What is Machine Learning?

- The automated process of “making sense” out of data:
  A tool to extract *information* from data and use it.

- ML has become everywhere!

  ✓*Search engines, recommendation systems,*

  ✓*Email spam detection, fraud detection in credit cards,*

  ✓*Personal assistance in smart phones, face detection in digital cameras,*

  ✓*Navigation, military applications, medicine, bioinformatics, astronomy,..*

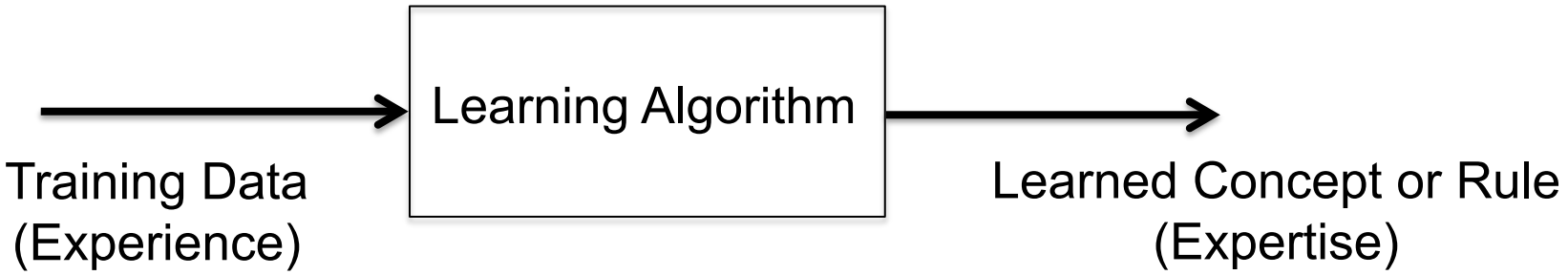
- How is ML different from traditional programming?
  - *Providing programs the ability to “learn” and adapt to data on their own.*

What is Learning?

- The process of transforming an experience into expertise or knowledge.

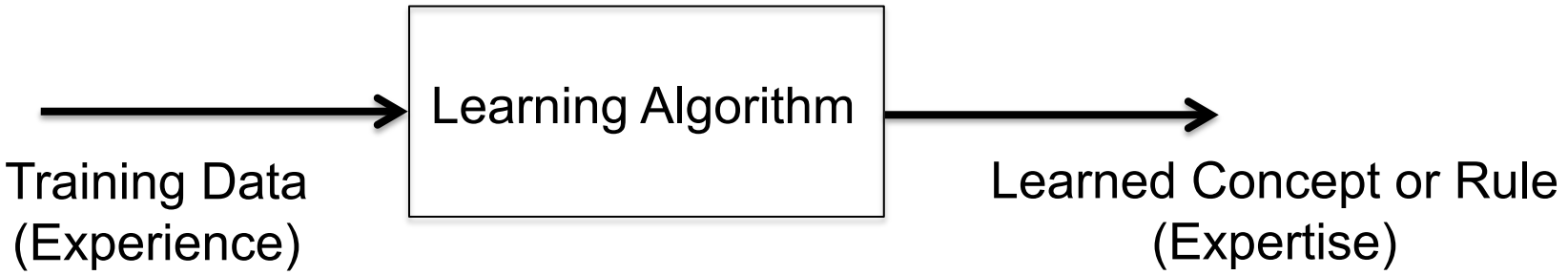
What is Learning?

- The process of transforming an experience into expertise or knowledge.



What is Learning?

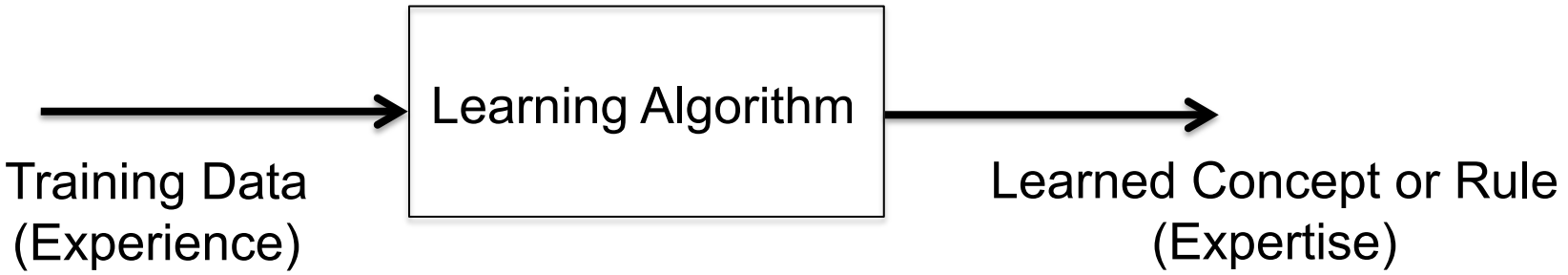
- The process of transforming an experience into expertise or knowledge.



- **Example:** Spam Detection
  - Input: Set of emails, each labeled: *Spam*, or *Not Spam.*
  - Output: Prediction rule to classify emails

What is Learning?

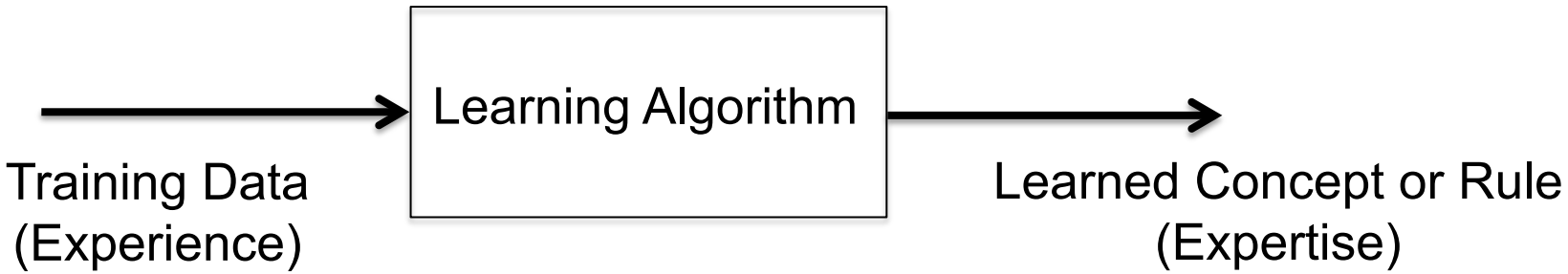
- The process of transforming an experience into expertise or knowledge.



- **Example:** Spam Detection
  - Input: Set of emails, each labeled: *Spam*, or *Not Spam.*
  - Output: Prediction rule to classify emails

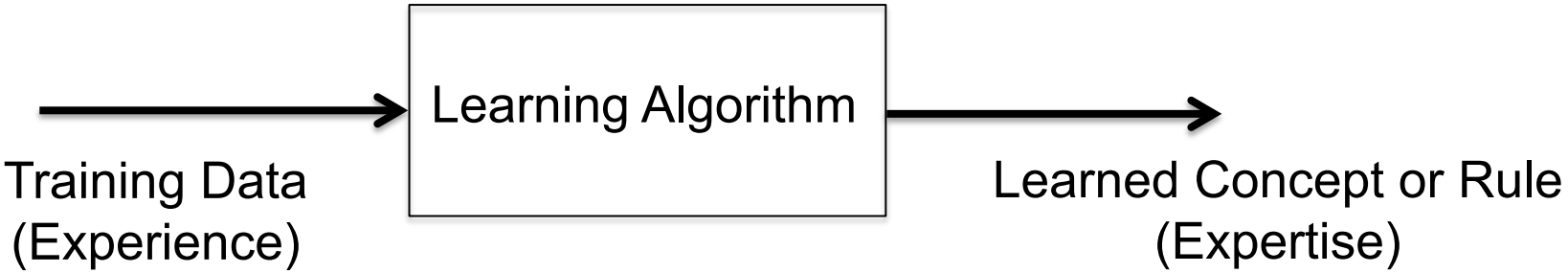
- How to evaluate a learning algorithm?
  - Test the output (e.g., the prediction rule) on new *unseen* data (called test data).

Fundamental Questions



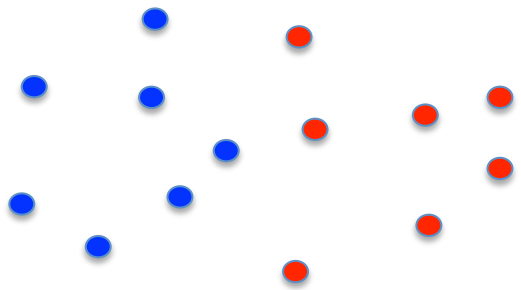
- What assumptions do we need for learning to be possible?
  - Training and test data are “similar” in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.

Fundamental Questions

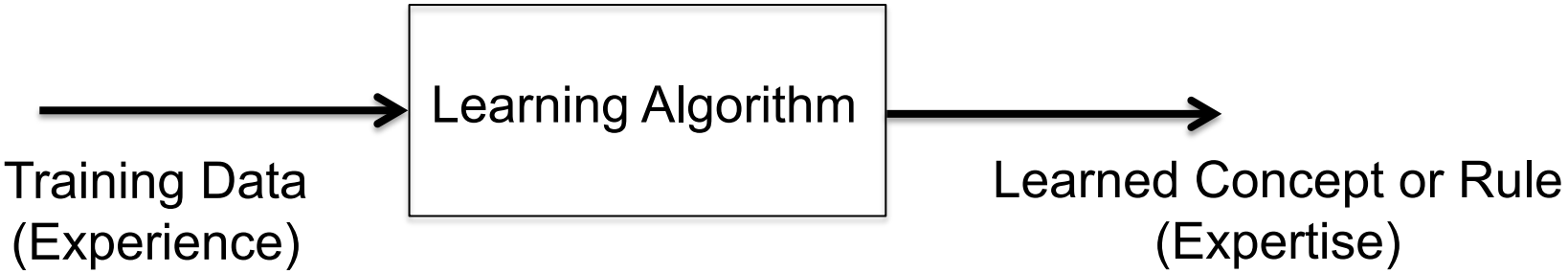


- What assumptions do we need for learning to be possible?
  - Training and test data are "similar" in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.

Example: Binary Classifiers

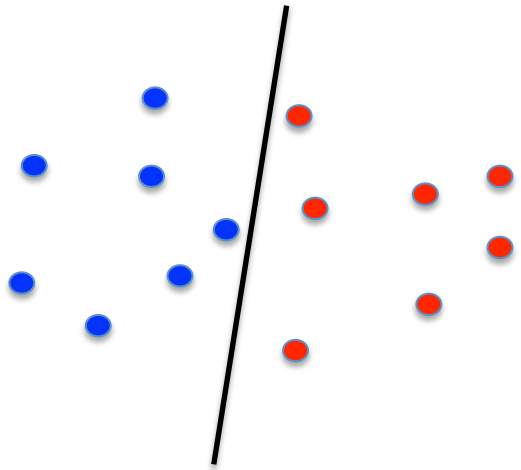


Fundamental Questions



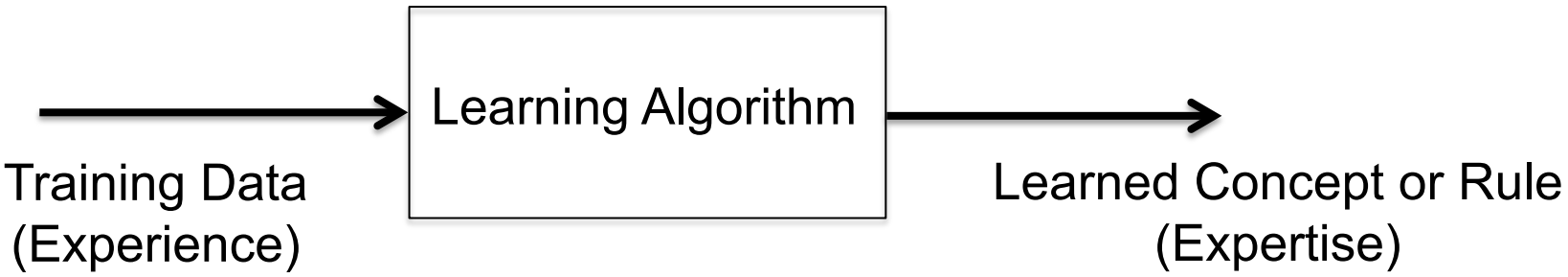
- What assumptions do we need for learning to be possible?
  - Training and test data are "similar" in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.

Example: Binary Classifiers



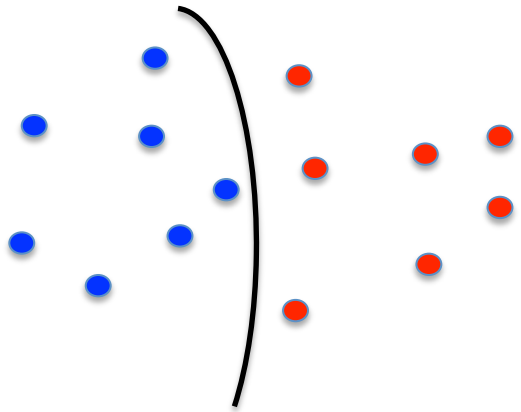
Should we consider linear classifiers?

Fundamental Questions



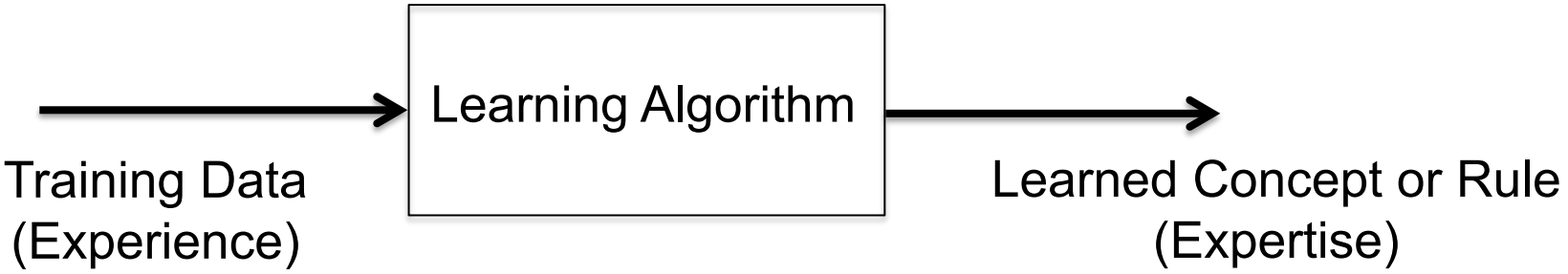
- What assumptions do we need for learning to be possible?
  - Training and test data are "similar" in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.

Example: Binary Classifiers



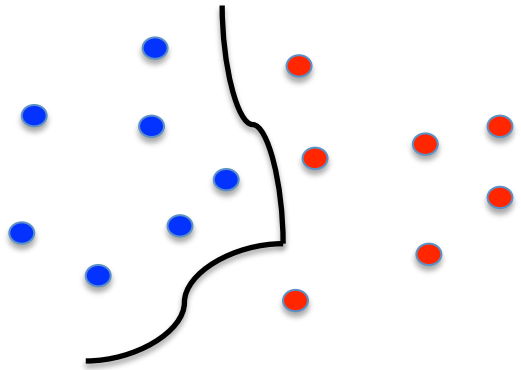
Should we consider 2nd degree poly classifiers?

Fundamental Questions



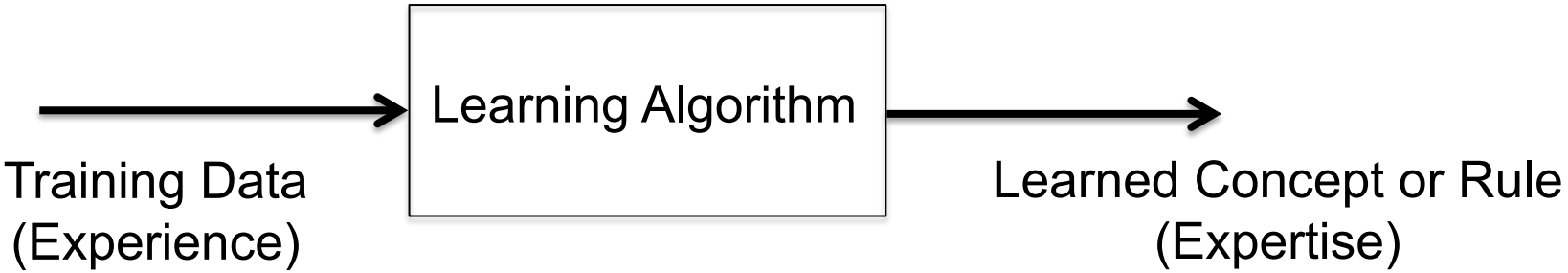
- What assumptions do we need for learning to be possible?
  - Training and test data are "similar" in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.

Example: Binary Classifiers



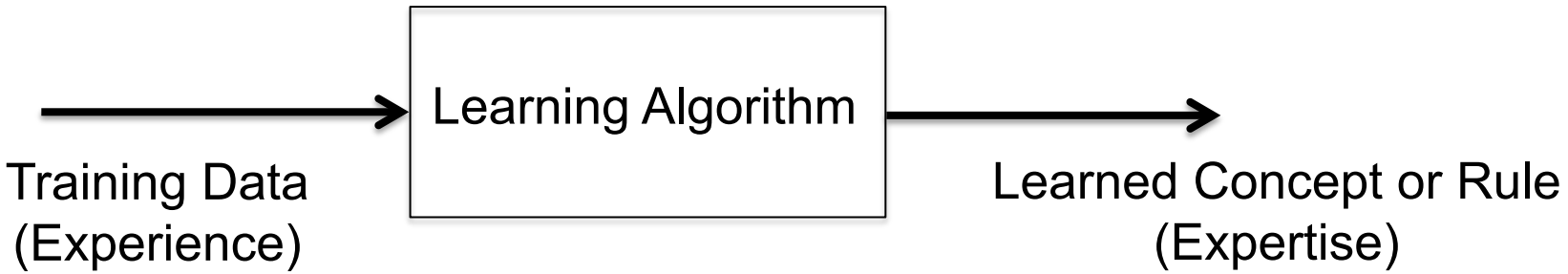
Should we consider higher degree poly or other complex functions?

Fundamental Questions



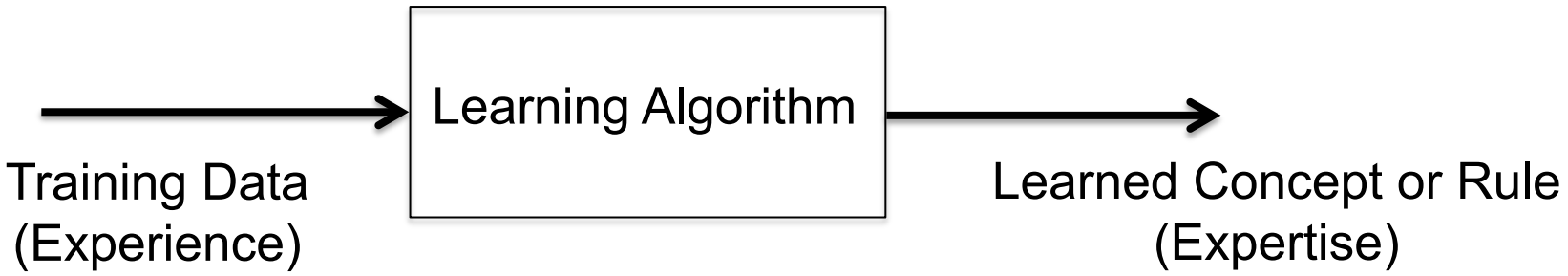
- What assumptions do we need for learning to be possible?
  - Training and test data are "similar" in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.
- Given a fixed model (e.g., a fixed type of prediction rules), how can the machine output the "right" prediction rule?

Fundamental Questions



- What assumptions do we need for learning to be possible?
  - Training and test data are "similar" in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.
- Given a fixed model (e.g., a fixed type of prediction rules), how can the machine output the "right" prediction rule?
- How many data samples are needed to ensure that the output prediction rule will generalize well to the unseen data?

Fundamental Questions



- What assumptions do we need for learning to be possible?
  - Training and test data are "similar" in some sense.
  - Put some restriction on the class of possible concepts (e.g., prediction rules) to be learned.
- Given a fixed model (e.g., a fixed type of prediction rules), how can the machine output the "right" prediction rule?
- How many data samples are needed to ensure that the output prediction rule will generalize well to the unseen data?
  - Simpler prediction rules are easier to generalize.

CSE 5523: Machine Learning Theory Fall 2017

Concentration Inequalities - Part 1

Instructor: Raef Bassily

Notation: In the sequel, we will use $[k]$ to denote the set $\{1, 2, \ldots, k\}$. For any numbers $a_1, \ldots, a_k$, we will use $\Pi_{i=1}^{k} a_i$ to denote the product $a_1 a_2 a_3 \ldots a_k$.

1.1 Things to review in basic probability:

- Recall that relationship between different events can be expressed either using set notation or logical expressions.
  - $A \subseteq B$ is equivalent to saying that event A *implies* event B (e.g., for a random variable X, the set of values for which $X \geq 2$ is a subset of the set of values for which $X \geq 0$; we can also see that the event $X \geq 2 \Rightarrow X \geq 0$). Whenever $A \subseteq B$, we have $P[A] \leq P[B]$.
  - The union of a collection of events $A_1, \ldots, A_k$, denoted as $\cup_{i=1}^{k} A_i$, is equivalent to saying that $\exists i \in [k]$ such that A_i occurs, i.e., *there is at least one event that occurs in the given collection.* In other words, the union is equivalent to "taking the logical OR" between the events in the collection.

    Recall that $P[\cup_{i=1}^{k} A_i] \leq \sum_{i=1}^{k} P[A_i]$. This is called **the union bound**. The bound is achieved with equality *iff* all the events in the collection are *disjoint.*
  - The intersection of a collection of events $A_1, \ldots, A_k$, denoted as $\cap_{i=1}^{k} A_i$, is equivalent to saying that $\forall i \in [k]$, A_i occurs, i.e., *all the events in the given collection occur simultaneously.* In other words, the intersection is equivalent to "taking the logical AND" between the events in the collection.

    Recall that $P[\cap_{i=1}^{k} A_i] = P[A_1] \cdot P[A_2|A_1] \cdot P[A_3|A_1 \cap A_2] \ldots P[A_k| \cap_{i=1}^{k-1} A_i]$, where $P[A_i \mid \cap_{j=1}^{i-1} A_j]$ denotes the conditional probability that A_i occurs given that $\cap_{j=1}^{i-1} A_j$ occurs. We will usually use ',' (i.e., commas) instead of $\cap$ to denote the intersection operator; that is, we will use $P[A_1, \ldots, A_k]$ to denote $P[\cap_{i=1}^{k} A_i]$.

    Note that when the events $A_1, \ldots, A_k$ are independent, we have $P[A_1, \ldots, A_k] = \Pi_{i=1}^{k} P[A_i]$.
- Please, review the following concepts:
  - Random variables, discrete and continuous distributions, joint distributions, conditional distributions, independence.
  - Expectation (and its properties including *linearity*), conditional expectation, variance, covariance, correlation.
  - Independent and Identically Distributed (I.I.D.) random variables.
  - Some important distributions: Bernoulli, Binomial, Guassian (both uni-variate and multi-variate).

We will discuss various concentration inequalities:

- Markov inequality
- Chebyshev inequality
- Chernoff-Hoeffding bound (special case)
- Hoeffding inequality (general case)
- McDiarmid's inequality

1.2 Markov's and Chebyshev's inequalities

Theorem 1.1 (Markov inequality)**.** Let X be a non-negative r.v. For any $\varepsilon > 0$,

$$P(X > \varepsilon) \leq \frac{E[X]}{\varepsilon},$$

assuming $E[X]$ exists.

Proof. Let $1(A)$ be the indicator function of event A. Note that we always have $X \geq \varepsilon \cdot 1(X \geq \varepsilon)$, and take the expectation of both sides:

$$E[X] \geq \varepsilon E[1(X \geq \varepsilon)] = \varepsilon P(X \geq \varepsilon).$$

□

Remark 1.2. The k^{th} moment of a r.v. X is $E[X^k]$. If the $(k+1)^{\text{th}}$ moment exists, then the k^{th} moment exists.

Definition 1.3. Variance is defined as

$$\text{var}(X) = E[(X - E[X])^2] = E[X^2] - E[X]^2.$$

Corollary 1.4 (Chebyshev's inequality)**.** Let X be a r.v. with finite variance. Then

$$P[|X - E[X]| > \varepsilon] < \frac{\text{var}(X)}{\varepsilon^2}.$$

Proof. Define $Y = |X - E[X]|^2$. Then

$$\begin{aligned} P[|X - E[X]| > \varepsilon] &= P[Y > \varepsilon^2] \\ &\leq \frac{E[Y]}{\varepsilon^2} && \text{(by Markov inequality)} \\ &= \frac{\text{var}[X]}{\varepsilon^2}. \end{aligned}$$

□

Remark 1.5. Let $X_1, \ldots, X_m$ be independent and identically distributed (i.i.d.) r.v.'s with finite variance. Recall

$$\text{var}\left[\frac{1}{m}\sum_{i=1}^{m} X_i\right] = \frac{1}{m^2}\sum_{i=1}^{m}\text{var}[X_i] = \frac{\text{var}[X_1]}{m}.$$

For any $\varepsilon > 0$,

$$P\left[\left|\frac{1}{m}\sum_{i=1}^{m} X_i - \mu\right| > \varepsilon\right] \leq \frac{\sigma^2}{m\varepsilon^2}$$

where $\mu = E[X_1]$ and $\sigma^2 = \text{var}[X_1]$.

Exercise. Let $X_1, \ldots, X_m$ be i.i.d. drawn from $[0, 1]$ where $E[X_1] = 1/3$. How likely is it to have $\frac{1}{m}\sum_{i=1}^{m} X_i > 2/3$?

1.3 Chernoff-Hoeffding Bound: A special (simple) case

Theorem 1.6 (Chernoff-Hoeffding bound: special case for Bernoulli r.v.'s)**.** Let $X_1, \ldots, X_m$ be independent Bernoulli r.v.'s where $P[X_i = 1] = p_i$. Let

$$\overline{X} = \frac{1}{m}\sum_{i=1}^{m} X_i,$$
$$p = \frac{1}{m}\sum_{i=1}^{m} p_i.$$

Then

$$P[\overline{X} > p + \varepsilon] \leq e^{-2\varepsilon^2 m} \qquad \text{and} \qquad P[\overline{X} < p - \varepsilon] \leq e^{-2\varepsilon^2 m}$$

Hence, $P\left[\ |\overline{X} - p| > \varepsilon\ \right] \leq 2e^{-2\varepsilon^2 m}$.

Remark 1.7. The weak law of large numbers says that the sample mean of n independent r.v.'s converges to the "true mean" in probability, but it doesn't say how fast. These concentration inequalities give us more information about the speed of convergence.

1.3.1 Moment Generating Functions: A preliminary tool for proving Theorem 1.6

Definition 1.8. The moment generating function (MGF) of a r.v. X is $M_X(t) = E[e^{tX}]$ for $t \in \mathbb{R}$.

Fact 1.9. If $M_X(t)$ is defined for $t \in (-t^*, t^*)$ for some $t^* > 0$, then:

1. All the moments of X exist, and all derivatives of $M_X(t)$ at $t = 0$ exist, and for all k, we have

$$E[X^k] = \left.\frac{\partial^k M}{\partial t^k}\right|_{t=0}.$$

2. For all $t \in (-t^*, t^*)$, $M(t)$ has Taylor expansion

$$M(t) = \sum_{k=0}^{\infty} \frac{E[X^k]}{k!} t^k.$$

Fact 1.10. If X, Y are independent r.v.'s, then

$$M_{X+Y}(t) = M_X(t)M_Y(t).$$

(For proof, use the fact that $E[XY] = E[X]E[Y]$ for independent variables.)

Fact 1.11. For any constants a and b,

$$M_{aX+b}(t) = e^{bt}M_X(at).$$

1.3.2 Proof of Chernoff-Hoeffding's bound (the special case)

Proof of Theorem 1.6. Define $Y_i = X_i - p_i$ (with zero mean), and $\overline{Y} = \frac{1}{m}\sum_{i=1}^{m} Y_i$.

We want to show that $P(|\overline{Y}| > \varepsilon) \leq 2e^{-2\varepsilon^2 m}$. We will consider the one-sided probabilities $P[\overline{Y} > \varepsilon]$ and $P[-\overline{Y} > \varepsilon]$ separately, then later combine them via the union bound.

For $t > 0$,

$$\begin{aligned} P[\overline{Y} > \varepsilon] &= P[e^{t\overline{Y}} > e^{t\varepsilon}] \\ &\leq \frac{E[e^{t\overline{Y}}]}{e^{t\varepsilon}} && \text{(Markov)} \\ &= \frac{M_{\overline{Y}}(t)}{e^{t\varepsilon}}. \end{aligned}$$

We use the following lemma to complete the proof.

Lemma 1.12. For all $t > 0$, $M_{Y_i}(t) \leq e^{t^2/8}$.

