Lecture 2: Probably Approximately Correct Learning

Tianpei Xie

Jul. 30th., 2015

Contents

1 PAC Learning in Deterministic Setting

1.1 Definitions

- **Remark** *In deterministic scenario*, denote a collection of n *independent identically distributed (i.i.d.)* ***random samples*** generated by P_X as $\mathcal{D}_n$, i.e.

$$\mathcal{D}_n := \{X_i : 1 \leq i \leq n\}.$$

Denote $\mathcal{C}$ as *the set of all concepts* we wish to learn as the ***concept class***:

$$\mathcal{C} := \{c : \mathcal{X} \to \mathcal{Y}\} = \mathcal{Y}^{\mathcal{X}}.$$

A ***learner*** considers a ***fixed*** *subset of concepts* $\mathcal{H} \subset \mathcal{C}$, which is referred as a ***hypothesis class***, and provides a ***hypothesis*** or a ***classifier*** or a ***decision function*** $g \in \mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$ based on $\mathcal{D}$. The task of ***supervised learning*** is to minimize *the generalization error*:

$$L(g) = \mathcal{P}_X \{g(X) \neq c(X)\} \equiv \mathbb{E}_X \left[\mathbb{1}_{g(X) \neq c(X)}\right] \tag{1}$$

where $\mathbb{1}_\omega$ is the indicator function of the event ω.

The generalization error of a hypothesis is not directly accessible to the learner since both the distribution $\mathcal{P}$ and the target concept c are unknown. However, the learner can measure ***the empirical error*** of a hypothesis on the labeled sample $\mathcal{D}_n$:

$$\widehat{L}_n(g) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1} \{g(X_i) \neq c(X_i)\}, \qquad \text{(deterministic setting)}.$$

Recall that the expectation of empirical error under $\mathcal{P}_X$ is the generalization error

$$\mathbb{E}_X \left[\widehat{L}_n(g)\right] = L(g).$$

- We denote by $\mathcal{O}(d)$ an *upper bound* on *the* ***cost*** *of the* ***computational representation*** of any element $x \in \mathcal{X} \subset \mathbb{R}^d$ and by $\text{size}(c)$ ***the maximal cost*** *of the computational representation of* $c \in \mathcal{C}$. For example, x may be a vector in $\mathbb{R}^d$, for which the cost of an array-based representation would be in $\mathcal{O}(n)$.

Definition (***Probably Approximately Correct (PAC) Learning***)
A ***concept class*** $\mathcal{C}$ is said to be ***Probably Approximately Correct-learnable*** if there exists an *algorithm* $\mathcal{A}$ and a ***polynomial function*** $\text{poly}(\cdot,\cdot,\cdot,\cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for ***all distributions*** $\mathcal{P}$ on X and *for* ***any target concept*** $c \in \mathcal{C}$, the following holds for ***any sample size*** $n \geq \text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$:

$$\begin{aligned} &\mathcal{P}_{\mathcal{D}_n} \{L(g_n(\cdot|\mathcal{D}_n)) \leq \epsilon\} \geq 1 - \delta \\ \Leftrightarrow\ &\mathcal{P}_{\mathcal{D}_n} \{L(g_n(\cdot|\mathcal{D}_n)) \geq \epsilon\} \leq \delta \end{aligned} \tag{2}$$

If $\mathcal{A}$ further runs in $\text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$, then $\mathcal{C}$ is said to be ***efficiently PAC-learnable***. When such an algorithm $\mathcal{A}$ exists, it is called a ***PAC-learning algorithm*** for $\mathcal{C}$.

- **Remark** *A concept class* $\mathcal{C}$ is thus *PAC-learnable* if the hypothesis returned by the algorithm after observing a number of points *polynomial* in $1/\epsilon$ and $1/\delta$ is

1. ***approximately correct*** (*error at most* ϵ)
2. ***with high probability*** (at least $1-\delta$),

which justifies *the PAC terminology*.

1. $\delta > 0$ is used to define ***the confidence*** $1-\delta$
2. and $\epsilon > 0$ ***the accuracy*** $1-\epsilon$.

Note that if the running time of the algorithm is ***polynomial*** in $1/\epsilon$ and $1/\delta$, then ***the sample size*** n must also be polynomial if the full sample is received by the algorithm.

- **Remark** Several key points of the PAC definition are worth emphasizing.

