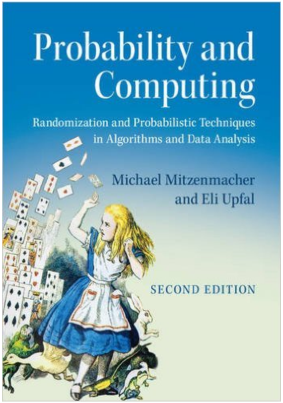
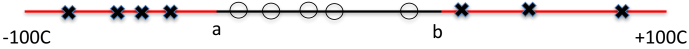
CS155/254: Probabilistic Methods in Computer Science

Chapter 14.1: Sample Complexity - Statistical Learning Theory



Statistical Learning – Learning From Examples

- We want to estimate the working temperature range of an iPhone.
  - We could study the physics and chemistry that affect the performance of the phone – too hard
  - We could sample temperatures in [-100C,+100C] and check if the iPhone works in each of these temperatures
  - We could sample users' iPhones for failures/temperature
- How many samples do we need?
- How good is the result?



Learning an Interval From Examples

- Our domain is $[A, B] \subset (-\infty, +\infty)$. There is an unknown distribution D on $[A, B]$
- There is an unknown classification of the domain to an interval of points in class In, the rest are in class Out.
- We get n random training (labeled) examples from the distribution D.
- We choose a rule $r = [a, b]$ based on the examples.
- We use this rule to decide on an unlabeled point drawn from D.
- Let $r^* = [c, d]$ be the correct rule.
- Let $\Delta(r, r^*) = ([a, b] - [c, d]) \cup ([c, d] - [a, b])$
- We are wrong only on examples in $\Delta(r, r^*)$.

What's the probability that we are wrong?

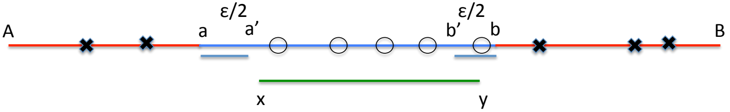
- If we select r, we are wrong only on examples in $\Delta(r, r^*)$.
- The probability that we are wrong is $Pr(\Delta(r, r^*))$.
- If $Prob(\Delta(r, r^*)) \leq \epsilon$ we don't care.
- We bound $Prob$(select r such that $Pr(\Delta(r, r^*) \geq \epsilon))$ as a function of the size of the training set.

Two probabilities:

1. ϵ - the probability that our rule gives a wrong answer.
2. δ - the probability that are sample is sufficiently good to generate such a rule.

Learning an Interval

- If the classification error is ≥ ε then the sample missed at least one of the the intervals [a,a’] or [b’,b] each of probability ≥ ε/2



Each sample excludes many possible intervals.
The union bound sums over overlapping hypothesis.
Need better characterization of concept's complexity!

Theorem

There is a learning algorithm that given a sample from $\mathcal{D}$ *of size* $m = \frac{2}{\epsilon} \ln \frac{2}{\delta}$*, with probability* $1 - \delta$*, returns a classification rule (interval)* $[x, y]$ *that is correct with probability* $1 - \epsilon$*.*

Proof.

Algorithm: Choose the smallest interval $[x, y]$ that includes all the "In" sample points.

- Clearly $a \leq x < y \leq b$, and the algorithm can only err in classifying "In" points as "Out" points.
- Fix $a < a'$ and $b' < b$ such that $Pr([a, a']) = \epsilon/2$ and $Pr([b, b']) = \epsilon/2$.
- If the probability of error when using the classification $[x, y]$ is $\geq \epsilon$ then either $a' \leq x$ or $y \leq b'$ or both.
- The probability that the sample of size $m = \frac{2}{\epsilon} \ln \frac{2}{\delta}$ did not intersect with one of these intervals is bounded by

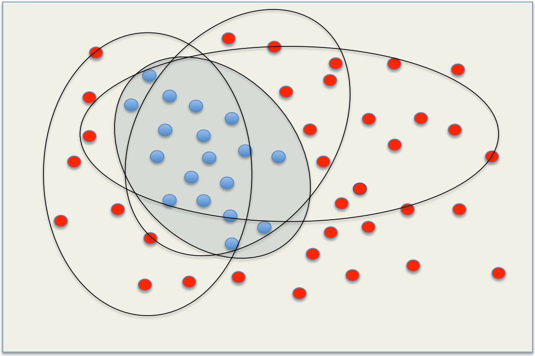
$$2(1 - \frac{\epsilon}{2})^m \leq e^{-\frac{\epsilon m}{2} + \ln 2} = e^{-\frac{\epsilon}{2} \frac{2}{\epsilon} \ln \frac{2}{\delta} + \ln 2} = \delta$$

Learning a Binary Classifier

- An unknown probability distribution $\mathcal{D}$ on a domain $\mathcal{U}$
- An unknown correct classification – a partition c of $\mathcal{U}$ to *In* and *Out* sets
- Input:
  - Concept class $\mathcal{C}$ – a collection of possible classification rules (partitions of U).
  - A training set $\{(x_i, c(x_i)) \mid i = 1, \dots, m\}$, where $x_1, \dots, x_m$ are sampled from $\mathcal{D}$.
- Goal: With probability $1 - \delta$ the algorithm generates a *good* classifier.
- A classifier is *good* if the probability that it errs on an item generated from $\mathcal{D}$ is $\leq opt(\mathcal{C}) + \epsilon$, where $opt(\mathcal{C})$ is the error probability of the best classifier in $\mathcal{C}$.
- Realizable case: $c \in \mathcal{C}$, $Opt(C) = 0$.
- Unrealizable case: $c \notin \mathcal{C}$, $Opt(C) > 0$.