Proof. Recall that

$$Y_i = \begin{cases} 1 - p_i & \text{with probability } p_i, \\ -p_i & \text{with probability } 1 - p_i. \end{cases}$$

Then,

$$\begin{aligned} M_{Y_i}(t) &= E[e^{ty_i}] \\ &= p_i e^{t(1-p_i)} + (1-p_i)e^{-tp_i} \\ &= \exp[\underbrace{-tp_i + \ln(p_i e^t + (1-p_i))}_{\triangleq\, g(t)}] \end{aligned}$$

Taylor-expand $g(t)$ to get

$$g(0) + g'(0)t + g''(t^*)\frac{t^2}{2!}$$

for some $t^* \in [0, t]$. The first two terms must be zero, and we can show that $g''(t^*) \leq 1/4$ for all possible t^*, which finishes the proof of the lemma. □

Proof of Theorem 1.6 (continued). Using properties of MGF's (Facts 1.10 and 1.11) together with the previous lemma, we have

$$\begin{aligned} M_{\overline{Y}}(t) &= \prod_{i=1}^{m} M_{Y_i}(t/m) \\ &\leq \prod_{i=1}^{m} e^{t^2/8m^2} \\ &= e^{t^2/8m}. \end{aligned}$$

Next, note that

$$\begin{aligned} P[\overline{Y} > \varepsilon] &\leq \frac{M_{\overline{Y}(t)}}{e^{t\varepsilon}} \\ &\leq \frac{e^{t^2/8m}}{e^{t\varepsilon}} \\ &= e^{-(t\varepsilon - t^2/8m)}. \end{aligned}$$

Since t was an arbitrary positive real number, we are free to select a value of t that yields the tightest bound. One can easily verify that such value is $t = 4m\varepsilon$, for which

$$P[\overline{Y} > \varepsilon] \leq e^{-2m\varepsilon^2}.$$

We can derive a similar bound for the other tail, and we can use a union bound to combine them and complete the proof. □

CSE 5523: Machine Learning Theory Fall 2017

2- Concentration Inequalities (cont'd)

Instructor: Raef Bassily

2.1 General Chernoff-Hoeffding's Bound

Theorem 2.1 (Chernoff-Hoeffding: the general case)**.** Let $X_1, \ldots, X_m$ be a sequence of r.v.'s such that for all i, $a_i \leq X_i \leq b_i$ with probability 1 (often written "w.p. 1"). Let $\overline{X} = \frac{1}{m}\sum_{i=1}^{m} X_i$. For any $\varepsilon > 0$,

$$P[\overline{X} - E[\overline{X}] > \varepsilon] \leq \exp\left(\frac{-2\varepsilon^2 m^2}{\sum_{i=1}^{m}(a_i - b_i)^2}\right), \text{ and}$$
$$P[\overline{X} - E[\overline{X}] < -\varepsilon] \leq \exp\left(\frac{-2\varepsilon^2 m^2}{\sum_{i=1}^{m}(a_i - b_i)^2}\right).$$

To prove the general bound, we will use the following important result on the expectation of a convex function of a random variable (known as *Jensen's inequality*).

Definition 2.2 (Convex functions)**.** A function $f : \mathbb{R} \to \mathbb{R}$ is convex if for any $0 \leq \lambda \leq 1$, and any $x, y \in \mathbb{R}$, we have

$$f\left(\lambda x + (1-\lambda)\, y\right) \leq \lambda f(x) + (1-\lambda) f(y).$$

Theorem 2.3 (Jensen's inequality)**.** Let X be a r.v. and let f be a convex function. Then

$$f(E[X]) \leq E[f(X)].$$

2.1.1 Proof of Theorem 2.1

Proof technique: Use Chernoff's generic method of bounding the probability tail using MGF's.

Lemma 2.4 (Hoeffding's lemma)**.** Let $Y \in [\alpha, \beta]$ w.p. 1 and $E[Y] = 0$. Then

$$M_Y(t) = E[e^{tY}] \leq e^{t^2(\beta-\alpha)^2/8}.$$

Proof. Since $f(y) = e^{ty}$ is convex, we know for all $\lambda \in [0,1]$ and all $y \in [\alpha, \beta]$ s.t. $y = \lambda\alpha + (1-\lambda)\beta$,

$$f(y) \leq \lambda f(\alpha) + (1-\lambda) f(\beta).$$

That is, for $\lambda = \frac{\beta - Y}{\beta - \alpha}$, we have

$$e^{tY} \leq \frac{\beta - Y}{\beta - \alpha} e^{t\alpha} + \frac{Y - \alpha}{\beta - \alpha} e^{t\beta}.$$

Taking the expectation of both sides and using the assumption that $E[Y] = 0$,

$$\begin{aligned} M_Y(t) = E[e^{tY}] &\leq E\left[\frac{\beta - Y}{\beta - \alpha}\right] e^{t\alpha} + E\left[\frac{Y - \alpha}{\beta - \alpha}\right] e^{t\beta} \\ &= E[Y] \cdot \left(\frac{e^{t\beta} - e^{t\alpha}}{\beta - \alpha}\right) + \frac{\beta}{\beta - \alpha} e^{t\alpha} - \frac{\alpha}{\beta - \alpha} e^{t\beta} \\ &= \frac{\beta}{\beta - \alpha} e^{t\alpha} - \frac{\alpha}{\beta - \alpha} e^{t\beta}. \end{aligned}$$

Then, we can rearrange the right-hand side to obtain

$$\begin{aligned} M_Y(t) &\le e^{t\alpha}\left(1+\frac{\alpha}{\beta-\alpha}\right)-\frac{\alpha}{\beta-\alpha}e^{t\beta} \\ &= \exp\left(t(\beta-\alpha)\frac{\alpha}{\beta-\alpha}\right)\cdot\left(1+\frac{\alpha}{\beta-\alpha}-\frac{\alpha}{\beta-\alpha}\exp(t(\beta-\alpha))\right). \end{aligned}$$

Defining $\tau = t(\beta-\alpha)$ and $\theta = \frac{\alpha}{\beta-\alpha}$,

$$\begin{aligned} M_Y(t) &\le e^{\tau\theta}(1+\theta-\theta e^{\tau}) \\ &= \exp(\underbrace{\tau\theta+\ln(1+\theta-\theta e^{\tau})}_{g(\tau)}) \\ &\le e^{\tau^2/8} \le e^{t^2(\beta-\alpha)^2/8}. \end{aligned}$$

The final inequality comes from a similar argument used in the proof of the Chernoff-Hoeffding bound; specifically, we can Taylor-expand $g(\tau)$ and bound the approximation error. □

Outline of proof of Hoeffding's inequality. First, let $Y_i = X_i - E[X_i]$ (bounded, with zero mean), and let $\overline{Y} = \frac{1}{m}\sum_{i=1}^m Y_i$. Then, use properties of MGF's together with Lemma 2.4 to bound $M_{\overline{Y}}(t)$. Next, use Markov inequality to get a bound on $P[\overline{Y} > \varepsilon] \le M_{\overline{Y}}(t)/e^{t\varepsilon}$. Finally, we pick the value of t that yields the tightest bound. □

Theorem 2.5 (McDiarmid's inequality)**.** Let $f : \mathbb{R}^m \to \mathbb{R}$ be a function with the following property: for all $i \in \{1,\dots,m\}$, there exists some $\Delta_i > 0$ such that for any $y \in \mathbb{R}$ and any $(x_1,\dots,x_m) \in \mathbb{R}^m$,

$$|f(x_1,\dots,x_m) - f(x_1,\dots,x_{i-1},y,x_{i+1},\dots,x_m)| \le \Delta_i.$$

(This is called "bounded sensitivity" or "bounded differences.") Let $X_1,\dots,X_m$ be independent r.v.'s. Then

$$\begin{aligned} P[f(X_1,\dots,X_m) - E[f(X_1,\dots,X_m)] > \varepsilon] &\le \exp\left(\frac{-2\varepsilon^2}{\sum_{i=1}^m \Delta_i^2}\right), \text{ and} \\ P[f(X_1,\dots,X_m) - E[f(X_1,\dots,X_m)] < -\varepsilon] &\le \exp\left(\frac{-2\varepsilon^2}{\sum_{i=1}^m \Delta_i^2}\right). \end{aligned}$$

Remark 2.6. Note that the Hoeffding inequality is a special case of McDiarmid's inequality, with $f(X_1,\dots,X_m) = \frac{1}{m}\sum_{i=1}^m (X_i - E[X_i]) \cdot \mathbf{1}(a_i \le X_i \le b_i)$. This works because Hoeffding requires the X_i to be bounded.

Definition 2.7. A function $f : \mathbb{R}^m \to \mathbb{R}$ is said to be λ-Lipschitz with respect to the L_p metric if for all $x, y \in \mathbb{R}^m$,

$$|f(x) - f(y)| \le \lambda \|x - y\|_p.$$

Note: In the one-dimensional case, this means the derivative is bounded. This will be useful later for analyzing gradient descent.

Corollary 2.8. Let $p > 0$ and suppose $f : \mathbb{R}^m \to \mathbb{R}$ is λ-Lipschitz w.r.t. L_p. Let $X_1,\dots,X_m$ be independent r.v.'s, bounded such that $a_i \le X_i \le b_i$ with probability 1. Then

$$P[|f(X_1,\dots,X_m) - E[f(X_1,\dots,X_m)]| > \varepsilon] \le 2\exp\left(\frac{-2\varepsilon^2}{\lambda^2 \sum_{i=1}^m |b_i - a_i|^2}\right).$$

CSE 5523: Machine Learning Theory Fall 2017

3- A Framework for Statistical Learning

Instructor: Raef Bassily

3.1 Learning Paradigm

To help understand the essential principles of statistical learning, we will focus in **the first part** of this course on a more basic framework for statistical learning. In particular, we will **focus on**

- **Binary classification:** Each data point is associated with a binary label (e.g., $\{0, 1\}$ or $\{-1, +1\}$). The goal is to predict the hidden label of any new data point.
- **Supervised learning:** The learning algorithm is given a *training data set with all labels included.*
- **Passive learning:** *All labels* for the training data are given beforehand, i.e., before training starts (as opposed to active learning, where labels are either partially or entirely absent and the learning algorithm has to actively interact with environment during the training phase, e.g., by querying the labels of some particular data points).
- **Proper learning:** The output of the learning algorithm is a predictor (a.k.a. hypothesis) from some *pre-specified* class of predictors.
- **Batch (Offline) learning:** The learning algorithm outputs a predictor only after seeing the entire training data set. That is, the learning algorithm does not have to respond to the environment in an online fashion throughout the learning process.

3.2 A Formal Model

3.2.1 Model components and main assumptions

The basic setup for statistical learning:

$$\text{Inputs} \rightarrow \text{Learner } A \rightarrow \text{Outputs}$$

1. Inputs to the learning algorithm A
   - Domain set $\mathcal{X}$ ("space of feature vectors" a.k.a *instance space*): An arbitrary set, which represents the set of objects (or, features) that we may wish to label.
   - Label set $\mathcal{Y}$ (for binary classification, usually $\{0, 1\}$ or $\{+1, -1\}$)
   - Training data (a.k.a. "training sample") of size n:

$$S = \{(x_1, y_1), \ldots, (x_n, y_n)\}, \text{ with } x_i \in \mathcal{X}, y_i \in \mathcal{Y}$$

2. Output of A
   - A outputs a prediction rule (a.k.a. hypothesis or classifier) $h : \mathcal{X} \rightarrow \mathcal{Y}$

- Note: h implicitly depends on S, and S is often a random set. We may sometimes write h as $A(S)$ or h_S.
- Hypothesis class (a.k.a. *concept class*): Usually we will assume classifiers (a.k.a. hypotheses) are chosen from some fixed family of classifiers (which will usually be denoted as $\mathcal{H}$). This family is called the hypothesis class. (This restriction is often known as "induction bias.") For example, we might choose $\mathcal{H}$ = linear classifiers or $\mathcal{H}$ = decision trees.

3. Structure of data (data generation assumptions)

   - We will assume each data point is drawn i.i.d. from an underlying distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, called the data distribution. Note that the data distribution $\mathcal{D}$ is a *joint distribution* for the pair of random variables (x, y) where x denotes the domain element and y denotes the label associated with x.
   - We write $\mathcal{D}^n$ to denote the distribution of n **i.i.d.** training examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$.
   - The distribution $\mathcal{D}$ is unknown to the learner, which only has access to examples drawn from it as in the case of training and test data.
   - **Special case:** Sometimes, we will assume that there is a true "labeling function" (or "target concept") $f : \mathcal{X} \to \mathcal{Y}$; that is, the label y is given by a deterministic function of the domain point x. In this case, we use $\mathcal{D}$ to denote the distribution over only the domain set $\mathcal{X}$. (The conditional distribution of $y \,|\, x$ is a degenerate distribution.) Also, in this case, we will use $(\mathcal{D}, f)^n$ to denote the distribution of n **i.i.d.** training examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ where, for all $i \in [n]$, $x_i \sim \mathcal{D}$ and $y_i = f(x_i)$.

3.2.2 Measure of error

First, we define a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, which represents the cost of outputting a certain label w.r.t. the true label. Then there are two main types of error related to this loss function:

1. Prediction error (a.k.a classification error, true error, risk)

   A quantity denoted by $\mathrm{err}(h; \mathcal{D})$. This captures the expected error incurred by a predictor h on a **new** random data point from $\mathcal{D}$.

   Definition 3.1. Given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$,

   $$\mathrm{err}(h; \mathcal{D}) = E_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$$

   Sometimes, we will make the following assumption:

   Realizability assumption: There is a true unknown labeling function $f : \mathcal{X} \to \mathcal{Y}$ that belongs to the given hypothesis class $\mathcal{H}$. More precisely, **realizability** implies that for any distribution $\mathcal{D}$ over the domain $\mathcal{X}$, there is some $h^* \in \mathcal{H}$ such that $\mathrm{err}(h^*; \mathcal{D}) = 0$.

2. Training error (a.k.a. empirical error, empirical risk)

   Denoted by $\widehat{\mathrm{err}}(h; S)$, this captures the error incurred by a predictor h on the training data S. Given a loss function ℓ,

   $$\widehat{\mathrm{err}}(h; S) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$

Remark 3.2. If h is a function of the random training sample $S \sim \mathcal{D}^n$, then h is a random variable. **In such case, not only the empirical error is a random quantity, but also the true error** since they both depend on the random training set S either explicitly (as it is the case with the empirical error) or implicitly via the dependency on h (as it is the case with both the true error and empirical error).

Example 3.3. One common loss function is binary loss (or 0-1 loss):

$$\begin{aligned}
\ell(h(x), y) &= 1(h(x) \neq y), \\
\mathrm{err}(h; \mathcal{D}) &= P_{(x,y)\sim\mathcal{D}}(h(x) \neq y), \\
\widehat{\mathrm{err}}(h; \mathcal{D}) &= \frac{1}{n}\sum_{i=1}^{n} 1(h(x_i) \neq y_i).
\end{aligned}$$

Note: In the first part of the course where we focus on binary classification problems, the loss function will be the binary loss function. Hence, $\mathrm{err}(h; \mathcal{D})$ will refer to $P_{(x,y)\sim\mathcal{D}}(h(x) \neq y)$.

Empirical Risk Minimization (ERM): Often we want to find the "best" predictor $\hat{h}_S$ in a given hypothesis class $\mathcal{H}$ after observing a training data set S. Generally speaking, this is done by solving the following optimization problem:

$$\hat{h}_S \in \arg\min_{h\in\mathcal{H}} \widehat{\mathrm{err}}(h; S)$$

where $\arg\min_{h\in\mathcal{H}} \widehat{\mathrm{err}}(h; S)$ refers to the set of hypotheses in $\mathcal{H}$ that minimize the empirical risk $\widehat{\mathrm{err}}(h; S)$. (Note that the minimizer of the empirical risk *need not* be unique; hence, $\arg\min_{h\in\mathcal{H}} \widehat{\mathrm{err}}(h; S)$ is viewed as a set in general.)

CSE 5523: Machine Learning Theory Fall 2017

4- Probably Approximately Correct (PAC) Learning - Part 1

Instructor: Raef Bassily

4.1 Probably Approximately Correct Learning

Definition 4.1 (**PAC Learning**). A hypothesis class $\mathcal{H}$ is PAC-learnable if *there exist* **a function**

$$n_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$$

and **an algorithm** A such that:

- *for any* $0 < \varepsilon, \delta < 1$,
- *for any distribution* $\mathcal{D}$ over $\mathcal{X}$, and
- *for any labeling function* $f : \mathcal{X} \to \mathcal{Y}$ that lies in $\mathcal{H}$,

if A is given a training set S of $n \geq n_{\mathcal{H}}(\varepsilon, \delta)$ examples from $\mathcal{D}$ labeled by f, **then** w.p. $\geq 1 - \delta$ (the probability is w.r.t. the randomness in S), A outputs $h_S \in \mathcal{H}$ for which

$$\mathrm{err}(h_S; (\mathcal{D}, f)) \leq \varepsilon$$

where $\mathrm{err}(h_S; (\mathcal{D}, f)) \triangleq \mathbb{P}_{x \sim \mathcal{D}}[h_S(x) \neq f(x)]$ (where the probability here is taken w.r.t. a fresh $x \sim \mathcal{D}$ that is independent of S).

Remark 4.2. The definition entails two parameters:

- ε is the "accuracy", representing how far the output hypothesis can be from the optimal (this is where the "approximately correct" in "PAC" comes from)
- $1-\delta$ is the "confidence", representing how likely the output hypothesis will achieve ε accuracy (this is where the "probably" in "PAC" comes from)

Both approximations are *unavoidable* for the strong results we'll see later in this course. We need ε because our only information about the unknown distribution comes from finite training examples. We need δ because there is a nonzero probability of getting a mis-representative training sample.

Remark 4.3. Note that Definition 4.1 entails the realizability assumption discussed earlier; that is, domain points are labeled by an unknown deterministic labeling function $f : \mathcal{X} \to \mathcal{Y}$ that lies in the given class $\mathcal{H}$. Note the *quantifiers* (i.e., *for all* and *there exists*) in the above definition and their order imply that the definition requires the **existence of one** algorithm with finite sample size function $n_{\mathcal{H}}$ that "works" **for any** distribution over $\mathcal{X}$ and **any** true labeling function $f \in \mathcal{H}$; that is, one algorithm that "works" regardless of the nature of the domain distribution or the true labeling function (given that the labeling function belongs to the pre-specified hypothesis class $\mathcal{H}$).

Agnostic PAC Learning: In a more general "agnostic PAC" model, we remove the assumption of the existence of a true deterministic labeling function (and hence remove the realizability assumption). Instead, the unknown distribution $\mathcal{D}$ is a distribution defined over $\mathcal{X} \times \mathcal{Y}$ (i.e., $\mathcal{D}$ is a joint distribution for the pair (x, y)). The definition is basically the same, except we require

$$\text{err}(h_S; \mathcal{D}) \leq \varepsilon + \inf_{\tilde{h} \in \mathcal{H}} \text{err}(\tilde{h}; \mathcal{D}).$$

Note this definition provides a competitive guarantee where we compare the prediction error of the output hypothesis to *the best possible prediction error that can be achieved by a hypothesis within the given class* $\mathcal{H}$.

Definition 4.4 (**Agnostic PAC Learning**)**.** A hypothesis class $\mathcal{H}$ is agnostic PAC-learnable if *there exist* **a function**

$$n_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$$

and **an algorithm** A such that:

- *for any* $0 < \varepsilon, \delta < 1$, and
- *for any distribution* $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$,

if A is given a training set S of $n \geq n_{\mathcal{H}}(\varepsilon, \delta)$ labeled examples from $\mathcal{D}$, **then** w.p. $\geq 1 - \delta$ (the probability is w.r.t. the randomness in S), A outputs $h_S \in \mathcal{H}$ for which

$$\text{err}(h_S; \mathcal{D}) \leq \varepsilon + \inf_{\tilde{h} \in \mathcal{H}} \text{err}(\tilde{h}; \mathcal{D})$$

where $\text{err}(h_S; \mathcal{D}) \triangleq \mathbb{P}_{(x,y)\sim\mathcal{D}}[h_S(x) \neq y]$ (where the probability here is taken w.r.t. a fresh $(x, y) \sim \mathcal{D}$ that is independent of S).

Example 4.5 (Axis-aligned rectangles)**.** Suppose we have

$$\begin{aligned}
&\mathcal{X} = \mathbb{R}^2, \\
&\mathcal{Y} = \{-, +\}, \\
&\mathcal{H} = \text{classifiers that output } + \text{ for points inside some axis-aligned rectangle,}
\end{aligned}$$

Let $\mathcal{D}$ denote the unknown domain distribution over $\mathcal{X}$, and *assume that realizability holds* (that is, there is some unknown rectangle $f \in \mathcal{H}$ that generates the true labels for the domain points.)

Suppose that, given a set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of training examples, our learner A picks the "tightest" rectangle that encloses all of the positive examples. Let h_S denote such rectangle. Note that h_S will never make an error on a domain point with a negative label whether it's in the training set or not (since, by construction, h_S always lies inside f.)

Next, we'll show that such algorithm A is indeed a PAC-learning algorithm for the class $\mathcal{H}$ described above. In the light of Definition 4.1, this proves that $\mathcal{H}$ is PAC-learnable.

Accuracy/confidence analysis for axis-aligned rectangles: First, fix $0 < \varepsilon$ and $0 < \delta < 1$.

What is the true error? It depends on the region between the rectangle h_S and the true rectangle f. Let (x, y) denote a fresh example from $(\mathcal{D}, f)$ (that is independent of the training set S). We have:

$$\begin{aligned}\operatorname{err}(h_S; \mathcal{D}) &= \mathbb{P}_{(x,y)\sim(\mathcal{D},f)} \ [h_S \text{ misclassifies } x] \\ &= \mathbb{P}_{(x,y)\sim(\mathcal{D},f)} \ [h_S(x) \neq y, y = f(x)] \\ &= \mathbb{P}_{(x,y)\sim(\mathcal{D},f)} \ [x \text{ lies between } h_S \text{ and } f] \,.\end{aligned}$$

Note that the training set is random, and therefore the region between h_S and f is also random.

Now, our goal is to bound

$$\mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [\operatorname{err}(h_S; (\mathcal{D}, f)) > \varepsilon] \,.$$

For simplicity, we will assume that the distribution $\mathcal{D}$ is continuous. It is important to note that the guarantees discussed in the sequel can be shown to hold for *any* distribution $\mathcal{D}$ (not necessarily continuous). However, in our treatment given here, we will assume that $\mathcal{D}$ is continuous to maintain clarity of the main proof ideas and make the argument more intuitive.

Consider four strips at the left, right, top, and bottom of f, and aim to get each of their measures equal to $\varepsilon/4$. Call these regions $R_1, \ldots, R_4$.

Suppose

$$\operatorname{err}(h_S; \mathcal{D}) > \varepsilon.$$

Then h_S must "miss" (i.e., must fail to overlap with) at least one of the strips $R_1, \ldots, R_4$. Therefore,

$$\begin{aligned}\mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [\operatorname{err}(h_S; \mathcal{D}) > \varepsilon] &\leq \mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [h_S \text{ misses at least one strip}] \\ &\leq \sum_{i=1}^{4} \mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [h_S \text{ misses } R_i] \\ &= 4\, \mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [h_S \text{ misses } R_1] \,.\end{aligned}$$

Furthermore,

$$\begin{aligned}\mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [h_S \text{ misses } R_1] &= \mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [\text{no point of } S \text{ lies in } R_1] \\ &= \left(1 - \frac{\varepsilon}{4}\right)^n \\ &\leq e^{-\varepsilon n/4}.\end{aligned}$$

The last inequality arises because $1 - x \leq e^{-x}$ for all x. Then, for $n \geq \frac{4}{\varepsilon} \ln\left(\frac{4}{\delta}\right)$, we have

$$\mathbb{P}_{S\sim(\mathcal{D},f)^n} \ [\operatorname{err}(h_S; \mathcal{D}) > \varepsilon] \leq \delta.$$

Hence, we can choose $n_{\mathcal{H}}(\varepsilon, \delta)$ in Definition 4.1 to be $\frac{4}{\varepsilon} \ln\left(\frac{4}{\delta}\right)$.

CSE 5523: Machine Learning Theory Fall 2017

5- PAC Learning (Part 2)

Instructor: Raef Bassily

5.1 Examples of PAC-Learnable Classes (cont'd)

Example 5.1 (Learning monotone conjunctions)**.** Consider the following scenario:

$$\mathcal{X} = \{0,1\}^d,$$
$$\mathcal{Y} = \{0,1\}, \text{ and}$$
$$\mathcal{H} = \text{conjunctions of positive literals.}$$

Each string $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_d)$ is considered an assignment of d bits to the variables $x_1, \ldots, x_d$.

A conjunction is a subset of these d variables (either positive or negated) combined via the logical AND operator. A monotone conjunction is a conjunction with no negated literals.

Note that $|\mathcal{H}| = 2^d$. This is smaller than the set of all possible conjunctions (which has size 3^d). Also, we can represent each $h \in \mathcal{H}$ using the subset of variables that appear as literals in the conjunction.

More formally, for a given $\tilde{x} \in \mathcal{X}$, we write $h(\tilde{x}) = 1$ iff for all i such that $x_i \in h$, we have $\tilde{x}_i = 1$.

Let's assume realizability. That is, assume there exists $h^* \in \mathcal{H}$ that makes no error on any $(\tilde{x}, \tilde{y})$ generated from the underlying distribution.

Consider the following algorithm: Ignore any examples labeled 0. For all remaining examples with label 1, delete any variable that ever gets set to 0. After processing all such examples, output the remaining set of variables. (This output represents the hypothesized monotone conjunction.)

In this case, $h_S \supseteq h^*$. Also, for any $\tilde{x}$, if $h^*(\tilde{x}) = 0$, then $h_S(\tilde{x}) = 0$. Therefore, h_S may make errors only on new examples with label 1. It will get all 0-labeled examples correct.

Accuracy and confidence analysis: Note that h_S may contain "extra" literals relative to h^*. Let x_j be an extra literal in h_S. If we are given a new example $(\tilde{x}, \tilde{y})$, then this extra literal would cause an error if $\tilde{x}_j = 0$ and $\tilde{y} = 1$.

For all $j \in [d] = \{1, \ldots, d\}$, let $q_j = P_{(\tilde{x},\tilde{y})\sim\mathcal{D}}(\tilde{x}_j = 0, \tilde{y} = 1)$.

Note that

$$\text{err}(h_S; \mathcal{D}) \leq \sum_{j: x_j \in h_S \setminus h^*} q_j.$$

Therefore $\text{err}(h_S; \mathcal{D}) > \varepsilon$ implies there exists $x_j \in h_S \setminus h^*$ such that $q_j > \varepsilon/d$.

We call a literal x_j "bad" if $q_j > \varepsilon/d$. For any bad literal x_j, we have

$$\begin{aligned}
&P(\text{the bad literal } x_j \in h_S \setminus h^*) \\
&\quad = P(S \text{ does not contain any } (\tilde{x}, \tilde{y}) \text{ with } \tilde{x}_j = 0 \text{ and } \tilde{y} = 1) \\
&\quad = (1 - q_j)^n \\
&\quad \leq (1 - \varepsilon/d)^n \\
&\quad \leq e^{-\varepsilon n/d}.
\end{aligned}$$

Then, for sufficiently large n, we can achieve the following bound:

$$\begin{aligned} P(\text{err}(h_S; D) > \varepsilon) &\leq \sum_{j=1}^{d} P(\text{bad literal } x_j \in h_S \setminus h^*) \\ &\leq de^{-\varepsilon n/d} \\ &\leq \delta. \end{aligned}$$

This implies $n_{\mathcal{H}}(\varepsilon, \delta) = \frac{d}{\varepsilon} \ln(d/\delta)$.

Example 5.2 (Learning conjunctions; not restricted to monotone)**.** There is a simple reduction: note that a conjunction over boolean variables $x_1, \ldots, x_d$ can be written as a monotone (i.e., positive) conjunction over $x_1, \ldots, x_d, \overline{x}_1, \ldots, \overline{x}_d$.

Using the result from last time:

$$n_{\mathcal{H}_{+\text{conj}}}(\varepsilon, \delta) = \frac{d}{\varepsilon} \log\left(\frac{d}{\delta}\right) \implies n_{\mathcal{H}_{\text{conj}}}(\varepsilon, \delta) = \frac{2d}{\varepsilon} \log\left(\frac{2d}{\delta}\right)$$

As an exercise, try to describe the algorithm for this general case.

5.2 Occam's razor

So far, we have seen examples where the algorithm, given a training set S (and assuming realizability), outputs h_S which is *consistent* with S. That is, it does not make any error on S:

$$\widehat{\text{err}}(h_S; S) = 0.$$

Hence, we could get an upper bound on the sample complexity in each of these examples.

Can we get a more general bound on the sample complexity of *any* hypothesis class without devising a specific algorithm?

The answer is unfortunately "no" in general; this might not be possible if the hypothesis class is overly complex. However, we will see that there is a general condition of learnability for any finite hypothesis class.

Theorem 5.3 (Occam's Razor)**.** Let $\mathcal{H}$ be any **finite** hypothesis class. Let $\mathcal{D}$ be any distribution over the domain $\mathcal{X}$. Suppose that realizability holds, and let $f \in \mathcal{H}$ be any function that generates the true labels for domain points drawn from $\mathcal{D}$. Let $0 < \varepsilon, \delta < 1$. Given a training set S of n i.i.d. examples drawn from $(\mathcal{D}, f)$, let h_S be any hypothesis in $\mathcal{H}$ that is **consistent** with S. If $n \geq \frac{1}{\varepsilon} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)$, then w.p. $\geq 1 - \delta$, we must have $\text{err}\left(h_S; (\mathcal{D}, f)\right) \leq \varepsilon$.

Proof. Let $\mathcal{H}_{\text{bad}} = \{h \in \mathcal{H} : \text{err}(h; \mathcal{D}) > \varepsilon\}$. Supposing h_S satisfies the premise in the theorem (i.e., it is consistent with S), we have

$$\begin{aligned} P_S(\text{err}(h_S; \mathcal{D}) > \varepsilon) &= P(h_S \in \mathcal{H}_{\text{bad}}) \\ &= P(\exists\, h \in \mathcal{H}_{\text{bad}} : h_S = h) \\ &= P(\exists\, h \in \mathcal{H}_{\text{bad}} : h \text{ is consistent with } S, h_S = h) \\ &\leq P(\exists\, h \in \mathcal{H}_{\text{bad}} : h \text{ is consistent with } S) \\ &\leq \sum_{h \in \mathcal{H}_{\text{bad}}} P(h \text{ is consistent with } S). \qquad \text{(via union bound)} \end{aligned}$$

Now, by the definition of $\mathcal{H}_{\text{bad}}$ each term in this sum is upper-bounded by $(1-\varepsilon)^n$. Hence

$$\begin{aligned} P(\text{err}(h_S;\mathcal{D}) > \varepsilon) &\leq |\mathcal{H}_{\text{bad}}|\,(1-\varepsilon)^n \\ &\leq |\mathcal{H}|\,(1-\varepsilon)^n \\ &\leq |\mathcal{H}|\,e^{-\varepsilon n}. \end{aligned}$$

Then, the probability of obtaining a large error has the desired bound if we set n such that $\delta \geq |\mathcal{H}|e^{-\varepsilon n}$, or equivalently,

$$n \geq \frac{1}{\varepsilon}\ln\left(\frac{|\mathcal{H}|}{\delta}\right).$$

□

Implications of Occam's Razor: For a finite hypothesis class, assuming realizability, a smaller $\mathcal{H}$ is easier to learn (equivalently, it is easier to generalize to the true unknown distribution). This means it has more predictive power.

Note, however, that realizability is a strong assumption. So there is a tradeoff between making $\mathcal{H}$ large enough that it is likely to contain the true labeling function, and small enough that it is easy to learn (in the PAC sense). This is sometimes referred to as the "bias-variance tradeoff."

