \mathcal{Y}$
    - Nature chooses label $y_t \in \mathcal{Y}$
    - Suffer loss $\ell(y_t, p_t) \in \mathbb{R}$
    - (In practice: learner updates model parameters)
  - Learner can make the prediction p_t based on the entire history $(x_{\leq t}, y_{<t}, p_{<t})$.
  - Mathematically, a learner is a function $\mathcal{A}$ that takes this history and returns a prediction $p_t = \mathcal{A}(x_{\leq t}, y_{<t}, p_{<t})$. But it's also useful to think of the learner as a procedure that updates model parameters at the end of each iteration.
- **Example 1 (online binary classification for spam filtering)**
  - Inputs: $\mathcal{X} = \{0,1\}^d$ are boolean feature vectors (presence or absence of a word).
  - Outputs: $\mathcal{Y} = \{0,1\}$: 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.

- Online learning algorithms usually learn much **faster** than traditional batch-oriented algorithms in practice, because they update parameters more frequently (after each example). This is true especially for large datasets.
- 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: how good is a learner?
  - Let $\mathcal{H}$ be a set of experts, where each **expert** is a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that predicts $h(x)$ on input $x \in \mathcal{X}$. Note that the learner can adapt whereas the experts are fixed.
  - **Definition 1 (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:

$$\text{Regret}(h) = \sum_{t=1}^{T} [\ell(y_t, p_t) - \ell(y_t, h(x_t))]. \tag{8}$$

    * The regret with respect to a class of experts $\mathcal{H}$ is the maximum regret (attained by best expert):

$$\text{Regret} = \max_{h \in \mathcal{H}} \text{Regret}(h). \tag{9}$$

  - Intuition: having fixed experts makes it possible to compete in an adversarial setting. If the adversary tries to screw over the learner, it will probably screw over the experts too.

- Objective: minimize regret. We want to obtain a result of the form: there exists an algorithm $\mathcal{A}$ (ideally by construction), such that for all $\mathcal{H}, T, x_{1:T}, y_{1:T}$, we have:

$$\text{Regret} \leq \text{SomeFunction}(\mathcal{H}, T). \tag{10}$$

  - Low regret is good. Usually, we want the regret to be sublinear in T, which means that the average regret goes to zero.
  - Aside: note that regret can even be negative in online learning (since the learner can adapt but the experts cannot).

2.2 Warm-up

We will give two examples, one where online learning is impossible and one where it is possible.

- **Example 2 (negative result)**
  - Assume the zero-one loss: $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$
  - Assume the learner is fully **deterministic**.
  - **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)
  - Consider two experts, $\mathcal{H} = \{h_0, h_1\}$, where h_y always predicts y.
  - Expert h_b's cumulative loss: $\leq T/2$, where b is the majority label of $y_1, \ldots, y_T$ (one of the experts errs on at most half of the examples).
  - Result: for *any* deterministic learner $\mathcal{A}$ (no matter how clever), there exists an input/output sequence $x_{1:T}, y_{1:T}$ such that:

$$\boxed{\text{Regret} \geq T/2} \quad \text{[awful!]} \tag{11}$$

- **Example 3 (positive result (learning with expert advice))**
  - Assume the 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 1 (realizable)**
    Assume the best expert $h^* \in \mathcal{H}$ has zero loss ($y_t = h^*(x_t)$). This restricts the set of input/output sequences that the adversary is allowed to choose.
  - We will design an algorithm that can query 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 set $V_t \subset \mathcal{H}$ of valid experts (those compatible with the first $t-1$ examples).
    - At each iteration t, learner predicts p_t to be the majority vote over predictions of valid experts $\{h(x_t) : h \in V_t\}$.
    - Keep experts which were correct: $V_{t+1} = \{h \subset V_t : y_t = h(x_t)\}$.
  - Analysis:
    - On each mistake, at least half of the experts are eliminated.

* So $1 \leq |V_{T+1}| \leq |\mathcal{H}|2^{-M}$, where M is the number of mistakes (equal to Regret since the best expert has zero loss).
* Take logs and rearrange:

$$\boxed{\text{Regret} \leq \log_2 |\mathcal{H}|.} \tag{12}$$

* However, realizability (some expert is perfect) is too strong of an assumption.
* Also, regret bound is useless if there are an infinite number of experts ($|\mathcal{H}| = \infty$).

2.3 Online convex optimization

- In order to obtain more general results, we move to a framework called online convex optimization. We will see that **convexity** is what makes everything work.
- Let $S \subset \mathbb{R}^d$ be a convex set (e.g., representing set of allowed weight vectors).
  - Example: $S = \{u : \|u\| \leq B\}$.
- **Definition 2 (convexity)**

  A function $f : S \mapsto \mathbb{R}$ is convex iff for all points $w \in S$, there is some vector $z \in \mathbb{R}^d$ such that

$$\boxed{f(u) \geq f(w) + z \cdot (u - w) \quad \text{for all } u \in S.} \tag{13}$$

  This says that we can always find a linear approximation at any $w \in S$ that lower bounds the function f.
- **Definition 3 (subgradient)**

  For each $w \in S$, the set of all z satisfying (13) are known as the subgradients at w:

$$\boxed{\partial f(w) \stackrel{\text{def}}{=} \{z : f(u) \geq f(w) + z \cdot (u - w) \text{ for all } u \in S\}.} \tag{14}$$

- Examples
  - What's convex?
    * 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: $f + g$
    * Scale: cf where $c \geq 0$
    * Max: $\max\{f, g\}$

  - Hinge loss
    * Function: $f(w) = \max\{0, 1 - y(w \cdot x)\}$.
    * Subgradient: $\partial f(w)$ contains $-yx$ if w does not achieve margin at least 1 ($y(w \cdot x) \geq 1$) and 0 otherwise; if $y(w \cdot x) = 1$, then many subgradients.

- The setup for online convex optimization is similar to online learning:
  - Iterate $t = 1, \ldots, T$:
    * Learner chooses $w_t \in S$
    * Nature chooses convex function $f_t : S \mapsto \mathbb{R}$
    * Suffer loss $f_t(w_t) \in \mathbb{R}$
  - Regret is defined in the usual way (cumulative difference of losses):

$$\text{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)]. \tag{15}$$

$$\text{Regret} \stackrel{\text{def}}{=} \max_{u \in S} \text{Regret}(u). \tag{16}$$

- Relationship to convex optimization
  - Convex optimization setup: $\min_{w \in S} f(w)$.
  - We can use a online convex optimization algorithm to minimize a convex function as follows:
    * Set $f_t(w) = f(w)$ for all t.
    * After T iterations, take $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w_t$.
    * By Jensen's inequality:

$$f(\bar{w}) \leq \frac{1}{T} \sum_{t=1}^{T} f(w_t) = \min_{w \in S} f(w) + \frac{\text{Regret}}{T}. \tag{17}$$

      Think of Regret as either $O(1)$, $O(\log T)$, or $O(\sqrt{T})$.
    * Note that by optimization standards, $O(1/\sqrt{T})$ or even $O(1/T)$ (which in statistics is a fast rate) is very slow convergence. Standard optimization algorithms such as gradient descent obtain linear convergence ($O(e^{-cT})$) or even superlinear convergence ($O(e^{-cT^2})$). We'll see later that for machine learning applications, these sublinear rates are actually good enough in a formal sense.

- The following examples show how to cast online learning as 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.

– **Example 4 (linear regression)**

  * Assume we are doing linear regression with the squared loss, $\ell(y_t, p_t) = (p_t - y_t)^2$, which is a convex function, and assume that the prediction is just the dot product between some weight vector $w_t \in \mathbb{R}^d$ and input $x_t \in \mathbb{R}^d$ ($p_t = w_t \cdot x_t$).
  * At iteration t, based on the example (x_t, y_t), nature chooses a function based on the squared loss:

$$f_t(w_t) = (\underbrace{w_t \cdot x_t}_{p_t} - y_t)^2. \tag{18}$$

    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.

– **Example 5 (learning with expert advice)**

  * Assume we have a finite number of experts $\mathcal{H}$, which the learner can query.
  * Assume we are doing binary classification with the zero-one loss, $\ell(y_t, p_t) = \mathbb{I}[y_t \neq p_t]$. Unfortunately, this is not a convex loss function.
  * Nonetheless, we can convexify the problem using randomization. Specifically, we allow the learner to produce a probability distribution $w_t \in \Delta_d$ 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, given (x_t, y_t), define a vector of losses incurred by each expert:

$$z_t = (\ell(y_t, h_1(x_t)), \ldots, \ell(y_t, h_d(x_t))) \in \{0, 1\}^d. \tag{19}$$

    The expected zero-loss from sampling an expert from w_t is then just:

$$f_t(w_t) = w_t \cdot z_t. \tag{20}$$

  * 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.

[begin lecture 1/14] (1/14)

2.4 Follow the leader (FTL)

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 fails for linear functions but works for quadratic functions. This will give us intuition about how to fix up our algorithm.

- **Algorithm 2 (follow the leader (FTL))**
  - The learner chooses the weight vector w_t that minimizes the cumulative loss so far on the previous $t-1$ iterations:

$$\boxed{w_t \in \arg\min_{w \in S} \sum_{i=1}^{t-1} f_i(w).} \tag{21}$$

    (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.
  - We can think of FTL as an empirical risk minimizer (we will study this in batch learning), where the training set is the first $t-1$ examples.

- **Lemma 1 (compare FTL with one-step lookahead cheater)**
  - Regret compares the learner (FTL) with the best expert; we will replace the best expert with something that is (i) at least as good and (ii) easier to compare with the learner.
  - Let $w_1, \ldots, w_T$ be produced by FTL. For any $u \in S$:

$$\boxed{\text{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)] \leq \sum_{t=1}^{T} [f_t(w_t) - f_t(w_{t+1})].} \tag{22}$$

  - 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!).
  - The RHS terms $f_t(w_t) - f_t(w_{t+1})$ measure how stable the algorithm is; smaller is better.
  - High-level intuition: recall that error in machine learning is due to a combination of bias (approximation error) and variance (estimation error). Intuitively we showed that FTL ensures the minimum possible bias (w_{t+1} is better than any u). It remains to show that the variance is small ($f_t(w_t)$ is close to $f_t(w_{t+1})$). Don't take this statement too literally; if it confuses you, just ignore it.

- Proof of Lemma 1:
  - Subtracting $\sum_t f_t(w_t)$ from both sides, it suffices to show

$$\text{Regret}(u) \stackrel{\text{def}}{=} \sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(u) \tag{23}$$

    for all $u \in S$. Intuitively, adapting helps one do better.
  - Proof by induction:
    - Assume the inductive hypothesis on $T-1$:

$$\sum_{t=1}^{T-1} f_t(w_{t+1}) \leq \sum_{t=1}^{T-1} f_t(u) \quad \text{for all } u \in S. \tag{24}$$

    - Add $f_T(w_{T+1})$ to both sides:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T-1} f_t(u) + f_T(w_{T+1}) \text{ for all } u \in S. \tag{25}$$

    - In particular, this holds for $u = w_{T+1}$, so we have:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(w_{T+1}). \tag{26}$$

    - Since $w_{T+1} \in \arg\min_u \sum_{t=1}^{T} f_t(u)$ by definition of FTL, we have:

$$\sum_{t=1}^{T} f_t(w_{t+1}) \leq \sum_{t=1}^{T} f_t(u) \text{ for all } u \in S, \tag{27}$$

      which is the inductive hypothesis for T.

- **Example 6 (quadratic optimization: FTL works)**
  - Assume nature always chooses quadratic functions:

$$f_t(w) = \frac{1}{2}\|w - z_t\|^2. \tag{28}$$

  - FTL (minimizing over $S = \mathbb{R}^d$) has a closed form solution, which is just the average of the previous points:

$$w_t = \frac{1}{t-1} \sum_{i=1}^{t-1} z_i. \tag{29}$$

- Let $L = \max_{1 \le t \le T} \|z_t\|$ be the maximum norm of a data point.
- Bound one term of the RHS of Lemma 1:

$$f_t(w_t) - f_t(w_{t+1}) = \frac{1}{2}\|w_t - z_t\|^2 - \frac{1}{2}\|(1 - 1/t)w_t + (1/t)z_t - z_t\|^2 \tag{30}$$
$$= \frac{1}{2}(1 - (1 - 1/t)^2)\|w_t - z_t\|^2 \tag{31}$$
$$\le (1/t)\|w_t - z_t\|^2 \tag{32}$$
$$\le (1/t)4L^2. \tag{33}$$

- Side calculation: summing $1/t$ yields $\log T$:

$$\sum_{t=1}^{T}(1/t) \le 1 + \int_1^T (1/t)dt = \log(T) + 1. \tag{34}$$

- Summing over t yields:

$$\boxed{\text{Regret} \le \sum_{t=1}^{T}[f_t(w_t) - f_t(w_{t+1})] \le 4L^2(\log(T) + 1).} \tag{35}$$

- 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 7 (linear optimization: FTL fails)**
  - We will construct an evil example to make FTL fail.
  - Let $S = [-1, 1]$ be FTL's possible predictions.
  - Consider linear functions $f_t(w) = wz_t$ in $d = 1$ dimension, where

$$(z_1, z_2, \dots) = (-0.5, 1, -1, 1, -1, 1, -1, \dots). \tag{36}$$

  - The minimizer computed by FTL will be attained at an extreme point, causing oscillating behavior.

$$(w_1, w_2, \dots) = (0, 1, -1, 1, -1, 1, -1, 1, \dots). \tag{37}$$

  - 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 as follows:

$$\boxed{\text{Regret} \geq T - 1}. \tag{38}$$

- What's the lesson?
  - For quadratic functions, w_t and w_{t+1} get closer and closer (low regret).
  - For linear functions, w_t and w_{t+1} can be always far apart (high regret).
  - FTL works when functions f_t offer "stability" (e.g., quadratic) but fail when they do not (e.g., linear).

2.5 Follow the regularized leader (FTRL)

- In general, we can't put restrictions on f_t, since the adversary controls those functions. But we can add some regularization of our own choosing to stablize the learner and make sure we don't bounce around.
- **Algorithm 3 (follow the regularized leader (FTRL))**
  - Let $S \subset \mathbb{R}^d$ be a convex set of parameter vectors.
  - Let $\psi : S \mapsto \mathbb{R}$ be a convex function (called a regularizer).
  - On iteration t, the learner chooses the weight vector that minimizes the regularizer plus the losses on the first $t-1$ examples:

$$\boxed{w_t \in \arg\min_{w \in S} \psi(w) + \sum_{i=1}^{t-1} f_i(w).} \tag{39}$$

- Quadratic ψ, linear f_t
  - For now, let's specialize to quadratic regularizers ψ and linear loss functions f_t:

$$\psi(w) = \frac{1}{2\eta}\|w\|^2, \qquad f_t(w) = w \cdot z_t. \tag{40}$$

  - Then the FTRL optimization problem is:

$$w_t = \arg\min_{w \in S} \left\{ \frac{1}{2\eta}\|w\|^2 - w \cdot \theta_t \right\}, \tag{41}$$

    where

$$\theta_t = -\sum_{i=1}^{t-1} z_i \tag{42}$$

    is the negative sum of the gradients z_t. Interpret θ_t as the direction we want to move in, but of course we're held back by the quadratic regularizer.

– If $S = \mathbb{R}^d$, then FTRL has a closed form solution:

$$w_t = \eta\theta_t, \tag{43}$$

a scaled down version of θ_t. Equivalently, we can think of the weights as being updated incrementally according to the following recurrence:

$$\boxed{w_{t+1} = w_t - \eta z_t.} \tag{44}$$

From this perspective, the recurrence in (44) 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 (41) by completing the square:

$$\boxed{w_t \in \arg\min_{w \in S} \|w - \eta\theta_t\| = \Pi_S(\eta\theta_t),} \tag{45}$$

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.

- **Theorem 1 (regret of FTRL)**

  – Let ψ be quadratic and f_t be linear according to (40).

  – Then the regret of FTRL with respect to u is as follows:

$$\boxed{\text{Regret}(u) \leq \frac{1}{2\eta}\|u\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\|z_t\|^2.} \tag{46}$$

  – Here, the step-size η allows us to trade off two terms:

    * First term: bias due to regularization. We need to get to u to achieve its loss. Smaller η means more regularization, which means it's harder to get there.
    * Second term: variance due to changing z_t. A smaller η means more regularization, which stabilizes the weights.

- Corollary:

  – To make the bound look cleaner:

    * Let $\|z_t\| \leq L$ (L controls how quickly the functions grow).
    * Recall that since $u \in S$, we have $\|u\| \leq B$.

  Now the bound can be rewritten as:

$$\text{Regret}(u) \leq \frac{B^2}{2\eta} + \frac{\eta T L^2}{2}. \tag{47}$$

- Side calculation:
  - Suppose we want to minimize $C(\eta) = a/\eta + b\eta$.
  - 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

$$\boxed{\text{Regret} \leq BL\sqrt{T}.} \tag{48}$$

Note that the average regret goes to zero as desired, even though not as fast as for quadratic functions ($\log T$).

[begin lecture 1/16] (1/16)

- We will prove Theorem 1 later using Bregman divergences, but just to give some intuition, we will prove a slightly weaker result (the second term is looser by a factor of 2):

$$\boxed{\text{Regret}(u) \leq \frac{1}{2\eta}\|u\|^2 + \eta \sum_{t=1}^{T} \|z_t\|^2.} \tag{49}$$

- Let's try to reduce FTRL to FTL. Observe that FTRL is the same as FTL where the first function is the regularizer ($f_0(w) = \psi(w)$). The key difference is the regret shouldn't include the regularizer, since that's the learner's invention.
- Let us then apply Lemma 1 to $\psi, f_1, \ldots, f_T$, resulting in the bound:

$$\sum_{t=1}^{T}[f_t(w_t) - f_t(u)] - \psi(u) \leq \sum_{t=1}^{T}[f_t(w_t) - f_t(w_{t+1})] - \psi(w_1). \tag{50}$$

- Shuffling terms around, and using the fact that $\psi(u) = \frac{1}{2\eta}\|u\|^2$ and $\psi(w_1) = 0$ (w_1 is the minimizer of the regularizer), we have:

$$\begin{aligned}
\text{Regret}(u) &= \sum_{t=1}^{T}[f_t(w_t) - f_t(u)] && (51)\\
&\leq \frac{1}{2\eta}\|u\|^2 + \sum_{t=1}^{T}[f_t(w_t) - f_t(w_{t+1})] && (52)\\
&= \frac{1}{2\eta}\|u\|^2 + \sum_{t=1}^{T} z_t \cdot (w_t - w_{t+1}) \quad [\text{since } f_t(w) = w \cdot z_t] && (53)\\
&\leq \frac{1}{2\eta}\|u\|^2 + \sum_{t=1}^{T} \|z_t\|\|w_t - w_{t+1}\| \quad [\text{Cauchy-Schwartz}]. && (54)
\end{aligned}$$

Now let's bound the difference between subsequent weight vectors:

$$\begin{aligned} \|w_t - w_{t+1}\| &= \|\Pi_S(\eta\theta_t) - \Pi_S(\eta\theta_{t+1})\| \quad [\text{since } w_t = \Pi_S(\eta\theta_t)] && (55)\\ &\leq \|\eta\theta_t - \eta\theta_{t+1}\| \quad [\text{projection decreases distance}] && (56)\\ &= \eta\|z_t\| \quad [\text{since } \theta_{t+1} = \theta_t - z_t]. && (57) \end{aligned}$$

Plugging this bound back completes the proof. The core of this proof is actually quite short; half of it is dealing with the projection.

2.6 Online Subgradient Descent (OGD)

- So far, we have proven regret bounds for FTRL with a quadratic regularizer and linear loss functions. We can derive FTRL for the general case, but we will not do that since we wouldn't be able to compute the solution efficiently anyway. Instead, we will take another approach.
- To get an efficient algorithm, our strategy is use a linear approximation of f_t ($f_t(w) = w \cdot z_t$ for some z_t) and run FTRL as before.
- What linear approximation z_t should we use? Let's just take the subgradient of f_t at the current weights w_t: take $z_t \in \partial f_t(w_t)$. Just to highlight the simplicitly of the algorithm, here it is (OGD is nothing more than FTRL on quadratic regularizers and linear approximations of the losses):
- **Algorithm 4 (Online subgradient descent (OGD))**
  - Let $w_1 = 0$.
  - On iteration t:
    - Predict using w_t.
    - Take any subgradient $z_t \in \partial f_t(w_t)$.
    - If $S = \mathbb{R}^d$, learner performs the update:

$$w_{t+1} = w_t - \eta z_t. \tag{58}$$

    - Otherwise, learner projects cumulative gradients onto S:

$$w_{t+1} = \Pi_S(\eta\theta_{t+1}), \quad \theta_{t+1} = \theta_t + z_t. \tag{59}$$

- We've already bounded the linearized losses of OGD (FTRL), so all we need to do is to relate the linearized losses to the original losses. Here's where convexity comes in a crucial way.

- Since $z_t \in \partial f_t(w_t)$, we have by the definition of subgradient:

$$f_t(u) \geq f_t(w_t) + z_t \cdot (u - w_t). \tag{60}$$

  Rearranging, we get a direct comparison for each term of the regret:

$$\boxed{f_t(w_t) - f_t(u) \leq (w_t \cdot z_t) - (u \cdot z_t).} \tag{61}$$

- The regret we get by running OGD on f_t (what we do) is at most that of running OGD on $w \mapsto w \cdot z_t$. (Both produce the same sequence of w_t). Intuitively, linear functions are the hardest to optimize using online convex optimization.
- Running OGD on $w \mapsto (w \cdot z_t)$ is the same algorithm as FTRL on $w \mapsto (w \cdot z_t)$.
- Therefore, the regret bound for $w \mapsto (w \cdot z_t)$ Theorem 1 also applies to OGD on f_t.
- 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.
  - It's most commonly thought of as just following the gradient, 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 still maintaining very efficient parameter updates.
- **Example 8 (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 ($x_t \in \mathbb{R}^d, y_t \in \{-1, +1\}$):

$$f_t(w) = \max\{0, 1 - y_t(w \cdot x_t)\}. \tag{62}$$

    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.[1]
    - Else: $w_{t+1} = w_t + \eta y_t x_t$.
  - Analysis:

[1] This algorithm is very similar to the Perceptron algorithm; the only difference is that Perceptron just requires any positive margin, not necessarily 1.

