Lecture 1: Prediction and PAC learning

Akshay Krishnamurthy
akshay@cs.umass.edu

September 5, 2017

1 Probabilistic Prediction

For the most part, in this course we will focus on prediction problems. The high level goal is to use training data to find a decision rule that makes good predictions. More formally a prediction problem is defined by

- An instance space $\mathcal{X}$, from which the items we are interested in making predictions for live.
- A label space $\mathcal{Y}$, where the actual predictions live.
- A loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that we use to measure the quality of our predictions.

Brief course overview. In the first half of the course, we will focus on a probabilistic or statistical framework for prediction problems, which is central to *statistical learning theory*. Toward the second half, we will explore the other popular model, known as *online learning*. Finally at the end we will depart from prediction problems and study *unsupervised learning*.

Probabilistic Prediction. In the probabilistic prediction setting, the main assumption we make is *distributional*, that is, all labeled examples are drawn iid from a fixed but unknown distribution P. Once we have committed to this assumption, we can evaluate any prediction rule $f : \mathcal{X} \to \mathcal{Y}$ by its *risk*:

$$R(f) = \mathbb{E}_{(X,Y)\sim P}\ell(f(X), Y).$$

Since ℓ is a *loss* function, we would like to find a prediction rule f that has low risk. For most problems of interest, this optimal f can be computed analytically *if* the distribution P is known.

Example 1. *In binary classification, the label space is $\mathcal{Y} = \{0, 1\}$ and the loss we are usually interested in is the $0/1$ loss $\ell(y, y') = \mathbf{1}\{y \neq y'\}$. In this case, the risk is:*

$$R(f) = \mathbb{E}_{(X,Y)\sim P}\mathbf{1}\{f(X) \neq Y\} = \mathbb{P}_{X,Y}[f(X) \neq Y],$$

which is just the probability of error. The optimal prediction rule f, called the Bayes optimal classifier is defined in terms of the conditional probabilities $\eta(x) = \mathbb{P}[Y = 1|X = x]$ as $f^\star(x) = \mathbf{1}\{\eta(x) \geq 1/2\}$. (Exercise: prove that $f^\star$ minimizes the risk. What is its risk?).

However, typically the distribution P is unknown! Instead of getting complete access to P, we instead assume that we have access to a training sample $S = ((X_1, Y_1), \ldots, (X_n, Y_n))$ drawn independently and identically from P. Our goal then is to use the training sample S to find a good predictor $f_S : \mathcal{X} \to \mathcal{Y}$. A learning algorithm is a procedure that takes a training set S and produces a prediction rule f_S.

One of the most popular learning algorithsm is *empirical risk minimization.* Here before seeing any data, we first commit to a subset of functions $\mathcal{F} \subset (\mathcal{X} \to \mathcal{Y})$, and then, upon seeing the data, we output

$$f_{S,ERM} = \operatorname*{argmin}_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \ell(f(X_i), Y_i).$$

Essentially all learning algorithms commit to a function class before hand, but not all of them end up computing the empirical risk minimizer. In such cases, to understand the performance of such a learning rule, it can be useful to decompose things as

$$R(f_S) - R(f^\star) \leq \underbrace{R(f_S) - R(f_{S,\mathrm{ERM}})}_{Computation} + \underbrace{R(f_{S,\mathrm{ERM}}) - \inf_{f\in\mathcal{F}} R(f)}_{Estimation} + \underbrace{\inf_{f\in\mathcal{F}} R(f) - R(f^\star)}_{approximation}$$

The left hand side is how well we are doing, relative to the best we could possible hope for on this problem. On the right we have three terms that help to isolate the main issues.

1. *Computation.* Finding the ERM can be computationally hard or we may want to run a very simple algorithm, in which case we may incur some error here.
2. *Estimation/Statistics.* How much worse it was to operate over the sample instead of the true distribution?
3. *Approximation.* How much worse is it to operate over the function class $\mathcal{F}$?