Remark 5.4 (Improving the bound on learning conjunctions)**.** Recall we showed earlier that we can have

$$n_{\mathcal{H}_{\text{conj}}}(\varepsilon,\delta) = \frac{2d}{\varepsilon}\ln\left(\frac{2d}{\delta}\right).$$

Now, using Occam's razor, we know we can do better. Recall $|\mathcal{H}_{\text{conj}}| = 3^d$. Therefore,

$$n_{\mathcal{H}_{\text{conj}}}(\varepsilon,\delta) = \frac{1}{\varepsilon}\left(\ln(1/\delta) + d\log 3\right).$$

This bound is better by $O(\log d)$. Note that we **don't** need a new algorithm to achieve this bound; it just means that our earlier bound was not as tight as it could have been.

CSE 5523: Machine Learning Theory Fall 2017

6 - PAC Learning (Part 3)

Instructor: Raef Bassily

There are two challenges we still need to address:

1. Realizability: We've been sweeping the realizability assumption under the rug; we'll need to return to the "agnostic PAC model" to address this.
2. Finiteness: We want results that guarantee learnability for some infinite hypothesis classes.

6.1 Agnostic PAC Learning and the Uniform Convergence Principle

Recall that: A hypothesis class $\mathcal{H}$ is *agnostic PAC learnable* if there is an algorithm A and a function $n_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ such that for every $0 < \varepsilon, \delta < 1$ and any distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, if we are given a set S of n i.i.d. examples from $\mathcal{D}$, then w.p. $\geq 1 - \delta$, A outputs a hypothesis $h_S \in \mathcal{H}$ for which

$$\operatorname{err}(h_S; \mathcal{D}) \leq \varepsilon + \inf_{h \in \mathcal{H}} \operatorname{err}(h; \mathcal{D}).$$

Remark 6.1. Previously, with the realizability assumption, our recipe for obtaining a good hypothesis was *consistency* with the training set.

Now, instead of consistency, we will consider a more general principle to find a good hypothesis $h_S \in \mathcal{H}$. This is Empirical Risk Minimization (ERM).

We will look for a hypothesis $h_S \in \mathcal{H}$ that minimizes the empirical error over the given training set S. That is,

$$h_S \in \operatorname*{argmin}_{h_S \in \mathcal{H}} \widehat{\operatorname{err}}(h; S).$$

Recall that for the 0-1 loss, for any data distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$:

$$\operatorname{err}(h; \mathcal{D}) = E_{(x,y)\sim\mathcal{D}}[1(h(x_i) \neq y_i)],$$
$$\widehat{\operatorname{err}}(h; S) = \frac{1}{n} \sum_{i=1}^{n} 1(h(x_i) \neq y_i).$$

Let h^* denote an optimal hypothesis with respect to the true error:

$$h^* \in \operatorname*{argmin}_{h \in \mathcal{H}} \operatorname{err}(h; \mathcal{D}).$$

Now, our goal is to obtain a high-confidence bound on

$$\operatorname{err}(h_S; \mathcal{D}) - \operatorname{err}(h^*; \mathcal{D}).$$

Claim 6.2. Let $h_S \in \mathcal{H}$ be an ERM hypothesis (i.e., h_S is a hypothesis that achieves minimum empirical error on S), and let $h^* \in \mathcal{H}$ be an optimal hypothesis w.r.t. the true error. Then,

$$\underbrace{\operatorname{err}(h_S; \mathcal{D}) - \operatorname{err}(h^*; \mathcal{D})}_{\text{"excess risk"}} \leq 2 \sup_{h \in \mathcal{H}} \left| \widehat{\operatorname{err}}(h; \mathcal{S}) - \operatorname{err}(h; \mathcal{D}) \right|.$$

Proof. First, note that

$$\begin{aligned}\operatorname{err}(h_S;\mathcal{D}) - \operatorname{err}(h^*;\mathcal{D}) = &(\operatorname{err}(h_S;\mathcal{D}) - \widehat{\operatorname{err}}(h_S;S)) \\ &+ (\widehat{\operatorname{err}}(h_S;S) - \widehat{\operatorname{err}}(h^*;S)) \\ &+ (\widehat{\operatorname{err}}(h^*;S) - \operatorname{err}(h^*;\mathcal{D}))\,.\end{aligned}$$

The first bracketed expression is always $\leq \sup_{h\in\mathcal{H}} |\operatorname{err}(h;\mathcal{D}) - \widehat{\operatorname{err}}(h;S)|$. Same for the third bracket. The second bracketed expression is always ≤ 0.

Hence, we can bound the difference above by

$$\operatorname{err}(h_S;\mathcal{D}) - \operatorname{err}(h^*;\mathcal{D}) \leq 2 \sup_{h\in\mathcal{H}} |\operatorname{err}(h;\mathcal{D}) - \widehat{\operatorname{err}}(h;S)|.$$

□

Lemma 6.3. Fix any $h \in \mathcal{H}$. Let S be a set of n i.i.d. examples from $\mathcal{D}$. Then,

$$P(|\widehat{\operatorname{err}}(h;S) - \operatorname{err}(h;\mathcal{D})| > \varepsilon) \leq 2e^{-2\varepsilon^2 n}.$$

Proof. This follows from the Chernoff-Hoeffding inequality, since the empirical error is the average of n Bernoulli random variables, each of which has expectation given by the true error. □

Next, we derive a general condition for agnostic PAC learnability of any finite hypothesis class.

Theorem 6.4 (Uniform Convergence Principle)**.** Let $\mathcal{H}$ be any finite hypothesis class. For any $0 < \varepsilon, \delta < 1$, suppose we have a set S of n i.i.d. examples drawn from an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. Let $h_S \in \mathcal{H}$ be an ERM hypothesis, and let h^* be an optimal hypothesis w.r.t. the true error on $\mathcal{D}$. If

$$n \geq \frac{2}{\varepsilon^2} \log\left(\frac{2|\mathcal{H}|}{\delta}\right).$$

Then, w.p. $\geq 1 - \delta$, we have

$$\operatorname{err}(h_S;\mathcal{D}) - \operatorname{err}(h^*;\mathcal{D}) \leq \varepsilon.$$

Proof. We can bound the probability of getting a bad hypothesis as follows:

$$\begin{aligned}&P(\operatorname{err}(h_S;\mathcal{D}) - \operatorname{err}(h^*;\mathcal{D}) > \varepsilon) \\ &\quad\leq P\left(\max_{h\in\mathcal{H}} |\widehat{\operatorname{err}}(h;S) - \operatorname{err}(h;\mathcal{D})| > \varepsilon/2\right) \\ &\quad= P(\exists\, h \in \mathcal{H} : |\widehat{\operatorname{err}}(h;S) - \operatorname{err}(h;\mathcal{D})| > \varepsilon/2) && \text{(using claim 6.2)} \\ &\quad\leq \sum_{h\in\mathcal{H}} P(|\widehat{\operatorname{err}}(h;S) - \operatorname{err}(h;\mathcal{D})| > \varepsilon/2) && \text{(via union bound)} \\ &\quad\leq 2\,|\mathcal{H}|\, e^{-\varepsilon^2 n/2}. && \text{(using lemma 6.3)}\end{aligned}$$

This is upper-bounded by δ as long as

$$n \geq \frac{2}{\varepsilon^2} \log\left(\frac{2|\mathcal{H}|}{\delta}\right).$$

□

CSE 5523: Machine Learning Theory Fall 2017

Example: Decision Lists

Instructor: Raef Bassily

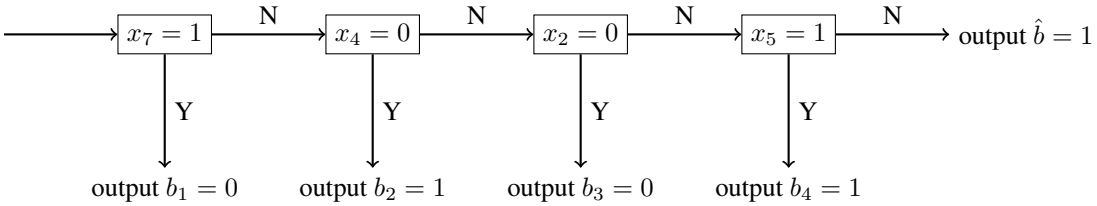
Problem: Learning Decision Lists

Suppose $\mathcal{X} = \{0,1\}^d$, $\mathcal{Y} = \{0,1\}$, and $\mathcal{H}_{\mathsf{List}}$ is the class of *uni-literal* decision lists.

A decision list $h \in \mathcal{H}_{\mathsf{List}}$ is an ordered sequence of ℓ if-then-else statements for some $\ell \in \mathbb{N}$. That is, it is a sequence $(C_1, b_1), \ldots (C_\ell, b_\ell)$, together with a default bit $\hat{b}$, where each C_i is a boolean formula and b_i is the output bit when C_i evaluates to true. For example:

$$\begin{aligned}
&\text{if } C_1 \text{ then output } b_1 \in \{0,1\}\\
&\text{else if } C_2 \text{ then output } b_2 \in \{0,1\}\\
&\qquad\vdots\\
&\text{else if } C_\ell \text{ then output } b_\ell \in \{0,1\}\\
&\qquad\text{else output } \hat{b} \in \{0,1\} \text{ (the default bit)}
\end{aligned}$$

In 1-decision lists (or uni-literal decision lists), each C_i is a conjunction of just **one** literal from the set of boolean literals $\{x_1, \ldots, x_d, \overline{x}_1, \ldots, \overline{x}_d\}$. Here is an illustrated example of a uni-literal decision list of length 4:



Given an unlabeled example $\mathbf{a} = (a_1, \ldots, a_d) \in \{0,1\}^d$ and a decision list h, we assign the variable $x_j = a_j$ for $j \in [d]$ and then run these assignments through the if-then-else statements. Then $h(\mathbf{a})$ is simply the output bit of this procedure.

1. Is $\mathcal{H}_{\mathsf{List}}$ PAC-learnable? Find a sample bound $n_{\mathcal{H}_{\mathsf{List}}}(\varepsilon, \delta)$.
2. Describe an algorithm that PAC-learns $\mathcal{H}_{\mathsf{List}}$.

Solution

First, note that even though $\mathcal{H}_{\text{list}}$ as defined may contain decision lists that are arbitrarily long, it is no loss of generality to restrict ourselves to the uni-literal decision lists of length at most d. To see why, note that if a variable appears twice in a decision list, once as a positive literal and once as a negative literal, if we ever reach the condition corresponding to its second appearance, then it will *always* be satisfied, hence adding the second literal is vacuous. Similarly, if a

variable appears twice as a positive literal (or twice as a negative literal), if we ever reach the second appearance, it will *never* be satisfied.

For example, consider the case where $\overline{x_1}$ is followed by x_1. If the first condition is satisfied, the evaluation stops immediately and the second condition is not evaluated. On the other hand, if the first condition is not satisfied (i.e., x_1 is true), then the second condition will always be satisfied.

Thus, it suffices to consider the set of all uni-literal decision lists where each variable appears at most once. Let $\mathcal{H}'$ denote this set. We can find an upper bound on the size of this set as follows. To simplify our task in computing an upper bound, we consider an equivalent representation of uni-literal decision lists where all lists are of length exactly d. Note that every decision list in $\mathcal{H}'$ of length $k < d$ can be equivalently respresented as a decision list of length exactly d where the first k stages in the new (longer) list are exactly the same as in the original (shorter) list and the output bits for the last $d - k$ conditions of the new list are set to the default bit of the original list.

Using this representation, each decision list can be fully specified by selecting

- a permutation of the d variables ($d!$ choices),
- whether each variable appears as a positive or a negative literal (2^d choices),
- the output bit corresponding to each condition in the decision list (2^d choices), and
- the default output bit (2 choices).

This means that there are at most $d!\, 2^{2d+1}$ distinct uni-literal decision lists involving d variables.

Note that

$$\begin{aligned}\ln\left|\mathcal{H}'\right| &\leq \ln\left(d!\, 2^{2d+1}\right) \\ &\leq \ln\left(d^d\, 2^{2d+1}\right) \\ &= d\ln d + (2d+1)\ln 2.\end{aligned}$$

Therefore, if we have an algorithm which always produces a hypothesis that is consistent with the training data, we can apply Occam's razor, which yields the following bound:

$$\begin{aligned}n_{\mathcal{H}_{\text{list}}}(\epsilon,\delta) &\leq \frac{1}{\epsilon}\ln\left(\frac{\left|\mathcal{H}'\right|}{\delta}\right) \\ &\leq \frac{1}{\epsilon}\left[d\ln d + (2d+1)\ln 2 + \ln\left(1/\delta\right)\right].\end{aligned}$$

Here is an algorithm which PAC-learns $\mathcal{H}_{\text{list}}$:

- Input: training set $S = \{(x^{(i)}, y^{(i)})\}_{i=\{1,\ldots,n\}}$
- If all labels are the same: return an empty decision list with a default bit equal to $y^{(1)}$.
- Otherwise, search for a literal $\ell = x_j$ or $\overline{x_j}$ (for some $j \in [d]$) such that the label $y^{(i)}$ is the same for all examples where ℓ evaluates to true. (Such a literal always exists, due to the realizability assumption.)
- Let b denote the value of the label $y^{(i)}$ shared by all examples where ℓ evaluates to true.
- Let S' denote the set of all examples for which ℓ evaluates to false.
- Recursively call this subroutine on input S'.
- Insert the condition (ℓ, b) at the beginning of the resulting decision list.
- Return the newly constructed decision list

This algorithm always produces a hypothesis that is consistent with the training data, which justifies our use of Occam's razor.

To see why, first note that the algorithm can always find such a literal ℓ in the third step due to realizability. Also, note that since the size of S decreases with each recursive call, the algorithm must terminate. Furthermore, by using induction on the length of the true decision list (which generated the training data), we can show that at every stage of recursion the list created so far is consistent with the examples exhausted up to that stage.

CSE 5523: Machine Learning Theory Fall 2017

7 - The Vapnik-Chervonenkis Dimension (Part 1)

Instructor: Raef Bassily

7.1 Intuition and Examples

This idea will allow us to address the finiteness limitation.

Intuition: Let $\mathcal{H}$ be a hypothesis class. Suppose we have a set $T_n = \{x_1, \ldots, x_n\}$ of n unlabeled domain points in $\mathcal{X}$. The size of the set

$$C_{\mathcal{H}}(T_n) = \{(h(x_1), \ldots, h(x_n)) : h \in \mathcal{H}\}$$

of all possible labelings of T_n under $\mathcal{H}$ is what really matters in describing the complexity of $\mathcal{H}$.

Often we are interested in how the quantity

$$\hat{C}_{\mathcal{H}}(n) = \max_{T_n : |T_n| = n} |C_{\mathcal{H}}(T_n)|$$

scales as a function of n.

Example 7.1 (Thresholds)**.** Suppose we have

$$\begin{aligned} \mathcal{X} &= \mathbb{R}, \\ \mathcal{Y} &= \{-, +\}, \\ \mathcal{H} &= \text{threshold functions } h_t : \mathcal{X} \to \mathcal{Y}, t \in \mathbb{R}, \\ h_t(x) &= \begin{cases} -, & \text{if } x < t, \\ +, & \text{if } x \geq t. \end{cases} \end{aligned}$$

Consider $T_5 = \{x_1, \ldots, x_5\} \subset \mathbb{R}$. We can place our threshold beyond the far left end or the far right end, or we can place it between any adjacent pair of points. Hence, $|C_{\mathcal{H}}(T_5)| = 6$.

In general, for this hypothesis class,

$$\hat{C}_{\mathcal{H}}(n) = \max_{T_n : |T_n| = n} |C_{\mathcal{H}}(T_n)| = n + 1.$$

Example 7.2 (Intervals)**.** Consider $\mathcal{H}$ = indicator functions for intervals on $\mathbb{R}$. We can generate all possible labelings of a set of 2 distinct points in $\mathbb{R}$; that is there is a set $T_2 = \{x_1, x_2\}$ such that $|C_{\mathcal{H}}(T_2)| = 2^2 = 4$. But, for any set of 3 points, we cannot generate all the $2^3 = 8$ labelings: we can only generate 7 labelings since the pattern '$101'$ cannot be generated by any interval indicator function. In fact, for any set T_n of $n \geq 3$ distinct domain points, we have $|C_{\mathcal{H}}(T_n)| = 1 + \binom{n+1}{2}$. Hence, $\hat{C}_{\mathcal{H}}(n) = O(n^2)$.

To see this, note that a set of n points divides the real line into $n + 1$ regions. If we place an interval such that both its endpoints lie between two adjacent data points, we get just one labeling (namely, all 0's). If we place an interval such that its endpoints lie within distinct regions (i.e., it

spans at least one data point), we have $\binom{n+1}{2}$ choices. In total, we have $1+\binom{n+1}{2}$ possible labelings. So,

$$\hat{C}_{\mathcal{H}}(n) = O(n^2).$$

What if we have "bi-directional" intervals? That is, we pick an interval and choose whether to assign 0 or 1 to the interior. Note that we can generate all possible 8 labelings for some set of size 3, but we cannot generate all possible labelings for *any* set of size 4.

Remark 7.3. IFor any class $\mathcal{H}$, we trivially have the following upper bound:

$$\hat{C}_{\mathcal{H}}(n) \leq 2^n.$$

But for many hypothesis classes, we have

$$\hat{C}_{\mathcal{H}}(n) \leq O(n^{K_{\mathcal{H}}}).$$

That is, this quantity is bounded by a polynomial in n of degree $K_{\mathcal{H}}$, where $K_{\mathcal{H}}$ only depends on $\mathcal{H}$, not on n. As we will see later, the degree $K_{\mathcal{H}}$ is in fact the *VC dimension* of $\mathcal{H}$.

Roughly speaking, $K_{\mathcal{H}}$ will play the role of $\ln(|\mathcal{H}|)$ in the previous PAC and agnostic PAC sample bounds (Theorems 5.3 and 6.4) that we have seen for the case of finite hypothesis classes.

7.2 Vapnik-Chervonenkis (VC) Dimension

Definition 7.4 (Set Shattering)**.** A hypothesis class $\mathcal{H}$ shatters a finite set $T_n = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ if the set $C_{\mathcal{H}}(T_n) = \{(h(x_1), \ldots, h(x_n)) : h \in \mathcal{H}\} = \{0,1\}^n$. (This means that it can generate all 2^n labelings of T_n.)

Definition 7.5 (VC Dimension)**.** The VC dimension of $\mathcal{H}$ is the size of the largest set of domain points that can be shattered by $\mathcal{H}$. The VC dimension is denoted $\text{VC}(\mathcal{H})$ or $K_{\mathcal{H}}$.

The VC dimension is infinite if, for all n, there is a set T_n of size n that can be shattered by $\mathcal{H}$.

Remark 7.6. The function $\hat{C}_{\mathcal{H}}(n) = \max_{T_n \subset \mathcal{X}:|T_n|=n} |C_{\mathcal{H}}(T_n)|$ is called the "growth function."

For the class of binary thresholds. We cannot shatter any set of size 2, so VC(thresholds) = 1. For this class, we saw that $\hat{C}_{\mathcal{H}}(n) = n + 1 = O(n)$.

Example 7.7 (Linear classifiers in $\mathbb{R}^2$)**.** For linear classifiers in $\mathbb{R}^2$, we can find a set of 3 points that is shattered by the class. However, we can not shatter **any** set of size 4. Hence, the VC dimension of this class is 3. More generally, for linear classifiers in $\mathbb{R}^d$, we can find a set of $d+1$ points that is shattered by the class, but we cannot shatter *any* set of size $d+2$ using this class. Hence, the VC dimension is $d+1$.

Example 7.8 (Axis-aligned rectangles)**.** There exist some sets of size 4 (not necessarily all sets of size 4) that we can shatter. This means 4 is a lower bound on the VC dimension.

For any set of 5 points, at least one point must not be "extremal" (i.e., not at the extreme left, right, top, or bottom). We cannot generate the labeling where this point gets assigned "$-$" and all others get assigned "+". So *no* collection of 5 points can be shattered.

Hence, VC(axis-aligned rectangles) = 4.

7.3 Some Properties of the VC Dimension

Theorem 7.9 (The VC dimension of finite hypothesis classes)**.** For any finite hypothesis class $\mathcal{H}$,

$$\mathrm{VC}(\mathcal{H}) \leq \log_2(|\mathcal{H}|).$$

Proof. Let $T \subset \mathcal{X}$ be any set that can be shattered by $\mathcal{H}$. Note that for each labeling in $C_{\mathcal{H}}(T)$, there is at least an $h \in \mathcal{H}$ that has generated this labeling. Then

$$|\mathcal{H}| \geq |C_{\mathcal{H}}(T)| = 2^{|T|}.$$

Since the above inequality is true for any set T that can be shattered by $\mathcal{H}$, then we must have

$$|\mathcal{H}| \geq 2^{\mathrm{VC}(\mathcal{H})}.$$

Therefore, $\mathrm{VC}(\mathcal{H}) \leq \log_2(|\mathcal{H}|)$. □

Example 7.10 (Thresholds over a finite set)**.** Suppose

$$\mathcal{X} = \{1, \ldots, \ell\},$$
$$\mathcal{H} = \{\text{threshold functions } h_t : t \in \mathcal{X}\}.$$

In this example, $\log_2(|\mathcal{H}|) = \log_2 \ell$, but the VC dimension is 1. This example shows that the VC dimension of a finite hypothesis class can be significantly smalles than log the size of the class. This can lead to improved sample bounds (i.e., better $n_{\mathcal{H}}(\varepsilon, \delta)$).

Fact 7.11. Let $\mathcal{H}_1, \mathcal{H}_2$ be two hypothesis classes. Suppose that for every $T \subset \mathcal{X}$ that is shattered by $\mathcal{H}_1$, T is also shattered by $\mathcal{H}_2$. Then $\mathrm{VC}(\mathcal{H}_1) \leq \mathrm{VC}(\mathcal{H}_2)$.

Proof. Let $\mathrm{VC}(\mathcal{H}_2) = K$. For the sake of contradiction, suppose $\mathrm{VC}(\mathcal{H}_1) \geq K + 1$. This means $\mathcal{H}_1$ can shatter a set of size $K + 1$. By the premise, this implies that $\mathcal{H}_2$ can shatter this set, which implies $\mathrm{VC}(\mathcal{H}_2) \geq K + 1$. This is a contradiction. □

Fact 7.12. Let $\mathcal{H}_1, \mathcal{H}_2$ be two hypothesis classes over domain $\mathcal{X}$. Suppose that $\mathrm{VC}(\mathcal{H}_1) < |\mathcal{X}|$ *(which is usually the case for all practically meaningful classes).* Suppose that for every $W \subset \mathcal{X}$ that is shattered by $\mathcal{H}_1$, there is some element $x \in \mathcal{X} \setminus W$ such that $W \cup \{x\}$ is shattered by $\mathcal{H}_2$. Then $\mathrm{VC}(\mathcal{H}_1) \leq \mathrm{VC}(\mathcal{H}_2) - 1$.

Proof. Let W be the largest set that can be shattered by $\mathcal{H}_1$. That is, $\mathrm{VC}(\mathcal{H}_1) = |W|$. By the premise, there is $x \in \mathcal{X} \setminus W$ s.t. $\tilde{W} = W \cup \{x\}$ is shattered by $\mathcal{H}_2$. Hence, $\mathrm{VC}(\mathcal{H}_2) \geq |W| + 1 = \mathrm{VC}(\mathcal{H}_1) + 1$. □

CSE 5523: Machine Learning Theory Fall 2017

8 - The Vapnik-Chervonenkis Dimension (Part 2)

Instructor: Raef Bassily

8.1 Fundamental Theorem of Statistical Learning

Theorem 8.1 (Characterization of learnability via VC dimension)**.** Let $\mathcal{H}$ be any hypothesis class. $\mathcal{H}$ is PAC (and agnostic-PAC) learnable *iff* $\mathcal{H}$ has a finite VC dimension.

Let's first consider the general agnostic case (i.e., without realizability). The proof of the above theorem involves showing that

$$\mathbb{P}_{S\sim\mathcal{D}^n}\left[\ \mathrm{err}(h_S;\mathcal{D}) - \mathrm{err}(h^*;\mathcal{D}) > \varepsilon\ \right] \leq\ O\left(n^{\mathrm{VC}(\mathcal{H})}\right) e^{-\Omega(\varepsilon^2 n)} \triangleq \delta.$$

This, in turn, is proved in two main big steps. The first step is given by following lemma.

Lemma 8.2 (Uniform convergence bound on the generalization error in the general (agnostic) case)**.** Let $\mathcal{H}$ be any hypothesis class. Let $\mathcal{D}$ be any distribution over $\mathcal{X}\times\mathcal{Y}$. Let S be a training set of n i.i.d. examples from $\mathcal{D}$. Then

$$\mathbb{P}_{S\sim\mathcal{D}^n}\left[\ \exists\, h\in\mathcal{H} : |\widehat{\mathrm{err}}(h;S) - \mathrm{err}(h;\mathcal{D})| > \varepsilon\ \right] \leq\ 4\,\hat{C}_{\mathcal{H}}(2n)\, e^{-\varepsilon^2 n/8}.$$

The second step is done via the following important lemma, usually know as *Sauer's Lemma*, which upper-bounds the growth function $\hat{C}_{\mathcal{H}}(n)$ in terms of $\mathrm{VC}(\mathcal{H})$. In particular, Sauer's Lemma asserts the observation we have made before in several examples that $\hat{C}_{\mathcal{H}}(n) = O(n^{\mathrm{VC}(\mathcal{H})})$. The lemma asserts that this is indeed true for *any* hypothesis class.

Lemma 8.3 (Sauer's Lemma)**.** If $\mathrm{VC}(\mathcal{H}) = k$, then for any $T\subset\mathcal{X}$ of size n, we have

$$\begin{aligned}
|C_{\mathcal{H}}(T)| &\leq \sum_{i=0}^{k}\binom{n}{i} \\
&\leq \left(\frac{en}{k}\right)^k \qquad \text{for } n > k \\
&= O(n^k) \qquad\qquad \text{for } n > k.
\end{aligned}$$

Now, putting Lemmas 8.2 and 8.3 together, we can prove the following theorem.

Theorem 8.4 (Characterization of agnostic PAC learning)**.** Let $\mathcal{H}$ be a hypothesis class with $\mathrm{VC}(\mathcal{H}) = k$. Then $\mathcal{H}$ is agnostic PAC learnable. In particular, $\mathcal{H}$ is agnostic PAC learnable via an *ERM* algorithm that, given a training sample S of n i.i.d. examples from the underlying unknown distribution $\mathcal{D}$, outputs an ERM hypothesis $h_S\in\mathcal{H}$.

For any $0<\varepsilon<1$, the output h_S satisfies

$$\mathbb{P}_{S\sim\mathcal{D}^n}\left[\ \mathrm{err}(h_S;\mathcal{D}) - \min_{h\in\mathcal{H}}\mathrm{err}(h;\mathcal{D}) > \varepsilon\ \right] \leq\ 4\left(\frac{2en}{k}\right)^k e^{-\frac{\varepsilon^2 n}{32}}$$

Hence, the sample complexity for agnostic-PAC learning of $\mathcal{H}$ is at most

$$n_{\mathcal{H}}(\varepsilon,\delta) = O\left(\frac{k\ln(1/\varepsilon)+\ln(1/\delta)}{\varepsilon^2}\right).$$

Proof outline. Let $h^* = \operatorname*{argmin}_{h \in \mathcal{H}} \operatorname{err}(h; \mathcal{D})$. Recall the approach we used in the proof for the case of finite hypothesis classes. For an ERM hypothesis $h_S \in \mathcal{H}$, we have

$$\begin{aligned}
\mathbb{P}_{S\sim\mathcal{D}^n}\left[\ |\operatorname{err}(h_S;\mathcal{D}) - \operatorname{err}(h^*;\mathcal{D})| > \varepsilon\ \right] &\le \mathbb{P}_{S\sim\mathcal{D}^n}\left[\ 2\max_{h\in\mathcal{H}} |\widehat{\operatorname{err}}(h;S) - \operatorname{err}(h;\mathcal{D})| > \varepsilon\ \right] \\
&= \mathbb{P}_{S\sim\mathcal{D}^n}\left[\ \exists\, h \in \mathcal{H} : |\widehat{\operatorname{err}}(h;S) - \operatorname{err}(h;\mathcal{D})| > \varepsilon/2\ \right] \\
&\le 4\,\hat{C}_{\mathcal{H}}(2n)\,\exp(-\varepsilon^2 n/32) \qquad \text{this follows from Lemma 8.2} \\
&\le 4\left(\frac{2en}{k}\right)^k \exp(-\varepsilon^2 n/32).
\end{aligned}$$

Then for sufficiently large n, we can get this to be less than δ. With a careful analysis to the bound we obtained in the last step and applying Lemma A.2 in S-Shwartz & Ben-David's textbook, we can show that $n_{\mathcal{H}}(\varepsilon, \delta)$ is as given in the theorem. □

8.1.1 Analogous statements for the realizable case

When realizability is assumed (i.e., in the PAC model), we can arrive at an analogous theorem to Theorem 8.4. As we saw before in the case of finite hypothesis classes, the statement we will arrive at for the realizable case gives a better sample complexity than that of Theorem 8.4. In particular, the dependency of $n_{\mathcal{H}}(\epsilon, \delta)$ on ϵ is improved from $\frac{1}{\epsilon^2}$ to $\frac{1}{\epsilon}$.

As in the previous section, we take two steps to arrive at the final statement. The first step is given by the following lemma (which is analogous to Lemma 8.2). The second step is the same as before: we apply Sauer's Lemma (Lemma 8.3 above).

Lemma 8.5 (Uniform convergence bound in the realizable case in terms of the growth function)**.** Let $\mathcal{H}$ be any hypothesis class. Assume realizability, and let $f \in \mathcal{H}$. Let $\mathcal{D}$ be any distribution over the domain $\mathcal{X}$. Let S be a training set of n i.i.d. examples from $(\mathcal{D}, f)$. Then,

$$\mathbb{P}_{S\sim(\mathcal{D},f)^n}\left[\ \exists\, h \in \mathcal{H} : \operatorname{err}(h;\mathcal{D}) > \varepsilon \text{ and } h \text{ is consistent with } S\ \right] \ \le\ 2\,\hat{C}_{\mathcal{H}}(2n)\,\exp(-\varepsilon n/4).$$

Putting this together with Sauer's Lemma, we can reach the following theorem.