1. First, *the PAC framework is a* ***distribution-free model***: *no particular assumption* is made about *the distribution* $\mathcal{P}_X$ from which examples are drawn.
2. Second, *the training sample* and *the test examples* used to define the error are drawn *according to* ***the same distribution*** $\mathcal{P}_X$. This is a necessary assumption for generalization to be possible in most cases.
3. Finally, *the PAC framework deals with the question of* ***learnability for a concept class*** $\mathcal{C}$ *and not a particular concept*. Note that ***the concept class*** $\mathcal{C}$ ***is known to the algorithm***, but of course target concept $c \in \mathcal{C}$ is unknown.

We may omit the polynomial dependency on n and size(c) in the PAC definition and *focus only on the* ***sample complexity***.

1.2 PAC-Learnable Guarantees for Finite Hypothesis Sets

- **Remark** (***Finite Hypothesis Sets***)
We will see if the size of hypothesis class $\mathcal{H}$ is finite, i.e. $|\mathcal{H}| < \infty$, we can achieve *PAC learnability* if the target $c \in \mathcal{H}$:

- **Proposition 1.1** ***(Learning bounds, Finite*** $\mathcal{H}$***, Consistent Case)*** *[Mohri et al., 2018] Let* $\mathcal{H}$ *be a* ***finite*** *set of functions mapping from* $\mathcal{X}$ *to* $\mathcal{Y}$. *Let* $\mathcal{A}$ *be an algorithm that for* ***any target concept*** $c \in \mathcal{H}$ *and i.i.d. sample* $\mathcal{D}_m$ *returns a* ***consistent hypothesis*** g_m, *i.e. the training error of* g_m *is zero:*

$$\widehat{L}_m(g_m) = 0$$

Then, for any $\epsilon, \delta > 0$, *the inequality*

$$\mathcal{P}_{\mathcal{D}_m}\left\{L(g_m(\cdot|\mathcal{D}_m)) \leq \epsilon\right\} \geq 1-\delta$$

holds if

$$m \geq \frac{1}{\epsilon}\left(\log|\mathcal{H}| + \log\frac{1}{\delta}\right) \tag{3}$$

This sample complexity result admits the following ***equivalent statement*** *as a generalization bound: for any* $\epsilon, \delta > 0$, *with* ***probability*** *at least* $1-\delta$,

$$L(g_m(\cdot|\mathcal{D}_m)) := \mathcal{P}_{\mathcal{D}_m}\left\{g_m(X|\mathcal{D}_m) \neq c(X)\right\} \leq \frac{1}{m}\left(\log|\mathcal{H}| + \log\frac{1}{\delta}\right). \tag{4}$$

- To prove the general case, we use a bound from *Hoeffding's inequality*:

  Proposition 1.2 *Fix $\epsilon > 0$ and let $\mathcal{D}_m$ denote an i.i.d. sample of size m. Then, for any hypothesis $g : \mathcal{X} \to \{0, 1\}$, the following inequalities hold:*

$$\mathcal{P}_{\mathcal{D}_m}\left\{\widehat{L}(g) - L(g) \geq \epsilon\right\} \leq \exp\left(-2m\epsilon^2\right) \tag{5}$$

$$\mathcal{P}_{\mathcal{D}_m}\left\{\widehat{L}(g) - L(g) \leq -\epsilon\right\} \leq \exp\left(-2m\epsilon^2\right) \tag{6}$$

  By the union bound, this implies the following two-sided inequality:

$$\mathcal{P}_{\mathcal{D}_m}\left\{\left|\widehat{L}(g) - L(g)\right| \geq \epsilon\right\} \leq 2\exp\left(-2m\epsilon^2\right). \tag{7}$$

  Setting the right-hand side of (7) to be equal to δ and solving for ϵ yields immediately the following bound for *a single hypothesis*

  Corollary 1.3 ***(Generalization bound for Single Hypothesis)*** *[Mohri et al., 2018] Fix a hypothesis $g : \mathcal{X} \to \{0, 1\}$. Then, for any $\delta > 0$, the following inequality holds with probability at least $1 - \delta$:*

$$L(g) \leq \widehat{L}(g) + \sqrt{\frac{\log \frac{2}{\delta}}{2m}} \tag{8}$$

  This error bound can be seen as coming from *the* ***randomness*** *of* ***coin tossing*** when approximate the generalization error $L(g)$ by training error. Thus it will always exist for any generalization error bound.