Let's see an example to fix some of the ideas.

Example 2 (Linear regression)**.** *Let $\mathcal{X} = B_2(1, d)$, the unit Euclidean ball in d dimensions, let $\mathcal{Y} = \mathbb{R}$, and let $\ell(y, y') = \frac{1}{2}(y - y')^2$ be the square loss. If we let $\mathcal{F} = \{f_w(x) = \langle w, x\rangle \mid w \in \mathbb{R}^d\}$, then we are doing standard linear regression. Given a dataset $\{(x_i, y_i)\}_{i=1}^n$, the empirical risk is:*

$$\hat{R}_n(w) = \frac{1}{2n}\sum_{i=1}^n (\langle w, x_i\rangle - y_i)^2.$$

An empirical risk minimizer can be computed analytically here by taking derivatives and setting equal to zero:

$$0 = \frac{\partial \hat{R}_n(w)}{\partial w} = \frac{1}{n}\sum_{i=1}^n x_i(\langle w, x_i\rangle - y_i).$$

From this we can see that $\hat{w} = (\sum_{i=1}^n x_i x_i^T)^\dagger \sum_i x_i y_i$ is an ERM.

For this $\hat{w}$, clearly Term 1 is zero. For Terms 2 and 3, most analyses assume that $\mathbb{E}y|x = \langle w^\star, x\rangle$ in which case the Bayes optimal predictor is $w^\star$ and Term 3 is zero, but note that the Bayes error $R(w^\star)$ is not zero. It requires more advanced tools but you can show an $O(d/n)$ bound on Term 2, and you'll see something related on Homework 1. We will revisit linear regression later.

We can also use gradient descent to compute an approximate ERM. The idea in gradient descent is to minimize the empirical risk with an iterative algorithm. At round t, we are at some parameter w_t, and we move to

$$w_{t+1} \leftarrow w_t - \eta_t \nabla \hat{R}_n(w_t) = w_t - \eta_t \frac{1}{n}\sum_{i=1}^n x_i(\langle w_t, x_i\rangle - y_i).$$

If we run T iterations of gradient descent here, we can make Term 1 $O(\exp(-T))$.

As a final algorithm, we can also run stochastic gradient descent, where instead of looking at all n terms, we sample one at random and just take the gradient with respect to that part of the empirical risk:

$$w_{t+1} \leftarrow w_t - \eta_t \nabla \ell_{i_t}(w_t) = w_t - \eta_t x_{i_t}(\langle w_t, x_{i_t}\rangle - y_{i_t}).$$

Running T rounds of stochastic gradient descent, we can make Term 1 $O(1/T)$.

Now we can start thinking about how to trade-off statistics and computation. The exact approach requires inverting a $d \times d$ matrix, which could take $O(d^3)$ time, while GD has an $O(nd)$-per iteration running time and SGD has $O(d)$-per iteration running time.

In the above example, we saw several choices pan out in the design of our algorithm

1. Which loss function? We used the squared loss above and already saw the 0/1 loss, but there are many others. This leads to a fairly rich theory on surrogate losses and calibration, which we'll study later in the course.

2. Which $\mathcal{F}$ should we use? We used linear functions in the example, but there are many other choices, polynomials, kernels, non-linear functions, etc. Thinking about what function class to use leads to topics like feature selection, model selection, regularization, and kernel methods.

3. What algorithm to compute f_S? We saw both an analytical method and also an iterative method above. But there are many others. This choice usually only affects Term 1, but can in some cases affect Term 2. We'll see many algorithms throughout the course.

In the above example we also saw one assumption that helps with the analysis. More generally for analysis it can be helpful to make assumptions on P. We'll see many different distributional assumptions as we go on.