Theorem 8.6 (Characterization of PAC learning (the realizable case))**.** Let $\mathcal{H}$ be a hypothesis class with $\mathrm{VC}(\mathcal{H}) = k$. Then $\mathcal{H}$ is PAC learnable[1]. In particular, $\mathcal{H}$ is PAC learnable via an *ERM* algorithm that, given a training sample S of n i.i.d. examples, outputs a hypothesis $h_S \in \mathcal{H}$ that is *consistent* with S.

For any $0 < \varepsilon < 1$, the output h_S satisfies

$$\mathbb{P}_{S\sim\mathcal{D}^n}\left[\ \operatorname{err}(h_S;\mathcal{D}) > \varepsilon\ \right] \le\ 2\left(\frac{2en}{k}\right)^k e^{-\frac{\varepsilon n}{4}}$$

Hence, the sample complexity for PAC-learning $\mathcal{H}$ is at most

$$n_{\mathcal{H}}(\varepsilon, \delta) = O\left(\frac{k\ln(1/\varepsilon) + \ln(1/\delta)}{\varepsilon}\right).$$

[1]Here, realizability is assumed. Recall that realizability is by default assumed in the PAC model.

CSE 5523: Machine Learning Theory Fall 2017

9 - Weak versus Strong Learnability & Boosting

Instructor: Raef Bassily

9.1 Weak versus Strong Learnability

Definition 9.1 (γ-weak learner)**.** Suppose γ be some small number bounded away from 1/2, say $\gamma \in (0, \frac{1}{4}]$. An algorithm A is a γ-weak learner for a hypothesis class $\mathcal{H}$ if there is a function $n_{\mathcal{H}} : (0,1) \to \mathbb{N}$ such that for every $0 < \delta < 1$ and every distribution $\mathcal{D}$, given $n \geq n_{\mathcal{H}}(\delta)$ i.i.d. examples from $\mathcal{D}$, w.p. $\geq 1 - \delta$, A outputs a hypothesis $h \in \mathcal{H}$ with

$$\text{err}(h; \mathcal{D}) \leq \frac{1}{2} - \gamma.$$

Informally, a weak learner merely has to be a little better than random guessing.

Definition 9.2 (Strong learner)**.** An algorithm B is a strong learner for a hypothesis class $\tilde{\mathcal{H}}$ if there is a function $n_{\mathcal{H}} : (0,1) \to \mathbb{N}$ s.t. for every $0 < \varepsilon, \delta < 1$, and for every distribution $\mathcal{D}$, given $n \geq n_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d. examples from $\mathcal{D}$, w.p. $\geq 1 - \delta$, B outputs $h \in \tilde{\mathcal{H}}$ with

$$\text{err}(h; \mathcal{D}) \leq \varepsilon.$$

Question: Given a weak learner for a class $\mathcal{H}$, can we find an efficient method that produces a strong learner for a possibly richer, more complex class $\tilde{\mathcal{H}}$?

In the transformation from weak to strong learnability, we are concerned with boosting the accuracy, and we are allowed to output a more "complex" or "powerful" hypothesis which is normally constructed based on several hypotheses (or, "predictions rules") generated from the weak learner.

9.2 Boosting

Generally, the boosting paradigm adheres to the following outline:

- Round 1: $h_1 \leftarrow \text{WeakLearner}(\mathcal{D}_1)$ *(The weak learner is given i.i.d. examples from $\mathcal{D}_1$.)*
- Round 2: $h_2 \leftarrow \text{WeakLearner}(\mathcal{D}_2)$
- $\vdots$
- Round T: $h_T \leftarrow \text{WeakLearner}(\mathcal{D}_T)$
- Return: weighted majority of $h_1, \ldots, h_T$: $\tilde{h}(x) = \text{sign}\left(\alpha_1 h_1(x), \ldots, \alpha_T h_T(x) \right)$.

Here, $\mathcal{D}_1$ is closely related to $\mathcal{D}$ the original (unknown) distribution. For $t > 1$, $\mathcal{D}_t$ is a modification of $\mathcal{D}_{t-1}$, with higher weight on the examples that were misclassified by h_{t-1}.

The weights $\alpha_1, \ldots, \alpha_T$ are chosen such that more weight is given to "more accurate" hypotheses.

Note that $\tilde{h}$ does not necessarily belong to the "base" hypothesis class $\mathcal{H}$ associated with the weak learners. It is a linear classifier composed over T weak hypotheses.

Why would we want to use boosting?

Controlling the bias-complexity tradeoff Remember that our goal is to output a hypothesis h_S that has small true risk $\mathrm{err}(h_S; \mathcal{D})$. Note that

$$\mathrm{err}(h_S; \mathcal{D}) = \left(\mathrm{err}(h_S; \mathcal{D}) - \min_{h \in \mathcal{H}} \mathrm{err}(h; \mathcal{D}) \right) + \min_{h \in \mathcal{H}} \mathrm{err}(h; \mathcal{D}).$$

The expression in parentheses is the *estimation error*, and the last term is the *approximation error.*

- Using a simpler hypothesis space typically increases the approximation error (more constrained search space) but decreases the estimation error (lower VC dimension).
- In contrast, using a richer hypothesis space typically decreases the approximation error (less constrained search space) but increases the estimation error (higher VC dimension).

This is the "bias-complexity tradeoff." Boosting lets us smoothly control this tradeoff by changing the value of T.

Better computational efficiency Boosting allows us to run a very simple (and therefore likely efficient) algorithm T times, where T is often small.

9.3 Adaboost ("Adaptive Boosting")

Adaboost is a particular boosting method due to Schapire and Freund (1995).

Let $\mathcal{Y} = \{-1, +1\}$ and let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y}$. We will construct "artificial" discrete distributions over S itself (as if S is a "new domain").

The distributions will be generated in an adaptive fashion. The initial distribution is a uniform distribution over S: $\mathcal{D}^{(1)} = \left(\mathcal{D}_1^{(1)}, \mathcal{D}_2^{(1)}, \ldots, \mathcal{D}_n^{(1)} \right) = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right)$.

function ADABOOST(WL, T)
Inputs: Weak learner WL, number of rounds T.
Initialize $\mathcal{D}^{(1)} = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n} \right)$
for $t = 1, \ldots, T$ **do**
$h_t \leftarrow \mathrm{WL}(\mathcal{D}^{(t)}, S)$
(i.e., WL *gets points from S, sampled with replacement using weights specified by $\mathcal{D}^{(t)}$. The number of examples fed to* WL *should be enough to ensure weak learnability, and is typically much less than the size of S.)*
$\varepsilon_t \leftarrow \sum_{i=1}^{n} \mathcal{D}_i^{(t)} \, 1(h_t(x_i) \neq y_i)$ *(Note: We expect $\varepsilon_t \leq \frac{1}{2} - \gamma$.)*
$\alpha_t \leftarrow \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$
Update the discrete distribution:

$$\mathcal{D}_i^{(t+1)} \leftarrow \frac{\mathcal{D}_i^{(t)} \exp(-\alpha_t \, y_i \, h_t(x_i))}{\sum_{j=1}^{n} \mathcal{D}_j^{(t)} \exp(-\alpha_t \, y_j \, h_t(x_j))}$$

end for
return $\tilde{h}$ defined as

$$\tilde{h}(x) = \mathrm{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right)$$

end function

Example 9.3. Suppose $S \subset \mathbb{R}^2$ and suppose the base class $\mathcal{H}$ is taken to be the class of horizontal or vertical half-planes (a.k.a. "decision stumps").

- First round ($t = 1$): Sample from $\mathcal{D}^{(1)}$ = uniform over S. (This is sampling with replacement. You can think of S as a "new domain".)
- Run weak learner on this sample to obtain h_1 (which is some horizontal or vertical line in $\mathbb{R}^2$ defining a decision boundary - this is called a half-plane or "decision stump").
- Identify misclassified points
- Obtain $\mathcal{D}^{(t+1)}$ by assigning higher weight to these misclassified points.
- Next round: sample from using $\mathcal{D}^{(t+1)}$ (again, this is sampling with replacement) and run weak learner, obtaining h_{t+1}.

CSE 5523: Machine Learning Theory Fall 2017

10 - Analysis of Adaboost

Instructor: Raef Bassily

10.1 Adaboost: Training error

Theorem 10.1. Let $0 < \hat{\delta} < 1,\ 0 < \gamma < 1/2$. Suppose for each $t \in [T]$, w.p. at least $1 - \hat{\delta}$, $\varepsilon_t \leq \frac{1}{2} - \gamma$. Then, w.p. at least $1 - \hat{\delta}T$, the **training error** of $\tilde{h}$ over the training set S is

$$\widehat{\mathrm{err}}(\tilde{h}; S) \leq \exp(-2\gamma^2 T).$$

Proof. For convenience, define

$$f(x) = \sum_{t=1}^{T-1} \alpha_t h_t(x), \text{ and}$$
$$z_t = \sum_{j=1}^{n} \mathcal{D}_j^{(t)} \exp(-\alpha_t\, y_j\, h_t(x_j)).$$

Thus, z_t is the normalizing factor in the expression for $\mathcal{D}^{(t)}$.
Then, note that we can expand the expression for the final distribution to obtain

$$\mathcal{D}_i^{(T)} = \frac{1}{n} \cdot \frac{\exp\left(-y_i \sum_{t=1}^{T-1} \alpha_t\, h_t(x_i)\right)}{\prod_{t=1}^{T-1} z_t}$$
$$= \frac{1}{n} \cdot \frac{\exp\left(-y_i\, f(x_i)\right)}{\prod_{t=1}^{T-1} z_t}$$

Furthermore,

$$\begin{aligned}
\widehat{\mathrm{err}}(\tilde{h}; S) &= \frac{1}{n} \sum_{i=1}^{n} 1(y_i f(x_i) \leq 0) \\
&\leq \frac{1}{n} \sum_{i=1}^{n} e^{-y_i f(x_i)} && \text{(because } 1(u \geq 0) \leq e^u \text{ for any } u\text{)} \\
&= \sum_{i=1}^{n} \mathcal{D}_i^{(T)} \prod_{t=1}^{T-1} z_t \\
&= \prod_{t=1}^{T-1} z_t \sum_{i=1}^{n} \mathcal{D}_i^{(T)} \\
&= \prod_{t=1}^{T-1} z_t.
\end{aligned}$$

Now, we just need to bound the product $\prod_{t=1}^{T-1} z_t$. We can rewrite each factor z_t as

$$\begin{aligned}
z_t &= \sum_{i=1}^{n} \mathcal{D}_i^{(t)} \exp(-\alpha_t\, y_i\, h_t(x_i)) \\
&= \sum_{i\,:\,h_t \text{ correct on } (x_i,y_i)} \mathcal{D}_i^{(t)} \exp(-\alpha_t\, y_i\, h_t(x_i)) \\
&\quad + \sum_{i\,:\,h_t \text{ incorrect on } (x_i,y_i)} \mathcal{D}_i^{(t)} \exp(-\alpha_t\, y_i\, h_t(x_i)) \\
&= \sum_{i\,:\,h_t \text{ correct on } (x_i,y_i)} \mathcal{D}_i^{(t)} \exp(-\alpha_t) \\
&\quad + \sum_{i\,:\,h_t \text{ incorrect on } (x_i,y_i)} \mathcal{D}_i^{(t)} \exp(\alpha_t) \\
&= \sum_{i\,:\,h_t \text{ correct on } (x_i,y_i)} \mathcal{D}_i^{(t)} \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} \\
&\quad + \sum_{i\,:\,h_t \text{ incorrect on } (x_i,y_i)} \mathcal{D}_i^{(t)} \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}.
\end{aligned}$$

Then, we can rewrite this in terms of probabilities:

$$\begin{aligned}
z_t &= \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} P_{(x,y)\sim\mathcal{D}^{(t)}}(h_t(x) = y) + \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} P_{(x,y)\sim\mathcal{D}^{(t)}}(h_t(x) \neq y) \\
&= \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}(1-\varepsilon_t) + \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}\varepsilon_t \\
&= 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \\
&\leq 2\sqrt{(1/2-\gamma)(1/2+\gamma)} \\
&= 2\sqrt{1/4-\gamma^2} \\
&\leq e^{-2\gamma^2}.
\end{aligned}$$

Hence,

$$\widehat{\mathrm{err}}(\tilde{h}; S) \leq \prod_{t=1}^{T-1} z_t \leq e^{-2\gamma^2 T}.$$

□

Corollary 10.2. If in each round, w.p. $\geq 1-\hat{\delta}$, WL returns hypothesis with error $\leq \frac{1}{2}-\gamma$, then after $T = \frac{1}{2\gamma^2}\ln\frac{1}{\varepsilon}$ rounds, w.p. $\geq 1-\frac{\hat{\delta}}{2\gamma^2}\ln\frac{1}{\varepsilon}$, the final hypothesis $\tilde{h}$ has $\widehat{\mathrm{err}}(\tilde{h}; S) \leq \varepsilon$.

10.2 Generalization error of Adaboost

Note that our final goal is to have a small *true* error for the output hypothesis of Adaboost. That is, our target quantity is $\mathrm{err}(\tilde{h}; \mathcal{D})$. Note that we can bound the true error as follows:

$$\mathrm{err}(\tilde{h}; \mathcal{D}) \leq \widehat{\mathrm{err}}(\tilde{h}; S) + \left|\widehat{\mathrm{err}}(\tilde{h}; S) - \mathrm{err}(\tilde{h}; \mathcal{D})\right|.$$

where S is a set of n i.i.d. examples from the unknown distribution $\mathcal{D}$.

By the previous theorem, we can ensure that the training error (the first term on the RHS) is small. Note the remaining term on the RHS is the generalization error of $\tilde{h}$: $\left|\widehat{\text{err}}(\tilde{h}; S) - \text{err}(\tilde{h}; \mathcal{D})\right|$.

So, to bound the true error of $\tilde{h}$, it remains to bound (with high probability) the generalization error of $\tilde{h}$.

Note that the output hypothesis $\tilde{h}$ of Adaboost is a random variable whose possible realizations depend on the realizations of the T output hypotheses $h_1, \ldots, h_T$ of the weak learner and the values of the weights $\alpha_1, \ldots, \alpha_T$. Let $\widetilde{\mathcal{H}}$ be the class that contains all possible realizations $\tilde{h}$; that is

$$\widetilde{\mathcal{H}} = \left\{ h : h \text{ has the form: } \forall x \in \mathcal{X},\ h(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right), h_t \in \mathcal{H}, \alpha_t > 0 \ \ \forall t \in [T] \right\} \quad (1)$$

Suppose for now that $\widetilde{\mathcal{H}}$ has a finite VC dimension $\tilde{k}$ (note that $\tilde{k}$ will depend on the number of rounds T and the VC dimension of the base class $\mathcal{H}$, as will be quantified later). Then, we can use Lemma 8.2 in the notes (that gives a uniform convergence bound on the generalization error in terms of the VC dimension), together with Lemma 8.3 (Sauer's Lemma) to bound the generalization error as follows:

$$\begin{aligned} P\left(\left|\widehat{\text{err}}(\tilde{h}; S) - \text{err}(\tilde{h}; \mathcal{D})\right| > \tilde{\varepsilon}\right) &\leq P(\exists\, h \in \widetilde{\mathcal{H}} : |\widehat{\text{err}}(h; S) - \text{err}(h; \mathcal{D})| > \tilde{\varepsilon}) \\ &\leq 4\, \hat{C}_{\widetilde{\mathcal{H}}}(2n) e^{-\tilde{\varepsilon}^2 n/8} \qquad \text{using Lemma 8.2} \\ &\leq 4 \left(\frac{en}{\tilde{k}}\right)^{\tilde{k}} e^{-\tilde{\varepsilon}^2 n/8} \qquad \text{using Lemma 8.3} \end{aligned}$$

Hence, to ensure that, with probability $1 - \tilde{\delta}$, the generalization error is below some desired level $\tilde{\varepsilon}$, it suffices to have number of examples $n = O\left(\frac{\tilde{k}\ln(1/\tilde{\varepsilon}) + \ln(1/\tilde{\delta})}{\tilde{\varepsilon}^2}\right)$. *Note that such n implicitly depends on T since $\tilde{k}$ depends on T as explained below.*

Theorem 10.3. Let $\mathcal{H}$ be a base hypothesis class (that is, the class associated with a weak learner) where $\text{VC}(\mathcal{H}) = k$. The class $\widetilde{\mathcal{H}}$ defined in (1) has VC dimension

$$\tilde{k} = O(kT \log(kT)).$$

Remark 10.4. In boosting, the "input" hypothesis all come from $\mathcal{H}$, and we feed them into a hypothesis from class $\mathcal{G}$ = homogeneous linear classifiers. We can think of the class $\widetilde{\mathcal{H}}$ as the composition of $\mathcal{G}$ applied to T instances of $\mathcal{H}$.

10.2.1 General Bound on the VC-dimension of a composition of hypothesis classes

More generally, we could describe a compound structure using a **directed acyclic graph** of compositions of hypothesis classes. The following theorem gives a general bound on the VC dimension of any compound hypothesis class represented by a directed acyclic graph:

Theorem 10.5. For any directed acyclic graph with T nodes where each node represents a hypothesis class $\mathcal{H}_i$ with $\text{VC}(\mathcal{H}_i) = k_i$, the equivalent compound hypothesis class $\mathcal{H}_{\mathsf{comp}}$ represented by the graph has VC dimension

$$\text{VC}(\mathcal{H}_{\mathsf{comp}}) \leq 2 \left(\sum_{i=1}^{T} k_i\right) \log\left(e \sum_{i=1}^{T} k_i\right).$$

Remark 10.6 (Boosting case)**.** For boosting, there are T base classes (all equal to $\mathcal{H}$), and one class $\mathcal{G}$ of *homogeneous* linear separators (with $\mathrm{VC}(\mathcal{G}) = T$).

The sum of all VC dimensions in the graph is $kT + T$. Applying Theorem 10.5 to the compound class of boosted hypotheses gives Theorem 10.3. That is,

$$\begin{aligned}\mathrm{VC}(\widetilde{\mathcal{H}}) &\leq 2(kT + T)\log_2\left(e(kT + T)\right) \\ &= O(kT\log(kT)).\end{aligned}$$

10.3 Margin-Based Explanation of Boosting (Optional Reading)

In many real scenarios, Adaboost has shown an interesting behavior. In such scenarios, when we plot both the training error and test error versus the number of boosting rounds, T, we notice that the training error goes to zero and stays at zero as we increase T, and the test error (which is an approximate estimate for the true error based on a separate testing set) remains decreasing as we increase T even after the training error goes to zero before it eventually starts to rise. So, why is the test error keep decreasing with T (at least for a while)? Isn't this counter to the insight we get from the upper bound on the VC-dimension we derived above?

An elegant explanation for this behavior is due to [Schapire, Freund, Bartlett, Lee '97]. The explanation is shows the dependence of Adaboost on a quantity called *the margin*.
Define the margin on a data point (x, y) as

$$\mu(x, y) = y\sum_{t=1}^{T}\tilde{\alpha}_t h_t(x)$$

where for all $t \in [T]$, $\tilde{\alpha}_t$ is a normalized version of α_t in the Adaboost algorithm. That is,

$$\tilde{\alpha}_t = \frac{\alpha_t}{\sum_{k=1}^{T}\alpha_k}.$$

Note that, previously, what we cared about is the sign of $\mu(x_i, y_i)$ over the data points $(x_1, y_1), \ldots, (x_n, y_n)$. In particular, when $\mu(x, y) > 0$, this means that the final classifier (the weighted majority vote) correctly classifies x. It turns out that Adaboost behavior is tied to both the magnitude and the sign of $\mu(x_i, y_i)$, and not just the sign.

The magnitude of this quantity signifies how confident is the final classifier about the label it produces. This intuitively means that the larger this quantity is (given that it's positive), the better. But, how to nail down exactly the relatioship between Adaboost's performance and μ.

It was shown (e.g., the reference above) that Adaboost, in essence, is an iterative minimization algorithm, and the cost function it is trying to minimize is $\frac{1}{n}\sum_{i\in[n]} e^{-\mu(x_i,y_i)}$ over the choice of $h_1, \ldots, h_T$. (Note that, by defintion, $\mu(\cdot, \cdot)$ depends on $h_1, \ldots, h_T$).

So, even after all points $(x_1, y_1), \ldots, (x_n, y_n)$ are correctly classified and the training error becomes zero, (i.e., $\mu(x_i, y_i) > 0\ \forall i \in [n]$), if we increase T beyond that point then Adaboost would try to derive the above cost function down by increasing the magnitude of the margin $\mu(x_i, y_i)$ for all points $(x_i, y_i), i \in [n]$. But, what implication does this have on the generalization error.

The following result characterizes the relationship between the margin and the generalization error. It conforms with our intuition: larger margin $\Rightarrow$ better generalization error.

Theorem 10.7. Let $\mathcal{D}$ be any distribution on $\mathcal{X} \times \{-1, +1\}$. Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training set of n i.i.d. examples from $\mathcal{D}$. Let $\mathcal{H}$ denote the base hypothesis class where $VC(\mathcal{H}) = K$. Define

$$\tilde{\mathcal{H}} = \left\{ \tilde{h} : \forall x \in \mathcal{X}, \tilde{h}(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \, h_t(x) \right) \text{ and } \forall t \in [T], \alpha_t \geq 0 \text{ s.t. } \sum_{t \in [T]} \alpha_t = 1, h_t \in \mathcal{H} \right\}$$

Let $0 < \delta < 1$. Then, with probability at least $1 - \delta$ over the choice of S, for all $\theta > 0$ and all $\tilde{h} \in \tilde{\mathcal{H}}$, we have

$$\text{err}(\tilde{h}; \mathcal{D}) \leq \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\left(\mu(x_i, y_i) < \theta\right) + O\left(\sqrt{\frac{K \log^2(n/K) + \theta^2 \log(1/\delta)}{\theta^2 m}} \right).$$

CSE 5523: Machine Learning Theory Fall 2017

11 - Introduction to Convex Learning

Instructor: Raef Bassily

11.1 Generalized Framework for both Classification and Regression

Now we will discuss an important generalization to our learning framework that we will allow us to deal not only with classification problems but also regression problems.

As before, $\mathcal{X}$ is the domain set (i.e., a set of feature vectors). Usually, we will consider the case where $\mathcal{X} \subseteq \mathbb{R}^m$. Instead of restricting the label set $\mathcal{Y}$ to a finite set of labels, $\mathcal{Y}$ can now be an arbitrary set. Usually, we will consider the case $\mathcal{Y} \subseteq \mathbb{R}$. In this context, $\mathcal{Y}$ is called the "target set."

In this case, $\mathcal{X}$ and $\mathcal{Y}$ describe the components of the data, and the goal is to "learn" a functional relationship between the features ($x \in \mathcal{X}$) and the target ($y \in \mathcal{Y}$).

For example, we could have

$$x = (\text{heart rate}, \text{blood pressure}, \ldots) \in \mathbb{R}^m,$$
$$y = \text{person's weight} \in \mathbb{R}.$$

As before, the hypothesis class $\mathcal{H}$ is a collection of functions from $\mathcal{X}$ to $\mathcal{Y}$. Each hypothesis $h \in \mathcal{H}$ describes a *functional relationship* between features in $\mathcal{X}$ and target in $\mathcal{Y}$. We will be looking at classes where each hypothesis $h \in \mathcal{H}$ is described by a real vector $w \in \mathbb{R}^d$ (a parameter vector of d dimensions); that is,

$$w \in C \subseteq \mathbb{R}^d,$$
$$\mathcal{H} = \{h_w : w \in C\}.$$

Here, C is called the parameter set. Each hypothesis is described by a parameter vector $w \in C$. In this case, we will sometimes abuse notation and refer to $w \in C$ as the hypothesis and C as the hypothesis class.

Definition 11.1 (Loss function)**.** Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. We will consider loss functions of the form

$$\ell : C \times \mathcal{Z} \to \mathbb{R}_+.$$

That is, ℓ takes as inputs a hypothesis described by $w \in C$ and a data point $z = (x, y) \in Z$, and outputs a non-negative number that represents the loss (or cost) incurred by choosing such hypothesis to describe the functional relationship between x and y.

Example 11.2 (Ordinary Least Squares Regression)**.** Suppose we have

$$\mathcal{X} = \mathbb{R}^m,$$
$$\mathcal{Y} = \mathbb{R}.$$

Let $\tilde{x} = (x, 1) \in \mathbb{R}^{m+1}$ be a vector where the constant 1 has been appended to the end of $x \in \mathbb{R}^m$. Then, consider the hypothesis class

$$\mathcal{H} = \{h_w : \forall\, x \in \mathbb{R}^m, h_w(x) = \langle w, \tilde{x} \rangle, w \in \mathbb{R}^{m+1}\},$$

In such case, the parameter set $C = \mathbb{R}^{m+1}$ (i.e., the dimensionality is $d = m + 1$). Here, $\langle \cdot, \cdot \rangle$ is the inner product.

Our loss function is squared loss:

$$\ell : \underbrace{\mathbb{R}^{m+1}}_{C} \times \underbrace{\mathbb{R}^{m+1}}_{\mathcal{Z}} \to \mathbb{R}_+,$$
$$\ell(w, (x, y)) = (\langle w, \tilde{x} \rangle - y)^2.$$

Remark 11.3 (Empirical Risk Minimization, revisited)**.** If we are given a training set $S = \{(x_i, y_i)\}_{i=1}^n$, the ERM notion still holds:

$$w^* = \operatorname*{argmin}_{w \in \mathbb{R}^d} \widehat{L}(w; S),$$
$$\text{where} \quad \widehat{L}(w; S) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell\left(w; (x_i, y_i)\right).$$

This optimization problem is "easy," in the sense that it can be solved efficiently (i.e., via a an efficient algorithm). This is due to the fact that (i) C is a convex set, and (ii) for every $(x, y) \in \mathcal{Z}$, $\ell(\cdot, (x, y))$ is a convex function over C (and hence for a fixed training set S, the objective function $\widehat{L}(w; S)$ is convex in w).

This kind of optimization is known as *convex optimization*. Convex optimization problems can be solved efficiently. Namely, there is a collection of algorithms that can be used to find the unique minimum of the objective function (and a corresponding minimizer) relatively fast (precisely, the running time of such algorithms scales at most polynomially (sometimes, even linearly) with n and d).

Next, we will give a generalized version of the definition we saw before for agnostic PAC learnability, which would be suitable for this more general setting. Recall the general setup:

$\mathcal{X} \subseteq \mathbb{R}^m$,
$\mathcal{Y} \subseteq \mathbb{R}$,
$\mathcal{H} = \{h_w : \mathcal{X} \to \mathcal{Y}\}$, where each h_w is parameterized by $w \in C \subseteq \mathbb{R}^d$.

The loss function is of the form $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$ where $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$.

Definition 11.4 (Generalized Agnostic PAC Learnability)**.** A class $\mathcal{H}$ is agnostic PAC-learnable with respect to $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and the loss function ℓ in the generalized model if there is a function

$$n_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$$

and a learning algorithm A such that:

- for every $0 < \varepsilon, \delta < 1$, and
- for every distribution $\mathcal{D}$ over $\mathcal{Z}$,

when running A on a set S of $n \geq n_{\mathcal{H}}(\varepsilon, \delta)$ i.i.d training examples from $\mathcal{D}$, then w.p. $\geq 1 - \delta$, A returns a hypothesis $h_S \in \mathcal{H}$ with

$$\mathbb{E}_{z \sim \mathcal{D}}[\ell(h_S, z)] \leq \min_{h \in \mathcal{H}} \mathbb{E}_{z \sim \mathcal{D}}[\ell(h, z)] + \varepsilon.$$

Remark 11.5. Note that the quantity $\mathbb{E}_{z\sim\mathcal{D}}[\ell(h_S, z)]$ is a random variable, since it depends on S, and the expectation is not taken over the choice of S.

Remark 11.6. A few observations:

- Learnability is hard to reason about in this general model.
- The notion of VC dimension is not necessarily valid for regression problems, since it deals with set-shattering and binary labelings. There is a nice generalization for the notion of VC dimensions (known as Fat-Shattering dimension), however, it does not give full characterization of learnability and sample complexity like it was the case with the VC dimension in the binary classification setting. We will not discuss such notion in this course.
- Nonetheless, under certain restrictions on ℓ and $\mathcal{H}$, one can still show that learnability can be achieved.

Remark 11.7. We will study this generalized definition of learnability for certain family of learning problems that satisfy the following assumptions (hence the name *"convex learning"*):

- For all $z \in \mathcal{Z}$, $\ell(\cdot, z)$ is a convex loss function over the parameter set C.
- The parameter set C is a convex set.

11.2 Preliminaries on Convex Analysis

Definition 11.8 (Convex set)**.** A set C in a vector space is convex if for any $u, v \in C$, and any $\lambda \in [0, 1]$,

$$\lambda u + (1 - \lambda)v \in C.$$

In other words, convex combinations of points from C also lie in C. For example: line segments joining points from C also lie entirely in C.

Definition 11.9 (Convex function)**.** Let C be a convex set. A function $f : C \to \mathbb{R}$ is convex if for all $u, v \in C$ and for all $\lambda \in [0, 1]$,

$$f(\lambda u + (1 - \lambda)v) \leq \lambda f(u) + (1 - \lambda)f(v).$$

Furthermore, f is called "strictly convex" if this inequality is strict when $u \neq v$ and $\lambda \in (0, 1)$. (For example: x^2 is strictly convex but $|x|$ is not.)

Remark 11.10. An equivalent definition: $f : C \to \mathbb{R}$ is convex if for every $u \in C$ there is a vector in C denoted by $\partial f(u)$ s.t. for all $v \in C$, we have

$$f(v) \geq f(u) + \langle \partial f(u), v - u \rangle.$$

Then f is strictly convex if the inequality is strict for all $v \neq u$.

- This is analogous to the fact that the tangent line of a convex function always lies at or below the function itself.
- Also, when f is convex and differentiable, the first order Taylor expansion (the RHS of the inequality above) is always $\leq$ the function value.
- The vector $\partial f(u)$ is called a *subgradient* of f at u.

- There can be more than one subgradient at any given point. However, if f is differentiable, then there is a unique subgradient at any given point x, called the *gradient* of the function at this point $\nabla f(x)$.

The gradient of a differentiable function at point $x \in \mathbb{R}^d$ is given by

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_d} \end{bmatrix}.$$

Fact 11.11. Let C be a convex set. If $f : C \to \mathbb{R}$ is convex, then f has a unique minimum over C, but not necessarily a unique minimizer. (For example, the function may have a flat "trough" at the minimum.)

If the function is strictly convex, then there is a unique minimum and a unique minimizer.