* Assume the data points are bounded: $\|x_t\| \leq L$. Then $z_t \in \partial f_t(w_t)$ also satisfies that bound $\|z_t\| \leq L$.
* Assume that expert weights are bounded: $\|u\| \leq B$.
* The regret bound from Theorem 1 is as follows:

$$\sum_{t=1}^{T} [f_t(w_t) - f_t(u)] \leq BL\sqrt{T}. \tag{63}$$

* The above bound gives us control on the hinge loss, but what we really care about is the zero-one loss. But since that the hinge loss is an upper bound on the zero-one loss, we can upper bound the zero-one loss too:

$$\underbrace{\sum_{t=1}^{T} \mathbb{I}[y_t(w_t \cdot x_t) \leq 0]}_{\text{zero-one loss of learner}} \leq \underbrace{\sum_{t=1}^{T} f_t(u)}_{\text{hinge loss of expert}} + BL\sqrt{T}. \tag{64}$$

- **Example 9 (Learning with expert advice)**
  - 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

$$z_t = (\ell(y_t, h_1(x_t)), \cdots, \ell(y_t, h_d(x_t))) \tag{65}$$

      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

$$\boxed{\text{Regret} \leq BL\sqrt{T} = \sqrt{dT}.} \tag{66}$$

  - 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?
  - Later, we will show that we can use another regularizer that better captures our prior knowledge to obtain better bounds.

2.7 Online mirror descent (OMD)

So far, we have analyzed FTRL for quadratic regularizers, which leads to (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 use a non-spherical regularizer. Or in the case of learning with expert advice, where $w \in \Delta$, we might prefer negative entropy as a measure of complexity. In this section, we will develop a general way of obtaining regret bounds for arbitrary convex regularizers. We make make extensive use of **Fenchel duality** and **Bregman divergences**, which allow us to generalize beyond the quadratic regularizer.

- The goal is to analyze FTRL (Algorithm 3) for linear functions and arbitrary regularizers. If we have an arbitrary convex function f_t, we can linearize using a subgradient (use used this trick to derive OGD for quadratic regularizers). The resulting algorithm is this:
- **Algorithm 5 (online mirror descent (OMD))**
  - Let $\theta_t = -\sum_{i=1}^{t-1} z_i$ be the subgradients aggregated over the first $t-1$ examples as usual.
  - On iteration t, the learner chooses weights w_t to minimize the regularized (linearized) loss:

$$w_t \in \arg\min_w \{\psi(w) - w \cdot \theta_t\}. \tag{67}$$

- Examples:
  - Quadratic regularizer: $\psi(w) = \frac{1}{2\eta}\|w\|^2$.
  - Entropic regularizer: $\psi(w) = \frac{1}{\eta}\sum_{j=1}^{d} w_j \log w_j$ for $w \in \Delta_d$ (this is the negative entropy defined on the probability simplex).
  - The difference between the two regularizers is that the entropic regularizer slopes up violently when w approaches the boundary of Δ (the gradient goes to infinity).
  - Remark: in the sequel, we will assume that $S = \mathbb{R}^d$ and use the regularizer to encode the hard constraints on the domain.

[begin lecture 1/23] (1/23)

- We now need to build up some tools to help us analyze OMD. First, we introduce the Fenchel conjugate, a core tool in optimization:
- Fenchel duality
  - **Definition 4 (Fenchel conjugate)**

* Let ψ be a convex function.
* Then the **Fenchel conjugate**[2] of ψ is

$$\boxed{\psi^*(\theta) \stackrel{\text{def}}{=} \max_w \{w \cdot \theta - \psi(w)\}.} \tag{68}$$

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.

– General facts:
* ψ^* is convex (even if ψ is not), since it's just a max over a collection of linear functions.
* $\psi^{**} = \psi$ iff ψ is convex and lower semi-continuous.
* $\psi^*(\theta) \geq w \cdot \theta - \psi(w)$ for all w (**Fenchel-Young inequality**); follows directly from the definition of ψ^*.

– To gain intuition, think of the following setup:
* w is the weight vector, $\psi(w)$ is the regularizer
* θ is the negative cumulative subgradients, $\psi^*(\theta)$ is the negative of the minimum regularized loss

– Property (scaling):
* If $r(w) = a\psi(w)$ with $a > 0$, then $r^*(\theta) = a\psi^*(\theta/a)$.

– **Example 10 (Quadratic regularizer)**
* Let $\psi(w) = \frac{1}{2\eta}\|w\|^2$.
* Then $\psi^*(\theta) = \frac{\eta}{2}\|\theta\|^2$, attained by $w = \eta\theta$.
* In this case, w and θ are simple scalings of each other.

– **Example 11 (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 $w_j = \frac{e^{\eta\theta_j}}{\sum_k e^{\eta\theta_k}}$.
* Note that this is maximum entropy duality in exponential families.
* Here, w and θ represent the mean and canonical parametrization of a multinomial distribution.

– Dual mapping:
* For simplicity, assume ψ and ψ^* are both differentiable (in our application as regularizers, they will be).

[2]Also known as the convex conjugate or Legendre-Fenchel transformation.

* Then there is a one-to-one mapping between weights w and negative cumulative subgradients θ, which are linked via the gradients of ψ and ψ^*:

$$w = \nabla\psi^*(\theta), \quad \theta = \nabla\psi(w). \tag{69}$$

– We can now rewrite OMD using Fenchel conjugacy:

$$\boxed{w_t = \nabla\psi^*(\theta_t).} \tag{70}$$

– Intuition: mirror descent

* Additive updates: $\theta_{t+1} = \theta_t - z_t$ yields sequence of negative cumulative subgradients $\theta_1, \theta_2, \ldots$
* Mapping from θ_t to w_t via $\nabla\psi^*$ (captures effect of regularizer) induces a "mirrored" sequence of weight vectors $w_1, w_2, \ldots$.

• Bregman divergence

– 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.

– Our goal is to use Bregman divergences to define this criteria in a way that exploits structure in the regularizer.

– **Definition 5 (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:

$$\boxed{D_f(w\|u) \stackrel{\text{def}}{=} f(w) - f(u) - \nabla f(u) \cdot (w - u).} \tag{71}$$

* Intuitively, the divergence captures the the error of the linear approximation of f based on a subgradient.

– Property: $D_f(w\|u) \geq 0$ (by definition of convexity).

– **Example 12 (Quadratic regularizer)**

* Let $f(w) = \frac{1}{2}\|w\|^2$.
* Then $D_f(w\|u) = \frac{1}{2}\|w - u\|^2$.

– **Example 13 (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 2 (regret of OMD)**
  - OMD Algorithm 5 obtains the following regret bound:

$$\text{Regret}(u) \leq [\psi(u) - \psi(w_1)] + \sum_{t=1}^{T} D_{\psi^*}(\theta_{t+1}\|\theta_t). \tag{72}$$

- Proof of Theorem 2:
  - The key pivoting quantity is $\psi^*(\theta_{T+1})$, the negative regularized loss of the best fixed expert, which 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:

$$\psi^*(\theta_{T+1}) \geq u \cdot \theta_{T+1} - \psi(u) \tag{73}$$

$$= \sum_{t=1}^{T} [-u \cdot z_t] - \psi(u) \tag{74}$$

    by the Fenchel-Young inequality.
  - The learner:

$$\psi^*(\theta_{T+1}) = \psi^*(\theta_1) + \sum_{t=1}^{T} [\psi^*(\theta_{t+1}) - \psi^*(\theta_t)] \quad \text{[telescoping sums]} \tag{75}$$

$$= \psi^*(\theta_1) + \sum_{t=1}^{T} [\nabla\psi^*(\theta_t) \cdot (\theta_{t+1} - \theta_t) + D_{\psi^*}(\theta_{t+1}\|\theta_t)] \quad \text{[Bregman definition]} \tag{76}$$

$$= \psi^*(\theta_1) + \sum_{t=1}^{T} [-w_t \cdot z_t + D_{\psi^*}(\theta_{t+1}\|\theta_t)] \quad \text{[definition of OMD (70) and } \theta_t\text{]}. \tag{77}$$

    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.

- So far, we have gotten a regret bound in terms of a Bregman divergence. In order to say something about the Bregman divergence, we need to examine some more properties of the regularizer.

[begin lecture 1/28] (1/28)

- **Definition 6 (strong convexity/smoothness)**
  - A function f is α-strongly convex with respect to a norm $\|\cdot\|$ iff

$$D_f(w\|u) \geq \frac{\alpha}{2}\|w-u\|^2. \tag{78}$$

  - A function f is α-strongly smooth with respect to a norm $\|\cdot\|$ iff

$$D_f(w\|u) \leq \frac{\alpha}{2}\|w-u\|^2. \tag{79}$$

  - Intuition:
    - Strong convexity means that f must be growing at least quadratically (L_1 regularizer is not strongly convex).
    - Strong smoothness means that f must be growing slower than some quadratic (L_1 regularizer is not strongly smooth).
  - Examples
    - Quadratic regularizer: $\psi(w) = \frac{1}{2\eta}\|w\|^2$ is $1/\eta$-strongly convex with respect to the 2-norm.
    - 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.

As you might expect, duality links strong convexity and strong smoothness:

- **Lemma 2 (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 $\|\cdot\|_*$.
  - Intuition: look at the quadratic regularizer $\psi(w) = \frac{1}{2\eta}\|w\|^2$ and $\psi^*(\theta) = \frac{\eta}{2}\|\theta\|^2$.
  - By definition of strong smoothness and the fact that $\theta_{t+1} = \theta_t - z_t$,

$$\boxed{D_{\psi^*}(\theta_{t+1}\|\theta_t) \leq \frac{\eta}{2}\|z_t\|_*^2.} \tag{80}$$

Plugging this bound into (72) gives us a general bound on regret of OMD for arbitrary strongly convex regularizers:

$$\boxed{\text{Regret}(u) \leq [\psi(u) - \psi(w_1)] + \frac{\eta}{2}\sum_{t=1}^{T} \|z_t\|_*^2.} \tag{81}$$

Note now (72) generalizes (46) from earlier for quadratic regularizers. We get to measure the size of z_t using the dual norm $\|\cdot\|_*$ rather than the default 2-norm. As we shall see shortly, this allows us to avoid the $\sqrt{d}$ dependence for learning with expert advice.

- **Example 14 (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 $\psi^*(\theta) = \frac{1}{\eta}\sum_{j=1}^{d} e^{\eta\theta_j}$ and $\nabla\psi_j^*(\theta) = \frac{e^{\eta\theta_j}}{\sum_{k=1}^{d} e^{\eta\theta_k}}$.
  - OMD updates:

$$\boxed{w_{t,j} \propto e^{\eta\theta_{t,j}}.} \tag{82}$$

The equivalent recursive formula:

$$\boxed{w_{t+1,j} \propto w_{t,j}e^{-\eta z_{t,j}}.} \tag{83}$$

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.

- **Example 15 (EG for learning with expert advice)**
  - Consider learning with expert advice (Example 5).
  - 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 $\|z_t\|_\infty \leq 1$.
  - Also, the regularizer ranges:
    - $\max_w \psi(w) = 0$ (minimum entropy)
    - $\min_w \psi(w) = \frac{\log(1/d)}{\eta} = \frac{-\log d}{\eta}$ (maximum entropy)
  - Then

$$\text{Regret} = \frac{\log(d)}{\eta} + \frac{\eta T}{2}. \tag{84}$$

- Setting $\eta = \sqrt{\frac{2\log d}{T}}$ yields:

$$\boxed{\text{Regret} = \sqrt{2\log(d)T}.} \tag{85}$$

- To compare with quadratic regularization (OGD):
  - Quadratic: $[\max_u \psi(u) - \min_u \psi(u)] = 1$, $\|z_t\|_2 \leq \sqrt{d}$
  - Entropic: $[\max_u \psi(u) - \min_u \psi(u)] = \log d$, $\|z_t\|_\infty \leq 1$
- 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. Let's look into this a bit more carefully. Central to the bounds are notions of norms and distances.
  - First some intuitions:
    - L_p norms decrease as p increases:

$$\|w\|_1 \geq \|w\|_2 \geq \|w\|_\infty. \tag{86}$$

    - L_p balls increase in size as p increases:

$$\{w : \|w\|_1 \leq B\} \subset \{w : \|w\|_2 \leq B\} \subset \{w : \|w\|_\infty \leq B\}. \tag{87}$$

    - The difference between the norms can be huge. For example, take $w = (1, \ldots, 1) \in \mathbb{R}^d$. Then $\|w\|_1 = d$, $\|w\|_2 = \sqrt{d}$, $\|w\|_\infty = 1$.
  - In OGD, we were just assuming the weights lived in a ball $\|u\|_2 \leq 1$ (this is too loose by a factor of $\sqrt{d}$ because we know u lives in the simplex, which is than the L^2 ball).
  - Using the 2-norm means that we use the bound $\|z_t\|_2 \leq \sqrt{d}$. To get rid of the dependence on d here, we have to use the L_∞ norm, as $\|z_t\|_\infty \leq 1$.
  - However, this means that ψ has strongly convex with respect to the L_1 norm. The quadratic regularizer isn't strong enough (only $1/d$-strongly convex with respect to L_1), 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 because the simplex is small, $[\psi(u) - \psi(w_1)]$ only grows from 1 (with the quadratic regularizer) to $\log d$ (with the entropic regularizer).
  - So the tradeoff was definitely to our advantage.

2.8 Multi-armed bandits

- The multi-armed bandit problem is basically an online learning problem with partial feedback. Because the learner now has incomplete information, it must actively explore to get more information, thus exposing the challenges of managing the exploration/exploitation tradeoff.

- Setup

  - 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 $t = 1, \ldots, T$:
    - Learner chooses a distribution $w_t \in \Delta_d$ over actions, and samples an action $a_t \sim w_t$.
    - Nature reveals the loss of *only that particular action* z_{t,a_t} and no others.
  - We will define regret to be the expected loss, just as in online learning with expert advice:

$$\text{Regret} = \max_{w \in \Delta_d} \sum_{t=1}^{T} [w_t \cdot z_t - w \cdot z_t]. \tag{88}$$

    Note that taking the max over randomized experts $w \in \Delta_d$ 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 to obtain a regret bound of Regret $\leq \sqrt{2\log(d)T}$ (see Example 15).
  - 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 we can observe, defined as follows:

$$\boxed{\hat{z}_{t,a} = \begin{cases} \frac{z_{t,a}}{w_{t,a}} & \text{if } a = a_t \\ 0 & \text{otherwise.} \end{cases}} \tag{89}$$

  - It's easy to check that $\hat{z}_t$ is an unbiased estimate of z_t: $\mathbb{E}_{a_t \sim w_t}[\hat{z}_t] = z_t$. Note that dividing $w_{t,a}$ compensates for sampling $a_t \sim w_t$.

- **Algorithm 6 (Bandit-EG)**

  - Run EG with the unbiased esimate $\hat{z}_t$ rather than z_t.

- **Theorem 3 (Bandit-EG analysis)**
  - Bandit-EG obtains expected regret (expectation taken over learner's randomness) of

$$\boxed{\mathbb{E}[\text{Regret}] \leq 2\sqrt{d\log(d)T}.} \tag{90}$$

- Comparison of Bandit-EG (bandit setting) and EG (online learning setting):
  - Compared with the bound for EG in the full information online learning setting (85), we see that this bound (90) 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 3:
  - In the analysis of EG, our bound depended on $\|z_t\|_\infty^2$. Intuitively, we need to replace z_t with $\hat{z}_t$, but there are two additional considerations:
    - $\hat{z}_t$ is random, so we need to take expectations.
    - While $z_t \in [0,1]^d$, $\hat{z}_t$ is actually unbounded because of dividing by $w_{t,a}$, which could be really small.
  - To address these two issues, we appeal to two theorems which we will not prove here (though both can be found in the Shalev-Shwartz survey paper):
    - Improved bound using local norms (Theorem 2.22 of Shalev-Shwartz):
      - For EG (assuming non-negative losses $z_t \succeq 0$), we have:

$$\text{Regret} \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \sum_{a\in[d]} w_{t,a} z_{t,a}^2. \tag{91}$$

      - Note that $\sum_{a\in[d]} w_{t,a} z_{t,a}^2 \leq \max_{a\in[d]} z_{t,a}^2 = \|z_t\|_\infty^2$. So by using the local norm (with respect to w_t) rather than the L_∞ norm, we get a bound stronger than the one we obtained for EG using the OMD analysis (81).
      - This allows us to have large loss gradients $z_{t,a}$ provided that $w_{t,a}$ is small (this is important).
      - The proof exploits additional structure of the entropic regularizer.
    - Noisy gradients (Theorem 4.1 of Shalev-Shwartz):

· If EG is run with the unbiased estimate $\hat{z}_t$ ($z_t = \mathbb{E}[\hat{z}_t]$), then the expected regret is simply:

$$\mathbb{E}[\text{Regret}] \leq \frac{\log d}{\eta} + \eta \sum_{t=1}^{T} \mathbb{E}_{a_t \sim w_t} \left[\sum_{a \in [d]} w_{t,a} \hat{z}_{t,a}^2 \right]. \tag{92}$$

This theorem can be proved easily by taking expectations.

– So now, we just have to compute the expected local norm:

$$\mathbb{E}_{a_t \sim w_t} \left[\sum_{a \in [d]} w_{t,a} \hat{z}_{t,a}^2 \right] = \sum_{a \in [d]} w_{t,a}^2 \left(\frac{z_{t,a}^2}{w_{t,a}^2} \right) \leq d. \tag{93}$$

– 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.

2.9 Online-to-batch conversion

- So far, we have been focused 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 **generalization error**.

- In this section, we will show that low regret implies low generalization error 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 $S \subset \mathbb{R}^d$.

  – As in online learning, we define a 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 **generalization error** of a weight vector $w \in S$ is defined as follows:

$$\boxed{L(w) = \mathbb{E}_{z \sim p^*}[\ell(z, w)].} \tag{94}$$

- Define the weight vector that minimizes the generalization error:

$$w^* \in \arg\min_{w \in S} L(w). \tag{95}$$

- The batch learner gets T i.i.d. training examples $(z_1, \ldots, z_T)$, 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 7 (Online-to-batch conversion)**
  - Input: T training examples $z_1, \ldots, z_T$.
  - Iterate $t = 1, \ldots, T$:
    - 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 (is in the sigma-algebra of) $z_{1:t-1}$.
  - Note that $L(w^*)$ is not random.

- **Theorem 4 (Online-to-batch conversion)**
  - Recall the usual definition of regret (here, we are explicit about the max over possible sequences of examples $z_{1:T}$):

$$\text{Regret}(u) = \max_{z_{1:T} \in \mathcal{Z}^T} \sum_{t=1}^{T} [f_t(w_t) - f_t(u)]. \tag{96}$$

  - Online-to-batch conversion obtains the following expected generalization error:

$$\boxed{\mathbb{E}[L(\bar{w})] \leq L(w^*) + \frac{\text{Regret}(w^*)}{T}.} \tag{97}$$

- Interpretation: the generalization error of the online-to-batch conversion $L(\bar{w})$ is bounded by the best possible generalization error (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 generalization error of w_t.

$$\mathbb{E}[f_t(w_t) \mid w_t] = L(w_t). \tag{98}$$

    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:

$$L(\bar{w}) \leq \frac{1}{T} \sum_{t=1}^{T} L(w_t). \tag{99}$$

  - Now, the rest is just putting the pieces together. Putting (99) and (98) together:

$$L(\bar{w}) \leq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}[f_t(w_t) \mid w_t]. \tag{100}$$

    Adding and subtracting $L(w^*)$ to the RHS, noting that $L(w^*) = \mathbb{E}[f_t(w^*)]$:

$$L(\bar{w}) \leq L(w^*) + \frac{1}{T} \sum_{t=1}^{T} (\mathbb{E}[f_t(w_t) \mid w_t] - \mathbb{E}[f_t(w^*)]). \tag{101}$$

  - Taking expectations on both sides:

$$\mathbb{E}[L(\bar{w})] \leq L(w^*) + \frac{1}{T} \sum_{t=1}^{T} (\mathbb{E}[f_t(w_t) - f_t(w^*)]). \tag{102}$$

  - The second term of the RHS is upper bounded by the regret, so we have:

$$\mathbb{E}[L(\bar{w})] \leq L(w^*) + \frac{\text{Regret}(w^*)}{T}. \tag{103}$$

- Remarks

- The online-to-batch conversion theorem says if you run online subgradient descent once over your training data,[3] you should expect $O(\sqrt{T})$ regret by our previous analyses.
- The resulting average weight vector will, *in expectation*,[4] have a generalization error 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.

2.10 Summary

- This concludes our tour of online learning.
- 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 $= o(T)$.
- 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 measure).
- 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 weighted 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

[3] In practice, it helps to run the online learning algorithm multiple times over the training data.
[4] You can get high probability bounds too, but we won't discuss those here.

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.

- In the bandit setting with partial information, we get $O(\sqrt{d\log(d)T})$ regret using local norms to control the size of the gradients.
- Finally, we showed that learners with low regret in the online setting directly lead to learners with low **generalization error** in the batch setting via an online-to-batch conversion.

2.11 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.

[begin lecture 1/30] (1/30)

3 Kernel methods

3.1 Motivation

- So far, we have considered only 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.
- Fortunately, all our algorithms so far (e.g., online gradient descent) actually only require the linearity in w to ensure that the loss functions are convex, not x. So we can sneakily replace x with an arbitrary feature vector $\phi(x) \in \mathbb{R}^d$.
  - Example: $\phi(x) = (1, x, x^2)$ for $x \in \mathbb{R}$
  - Example: $\phi(x) = (\text{count of } a \text{ appearing in } x, \dots)$ for a string x.
- Note that x does not even need to be a real vector. In general, we assume that $x \in \mathcal{X}$, where $\mathcal{X}$ is all possible inputs (we won't assume any further structure on $\mathcal{X}$ for now).
- We can actually represent very expressive **non-linear** functions by simply augmenting $\phi(x)$, but the problem is that $\phi(x)$ would have to be very high-dimensional in order to attain the desired degree of expressiveness, resulting in computationally expensive algorithms. A secondary consideration is that for some problems, it might be hard to design good features directly.
- Kernels address the two issues above above by offering:
  - 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.
- Before we define kernels formally, let's provide some intuition.
  - Consider online gradient descent on, say, the squared loss: $\ell(x, y, w) = \frac{1}{2}(y - \langle w, \phi(x) \rangle)^2$.
  - The weight update is ($w_1 = 0$):

$$w_{t+1} = w_t + \underbrace{\eta_t(y_t - \langle w_t, \phi(x_t) \rangle)}_{\stackrel{\text{def}}{=} \alpha_t} \phi(x_t). \tag{104}$$

- We see that the weight vector will always be a linear combination of the feature vectors (we will revisit this property in much greater generality when we study the representer theorem):

$$w_t = \sum_{i=1}^{t-1} \alpha_i \phi(x_i), \tag{105}$$

and the prediction is:

$$\langle w_t, \phi(x_t) \rangle = \sum_{i=1}^{t-1} \alpha_i \langle \phi(x_i), \phi(x_t) \rangle. \tag{106}$$

Note that predictions only depend on the **inner product** between the feature vectors.