2 PAC learning

Possibly the simplest learning setting is the PAC, or Probably Approximately Correct learning model. This was introduced by Les Valiant in 1984. Going back to the modeling choices, the main ones are:

1. Label space $\mathcal{Y} = \{0, 1\}$ and the focus is on binary classification.

2. Loss function $\ell(y, y') = \mathbf{1}\{y \neq y'\}$ is the 0/1 loss.

3. Function class: $\mathcal{H} \subset (\mathcal{X} \to \{0, 1\})$. In PAC learning we usually call $\mathcal{H}$ a hypothesis class.

4. Distribution Assumptions: *realizability* $h^\star \in \mathcal{H}$ is a perfect classifier, so $\mathcal{X}$-marginal is arbitrary, but $P(Y = h^\star(x)|X = x) = 1$ for all $x \in \mathcal{X}$.

Definition 1 (PAC Learnability)**.** *A hypothesis class $\mathcal{H}$ is PAC-learnable if there exists a function $n_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0, 1)$, every distribution $\mathcal{D}$ over $\mathcal{X}$ and every labeling function $f : \mathcal{X} \to \{0, 1\}$, if $f \in \mathcal{H}$, then when running the learning algorithm on a training set of size $n \geq n_{\mathcal{H}}(\epsilon, \delta)$ examples drawn from $\mathcal{D}$ and labeled by f, the algorithm outputs a hypothesis $\hat{h}$ such that with probability at least $1 - \delta$ (over the training set), $R(\hat{h}) \leq \epsilon$.*

Definition 2 (Sample Complexity)**.** *The pointwise minimal function $n_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ that satisfies the definition of PAC-learnability is called the sample complexity.*

There are two parameters we need to unpack.

1. Failure probability δ: When the training set is random, there is always some probability that it is completely noninformative. Consider aproblem with $|\mathcal{X}| = 2$ with uniform $\mathcal{D}$. With probability $1/2^n$ the dataset will be all x_1 so you will err on x_2. This hints that the dependence on δ should be exponentially small. Such circumstances fall into the δ failure probability.

2. Approximation parameter ϵ: even when the training set is informative there will always be some parts of the decision rule that training sample fails to capture. Think about a problem where two hypotheses disagree on just one example but $\mathcal{X}$ is quite large. It is likely you will not see that one example, so you'll confuse these two hypotheses, but they aren't too different.

To start off our PAC analysis, let us analyze the empirical risk minimizer.

$$\hat{h}_S = \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{h(X_i) \neq Y_i\}$$

Theorem 3 (PAC-bound for finite hypothesis class)**.** *Let $\mathcal{H}$ be a hypothesis class with finite size. Then $\mathcal{H}$ is PAC-learnable with*

$$n_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{\log(|\mathcal{H}|/\delta)}{\epsilon} \right\rceil.$$

Proof. First, due to realizability, there exists $h \in \mathcal{H}$ with $\hat{R}(h) = 0$, namely $h^\star$, so the ERM h_S also has $\hat{R}(h_S) = 0$.

Let $\mathcal{H}_B = \{h \in \mathcal{H} \mid R(h) \geq \epsilon\}$ denote the bad hypotheses and let $S_B = \{S \in (\mathcal{X} \times \mathcal{Y})^n \mid \exists h \in \mathcal{H}_B, R_S(h) = 0\}$ denote the training sets that might mislead us into choosing one of these bad hypotheses. It is important to observe that the only way we can fail is if we get one of these misleading samples. Thus, we want to upper bound the probability of seeing a training set from S_B,

We can re-write

$$S_B = \bigcup_{h \in \mathcal{H}_B} \{S \in (\mathcal{X} \times \mathcal{Y})^n \mid R_S(h) = 0\}.$$