Fact 11.12. Let $f_i : \mathbb{R}^d \to \mathbb{R}$ for $i = 1, \ldots, r$ be convex functions, then

- $g(x) = \max_{i \in [r]} f_i(x)$ is convex, and
- $g(x) = \sum_{i \in [r]} \gamma_i f_i(x)$, where $\gamma_i \geq 0$ for all i, is convex.

Given $f_1 : \mathbb{R}^d \to \mathbb{R}$ and $f_i : \mathbb{R} \to \mathbb{R}$ for $i = 2, \ldots, r$, the composition

$$f_r \circ f_{r-1} \circ \cdots \circ f_2 \circ f_1$$

is convex if each individual function is convex and *one* of the following conditions holds:

- $f_{r-1}, \ldots, f_1$ are affine,
- $f_r, f_{r-1}, \ldots, f_2$ are non-decreasing,
- there is a $j \in \{2, \ldots, r-1\}$ s.t. $f_{j-1}, \ldots f_1$ are affine, and $f_r, f_{r-1}, \ldots, f_{j+1}$ are non-decreasing.

Convex analysis: preliminaries

Convex set:

A subset C of a vector space is convex if for any $\mathbf{u}, \mathbf{v} \in C, \ \forall \ \lambda \in [0,1],$

$$\lambda \mathbf{u} + (1-\lambda)\mathbf{v} \in C$$

Convex function

Let C be a convex set. A fn. $f : C \to \mathbb{R}$ is convex if

$$\forall \ \mathbf{u}, \mathbf{v} \in C, \forall \ \lambda \in [0,1], \quad f(\lambda \mathbf{u} + (1-\lambda)\mathbf{v}) \quad \leq \quad \lambda f(\mathbf{u}) + (1-\lambda) f(\mathbf{v})$$

f is called *strictly convex* if the above inequality is strict for all $\lambda \in (0,1).$

Convex analysis: preliminaries

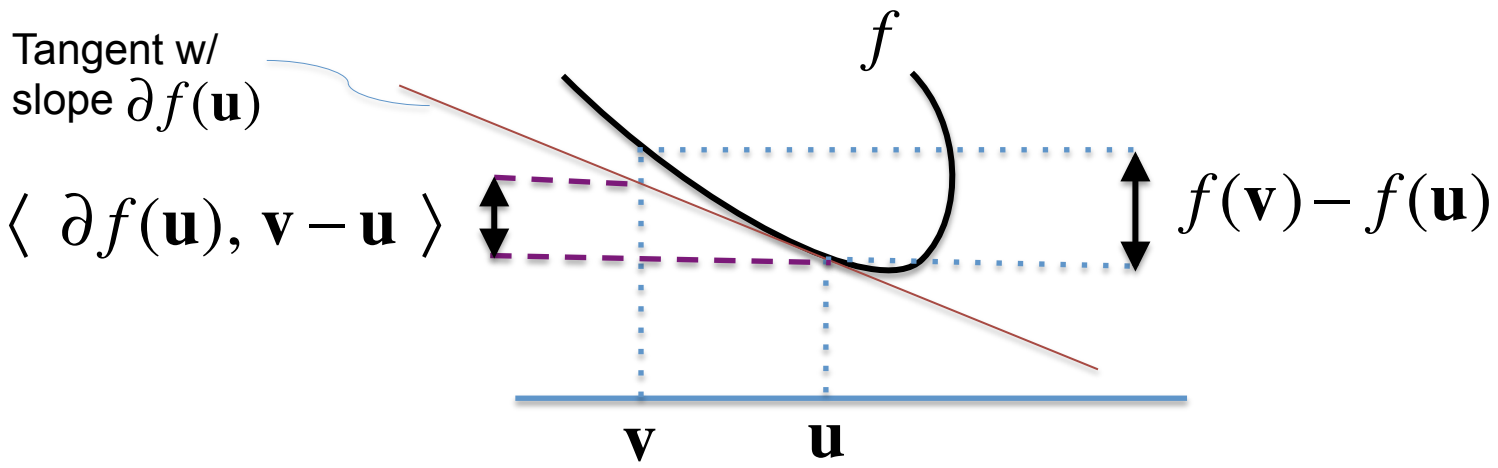
Convex function (equivalent definition)

Let C be a convex set. A fn. $f : C \to \mathbb{R}$ is convex if

$\forall\ \mathbf{u} \in C,\ \exists$ a vector $\partial f(\mathbf{u})$ such that $\forall\ \mathbf{v} \in C,$ we have

$$f(\mathbf{v}) \geq f(\mathbf{u}) + \langle\ \partial f(\mathbf{u}), \mathbf{v} - \mathbf{u}\ \rangle$$

f is called *strictly convex* if the above inequality is strict for all $\mathbf{v} \neq \mathbf{u}$



Convex functions – cont'd

- The definition says that *f* is convex if at any point there is a *hyperplane* **tangential to and lies below** the surface of *f.*
- $\partial f(\mathbf{u})$ is called a subgradient of *f* at $\mathbf{u}$.
- The subgradient of *f* at any given point is not necessarily unique.

  However, if *f* is **differentiable** over *C,* then at every point $\mathbf{u} \in C$ there is a **unique subgradient** for *f* denoted as $\nabla f(\mathbf{u})$.

Ex: $f(x) = |x|$, $x \in \mathbb{R}$ has infinitely many subgradients at $x = 0$.

- When f is differentiable over $C \subseteq \mathbb{R}^d$, then $\forall\ \mathbf{w} = (w_1, \ldots, w_d) \in C$, $\nabla f(\mathbf{w}) = \begin{bmatrix} \frac{\partial f(\mathbf{w})}{\partial w_1} \\ \cdot \\ \frac{\partial f(\mathbf{w})}{\partial w_d} \end{bmatrix}$

Convex functions – cont'd

Minimum/minimizer of a convex function:

Let *C* be a convex set. If $f : C \to \mathbb{R}$ is a convex function, then f has a unique minimum over *C* (*but not necessarily unique minimizer*). If f is strictly convex over *C,* then the minimizer of over *C* is also unique.

Convexity-preserving operations

Let $f_i : \mathbb{R}^d \to \mathbb{R}, \quad i = 1, \ldots, r$ be convex functions, then

- $g(x) \triangleq \max_{i \in [r]} f_i(x)$ is convex
- $g(x) = \sum_{i \in [r]} \gamma_i f_i(x)$, where $\forall i, \quad \gamma_i \geq 0,$ is convex.

Convexity-preserving operations

Composition of convex functions:

Let $f_1 : \mathbb{R}^d \to \mathbb{R}$ and $f_2 : \mathbb{R} \to \mathbb{R}$ be convex functions. Then, the composition $f_2 \circ f_1$ defined as $f_2 \circ f_1(\mathbf{w}) = f_2\left(f_1\left(\mathbf{w}\right)\right)$ is convex if **either** one of the following conditions holds:

- f_1 is affine (i.e., linear with a constant term, e.g., $f(\mathbf{w}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$)
- f_2 is non-decreasing.

One can easily generalize this to more than two functions:

$f_1 : \mathbb{R}^d \to \mathbb{R},\ f_i : \mathbb{R} \to \mathbb{R},\ i = 2,\ldots,r$

$f_r \circ f_{r-1} \circ \ldots \circ f_j \circ f_{j-1} \circ \ldots \circ f_1$ is convex if each individual function is convex, and **either** one of the following conditions holds:

- $f_{r-1}, \ldots, f_1$ are affine
- $f_r, f_{r-1}, \ldots, f_2$ are non-decreasing
- there is a $j \in \{2,\ldots,r-1\}$ s.t. $f_{j-1},\ldots,f_1$ are affine, and $f_r, f_{r-1}, \ldots, f_{j+1}$ are non-decreasing.

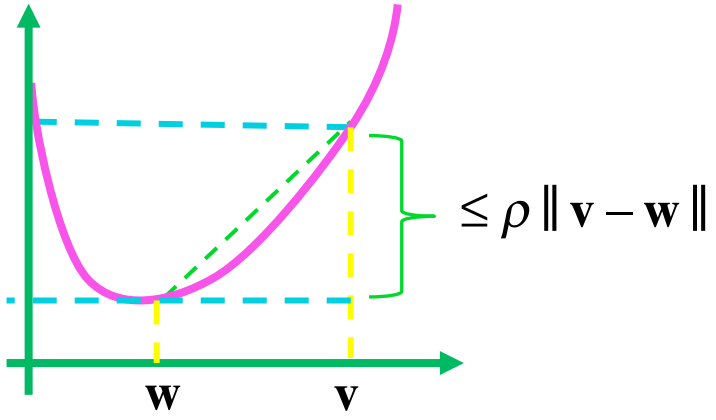
Lipschitz functions

Definition: (ρ-Lipschitz function)

A function $f : \mathbb{R}^d \to \mathbb{R}$ is ρ-Lipschitz (w.r.t. L_2 norm) if for all $\mathbf{v}, \mathbf{w} \in \mathbb{R}^d$,

$$|f(\mathbf{v}) - f(\mathbf{w})| \leq \rho \, \| \mathbf{v} - \mathbf{w} \|$$

(From now on, we will use $\| \cdot \|$ to denote the L_2 norm.)



Remark:

When $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable over its domain, then ρ-Lipschitzness of f is equivalent to: $\forall \mathbf{w} \in \mathbb{R}^d, \; \|\nabla f(\mathbf{w})\| \leq \rho$

Fact: Let $f(\mathbf{w}) = g_2\left(g_1(\mathbf{w})\right) \; \forall \mathbf{w}$. If g_1 is ρ_1-Lipschitz and g_2 is ρ_2-Lipschitz, then f is $\rho_1\rho_2$-Lipschitz.

Learnability of convex-Lipschitz-bounded problems

Theorem:

In the convex learning model, a hypothesis class $\mathcal{H}$ (described by a parameter set $C \subset \mathbb{R}^d$) is learnable (in the sense of the generalized agnostic-PAC learnability) if **all** the following conditions hold:

- The parameter set C is convex and M-bounded for some $M > 0$, (i.e., $\exists\, M > 0$ s.t. $\forall \mathbf{v}, \mathbf{w} \in C,\ \|\mathbf{v} - \mathbf{w}\| \leq M$).
- The loss function $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$ satisfies the following:
  $\exists\, \rho > 0$ s.t. $\forall z \in \mathcal{Z},\ \ell(\cdot\,, z)$ is **convex** and ρ**-Lipschitz.**

Under these conditions, $\mathcal{H}$ is learnable via an **efficient convex optimization** algorithm that requires a training set of size

$$n_{\mathcal{H}}(\epsilon, \delta) = O\left(\frac{M^2 \rho^2}{\epsilon^2 \delta^2}\right)$$

Examples of Convex-Lipschitz loss

- **Hinge loss:**

$$\ell\big(\mathbf{w},(\mathbf{x},y)\big) = \max\big(\ 0\ ,\ 1 - y\langle \mathbf{w}, \tilde{\mathbf{x}}\rangle\ \big)$$

where $\mathbf{x} \in \mathbb{R}^m$, $\tilde{\mathbf{x}} \triangleq (\mathbf{x}\ ,\ 1)$, $y \in \{-1,\ +1\}$

$$\mathbf{w} \in \mathbb{R}^{m+1}$$

Let's define

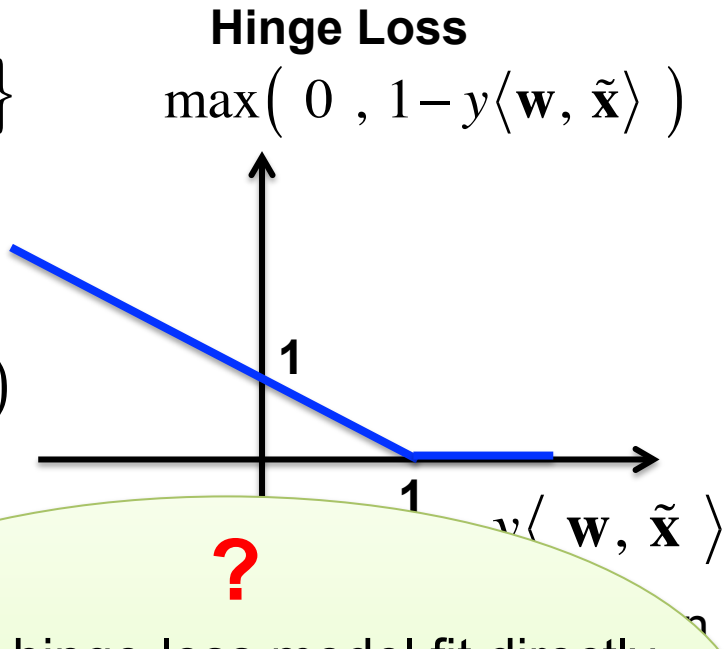
$$g_1(\mathbf{w}) \triangleq y\langle\ \mathbf{w},\ \tilde{\mathbf{x}}\ \rangle\ ,\ g_2(a) = \max(\ 0\ ,\ 1-a\ )$$

Note $\ell(\ \cdot\ ,\ (\mathbf{x},y)) = g_2 \circ g_1$

g_1 is linear and $\|\tilde{\mathbf{x}}\|$-Lipschitz.

g_2 is convex and 1-Lipschitz.

Hence, $\ell\big(\ \cdot\ ,\ (\mathbf{x},y)\big)$ is convex and $\|\tilde{\mathbf{x}}\|$-Lipschitz.



?

Does this hinge-loss model fit directly into the convex-Lipschitz-bounded learning model, or are there extra conditions that must be satisfied?

Examples of Convex-Lipschitz loss

- **Logistic loss:**

$$\ell\big(\mathbf{w},(\mathbf{x},y)\big)=\log\Big(1+\exp\big(-y\langle\mathbf{w},\tilde{\mathbf{x}}\rangle\big)\Big)$$

where $\mathbf{x}\in\mathbb{R}^m$, $\tilde{\mathbf{x}}\triangleq(\mathbf{x}\ ,\ 1)$, $y\in\{-1,\ +1\}$

$$\mathbf{w}\in\mathbb{R}^{m+1}$$

Let's define

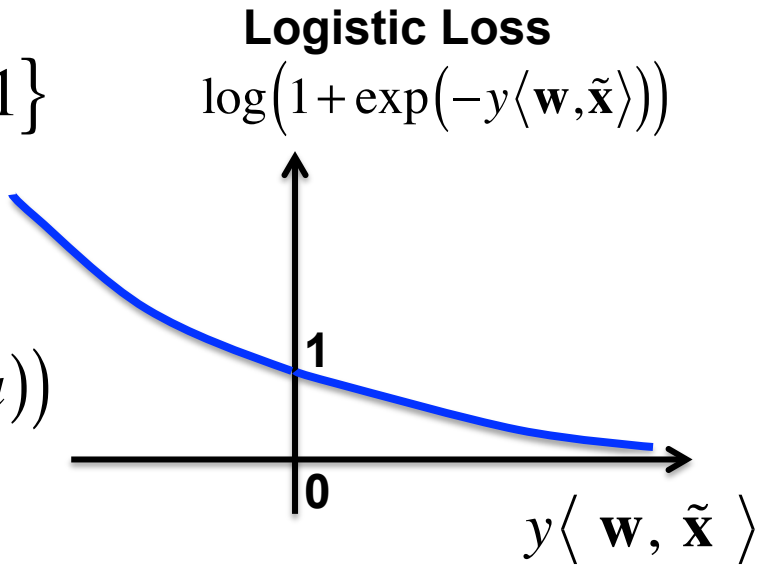
$$g_1(\mathbf{w})\triangleq y\langle\ \mathbf{w},\ \tilde{\mathbf{x}}\ \rangle\ ,\ g_2(a)=\log\big(1+\exp(-a)\big)$$

Note $\ell(\ \cdot\ ,(\mathbf{x},y))=g_2\circ g_1$

g_1 is linear and $\|\tilde{\mathbf{x}}\|$-Lipschitz.

g_2 is convex and 1-Lipschitz.

Hence, $\ell\big(\ \cdot\ ,(\mathbf{x},y)\big)$ is convex and $\|\tilde{\mathbf{x}}\|$-Lipschitz.



- Soft-threshold loss for linear classification (Smoother than hinge loss).
- Used in logistic regression (a model for linear classification).

The Gradient Descent Algorithm

Outline:

$C = \mathbb{R}^d$

We will first consider the general form of the **basic** (non-stochastic) GD algorithm for **unconstrained** convex optimization of Lipschitz functions (not specifically in the learning context).

Then, we will consider a more advanced algorithm:

- **Constrained** optimization: $C \subset \mathbb{R}^d$
- **Stochastic** GD

By a direct instantiation of the SGD algorithm, we obtain an efficient learner in the convex-Lipschitz-bounded model.

Basic GD algorithm (non-stochastic, unconstrained optimization)

- **Inputs:**

➢ A convex ρ-**Lipschitz** function: $f : \mathbb{R}^d \rightarrow \mathbb{R}$

(for simplicity, **we will assum**e that f is **differentiable**, i.e., the gradient $\nabla f(\mathbf{w})$ exists for all $\mathbf{w} \in \mathbb{R}^d$. Note that this is not necessary: we can use a sub-gradient $\partial f(\mathbf{w})$ instead of $\nabla f(\mathbf{w})$ whenever f is non-differentiable.)

➢ Number of iterations: T

➢ Set of scalars: $\{\alpha_t : t = 1, \ldots, T-1\}$ (α_t called the ***learning rate***).

Basic GD algorithm (non-stochastic, unconstrained optimization)

- **Inputs:**

➢ A convex ρ-**Lipschitz** function: $f : \mathbb{R}^d \rightarrow \mathbb{R}$

(for simplicity, **we will assume** that f is **differentiable**, i.e., the gradient $\nabla f(\mathbf{w})$ exists for all $\mathbf{w} \in \mathbb{R}^d$. Note that this is not necessary: we can use a sub-gradient $\partial f(\mathbf{w})$ instead of $\nabla f(\mathbf{w})$ whenever f is non-differentiable.)

➢ Number of iterations: T

➢ Set of scalars: $\{\alpha_t : t = 1, \ldots, T-1\}$ (α_t called the ***learning rate***).

- **Output:**

An estimate $\hat{\mathbf{w}}$ of $\mathbf{w}^*$ where $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w})$

Basic GD algorithm (non-stochastic, unconstrained optimization)

- **Inputs:** $f : \mathbb{R}^d \to \mathbb{R}$, T, $\{\alpha_t : t = 1, \ldots, T-1\}$

1. Initialization: Choose an initial point $\mathbf{w}_1 = \mathbf{0}$ (all-zero vector in $\mathbb{R}^d$)

2. **FOR** *t = 1, …, T – 1*

   Take a GD step: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \nabla f(\mathbf{w}_t)$

3. **Return** $\hat{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$

Basic GD algorithm (non-stochastic, unconstrained optimization)

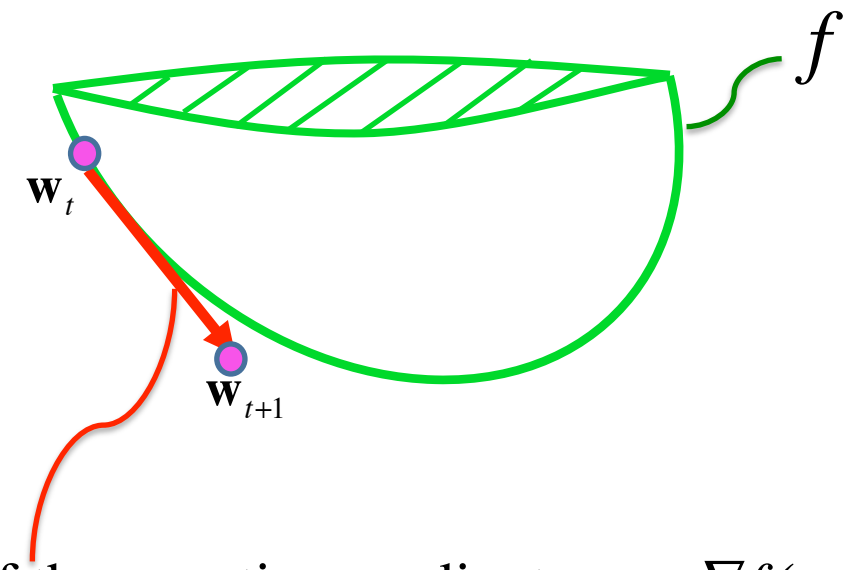
- **Inputs:** $f : \mathbb{R}^d \to \mathbb{R}$, T, $\{\alpha_t : t = 1, \ldots, T-1\}$

1. Initialization: Choose an initial point $\mathbf{w}_1 = \mathbf{0}$ (all-zero vector in $\mathbb{R}^d$)

2. **FOR** *t = 1, …, T – 1*

   Take a GD step: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \nabla f(\mathbf{w}_t)$

3. **Return** $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$



Basic GD algorithm (non-stochastic, unconstrained optimization)

- **Inputs:** $f : \mathbb{R}^d \to \mathbb{R}$, T, $\{\alpha_t : t = 1, \ldots, T-1\}$

1. Initialization: Choose an initial point $\mathbf{w}_1 = \mathbf{0}$ (all-zero vector in $\mathbb{R}^d$)

2. **FOR** *t = 1, …, T – 1*

   Take a GD step: $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t \nabla f(\mathbf{w}_t)$

3. **Return** $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

Theorem: (Convergence of Basic GD Algorithm)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex, ρ-Lipschitz function. Let $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w})$ and suppose that $\| \mathbf{w}^* \| \leq M$. If we run the GD algorithm above for ***T*** steps with $\alpha_t = \alpha = \frac{M}{\rho\sqrt{T}}, \forall t$. Then, $f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \leq \frac{M\rho}{\sqrt{T}}$

Basic GD algorithm: Proof of the main theorem

Theorem: (Convergence of Basic GD Algorithm)

Let $f:\mathbb{R}^d \to \mathbb{R}$ be a convex, ρ-Lipschitz function. Let $\mathbf{w}^* \in \arg\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w})$ and suppose that $\|\mathbf{w}^*\| \le M$. If we run the GD algorithm above for T steps with $\alpha_t = \alpha = M/\left(\rho\sqrt{T}\right)$. Then, $f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \le M\rho/\sqrt{T}$

First, let's define $\psi_t = \left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2, \ \forall t$ (ψ_t usually called the potential).

Basic GD algorithm: Proof of the main theorem

Theorem: (Convergence of Basic GD Algorithm)

Let $f:\mathbb{R}^d \to \mathbb{R}$ be a convex, ρ-Lipschitz function. Let $\mathbf{w}^* \in \arg\min_{\mathbf{w}\in\mathbb{R}^d} f(\mathbf{w})$ and suppose that $\|\mathbf{w}^*\| \le M$. If we run the GD algorithm above for T steps with $\alpha_t = \alpha = M/\left(\rho\sqrt{T}\right)$. Then, $f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \le M\rho/\sqrt{T}$

First, let's define $\psi_t = \left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2, \ \forall t$ (ψ_t usually called the potential).

We prove the theorem using the following claims:

Claim 1: $\forall t, \ \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \le \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2: $\forall t, \ \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \ge f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Basic GD algorithm: Proof of the main theorem

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Basic GD algorithm: Proof of the main theorem

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Combining the two claims: $\forall t,\ f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$ **(#)**

Basic GD algorithm: Proof of the main theorem

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Combining the two claims: $\forall t,\ f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$ **(#)**

Given those claims are true, now observe

$$f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \leq \frac{1}{T}\sum_{t=1}^{T}\left(f(\mathbf{w}_t) - f(\mathbf{w}^*)\right)$$

By convexity of f and the definition of $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$

Basic GD algorithm: Proof of the main theorem

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Combining the two claims: $\forall t,\ f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$ **(#)**

Given those claims are true, now observe

$$f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \leq \frac{1}{T}\sum_{t=1}^{T}\left(f(\mathbf{w}_t) - f(\mathbf{w}^*)\right)$$

$$= \frac{1}{2\alpha T}\sum_{t=1}^{T}\left(\psi_t - \psi_{t+1}\right) + \frac{\alpha\rho^2}{2}$$

By convexity of f and the definition of $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$

Follows directly from **(#)**

Basic GD algorithm: Proof of the main theorem

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Combining the two claims: $\forall t,\ f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$ **(#)**

Given those claims are true, now observe

$$f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \leq \frac{1}{T}\sum_{t=1}^{T}\left(f(\mathbf{w}_t) - f(\mathbf{w}^*)\right)$$

$$= \frac{1}{2\alpha T}\sum_{t=1}^{T}\left(\psi_t - \psi_{t+1}\right) + \frac{\alpha\rho^2}{2}$$

$$= \frac{1}{2\alpha T}\left(\psi_1 - \psi_{T+1}\right) + \frac{\alpha\rho^2}{2}$$

By convexity of f and the definition of $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$

Follows directly from **(#)**

Telescopic sum: terms cancel.

Basic GD algorithm: Proof of the main theorem

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Combining the two claims: $\forall t,\ f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$ **(#)**

Given those claims are true, now observe

$$f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \leq \frac{1}{T}\sum_{t=1}^{T}\left(f(\mathbf{w}_t) - f(\mathbf{w}^*)\right)$$

$$= \frac{1}{2\alpha T}\sum_{t=1}^{T}\left(\psi_t - \psi_{t+1}\right) + \frac{\alpha\rho^2}{2}$$

$$= \frac{1}{2\alpha T}\left(\psi_1 - \psi_{T+1}\right) + \frac{\alpha\rho^2}{2}$$

$$\leq \frac{\psi_1}{2\alpha T} + \frac{\alpha\rho^2}{2} \leq \frac{M^2}{2\alpha T} + \frac{\alpha\rho^2}{2} = \frac{M\rho}{\sqrt{T}}$$

By convexity of f and the definition of $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$

Follows directly from **(#)**

Telescopic sum: terms cancel.

2nd ineq.: $\psi_1 = \left\|\mathbf{w}^*\right\|^2 \leq M^2$,
Last equality: by substitution with $\alpha = M/(\rho\sqrt{T})$

Basic GD algorithm: Proof of the main theorem

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Proof :

$$\psi_{t+1} = \left\|\mathbf{w}_{t+1} - \mathbf{w}^*\right\|^2 = \left\|\mathbf{w}_t - \alpha\, \nabla f(\mathbf{w}_t) - \mathbf{w}^*\right\|^2$$

$$= \left\langle \mathbf{w}_t - \mathbf{w}^* - \alpha\, \nabla f(\mathbf{w}_t)\, ,\, \mathbf{w}_t - \mathbf{w}^* - \alpha\, \nabla f(\mathbf{w}_t) \right\rangle$$

$$= \left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2 - 2\alpha \left\langle \nabla f(\mathbf{w}_t)\, ,\, \mathbf{w}_t - \mathbf{w}^* \right\rangle + \alpha^2 \left\|\nabla f(\mathbf{w}_t)\right\|^2$$

$$\leq \psi_t - 2\alpha \left\langle \nabla f(\mathbf{w}_t)\, ,\, \mathbf{w}_t - \mathbf{w}^* \right\rangle + \alpha^2\rho^2$$

since $\left\|\nabla f(\mathbf{w}_t)\right\| \leq \rho$ by ρ-Lipschitzness of f

By rearranging the terms, the proof is complete.

Basic GD algorithm: Proof of the main theorem

Claim 2: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$

Proof :

By convexity of f (recall the equivalent definition of convex functions):

$$f(\mathbf{w}^*) \geq f(\mathbf{w}_t) + \left\langle \nabla f(\mathbf{w}_t) , \mathbf{w}^* - \mathbf{w}_t \right\rangle$$

$$= f(\mathbf{w}_t) - \left\langle \nabla f(\mathbf{w}_t) , \mathbf{w}_t - \mathbf{w}^* \right\rangle$$

Note the change in the sign in the second term inside the inner-product.

By rearranging the terms, the proof is complete.

We just proved:

Theorem: (Convergence of Basic GD Algorithm)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex, ρ-Lipschitz function. Let $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w})$ and suppose that $\| \mathbf{w}^* \| \leq M$. If we run the GD algorithm above for $\boldsymbol{T}$ steps with $\alpha_t = \alpha = M / \left(\rho \sqrt{T} \right)$. Then, $f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \leq M \rho / \sqrt{T}$

Recall: *Learnability of convex-Lipschitz-bounded problems*

Theorem:

In the convex learning model, a hypothesis class $\mathcal{H}$ (described by a parameter set $C \subset \mathbb{R}^d$) is learnable (in the sense of the generalized agnostic-PAC learnability) if **all** the following conditions hold:

- The parameter set C is convex and M-bounded for some $M > 0$, (i.e., $\exists\, M > 0$ s.t. $\forall \mathbf{v}, \mathbf{w} \in C,\ \|\mathbf{v} - \mathbf{w}\| \leq M$).
- The loss function $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$ satisfies the following:
  $\exists\, \rho > 0$ s.t. $\forall z \in \mathcal{Z},\ \ell(\cdot\,, z)$ is **convex** and ρ**-Lipschitz.**

Under these conditions, $\mathcal{H}$ is learnable via an **efficient convex optimization** algorithm that requires a training set of size

$$n_{\mathcal{H}}(\epsilon, \delta) = O\left(\frac{M^2 \rho^2}{\epsilon^2 \delta^2}\right)$$

Stochastic Gradient Descent

Outline:

We will first consider the general form of the **basic** (non-stochastic) GD algorithm for **unconstrained** convex optimization of Lipschitz functions (not specifically in the learning context). ✔

Then, we will consider a more advanced algorithm:

- **Constrained** optimization: $C \subset \mathbb{R}^d$
- **Stochastic** GD

By a direct instantiation of the SGD algorithm, we obtain an efficient learner in the convex-Lipschitz-bounded model.