- The algorithmic tradeoff of expressing online gradient descent in this peculiar way is that we store the α_t's (T numbers) rather than the w (d numbers). If d is much larger than T, then we win. On the other hand, if we have a lot of data (T), kernel methods can be quite slow, and we must resort to approximations.
- Example ($x \in \mathbb{R}^b$)
  - Feature map with all quadratic terms:

$$\phi(x) = (x_1^2, \ldots, x_b^2, \sqrt{2}x_1x_2, \ldots, \sqrt{2}x_1x_b, \sqrt{2}x_2x_3, \ldots, \sqrt{2}x_2x_b, \ldots, \sqrt{2}x_{b-1}x_b), \tag{107}$$

  - Naive computation: $O(b^2)$ time.
  - Observation: $\langle \phi(x), \phi(x') \rangle = \langle x, x' \rangle^2$, which takes $O(b)$ time.
- It's important to realize that mathematically, we're still running online gradient descent, and that all we've done is perform a computational sleight of hand, known as the **kernel trick**.
- Note: kernels are not always needed to compute dot products efficiently in high dimensions. If $\phi(x)$ is sparse (as is often the case in natural language processing), and has s non-zeros, then $\langle \phi(x), \phi(x') \rangle$ can be computed in $O(s)$ time rather than $O(d)$ time.

- Summary thus far
  - We start with a feature map $\phi : \mathcal{X} \mapsto \mathbb{R}^d$.
  - We can recast algorithms such as online gradient descent in a way so that they only depend on the inner product $\langle \phi(x), \phi(x') \rangle$.
  - We can sometimes compute this inner product efficiently.

- 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 a similarity $k(x, x')$ between two examples x and x'. Sometimes, similarities might be more convenient from a modeling point of view.
- But how can we construct a k so that we are sure it defines an inner product with respect to some feature map ϕ? As we'll show later, the right characterization of such k are positive semidefinite kernels.
- **Definition 7 (kernel)**
  - A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a positive semidefinite kernel (or more simply, a kernel) iff for every finite set of points $x_1, \ldots, x_n \in \mathcal{X}$, the **kernel matrix** $K \in \mathbb{R}^{n \times n}$ defined by $K_{ij} = k(x_i, x_j)$ is positive semidefinite.
- **Theorem 5 (Kernels define inner products)**
  - For every kernel k satisfying Definition 7, there exists an inner product space $\mathcal{H}$ and a feature map $\phi : \mathcal{X} \mapsto \mathcal{H}$ such that

$$\boxed{k(x, x') = \langle \phi(x), \phi(x') \rangle} \tag{108}$$

    and vice-versa.
- We won't prove Theorem 5 in full generality quite yet since it requires more machinery, but to get some intuition, let's prove it for the case where the number of inputs $\mathcal{X}$ is finite.
  - Let $\mathcal{X} = \{x_1, \ldots, x_n\}$.
  - A positive semidefinite kernel matrix $K \in \mathbb{R}^{n \times n}$ has an eigendecomposition, $K = UDU^\top$.
  - Defining the feature matrix $\Phi = UD^{1/2}$ where the i-th row of Φ is a feature vector $\phi(x_i)$, we see that $K = \Phi\Phi^\top$ indeed is an inner product.
  - Given an arbitrary feature matrix $\Phi \in \mathbb{R}^{n \times d}$, $K = \Phi\Phi^\top$ is positive semidefinite.
- Next, we give some examples of kernels. Later, we will revisit Theorem 5 from a different viewpoint, that of reproducing kernel Hilbert spaces.

3.2 Kernels

- We will first give some examples of kernels, and then prove that they are indeed valid according to Definition 7.
- We can visualize a kernel for $\mathcal{X} = \mathbb{R}$ by fixing x to some value (say 1) and plotting $k(x, x')$ against x'.
- Examples (assume the input space is $\mathcal{X} \in \mathbb{R}^b$)
  - Linear kernel:

$$k(x, x') = \langle x, x' \rangle . \tag{109}$$

    * This is the "no-kernel" kernel.
  - Polynomial kernel:

$$k(x, x') = (1 + \langle x, x' \rangle)^p. \tag{110}$$

    * Intuition: for boolean features, 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:

$$k(x, x') = \exp\left(\frac{-\|x - x'\|^2}{2\sigma^2}\right). \tag{111}$$

    * 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, so computationally, we really have no choice but to work with kernels.
- Non-example: $k(x, x') = \mathbb{I}[\|x - x'\| \leq 1]$
  - Exercise: show that k is not a kernel function.
- Now we show the above kernels are actually valid (positive semidefinite). Let $x_1, \ldots, x_n \in \mathcal{X}$ be any points.
- Base case: for any function $f : \mathcal{X} \mapsto \mathbb{R}$, $k(x, x') = f(x)f(x')$ is positive semidefinite.
  - Proof: the kernel matrix is $K = uu^\top \succeq 0$, 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: $k(x, x') = k_1(x, x') + k_2(x, x')$
  - Since positive semidefiniteness is closed under addition, $K = K_1 + K_2 \succeq 0$.
- Product: $k(x, x') = k_1(x, x')k_2(x, x')$
  - $K = K_1 \circ K_2$ is the corresponds to elementwise product.
  - Since K_1, K_2 are positive semidefinite, we can take their eigendecompositions:
    - $K_1 = \sum_{i=1}^n \lambda_i u_i u_i^\top$
    - $K_2 = \sum_{i=1}^n \lambda_i' v_i v_i^\top$
  - Taking the elementwise product yields the following eigendecomposition, showing that K is also positive semidefinite:
    - $K = \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j' (u_i \circ v_j)(u_i \circ v_j)^\top$

- 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, add 1 (which is a kernel), and take $p - 1$ products.
  - Gaussian kernel:
    - Rewrite
    $$k(x, x') = \exp\left(\frac{-\|x\|^2}{2\sigma^2}\right) \exp\left(\frac{-\|x'\|^2}{2\sigma^2}\right) \exp\left(\frac{\langle x, x' \rangle}{\sigma^2}\right). \tag{112}$$
    - The first two factors is handled by the base case.
    - For the third factor, take the Taylor expansion:
    $$\exp\left(\frac{\langle x, x' \rangle}{\sigma^2}\right) = 1 + \frac{\langle x, x' \rangle}{\sigma^2} + \frac{1}{2}\frac{\langle x, x' \rangle^2}{\sigma^2} + \frac{1}{6}\frac{\langle x, x' \rangle^3}{\sigma^2} + \cdots \tag{113}$$
    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 are closed).

3.3 Reproducing kernel Hilbert spaces (RKHS)

- So far, we have thought about the feature map view and the kernel view of learning. Now, we will introduce a third view based on functions. This won't have any algorithmic implications, but will provide more mathematical insight into the problem.

- Recall that in ridge regression, we are given n examples $(x_i, y_i)_{i=1}^n$ and we want to find a weight vector that minimizes the regularized loss:
$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^n (\langle w, \phi(x_i) \rangle - y_i)^2 + \lambda \|w\|_2^2. \tag{114}$$
Note that the weight vector defines a prediction function $f(x) = \langle w, \phi(x) \rangle$.

- But really, at the end of the day, all we want is a function f, so let's write our optimization problem like this:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2, \tag{115}$$

  where $\mathcal{H}$ is a RKHS, which we'll define shortly. Rather than working with weight vectors and using them to define prediction functions, we will work with the functions directly, which is mathematically more elegant.

- **Definition 8 (Hilbert space)**
  - An inner product space $\mathcal{H}$ is a complete[5] vector space equipped with an **inner product** $\langle \cdot, \cdot \rangle : \mathcal{H} \times \mathcal{H} \mapsto \mathbb{R}$ 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: $\langle f, f \rangle \geq 0$ 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
    - 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 $[0, 1]$: $L^2([0, 1]) = \{f : \int_0^1 f(x)^2 < \infty\}$, with $\langle f, g \rangle = \int_0^1 f(x) g(x) dx$.

- However, general Hilbert spaces are not suitable for learning because they are too unconstrained.
  - For example, consider $\mathcal{H} = L^2([0, 1])$, and take $f, g \in \mathcal{H}$ which differ at a single point $x \in [0, 1]$.
  - Then the two functions have the same norm ($\|f\|_{\mathcal{H}} = \|g\|_{\mathcal{H}}$), and if x did not appear in the training data, then we would be completely indifferent to f versus g.
  - Being indifferent to pointwise evaluations is a very bad property 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.

- **Definition 9 (reproducing kernel Hilbert space)**

[5] See Definition 23.

- A reproducing kernel Hilbert space $\mathcal{H}$ is a Hilbert space over functions $f : \mathcal{X} \mapsto \mathbb{R}$ such that for each $x \in \mathcal{X}$, the **evaluation functional** $L_x : \mathcal{H} \mapsto \mathbb{R}$ defined by

$$L_x(f) \stackrel{\text{def}}{=} f(x) \tag{116}$$

is bounded (equivalently, continuous):

$$L_x(f) \leq M_x \|f\|_{\mathcal{H}} \text{ for all } f \in \mathcal{H}. \tag{117}$$

- Note that this rules out $L^2([0,1])$ because the function f which is zero everywhere except for $f(x) = 1$ has $\|f\|_{\mathcal{H}} = 0$, which would mean that $f(x) = L_x(f) \leq 0$ no matter what M_x was.

- **Theorem 6 (RKHS defines a kernel)**
  - Every RKHS $\mathcal{H}$ over functions $f : \mathcal{X} \mapsto \mathbb{R}$ defines a unique kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

- Proof (construction of the kernel)
  - Note that $L_x(f)$ is *linear*: $L_x(cf) = cL_x(f)$ and $L_x(f+g) = f(x) + g(x)$.
  - 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 some 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**:

$$\boxed{f(x) = \langle R_x, f \rangle \text{ for all } f \in \mathcal{H}.} \tag{118}$$

    In other words, function evaluations can be expressed as inner products.
  - Now let's define a function k:

$$\boxed{k(x, x') \stackrel{\text{def}}{=} R_x(x').} \tag{119}$$

  - We can show that k defined in (119) is a kernel according to Definition 7:
    - Setting $f = R_x$, we have that $k(x, x') = R_x(x') = \langle R_x, R_{x'} \rangle$. This shows that k corresponds to an inner product.
    - Take any $x_1, \ldots, x_n \in \mathcal{X}$ and define the matrix K with entries $K_{ij} = k(x_i, x_j)$.
    - For any $\alpha \in \mathbb{R}^n$,

$$\alpha^\top K \alpha = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \left\langle R_{x_i}, R_{x_j} \right\rangle = \left\langle \sum_{i=1}^n \alpha_i R_{x_i}, \sum_{j=1}^n \alpha_j R_{x_j} \right\rangle \geq 0. \tag{120}$$

      Here, we used the fact that $k(x, x') = \langle R_x, R_{x'} \rangle$, linearity of the inner product, and positive definiteness of the inner product. Therefore, k is a valid kernel.

* In fact, the same argument holds for any function k such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$ (here, we have just used $\phi(x) = R_x$), so this proves the reverse direction of Theorem 5.

– 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, the hallmark of a kernel.

Now, the converse (covered in section):

- **Theorem 7 (Moore-Aronszajn theorem)**

  – Every kernel k corresponds to 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

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x), \tag{121}$$

  for all $n, \alpha_{1:n}, x_{1:n}$. By construction, $\mathcal{H}_0$ is a vector space (not necessarily complete though).

  – 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:

$$\langle f, g \rangle \stackrel{\text{def}}{=} \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(x_i, x'_j). \tag{122}$$

  * With this definition, we can identify a representer R_x for each $x \in \mathcal{X}$ as follows. Taking $g = k(x, \cdot)$, we have that

$$\langle f, k(x, \cdot) \rangle = \sum_{i=1}^{n} \alpha_i k(x_i, x) = f(x), \tag{123}$$

  which is exactly the reproducing property with representer $R_x = k(x, \cdot)$.

  We still need to check that our definition of $\langle \cdot, \cdot \rangle$ satisfies the properties of an 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.
  - If $\langle f, f \rangle = 0$, then for all $x \in \mathcal{X}$:

$$f(x) = \langle f, k(x, \cdot) \rangle \leq \|f\|_{\mathcal{H}_0} k(x, x)^{1/2} = 0, \tag{124}$$

    by the reproducing property and the Cauchy-Schwartz inequality.

– Finally, let $\mathcal{H}$ be the completion of $\mathcal{H}_0$ (by including all limit points of sequences in $\mathcal{H}_0$).

– This proves the forward direction of Theorem 5 because the RKHS $\mathcal{H}$ is an inner product space by construction.

[begin lecture 2/4] (2/4)

- Summary
  - A feature map $\phi : \mathcal{X} \mapsto \mathcal{H}$: maps points in $\mathcal{X}$ to some inner product space $\mathcal{H}$.
  - A (positive semidefinite) kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$: every derived kernel matrix K is positive semidefinite
  - A reproducing kernel Hilbert Space (RKHS) $\mathcal{H}$ containing functions $f : \mathcal{X} \mapsto \mathbb{R}$ 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)

3.4 Learning using kernels

- We have established that kernels k provide a space of functions $\mathcal{H}$; this is the hypothesis class.[6] Now let's talk about learning, which is about combining the hypothesis class $\mathcal{H}$ with actual data.

- A general learning problem can be posed as the following optimization problem:

$$f^* \in \arg\min_{f \in \mathcal{H}} L(\{(x_i, y_i, f(x_i))\}) + \Omega(\|f\|_{\mathcal{H}}^2), \tag{125}$$

  where

[6] 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.

- $L : (\mathcal{X} \times \mathcal{Y} \times \mathbb{R})^n \mapsto \mathbb{R}$ is an arbitrary loss function on n examples.
  - Example (regression): $L(\{(x_i, y_i, f(x_i))\}) = \sum_{i=1}^n \frac{1}{2}(f(x_i) - y_i)^2$.
- $\Omega : [0, \infty) \mapsto \mathbb{R}$ is a strictly increasing function (regularizer).
  - Example (quadratic): $\Omega(\|f\|_{\mathcal{H}}^2) = \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2$.

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 8 (representer theorem)**
  - Let V denote the span of the representers of the training points:

$$V \stackrel{\text{def}}{=} \text{span}(\{k(x_i, \cdot) : i = 1, \ldots, n\}) = \left\{ \sum_{i=1}^n \alpha_i k(x_i, \cdot) : \alpha \in \mathbb{R}^n \right\}. \tag{126}$$

  - Then all minimizers f^* of (125) satisfy $f^* \in V$.

- Proof
  - Define the orthogonal complement:

$$V_\perp = \{g \in \mathcal{H} : \langle f, g \rangle = 0 \text{ for all } f \in V\}. \tag{127}$$

  - Any $f \in \mathcal{H}$ can be decomposed in to a part in the span of the examples and an orthogonal part:

$$f^* = f + f_\perp, \tag{128}$$

    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 $\{f^*(x_j) : j = 1, \ldots, n\}$, which can be written as:

$$f^*(x_j) = f(x_j) + \langle f_\perp, k(x_j, \cdot) \rangle . \tag{129}$$

    The second term is zero, so the loss doesn't depend on $f_\perp$.
  - The regularizer:

$$\Omega(\|f^*\|_{\mathcal{H}}^2) = \Omega(\|f\|_{\mathcal{H}}^2 + \|f_\perp\|_{\mathcal{H}}^2). \tag{130}$$

    Since Ω is strictly monotonic and f^* is a minimizer, we must have $f_\perp = 0$.

– Therefore, $f^* \in \mathcal{V}$.

- 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.
- Kernelized ridge regression
  - Recall the optimization problem for regression:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} \frac{1}{2}(f(x_i) - y_i)^2 + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2. \tag{131}$$

    By the representer theorem, we have the equivalent optimization problem:

$$\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \frac{1}{2}((K\alpha)_i - y_i)^2 + \frac{\lambda}{2}\alpha^\top K\alpha, \tag{132}$$

    where $K \in \mathbb{R}^{n \times n}$ is the kernel matrix.
  - Letting $Y \in \mathbb{R}^n$ denote the vector of outputs, we have:

$$\min_{\alpha \in \mathbb{R}^n} \frac{1}{2}\|K\alpha - Y\|_2^2 + \frac{\lambda}{2}\alpha^\top K\alpha. \tag{133}$$

    Differentiating with respect to α and setting to zero:

$$K(K\alpha - Y) + \lambda K\alpha = 0. \tag{134}$$

    Rearranging:

$$K(K + \lambda I)\alpha = KY. \tag{135}$$

    Solving yields a solution:

$$\boxed{\alpha = (K + \lambda I)^{-1}Y.} \tag{136}$$

    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

$$y = c^\top \alpha. \tag{137}$$

- SVM classification

- This was done in CS229, so we won't go through it again.
- Primal ($y_i \in \{-1, +1\}$):

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} \max\{0, 1 - y_i f(x_i)\} + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2. \tag{138}$$

- Dual (define $\tilde{K}_{ij} = y_i y_j K_{ij}$):

$$\min_{\alpha \in \mathbb{R}^n} -\mathbf{1}^\top \alpha + \alpha^\top \tilde{K} \alpha \quad \text{subject to} \quad 0 \leq \alpha_i \leq \frac{1}{\lambda}, Y^\top \alpha = 0. \tag{139}$$

  The dual is computed by taking the Lagrange dual of the primal optimization problem.

- 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 $x_1, \ldots, x_n$ and a feature map $\phi : \mathcal{X} \mapsto \mathbb{R}^d$.
    - Assume that the data points are centered at zero: $\sum_{i=1}^n \phi(x_i) = 0$.
    - Define the empirical covariance matrix as follows:

$$C \stackrel{\text{def}}{=} \frac{1}{n} \Phi^\top \Phi, \tag{140}$$

      where the i-th row of Φ is $\phi(x_i)$.
    - Then PCA seeks to find a eigendecomposition of C:

$$Cv = \lambda v. \tag{141}$$

      In practice, we will compute the eigendecomposition of C and take the first r eigenvectors $v_1, \ldots, v_r$ (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:

$$\left\| \sum_{i=1}^{r} \langle \phi(x), v_i \rangle \, v_i - \phi(x) \right\|_2^2. \tag{142}$$

      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 $v = \sum_{i=1}^{n} \alpha_i \phi(x_i)$, so an equivalent characterization is to project the vectors on the data points:

$$\Phi C v = \lambda \Phi v. \tag{143}$$

* Using the definition of C and the fact that $\Phi v = K\alpha$, we have

$$K^2 \alpha = \lambda n K \alpha. \tag{144}$$

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):

$$\boxed{K\alpha = \lambda n \alpha.} \tag{145}$$

* 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) $C : \mathcal{H} \mapsto \mathcal{H}$. The intuition is the same.
* Recall the principal component is now a function $v \in \mathcal{H}$ with $\|v\|_{\mathcal{H}} = 1$,[7] where $v(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$.
* The squared reconstruction error of a new point x is:

$$\|v(x)v - \phi(x)\|_{\mathcal{H}}^2 = v(x)^2 - 2v(x)^2 + k(x,x) = k(x,x) - v(x)^2. \tag{146}$$

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 okay, because the point of kernels is really 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.

[7] To make v a unit vector, we just rescale v by $\|v(x)\|_{\mathcal{H}}^{-1}$.

3.5 Gaussian processes

- First, we developed RKHSes so that we could talk about the space of functions $\mathcal{H}$ we'd like to use to model our data.
- We then saw how the inner product and reproducing kernel structure enabled us to do learning from an optimization point of view, where the goal is to recover the best $f \in \mathcal{H}$.
- Now, we will take RKHSes in a **Bayesian** direction, resulting in Gaussian processes, in which we will define a prior over functions and compute posteriors over functions given our data.
- One of the most useful aspects of working with a full posterior is that we can talk about **uncertainty** over predictions in a natural way.
- **Definition 10 (Gaussian process)**
  - A **Gaussian process** is a collection of random variables $\{f(x) : x \in \mathcal{X}\}$, such that any finite subset has a joint multivariate Gaussian distribution.
  - Treat f as a random function, and we write

$$f \sim \mathrm{GP}(0, k), \tag{147}$$

    where $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is the **covariance** (kernel) function.[8]
  - **Marginalization** property: function values are jointly distributed as follows:

$$[f(x_1), \ldots, f(x_n)] \sim \mathcal{N}(0, K), \tag{148}$$

    where K is the usual kernel matrix ($K_{ij} = k(x_i, x_j)$).
- Visualization
  - We can sample Gaussian processes to visualize what kind of functions are actually encoded in our prior (kernel/covariance function). This can be useful even if you are not taking a Bayesian approach.
  - (Demo: show linear, polynomial, Laplace, Gaussian, etc.)
  - Later, we'll examine how the spectral properties of the kernel relate to smoothness properties of the functions.
- We now consider Gaussian process regression, which is basically kernelized regression with a prior over $f \in \mathcal{H}$. Here's the model:

[8] In practice, one can add a mean function but we use 0 for clarity of presentation.

- Prior ($\mu(F)$):[9]

$$f \sim \mathrm{GP}(0, k). \tag{149}$$

- Likelihood ($p(y_i \mid x_i, f)$):

$$y_i \sim \mathcal{N}(f(x_i), \sigma^2). \tag{150}$$

- Posterior (normalized product of prior and likelihood):

$$\mu(df \mid X, Y) \propto \prod_{i=1}^{n} p(y_i \mid x_i, f)\mu(df). \tag{151}$$

- Notation
  - $X \in \mathcal{X}^n$ is the vector of training inputs.
  - $Y \in \mathbb{R}^n$ is the vector of training outputs.
  - $x \in \mathcal{X}$: new input
  - $y \in \mathbb{R}$: new output
  - $K \in \mathbb{R}^n$ is the usual kernel matrix over training points: $K_{ij} = k(x_i, x_j)$.
  - $c \in \mathbb{R}^n$ are the kernel evaluations of the training data against the new point x: $c_i = k(x_i, x)$.

- Suppose we want to predict y at a new point x given the training data (X, Y). The predictive distribution is:

$$p(y \mid x, X, Y) = \int_{\mathcal{H}} p(y \mid x, f)\mu(df \mid X, Y), \tag{152}$$

  where we are marginalizing out the Gaussian process f with respect to the posterior $\mu(\cdot \mid X, Y)$.

- In general, this kind of computation would be intractable, but for regression, we can actually get a closed form solution by exploiting the marginalization property of the GP. First form the augmented kernel matrix (this marginalizes out f):

$$\begin{pmatrix} Y \\ y \end{pmatrix} \mid X, x \sim \mathcal{N}\left(0, \begin{pmatrix} K + \sigma^2 I & c \\ c^\top & k(x,x) + \sigma^2 \end{pmatrix}\right). \tag{153}$$

- Now, it's just a matter of computing the conditional distribution of ordinary multivariate Gaussians.

$$y \mid Y, X, x \sim \mathcal{N}\left(\underbrace{c(K + \sigma^2 I)^{-1} Y}_{\text{mean}}, \underbrace{(k(x,x) + \sigma^2 I) - c^\top (K + \sigma^2 I)^{-1} c}_{\text{variance}}\right). \tag{154}$$

[9] We use $\mu(F)$ to denote the Gaussian process prior measure over functions $F \subset \mathcal{H}$.