- **Proposition 1.4** ***(Learning bounds, Finite $\mathcal{H}$, Inconsistent Case)*** *[Mohri et al., 2018] Let $\mathcal{H}$ be a* ***finite*** *set of functions mapping from $\mathcal{X}$ to $\mathcal{Y}$. Then, for any $\delta > 0$ with probability at least $1 - \delta$, the following inequality holds: for all $g \in \mathcal{H}$*

$$L(g) \leq \widehat{L}(g) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{2}{\delta}}{2m}} \tag{9}$$

  Thus for a finite hypothesis set $\mathcal{H}$,

$$L(g) \leq \widehat{L}(g) + O\left(\sqrt{\frac{\log |\mathcal{H}|}{m}}\right) \tag{10}$$

- **Remark** (***Sample Efficiency*** $\Leftarrow$ ***Representation Efficiency of Hypothesis Class***)
  As already pointed out, $\log |\mathcal{H}|$ can be interpreted as ***the number of bits*** *needed to* ***represent*** $\mathcal{H}$. Several other remarks similar to those made on the generalization bound in the consistent case can be made here:

  a larger sample size m guarantees ***better generalization****, and the bound* ***increases*** *with $|\mathcal{H}|$, but only* ***logarithmically****.*

  But, here, the bound is a less favorable function of $\log |\mathcal{H}| / m$; it varies as ***the square root*** *of this term.* This is not a minor price to pay: for a *fixed* $|\mathcal{H}|$, to attain the *same guarantee* as *in the* ***consistent case****, a* ***quadratically larger labeled*** *sample is needed.*

- **Example** (***Conjunction of Boolean Literals***) [Mohri et al., 2018]

- **Example** (***k-term DNF Formulae***) [Mohri et al., 2018]

- **Example** (***k-CNF Formulae***) [Mohri et al., 2018]

1.3 PAC-Learnable Examples for Infinite Hypothesis Sets

- **Example** (***Learning Axis-Aligned Rectangles***) [Mohri et al., 2018]
  Consider the case where the set of instances are *points* in the plane, $\mathcal{X} = \mathbb{R}^2$, and the concept class $\mathcal{C}$ is the set of ***all axis-aligned rectangles*** lying in $\mathbb{R}^2$. Thus, each concept c is *the set of points inside a particular axis-aligned rectangle.* The learning problem consists of determining with small error a target axis-aligned rectangle using the labeled training sample. We will show that *the concept class of axis-aligned rectangles* is ***PAC-learnable***.

- **Example** (***Threshold Function Class is Learnable***) [Shalev-Shwartz and Ben-David, 2014, Mohri et al., 2018]
  Let $\mathcal{H}$ be the set of threshold functions over the *real line*, namely,

$$\mathcal{H} = \{h_\alpha : \alpha \in \mathbb{R}\}, \quad \text{where } h_\alpha(x) = \mathbb{1}_{\{x<\alpha\}}.$$

  Clearly, $\mathcal{H}$ is of infinite size.

  Nevertheless, the following lemma shows that $\mathcal{H}$ is *learnable* in *the PAC model* using *the ERM algorithm.*

  Lemma 1.5 *[Shalev-Shwartz and Ben-David, 2014]*
  Let $\mathcal{H}$ be the class of ***thresholds*** *as defined earlier. Then, $\mathcal{H}$ is* ***PAC learnable****, using the ERM rule, with* ***sample complexity*** *of*

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \log\left(\frac{2/\delta}{\epsilon}\right) \right\rceil.$$

2 PAC Learning in Stochastic Setting

- **Definition** (***Agnostic PAC-Learning***)
  Let $\mathcal{H}$ be a hypothesis set. $\mathcal{A}$ is an ***agnostic PAC-learning algorithm*** if there exists a ***polynomial function*** $\text{poly}(\cdot,\cdot,\cdot,\cdot)$ such that for any $\epsilon > 0$ and $\delta > 0$, for all distributions $\mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$, the following holds for any sample size $m \geq \text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$:

$$\mathcal{P}_{\mathcal{D}_m}\left\{L(g_m(\cdot|\mathcal{D}_m)) - \inf_{g\in\mathcal{H}} L(g) \leq \epsilon\right\} \geq 1 - \delta. \tag{11}$$

  If $\mathcal{A}$ further runs in $\text{poly}(1/\epsilon, 1/\delta, d, \text{size}(c))$, then $\mathcal{C}$ is said to be ***efficiently agnostic PAC-learnable***.

- **Definition** (***Bayes Error in Stochastic Scenario***)
  Under a given distribution $\mathcal{P}_{X,Y}$, the ***Bayes error*** L^* or ***Bayes risk*** R^* is defined as

$$L^* = \inf_{h \in \mathcal{Y}^{\mathcal{X}}, \text{ measurable}} \{L(h)\}, \tag{12}$$

  where the infimum is with respect to *all measureable function* $g : \mathcal{X} \to \mathcal{Y}$. And the *hypothesis* g^* such that $L(g^*) = L^*$ is called the ***Bayes classifier***.