Now by the union bound and elementary calculations

$$\begin{aligned}
\mathbb{P}_{S\sim\mathcal{D}}[S \in S_B] = \mathbb{P}_{S\sim\mathcal{D}}[\bigcup_{h\in\mathcal{H}_B} \{R_S(h) = 0\}] &\leq \sum_{h\in\mathcal{H}_B} \mathbb{P}_{S\sim\mathcal{D}}[\{R_S(h) = 0\}] && \text{(Union bound)} \\
&= \sum_{h\in\mathcal{H}_B} \mathbb{P}_{S\sim\mathcal{D}}[\{\forall i. h(x_i) = h^\star(x_i)\}] && \text{(Realizability)} \\
&= \sum_{h\in\mathcal{H}_B} \prod_{i=1}^{n} \mathbb{P}_{x\sim\mathcal{D}}[\{h(x) = h^\star(x)\}] && \text{(Independence)} \\
&\leq \sum_{h\in\mathcal{H}_B} \prod_{i=1}^{n} (1-\epsilon) \leq |\mathcal{H}|(1-\epsilon)^n \leq |\mathcal{H}| \exp(-\epsilon n) && (h \text{ is bad})
\end{aligned}$$

Rearranging, we can conclude that if $n \geq \lceil \log(|\mathcal{H}/\delta)/\epsilon \rceil$ the probability of this event, and hence the failure probability, will be at most δ, proving the theorem. □

Example 3 (Boolean monotone conjunctions)**.** *Let $\mathcal{X} = \{0,1\}^d$ be the boolean hypercube and let $\mathcal{H}$ be the class of boolean conjunctions with only positive literals (e.g., $h(x) = x_1 \wedge x_3 \wedge x_7$). Observe that $|\mathcal{H}| \leq 2^d$ since each variable can be in or out of the clause. Thus the sample complexity of learning boolean conjunction is $O(d \log(1/\delta)/\epsilon)$.*

Moreover while the hypothesis space is exponentially large, there is an efficient algorithm for computing the ERM. Simply look through all the positive instances and find any variable set to zero in any of these positive examples. That variable cannot possibly be included in the ERM conjunction. So include all the variables that remain.

Example 4 (Perceptron)**.** *A more interesting example involves learning linear separators. Let $\mathcal{X} = \mathbb{R}^d$ and let $\mathcal{H}$ be the set of linear functions, i.e. $\mathcal{H} = \{h_w(x) = \text{sign}\{\langle w, x\rangle > 0\} \mid w \in \mathbb{R}^d\}$. This class is also PAC-learnable, although the function class is infinitely large, so we cannot apply Theorem 3. We will build up the tools to analyze this class in the next few lectures.*

In the realizable case we can compute the ERM using linear programming. For convenience let $\mathcal{Y} = \{-1, 1\}$. The ERM is just a linear feasibility problem:

$$?\exists w \text{ s.t. } \forall i \in [n], y\langle w, x\rangle \geq 0.$$

We can also use the perceptron algorithm. The algorithm is iterative and starts with $w_0 = 0$. Then while some i has $y_i \neq \text{sign}\{\langle w_t, x_i\rangle\}$, we update

$$w_{t+1} \leftarrow w_t + y_i x_i.$$

Claim 4. *Suppose the data is linearly separable by $w^\star$ and define $r = \max_i \|x_i\|^2$ and $\gamma = \min_i(\langle w^\star, x_i\rangle y_i / \|w^\star\|_2)$. Then the perceptron algorithm is guaranteed to terminate in at most r^2/γ^2 updates.*

Proof.

$$\langle w_{t+1}, w^\star\rangle = \langle w_t, w^\star\rangle + \langle y_i x_i, w^\star\rangle \geq \langle w_t, w^\star\rangle + \gamma\|w^\star\|_2$$

And hence $\langle w_T, w^\star\rangle \geq T\gamma\|w^\star\|_2$. At the same time $\|w_{t+1}\|_2^2 = \|w_t + y_i x_i\|_2^2 \leq \|w_t\|_2^2 + r^2$ since the cross term is negative. Thus after T iterations, we have

$$T\gamma\|w^\star\|_2 \leq \langle w_T, w^\star\rangle \leq \|w_T\|_2\|w^\star\|_2 \leq \sqrt{T}r\|w^\star\|_2$$

If $T > r^2/\gamma^2$ this is a contradiction, which means the algorithm must terminate. □