- Note that before, we had called $\alpha = (K + \sigma^2)^{-1}Y$, so the mean $c^\top \alpha$ of the posterior over y is exactly the same as in kernelized ridge regression![10] As a bonus, we get a variance estimate which captures how confident we are about our prediction.
  - Intuition: $k(x, x) + \sigma^2 I$ would be the variance over y in the absence of any other data. The $c^\top (K + \sigma^2 I)^{-1} c$ term is the reduction in variance, which is based on the correlation with the training points.

[begin lecture 2/6] (2/6)

- Example
  - Assume no noise ($\sigma^2 = 0$).
  - Suppose we observed one point ($f(x_1)$), and we're asking for the conditional distribution of $f(x)$.
  - Then the augmented kernel matrix is:

$$\begin{pmatrix} s_1^2 & \rho s_1 s \\ \rho s_1 s & s^2. \end{pmatrix}, \tag{155}$$

    where ρ is the correlation between $f(x_1)$ and $f(x)$.
  - Then the variance of $f(x)$ is $s^2 - \rho^2 s^2 = s^2(1 - \rho^2)$. This is the marginal variance of $f(x)$ (s^2) scaled by $1 - \rho^2$, which is small when there is more correlation with $f(x_1)$.
- Computational tractability
  - Gaussian processes can also be used for **classification** as well, but we no longer get nice closed form solutions like we do in regression.
  - As a result, we resort to sampling or variational approximations.
- Utility of uncertainty estimates
  - By maintaining a posterior, GP-based models allow us to talk about the uncertainty over function values (indeed, $f(x)$ is a random variable).
  - One compelling use case is in online learning or optimization of a function. Here, the learner does not simply get n labeled examples as in batch learning. Rather, the learner chooses $x \in \mathcal{X}$, observes $f(x)$ (we assume no noise for simplicity), and repeats.
  - This is similar to the multi-armed bandit setting, with the exception that $\mathcal{X}$ is possibly infinite, and that the f has additional structure given by the kernel function, which helps us reason about points we haven't seen.

[10] This equivalence holds since the mean and the mode coincide in regression, but is not true in general.

- We will consider two possible objectives:
  - Active learning: we care about the value of the function f everywhere.
  - Bayesian optimization: we only care about finding the maximum of the function f.
- Both objectives can be captured by an **acquisition function** $A(x; x_{1:i-1}, y_{1:i-1})$, which represents the expected utility gained from querying x.
- The general algorithm iteratively query points at maxima of the acquisition function:
- **Algorithm 8 (General algorithm)**
  - For $i = 1, \ldots, n$:
    - Query $x_i = \arg\max_{x \in \mathcal{X}} A(x; x_{1:i-1}, y_{1:i-1})$.
    - Observe $y_i = f(x_i)$.
- Active learning
  - Define the acquisition function at x to be the variance of the function at x given the previous points:

$$A(x; x_{1:i-1}, y_{1:i-1}) = \text{var}[f(x) \mid x_{1:i-1}, y_{1:i-1}]. \tag{156}$$

  - The idea is that we want to find the x such that observing $f(x)$ results in the most reduction in uncertainty. Intuitively, this corresponds to learning the most information.
  - Note that after observing $f(x)$, the variance of highly correlated points will also be reduced, so the next iteration, we will try to seek a point which is quite different (exploration).
- Bayesian optimization
  - Define the acquisition function at x to be the **expected improvement** in the maximum function value from observing x.

$$A(x; x_{1:i-1}, y_{1:i-1}) = \mathbb{E}[\max\{0, f(x) - \max_{j<i} y_j\} \mid x_{1:i-1}, y_{1:i-1}]. \tag{157}$$

  - In optimization, we want to only learn about $f(x)$ if it has a good chance of improving the current maximum $\max_{j<i} y_j$.
  - If $f(x)$ has high variance, but low enough expectation, then there's no reason to observe $f(x)$.
  - If $f(x)$ and $f(x')$ have equal function expectation but $f(x)$ has much higher variance, then x preferable to x'. Note that if $f(x)$ turns out to be low, the current maximum is still maintained.
  - Another popular acquisition function for optimization is GP-UCB (upper confidence bound), taken from the multi-armed bandit literature, where we maximize the sum of the expected value plus the standard deviation to encourage exploration.

- Maximizing the acquisition function A over $x \in \mathcal{X}$
  - Computing $\arg\max_{x \in \mathcal{X}} A(x; x_{1:i-1}, y_{1:i-1})$ is non-trivial since A is not convex and x could even be discrete.
  - We therefore resort to sampling or other derivative-free methods for optimizing nasty functions.
  - So Algorithm 8 really only make sense when the actual function evaluations $f(x)$ are *much more expensive* than maximizing the acquisition function or solving the GP with the resulting observations.

3.6 Fourier properties of shift-invariant kernels

- Having explored how kernels can be used in practice, let us turn back to studying their properties. 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 11 (shift-invariant kernel)**
  - A kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ where $\mathcal{X} \subset \mathbb{R}^b$ 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 ($(x + a)(x' + a) \neq xx'$)
- Our goal is to understand shift-invariant kernels in the frequency domain.
- First, let's warmup with some basic facts:
  - $e^{i\omega t} = \cos(\omega t) + i \sin(\omega t)$
  - $e^{-i\omega t} = \cos(\omega t) - i \sin(\omega t)$
  - $\cos(\omega t) = \frac{1}{2} e^{i\omega t} + \frac{1}{2} e^{-i\omega t}$
  - 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 feature map based on the Fourier basis:
  - Define a feature map $\phi(x) = e^{-i\langle \omega, x \rangle}$.
  - Using this feature map, we could define a kernel:

$$k(x, x') = \phi(x)\overline{\phi(x')} = e^{-i\langle \omega, x - x' \rangle}, \tag{158}$$

    where $\bar{a}$ denotes the complex conjugate of a. This kernels deems two points to be similar if they are close modulo $2\pi/\omega$ (for scalar ω).

- To incorporate multiple frequencies, let $\mu(\cdot)$ be a finite non-negative measure over frequencies. Then

$$k(x, x') = \int e^{-i\langle \omega, x - x' \rangle} \mu(d\omega). \tag{159}$$

  is also a valid kernel.
- 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 (single frequency)
  - Suppose μ places mass only at $-\omega$ and ω:

$$\mu = \frac{1}{2}(\delta_{-\omega} + \delta_{\omega}). \tag{160}$$

  - Then the resulting kernel is:

$$h(t) = \cos(\omega t). \tag{161}$$

- Symmetry implies real
  - Note that $k(x, x')$ will be in general complex-valued.
  - However, if we assume that μ is symmetric (that is, $\mu(A) = \mu(-A)$ for all measurable sets A), then k will be real.

- 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 (159) for some appropriate choice of μ. This statement is precisely Bochner's theorem:

- **Theorem 9 (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 finite **non-negative** measure μ on $\mathbb{R}^b$ such that

$$\boxed{h(t) = \int e^{-i\langle t, \omega \rangle} \mu(d\omega).} \tag{162}$$

  - Furthermore, if μ has a density s,

$$\mu(d\omega) = s(\omega) d\omega, \tag{163}$$

  then we call s the **spectral density**, and h is the Fourier transform of s.

- Example (constant)
  - Let the spectral measure $\mu = \delta_0$ place all its mass at 0.
  - Then $h(t) = 1$ is the constant kernel.
- Example (sinc)
  - Let the spectral density s be the uniform distribution from $[-1, 1]$ (that is, we only keep low frequencies):

$$s(\omega) = \mathbb{I}[-a \leq \omega \leq a]. \tag{164}$$

  - Then the resulting kernel is:

$$h(t) = \frac{2\sin(at)}{t}. \tag{165}$$

  - Proof: just integrate:

$$h(t) = \int_{-a}^{a} e^{-i\omega t} d\omega = \frac{1}{-it}(e^{-iat} - e^{iat}) = \frac{1}{-it}(-2i\sin(at)). \tag{166}$$

  - Note: as $a \to \infty$, we cover more frequencies, and $h(t)$ converges to a delta function at 0, which corresponds to the degenerate kernel $k(x, x) = \mathbb{I}[x = x']$.
- Given a candidate kernel function $k(x, x') = h(x - x')$, we can take the inverse Fourier transform of h (which for symmetric functions is the Fourier transform times $1/(2\pi)$) to get the spectral density s.
- Non-example (box):
  - Consider the following candidate kernel:

$$h(t) = \mathbb{I}[-1 \leq t \leq 1]. \tag{167}$$

  - However, the inverse Fourier transform is

$$s(\omega) = \frac{\sin(\omega)}{\pi\omega}, \tag{168}$$

    reusing the result from above.
  - But notice that $s(\omega)$ is negative in some places, which means that $h(t)$ is not a valid (positive semidefinite) kernel.
- Example (Gaussian kernel):

- Let the spectral density s be the multivariate Gaussian distribution with variance $1/\sigma^2$ (note this is inverted):

$$s(\omega) = \left(\frac{2\pi}{\sigma^2}\right)^{-d/2} \exp\left(\frac{-\sigma^2\|\omega\|^2}{2}\right). \tag{169}$$

- Then the resulting kernel is the Gaussian distribution with variance σ^2:

$$h(t) = \exp\left(\frac{-\|t\|_2^2}{2\sigma^2}\right). \tag{170}$$

- Proof:

$$h(t) = \int \left(\frac{2\pi}{\sigma^2}\right)^{-d/2} \exp\left(\frac{(-\sigma^2\|\omega\|^2 - 2i\langle\omega, t\rangle - \sigma^{-2}i^2\|t\|^2) + \sigma^{-2}i^2\|t\|^2}{2}\right) d\omega. \tag{171}$$

  Complete the square and note that the Gaussian distribution (with mean it/σ) integrates to 1.

- The larger σ^2 is, the smoother the kernel, and more quickly the high frequency components are dampened.

- Example (rational quadratic kernel):
  - Motivation: with the 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

$$\int k_\tau(x, x')p(\tau)d\tau \tag{172}$$

    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**:

$$h(t) = \left(1 + \frac{\beta t^2}{2\alpha}\right)^{-\alpha}. \tag{173}$$

  - 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 $\alpha \to \infty$, the rational quadratic approaches the Gaussian kernel.

- Universality

  - 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 12 (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 $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ 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).
  - 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 10 (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.

3.7 RKHS embedding of probability distributions

- 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 probability distributions.

- 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.
- 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:

$$\boxed{D(P, Q, \mathcal{F}) \stackrel{\text{def}}{=} \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim P}[f(x)] - \mathbb{E}_{x \sim Q}[f(x)] \right),} \tag{174}$$

  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

$$D(P, Q, \mathcal{F}) = 0 \Leftrightarrow P = Q? \tag{175}$$

  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 11 (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:

[begin lecture 2/11] (2/11)

- **Theorem 12 (Steinwart, 2001)**
  - Let $\mathcal{F} = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq 1\}$, 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., $\mathcal{C}_0(\mathcal{X})$ is dense in $\mathcal{X}$ with respect to the uniform norm), there exists $g \in \mathcal{H}$ with $g \neq 0$ such that

$$\|f - g\|_\infty \stackrel{\text{def}}{=} \max_{x \in \mathcal{X}} |f(x) - g(x)| \leq \epsilon/3. \tag{176}$$

- This means $|\mathbb{E}_P[f] - \mathbb{E}_P[g]| \leq \epsilon/3$ and $|\mathbb{E}_Q[f] - \mathbb{E}_Q[g]| \leq \epsilon/3$.
- By the triangle inequality, $|\mathbb{E}_P[g] - \mathbb{E}_Q[g]| \geq \epsilon/3 > 0$.
- Rescale g: let $u = g/\|g\|_{\mathcal{H}} \in \mathcal{F}$.
- We still have $D(P, Q, \mathcal{F}) \geq |\mathbb{E}_P[u] - \mathbb{E}_Q[u]| > 0$.

- Computing $D(P, Q, \mathcal{F})$
  - 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:

$$\mathbb{E}_{x \sim P}[f(x)] = \mathbb{E}_{x \sim P}[\langle k(\cdot, x), f \rangle] = \left\langle \underbrace{\mathbb{E}_{x \sim P}[k(x, \cdot)]}_{\stackrel{\text{def}}{=} \mu_P}, f \right\rangle. \tag{177}$$

    Here, $\mu_P \in \mathcal{H}$ is the **RKHS embedding** of the probability distribution P.
  - We can now write the MMD solution as follows:

$$D(P, Q, \mathcal{F}) = \sup_{f \in \mathcal{F}} \langle \mu_P - \mu_Q, f \rangle = \|\mu_P - \mu_Q\|_{\mathcal{H}}, \tag{178}$$

    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:

$$\|\mu_P - \mu_Q\|_{\mathcal{H}}^2 = \mathbb{E}_{P \times P}[k(x, x')] - \mathbb{E}_{P \times Q}[k(x, y)] - \mathbb{E}_{Q \times P}[k(y, x)] + \mathbb{E}_{Q \times Q}[k(y, y')]. \tag{179}$$

- Of course, in practice, we only have samples from P, Q: let $x_1, \ldots, x_n \sim P$ and $y_1, \ldots, y_n \sim Q$ all be drawn independently.
- We can obtain an empirical estimate of $D(P, Q, \mathcal{F})$ as a U-statistic (average over all pairs of points):

$$\hat{D}_n(P, Q, \mathcal{F}) = \frac{1}{\binom{n}{2}} \sum_{i<j} [k(x_i, x_j) - k(x_i, y_j) - k(y_i, x_j) + k(y_i, y_j)]. \tag{180}$$

  This estimate is unbiased, since the expectation of each term is $D(P, Q, \mathcal{F})$.

- But in order to use this 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 distributions such as this.

3.8 Summary

- We began by noting that some algorithms (e.g., online 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 semi-definite 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).
- **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.
- 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.
- **Gaussian processes** allow us to work with functions over an RKHS but have uncertainty estimates over function values. Uncertainty estimates are critical for active learning and Bayesian optimization.

3.9 References

- Hofmann/Scholkopf/Smola, 2008: Kernel Methods in Machine Learning
- Rasmussen/Williams, 2006: Gaussian Processes for Machine Learning

4 Uniform convergence

4.1 Motivation

- As a learning algorithm gets increasingly more training data, one would expect its accuracy to improve, and hope that it will converge to the best predictor in a certain class.
- This intuition is quite natural and one can easily observe it in practice. The goal of this section is to formalize this intuition mathematically.
- We also aim to obtain rates of convergence, which allows us to answer: given n training examples, what is the generalization error of the learned predictor?

4.2 Ingredients

- We will study the **empirical risk minimizer**, which corresponds to any algorithm that returns the predictor from a class with the lowest training error. This is the batch learning analog of follow the leader (FTL) from online learning.
- The main object of study is the **generalization error**. We already saw that we could bound the *expected* generalization error of online gradient descent using online-to-batch conversion. This section will allow us to derive high probability bounds as well as more general results for function classes without relying on convexity.
- The generalization error of the empirical risk minimizer can be quite a complicated random variable depending on the training data. Our approach for studying generalization error is based on **uniform convergence**. In the process, we will develop some fairly general probabilistic machinery, which are more broadly applicable outside the learning setting.

4.3 Supervised learning

- In this section, we formalize the supervised learning setting. Much of what we will do is more broadly applicable, 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 $x \in \mathcal{X}$.
- Let $\mathcal{H} \subset (\mathcal{X} \mapsto \mathbb{R})$ be a set of **hypotheses** (i.e. experts), where each $h \in \mathcal{H}$ maps $\mathcal{X}$ to $\mathbb{R}$. Example: $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{H}$ is all linear functions.
- Let $\ell : \mathcal{Y} \times \mathbb{R} \mapsto [0, \infty)$ be a **loss function**. Example: $\mathcal{Y} = \{-1, +1\}$ and $\ell(y, p) = \mathbb{I}[yp \leq 0]$ is the classic zero-one loss.

- Let p^* denote the true underlying data generating distribution over input-output pairs $\mathcal{X} \times \mathcal{Y}$.
- **Definition 13 (generalization error (expected risk))**
  - Let $L(h)$ be the **generalization error** (also known as the expected risk) of a hypothesis $h \in \mathcal{H}$, which is the loss that h incurs on a new test example (x, y) in expectation:

$$\boxed{L(h) \stackrel{\text{def}}{=} \mathbb{E}_{(x,y)\sim p^*}[\ell(y, h(x))].} \tag{181}$$

  - Let h^* be any hypothesis that minimizes the generalization error:

$$h^* \in \arg\min_{h\in\mathcal{H}} L(h). \tag{182}$$

- Suppose we are given n training examples, which are a set of input-output pairs: $D \stackrel{\text{def}}{=} \{(x^{(i)}, y^{(i)})\}_{i=1}^n$.
- **Assumption 2 (i.i.d.)**
  - Assume both training examples and test examples are drawn i.i.d. from the underlying data distribution p^*.
  - Note that this assumption links the training set and test set, and ensures that more examples gives us more information (we don't get stuck with the same training example over and over again).
- **Definition 14 (training error (empirical risk))**
  - Let $\hat{L}(h)$ be the **training error** (also known as empirical risk) of a hypothesis $h \in \mathcal{H}$ as the average loss over the training examples:

$$\boxed{\hat{L}(h) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{(x,y)\in D} \ell(y, h(x)).} \tag{183}$$

  - Define the **empirical risk minimizer** be any hypothesis that minimizes the empirical risk:

$$\hat{h} \in \arg\min_{h\in\mathcal{H}} \hat{L}(h). \tag{184}$$

4.4 Confidence and error

- We want to study the generalization error of the empirical risk miminimizer (ERM) $L(\hat{h})$. This is a random quantity that depends on the training examples.
- In principle, $L(\hat{h})$ could be bad even for large n (for instance, if we saw the same example over and over again), so we can't pin down the value of $L(\hat{h})$ with 100% certainty.
- Fortunately, we can show that bad outcomes are not too likely and aim to be 95% confident that $L(\hat{h})$ is small.
- Let ϵ be an error tolerance (say, 0.01), we want to have confidence at least $1 - \delta$ that that our chosen predictor $\hat{h}$ will have generalization **error** at most the minimum possible error plus ϵ. Formally, the this statement can be written as:

$$\boxed{\mathbb{P}[L(\hat{h}) - L(h^*) \geq \epsilon] \leq \delta.} \tag{185}$$

  Note the similarity with regret in the online setting.
- It is important to note that there are two sources of randomness at play here:
  - Error $L(h)$ (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

4.5 Realizable finite hypothesis classes

- 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 have a finite number of hypotheses, at least one of which has zero generalization error (the outcome will be analogous to the analysis of the majority algorithm in the online learning setting). These assumptions are as formalized as follows:
- **Assumption 3 (finite hypothesis space)**
  - Assume $\mathcal{H}$ is finite.
  - This rules out hypothesis spaces which are too large (e.g., $\mathcal{H}$ is the set of all boolean ($\mathcal{Y} = \{0, 1\}$) functions on $\mathcal{X} = \mathbb{R}$).

- **Assumption 4 (realizable)**
  - Assume there exists a predictor $h^* \in \mathcal{H}$ that obtains zero generalization error, that is:

$$L(h^*) = \mathbb{E}_{(x,y)\sim p^*}[\ell(y, h^*(x))] = 0. \tag{186}$$

  - This makes the problem (and the analysis) easier, and we actually get stronger theoretical guarantees in this easy setting.

- Analysis
  - Assume that the loss function is the zero-one loss: $\ell(y, p) = \mathbb{I}[yp < 0]$.
  - Let $B \subset \mathcal{H}$ be the set of bad predictors h: those for which $L(h) > \epsilon$ (have too much error).
  - After n examples, we say a predictor h has *survived* if it is compatible with all the training examples thus far. Note that h^* will definitely survive, but it's possible that a bad predictor $h \in B$ will also survive.
  - In order to guarantee that $\hat{h}$ have error rate at most ϵ, all the bad predictors $h \in B$ must be eliminated.
  - Let S_h be the event that predictor h survives. The probability that a **fixed** bad $h \in B$ survives after n training examples is

$$\mathbb{P}[S_h] = (1 - L(h))^n \leq (1 - \epsilon)^n. \tag{187}$$

    * Remark: this survival probability **decreases exponentially** with n; the worse an h, the more likely it will die sooner; this dependence n will be important.
  - The probability that $\hat{h}$ is a bad predictor can be upper bounded as follows:

$$\begin{aligned}
\mathbb{P}[L(\hat{h}) > \epsilon] &= \mathbb{P}[\exists h \in B : S_h] && (188)\\
&\leq \sum_{h\in B} \mathbb{P}[S_h] \quad \text{[union bound]} && (189)\\
&\leq |B|(1 - \epsilon)^n && (190)\\
&\leq |\mathcal{H}|(1 - \epsilon)^n && (191)\\
&\leq |\mathcal{H}|e^{-\epsilon n} \quad \text{[since } 1 - a \leq e^{-a}\text{]} && (192)\\
&\stackrel{\text{def}}{=} \delta. && (193)
\end{aligned}$$

  - Let's take logs:

$$\log \delta = \log |\mathcal{H}| - \epsilon n. \tag{194}$$

- Rearranging:

$$\epsilon n = \log |\mathcal{H}| + \log(1/\delta). \tag{195}$$

- To summarize, we have the following theorem:

- **Theorem 13 (realizable finite hypothesis class)**
  - Let $\mathcal{H}$ be a finite hypothesis class with some $h^* \in \mathcal{H}$ attaining zero error.
  - Let $\hat{h}$ be the empirical risk minimizer.
  - Let ℓ be the zero-one loss.
  - Then we have two equivalent statements, each with a different interpretation depending on what we're interested in.
  - Interpretation 1: what is the error after training on n examples (**generalization error**)? Answer: with probability at least $1 - \delta$,

$$\boxed{L(\hat{h}) \leq \epsilon = \frac{\log |\mathcal{H}| + \log(1/\delta)}{n}.} \tag{196}$$

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

$$\underbrace{L(\hat{h})}_{\text{error}} \leq O\left(\frac{\overbrace{\log |\mathcal{H}|}^{\text{complexity}}}{\underbrace{n}_{\text{number of training examples}}}\right) \tag{197}$$

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

$$\boxed{n \geq \frac{\log |\mathcal{H}| + \log(1/\delta)}{\epsilon}.} \tag{198}$$

- Remarks
  - This is a basic form of **uniform convergence**, where we want to make a statement simultaneously for *every* $h \in B$.
  - Statisticians seem to prefer to talk about error, learning theorists (from the computer science community) like to talk about sample complexity.
  - $\epsilon = O(1/n)$: we get a "fast" rate (analogous to $O(1/T)$ average regret) because we've assumed realizability; as soon h makes even a single mistake on a training example, we can throw it away.