- **Remark** (***Estimation Error vs. Approximation Error***)
  The difference between *the error of a hypothesis* $g \in \mathcal{H}$ and *the Bayes error* can be decomposed as:

$$L(g) - L^* = \underbrace{\left(L(g) - \inf_{g \in \mathcal{H}} L(g)\right)}_{\textit{estimation error}} + \underbrace{\left(\inf_{g \in \mathcal{H}} L(g) - L^*\right)}_{\textit{approximation error}}.$$

  where the first term is called ***estimation error*** and the second term is called ***approximation error***.

  When $\mathcal{H}$ and $\mathcal{P}_{X,Y}$ is *fixed*, *the approximation error is fixed*. The definition of ***PAC learnability*** requires that ***the estimation error*** would be *bounded* ***uniformly*** *over* ***all distributions***.

$$\begin{aligned} L(g_m) - \inf_{g \in \mathcal{H}} L(g) &= L(g_m) - \widehat{L}(g_m) + \widehat{L}(g_m) - L(g^*) \\ &\leq L(g_m) - \widehat{L}(g_m) + \widehat{L}(g^*) - L(g^*) \\ &\leq 2 \sup_{g \in \mathcal{H}} \left| L(g) - \widehat{L}(g) \right| \end{aligned}$$

  where $g^* = \operatorname{argmin}_{g \in \mathcal{H}} L(g)$ and $\widehat{L}(g)$ is the training error of g. Thus *the estimation error* can be bounded uniformly by *the generalization error bound* $|L(g) - \widehat{L}(g)|$ for any $g \in \mathcal{H}$.

3 PAC Learning vs. Universal Consistency

- **Remark** (***Universal Consistency is Not Enough***) [Shalev-Shwartz and Ben-David, 2014]
  We compare the *universal consistency* and *PAC learnability* as ***performance guarantee*** for a classification rule:

  1. ***The universal consistency*** of a classification rule provides *a performance guarantee* in terms of ***asymptotic analysis***. It concerns that if *the generalization error* of classfication rule $\{g_n\}$ will reach to *infimum* within given class $\mathcal{H}$ when the sample size is ***infinity*** $n \to \infty$. ***The universal consistency*** is a *meaurable of* ***correctness***, i.e. the classification rule can reach to *correct* solution *eventually* regardless of the underlying distribution of data.

     On the other hand, it does not anwser *how many examples are required to be as good as the best hypothesis in* $\mathcal{H}$. The answer to this question depends on the underlying distribution $\mathcal{P}_{X,Y}$ in consistency statement. *The consistency statement* is more related to ***large sample statistical behavior***.

2. ***The PAC learnability*** of a classification rule provides *a performance guarantee* in terms of ***non-asymptotic analysis***. It provide a *measure of* ***efficiency***. *An* ***algorithm*** *is efficient* if it obtain a solution *close* to *correct* solution (*within an error rate* ϵ) *with high probability* $(1-\delta)$ ***given a finite set of*** $n \geq n(\epsilon, \delta)$ ***samples***.

   The PAC learnability is a measure for the ***algorithm*** as well as ***the statistical nature*** of the problem. Its formulation is closer to *computer science* than to *statistics*.

   Moreover, *the definition of PAC learning* yields ***the limitation*** *of learning (via the* ***No-Free-Lunch theorem****)* and *the* ***necessity*** *of* ***prior knowledge***. It gives us a crisp way to encode prior knowledge by choosing *a hypothesis class*, and once this choice is made, we have a generic learning rule - *ERM*. Compared to PAC learning, the definition of ***consistency*** *does* ***not*** *yield* a natural learning paradigm or a way to ***encode prior knowledge***. In fact, in many cases there is *no need for prior knowledge at all* since the consistency only cares about the *asymptotic behavior*.

Consider the classification prediction algorithm ***Memorize*** defined as follows. The algorithm *memorizes* the training examples, and, given a test point x, it predicts the majority label among all labeled instances of x that exist in the training sample (and some fixed default label if no instance of x appears in the training set). It is possible to show (see Exercise below) that the ***Memorize*** algorithm is ***universally consistent*** for ***every countable domain*** $\mathcal{X}$ and a ***finite label*** set $\mathcal{Y}$ (w.r.t. the zero-one loss).

Intuitively, it is *not obvious* that the *Memorize* algorithm should be viewed as a *learner*, since it ***lacks*** the aspect of ***generalization***, namely, of using observed data to *predict* the labels of unseen examples. The fact that *Memorize* is a *consistent algorithm* for the class of *all functions over any countable domain set* therefore raises *doubt* about ***the usefulness of consistency guarantees***. Furthermore, the sharp-eyed reader may notice that the "*bad learner*" we introduced in Chapter 2, which led to *overfitting*, is in fact the *Memorize* algorithm.