Learning a Binary Classifier

- Out and In items, and a concept class C of possible classification rules



When does the sample specify a *good* rule? The realizable case

- The realizable case - the correct classification $c \in \mathcal{C}$.
- For any $h \in \mathcal{C}$ let $\Delta(c, h)$ be the set of items on which the two classifiers differ: $\Delta(c, h) = \{x \in U \mid h(x) \neq c(x)\}$
- Algorithm: choose $h^* \in \mathcal{C}$ that agrees with all the training set (there must be at least one).
- If the sample (training set) intersects every set in

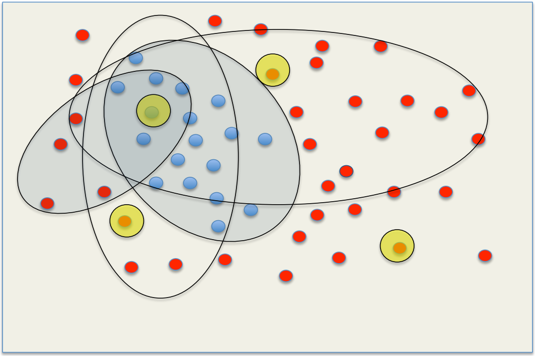
$$\{\Delta(c, h) \mid Pr(\Delta(c, h)) \geq \epsilon\},$$

then

$$Pr(\Delta(c, h^*)) \leq \epsilon.$$

Learning a Binary Classifier

- Red and blue items, possible classification rules, and the sample items



When does the sample identify a *good* rule? The unrealizable (agnostic) case

- The unrealizable case - c may not be in $\mathcal{C}$.
- For any $h \in \mathcal{C}$, let $\Delta(c, h)$ be the set of items on which the two classifiers differ: $\Delta(c, h) = \{x \in U \mid h(x) \neq c(x)\}$
- For the training set $\{(x_i, c(x_i)) \mid i = 1, \ldots, m\}$, let

$$\tilde{Pr}(\Delta(c, h)) = \frac{1}{m} \sum_{i=1}^{m} 1_{h(x_i) \neq c(x_i)}$$

- Algorithm: choose $h^* = \arg\min_{h \in \mathcal{C}} \tilde{Pr}(\Delta(c, h))$.
- If for every set $\Delta(c, h)$,

$$|Pr(\Delta(c, h)) - \tilde{Pr}(\Delta(c, h))| \leq \epsilon,$$

then

$$Pr(\Delta(c, h^*)) \leq opt(\mathcal{C}) + 2\epsilon.$$

where $opt(\mathcal{C})$ is the error probability of the best classifier in $\mathcal{C}$.

If for every set $\Delta(c, h)$,

$$|Pr(\Delta(c, h)) - \tilde{P}r(\Delta(c, h))| \leq \epsilon,$$

then

$$Pr(\Delta(c, h^*)) \leq opt(\mathcal{C}) + 2\epsilon.$$

where $opt(\mathcal{C})$ is the error probability of the best classifier in $\mathcal{C}$. Let $\bar{h}$ be the best classifier in $\mathcal{C}$. Since the algorithm chose h^*,

$$\tilde{P}r(\Delta(c, h^*)) \leq \tilde{P}r(\Delta(c, \bar{h})).$$

Thus,

$$\begin{aligned} Pr(\Delta(c, h^*)) - opt(\mathcal{C}) &\leq \tilde{P}r(\Delta(c, h^*)) - opt(\mathcal{C}) + \epsilon \\ &\leq \tilde{P}r(\Delta(c, \bar{h})) - opt(\mathcal{C}) + \epsilon \leq 2\epsilon \end{aligned}$$

Detection vs. Estimation

- Input:
  - Concept class $\mathcal{C}$ – a collection of possible classification rules (partitions of U).
  - A training set $\{(x_i, c(x_i)) \mid i = 1, \ldots, m\}$, where $x_1, \ldots, x_m$ are sampled from $\mathcal{D}$.
- For any $h \in \mathcal{C}$, let $\Delta(c, h)$ be the set of items on which the two classifiers differ: $\Delta(c, h) = \{x \in U \mid h(x) \neq c(x)\}$
- For the realizable case we need a training set (sample) that with probability $1 - \delta$ intersects every set in

$$\{\Delta(c, h) \mid Pr(\Delta(c, h)) \geq \epsilon\} \quad (\epsilon\text{-net})$$