- ϵ only grows logarithmically with $|\mathcal{H}|$, so we can use pretty big hypothesis classes.
- Note that we don't make use of $p^*(x, y)$ at all. This is known as a **distribution-free** result.

- **Example 16 (learning conjunctions)**
  - Let $\mathcal{X} = \{0, 1\}^d$ (we have d boolean features)
  - Let $\mathcal{Y} = \{0, 1\}$ (doing binary classification)
  - $\mathcal{H} = \{f_J : J \subset \{1, \ldots, d\}\}$ is the set of conjunctive formulas, where $f_J(x) = \prod_{j \in J} x_j$ is a conjunction over the features in J.
  - Here, $|\mathcal{H}| = 2^d$, but $\log |\mathcal{H}| = O(d)$, so we can learn as long as the number of examples is at least the number of features.
  - Here, we have an efficient algorithm, whereby the set of empirical risk minimizers is represented by all subsets of the features which were 1 when $y = 1$.

- Probably Approximately Correct (PAC) framework [Leslie Valient, 1984]
  - A learning algorithm $\mathcal{A}$ PAC learns $\mathcal{H}$ if for any distribution $p^*(x, y)$, $\epsilon > 0$, $\delta > 0$, $\mathcal{A}$ (which takes as input n training examples along with ϵ and δ), returns $\hat{h} \in \mathcal{H}$ such that with probability at least $1 - \delta$, $L(\hat{h}) \leq \epsilon$, and $\mathcal{A}$ runs in $\text{poly}(\text{size}(x), 1/\epsilon, 1/\delta)$ time.
  - Note that time complexity upper bounds sample complexity.
  - We will not focus so much on the computational aspect in our presentation, but just work with the ERM.

- At an abstract level, there are two steps for deriving generalization bounds:
  - Step 1: Bound the error with a given confidence for a fixed hypothesis $h \in \mathcal{H}$ using probabilistic arguments.
  - Step 2: Apply union bound to bound the error simultaneously for *all* relevant $h \in \mathcal{H}$.

  We will use the above template for proving bounds, even though the techniques for the two steps will become more sophisticated to handle the following two questions:
  - 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**.
  - What happens when the number of hypotheses is infinite? To answer this, we need to have more suitable ways of measuring the "size" of a set other than cardinality. This leads to **VC dimension**, **covering numbers**, and **Rademacher complexity** as ways for measuring the "size" of an infinite set.

4.6 Concentration inequalities

- Mean estimation
  - Let $X, X_1, \ldots, X_n$ be i.i.d. real-valued random variables with mean $\mu \stackrel{\text{def}}{=} \mathbb{E}[X]$ (the X is only for notational convenience).
  - Suppose our goal is to estimate μ from $X_1, \ldots, X_n$.
  - Define the empirical mean as follows:

$$\hat{\mu}_n \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} X_i \tag{199}$$

  - 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* classifier $h \in \mathcal{H}$ on the i-th data point.

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

$$\hat{\mu}_n - \mu \xrightarrow{P} 0, \tag{200}$$

    where $\xrightarrow{P}$ denotes convergence in probability.[11] 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**: by the central limit theorem,

$$\frac{\sqrt{n}(\hat{\mu}_n - \mu)}{\sigma} \xrightarrow{d} \mathcal{N}(0, 1), \tag{201}$$

    where $\xrightarrow{d}$ denotes convergence in distribution.[12] Asymptotic normality says that if n is large enough, then $\hat{\mu}_n - \mu$ behaves like $\mathcal{N}(0, \frac{\sigma^2}{n})$, 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, $n = 10$).
  - **Tail bounds**: ideally, we want a statement of the following form:

$$\mathbb{P}[|\hat{\mu}_n - \mu| \geq \epsilon] \leq \text{SomeFunction}(n, \epsilon) = \delta. \tag{202}$$

    Based on the Gaussian approximation, we expect that the bounding function on the RHS would decay double exponentially. We shall see shortly that this intuition is indeed true. Here, ϵ plays the role of the generalization error, and $1 - \delta$ plays the role of the confidence.

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

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

[begin lecture 2/13] (2/13)

- Our starting point is Markov's inequality, a very simple tool that allows us to control the deviation of a 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 14 (Markov's inequality)**
  - Let $Z \geq 0$ be a random variable.
  - Then

$$\mathbb{P}[Z \geq t] \leq \frac{\mathbb{E}[Z]}{t}. \tag{203}$$

- 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**:

$$\mathbb{P}[|X - \mu| \geq \epsilon] \leq \frac{\text{var}(X)}{\epsilon^2}. \tag{204}$$

    Applying the inequality to the average over i.i.d. variables ($\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^n X_i$), then $\text{var}(\hat{\mu}_n) = \frac{\text{var}(X_1)}{n}$. This is a weak concentration result, because the tail probability is decaying only at $1/n$.
  - 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.

But first, an important definition:

- **Definition 15 (moment generating function)**
  - For a random variable X, the moment generating function (MGF) of X is:

$$\boxed{M_X(t) \stackrel{\text{def}}{=} \mathbb{E}[e^{tX}].} \tag{205}$$

- One important property is that the MGF of a sum of independent random variables is simply the product of the MGFs

$$M_{X_1+X_2}(t) = M_{X_1}(t)M_{X_2}(t). \tag{206}$$

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

$$\mathbb{P}[X \geq \epsilon] \leq \frac{M_X(t)}{e^{t\epsilon}} \text{ for all } t \geq 0. \tag{207}$$

Applying the same inequality to $\mathbb{P}[X_1 + \cdots + X_n \geq n\epsilon]$, we see that for all $t \geq 0$:

$$\mathbb{P}[\hat{\mu}_n \geq \epsilon] \leq \left(\frac{M_X(t)}{e^{t\epsilon}}\right)^n. \tag{208}$$

- Provided that $\frac{M_X(t)}{e^{t\epsilon}} < 1$, getting n independent samples means that our tail probability will decrease exponentially.
- Note that $M_X(t)$ could be infinite for some t, in which case the bounds are vacuous. We will work with distributions X such that $M_X(t) < \infty$.

- Now let's actually compute the MGF of some probability distributions of interest.
- MGF of Gaussians
  - The Gaussian is a natural candidate since the sum of independent random variables converges to a Gaussian.
  - Let $X \sim \mathcal{N}(0, \sigma^2)$.
  - Then $M_X(t) = e^{\sigma^2 t^2/2}$.
  - Derivation (by completing the square):

$$M_X(t) = \mathbb{E}[e^{tX}] = \int (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{(\frac{x}{\sigma})^2 - 2tx + \sigma^2 t^2 - \sigma^2 t^2}{2}\right) dx = e^{\sigma^2 t^2/2}. \tag{209}$$

- Tail bound for Gaussians (and sub-Gaussians, as we'll see soon)
  - Having control on the Gaussian MGF, we can now derive a tail bound by plugging the form of the MGF into (207). This yields:

$$\mathbb{P}[X \geq \epsilon] \leq \inf_t \exp\left(\frac{\sigma^2 t^2}{2} - t\epsilon\right). \tag{210}$$

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

$$\boxed{\mathbb{P}[X \geq \epsilon] \leq \exp\left(\frac{-\epsilon^2}{2\sigma^2}\right).} \tag{211}$$

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

$$\boxed{M_X(t) \leq \exp\left(\frac{\sigma^2 t^2}{2}\right).} \tag{212}$$

- Examples
  - Gaussian random variables: If $X \sim \mathcal{N}(0, \sigma^2)$, then X is sub-Gaussian parameter σ (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 proof): the variance of X is maximized when all the mass is at a and b, in which case the variance is $-ab \leq \frac{(b-a)^2}{4}$ (equality when $a = -b$).
- 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: 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.
- Given our setup, we can easily obtain the following classic tail bound for bounded random variables:
- **Theorem 15 (Hoeffding's inequality)**
  - Let $X, X_1, \ldots, X_n$ 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

$$\mathbb{P}[\hat{\mu}_n \geq \mathbb{E}[\hat{\mu}_n] + \epsilon] \leq \exp\left(\frac{-2n^2\epsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right). \tag{213}$$

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

$$\mathbb{P}[\hat{\mu}_n \geq \mathbb{E}[\hat{\mu}_n] + \epsilon] \leq \exp\left(\frac{-n\epsilon^2}{2B^2}\right). \tag{214}$$

- Proof:
  - 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 (211).
- Summary so far
  - We've shown that Gaussian and bounded random variables are sub-Gaussian and enjoy exponential tail bounds.
  - Furthermore, if we have an average of n i.i.d. sub-Gaussian variables, then the bound simply gets powered up by n, which results in sharp concentration.
- Next, we will show a generalization of Hoeffding's inequality, where we want to bound not the average of $X_1, \ldots, X_n$, but any function on $X_1, \ldots, X_n$ satisfying an appropriate bounded differences condition. After all, learning algorithms do more complex things than taking averages.
- **Theorem 16 (McDiarmid's inequality (bounded differences inequality))**
  - Let $X_1, \ldots, X_n$ be independent random variables.
  - Let f be a function satisfying the following bounded differences condition:

$$|f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)| \leq c_i \text{ for all } i = 1, \ldots, n. \tag{215}$$

    Intuition: modifying one coordinate doesn't change the function value too much.
  - Then

$$\mathbb{P}[f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)] \geq \epsilon] \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^n c_i^2}\right). \tag{216}$$

- Remark
  - Note that the form of this bound coincides exactly with Hoeffding's inequality, where $c_i = \frac{1}{n}(b_i - a_i)$.

4.7 Generalization bounds

- Recall that our goal is to obtain an tail bound on the regret, the amount by which ERM's generalization error exceeds the optimum:

$$\mathbb{P}[L(\hat{h}) - L(h^*) \geq \epsilon] \leq \delta. \tag{217}$$

- In this section, we will show how to obtain this objective by reducing the problem to one of uniform convergence.

- In particular, let us relate generalization errors to training errors:

$$L(\hat{h}) - L(h^*) = \underbrace{[L(\hat{h}) - \hat{L}(\hat{h})]}_{\text{concentration}} + \underbrace{[\hat{L}(\hat{h}) - \hat{L}(h^*)]}_{\leq 0} + \underbrace{[\hat{L}(h^*) - L(h^*)]}_{\text{concentration}}. \tag{218}$$

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

- The first and third term can tackled by concentration inequalities and uniform convergence. Specifically,

$$\mathbb{P}[L(\hat{h}) - L(h^*) \geq \epsilon] \leq \mathbb{P}\left[\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h)| \geq \frac{\epsilon}{2}\right]. \tag{219}$$

  So it suffices to the bound the RHS, which is what uniform convergence is all about: convergence for all $h \in \mathcal{H}$. Bounding this quantity is the main technical challenge.

- Note: $\{L(h) - \hat{L}(h)\}_{h \in \mathcal{H}}$ is called an **empirical process**, which is a stochastic process (collection of random variables indexed by $h \in \mathcal{H}$). Empirical process theory focuses on studying empirical processes, especially their suprema.

- Some basic but useful facts
  - If event A implies event B, then $\mathbb{P}[A] \leq \mathbb{P}[B]$.
  - If random variables X, Y satisfy $X \leq Y$, then $\mathbb{P}[X \geq \epsilon] \leq \mathbb{P}[Y \geq \epsilon]$.
  - If random variables X, Y satisfy $X, Y \geq 0$, then $\mathbb{P}[X+Y \geq \epsilon] \leq \mathbb{P}[X \geq \frac{\epsilon}{2}] + \mathbb{P}[Y \geq \frac{\epsilon}{2}]$.

[begin lecture 2/20] (2/20)

4.8 Finite hypothesis classes

- For a fixed $h \in \mathcal{H}$, $\hat{L}(h)$ converges to $L(h)$ in probability and we can use our newly developed concentration bounds to bound this quantity.

- Assume zero one-loss $\ell(y, h(x)) = \mathbb{I}[yh(x) \leq 0]$ (actually all we need is that the loss function is bounded).

- Then applying Hoeffding's inequality, noting that $\hat{L}(h)$ is an empirical average over n i.i.d. loss terms with expectation $L(h)$, we have:

$$\mathbb{P}[\hat{L}(h) - L(h) \geq \epsilon] \leq \exp\left(-2n\epsilon^2\right). \tag{220}$$

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

$$\mathbb{P}[|\hat{L}(h) - L(h)| \geq \epsilon] \leq 2\exp\left(-2n\epsilon^2\right). \tag{221}$$

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

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |\hat{L}(h) - L(h)| \geq \epsilon\right] \leq |\mathcal{H}| \cdot 2\exp\left(-2n\epsilon^2\right) \stackrel{\text{def}}{=} \delta. \tag{222}$$

- Recall that for the generalization bound, we need error tolerance $\frac{\epsilon}{2}$.

- Let us rearrange the expression above to obtain the following result: with probability at least $1-\delta$,

$$L(\hat{h}) - L(h^*) \leq \sqrt{\frac{2(\log|\mathcal{H}| + \log(2/\delta))}{n}} = O\left(\frac{1}{\sqrt{n}}\right). \tag{223}$$

  Note that if we grow our hypothesis class $\mathcal{H}$, then we reduce approximation error $L(h^*)$ but increase the estimation error (regret).

- Let's compare with (196), which is the generalization bound we got in the realizable case.

  - Many of the quantities are the same: 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 get a $\frac{1}{\sqrt{n}}$ rate.
  - This rate difference should be reminiscent of the situation in online learning, where we got $O(\frac{\log T}{T})$ and $O(\frac{1}{\sqrt{T}})$ average regret with and without a perfect expert, respectively. Of course the setting is different: we have i.i.d. assumptions on the data but no assumptions on convexity.
  - Even though these are upper bounds, these rates are essentially tight (though we won't show this).

4.9 Infinite hypothesis classes

- Motivation

  - With infinite $|\mathcal{H}|$, the union bound gives us vacuous results. And actually in general, if we know nothing else about $\mathcal{H}$, this is a hopeless problem.

- But suppose $\mathcal{H}$ were binary linear classifiers, where each $h \in \mathcal{H}$ maps $\mathcal{X}$ to $\{1, -1\}$.
- In this case, we can exploit the geometry of $\mathcal{H}$ a bit more.
- Intuitively, two classifiers that are really close have the same behavior on n training points ($\hat{L}(h)$).
- But the challenge is that $L(h)$ involves an expectation over all points.
- We can eliminate the dependence on $L(h)$ via a clever trick called **symmetrization**, which allows us to work with data points.
- Then by conditioning on these points, we effectively reduce the set of hypotheses to a finite set, on which we can apply the standard union bound.

- Notation
  - We use $Z_1, \ldots, Z_n$ to denote the data points ($Z_i = (x^{(i)}, y^{(i)})$).
  - Abusing notation, we write the loss function as

$$\ell(Z_i, h) = \ell(y^{(i)}, h(x^{(i)})). \tag{224}$$

- Step 1: symmetrization
  - **Lemma 3 (symmetrization)**
    - Let D' be a set of n independent i.i.d. data points (called a ghost sample) and define the corresponding empirical risk on that sample:

$$\hat{L}'(h) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell(Z_i', h). \tag{225}$$

    - If $n\epsilon^2 \geq 2$, we have

$$\mathbb{P}\left[\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h)| \geq \epsilon\right] \leq 2\,\mathbb{P}\left[\sup_{h \in \mathcal{H}} |\hat{L}(h) - \hat{L}'(h)| \geq \frac{\epsilon}{2}\right]. \tag{226}$$

    - Remarks
      - The condition $n\epsilon^2 \geq 2$ is a mild technical condition.
      - We only lose a factor of 2 in $1 -$ confidence if we compare the difference between two independent empirical risks to half the error tolerance.
  - Proof
    - Let h be the maximizer of $L(h) - \hat{L}(h)$.
    - We have (by picture):

$$\left(|L(h) - \hat{L}(h)| \geq \epsilon \text{ and } |L(h) - \hat{L}'(h)| \leq \frac{\epsilon}{2}\right) \text{ implies } |\hat{L}(h) - \hat{L}'(h)| \geq \frac{\epsilon}{2}. \tag{227}$$

* The two factors on the left side of the implication are independent, so we have:

$$\mathbb{P}\left(|L(h) - \hat{L}(h)| \geq \epsilon\right) \mathbb{P}\left(|L(h) - \hat{L}'(h)| \leq \frac{\epsilon}{2}\right) \leq \mathbb{P}\left(|\hat{L}(h) - \hat{L}'(h)| \geq \frac{\epsilon}{2}\right). \tag{228}$$

* We can bound one minus the second factor using Chebyshev's inequality:

$$\mathbb{P}\left(|L(h) - \hat{L}'(h)| \geq \frac{\epsilon}{2}\right) \leq \frac{4\text{var}(\hat{L}'(h))}{\epsilon^2} \leq \frac{1}{n\epsilon^2} \leq \frac{1}{2}, \tag{229}$$

where we are using the fact that the loss is bounded in $[0, 1]$ (so the variance of a single term in the loss is at most $1/4$), and the variance falls off as $1/n$. The final inequality is due to the assumption that $n\epsilon^2 \geq 2$.

* Plugging the bound for the second factor and moving to the RHS:

$$\mathbb{P}\left(|L(h) - \hat{L}(h)| \geq \epsilon\right) \leq 2\mathbb{P}\left(|\hat{L}(h) - \hat{L}'(h)| \geq \frac{\epsilon}{2}\right). \tag{230}$$

* We can put sup on LHS by definition of h, and put sup on the RHS because the value of the sup is at least as large as the value of h:

$$\mathbb{P}\left(\sup_{h\in\mathcal{H}} |L(h) - \hat{L}(h)| \geq \epsilon\right) \leq 2\mathbb{P}\left(\sup_{h\in\mathcal{H}} |\hat{L}(h) - \hat{L}'(h)| \geq \frac{\epsilon}{2}\right). \tag{231}$$

* This completes the proof.

- Step 2: introduce random signs (**Rademacher variables**)
  - Define n random variables $\sigma_1, \ldots, \sigma_n$ (called Rademacher variables), where each σ_i is uniform on $\{-1, +1\}$. Think of these as random binary labels on the data points.
  - We can use these random signs to remove one of the random samples:

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}} |\hat{L}(h) - \hat{L}'(h)| \geq \frac{\epsilon}{2}\right] = \mathbb{P}\left[\sup_{h\in\mathcal{H}} \left|\frac{1}{n}\sum_{i=1}^{n} \sigma_i[\ell(Z_i, h) - \ell(Z_i', h)]\right| \geq \frac{\epsilon}{2}\right] \tag{232}$$

$$\leq 2\,\mathbb{P}\left[\sup_{h\in\mathcal{H}} \left|\frac{1}{n}\sum_{i=1}^{n} \sigma_i \ell(Z_i, h)\right| \geq \frac{\epsilon}{4}\right]. \tag{233}$$

* First equality: we can inject σ_i by symmetry (Z_i and Z_i' come from the same distribution).
* Second inequality: by triangle inequality ($|a + b| \leq |a| + |b|$), swapping max and sum ($\max_i a_i + b_i \leq \max_i a_i + \max_i b_i$), ($a + b \geq \frac{\epsilon}{2}$ implies $a \geq \frac{\epsilon}{4}$ or $b \geq \frac{\epsilon}{4}$), and the union bound.

- Step 3: condition and apply Hoeffding's inequality
  - Condition on $Z_1, \ldots, Z_n$.
  - Even though there is sup over an infinite $\mathcal{H}$, multiple $h \in \mathcal{H}$ might have the same behavior on $Z_1, \ldots, Z_n$. In fact, here are all the possible "behaviors" (summarized as a vector of losses on the n points):

$$\{[\ell(Z_1, h), \ldots, \ell(Z_n, h)] : h \in \mathcal{H}\}. \tag{234}$$

  - To compress notation, we compose the loss ℓ with each hypothesis $h \in \mathcal{H}$, defining the **loss class**, which contains functions that map a point z to the loss of some hypothesis h on z:

$$\mathcal{A} \stackrel{\text{def}}{=} \{z \mapsto \ell(z, h) : h \in \mathcal{H}\}. \tag{235}$$

  - Assume that the loss function can only return two possible values: $\ell(z, h) \in \{0, 1\}$. For example, the zero-one loss satisfies this criterion. Then $\mathcal{A}$ consist of functions with binary outputs. We define the shattering coefficient to capture the number of possible binary labelings on n points defined by $\mathcal{A}$:
  - **Definition 17 (shattering coefficient (growth function))**
    - Let $\mathcal{A}$ be a family of functions with binary outputs over some set $\mathcal{Z}$ (for example, the loss class).
    - The **shattering coefficient** of $\mathcal{A}$ is the maximum number of binary labelings that can be defined over n points:

$$s(\mathcal{A}, n) \stackrel{\text{def}}{=} \max_{z_1, \ldots, z_n \in \mathcal{Z}} |\{[A(z_1), \ldots, A(z_n)] : A \in \mathcal{A}\}|. \tag{236}$$

    - If $s(\mathcal{A}, n) = 2^n$ (we obtain all possible labelings), then we say $\mathcal{A}$ **shatters** the n points that achieves the maximum.
  - We can apply Hoeffding's inequality to terms $\sigma_i \ell(Z_i, h) \in [-1, 1]$, and along with the union bound, we get:

$$\mathbb{P}\left[\sup_{h \in \mathcal{H}} \left| \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(Z_i, h) \right| \geq \frac{\epsilon}{4}\right] \leq s(\mathcal{A}, n) \cdot 2 \exp\left(\frac{-n\epsilon^2}{32}\right), \tag{237}$$

    where $s(\mathcal{A}, n)$ is the shattering coefficient of $\mathcal{A}$.
  - Note that the bound doesn't depend on $Z_1, \ldots, Z_n$ (it holds pointwise), so we just took expectations over $Z_1, \ldots, Z_n$ without affecting the RHS.
- Step 4: synthesis

– Putting everything together, we have:

$$\mathbb{P}\left[\sup_{h\in\mathcal{H}}|L(h)-\hat{L}(h)|\geq\epsilon\right]\leq 8\,s(\mathcal{A},n)\exp\left(\frac{-n\epsilon^2}{32}\right). \tag{238}$$

– Applying (219) (plugging in $\frac{\epsilon}{2}$ rather than ϵ), we can get a generalization bound: with probability at least $1-\delta$,

$$\boxed{L(\hat{h})-L(h^*)\leq\sqrt{\frac{128(\log s(\mathcal{A},n)+\log(8/\delta))}{n}}.} \tag{239}$$

Note that we've replaced $|\mathcal{H}|$ as our measure of complexity with the shattering coefficient $s(\mathcal{A},n)$ of the loss class $\mathcal{A}$. The intuition is that all that matters for measuring the size of a hypotheses class is how its loss acts on data points.