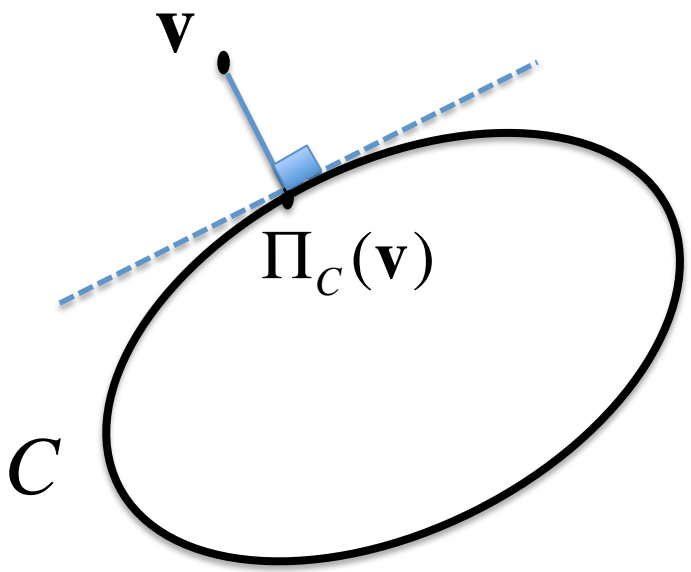
Euclidean projection

Definition: (Euclidean Projection)

Let $C \subseteq \mathbb{R}^d$ be a closed convex set. The Euclidean projection $\Pi_C : \mathbb{R}^d \to C$ is defined as:

$$\forall \mathbf{v} \in \mathbb{R}^d, \ \Pi_C(\mathbf{v}) = \arg\min_{\mathbf{w} \in C} \| \mathbf{v} - \mathbf{w} \|$$

That is, $\Pi_C(\mathbf{v})$ is the "closest" point in C (w.r.t. the Euclidean distance) to $\mathbf{v}$



Euclidean projection

Fact:

$\forall \mathbf{v} \in \mathbb{R}^d,\ \forall \mathbf{w} \in C,$ we have $\|\mathbf{v}-\mathbf{w}\| \geq \|\Pi_C(\mathbf{v})-\mathbf{w}\|$

Proof:

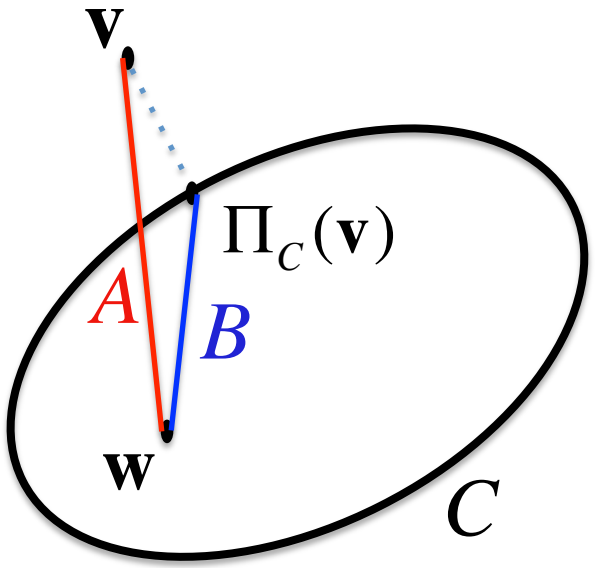
$$\|\mathbf{v}-\mathbf{w}\|^2 = \|\mathbf{v}-\Pi_C(\mathbf{v})+\Pi_C(\mathbf{v})-\mathbf{w}\|^2$$

$$= \|\Pi_C(\mathbf{v})-\mathbf{w}\|^2 + 2\langle \mathbf{v}-\Pi_C(\mathbf{v}),\ \Pi_C(\mathbf{v})-\mathbf{w}\rangle + \|\mathbf{v}-\Pi_C(\mathbf{v})\|^2$$

$$\leq \|\Pi_C(\mathbf{v})-\mathbf{w}\|^2 + 2\langle \mathbf{v}-\Pi_C(\mathbf{v}),\ \Pi_C(\mathbf{v})-\mathbf{w}\rangle$$

The following claim completes the proof

Claim: $\langle \mathbf{v}-\Pi_C(\mathbf{v}),\ \Pi_C(\mathbf{v})-\mathbf{w}\rangle \geq 0$



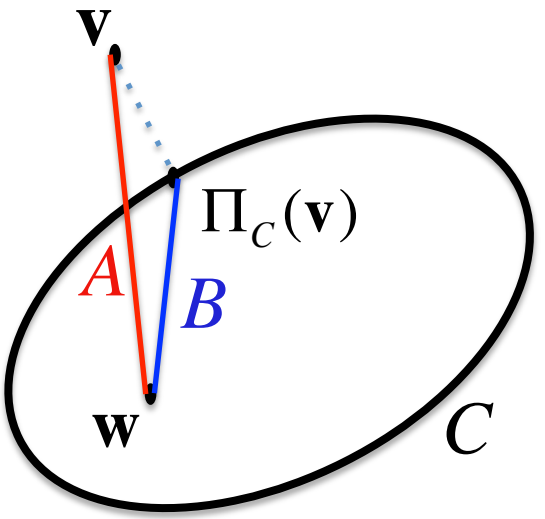
$A \geq B$ for any $\mathbf{w} \in C$

Euclidean projection

Proof of the claim: $\langle \mathbf{v} - \Pi_C(\mathbf{v}),\ \Pi_C(\mathbf{v}) - \mathbf{w} \rangle \geq 0$

Since ***C*** is a convex set, then for any $\lambda \in (0,\ 1)$

$$\underbrace{(1-\lambda)\ \Pi_C(\mathbf{v}) + \lambda\ \mathbf{w}}_{\mathbf{u}_\lambda} \in C$$



$A \geq B$ for any $\mathbf{w} \in C$

By the definition of $\Pi_C(\mathbf{v})$

$$\| \Pi_C(\mathbf{v}) - \mathbf{v} \|^2 \ \leq\ \| \mathbf{u}_\lambda - \mathbf{v} \|^2$$

$$= \| \Pi_C(\mathbf{v}) - \mathbf{v} \|^2 + 2\lambda \langle \Pi_C(\mathbf{v}) - \mathbf{v},\ \mathbf{w} - \Pi_C(\mathbf{v}) \rangle + \lambda^2 \| \Pi_C(\mathbf{v}) - \mathbf{w} \|^2$$

Rearranging: $\langle \Pi_C(\mathbf{v}) - \mathbf{v},\ \mathbf{w} - \Pi_C(\mathbf{v}) \rangle \ \geq\ -\frac{\lambda}{2} \| \Pi_C(\mathbf{v}) - \mathbf{w} \|^2$

Taking the limit as $\lambda \to 0$, we reach the desired result.

GD for constrained optimization: Projection step

- **Inputs:**
  - A convex ρ-**Lipschitz** function: $f : \mathbb{R}^d \to \mathbb{R}$
  - A convex constraint set: $C \subset \mathbb{R}^d$
  - Number of iterations: T
  - Set of scalars: $\{\alpha_t : t = 1, \ldots, T-1\}$ (the ***learning rate***).

- **Output:**

An estimate $\hat{\mathbf{w}}$ of $\mathbf{w}^*$ where $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in C} f(\mathbf{w})$

GD for constrained optimization: Projection step

- **Inputs:** $f,\ C,\ T,\ \{\alpha_t : t = 1,\ \ldots,\ T-1\}$

1. Initialization: $\mathbf{w}_1 = \mathbf{0}$ (assume all-zero vector is in C)

2. **FOR** *t = 1, …, T – 1*

   Take a projected GD step: $\mathbf{w}_{t+1} = \Pi_C\left(\mathbf{w}_t - \alpha_t\ \nabla f(\mathbf{w}_t)\right)$

3. **Return** $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$

Theorem: (Convergence of Projected GD Algorithm)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex, ρ-Lipschitz function. Let $C \subset \mathbb{R}^d$ be a closed convex set. Let $\mathbf{w}^* \in \arg\min_{\mathbf{w}\in C} f(\mathbf{w})$ where $\|\mathbf{w}^*\| \le M$. If we run the projected GD algorithm above for ***T*** steps with $\alpha_t = \alpha = \frac{M}{\rho\sqrt{T}}, \forall t.$

Then, $f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \le M\rho/\sqrt{T}$

GD for constrained optimization: Projection step

- **Inputs:** f C T $\{\alpha_t : t = 1, \ldots, T-1\}$

1. ...e all-zero vector is in C)

Note that if C is M-bounded, then this directly implies that $\| \mathbf{w}^* - \mathbf{w}_1 \| \leq M$

$= \Pi_C \left(\mathbf{w}_t - \alpha \right.$

3. ...

In general, $\| \mathbf{w}^* - \mathbf{w}_1 \| \leq M$

Theorem: (Convergence of Projected GD Algorithm)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex, ρ-Lipschitz function. Let $C \subset \mathbb{R}^d$ be a closed convex set. Let $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in C} f(\mathbf{w})$ where $\| \mathbf{w}^* \| \leq M$. If we run the projected GD algorithm above for T steps with $\alpha_t = \alpha = \frac{M}{\rho\sqrt{T}}, \forall t.$

Then, $f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \leq M\rho / \sqrt{T}$

GD for constrained optimization: Projection step

Theorem: (Convergence of Projected GD Algorithm)

Let $f:\mathbb{R}^d \to \mathbb{R}$ be a convex, ρ-Lipschitz function. Let $C \subset \mathbb{R}^d$ be a closed convex set. Let $\mathbf{w}^* \in \arg\min_{\mathbf{w}\in C} f(\mathbf{w})$ where $\|\mathbf{w}^*\| \le M$. If we run the projected GD algorithm above for T steps with $\alpha_t = \alpha = \frac{M}{\rho\sqrt{T}}, \forall t.$

Then, $f(\hat{\mathbf{w}}) - f(\mathbf{w}^*) \le M\rho/\sqrt{T}$

As before, define $\psi_t = \left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2, \ \forall t$

Recall the two claims used in the proof of the basic GD:

Claim 1: $\forall t, \ \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \le \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

remains valid (with an extra step in the proof)

Claim 2: $\forall t, \ \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \ge f(\mathbf{w}_t) - f(\mathbf{w}^*)$

remains valid (same proof: convexity of f)

Projected GD algorithm: small modification in the proof

Claim 1: $\forall t, \langle \nabla f(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}^* \rangle \leq \frac{\psi_t - \psi_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Proof :

$$\psi_{t+1} = \left\|\mathbf{w}_{t+1} - \mathbf{w}^*\right\|^2 = \left\|\Pi_C\left(\mathbf{w}_t - \alpha\ \nabla f(\mathbf{w}_t)\right) - \mathbf{w}^*\right\|^2$$

$$\leq \left\|\mathbf{w}_t - \alpha\ \nabla f(\mathbf{w}_t) - \mathbf{w}^*\right\|^2$$

$$\vdots$$

Follows from the **Fact:** the property of Euclidean projections we just proved (since $\mathbf{w}^* \in C$)

Continue the proof exactly as before.

Stochastic Gradient Descent

- **Stochastic Gradients:**
  - Does not require that the direction taken in each iteration to be exactly equal to the (negative of the) gradient
  - Instead the direction can be a random vector whose expected is equal to the (negative of the) gradient.

In each iteration *t = 1, …, T, given that the current parameter value is* $\mathbf{w}_t$, a step will be taken in a random direction $-G_t$ where the random vector G_t satisfies: $\mathbb{E}\left[G_t \middle| \mathbf{w}_t\right] = \nabla f(\mathbf{w}_t)$ where $\nabla f(\mathbf{w}_t)$ is the true gradient at $\mathbf{w}_t$.

- Instead of having f given to the algorithm as an input, the algorithm will be given access to an "oracle" that given an input $\mathbf{w} \in \mathbb{R}^d$, it outputs a random vector G_t with the property: $\mathbb{E}\left[G_t \middle| \mathbf{w}_t\right] = \nabla f(\mathbf{w}_t)$
- The oracle knows f, but the algorithm does not need to. The oracle is a **"black-box"** (the algorithm does **not need** to know how it runs).

Stochastic Gradient Descent Algorithm

- **Inputs:** A gradient oracle (denoted as **G-Oracle),**
  Constraint set C ,
  Number of iterations T,
  Learning rate $\{\alpha_t : t = 1, \ldots, T-1\}$

1. Initialization: $\mathbf{w}_1 = \mathbf{0}$ (assume all-zero vector is in C)

2. **FOR** *t = 1, …, T – 1*

   a. Given $\mathbf{w}_t$, **G-Oracle($\mathbf{w}_t$)** generates a random vector G_t such that

$$\mathbb{E}\left[G_t \middle| \mathbf{w}_t \right] = \nabla f(\mathbf{w}_t)$$

   b. Take a projected step in direction of G_t : $\mathbf{w}_{t+1} = \Pi_C \left(\mathbf{w}_t - \alpha_t \ \mathrm{G}_t \right)$

3. **Return** $\hat{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$

SGD: Convergence guarantees

Theorem: (Convergence of SGD Algorithm)

Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function. Let $C \subset \mathbb{R}^d$ be a closed convex set.

Let $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in C} f(\mathbf{w})$ where $\| \mathbf{w}^* \| \leq M$. Suppose we run SGD algorithm for T steps with $\alpha_t = \alpha = \frac{M}{\rho\sqrt{T}}, \forall t$. Suppose that $\forall t, \|G_t\| \leq \rho$ with probability 1. Then, $\mathbb{E}[f(\hat{\mathbf{w}})] - f(\mathbf{w}^*) \leq \frac{M\rho}{\sqrt{T}}$

taken w.r.t. $\hat{\mathbf{w}}$ (as it depends on **all** $G_t, t = 1, \ldots, T-1$)

If C is M-bounded, then this will immediately follow

This implies that f is ρ-Lipschitz

Before we prove this theorem, we will first show an important implication.

Learning with SGD

Given a parameter set C, and a loss function $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$, define

$$L(\mathbf{w} \; ; \; D) \triangleq \mathbb{E}_{z \sim D}\left[\ell(\mathbf{w}, \, z)\right]$$

The true risk. Analogous to $err\left(h_{\mathbf{w}} \; ; \; D\right)$ in the standard PAC model (with 0-1 loss)

The excess risk incurred by an algorithm that uses a training set $S = \{z_1, \, \ldots, \, z_n\} \sim D^n$ to output a hypothesis $\hat{\mathbf{w}}_S \in C$ is given by

$$L\left(\hat{\mathbf{w}}_S \; ; \; D\right) - \min_{\mathbf{w} \in C} L(\mathbf{w} \; ; \; D)$$

Recall that learnability is tied to this quantity

Will show that SGD aims at directly minimizing the expected excess risk

$$\mathbb{E}_{S \sim D^n}\left[L\left(\hat{\mathbf{w}}_S \; ; \; D\right)\right] - \min_{\mathbf{w} \in C} L(\mathbf{w} \; ; \; D)$$

SGD Learner

- **Inputs:** A loss function $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$,
  A sample oracle: returns a fresh sample $z \sim D$
  Parameter set C,
  Number of iterations T,
  Learning rate $\{\alpha_t : t = 1, \ldots, T-1\}$

1. Initialization: $\mathbf{w}_1 = \mathbf{0}$ (assume all-zero vector is in C)

2. **FOR** *t = 1, …, T – 1*

   a. Draw a fresh example $z_t \sim D$

   b. Compute $G_t = \nabla \ell(\mathbf{w}_t, z_t)$

   c. Update: $\mathbf{w}_{t+1} = \Pi_C \left(\mathbf{w}_t - \alpha_t \, G_t \right)$

3. **Return** $\hat{\mathbf{w}}_S = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$ (where $S = \{z_1, \ldots, z_{T-1}\}$. The notation $\hat{\mathbf{w}}_S$ is to remind us that the output depends on S)

SGD Learner

- **Inputs:** A loss function $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$,
  A sample oracle: returns a fresh sample $z \sim D$
  Parameter set C,
  Number of iterations T,
  Learning rate $\{\alpha_t : t = 1, \ldots, T-1\}$

1. Initialization: $\mathbf{w}_1 = \mathbf{0}$

2. **FOR** *t = 1, …, T – 1*

   a. Draw a fresh example $z_t \sim D$

   b. Compute $G_t = \nabla \ell(\mathbf{w}_t, z_t)$

   c. Update: $\mathbf{w}_{t+1} = \Pi_C \left(\mathbf{w}_t - \alpha_t G_t \right)$

3. **Return** $\hat{\mathbf{w}}_S = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$

This plays the role of the **G-Oracle:** Randomness here is due to $z_t \sim D$

Note that at step *t*, given that the current point is $\mathbf{w}_t$, the random vector G_t **is given by** $\nabla \ell(\mathbf{w}_t, z_t)$

SGD Learner

- **Inputs:** A lo
  A

$$\mathbb{E}\left[G_t \middle| \mathbf{w}_t\right] = \mathbb{E}_{z_t \sim D}\left[\nabla \ell(\mathbf{w}_t, z_t)\right]$$

Conditioned on $\mathbf{w}_t$, the remaining randomness comes from z_t.
Note that $\mathbf{w}_t$ is independent of z_t.

1. Initializati

2. **FOR** *t = 1, …,*

   a. Draw a fresh example $z_t \sim D$

   b. Compute $G_t = \nabla \ell(\mathbf{w}_t, z_t)$

   c. Update: $\mathbf{w}_{t+1} = \Pi_C\left(\mathbf{w}_t - \alpha_t G_t\right)$

3. **Return** $\hat{\mathbf{w}}_S = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

SGD Learner

- **Inputs:** A lo
  A

$$\mathbb{E}\left[G_t \middle| \mathbf{w}_t\right] = \mathbb{E}_{z_t \sim D}\left[\nabla \ell(\mathbf{w}_t, z_t)\right]$$
$$= \nabla\, \mathbb{E}_{z_t \sim D}\left[\ell(\mathbf{w}_t, z_t)\right]$$

Swapping order of $\mathbb{E}$ (integration) w.r.t. z_t and ∇ (differentiation) w.r.t. $\mathbf{w}$ at $\mathbf{w}_t$ (Note again that $\mathbf{w}_t$ does not depend on z_t)

1. Initializati

2. **FOR** *t = 1, …,*

   a. Draw a fresh example $z_t \sim$

   b. Compute $G_t = \nabla \ell(\mathbf{w}_t, z_t)$

   c. Update: $\mathbf{w}_{t+1} = \Pi_C\left(\mathbf{w}_t - \alpha_t G_t\right)$

3. **Return** $\hat{\mathbf{w}}_S = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

SGD Learner

- **Inputs:** A lo
  A

$$\mathbb{E}\left[G_t \middle| \mathbf{w}_t\right] = \mathbb{E}_{z_t \sim D}\left[\nabla \ell(\mathbf{w}_t, z_t)\right]$$
$$= \nabla\, \mathbb{E}_{z_t \sim D}\left[\ell(\mathbf{w}_t, z_t)\right]$$
$$= \nabla \mathbb{E}_{z \sim D}\left[\ell(\mathbf{w}_t, z)\right]$$

Since z_t is a fresh sample that is independent of $\mathbf{w}_t$, the expectation will remain the same if we replace z_t by a fresh sample $z \sim D$.

1. Initializati

2. **FOR** *t = 1, …,*

   a. Draw a fresh example $z_t \sim D$

   b. Compute $G_t = \nabla \ell(\mathbf{w}_t, z_t)$

   c. Update: $\mathbf{w}_{t+1} = \Pi_C\left(\mathbf{w}_t - \alpha_t G_t\right)$

3. **Return** $\hat{\mathbf{w}}_S = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

SGD Learner

- **Inputs:** A lo
  A

$$\mathbb{E}\left[G_t \middle| \mathbf{w}_t\right] = \mathbb{E}_{z_t \sim D}\left[\nabla \ell(\mathbf{w}_t, z_t)\right]$$
$$= \nabla\ \mathbb{E}_{z_t \sim D}\left[\ell(\mathbf{w}_t, z_t)\right]$$
$$= \nabla \mathbb{E}_{z \sim D}\left[\ell(\mathbf{w}_t, z)\right]$$
$$= \nabla L(\mathbf{w}_t\ ; D)$$

By definition of the true risk $L\left(\mathbf{w}_t\ ; D\right)$

1. Initializati

2. **FOR** *t = 1, …,*

   a. Draw a fresh example $z_t \sim$

   b. Compute $G_t = \nabla \ell(\mathbf{w}_t, z_t)$

   c. Update: $\mathbf{w}_{t+1} = \Pi_C\left(\mathbf{w}_t - \alpha_t G_t\right)$

3. **Return** $\hat{\mathbf{w}}_S = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

SGD Learner

- **Inputs:** A lo
  A

1. Initializati

2. **FOR** *t = 1, …,*

   a. Draw a fresh example $z_t \sim$

   b. Compute $G_t = \nabla \ell(\mathbf{w}_t, z_t)$

   c. Update: $\mathbf{w}_{t+1} = \Pi_C\left(\mathbf{w}_t - \alpha_t G_t\right)$

3. **Return** $\hat{\mathbf{w}}_S = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$

Hence, we have

$$\mathbb{E}\left[G_t \middle| \mathbf{w}_t\right] = \nabla L(\mathbf{w}_t\ ;\ D)$$

That is, the objective function here is actually $L(\ \cdot\ ;\ D)$

i.e., here $L(\ \cdot\ ;\ D)$ is an instantiation of f in the generic SGD algorithm.

Learning with SGD

Theorem: (Convergence of the Generic SGD Algorithm)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. Let $C \subset \mathbb{R}^d$ be a closed convex set.

Let $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in C} f(\mathbf{w})$ where $\| \mathbf{w}^* \| \leq M$. Suppose we run SGD algorithm for T steps with $\alpha_t = M / (\rho\sqrt{T}), \forall t$. Suppose that $\forall t, \|G_t\| \leq \rho$ with probability 1.

Then, $\mathbb{E}[f(\hat{\mathbf{w}})] - f(\mathbf{w}^*) \leq M\rho / \sqrt{T}$

⇓

Corollary: (Excess Risk of the SGD Learner)

Consider a convex-Lipschitz-bounded model where the parameter set $C \subset \mathbb{R}^d$ is M-bounded and there is $\rho > 0$ such that for all $z \in \mathcal{Z}$, the loss function $\ell(\cdot\,; z)$ is convex and ρ-Lipschitz. Suppose we run the above **SGD learner** for T iterations (i.e., **number of examples =** T), and with $\alpha_t = M / (\rho\sqrt{T}), \forall t$.

Then, $$\mathbb{E}_{S \sim D^T}\left[L\left(\hat{\mathbf{w}}_S\,;\, D\right)\right] - \min_{\mathbf{w} \in C} L(\mathbf{w}\,;\, D) \leq \frac{M\rho}{\sqrt{T}}$$

Learning with SGD: Remarks

- The above SGD takes a rather direct approach towards learning.
- Instead of minimizing the empirical error $\hat{L}(\mathbf{w}\ ;\ S) \triangleq \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}\ ,\ z_i)$ over $\mathbf{w} \in C$ w.r.t. a training set $S = \{z_1,\ \ldots,\ z_n\}$, it seeks to minimize the true risk directly.
- This is done by taking a random update step at each iteration. In each iteration *t,* given a current parameter estimate $\mathbf{w}_t$, the update step is taken along $-\nabla \ell(\mathbf{w}_t,\ z_t)$ where z_t is a fresh sample from the unknown distribution.
- So, this instantiation of SGD is **not** an ERM algorithm.
- Note that the SGD learner is an online algorithm where training samples can arrive one by one (or in small batches) rather than requiring that the entire training set to be available in advance.

SGD: Convergence guarantees of generic SGD - Proof

Theorem: (Convergence of SGD Algorithm)

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. Let $C \subset \mathbb{R}^d$ be a closed convex set.

Let $\mathbf{w}^* \in \arg\min_{\mathbf{w} \in C} f(\mathbf{w})$ where $\| \mathbf{w}^* \| \leq M$. Suppose we run SGD algorithm for T steps with $\alpha_t = \alpha = \frac{M}{\rho\sqrt{T}}, \forall t$. Suppose that $\forall t, \left\| G_t \right\| \leq \rho$ with probability 1. Then, $\mathbb{E}\left[f(\hat{\mathbf{w}}) \right] - f(\mathbf{w}^*) \leq \frac{M\rho}{\sqrt{T}}$

Define $\tilde{\psi}_t = \mathbb{E}\left[\left\| \mathbf{w}_t - \mathbf{w}^* \right\|^2 \right], \forall t$

Claim 1*: $\forall t, \mathbb{E}\left[\langle \mathbf{G}_t, \mathbf{w}_t - \mathbf{w}^* \rangle \right] \leq \frac{\tilde{\psi}_t - \tilde{\psi}_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2*: $\forall t, \mathbb{E}\left[\langle \mathbf{G}_t, \mathbf{w}_t - \mathbf{w}^* \rangle \right] \geq \mathbb{E}\left[f(\mathbf{w}_t) \right] - f(\mathbf{w}^*)$

Given these claims, the rest of the proof follows similar steps as in the proof of the analogous theorem for the basic GD algorithm.

Generic SGD algorithm: Proof of convergence guarantees

Claim 1*: $\forall t,\ \mathbb{E}\left[\left\langle \mathrm{G}_t,\ \mathbf{w}_t - \mathbf{w}^* \right\rangle\right] \leq \frac{\tilde{\psi}_t - \tilde{\psi}_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Claim 2*: $\forall t,\ \mathbb{E}\left[\left\langle \mathrm{G}_t,\ \mathbf{w}_t - \mathbf{w}^* \right\rangle\right] \geq \mathbb{E}\left[f(\mathbf{w}_t)\right] - f(\mathbf{w}^*)$

The key to prove these claims is the following **fact:**

Fact: $\mathbb{E}\left[\left\langle \mathrm{G}_t,\ \mathbf{w}_t - \mathbf{w}^* \right\rangle\right] = \mathbb{E}\left[\left\langle \nabla f(\mathbf{w}_t),\ \mathbf{w}_t - \mathbf{w}^* \right\rangle\right]$

Generic SGD algorithm: Proof of convergence guarantees

Let's first prove this fact. Observe that by applying the rule of iterated expectation, we have

$$\mathbb{E}\Big[\langle\, \mathbf{G}_t\,,\, \mathbf{w}_t - \mathbf{w}^*\,\rangle\Big] = \mathbb{E}_{\mathbf{w}_t}\Big[\mathbb{E}_{G_t|\mathbf{w}_t}\Big[\langle\, G_t\,,\, \mathbf{w}_t - \mathbf{w}^*\,\rangle \Big| \mathbf{w}_t\Big]\Big]$$

Since we condition on $\mathbf{w}_t$ in the inner expectation, then randomness of $\mathbf{w}_t$ is fixed in that expectation. Hence, using linearity of expectation, the inner expectation becomes:

$$\mathbb{E}_{G_t|\mathbf{w}_t}\Big[\langle\, G_t\,,\, \mathbf{w}_t - \mathbf{w}^*\,\rangle \Big| G_1^{t-1}\Big] = \langle \mathbb{E}_{G_t|\mathbf{w}_t}\Big[G_t \Big| \mathbf{w}_t\Big]\,,\, \mathbf{w}_t - \mathbf{w}^*\,\rangle$$

By the unbiasedness property of the gradient oracle, we have

$$\mathbb{E}_{G_t|\mathbf{w}_t}\Big[G_t \Big| \mathbf{w}_t\Big] = \nabla f(\mathbf{w}_t)$$

By applying the outer expectation, we get: $\mathbb{E}\Big[\langle\, \nabla f(\mathbf{w}_t),\, \mathbf{w}_t - \mathbf{w}^*\,\rangle\Big]$

Generic SGD algorithm: Proof of convergence guarantees

Namely, we get

$$\mathbb{E}\Big[\langle\, \mathbf{G}_t\,,\, \mathbf{w}_t - \mathbf{w}^*\,\rangle\Big] = \mathbb{E}_{G_1^{t-1}}\Big[\langle\, \nabla f(\mathbf{w}_t)\,,\, \mathbf{w}_t - \mathbf{w}^*\,\rangle\Big]$$

$$= \mathbb{E}\Big[\langle\, \nabla f(\mathbf{w}_t)\,,\, \mathbf{w}_t - \mathbf{w}^*\,\rangle\Big]$$

Given this fact, the proofs of the Claims 1* and 2* should be straight forward as they follow similar lines to their counterparts for the basic GD.

For completeness, let's briefly go over their proofs.

Generic SGD algorithm: Proof of convergence guarantees

Claim 1*: $\forall t,\ \mathbb{E}\left[\left\langle \mathrm{G}_t,\ \mathbf{w}_t - \mathbf{w}^* \right\rangle\right] \leq \frac{\tilde{\psi}_t - \tilde{\psi}_{t+1}}{2\alpha} + \frac{\alpha\rho^2}{2}$

Proof :

$$\tilde{\psi}_{t+1} = \mathbb{E}\left[\left\|\mathbf{w}_{t+1} - \mathbf{w}^*\right\|^2\right] = \mathbb{E}\left[\left\|\Pi_C\left(\mathbf{w}_t - \alpha\ G_t\right) - \mathbf{w}^*\right\|^2\right]$$

$$\leq\ \mathbb{E}\left[\left\|\mathbf{w}_t - \alpha\ G_t - \mathbf{w}^*\right\|^2\right]$$

By the property of Euclidean projections that we proved

$$= \mathbb{E}\left[\left\langle \mathbf{w}_t - \mathbf{w}^* - \alpha\ G_t\ ,\ \mathbf{w}_t - \mathbf{w}^* - \alpha\ G_t \right\rangle\right]$$

$$= \mathbb{E}\left[\left\|\mathbf{w}_t - \mathbf{w}^*\right\|^2\right] - 2\alpha\ \mathbb{E}\left[\left\langle G_t\ ,\ \mathbf{w}_t - \mathbf{w}^*\ \right\rangle\right] + \alpha^2\mathbb{E}\left[\left\|G_t\right\|^2\right]$$

$$\leq \tilde{\psi}_t - 2\alpha\ \mathbb{E}\left[\left\langle G_t\ ,\ \mathbf{w}_t - \mathbf{w}^*\ \right\rangle\right] + \alpha^2\rho^2$$

since $\left\|G_t\right\| \leq \rho$ with prob. 1

By rearranging the terms, the proof is complete.

Generic SGD algorithm: Proof of convergence guarantees

Claim 2*: $\forall t,\ \mathbb{E}\left[\langle\, \mathrm{G}_t,\ \mathbf{w}_t - \mathbf{w}^* \,\rangle\right] \geq \mathbb{E}\left[\, f(\mathbf{w}_t)\right] - f(\mathbf{w}^*)$

Proof :

By convexity of f, we have:

$$\langle\, \nabla f(\mathbf{w}_t),\ \mathbf{w}_t - \mathbf{w}^* \,\rangle \geq f(\mathbf{w}_t) - f(\mathbf{w}^*)$$

Taking expectation of both sides:

$$\mathbb{E}\left[\langle \nabla f(\mathbf{w}_t),\ \mathbf{w}_t - \mathbf{w}^* \,\rangle\right] \geq \mathbb{E}\left[\, f(\mathbf{w}_t)\right] - f(\mathbf{w}^*)$$

Using the **fact** we just proved the LHS is equal to $\mathbb{E}\left[\langle\, \mathrm{G}_t,\ \mathbf{w}_t - \mathbf{w}^* \,\rangle\right]$ which completes the proof.