- **Exercise 3.1** ***(Memorize Algorithm)***
  In this example we wish to show that the algorithm ***Memorize*** *is a* ***consistent learner*** *for* ***every class of (binary-valued) functions*** *over* ***any countable domain****.*

  Let $\mathcal{X}$ be a ***countable*** *domain and let $\mathcal{P}$ be a probability distribution over $\mathcal{X}$.*

  1. *Let $\{x_i : i \in \mathbb{N}\}$ be an enumeration of the elements of $\mathcal{X}$ so that for all $i \leq j$, $\mathcal{P}(\{x_i\}) \leq \mathcal{P}(x_j)$. Prove that*

     $$\lim_{n\to\infty} \sum_{i \geq n} \mathcal{P}(\{x_i\}) = 0.$$

     Note that $\sum_{i=0}^{\infty} \mathcal{P}(\{x_i\}) = 1$ by definition of probability measure.

  2. *Given any $\epsilon > 0$, prove that there exists $\epsilon_{\mathcal{P}} > 0$ such that*

     $$\mathcal{P}\{x \in \mathcal{X} : \mathcal{P}(\{x\}) < \epsilon_{\mathcal{P}}\} < \epsilon.$$

  3. *Prove that for every $\eta > 0$, if n is such that $\mathcal{P}(\{x_i\}) < \eta$ for all $i > n$, then for every $m \in \mathbb{N}$, let $\mathcal{D}_m$ be the sample of size m generated according to $\mathcal{P}$*

     $$\mathcal{P}_{\mathcal{D}_m}\{\exists x_i : \mathcal{P}(\{x_i\}) > \eta \text{ and } x_i \notin \mathcal{D}_m\} \leq ne^{-\eta m}.$$

4. *Conclude that if $\mathcal{X}$ is* ***countable*** *then for* ***every probability distribution*** $\mathcal{P}$ *over $\mathcal{X}$ there exists a function $m_{\mathcal{P}} : (0,1)\times(0,1) \to \mathbb{N}$ such that for every $\epsilon, \delta > 0$ if $m > m_{\mathcal{P}}(\epsilon, \delta)$ then*

$$\mathcal{P}_{\mathcal{D}_m}\{\mathcal{P}(\{x : x \notin \mathcal{D}_m\}) > \epsilon\} < \delta.$$

5. *Prove that* ***Memorize*** *is a consistent learner for every class of (binary-valued) functions over any countable domain.*

- **Remark** (***Universal Consistency May Not be Good Preference***) [Shalev-Shwartz and Ben-David, 2014]
  One may argue that even though *consistency is a weak requirement*, it is desirable that *a learning algorithm will be* ***consistent*** *with respect to the set of* ***all functions*** *from $\mathcal{X}$ to $\mathcal{Y}$*, which gives us a *guarantee* that for enough training examples, we will always be as good as *the Bayes optimal predictor*. Therefore, if we have two algorithms, where one is *consistent* and the other one is *not consistent*, we should *prefer* the consistent algorithm. ***However***, *this argument is* ***problematic*** for two reasons.

  1. First, maybe it is the case that for most "*natural*" distributions we will observe in practice that ***the sample complexity*** *of the* ***consistent*** *algorithm* will be *so* ***large*** so that in every practical situation we will *not obtain enough examples* to enjoy this guarantee.

  2. Second, it is ***not very hard to make any PAC*** *or* ***nonuniform learner consistent*** *with respect to the class of all functions from $\mathcal{X}$ to $\mathcal{Y}$.*

     Concretely, consider a countable domain, $\mathcal{X}$, a finite label set $\mathcal{Y}$, and a hypothesis class, $\mathcal{H}$, of functions from $\mathcal{X}$ to $\mathcal{Y}$. We can make any nonuniform learner for H be consistent with respect to the class of all classifiers from $\mathcal{X}$ to $\mathcal{Y}$ using the following simple trick: Upon receiving a training set, we will first run the nonuniform learner over the training set, and then we will obtain a *bound* on the true risk of the learned predictor. If this bound is *small enough* we are done. Otherwise, we revert to the *Memorize* algorithm. This simple modification makes the algorithm consistent with respect to all functions from $\mathcal{X}$ to $\mathcal{Y}$.

     Since ***it is easy to make any algorithm consistent***, it may ***not be wise*** to ***prefer*** one algorithm over the other just because of consistency considerations.

References

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.