– Next, we will try to get a handle on $s(\mathcal{A},n)$ using VC dimension.

4.10 VC dimension

- In the previous section, we saw that we can get a generalization bound (239) that depends on the shattering coefficient as a measure of complexity of a hypothesis class $\mathcal{H}$. Although the shattering coefficient nicely captures the action of an infinite $\mathcal{H}$, it is not the most convenient quantity to get a handle on for various classes of $\mathcal{H}$. In this section, we will use a concept called VC dimension to quantify the shattering coefficient.

- VC dimension only applies to boolean loss functions (which define sets). For real-valued functions, we need to use covering numbers or Rademacher complexity, which we will discuss later.

- Hypotheses and losses

  – The bounds we derive (239) depend on the shattering coefficients $s(\mathcal{A},n)$ of the loss class.

  – However, it will be more convenient to work with the hypothesis class $\mathcal{H}$.

  – Let's specialize to binary classifiers $h:\mathcal{X}\mapsto\{0,1\}$ and zero-one loss $\ell((x,y),h)=\mathbb{I}[y\neq h(x)]$.

  – Note that the hypothesis class $\mathcal{H}$ contains functions on $\mathcal{X}$, and the loss class $\mathcal{A}$ contains functions on $\mathcal{X}\times\{0,1\}$.

  – The key point is that

$$\boxed{s(\mathcal{H},n)=s(\mathcal{A},n),} \tag{240}$$

since for all $(x_1, y_1), \ldots, (x_n, y_n)$, there is a bijection between the behavior of the losses and the hypotheses:

$$[\ell((x_1, y_1), h), \ldots, \ell((x_n, y_n), h)] \quad \Leftrightarrow \quad [h(x_1), \ldots, h(x_n)]. \tag{241}$$

Interpretation: $\mathbb{I}[y_i \neq h(x_i)]$ represents the same information as $h(x_i)$, where the translation is given by y_i.

- **Definition 18 (VC dimension)**
  - The **VC dimension** of a family of functions $\mathcal{A}$ with binary outputs is the maximum number of points that can be shattered by $\mathcal{A}$:

$$\mathrm{VC}(\mathcal{A}) = \sup\{n : s(\mathcal{A}, n) = 2^n\}. \tag{242}$$

  - Intuition: the VC dimension of $\mathcal{A}$ is the maximum number of points whose binary labels can be memorized perfectly.

- Example (intervals)
  - Let $\mathcal{A} = \{[a, b] : a, b \in \mathbb{R}\}$.
  - $s(\mathcal{A}, 1) = 2 = 2^1$ (can shatter)
  - $s(\mathcal{A}, 2) = 4 = 2^2$ (can shatter)
  - $s(\mathcal{A}, 3) = 7 < 2^3$ (can't shatter, because can't isolate the middle point)
  - $s(\mathcal{A}, n) = \binom{n+1}{2} + 1$
  - Therefore, $\mathrm{VC}(\mathcal{A}) = 2$.

- 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 $\mathcal{A} = \{\{x : \sin(\theta x) \geq 0\} : \theta \in \mathbb{R}\}$.
  - We have that $\mathrm{VC}(\mathcal{A}) = \infty$.

[begin lecture 2/25] (2/25)

- It turns out that it's not the number of parameters that matter, but the dimension of the function space:

- **Theorem 17 (finite-dimensional function class)**

- Let $\mathcal{F}$ be a function class containing functions $f : \mathcal{X} \mapsto \mathbb{R}$.
- Recall the dimension of $\mathcal{F}$ to be the size of a basis of $\mathcal{F}$.
- Let $\mathcal{A} = \{\{x : f(x) \geq 0\} : f \in \mathcal{F}\}$ (for example, $\mathcal{F}$ could be linear functions, but not necessarily).
- Then we have

$$\boxed{\text{VC}(\mathcal{A}) \leq \dim(\mathcal{F}).} \tag{243}$$

- Remark: this allows us to connect the linear algebraic properties of $\mathcal{F}$ (dimension) with the combinatorial properties of $\mathcal{A}$ (VC dimension).

- Proof
  - Take any n points $x_1, \ldots, x_n$ with $n > \dim(\mathcal{F})$. We will show that these n points cannot be shattered.
  - Consider the linear map $H(f) \stackrel{\text{def}}{=} (f(x_1), \ldots, f(x_n)) \in \mathbb{R}^n$ which maps each function $f \in \mathcal{F}$ to the function values on the n points (function evaluation is linear).
  - The vector space $\{H(f) : f \in \mathcal{F}\} \in \mathbb{R}^n$ 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 $H(f) \cdot c = 0$ for all $f \in \mathcal{F}$.
  - Without loss of generality, some component of c is negative (otherwise, just take $-c$, which satisfies $H(f) \cdot c = 0$ too).
  - So we have

$$\sum_{i:c_i \geq 0} c_i f(x_i) + \sum_{i:c_i < 0} c_i f(x_i) = 0. \tag{244}$$

  - For the purposes of contradiction, suppose $\mathcal{A}$ shatters $\{x_1, \ldots, x_n\}$.
    - Then we could find an $A = \{x : f(x) \geq 0\} \in \mathcal{A}$ (equivalently some $f \in \mathcal{F}$) such that
      - $x_i \in A$ $(f(x_i) \geq 0)$ whenever $c_i \geq 0$ (so the first term of (244) is non-negative)
      - $x_i \notin A$ $(f(x_i) < 0)$ whenever $c_i < 0$ (so the second term of (244) is strictly positive).
    - The sum of the two terms couldn't equal zero, which contradicts (244).
  - Therefore, $\mathcal{A}$ can't shatter $\{x_1, \ldots, x_n\}$ for any choice of $x_1, \ldots, x_n$, so $\text{VC}(\mathcal{A}) \leq \dim(\mathcal{F})$.

- Example (half-spaces passing through the origin)

- Let $\mathcal{A} = \{\{x : w \cdot x \geq 0\} : w \in \mathbb{R}^d\}$.
- By Theorem 17, the VC dimension of $\mathcal{A}$ is at most d (this is a linear function class of dimension d).
- It remains to show that the VC dimension is at least d.
- In general, upper bounds suffice for obtaining generalization bounds, but just for fun, let's get a lower bound on the VC dimension.
- In general, showing that the VC dimension of some $\mathcal{A}$ is at least d is easier because we just have to construct some set of d points that can be shattered rather than show there *no* $d+1$ points can be shattered.
- Create d points:

$$x_1 = (1, 0, \ldots, 0, 0) \tag{245}$$
$$\cdots \tag{246}$$
$$x_d = (0, 0, \ldots, 0, 1) \tag{247}$$

- Given any subset $I \subset \{1, \ldots, d\}$ (labeling of $x_1, \ldots, x_d$), 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 $\text{VC}(\mathcal{A}) \geq d$.
- Putting the upper and lower bounds, we get that $\boxed{\text{VC}(\mathcal{A}) = d}$.

- Now let us relate VC dimension to shattering coefficients:
- **Lemma 4 (Sauer's lemma)**
  - For a class $\mathcal{A}$ be a class with VC dimension d.
  - Then

$$\boxed{s(\mathcal{A}, n) \leq \sum_{i=0}^{d} \binom{n}{i} \leq \begin{cases} 2^n & \text{if } n \leq d \\ \left(\frac{en}{d}\right)^d & \text{if } n > d. \end{cases}} \tag{248}$$

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

- The idea is to take $\mathcal{A}$ and transform it into $\mathcal{A}'$ with the same shattering coefficient ($s(\mathcal{A}, n) = s(\mathcal{A}', n)$) but where $s(\mathcal{A}', n)$ will be easier to bound.
- Key: all that matters is the action of $\mathcal{A}$ and $\mathcal{A}'$ 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 $s(\mathcal{A}, n)$ 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 $\mathcal{A}'$):

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 $s(\mathcal{A}, n) = s(\mathcal{A}', n)$. So we just have to compute $s(\mathcal{A}', n)$, which should be easier.
- Step 2: We show that the VC dimension doesn't increase by transformation ($\mathrm{VC}(\mathcal{A}') \leq \mathrm{VC}(\mathcal{A})$)
  - The transformations proceed one column at a time:

$$T \to T_1 \to \cdots T_k \overset{\text{transform column } j}{\to} T_{k+1} \to \cdot \to T'. \tag{249}$$

  - Claim: After transforming any column j, if some subset $S \subset \{1, \ldots, n\}$ 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 $j' \in \{1, \ldots, n\} \backslash \{j\}$, 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 $S \subset \{1,\ldots,n\}$.

∗ Then for each $j \in S$, there must be another row with a labeling that assigns ones to exactly $S \subset \{j\}$ (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, $k \leq d$.

∗ 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 (248).

– Finishing the second part of the inequality of (248) is just algebra. Observe that for $n \geq d$,

$$\sum_{i=0}^{d} \binom{n}{i} \leq \left(\frac{n}{d}\right)^d \sum_{i=0}^{d} \binom{n}{i} \left(\frac{d}{n}\right)^i \tag{250}$$

$$\leq \left(\frac{n}{d}\right)^d \sum_{i=0}^{n} \binom{n}{i} \left(\frac{d}{n}\right)^i \tag{251}$$

$$= \left(\frac{n}{d}\right)^d \left(1 + \frac{d}{n}\right)^n \tag{252}$$

$$\leq \left(\frac{n}{d}\right)^d e^d. \tag{253}$$

- Conclusion

– Recall that $s(\mathcal{A}, n)$ allows us to measure the "size" of an infinite hypothesis class which is finite because symmetrization allows us to condition on the n data points.

– Recall that $s(\mathcal{A}, n)$ is used in the union bound (238). Here the tail probability is upper bounded by $s(\mathcal{A}, n) \cdot c^{-n}$.

– As n grows, concentration will make the second term decay exponentially fast.

– At the same time, $s(\mathcal{A}, n)$ is growing. It grows exponentially while $n < d$ (which means that the error isn't going down), but then grows polynomially, which is much slower than the exponential. The ability of the exponential to kill off the polynomial is what enables generalization.

– This is also seen by plugging in the VC bound into (239):

$$\boxed{L(\hat{h}) - L(h^*) \leq \sqrt{\frac{128(d\log(en/d) + \log(8/\delta))}{n}} = O\left(\sqrt{\frac{d\log n}{n}}\right).} \tag{254}$$

4.11 Rademacher complexity

- Motivation/summary
  - So far, we have gotten uniform convergence tail bounds $\mathbb{P}(\sup_{h\in\mathcal{H}} |L(h) - \hat{h}(h)| \geq \epsilon)$ by using symmetrization to convert the $\sup_{h\in\mathcal{H}}$ into a discrete union, which can be handled by a union bound.
  - The complexity of this union was handled completely using combinatorial means (shattering coefficients and VC dimension), allowing us to deal effectively with infinite hypothesis classes.
  - However, there are two shortcomings of the current approach:
    * VC dimension and shattering works only for loss functions with discrete outputs, but we'd like to analyze the performance for the regression setting where the loss function is **real-valued**.
    * VC dimension is intrinsically tied to the dimension of the space, but we'd like to analyze hypothesis classes which are **infinite-dimensional** (not just infinite), for example, SVMs with the Gaussian kernel.
  - In this section, we will develop a framework for dealing with uniform convergence based on Rademacher complexity which will address the two points above.
  - In the next two parts (concentration and symmetrization), we will bound the maximum difference between empirical and expected risk by the Rademacher complexity.
- Symmetrization
  - Consider the worst difference between the expected and the empirical risk over all possible hypotheses:

$$\boxed{G_n \stackrel{\text{def}}{=} \sup_{h\in\mathcal{H}} L(h) - \hat{L}(h).} \tag{255}$$

    Here, G_n is a random variable that depends on the data points $Z_1, \ldots, Z_n$. This is the quantity that we want to control. (Note that we're starting with the one-sided version, which no absolute values.)
  - In our previous analysis, we bounded $\mathbb{P}(G_n \geq \epsilon)$ directly, working exclusively using tail probabilities.

– We will now pursue a slightly different strategy, in which we try to analyze the expectation $\mathbb{E}[G_n]$. Working with expectations will produce more elegant results and be the launching point for Rademacher complexity.

– Step 1 (concentration): convert tail bound into expectation

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

$$|g(Z_1, \ldots, Z_i, \ldots, Z_n) - g(Z_1, \ldots, Z_i', \ldots, Z_n)| \leq \frac{1}{n}. \tag{256}$$

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

$$\Big| \underbrace{\sup_{h\in\mathcal{H}}[L(h) - \hat{L}(h)]}_{g(Z_1,\ldots,Z_i,\ldots,Z_n)} - \underbrace{\sup_{h\in\mathcal{H}}[L(h) - \hat{L}(h) + \frac{1}{n}(\ell(Z_i,h) - \ell(Z_i',h))]}_{g(Z_1,\ldots,Z_i',\ldots,Z_n)} \Big| \leq \frac{1}{n}. \tag{257}$$

    · 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: $\ell(z, h) \in [0, 1]$.
  * Now we can apply McDiarmid's inequality (Theorem 16) to get that:

$$\mathbb{P}(G_n \geq \mathbb{E}[G_n] + \epsilon) \leq \exp(-2n\epsilon^2). \tag{258}$$

    Think Hoeffding's inequality, generalized to functions which are not just sums of i.i.d. random variables.

[begin lecture 2/27] (2/27)

  * We can rewrite this bound as follows: with probability at least $1 - \delta$,

$$\boxed{G_n \leq \mathbb{E}[G_n] + \sqrt{\frac{\log(1/\delta)}{2n}}.} \tag{259}$$

  * Note that in one fell swoop, we've taken care of the confidence, and it just remains to bound the expectation $\mathbb{E}[G_n]$.

– Step 2 (symmetrization)

  * Introduce a ghost dataset $Z_1', \ldots, Z_n'$, drawn i.i.d. from p^*, and let $\hat{L}'(h) = \frac{1}{n}\sum_{i=1}^n \ell(Z_i', h)$ be the empirical risk with respect to this ghost sample.
  * Rewriting $L(h)$ in terms of the ghost dataset:

$$\mathbb{E}[G_n] = \mathbb{E}[\sup_{h\in\mathcal{H}} \mathbb{E}[\hat{L}'(h)] - \hat{L}(h)]. \tag{260}$$

* Bring $\hat{L}(h)$ into the inner expectation by conditioning on the original dataset $Z_1, \ldots, Z_n$ (the two datasets are independent):

$$\mathbb{E}[G_n] = \mathbb{E}[\sup_{h \in \mathcal{H}} \mathbb{E}[\hat{L}'(h) - \hat{L}(h) \mid Z_1, \ldots, Z_n]]. \tag{261}$$

* Pushing the sup inside the expectation can only increase:

$$\mathbb{E}[G_n] \leq \mathbb{E}[\mathbb{E}[\sup_{h \in \mathcal{H}} \hat{L}'(h) - \hat{L}(h) \mid Z_1, \ldots, Z_n]]. \tag{262}$$

* Apply law of iterated conditional expectation:

$$\mathbb{E}[G_n] \leq \mathbb{E}[\sup_{h \in \mathcal{H}} \hat{L}'(h) - \hat{L}(h)]. \tag{263}$$

* Introduce Rademacher variables (expanding definition of the empirical risk):

$$\mathbb{E}[G_n] \leq \mathbb{E}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i [\ell(Z_i', h) - \ell(Z_i, h)]\right]. \tag{264}$$

* Pusing the sup inside $\sup_h[a_h - b_h] \leq \sup_h[a_h] + \sup_h[-b_h]$:

$$\mathbb{E}[G_n] \leq \mathbb{E}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(Z_i', h) + \sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (-\sigma_i) \ell(Z_i, h)]\right]. \tag{265}$$

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

$$\boxed{\mathbb{E}[G_n] \leq 2\underbrace{\mathbb{E}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell(Z_i, h)\right]}_{\text{Rademacher complexity}}}. \tag{266}$$

- **Definition 19 (Rademacher complexity)**
  - Let $\mathcal{F}$ be a class of real-valued functions $f : \mathcal{Z} \mapsto \mathbb{R}$.
  - Define the **Rademacher complexity** (or Rademacher average) of $\mathcal{F}$ to be

$$\boxed{R_n(\mathcal{F}) \stackrel{\text{def}}{=} \mathbb{E}\left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(Z_i)\right]}, \tag{267}$$

    where

    * $Z_1, \ldots, Z_n$ are drawn i.i.d. from p^* (data points); and

    - $\sigma_1, \ldots, \sigma_n$ are drawn i.i.d. from the uniform distribution over $\{-1, +1\}$ (Rademacher variables).
  - Define the **empirical Rademacher complexity** of $\mathcal{F}$ to be:

$$\boxed{\hat{R}_n(\mathcal{F}) \stackrel{\text{def}}{=} \mathbb{E}\left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(Z_i) \mid Z_1, \ldots, Z_n\right],} \tag{268}$$

    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.
  - Intuitively, the Rademacher complexity captures how well the best function from a function class $\mathcal{F}$ aligns with random signs σ_i.

- Define $\mathcal{A} = \{z \mapsto \ell(z, h) : h \in \mathcal{H}\}$ to be the loss class. By combining (259) and (266), we get that with probability at least $1 - \delta$,

$$\sup_{h \in \mathcal{H}} [L(h) - \hat{L}(h)] \leq 2R_n(\mathcal{A}) + \sqrt{\frac{\log(1/\delta)}{2n}}. \tag{269}$$

- If we apply the above derivation to the negative loss $(-\ell(z, h))$ and apply the union bound, we can get a two-sided bound for $\sup_{h \in \mathcal{H}} |L(h) - \hat{L}(h)|$, which allows us to control the regret $L(\hat{h}) - L(h^*)$. With probability at least $1 - \delta$:

$$L(\hat{h}) - L(h^*) \leq 4R_n(\mathcal{A}) + \sqrt{\frac{2\log(2/\delta)}{n}}. \tag{270}$$

- In the next section, 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**
  - 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
    - $R_n(\mathcal{F}_1) \leq R_n(\mathcal{F}_2)$ if $\mathcal{F}_1 \subset \mathcal{F}_2$
    - 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.
  - Convex hull

* $R_n(\text{convex-hull}(\mathcal{F})) = R_n(\mathcal{F})$ for finite $\mathcal{F}$
* Proof: to be filled in later
* This property is useful because if we want to compute the Rademacher of a polytope, it suffices to compute the Rademacher complexity of its vertices (we will use this property when we look at L_1 regularization).

– 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: linearity of expectation

– Lipschitz composition (kind of a generalization of scaling):

* $R_n(\phi \circ \mathcal{F}) \leq c_\phi R_n(\mathcal{F})$, where $\phi \circ \mathcal{F} = \{z \mapsto \phi(f(z)) : f \in \mathcal{F}\}$ and c_ϕ is the Lipschitz constant of ϕ: $|\phi(z) - \phi(z')| \leq c_\phi |z - z'|$.
* Proof: to be filled in later
* This property is useful because we can analyze the complexity of our hypothesis class (linear functions), and compose with ϕ to get the loss class.

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

– Let $\mathcal{F}$ be a finite class of functions.

– Let M be a bound (which depends on the data $Z_1, \ldots, Z_n$) satisfying:

$$\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} f(Z_i)^2 \leq M^2. \tag{271}$$

– Then

$$\boxed{\hat{R}_n(\mathcal{F}) \leq \sqrt{\frac{2M^2 \log |\mathcal{F}|}{n}}.} \tag{272}$$

- Proof of Lemma 5:

– 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:

$$\exp(t\mathbb{E}[\sup_{f \in \mathcal{F}} W_f \mid Z_{1:n}]) \leq \mathbb{E}[\exp(t \sup_{f \in \mathcal{F}} W_f) \mid Z_{1:n}]. \tag{273}$$

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

$$\leq \mathbb{E}[\sup_{f \in \mathcal{F}} \exp(tW_f) \mid Z_{1:n}]. \tag{274}$$

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

$$\leq \sum_{f \in \mathcal{F}} \mathbb{E}[\exp(tW_f) \mid Z_{1:n}]. \tag{275}$$

- Now we have to bound $\mathbb{E}[\exp(tW_f)]$, which is the moment generating function of W_f.
  - By Hoeffding's lemma (after Lemma 4.6), 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 $\frac{1}{n^2} \sum_{i=1}^n f(Z_i)^2 \leq \frac{M^2}{n}$.
  - By the definition of sub-Gaussian, we have

$$\mathbb{E}[\exp(tW_f) \mid Z_{1:n}] \leq \exp\left(\frac{t^2 M^2}{2n}\right). \tag{276}$$

- Taking logs and re-arranging:

$$\mathbb{E}[\sup_{f \in \mathcal{F}} W_f \mid Z_{1:n}] \leq \frac{\log |\mathcal{F}|}{t} + \frac{tM^2}{2n}. \tag{277}$$

- Optimizing over t yields the result as desired (by just taking the twice the geometric average of the two terms without t):

$$\hat{R}_n(\mathcal{F}) = \mathbb{E}[\sup_{f \in \mathcal{F}} W_f \mid Z_{1:n}] \leq \sqrt{\frac{2M^2 \log |\mathcal{F}|}{n}}. \tag{278}$$

- **Application to fixed dimension**
  - We can immediately apply Massart's finite lemma to any finite loss class $\mathcal{A}$.
  - But we can also apply it to infinite loss classes with finite shattering coefficient, which is the number of distinct vectors $[\ell(Z_1, h), \ldots, \ell(Z_n, h)]$ we get when ranging h over $\mathcal{H}$. This is the effective number of functions that we're taking the sup over.
  - It is important that we use the empirical Rademacher complexity in order to condition on the data. After applying Massart's finite lemma, we can take expectations over the data without changing the bound.
  - This yields, for the zero-one loss ($M = 1$):

$$\boxed{R_n(\mathcal{A}) \leq \sqrt{\frac{\log s(\mathcal{A}, n)}{n}}.} \tag{279}$$

- **Rademacher complexity of linear functions with weights bounded in an L_2 ball**:
  - Let $\mathcal{F} = \{z \mapsto w \cdot z : \|w\|_2 \leq B\}$ (bound on weight vectors).
  - Assume $\mathbb{E}_{Z \sim p^*}[\|Z\|_2^2] \leq C^2$ (bound on spread of data points).
  - Then

$$\boxed{R_n(\mathcal{F}) \leq \frac{BC}{\sqrt{n}}.} \tag{280}$$

  - Proof:
    * Expand the definition:

$$R_n(\mathcal{F}) = \frac{1}{n}\mathbb{E}\left[\sup_{\|w\|_2 \leq B} \sum_{i=1}^{n} \sigma_i (w \cdot Z_i)\right]. \tag{281}$$

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

$$\leq \frac{B}{n}\mathbb{E}\left[\|\sum_{i=1}^{n} \sigma_i Z_i\|_2\right]. \tag{282}$$

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

$$\leq \frac{B}{n}\sqrt{\mathbb{E}\left[\|\sum_{i=1}^{n} \sigma_i Z_i\|_2^2\right]}. \tag{283}$$

    * 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):

$$= \frac{B}{n}\sqrt{\mathbb{E}\left[\sum_{i=1}^{n} \|\sigma_i Z_i\|_2^2\right]}. \tag{284}$$

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

$$= \frac{B}{n}\sqrt{\mathbb{E}\left[\sum_{i=1}^{n} \|Z_i\|_2^2\right]}. \tag{285}$$

    * Use the bound on Z_i:

$$\leq \frac{B}{n}\sqrt{nC^2}. \tag{286}$$

    Simple algebra completes the proof.