Generic SGD algorithm: Proof of convergence guarantees

Combining those claims, we have

$$\mathbb{E}\left[f(\hat{\mathbf{w}})\right] - f(\mathbf{w}^*) \le \frac{1}{T}\sum_{t=1}^{T}\left(\mathbb{E}\left[f(\mathbf{w}_t)\right] - f(\mathbf{w}^*)\right)$$

$$= \frac{1}{2\alpha T}\sum_{t=1}^{T}\left(\tilde{\psi}_t - \tilde{\psi}_{t+1}\right) + \frac{\alpha\rho^2}{2}$$

$$\vdots$$

By convexity of f, the definition of $\hat{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$, and linearity of expectation.

Continue exactly as the proof of the basic GD theorem.

CSE 5523: Machine Learning Theory Fall 2017

15 - Regularization and Stability - Part 1

Instructor: Raef Bassily

Today, we'll discuss regularized loss minimization (RLM).

15.1 Regularized Loss Minimization (RLM)

Definition 15.1 (True risk and empirical risk)**.** Let D denote the distribution over $\mathcal{Z}$. For a given loss function $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$, the true risk of $w \in C \subset \mathbb{R}^d$ is

$$L(w; D) = E_{z\sim D}[\ell(w, z)].$$

Also define the empirical risk over a dataset:

$$\hat{L}(w; S) = \frac{1}{n}\sum_{i=1}^{n} \ell(w, z_i),$$

where $S = (z_1, \ldots, z_n) \sim D^n$ is the training set.

In RLM, instead of minimizing $\hat{L}(w; S)$ over $w \in C$, we will minimize the sum

$$\hat{L}(w; S) + R(w).$$

Here, $R : \mathbb{R}^d \to \mathbb{R}_+$ is a "regularizer" and only depends on the hypothesis. Intuitively, the regularizer captures some notion of the "complexity" of a hypothesis. This effectively penalizes complicated hypotheses, which helps us manage the tradeoff between empirical risk and generalization error which is essentially the same as the tradeoff between bias and complexity we discussed before.

We will focus on *Tikhonov regularization*, which uses the regularizer

$$R(w) = \Lambda \|w\|^2,$$

for some $\Lambda > 0$ (the "regularization parameter").

15.2 Stability

Recall that

$$L(w; D) = \underbrace{L(w; D) - \hat{L}(w; S)}_{\text{(signed) generalization error}} + \underbrace{\hat{L}(w; S)}_{\text{empirical risk}}.$$

A hypothesis that has high generalization error despite low empirical risk is said to "overfit" the training data.

Stability, roughly, means that small changes in the input (i.e., in the training data) lead to small changes in the output (i.e., the trained hypothesis).

We will view regularization as a "stabilizer" for the learning algorithm, and we will see that stable algorithms typically do not overfit.

Definition 15.2 (Notation)**.** Given the training set $S = (z_1, \ldots, z_n)$ and an extra example z', let

$$S^{(i)} = (z_1, \ldots, z_{i-1}, z', z_{i+1}, \ldots, z_n).$$

This is a dataset in which the i^{th} data point has been replaced by z'.

Also, let A be a learning algorithm. We will denote the output hypothesis of A given S by $A(S)$.

Consider the difference

$$\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i).$$

Intuitively, this difference is ≥ 0 for a good learner. However, if this difference is very large, this indicates that the learner is not very stable and does not generalize well to new examples.

Definition 15.3 (On-Average Replace-One (OARO) Stability)**.** Let $\tau : \mathbb{N} \to \mathbb{R}_+$ be a decreasing function. A learner A is said to be τ-OARO stable if for all distributions D, we have

$$\underset{\substack{(S,z')\sim D^{n+1} \\ i\sim\text{Unif}([n])}}{E} [\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)] \leq \tau(n).$$

Note: if we rewrite the expectation over i explicitly, this becomes

$$\underset{(S,z')\sim D^{n+1}}{E} \left[\frac{1}{n} \sum_{i=1}^{n} (\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)) \right] \leq \tau(n).$$

Theorem 15.4. Let A be a learner. If A is τ-OARO stable, then for all distributions D,

$$\underset{S\sim D^n}{E} [L(A(S); D) - \hat{L}(A(S); S)] \leq \tau(n).$$

Proof. Let D be a distribution, let $S \sim D^n$, and let z' be another independent example from D. Let $S^{(i)}$ be as defined earlier. For any learner A, we have

$$\underset{S\sim D^n}{E} [L(A(S); D) - \hat{L}(A(S); S)] = \underset{\substack{(S,z')\sim D^{n+1} \\ i\sim\text{Unif}([n])}}{E} [\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)]. \tag{1}$$

To see this, observe that for all $i \in [n]$,

$$\begin{aligned} E_{S\sim D^n}[L(A(S); D)] &= E_{(S,z')\sim D^{n+1}}[\ell(A(S), z')] \\ &= E_{(S,z')\sim D^{n+1}}[\ell(A(S^{(i)}), z_i)], \end{aligned}$$

because S and z' are i.i.d. On the other hand,

$$E_{S\sim D^n}[\hat{L}(A(S); S)] = \underset{\substack{S\sim D^n \\ i\sim\text{Unif}([n])}}{E} [\ell(A(S), z_i)].$$

Combining these facts yields (1). Now, the proof directly follows from the definition of OARO stability (Definition 15.3) and equation (1). □

CSE 5523: Machine Learning Theory Fall 2017

16 - Regularization and Stability - Part 2

Instructor: Raef Bassily

16.1 Stability of RLM and Bound on the Generalization Error

The stability of RLM relies on a property called "strong convexity."

Definition 16.1 (Strongly convex function)**.** Let $\Lambda > 0$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is Λ-strongly convex if for all $u, v \in \mathbb{R}^d$ and for all $\alpha \in (0, 1)$ we have

$$f(u) \geq f(v) + \langle \nabla f(v), u - v \rangle + \frac{\Lambda}{2} \|u - v\|^2 .$$

(For non-differentiable functions, we simply use a subgradient instead of the gradient.)

Lemma 16.2. Strongly convex functions have the following properties:

1. $f(w) = \Lambda \|w\|^2$ is 2Λ-strongly convex.
2. If f is Λ-strongly convex and g is convex, then $f + g$ is Λ-strongly convex.
3. If f is Λ-strongly convex and $\hat{w}$ is a (constrained) minimizer of f over a convex set $C \subseteq \mathbb{R}^d$, then for all $w \in C$,
$$f(w) - f(\hat{w}) \geq \frac{\Lambda}{2} \|w - \hat{w}\|^2 .$$

Theorem 16.3 (Stability of RLM and Bounded Generalization)**.** Let $C \subseteq \mathbb{R}^d$ be a convex set. Let $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$ be a loss function which is convex and ρ-Lipschitz (with respect to its first argument, for all possible values of its second argument). Then the RLM learner with regularizer $\Lambda \|w\|^2$ (denoted by algorithm A) is $2\rho^2/(\Lambda n)$-OARO stable.

Moreover, using Theorem 15.4 (from previous lecture), we have, for all distributions D,

$$\mathop{E}_{S \sim D^n} \left[L(A(S); D) - \hat{L}(A(S); S) \right] \leq \frac{2\rho^2}{\Lambda n}.$$

Proof. Let $S = (z_1, \ldots, z_n)$, let z' be an extra example, and let

$$S^{(i)} = (z_1, \ldots, z_{i-1}, z', z_{i+1}, \ldots, z_n).$$

We know that

$$A(S) = \mathop{\mathrm{argmin}}_{w \in C} (\hat{L}(w; S) + \Lambda \|w\|^2).$$

Define

$$f_S(w) = \hat{L}(w; S) + \Lambda \|w\|^2 .$$

Note that from properties 1 and 2 in Lemma 16.2 f_S is 2Λ-strongly convex. Moreover, from property 3 of the same lemma, for all $v \in C$, we have

$$f_S(v) - f_S(A(S)) \geq \Lambda \left\| v - A(S) \right\|^2 . \tag{1}$$

On the other hand, for all $u, v \in C$, and all indices $i \in [n]$,

$$\begin{aligned} f_S(v) - f_S(u) &= (\hat{L}(v; S) + \Lambda \left\| v \right\|^2) - (\hat{L}(u; S) + \Lambda \left\| u \right\|^2) \\ &= (\hat{L}(v; S^{(i)}) + \Lambda \left\| v \right\|^2) - (\hat{L}(u; S^{(i)}) + \Lambda \left\| u \right\|^2) \\ &\quad + \frac{\ell(v, z_i) - \ell(v, z')}{n} + \frac{\ell(u, z') - \ell(u, z_i)}{n} \\ &= f_{S^{(i)}}(v) - f_{S^{(i)}}(u) + \frac{\ell(v, z_i) - \ell(v, z')}{n} + \frac{\ell(u, z') - \ell(u, z_i)}{n} \end{aligned}$$

Choose $v = A(S^{(i)})$ and $u = A(S)$. Then,

$$\begin{aligned} f_S(A(S^{(i)})) - f_S(A(S)) =& f_{S^{(i)}}(A(S^{(i)})) - f_{S^{(i)}}(A(S)) \\ &+ \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S^{(i)}), z')}{n} + \frac{\ell(A(S), z') - \ell(A(S), z_i)}{n} \\ \leq& \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}{n} + \frac{\ell(A(S), z') - \ell(A(S^{(i)}), z')}{n}. \end{aligned} \tag{2}$$

where the last inequality follows from the fact that $A(S^{(i)})$ is the minimizer of $f_{S^{(i)}}$ (that follows from the fact that A is RLM algorithm). Now, combining (1) and (2),

$$\Lambda \left\| A(S^{(i)}) - A(S) \right\|^2 \leq \frac{\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i)}{n} + \frac{\ell(A(S), z') - \ell(A(S^{(i)}), z')}{n}. \tag{3}$$

Since ℓ is ρ-Lipschitz,

$$\ell(A(S^{(i)}, z_i) - \ell(A(S), z_i) \leq \rho \left\| A(S^{(i)}) - A(S) \right\| . \tag{4}$$

A similar bound holds for the second term involving z'. Hence,

$$\Lambda \left\| A(S^{(i)}) - A(S) \right\|^2 \leq \frac{2\rho \left\| A(S^{(i)}) - A(S) \right\|}{n},$$

and so

$$\left\| A(S^{(i)}) - A(S) \right\| \leq \frac{2\rho}{\Lambda n}.$$

Substituting this into (4), we have

$$\ell(A(S^{(i)}), z_i) - \ell(A(S), z_i) \leq \frac{2\rho^2}{\Lambda n}.$$

Note that we have not made any restrictions on S, and so we can conclude this bound holds in expectation, thus completing the proof.

□

16.2 Bounded Empirical Error of RLM

Note that

$$\underset{S\sim D^n}{E}\left[L(A(S);D)\right] = \underset{S\sim D^n}{E}\left[\widehat{L}(A(S);S)\right] + \underbrace{\underset{S\sim D^n}{E}\left[L(A(S);D) - \widehat{L}(A(S);S)\right]}_{\leq 2\rho^2/(\Lambda n)}. \qquad (*)$$

Recall also that

$$A(S) = \underset{w\in C}{\operatorname{argmin}}\left(\widehat{L}(A(S);S) + \Lambda\left\|w\right\|^2\right).$$

As Λ increases, the expected generalization error decreases, but the empirical risk starts to increase since the minimizer of the regularized risk (the sum of the empirical risk and regularization function) starts moving away from the minimizer of the empirical risk. When $\Lambda = 0$ (i.e., no regularization), the empirical risk is minimized, but we no longer have control on the generalization error (risk of overfitting). On the other hand, in the limit as $\Lambda \to \infty$, the algorithm will no longer depend on the data since it will effectively try to minimize the $\Lambda\left\|w\right\|^2$ term only (hence, when Λ is too large, we are at risk of underfitting). We need to tune Λ to balance the tradeoff between empricial risk and generalization error (or, equivalently, the tradeoff between overfitting and underfitting, or, equivalently, bias and complexity).

Theorem 16.4. As before, suppose $C \subset \mathbb{R}^d$ is a closed convex set and $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$ is convex ρ-Lipschitz. Let D be any distribution on $\mathcal{Z}$. Then, the RLM learner (denoted by A) with regularization parameter Λ satisfies the following for all $w \in C$:

$$\underset{S\sim D^n}{E}\left[\widehat{L}(A(S);S)\right] \leq L(w;D) + \Lambda\left\|w\right\|^2.$$

Note that this theorem can give us a bound on the excess risk of $A(S)$. It also captures the tradeoff involved in changing Λ, since one term increases with Λ and the other decreases with Λ.

Proof. Fix any $w \in C$. Then

$$\begin{aligned}\widehat{L}(A(S);S) &\leq \widehat{L}(A(S);S) + \Lambda\left\|A(S)\right\|^2 \\ &\leq \widehat{L}(w;S) + \Lambda\left\|w\right\|^2.\end{aligned}$$

Now take the expectation of both sides:

$$\begin{aligned}\underset{S\sim D^n}{E}\left[\widehat{L}(A(S);S)\right] &\leq \underset{S\sim D^n}{E}\left[\widehat{L}(w;S)\right] + \Lambda\left\|w\right\|^2 \\ &= L(w;D) + \Lambda\left\|w\right\|^2.\end{aligned}$$

□

Plugging the result of this theorem into (*) above and using Theorem 16.3, then $\forall w \in C$, we have:

$$\underset{S\sim D^n}{E}\left[\widehat{L}(A(S);S)\right] \leq L(w;D) + \Lambda\left\|w\right\|^2 + \frac{2\rho^2}{\Lambda n} \qquad (5)$$

Note that the above bound leads directly to a bound on the excess risk of $A(S)$. It also captures the tradeoff involved in changing Λ, since one term increases with Λ and the other decreases with Λ. By setting Λ such that the bound on the right-hand side is minimized, we obtain the main result of this part, which is stated formally in the following corollary:

Corollary 16.5. Consider a convex-Lipschitz-bounded model where the parameter set $C \subset \mathbb{R}^d$ is M-bounded and the loss function $\ell : C \times \mathcal{Z} \to \mathbb{R}_+$ is convex and ρ-Lipschitz. Let D be any distribution. If we set $\Lambda = \frac{\rho}{M}\sqrt{\frac{2}{n}}$, then the RLM A satisfies

$$\underset{S\sim D^n}{E}[L(A(S);D)] \leq \min_{w\in C} L(w;D) + \rho M\sqrt{\frac{8}{n}}.$$

In particular, for any $\varepsilon > 0$, if $n \geq 8M^2\rho^2/\varepsilon^2$, then

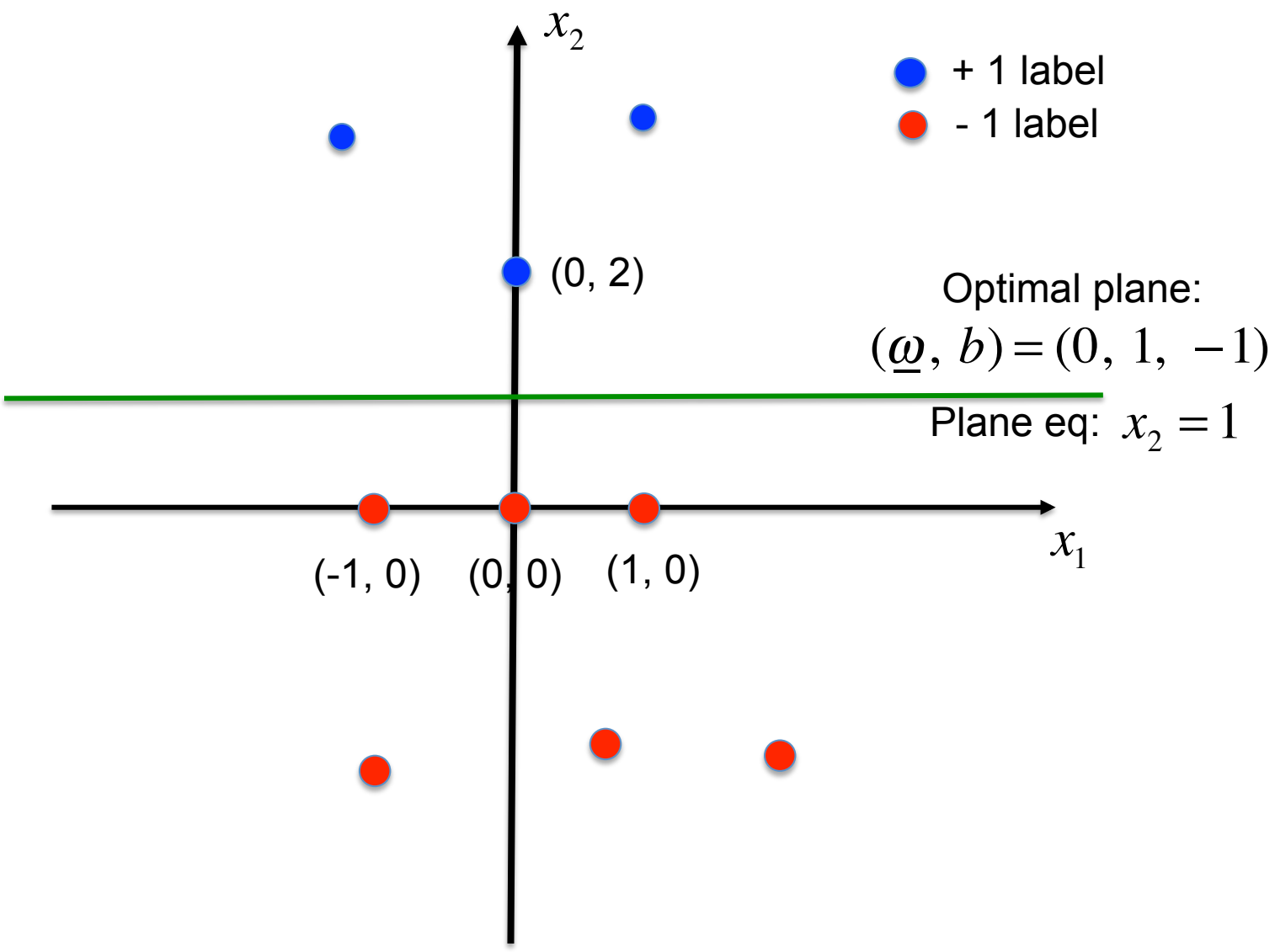
$$\underset{S\sim D^n}{E}[L(A(S);D)] \leq \min_{w\in C} L(w;D) + \varepsilon.$$

Proof. Choose w in Theorem 16.4 to be $w^* = \operatorname{argmin}_{w\in C} L(w;D)$. Then, pick Λ to give the best bound in (5):

$$\Lambda = \frac{\rho}{M}\sqrt{\frac{2}{n}}.$$

We hence get the desired result. □

Remark 16.6. We can get a more "agnostic-PAC-like" bound by turning the expectation bound into a probability bound (using Markov's inequality). If the loss function is bounded, one can further amplify the confidence using repeated training/validation sets (similar to an earlier homework problem).


x_2
+ 1 label
- 1 label
(0, 2)
Optimal plane:
$(\underline{\omega}, b) = (0, 1, -1)$
Plane eq: $x_2 = 1$
x_1
(-1, 0)
(0, 0)
(1, 0)

Algorithms for Linear Classifiers: Perceptron & SVMs

Linear classifiers in the realizable case

Linearly separable case (i.e., **realizability** holds):

Input: training data $\left((\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\right) \in \mathbb{R}^{d-1} \times \{-1, +1\}$

Output: linear classifier parameter vector $\mathbf{w}^* \in \mathbb{R}^d$
such that

$$y^{(i)} \langle\, \mathbf{w}^*,\, \tilde{\mathbf{x}}^{(i)} \,\rangle \;>\; 0 \quad \text{for} \quad i = 1,\ \ldots,\ n$$

where $\tilde{\mathbf{x}}^{(i)} = \left(\mathbf{x}^{(i)},\, 1\right)$

Linear classifiers in the realizable case

Linearly separable case (i.e., **realizability** holds):

Input: training data $\left((\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\right) \in \mathbb{R}^{d-1} \times \{-1, +1\}$

Output: linear classifier parameter vector $\mathbf{w}^* \in \mathbb{R}^d$
such that

$$y^{(i)} \langle\, \mathbf{w}^*,\, \tilde{\mathbf{x}}^{(i)} \,\rangle \;>\; 0 \quad \text{for} \quad i = 1,\ \ldots,\ n$$

where $\tilde{\mathbf{x}}^{(i)} = \left(\mathbf{x}^{(i)},\, 1\right)$

Note that such $\mathbf{w}^*$ must exist since we assume realizability.

Linear classifiers in the realizable case

Linearly separable case (i.e., **realizability** holds):

Input: training data $\left((\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\right) \in \mathbb{R}^{d-1} \times \{-1, +1\}$

Output: linear classifier parameter vector $\mathbf{w}^* \in \mathbb{R}^d$
such that

$$y^{(i)} \langle\, \mathbf{w}^*,\, \tilde{\mathbf{x}}^{(i)} \,\rangle \;>\; 0 \quad \text{for} \quad i = 1,\, \ldots,\, n$$

where $\tilde{\mathbf{x}}^{(i)} = \left(\mathbf{x}^{(i)},\, 1\right)$

Note that such $\mathbf{w}^*$ must exist since we assume realizability.

Define $\tilde{\gamma} \triangleq \min_i \; y^{(i)} \; \langle \mathbf{w}^*, \tilde{\mathbf{x}}^{(i)} \rangle$

Linear classifiers in the realizable case

Linearly separable case (i.e., **realizability** holds):

Input: training data $\left((\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\right) \in \mathbb{R}^{d-1} \times \{-1, +1\}$

Output: linear classifier parameter vector $\mathbf{w}^* \in \mathbb{R}^d$
such that

$$y^{(i)} \langle\, \mathbf{w}^*,\, \tilde{\mathbf{x}}^{(i)} \,\rangle \;>\; 0 \quad \text{for} \quad i = 1,\, \ldots,\, n$$

where $\tilde{\mathbf{x}}^{(i)} = \left(\mathbf{x}^{(i)},\, 1\right)$

Note that such $\mathbf{w}^*$ must exist since we assume realizability.

Define $\tilde{\gamma} \triangleq \min_i \; y^{(i)} \; \langle \mathbf{w}^*, \tilde{\mathbf{x}}^{(i)} \rangle$ ⟵ $\tilde{\gamma}$ is an indicator of the "margin" of $\mathbf{w}^*$

Linear classifiers in the realizable case

Linearly separable case (i.e., **realizability** holds):

Input: training data $\left((\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\right) \in \mathbb{R}^{d-1} \times \{-1, +1\}$

Output: linear classifier parameter vector $\mathbf{w}^* \in \mathbb{R}^d$
such that

$$y^{(i)} \langle\, \mathbf{w}^*,\, \tilde{\mathbf{x}}^{(i)} \,\rangle \;>\; 0 \quad \text{for} \quad i = 1,\, \ldots,\, n$$

where $\tilde{\mathbf{x}}^{(i)} = \left(\mathbf{x}^{(i)},\, 1\right)$

Note that such $\mathbf{w}^*$ must exist since we assume realizability.

Define $\tilde{\gamma} \triangleq \min_i \; y^{(i)} \; \langle \mathbf{w}^*, \tilde{\mathbf{x}}^{(i)} \rangle$ ⟵ $\tilde{\gamma}$ is an indicator of the "margin" of $\mathbf{w}^*$

Let $\widehat{\mathbf{w}} = \frac{1}{\tilde{\gamma}} \mathbf{w}^* \quad \Longrightarrow \quad y^{(i)} \langle \widehat{\mathbf{w}} \;,\; \tilde{\mathbf{x}}^{(i)} \rangle \geq 1 \quad \forall \; i \in [n]$

Linear classifiers in the realizable case

This shows there exists a vector $\mathbf{w} \in \mathbb{R}^d$ that satisfies all the *n constraints*

$$y^{(i)} \langle\, \mathbf{w},\, \tilde{\mathbf{x}}^{(i)} \,\rangle \;\geq\; 1 \quad \text{for} \quad i = 1,\, \ldots,\, n$$

Such a vector $\mathbf{w}$ is clearly an ERM for the standard binary (0-1) loss function.

Finding such vector can be formulated as a linear program, which can be solved using generic LP solvers such as the simplex method.

However, we *won't be using such LP solvers*. There is a rather simple and more direct algorithm well-suited for such problem, known as ***Perceptron Algorithm*.**

Perceptron Algorithm

Input: $\left\{(\tilde{\mathbf{x}}^{(1)}, y^{(1)}), \ldots, (\tilde{\mathbf{x}}^{(n)}, y^{(n)})\right\}$

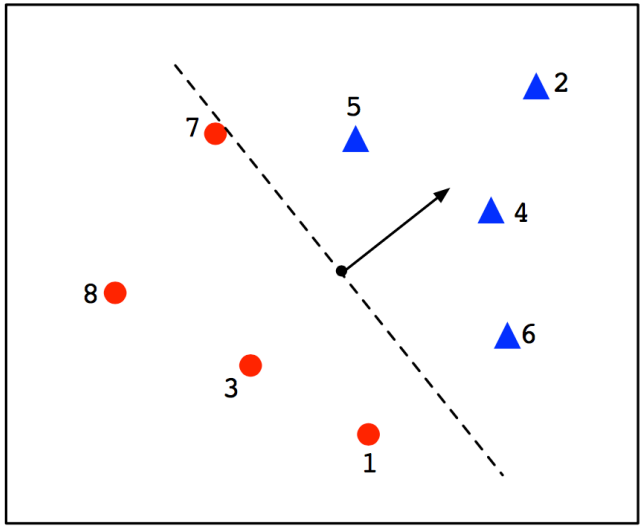
Initialize $\mathbf{w}_1 = \mathbf{0}$

For $t = 1, 2, \ldots$

If $\exists \; i \;$ s.t. $\; y^{(i)} \langle \mathbf{w}_t, \tilde{\mathbf{x}}^{(i)} \rangle \leq 0$ **then**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y^{(i)} \; \tilde{\mathbf{x}}^{(i)}$$

Else **return** $\mathbf{w}_t$



Perceptron Algorithm

Input: $\left\{(\tilde{\mathbf{x}}^{(1)}, y^{(1)}), \ldots, (\tilde{\mathbf{x}}^{(n)}, y^{(n)})\right\}$

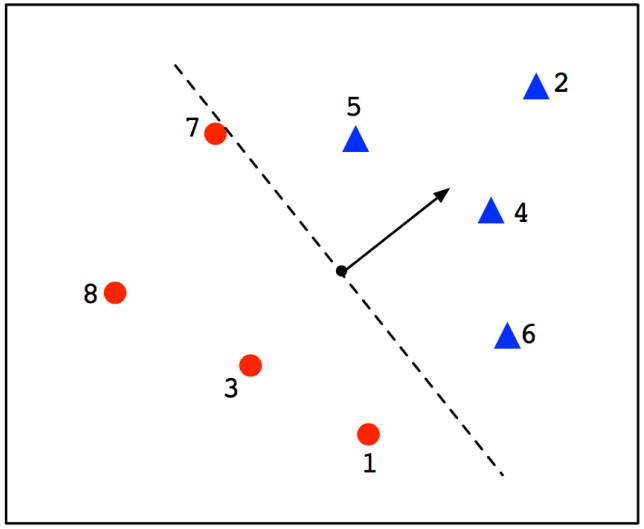
Initialize $\mathbf{w}_1 = \mathbf{0}$

For *t = 1, 2, …*

If $\exists\ i$ s.t. $y^{(i)} \langle \mathbf{w}_t, \tilde{\mathbf{x}}^{(i)} \rangle \leq 0$ **then**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y^{(i)}\ \tilde{\mathbf{x}}^{(i)}$$

Else **return** $\mathbf{w}_t$



Note that

$$y^{(i)} \langle \mathbf{w}_{t+1}\ ,\ \tilde{\mathbf{x}}^{(i)} \rangle = y^{(i)} \langle \mathbf{w}_t + y^{(i)}\ \tilde{\mathbf{x}}^{(i)}\ ,\ \tilde{\mathbf{x}}^{(i)} \rangle$$

$$= y^{(i)} \langle \mathbf{w}_t\ ,\ \tilde{\mathbf{x}}^{(i)} \rangle + \| \tilde{\mathbf{x}}^{(i)} \|^2$$

Perceptron Algorithm

Input: $\left\{(\tilde{\mathbf{x}}^{(1)}, y^{(1)}), \ldots, (\tilde{\mathbf{x}}^{(n)}, y^{(n)})\right\}$

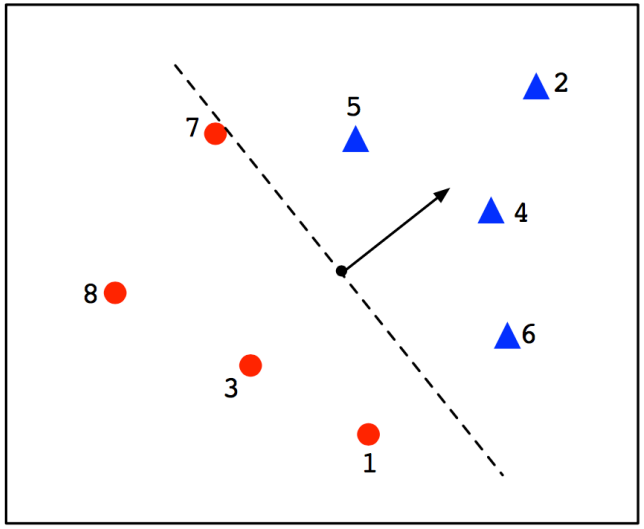
Initialize $\mathbf{w}_1 = \mathbf{0}$

For *t = 1, 2, …*

If $\exists\ i$ s.t. $y^{(i)} \langle \mathbf{w}_t, \tilde{\mathbf{x}}^{(i)} \rangle \leq 0$ **then**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y^{(i)}\ \tilde{\mathbf{x}}^{(i)}$$

Else **return** $\mathbf{w}_t$



Note that

$$y^{(i)} \langle \mathbf{w}_{t+1}\ ,\ \tilde{\mathbf{x}}^{(i)} \rangle = y^{(i)} \langle \mathbf{w}_t + y^{(i)}\ \tilde{\mathbf{x}}^{(i)}\ ,\ \tilde{\mathbf{x}}^{(i)} \rangle$$

$$= y^{(i)} \langle \mathbf{w}_t\ ,\ \tilde{\mathbf{x}}^{(i)} \rangle + \underbrace{\| \tilde{\mathbf{x}}^{(i)} \|^2}_{> 0}$$

Recall our goal is to have $y^{(i)} \langle\ \mathbf{w}\ ,\ \tilde{\mathbf{x}}^{(i)}\ \rangle\ >\ 0\ \ \forall i$

The Perceptron update pushes the solution to be “more correct” on the *i-th* point.

Perceptron Algorithm

Input: $\left\{(\tilde{\mathbf{x}}^{(1)}, y^{(1)}), \ldots, (\tilde{\mathbf{x}}^{(n)}, y^{(n)})\right\}$

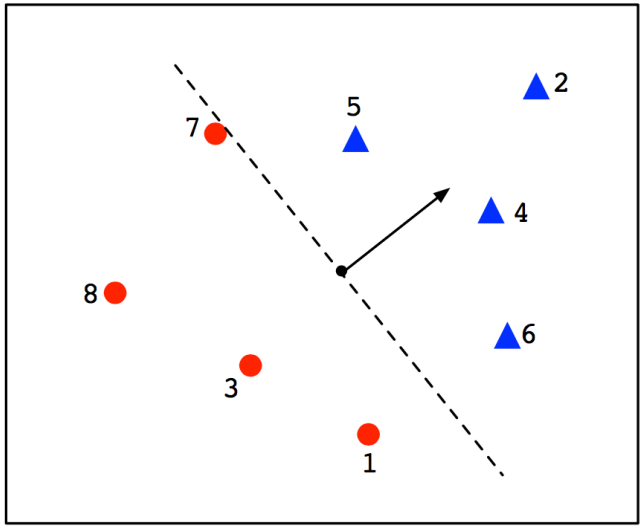
Initialize $\mathbf{w}_1 = \mathbf{0}$

For $t = 1, 2, \ldots$

If $\exists\ i$ s.t. $y^{(i)} \langle \mathbf{w}_t, \tilde{\mathbf{x}}^{(i)} \rangle \leq 0$ **then**

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y^{(i)}\ \tilde{\mathbf{x}}^{(i)}$$

Else **return** $\mathbf{w}_t$



Theorem: [Convergence of Perceptron Algorithm]

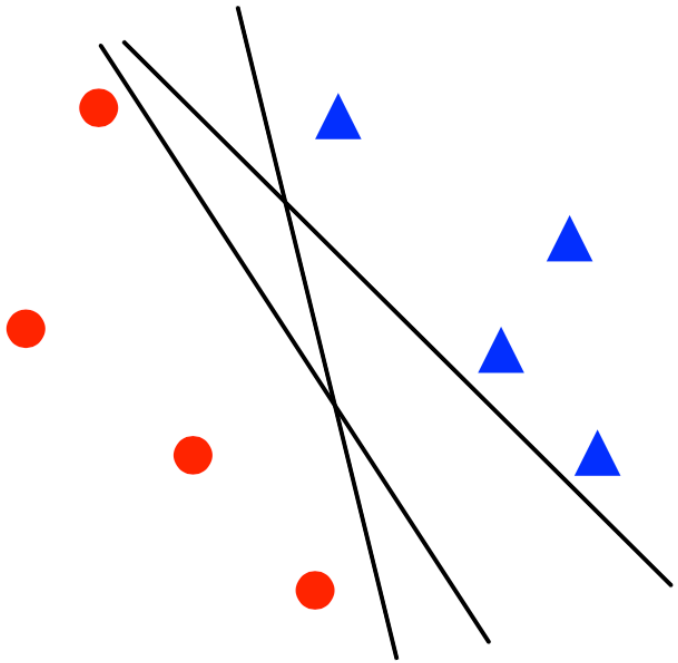
Suppose the training set is linearly separable.

Let $M = \min\left\{ \|\mathbf{w}\| : y^{(i)} \langle \mathbf{w}, \tilde{\mathbf{x}}^{(i)} \rangle \geq 1 \ \ \forall i \in [n] \right\}$, let $B = \max_i \|\tilde{\mathbf{x}}^{(i)}\|$.

Then, the perceptron algorithm stops after at most $M^2 B^2$ iterations, and when it stops we have $y^{(i)} \langle \mathbf{w}_t, \tilde{\mathbf{x}}^{(i)} \rangle > 0 \ \ \forall i \in [n]$

A better separator?

For a linearly separable data set, there are in general many possible separating hyperplanes, and Perceptron is guaranteed to find one of them.



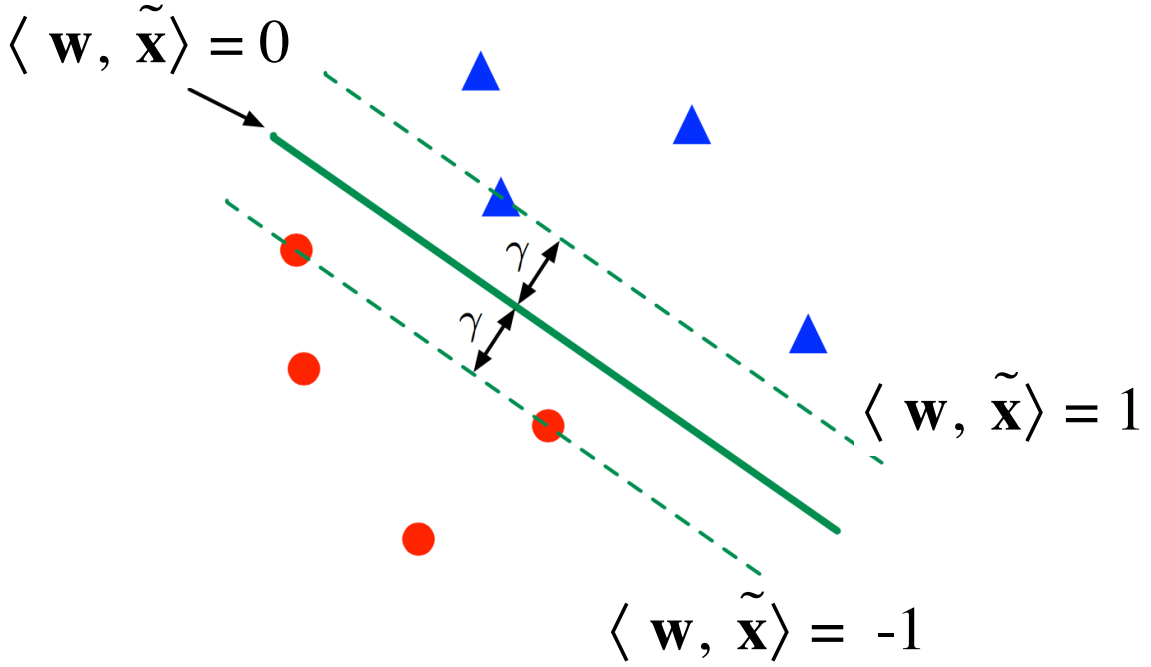
But is there a better, more systematic choice of separator? The one with the most buffer around it, for instance?

Maximizing the margin

Want $y^{(i)} \langle \mathbf{w}, \tilde{\mathbf{x}}^{(i)} \rangle > 0 \quad \text{for} \quad i = 1, \ldots, n$

Like before, by normalizing $\mathbf{w}$ by its "margin" γ, we can equally ask for

$$y^{(i)} \langle \mathbf{w}, \tilde{\mathbf{x}}^{(i)} \rangle \geq 1 \quad \forall \quad i \in [n]$$

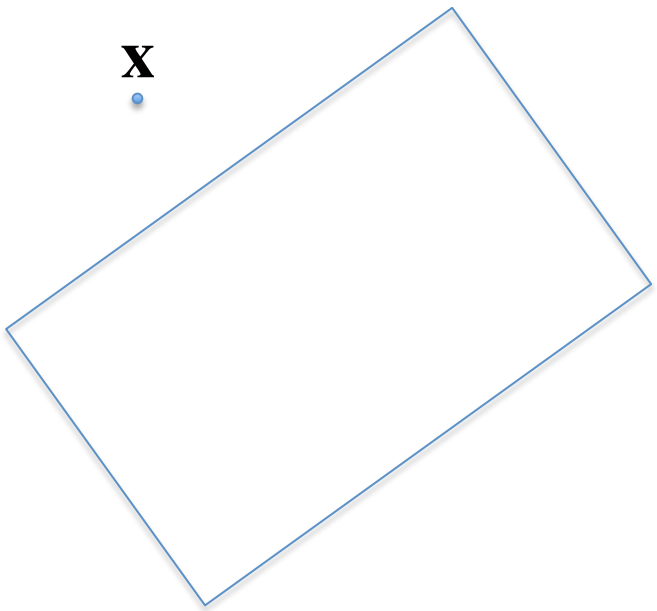


Maximize the **margin** γ.

Maximizing the margin

Let $\mathbf{w} = (\underline{\omega}, b)$ where $\underline{\omega} = (w_1, \ldots, w_{d-1})$
and b is the constant term of the linear classifier.

Distance from any point $\mathbf{x}$ and the plane:

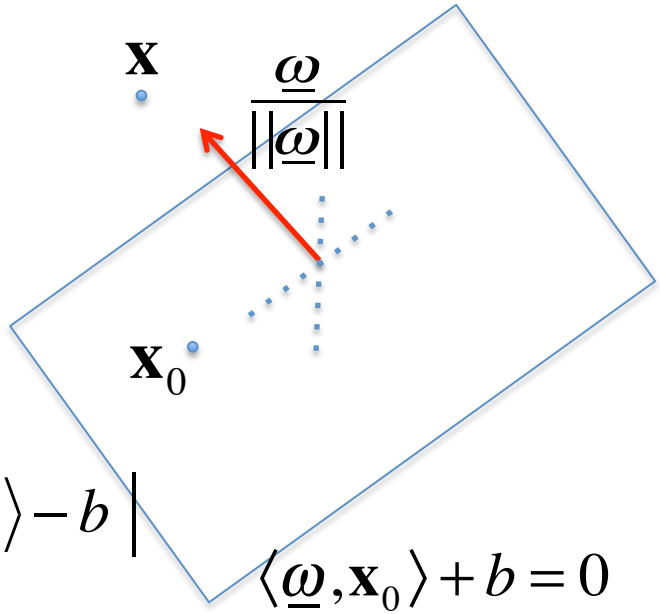


Maximizing the margin

Let $\mathbf{w} = (\underline{\omega}, b)$ where $\underline{\omega} = (w_1, \ldots, w_{d-1})$ and b is the constant term of the linear classifier.

Distance from any point $\mathbf{x}$ and the plane:

Let $\mathbf{x}_0$ be a point **on the plane.** Then, the distance from $\mathbf{x}$ to the plane is



$$\left| \left\langle \frac{\underline{\omega}}{\|\underline{\omega}\|} , \mathbf{x} - \mathbf{x}_0 \right\rangle \right| = \frac{1}{\|\underline{\omega}\|} \left| \langle \underline{\omega} , \mathbf{x} \rangle + b - \langle \underline{\omega} , \mathbf{x}_0 \rangle - b \right|$$

$$= \frac{1}{\|\underline{\omega}\|} \left| \langle \underline{\omega} , \mathbf{x} + b \rangle \right|$$

Maximizing the margin

Let $\mathbf{w} = (\underline{\omega}, b)$ where $\underline{\omega} = (w_1, \ldots, w_{d-1})$
and b is the constant term of the linear classifier.

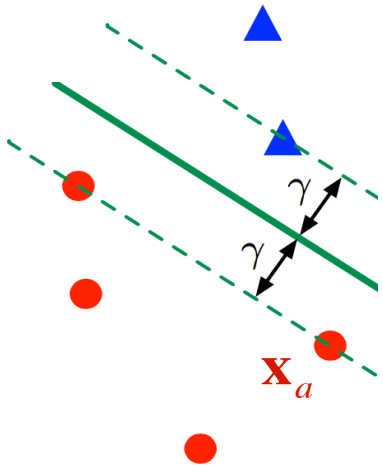
Distance from any point $\mathbf{x}$ and the plane:

Let $\mathbf{x}_0$ be a point **on the plane.** Then, the distance from $\mathbf{x}$ to the plane is $\frac{1}{\|\underline{\omega}\|} \left| \langle \underline{\omega} , \mathbf{x} \rangle + b \right|$

Let $\mathbf{x}_a$ be one of the closest points (feature vectors) to the separating plane.

Due to aforementioned normalization, we have

$$\left| \langle \underline{\omega}, \mathbf{x}_a \rangle + b \right| = 1$$



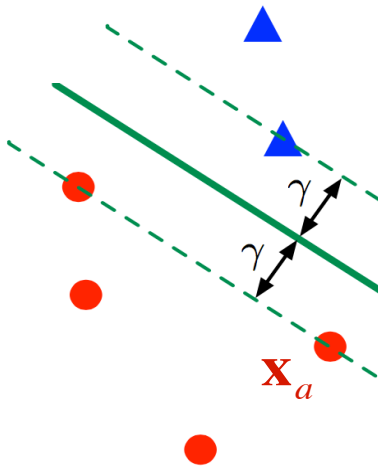
Maximizing the margin

Let $\mathbf{w} = (\underline{\omega}, b)$ where $\underline{\omega} = (w_1, \ldots, w_{d-1})$
and b is the constant term of the linear classifier.

Distance from any point $\mathbf{x}$ and the plane:

Let $\mathbf{x}_0$ be a point **on the plane.** Then, the distance from $\mathbf{x}$ to the plane is $\frac{1}{\|\underline{\omega}\|} \left| \langle \underline{\omega} , \mathbf{x} \rangle + b \right|$

Let $\mathbf{x}_a$ be one of the closest points (feature vectors) to the separating plane.



Due to aforementioned normalization, we have

$$\left| \langle \underline{\omega}, \mathbf{x}_a \rangle + b \right| = 1$$

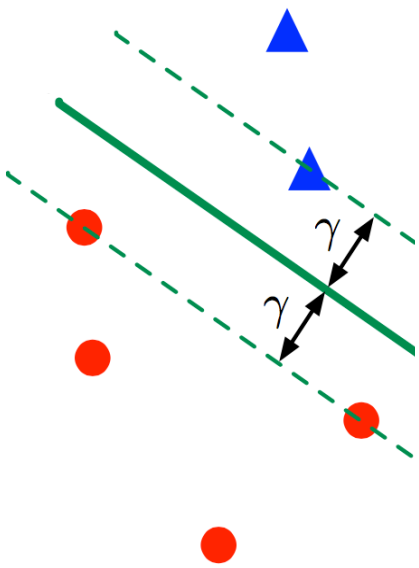
Margin γ = distance from $\mathbf{x}_a$ to the plane. Hence,

$$\gamma = \frac{1}{\|\underline{\omega}\|} \left| \langle \underline{\omega} , \mathbf{x}_a \rangle + b \right| = \frac{1}{\|\underline{\omega}\|}$$

Maximizing the margin

Hence maximizing the margin γ is equivalent to **minimizing** $\| \underline{\omega} \|$ *subject to*

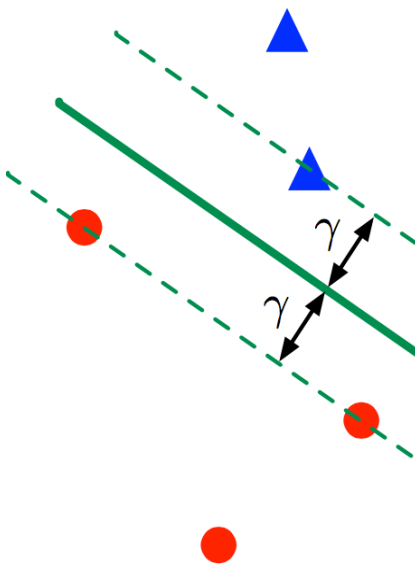
$$y^{(i)}\left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b\right) \geq 1 \quad \forall \quad i \in [n]$$



Maximizing the margin

Hence maximizing the margin γ is equivalent to **minimizing** $\|\underline{\omega}\|$ *subject to*

$$y^{(i)}\left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b\right) \geq 1 \quad \forall \quad i \in [n]$$



This is a convex constrained optimization problem that has the following equivalent **dual** form:

$$\text{(DUAL)} \quad \max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} (x^{(i)} \cdot x^{(j)})$$

$$\text{s.t.:} \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

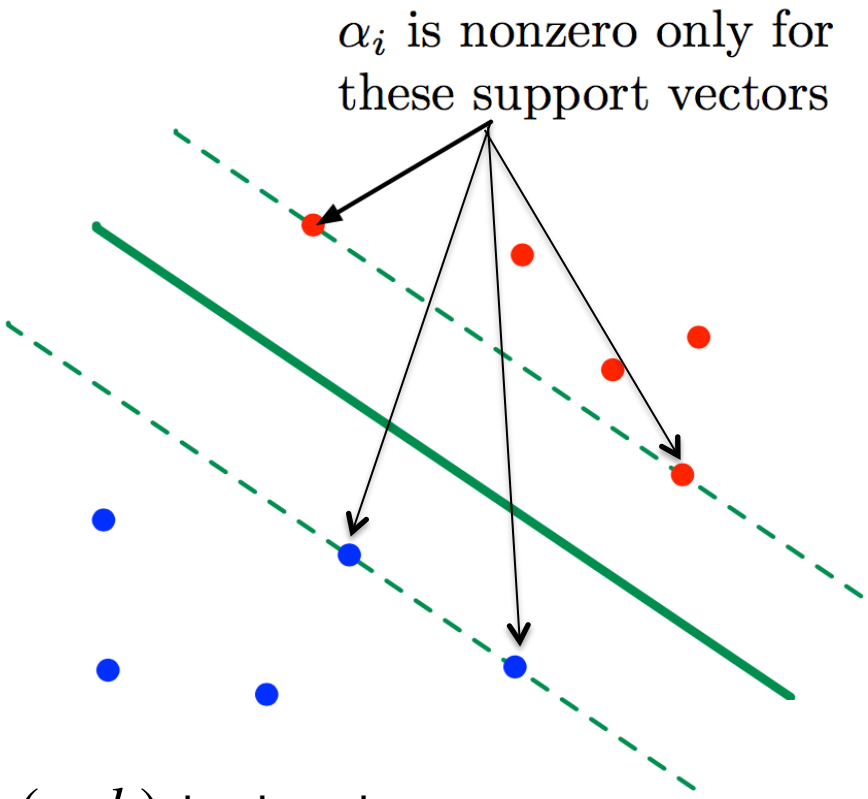
$$\alpha \geq 0$$

At optimality, $w = \sum_{i=1}^{n} \alpha_i y^{(i)} x^{(i)}$ and moreover

$$\alpha_i > 0 \quad \Rightarrow \quad y^{(i)}\left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b\right) = 1$$

Points $x^{(i)}$ with $\alpha_i > 0$ are called **support vectors**.

Support vectors



Linear classifier $\mathbf{w} = (\underline{\omega}, b)$ is given by:

$$\underline{\omega} = \sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)} = \sum_{i \in \text{Supp-Vec}} \alpha_i y^{(i)} \mathbf{x}^{(i)}$$

$b = y^{(s)} - \langle \underline{\omega} \ , \ \mathbf{x}^{(s)} \rangle$ where *(s)* denotes index of one of the support vectors

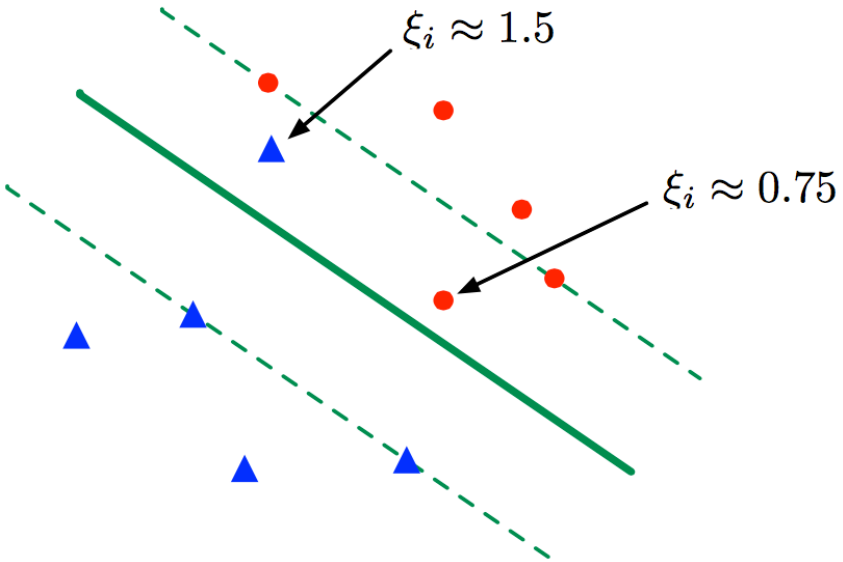
The non-separable case: Soft margin

Idea: allow each point $\mathbf{x}^{(i)}$ to have some slack ξ_i

I.e., modify optimization problem to

$$\min_{\underline{\omega},\ \{\xi_i\}_{i=1}^n} \left(\Lambda \| \underline{\omega} \|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \right)$$

subject to $y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right) \geq 1 - \xi_i \quad , \quad \xi_i \geq 0 \quad \forall \ i \in [n]$



The non-separable case: Soft margin

$$\min_{\underline{\omega},\ \{\xi_i\}_{i=1}^n} \left(\Lambda \| \underline{\omega} \|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \right)$$

subject to $y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right) \geq 1 - \xi_i \quad , \quad \xi_i \geq 0 \quad \forall \ i \in [n]$

Note that the **first** set of constraints above implies

$$\xi_i \geq 1 - y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right)$$

The non-separable case: Soft margin

$$\min_{\underline{\omega},\ \{\xi_i\}_{i=1}^n} \left(\Lambda \| \underline{\omega} \|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \right)$$

subject to $y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right) \geq 1 - \xi_i \ , \ \xi_i \geq 0 \quad \forall \ i \in [n]$

Note that the **first** set of constraints above implies

$$\xi_i \geq 1 - y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right)$$

Together with the **second** set of constraints, this implies

$$\xi_i \geq \max \left(0 \ , \ 1 - y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right) \right)$$

The non-separable case: Soft margin

$$\min_{\underline{\omega},\, \{\xi_i\}_{i=1}^n} \left(\Lambda \| \underline{\omega} \|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \right)$$

subject to $y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right) \geq 1 - \xi_i \;\; , \;\; \xi_i \geq 0 \quad \forall \;\; i \in [n]$

Note that the **first** set of constraints above implies

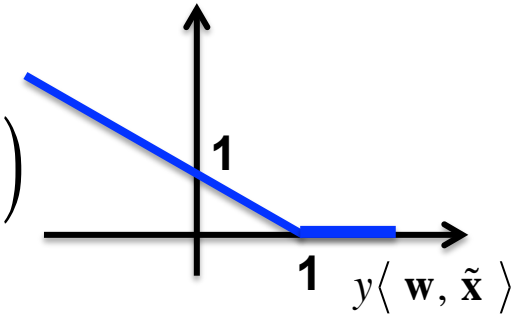
$$\xi_i \geq 1 - y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right)$$

Together with the **second** set of constraints, this implies

$$\xi_i \geq \max \left(0 \;\; , \;\; 1 - y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right) \right)$$

Hinge loss

$$\ell_{\text{hinge}} \left(\mathbf{w} \;;\; \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right)$$



The non-separable case: Soft margin

$$\min_{\underline{\omega},\ \{\xi_i\}_{i=1}^n} \left(\Lambda \| \underline{\omega} \|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \right)$$

subject to $y^{(i)} \left(\langle \underline{\omega},\ \mathbf{x}^{(i)} \rangle + b \right) \ \geq \ 1 - \xi_i \ , \ \xi_i \geq 0 \quad \forall \ i \in [n]$

Note that the **first** set of constraints above implies

$$\xi_i \geq 1 - y^{(i)} \left(\langle \underline{\omega},\ \mathbf{x}^{(i)} \rangle + b \right)$$

Together with the **second** set of constraints, this implies

$$\xi_i \geq \ell_{\text{hinge}} \left(\mathbf{w} \ ; \ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right)$$

Hence, in the above minimization problem, the best assignment for ξ_i is

$$\ell_{\text{hinge}} \left(\mathbf{w} \ ; \ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right)$$

The non-separable case: Soft margin

$$\min_{\underline{\omega},\, \{\xi_i\}_{i=1}^n} \left(\Lambda \| \underline{\omega} \|^2 + \frac{1}{n} \sum_{i=1}^{n} \xi_i \right)$$

subject to $y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right) \geq 1 - \xi_i \quad , \quad \xi_i \geq 0 \quad \forall \; i \in [n]$

Note that the **first** set of constraints above implies

$$\xi_i \geq 1 - y^{(i)} \left(\langle \underline{\omega}, \mathbf{x}^{(i)} \rangle + b \right)$$

Together with the **second** set of constraints, this implies

$$\xi_i \geq \ell_{\text{hinge}} \left(\mathbf{w} \; ; \; \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right)$$

Hence, in the above minimization problem, the best assignment for ξ_i is $\ell_{\text{hinge}} \left(\mathbf{w} \; ; \; \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right)$. Thus, we end up with the following **RLM**:

$$\min_{\underline{\omega},\, b} \left(\hat{L}_{\text{hinge}} \left((\underline{\omega}, b) \; ; \; S \right) + \Lambda \| \underline{\omega} \|^2 \right)$$

Concluding Remarks

We learned

- The PAC model: A formal model to assess and quantify machine learning.

We learned

- The PAC model: A formal model to assess and quantify machine learning.
- The concept of uniform convergence and a general and precise characterization of learnability via the notion of VC dimension.

We learned

- The PAC model: A formal model to assess and quantify machine learning.
- The concept of uniform convergence and a general and precise characterization of learnability via the notion of VC dimension.
- Sample complexity and its connection to VC dimension.

We learned

- The PAC model: A formal model to assess and quantify machine learning.
- The concept of uniform convergence and a general and precise characterization of learnability via the notion of VC dimension.
- Sample complexity and its connection to VC dimension.
- For classification problems, ERM algorithms can learn with optimal sample complexity, or equivalently, optimal accuracy.

We learned

- The PAC model: A formal model to assess and quantify machine learning.
- The concept of uniform convergence and a general and precise characterization of learnability via the notion of VC dimension.
- Sample complexity and its connection to VC dimension.
- For classification problems, ERM algorithms can learn with optimal sample complexity, or equivalently, optimal accuracy.
- Boosting: a paradigm to smoothly control the bias-complexity tradeoff by gradually increasing the model complexity.

We learned

- The PAC model: A formal model to assess and quantify machine learning.
- The concept of uniform convergence and a general and precise characterization of learnability via the notion of VC dimension.
- Sample complexity and its connection to VC dimension.
- For classification problems, ERM algorithms can learn with optimal sample complexity, or equivalently, optimal accuracy.
- Boosting: a paradigm to smoothly control the bias-complexity tradeoff by gradually increasing the model complexity.
- ERM is not always efficient when working the ‘0-1’ loss.

We learned

- The PAC model: A formal model to assess and quantify machine learning.
- The concept of uniform convergence and a general and precise characterization of learnability via the notion of VC dimension.
- Sample complexity and its connection to VC dimension.
- For classification problems, ERM algorithms can learn with optimal sample complexity, or equivalently, optimal accuracy.
- Boosting: a paradigm to smoothly control the bias-complexity tradeoff by gradually increasing the model complexity.
- ERM is not always efficient when working the '0-1' loss.
- When the direct ERM approach is computationally inefficient:
  - boosting can help circumventing the problem since it usually relies on multiple calls of a simple learning rule.
  - or, we may want to resort to a "surrogate" convex loss function instead of the '0-1' loss, as in logistic regression. Hence, the problem becomes efficiently solvable (convex optimization).

We learned

- The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).

We learned

- The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).
- Many regression and classification problems can be expressed and dealt with using the convex learning framework.

We learned

- The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).
- Many regression and classification problems can be expressed and dealt with using the convex learning framework.
- Can be used as a surrogate model for binary classification to circumvent efficiency problems. (E.g., linear classification under the ‘0-1’ loss for non-separable data is NP-hard, but classification via logistic regression or soft-margin SVM is efficient.)

We learned

- The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).
- Many regression and classification problems can be expressed and dealt with using the convex learning framework.
- Can be used as a surrogate model for binary classification to circumvent efficiency problems. (E.g., linear classification under the '0-1' loss for non-separable data is NP-hard, but classification via logistic regression or soft-margin SVM is efficient.)
- Under relatively weak assumptions on the model, e.g., Lipschitzness of the loss and boundedness of the hypothesis class, we can efficiently learn the model (in the agnostic PAC sense) via:

  Stochastic Gradient Descent ***or*** Regularized Loss Minimization

We learned

- The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).
- Many regression and classification problems can be expressed and dealt with using the convex learning framework.
- Can be used as a surrogate model for binary classification to circumvent efficiency problems. (E.g., linear classification under the '0-1' loss for non-separable data is NP-hard, but classification via logistic regression or soft-margin SVM is efficient.)
- Under relatively weak assumptions on the model, e.g., Lipschitzness of the loss and boundedness of the hypothesis class, we can efficiently learn the model (in the agnostic PAC sense) via:
  Stochastic Gradient Descent ***or*** Regularized Loss Minimization
- Regularization is tied to the algorithmic stability of the learner.

We learned

- The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).
- Many regression and classification problems can be expressed and dealt with using the convex learning framework.
- Can be used as a surrogate model for binary classification to circumvent efficiency problems. (E.g., linear classification under the '0-1' loss for non-separable data is NP-hard, but classification via logistic regression or soft-margin SVM is efficient.)
- Under relatively weak assumptions on the model, e.g., Lipschitzness of the loss and boundedness of the hypothesis class, we can efficiently learn the model (in the agnostic PAC sense) via:

  Stochastic Gradient Descent ***or*** Regularized Loss Minimization
- Regularization is tied to the algorithmic stability of the learner.
- Stability is directly related to the ability of the learner to generalize.

We learned

- The convex learning model is a powerful generalization to the PAC model (originally introduced for classification problems).
- Many regression and classification problems can be expressed and dealt with using the convex learning framework.
- Can be used as a surrogate model for binary classification to circumvent efficiency problems. (E.g., linear classification under the '0-1' loss for non-separable data is NP-hard, but classification via logistic regression or soft-margin SVM is efficient.)
- Under relatively weak assumptions on the model, e.g., Lipschitzness of the loss and boundedness of the hypothesis class, we can efficiently learn the model (in the agnostic PAC sense) via:
  Stochastic Gradient Descent ***or*** Regularized Loss Minimization
- Regularization is tied to the algorithmic stability of the learner.
- Stability is directly related to the ability of the learner to generalize.
- Two popular algorithms for linear classifiers: Perceptron and SVMs.