- Now let's use this result to obtain an actual generalization bound.
  - So far, we have bounded the Rademacher complexity of the function class $\mathcal{F}$; we need to turn that into a bound on the loss class $\mathcal{A}$.
    - Think of data points as $z = xy$, where $x \in \mathbb{R}^d$, $y \in \{-1, +1\}$ (binary classification).
    - Recall the function class: $\mathcal{F} = \{z \mapsto w \cdot z : \|w\|_2 \leq B\}$, which maps $z = xy$ to the margin that we get on that data point $m = (w \cdot x)y$.
    - The loss class can now be expressed as the composition: $\mathcal{A} = \phi \circ \mathcal{F}$, where $\phi(m) = \mathbb{I}[m \leq 0]$ maps the margin m to the zero-one loss.
  - The problem is that we can't apply the composition directly because ϕ is not Lipschitz due to the discontinuity at 0.
  - So our strategy will be to introduce a surrogate loss function, the **truncated hinge loss**, which upper bounds the zero-one loss:

$$\phi_{\text{surrogate}}(m) = \min(1, \max(0, 1 - m)). \tag{287}$$

    Similarly, define the loss class $\mathcal{A}_{\text{surrogate}} = \phi_{\text{surrogate}} \circ \mathcal{F}$ and expected risk $L_{\text{surrogate}}(h)$ in terms of the surrogate loss. Note that the Lipschitz constant of $\phi_{\text{surrogate}}$ is 1, so we can apply the Lipschitz composition rule:
  - By applying the Lipschitz composition rule (c_ϕ is the Lipschitz constant).

$$R_n(\mathcal{A}_{\text{surrogate}}) \leq R_n(\mathcal{F}) \leq \frac{BC}{\sqrt{n}}. \tag{288}$$

  - Plugging this bound on the Rademacher complexity into (269) yields:

$$\sup_{h \in \mathcal{H}} [L_{\text{surrogate}}(h) - \hat{L}_{\text{surrogate}}(h)] \leq \frac{2BC}{\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{2n}}. \tag{289}$$

  - Combining (289) and (270) and using the fact that the surrogate is an upper bound, we obtain:

$$L(\hat{h}) \leq L_{\text{surrogate}}(\hat{h}) \leq L_{\text{surrogate}}(h^*) + \frac{4BC}{\sqrt{n}} + \sqrt{\frac{2\log(2/\delta)}{n}}. \tag{290}$$

    Note that on the LHS, we have the zero-one loss of the ERM and on the RHS, we have the surrogate loss of the minimizer of expected risk.
  - Remarks
    - These bounds do not depend on the dimensionality d, but only on the L_2 norm of the weight vectors. 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 L_2 norm of the weights), then you will still have good generalization. We will take this one step further when we look at kernels.

* The bounds derived using Rademacher complexity have much better constants than what we had before working directly with tail bounds (we are not losing factors of two everywhere).
* Compare this bound to the regret bound (48) we got in online learning. Note that this essentially gives the same result, but with an extra term that gives high probability guarantees.
* Note that we make no assumptions about convexity here (only Lipschitz is needed). This allows us to use the truncated hinge-loss rather than the zero-one loss, which is a tighter approximation (of course, computation with non-convex loss functions is hard).

– Regularization

* In machine learning, the training objective is usually

$$\min_w \sum_{i=1}^n \ell(Z_i, h) + \frac{\lambda}{2}\|w\|_2^2. \tag{291}$$

* We can interpret λ as the Lagrange multipliers to an equivalent constrained optimization problem:

$$\min_{\|w\|_2 \leq B} \sum_{i=1}^n \ell(Z_i, h), \tag{292}$$

which is exactly the empirical risk minimizer over the L_2-bounded hypothesis class. Note that the relationship between B and λ in general depends on the data.

* Rademacher complexity therefore sheds light on how regularization exactly provides **complexity control**. Note that in online learning, we had mostly talked about (strongly convex) regularization as a source of stability.

- **Kernel methods**

– Recall that kernel methods choose a function $f \in \mathcal{H}$, where $\mathcal{H}$ is an RKHS. The methods generally have the form in (125). Equivalently, we can think of these methods as empirical risk minimizers over a ball of bounded RKHS norm $\mathcal{F} = \{f : \|f\|_{\mathcal{H}} \leq B\}$.

– The empirical Rademacher complexity of a RKHS is bounded as follows:

$$R_n(\mathcal{F}) = \mathbb{E}\left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \sigma_i f(Z_i)\right]. \tag{293}$$

– The key point is that by the **reproducing property**, $f(Z_i) = \langle f, k(Z_i, \cdot)\rangle$, so we can substitute this expression into $w \cdot Z_i$ in (281). Since we have an inner product,

we can carry out the same calculations as before, leaving us with

$$R_n(\mathcal{F}) \leq \frac{B}{n}\sqrt{\mathbb{E}\left[\sum_{i=1}^{n} k(Z_i, Z_i)\right]} = \frac{B\sqrt{\mathbb{E}_{Z\sim p^*}[k(Z,Z)]}}{\sqrt{n}}, \tag{294}$$

where $\sqrt{\mathbb{E}_{Z\sim p^*}[k(Z,Z)]}$ plays the role of $C = \sqrt{\mathbb{E}_{Z\sim p^*}[\|Z\|_2^2]}$.

[begin lecture 3/4] (3/4)

- **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 B 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 simarily, assume that the weights satisfy $\|w\|_1 \leq B$.
  - Let us compute the Rademacher complexity of $\mathcal{F} = \{z \mapsto w \cdot z : \|w\|_1 \leq B\}$.
    - Assume that the coordinates are bounded: $\|Z_i\|_\infty \leq C$ with probability 1 for all data points $i = 1, \ldots, n$.
    - Then

$$\boxed{R_n(\mathcal{F}) \leq \frac{BC\sqrt{2\log(2d)}}{\sqrt{n}}.} \tag{295}$$

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

$$W = \cup_{j=1}^{d}\{Be_j, -Be_j\}. \tag{296}$$

      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:

$$R_n(\mathcal{F}) = \mathbb{E}\left[\sup_{w\in W}\frac{1}{n}\sum_{i=1}^{n}\sigma_i(w\cdot Z_i)\right]. \tag{297}$$

* Applying Massart's finite lemma (Lemma 5), we get that

$$R_n(\mathcal{F}) \leq \mathbb{E}\left[\sqrt{\frac{2B^2(\max_{j\in[d]} \frac{1}{n}\sum_{i=1}^n Z_{ij}^2)\log(2d)}{n}}\right]. \tag{298}$$

* Since each $Z_{ij}^2 \leq C^2$ by definition of C, we have:

$$R_n(\mathcal{F}) \leq \frac{BC\sqrt{2\log(2d)}}{\sqrt{n}}. \tag{299}$$

– Remarks

* It is useful to recall that as p increases,
  · p-norms decrease: $\|w\|_p \geq \|w\|_q$
  · Size of balls increase: $\{w : \|w\|_p \leq B\} \subset \{w : \|w\|_q \leq B\}$
* 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.
* Note that this came up earlier when we looked at exponentiated gradient (EG) for online learning, where the experts resided in an L_1 ball, which allowed us to get a bound that depended logarithmically on the number of experts.

– Ramifications under **sparsity**

* 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 tell us anything about generalization behavior. We seek a stronger justification.
* For convenience, assume that all entries of w and x have magnitude at most 1 ($\|w\|_\infty \leq 1, \|x\|_\infty \leq 1$).
* It suffices to consider the hypothesis class $\|w\|_1 \leq B = s$.
* Then the Rademacher complexity (and thus the generalization error) 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 $B = \sqrt{s}$ and $C = \sqrt{d}$. The Rademacher complexity of $\{\|w\|_2 \leq B\}$ 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 $\sqrt{\log d}$.
* Disclaimer: here, we are comparing upper bounds, which is unsound, but in this case, there are matching lower bounds (though we won't cover them).

4.12 Interpretation of bounds

- 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?
- Assumptions
  - It is healthy to revisit the assumptions of our results.
  - We assume that training examples are i.i.d. In practice, this assumption is violated in various degrees of severity, but some degree of **independence** is really necessary for learning. One can relax independence assumptions by assuming data is generated from a Markov chain, in which case the effective number of examples n must be adjusted by the mixing time of the Markov chain.
  - We assumed that test examples are drawn from the same distribution as the training. In practice, this assumption is false as data drifts over time. These issues are addressed by **domain adaptation**, but again, some relationship between training and test is really necessary.
  - We made very few assumptions about the distribution over the data (other than bounded support or finite second moment). The good news is that the true data generating distribution is probably covered. The bad news is that real data often has more structure that we're not capturing.
- 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 formalize the relevant properties of a learning problem and characterize their relationship to the generalization error, 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 extreme realizable setting, some predictor obtains zero generalization error), then estimation error is smaller ($O(1/n)$ versus $O(1/\sqrt{n})$ convergence).

- **Focus on estimation error**
  - It is important to note that generalization bounds focus on addressing the estimation error (regret) $L(\hat{h}) - L(h^*)$, not the approximation error $L(h^*)$.
  - For example, if d is the number of features, then $L(\hat{h}) - L(h^*) = O(\frac{d}{n})$ shows that by adding more features, the estimation error will worsen. However, we hope that $L(h^*)$ will improve.
  - 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.
- Loss function
  - Is the bound on the right measure of error?
  - 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 truncated hinge loss or 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).
  - Note that this distinction mirrors our online learning regret bounds as well: in the finite experts case, we obtained bounds on expected loss for arbitrary (bounded) loss functions, whereas in general, we needed convexity (which is even stronger than what we require here).

4.13 Summary

- The focus of this section was to study the **generalization error** $L(\hat{h})$ of the **empirical risk minimizer** $\hat{h}$. In particular, we wanted that with probability at least $1 - \delta$, the regret $L(\hat{h}) - L(h^*)$ is upper bounded by something that depends on the complexity of the learning problem and n, the number of i.i.d. training examples.
- The regret 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 regret $L(\hat{h}) - L(h^*)$:
  - Realizable, finite hypothesis class: $O\left(\frac{\log|\mathcal{H}|}{n}\right)$
  - Unrealizable, finite hypothesis class: $O\left(\sqrt{\frac{\log|\mathcal{H}|}{n}}\right)$
  - Unrealizable, shattering/VC: $O\left(\sqrt{\frac{\log s(\mathcal{H},n)}{n}}\right) = O\left(\sqrt{\frac{\mathrm{VC}(\mathcal{H})}{n}}\right)$
  - Unrealizable, L_2 regularization—kernels ($\|w\|_2 \le B, \|x\|_2 \le C$): $O\left(\frac{BC}{\sqrt{n}}\right)$
  - Unrealizable, L_1 regularization—sparsity ($\|w\|_1 \le B, \|x\|_\infty \le C$): $O\left(\frac{BC\sqrt{\log d}}{\sqrt{n}}\right)$
- 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: $\mathbb{P}[G_n - \mathbb{E}[G_n] \ge \epsilon] \le c^{-n\epsilon^2}$.
    * 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 distribution, which has heavier tails.
    * 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 sample, 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 bounded by 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.
- We can compute the Rademacher complexity for L_2 regularization (using linearity and Cauchy-Schwartz), which enables us to analyze kernels.
- We can compute the Rademacher complexity for L_1 regularization (the L_1 polytope is really simple as it only has $2d$ vertices).

4.14 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

5 Direct analysis

5.1 Classical asymptotics

- **Motivation**:
  - Suppose we have two algorithms (estimators) that return $\hat{h}_1$ and $\hat{h}_2$. Which one is better?
  - Let $L(\hat{h}_1; p^*)$ and $L(\hat{h}_2; p^*)$ denote the generalization errors of the two, where we've made the dependence on the data generating distribution $p^* \in \mathcal{P}$ explicit ($\mathcal{P}$ is the family of distributions we're considering).
  - We can derive upper bounds on the generalization error, but it's invalid to just compare these two upper bounds, because discrepancies could reflect our ability to prove bounds rather than real phenomena.
  - There are techniques to get lower bounds (e.g., using Fano's inequality). So suppose we get lower and upper bounds:

$$\text{Lower}_i \leq \sup_{p^* \in \mathcal{P}} L(\hat{h}_i; p^*) \leq \text{Upper}_i \quad \text{for } i \in \{1, 2\}. \tag{300}$$

    To show $\hat{h}_1$ is better than $\hat{h}_2$, we would show that $\text{Upper}_1 \leq \text{Lower}_2$.
  - However, these lower bounds are obtained on the worst case p^*, and moreover, the p^* might be different between $\hat{h}_1$ and $\hat{h}_2$.
  - What we really want is control over

$$L(\hat{h}_1; p^*) - L(\hat{h}_2; p^*), \tag{301}$$

    for any $p^* \in \mathcal{P}$.
  - We will show that classical asymptotics can give us a handle on this. The main idea is to approximate $L(\hat{h}_i; p^*)$ directly in terms of some function of p^* plus some lower order terms in n.

$$L(\hat{h}_i; p^*) \approxeq \text{Approx}_i(p^*). \tag{302}$$

    We can compute $\text{Approx}_1(p^*) - \text{Approx}_2(p^*)$, which converges to the correct difference *for any* $p^* \in \mathcal{P}$.
  - The price we pay is that n has to be sufficiently large and that the loss must be twice differentiable.

- Setup
  - $z = (x, y)$ is an example
  - $\ell(z, \theta)$ is a loss function on example z with parameters $\theta \in \mathbb{R}^d$

- Let $p^*(z)$ be the true distribution over examples z

- Assumptions
  - Assume $z, z^{(1)}, \ldots, z^{(n)} \sim p^*$ (i.i.d.),
  - 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.
  - Let $\nabla^2 \ell(z, \theta) \in \mathbb{R}^{d \times d}$ be the Hessian of the loss at θ.
  - Assume that the expected loss Hessian $\mathbb{E}[\nabla^2 \ell(z, \theta)] \succ 0$ is positive definite. This assumption is actually not needed, but it will make the math simpler.[13]
  - **Consistency**: We assume that $\hat{\theta} \xrightarrow{P} \theta^*$ (not hard since we're in finite dimensional spaces).

- Definitions
  - Let $\theta^* \in \mathbb{R}^d$ be the minimizer of the expected risk:

$$\theta^* \stackrel{\text{def}}{=} \arg\min_{\theta \in \mathbb{R}^d} L(\theta), \quad L(\theta) \stackrel{\text{def}}{=} \mathbb{E}[\ell(z, \theta)] \tag{303}$$

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

$$\hat{\theta} \stackrel{\text{def}}{=} \arg\min_{\theta \in \mathbb{R}^d} \hat{L}(\theta), \quad \hat{L}(\theta) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell(z^{(i)}, \theta). \tag{304}$$

  - 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 $\mathbb{P}[|X_n / f(n)| > M_\epsilon] < \epsilon$.

- Outline of analysis
  - The main idea of the analysis is to perform **Taylor expansions**.
  - Step 1: obtain an asymptotic expression for the **parameter error** by Taylor expanding the gradient of the empirical risk.

$$\boxed{\hat{\theta} - \theta^*.} \tag{305}$$

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

$$\boxed{L(\hat{\theta}) - L(\theta^*).} \tag{306}$$

[13] 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.

[begin lecture 3/6] (3/6)

- **Definition 20 (Well-specified model)**
  - Assume that the loss function corresponds to the log-likelihood under a probabilistic model:

$$\ell((x,y);\theta) = -\log p_\theta(y \mid x). \tag{307}$$

  - We say that this model is
    - **jointly well-specified** if $p^*(x,y) = p_{\theta^*}(x,y)$ for some parameter $\theta^* \in \mathbb{R}^d$. This places a very assumption about the data.
    - **conditionally well-specified** if $p^*(x,y) = p^*(x)p_{\theta^*}(y \mid x)$ for some parameter $\theta^* \in \mathbb{R}^d$. This assumption is milder because $p^*(x)$ can have any form.
  - In the well-specified case (jointly or conditionally), the following identity holds:

$$\boxed{\nabla^2 L(\theta^*) = \mathrm{var}(\nabla \ell(z,\theta^*)).} \tag{308}$$

    This quantity is known as the (conditional) **Fisher information** matrix.
    - Proof:
      - Using the fact that probability densities integrate to one:

$$\int e^{-\ell(z,\theta^*)} p^*(x)dx = 1. \tag{309}$$

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

$$\int e^{-\ell(z,\theta^*)}(-\nabla \ell(z,\theta^*))p^*(x)dx = 0. \tag{310}$$

        Note that this implies $\mathbb{E}[\nabla \ell(z,\theta^*)] = 0$.
      - Differentiating again, using the product rule:

$$\int [-e^{-\ell(z,\theta^*)}\nabla^2 \ell(z,\theta^*) + e^{-\ell(z,\theta^*)}\nabla \ell(z,\theta^*)\nabla \ell(z,\theta^*)^\top]p^*(x)dx = 0. \tag{311}$$

      - Re-arranging:

$$\mathbb{E}[\nabla^2 \ell(z,\theta^*)] = \mathbb{E}[\nabla \ell(z,\theta^*)\nabla \ell(z,\theta^*)^\top]. \tag{312}$$

      - Using the fact that $\mathbb{E}[\nabla \ell(z,\theta^*)] = 0$ and the definition of $L(\theta)$ yields the result.
  - We will not assume the model is well-specified in general. We will only make the assumptions for various examples to get simple expressions to gain intuition.

- **Example 17 (well-specified 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 $\text{var}(\epsilon) = 1$
  - Loss function: $\ell((x, y), \theta) = \frac{1}{2}(\theta \cdot x - y)^2$ (note that the model is well-specified)
  - Hession of expected risk: $\nabla^2 L(\theta) = \mathbb{E}[xx^\top]$ (covariance matrix of data)

  Now let us analyze the parameter error (step 1) and generalization error (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^*:

$$\nabla \hat{L}(\hat{\theta}) = \nabla \hat{L}(\theta^*) + \nabla^2 \hat{L}(\theta^*)(\hat{\theta} - \theta^*) + O_p(\|\hat{\theta} - \theta^*\|_2^2). \tag{313}$$

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

$$\hat{\theta} - \theta^* = -\nabla^2 \hat{L}(\theta^*)^{-1} \left(\nabla \hat{L}(\theta^*) + O_p(\|\hat{\theta} - \theta^*\|_2^2) \right). \tag{314}$$

  - As $n \to \infty$:
    - Since the inverse is a smooth function around θ^* (we assumed $\nabla^2 L(\theta^*) \succ 0$), we have convergence in probability:

$$\nabla^2 \hat{L}(\theta^*)^{-1} \xrightarrow{P} \nabla^2 L(\theta^*)^{-1}. \tag{315}$$

    - $\hat{L}(\theta^*)$ 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:

$$\sqrt{n} \cdot \nabla \hat{L}(\theta^*) \xrightarrow{d} \mathcal{N}(0, \text{var}(\nabla \ell(z, \theta^*))). \tag{316}$$

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

    - Suppose $\hat{\theta} - \theta^* = O_p(f(n))$. By (314), $f(n)$ 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:

$$\sqrt{n} \cdot O_p(\|\hat{\theta} - \theta^*\|_2^2) \xrightarrow{P} 0. \tag{317}$$

– By standard results involving convergence in probability and distribution (Slutsky's theorem), we can substitute the limits into (314) to obtain:

$$\boxed{\sqrt{n} \cdot \underbrace{(\hat{\theta} - \theta^*)}_{\text{parameter error}} \xrightarrow{d} \mathcal{N}\left(0, \nabla^2 L(\theta^*)^{-1} \text{var}(\nabla \ell(z, \theta^*)) \nabla^2 L(\theta^*)^{-1}\right),} \tag{318}$$

where we used the fact that if $x_n \xrightarrow{d} \mathcal{N}(0, \Sigma)$, then $Ax_n \xrightarrow{d} \mathcal{N}(0, A\Sigma A^\top)$. 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.
* $\text{var}(\nabla \ell(z, \theta^*))$: measures the **variance** in the loss gradient. The less there is, the better.

– Example (well-specified linear regression):

* In this case, due to (308), we have $\text{var}(\nabla \ell(z, \theta^*)) = \mathbb{E}[\nabla^2 \ell(z, \theta^*)] = \mathbb{E}[xx^\top]$, so the variance factor is canceled out by one of the curvature factors.

$$\boxed{\sqrt{n} \cdot (\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}\left(0, \mathbb{E}[xx^\top]^{-1}\right).} \tag{319}$$

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 expected risk

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

$$L(\hat{\theta}) = L(\theta^*) + \nabla L(\theta^*)^\top (\hat{\theta} - \theta^*) + \frac{1}{2}(\hat{\theta} - \theta^*)^\top \nabla^2 L(\theta^*)(\hat{\theta} - \theta^*) + O_p(\|\hat{\theta} - \theta^*\|_2^3). \tag{320}$$

– 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.

– Substituting in the parameter error:

* Fact: if $x_n \xrightarrow{d} \mathcal{N}(0, \Sigma)$, then $x_n x_n^\top \xrightarrow{d} \mathcal{W}(\Sigma, 1)$ and $x_n^\top x_n = \text{tr}(x_n x_n^\top) \xrightarrow{d} \text{tr}(\mathcal{W}(\Sigma, 1))$,[14] which is a sum of χ^2 distributed variables.

$$\boxed{n(L(\hat{\theta}) - L(\theta^*)) = \frac{1}{2} \text{tr}\, \mathcal{W}\left(\nabla^2 L(\theta^*)^{-\frac{1}{2}} \text{var}(\nabla \ell(z, \theta^*)) \nabla^2 L(\theta^*)^{-\frac{1}{2}}, 1\right),} \tag{321}$$

where $\mathcal{W}(V, n)$ is the Wishart distribution with scale matrix V and n degrees of freedom.

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

- Example (well-specified models):
  - Since the model is well-specified, everything cancels nicely, resulting in:

$$\boxed{n(L(\hat{\theta}) - L(\theta^*)) \xrightarrow{d} \frac{1}{2} \operatorname{tr} \mathcal{W}(I_{d\times d}, 1).} \tag{322}$$

  - The limiting distribution is half times a χ^2_d distributed random variable.
  - 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: $\operatorname{var}(n(L(\hat{\theta}) - L(\theta^*))) \to \frac{d}{2}$.

    In short, $L(\hat{\theta}) - L(\theta^*) = O_p\left(\frac{d}{n}\right)$.
  - Interestingly, in the well-specified case, the generalization error does not dependence 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 regret $L(\hat{\theta}) - L(\theta^*)$ 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.
  - Question: can we improve on the empirical risk minimizer (maximum likelihood estimator)?
    - The short answer is no, in that it achieves the lowest asymptotic variance.
    - The Cramer-Rao lower bound states that the variance of an estimator is lower bounded by the inverse of the Fisher information matrix.

- Let us compare the asymptotic expression (322) with results that we've derived previously in this class.

- Using uniform convergence, we were only able to get a $\frac{1}{\sqrt{n}}$ convergence rate for the unrealizable setting. This was unavoidable using our techniques since we relied on concentration of empirical risk to expected risk, which even for a single hypothesis is already $\frac{1}{\sqrt{n}}$.
- How did we get a faster rate? While asymptotics gives us the correct rate (as long as $n \to \infty$ while all other quantities such as dimensionality remain constant), there are non-asymptotic effects hidden away in the $O_p\left(n^{-\frac{3}{2}}\right)$ terms. The picture is more complicated: there can be both $\frac{1}{\sqrt{n}}$ and $\frac{1}{n}$ terms with different constants capturing different aspects regarding the complexity of the learning problem. For instance, $\frac{1}{\sqrt{n}}$ is the rate associated with the norm, while $\frac{1}{n}$ is associated with the dimensionality.
- In online learning, we were able to get a $\frac{\log n}{n}$ bound for strongly convex loss functions. However, strong convexity is too much to ask for, since any linear model will not satisfy this. In the asymptotic setting, we only need strong convexity in expectation (averaged over data points) at θ^* (remember, the entire analysis operates locally around θ^*).

- Comparison of generative versus discriminative models
  - One of the main motivations of asymptotic analysis was that we could compare different estimators.
  - In this section, let's assume that the model is *jointly* well-specified and is an exponential family (includes logistic regression, conditional random fields, MRFs, etc):

$$p_\theta(x, y) = \exp\left(\phi(x, y)^\top \theta - A(\theta)\right), \tag{323}$$

    where
    - $\phi(x, y) \in \mathbb{R}^d$ is the feature vector,
    - $\theta \in \mathbb{R}^d$ are the parameters, and
    - $A(\theta) = \log \int_{\mathcal{X} \times \mathcal{Y}} \exp(\phi(x, y)^\top \theta) dx dy$ is the joint log-partition function.
    - $A(\theta; x) = \log \int_{\mathcal{Y}} \exp(\phi(x, y)^\top \theta) dy$ is the conditional log-partition function (useful later).
  - We consider two estimators which are used to train:
    - Generative: $\ell_{\text{gen}}((x, y), \theta) = -\log p_\theta(x, y)$ defines estimator $\hat{\theta}_{\text{gen}}$
    - Discriminative: $\ell_{\text{dis}}((x, y), \theta) = -\log p_\theta(y \mid x)$ defines estimator $\hat{\theta}_{\text{dis}}$

    Here, we are being careful to define the estimators with respect to the same model, but only changing the estimator as to pin down the underlying essence between generative and discriminative estimation.

– Important: note that we are using different loss functions at training time, although at test time, we still evaluate using the discriminative loss ℓ_{dis}.

– Recall that the asymptotic variance of the estimators $\hat{\theta} - \theta^*$ are functions is $\nabla^2 \ell(z;\theta)^{-1}$.

– For exponential families, the derivatives are simply the moments of the distributions:

  * $L_{\text{gen}}(\theta^*) = \mathbb{E}[\nabla^2 \ell_{\text{gen}}((x,y),\theta^*)] = \nabla^2 A(\theta^*) = \text{var}(\phi(x,y))$.
  * $L_{\text{dis}}(\theta^*) = \mathbb{E}[\nabla^2 \ell_{\text{dis}}((x,y),\theta^*)] = \mathbb{E}[\nabla^2 A(\theta;x)] = \mathbb{E}[\text{var}(\phi(x,y) \mid x)]$.

– Key variance decomposition identity:

$$\text{var}(\phi(x,y)) = \mathbb{E}[\text{var}(\phi(x,y) \mid x)] + \text{var}(\mathbb{E}[\phi(x,y) \mid x]). \tag{324}$$

Since variance matrices are PSD, we have that

$$\text{var}(\phi(x,y)) \succeq \mathbb{E}[\text{var}(\phi(x,y) \mid x)]. \tag{325}$$

Inverting:

$$\underbrace{\text{var}(\phi(x,y))}_{\text{asymptotic variance of } \hat{\theta}_{\text{gen}} - \theta^*} \preceq \underbrace{\mathbb{E}[\text{var}(\phi(x,y) \mid x)]}_{\text{asymptotic variance of } \hat{\theta}_{\text{dis}} - \theta^*}. \tag{326}$$

This says that the asymptotic variance of generative estimator ($\hat{\theta}_{\text{gen}}$) to be at most the asymptotic variance of the discriminative estimator ($\hat{\theta}_{\text{dis}}$).

– To get the generalization error from the parameter error, we simply left and right multiply by the *same* matrix $\nabla^2 L_{\text{dis}}(\theta^*)^{-\frac{1}{2}}$. Therefore, the generalization error of the generative estimator is at most the generaliztaion of the discriminative estimator.

– However, if the model is not jointly well-specified, then the two estimators will not even converge to the same θ^* in general, and the discriminative estimator will clearly be better since it converges to the expected risk minimizer.

[begin lecture 3/11] (3/11)

6 Method of moments

6.1 Motivation

- In this section, we will consider the problem of **parameter estimation** in latent-variable models such as Gaussian mixture models (GMMs), Hidden Markov Models (HMMs), and Latent Dirichlet Allocation.
- The learning problem is as follows:
  - Model: $p_\theta(x, h)$ for $\theta \in \Theta \subset \mathbb{R}^d$
  - Input: examples $x^{(1)}, \ldots, x^{(n)}$ drawn i.i.d. from p_{θ^*}
  - Output: parameter estimates $\hat{\theta}$
- Parameter estimation in these models is typically based on the principle of **maximum likelihood**. Maximizing the likelihood exactly is computationally difficult, and is usually tackled by **local** methods such as EM, gradient descent, variational inference which all converge to only local optima. MCMC methods are guaranteed to converge in the limit, but will take exponentially long.
- In this section, we will show that we can actually construct an algorithm that converges in a **global** sense to the correct parameters. The algorithm is not based on the likelihood, but rather on method of moments, which allows for a computationally more efficient solution at the cost of some statistical efficiency.
- Throughout this section, we will assume that our models are well-specified, that is, data is actually generated from our model.
- As a concrete example, we consider learning the following latent-variable model:
- **Example 18 (Naive Bayes clustering model)**
  - Let k be the number of possible document clusters.
  - Let d be the number of possible word types in a vocabulary.
  - Model parameters $\theta = (\pi, B)$
    - $\pi \in \Delta_k$: prior distribution over clusters.
    - $B = (\beta_1, \ldots, \beta_k) \in (\Delta_d)^k$: distributions over words given cluster

    Let Θ denote the set of valid parameters.
  - The generative model is as follows: For each document $i = 1, \ldots, n$:

    * Sample the label: $h^{(i)} \sim \pi$
    * For each word $j = 1, \ldots, L$:
        · Sample a word: $x_j^{(i)} \sim \beta_{h^{(i)}}$.

- A typical approach to parameter estimation is to maximize the marginal likelihood of the observations:

$$\min_{\theta \in \Theta} \sum_{i=1}^{n} -\log \sum_{h=1}^{k} p_\theta(h, x^{(i)}). \tag{327}$$

- However, the optimization problem is non-convex, so in general there is no efficient algorithm to find the optimum. A popular solution is to use the EM algorithm, which can be shown to be a bound optimization algorithm (repeatedly construct a lower bound and optimize) or coordinate-wise ascent on an objective.

  – E-step: for each example i, compute $q_i(h) = p_\theta(h^{(i)} = h \mid x^{(i)})$.
  – M-step: optimize the expected log-likelihood: $\max_\theta \sum_{i=1}^{n} \sum_{h=1}^{k} q_i(h) \log p_\theta(h, x^{(i)})$.

  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.

- The goal of this section is to develop a new method of estimating the parameters of these latent variable models which completely sidesteps problems with local optima.

- How is this possible given that maximizing the likelihood function is intrinisically NP hard in general?

- The trick is to move away from likelihood approaches to parameter estimation and adopt a **method of moments** approach.

6.2 Method of moments framework

- The method of moments is an old but simple idea from statistics.

  – Step 1: define a **moment mapping** relating the model parameters to properties (i.e., moments) of the data distribution specified by those parameters.
  – Step 2: **plug in** the empirical moments and invert the mapping to get parameter estimates.

- Moment mapping

  – Let $\phi(x) \in \mathbb{R}^p$ be an observation function which only depends on the observed variables x.

* For example: $\phi(x) = (x, x^2)$.

– Define the **moment mapping**

$$\boxed{M(\theta) \stackrel{\text{def}}{=} \mathbb{E}_{x \sim p_\theta}[\phi(x)],} \tag{328}$$

which maps each parameter vector $\theta \in \mathbb{R}^d$ to the expected value of $\phi(x)$ with respect to $p_\theta(x)$.

– Gaussian example:

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

$$M((\mu, \sigma^2)) = \mathbb{E}_{x \sim \mathcal{N}(\mu, \sigma^2)}[(x, x^2)] = (\mu, \sigma^2 + \mu^2). \tag{329}$$

– Suppose that someone told us m^* (where $m^* = M(\theta^*)$). Then assuming M were invertible, we could solve for $\theta^* = M^{-1}(m^*)$. Invertibility is important enough that it has a name in statistics:

– **Definition 21 (identifiability)**

* Let $\Theta \subset \mathbb{R}^d$ be a model family.
* Define the confusable set for $\theta \in \Theta$ as

$$S(\theta) \stackrel{\text{def}}{=} \{\theta' \in \Theta : M(\theta) = M(\theta')\}. \tag{330}$$

to be the set of parameters that have the same moments as θ.

* We say that Θ *is identifiable from* ϕ if for almost $\theta \in \Theta$, we have $|S(\theta)| = 1$.

– Gaussian example

* We can recover the parameters $\theta^* = (\mu^*, \sigma^{2*})$ from $m^* = (m_1^*, m_2^*)$ as follows:
  · $\mu^* = m_1^*$
  · $\sigma^{2*} = m_2^* - m_1^{2*}$

- 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 using sample averages:

$$\boxed{\hat{m} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \phi(x^{(i)}).} \tag{331}$$

– We can simply **plug in** $\hat{m}$ for m^*:

$$\boxed{\hat{\theta} \stackrel{\text{def}}{=} M^{-1}(\hat{m}).} \tag{332}$$

- As $\hat{m}$ approaches m^*, $\hat{\theta}$ approaches θ^*. But how fast?
- It is relatively easy to study how fast $\hat{m}$ converges to m^*. Since $\hat{m}$ is just an average of i.i.d. variables, we can apply the central limit theorem:

$$\sqrt{n}(\hat{m} - m^*) \xrightarrow{d} \mathcal{N}(0, \text{var}_{x \sim p^*}(\phi(x))). \tag{333}$$

- Assuming that M^{-1} is continuous around m^*, how fast $\hat{\theta}$ approaches θ^* can be gotten by the delta-method:

$$\sqrt{n}(\hat{\theta} - \theta^*) \xrightarrow{d} \mathcal{N}(0, \nabla M^{-1}(m^*) \text{var}_{x \sim p^*}(\phi(x)) \nabla M^{-1}(m^*)^\top), \tag{334}$$

  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 $\dagger$ 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.

- Example (exponential families):
  - For exponential families, method of moments with the sufficient statistics corresponds exactly the maximum likelihood. Recall the exponential family:

$$p_\theta(x) = \exp\left(\phi(x)^\top \theta - A(\theta)\right), \tag{335}$$

    where
    - $\phi(x) \in \mathbb{R}^d$ is the feature vector (observation function),
    - $\theta \in \mathbb{R}^d$ are the parameters, and
    - $A(\theta) = \log \int_{\mathcal{X}} \exp(\phi(x)^\top \theta) dx$ is the log-partition function.
  - Proposition: the maximum likelihood estimator is exactly the method of moments estimator
  - Proof
    - Key fact (by properties of the log-partition function):

$$\nabla A(\theta) = \mathbb{E}_{x \sim p_\theta}[\phi(x)] = M(\theta). \tag{336}$$

    - Optimality conditions for maximum likelihood:

$$\frac{1}{n} \sum_{i=1}^n \nabla \log p_{\hat{\theta}}(x^{(i)}) = 0 \quad \Leftrightarrow \quad \hat{m} = M(\hat{\theta}). \tag{337}$$

- Maximum likelihood estimation is a convex optimization problem for exponential families, but in general, maximum likelihood is not convex, nor does it coincide with the method of moments.

- In general, computing the inverse moment mapping M^{-1} is just as hard as finding 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 well-conditioned too (M provides enough information about the parameters θ).
  - The inverse moment mapping M^{-1} is computationally tractable.

6.3 Method of moments for latent-variable models

- Now that we have established the principles behind method of moments, let us tackle the Naive Bayes clustering model (Example 18).
- Our strategy is to just start writing some moments and see what kind of equations we get (they turn out to be product of matrices).
- Assume each document has $L \geq 3$ words (this is an unimportant technicality).
- Notation: we are going to switch to using d as the number of words rather than the number of parameters.
- For notational convenience, assume that each word $x_j \in \mathbb{R}^d$ is represented as an indicator vector which is one in the entry of that word and zero elsewhere.
- Let's start with the first-order moments:

$$M_1 \stackrel{\text{def}}{=} \mathbb{E}[x_1] = \sum_{h=1}^{k} \pi_h \beta_h = B\pi. \tag{338}$$

  - Note that the moments require marginalizing out the latent variables, and marginalization corresponds to matrix products.
  - Interpretation: $M_1 \in \mathbb{R}^d$ is a vector of marginal word probabilities.
  - Clearly this is not enough information to identify the parameters.

- We can write the second-order moments:

$$M_2 \stackrel{\text{def}}{=} \mathbb{E}[x_1 x_2^\top] = \sum_{h=1}^{k} \pi_h \beta_h \beta_h^\top = B\text{diag}(\pi)B^\top. \tag{339}$$

  - 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.

- It turns out that the model family is non-identifiable from second-order moments. So proceed to third-order moments to get more information.

$$M_3 \stackrel{\text{def}}{=} \mathbb{E}[x_1 \otimes x_2^\top \otimes x_3] = \sum_{h=1}^{k} \pi_h \beta_h \otimes \beta_h \otimes \beta_h. \tag{340}$$

  - Interpretation $M_3 \in \mathbb{R}^{d \times d \times d}$ is a rank-3 tensor of co-occurrence word probabilities.
  - We will show that three moments is sufficient for identifying the model parameters.

- We will show one approach for using M_1, M_2, M_3 to identify $\theta = (\pi, B)$.

  - Let $\eta \in \mathbb{R}^d$ be a random vector.
  - Define the projection of M_3 onto η as follows:

$$M_3(\eta) \stackrel{\text{def}}{=} \sum_{h=1}^{k} \pi_h \beta_h \otimes \beta_h (\beta_h^\top \eta) = B \operatorname{diag}(\pi) \operatorname{diag}(B^\top \eta) B^\top. \tag{341}$$

  - 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.

- Lemma

  - Suppose we observe matrices $X = BDB^\top$ and $Y = BEB^\top$.
  - Assume
    - D, E are diagonal matrices such that the ratios $\{D_{ii}/E_{ii}\}$ are all non-zero and distinct.
    - $B \in \mathbb{R}^{d \times k}$ 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 $d \geq k$.
  - Then we can recover B (up to permutation/scaling of its columns).

- Proof:

  - Simple case
    - Assume B is invertible ($d = k$).
    - Then X and Y are also invertible.
    - Compute

$$YX^{-1} = BEB^\top B^{-\top} D^{-1} B^{-1} = B \underbrace{ED^{-1}}_{\text{diagonal}} B^{-1}. \tag{342}$$

    - 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.
    - For this to work, we need the diagonal elements of ED^{-1} to be distinct and non-zero so that the eigenvalues are identifiable.
  - Now suppose X and Y are not invertible. Note that X and Y have the same column space, so we can project them down into that column space where they will be invertible.
  - Let $U \in \mathbb{R}^{d \times k}$ 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 $B \in \mathbb{R}^{d \times k}$ is not invertible, $\tilde{B} \stackrel{\text{def}}{=} U^\top B \in \mathbb{R}^{k \times k}$ 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, $D = \text{diag}(\pi)$ and $E = \text{diag}(\pi)\,\text{diag}(B^\top \eta)$ 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 demonstrated the essence of the method of moments approach [Anandkumar/Hsu/Kakade, 2012].

- Extensions

  - We can also derive finite sample complexity bounds that $\hat{\theta}$ deviates from θ^* (up to permutation of clusters) using results from matrix perturbation theory (e.g., if perturb matrix with noise, how do eigenvectors change?)
  - These techniques can be further extended to obtain consistent parameter estimates for GMMs, HMMs, LDA, restricted PCFGs, ICA, linear Bayesian networks, mixture of linear regressions, etc.

- Practical considerations

  - The algorithm requires making one pass over the data. The first step is to estimate the moments, which is a simple operation which can be parallelized.

- We can avoid forming $d \times d$ matrices by randomly projecting down into lower dimensions.
- As presented, the algorithm is to only look at the first three words of each document, which is absolutely crazy. In practice, one would consider all combinations of three words. This point also highlights a caveat with these techniques: one should not be blind to the fact that models are never well-specified in practice, so an estimator that works under the well-specified assumption might need to be adapted to make it more robust.
- It turns out that the method of moments estimator that we derived from the third-order moments is not statistically efficient: that is, the asymptotic variance of this estimator (334) will be higher than that of the maximum likelihood estimator (318). Intuitively this is because we are only using the first three moments and higher-order moments provide more information. However, the statistical deficiency is made up for by the computational gains. These methods are perfect for the large data regime , where we really need to be able to process large amounts of data quickly. For unsupervised learning using latent-variable models, this is exactly the regime we're in.
- In practice, lower statistical efficiency can be problematic. It is beneficial to use the method of moments as initialization and run local optimization (EM), which will improve the (empirical) log-likelihood.

- In summary, the key idea of method of moments for these latent-variable models is to have three views of h which are conditionally independent. Using two views allows us to identify the column space, but not the parameters, while three views also pins down the exact basis.

6.4 Summary

- The goal is to obtain computationally efficient procedures that return parameter estimates $\hat{\theta}$ that provably converge to θ^*.
- The method of moments provides a general framework for exploring these procedures. By selecting different observation functions ϕ, we can reveal more or less information about the data, making it harder or easier to invert the moment mapping to obtain parameter estimates.
- In the context of latent-variable models where parameter estimation has traditionally been intractable due to non-convexity, we see that method of moments can provide a fresh perspective. By using just the first-, second-, and third-order moments, we can obtain consistent parameter estimates efficiently by simply performing an eigendecomposition.

6.5 References

- Anandkumar/Hsu/Kakade, 2012: A Method of Moments for Mixture Models and Hidden Markov Models
- Anandkumar/Foster/Hsu/Kakade/Liu, 2012: Two SVDs Suffice: Spectral decompositions for probabilistic topic modeling and latent Dirichlet allocation
- Anandkumar/Ge/Hsu/Kakade/Telgarsky, 2012: Spectral learning of latent-variable models

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 should hopefully be clear from context.

- General 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:

$$\mathbb{I}[\text{predicate}] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if predicate is true} \\ 0 & \text{otherwise.} \end{cases} \tag{343}$$

  - Indicator (zero-infinity) function:

$$\mathbb{W}[\text{predicate}] \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if predicate is true} \\ \infty & \text{otherwise.} \end{cases} \tag{344}$$

  - Probability simplex:

$$\Delta_d \stackrel{\text{def}}{=} \left\{ w \in \mathbb{R}^d : w \succeq 0 \text{ and } \sum_{i=1}^{d} w_i = 1 \right\}. \tag{345}$$

  - Euclidean projection:

$$\Pi_S(w) \stackrel{\text{def}}{=} \arg\min_{u \in S} \|u - w\|_2 \tag{346}$$

    is the closest point (measured using Euclidean distance) to w that's in S.

- 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 square matrix $A \in \mathbb{R}^{n \times n}$ is PSD iff A is symmetric and $v^\top A v \geq 0$ for all vectors $v \in \mathbb{R}^n$.
- Eigenvalues: for a PSD matrix $A \in \mathbb{R}^{n \times n}$, let $\lambda_1(A), \ldots, \lambda_n(A)$ be the eigenvalues of A, sorted in decreasing 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.

- Probability and statistics
  - Probability measure: $\mathbb{P}$
  - Expectation:
    - $\mathbb{E}[m(x)] = \int_{\mathcal{X}} m(x)p(x)dx$, where $p(x)$ is a density function
    - $\mathbb{E}[m(x)] = \sum_{x \in \mathcal{X}} m(x)p(x)$, where $p(x)$ is a probability mass function

- Functional analysis
  - **Definition 22 (Cauchy sequence)**

    A Cauchy sequence in a metric space (X, d) is $(x_i)_{i \geq 1}$ such that for any $\epsilon > 0$, there exists an integer n such that all $i, j > n$, $d(x_i, x_j) < \epsilon$.
  - **Definition 23 (Complete metric space)**

    A complete metric space (X, d) 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:

$$\|f\|_p \stackrel{\text{def}}{=} \left(\int |f(x)|^p dx \right)^{1/p} < \infty. \tag{347}$$

  - 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.

- We will try to stick with the following conventions:
  - x: input
  - y: output

- 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 Basic theorems

- Jensen's inequality
  - For any convex function $f : \mathbb{R}^d \mapsto \mathbb{R}$ and random variable $X \in \mathbb{R}^d$,
$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]. \tag{348}$$
- Hölder's inequality
  - For any real vectors $u, v \in \mathbb{R}^d$,
$$|u \cdot v| \leq \|u\|_p \|v\|_q, \tag{349}$$
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).
- Markov's inequality
  - Let $X \geq 0$ be a non-negative random variable.
  - Then for any $\epsilon > 0$,
$$\mathbb{P}[X \geq \mathbb{E}[X] + \epsilon] \leq \frac{\mathbb{E}[X]}{\epsilon}. \tag{350}$$
This is the most basic (and weakest) tail bound. If we know something about the mean $\mathbb{E}[X]$, we can bound the deviation of X from that mean.