- For the unrealizable case we need a training set that with probability $1 - \delta$ estimates, within additive error ϵ, every set in

$$\Delta(c, h) = \{x \in U \mid h(x) \neq c(x)\} \quad (\epsilon\text{-sample}).$$

Uniform Convergence Sets

Given a collection R of sets in a universe X, under what conditions a finite sample N from an arbitrary distribution $\mathcal{D}$ over X, satisfies with probability $1-\delta$,

1. $$\forall r \in R, \ \Pr_{\mathcal{D}}(r) \geq \epsilon \ \Rightarrow \ r \cap N \neq \emptyset \qquad (\epsilon\text{-net})$$

2. for any $r \in R$,

$$\left| \Pr_{\mathcal{D}}(r) - \frac{|N \cap r|}{|N|} \right| \leq \varepsilon \qquad (\epsilon\text{-sample})$$

Learnability - Uniform Convergence

Theorem

In the realizable case, any concept class $\mathcal{C}$ *can be learned with* $m = \frac{1}{\epsilon}(\ln|\mathcal{C}| + \ln\frac{1}{\delta})$ *samples.*

Proof.

We need a sample that intersects every set in the family of sets

$$\{\Delta(c, c') \mid Pr(\Delta(c, c')) \geq \epsilon\}.$$

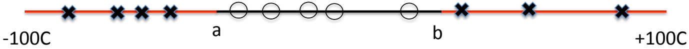
There are at most $|\mathcal{C}|$ such sets, and the probability that a sample is chosen inside a set is $\geq \epsilon$.
The probability that m random samples did not intersect with at least one of the sets is bounded by

$$|\mathcal{C}|(1-\epsilon)^m \leq |\mathcal{C}|e^{-\epsilon m} \leq |\mathcal{C}|e^{-(\ln|\mathcal{C}|+\ln\frac{1}{\delta})} \leq \delta.$$

□

How Good is this Bound?

- Assume that we want to estimate the working temperature range of an iPhone.
- We sample temperatures in [-100C,+100C] and check if the iPhone works in each of these temperatures.



Learning an Interval

- A distribution $\mathcal{D}$ is defined on universe that is an interval $[A, B]$.
- The true classification rule is defined by a sub-interval $[a, b] \subseteq [A, B]$.
- The concept class $\mathcal{C}$ is the collection of all intervals,

$$\mathcal{C} = \{[c, d] \mid [c, d] \subseteq [A, B]\}$$

Theorem

There is a learning algorithm that given a sample from $\mathcal{D}$ of size $m = \frac{2}{\epsilon} \ln \frac{2}{\delta}$, with probability $1 - \delta$, returns a classification rule (interval) $[x, y]$ that is correct with probability $1 - \epsilon$.

Note that the sample size is independent of the size of the concept class $|\mathcal{C}|$, which is infinite.

- The union bound is far too loose for our applications. It sums over overlapping hypothesis.
- Each sample excludes many possible intervals.
- Need better characterization of concept's complexity!

Probably Approximately Correct Learning (PAC Learning)

- The goal is to learn a concept (hypothesis) from a pre-defined concept class. (An interval, a rectangle, a k-CNF boolean formula, etc.)
- There is an unknown distribution D on input instances.
- Correctness of the algorithm is measured with respect to the distribution D.
- The goal: a polynomial time (and number of samples) algorithm that with probability $1 - \delta$ computes an hypothesis of the target concept that is correct (on each instance) with probability $1 - \epsilon$.

Formal Definition

- We have a unit cost function $Oracle(c, D)$ that produces a pair $(x, c(x))$, where x is distributed according to D, and $c(x)$ is the value of the concept c at x. Successive calls are independent.
- A concept class $\mathcal{C}$ over input set X is PAC learnable if there is an algorithm L with the following properties: For every concept $c \in \mathcal{C}$, every distribution D on X, and every $0 \leq \epsilon, \delta \leq 1/2$,
  - Given a function $Oracle(c, D)$, ϵ and δ, with probability $1 - \delta$ the algorithm output an hypothesis $h \in \mathcal{C}$ such that $Pr_D(h(x) \neq c(x)) \leq \epsilon$.
  - The concept class $\mathcal{C}$ is efficiently PAC learnable if the algorithm runs in time polynomial in the size of the problem,$1/\epsilon$ and $1/\delta$.

So far we showed that the concept class "intervals on the line" is efficiently PAC learnable.

Learning Boolean Conjunctions

- A Boolean literal is either x or $\bar{x}$.
- A conjunction is $x_i \wedge x_j \wedge \bar{x}_k \ldots$.
- $\mathcal{C} =$ is the set of conjunctions of up to $2n$ literals.
- The input space is $\{0,1\}^n$
- $c \in \mathcal{C}$ is the correct formula.

Theorem

The class of conjunctions of Boolean literals is efficiently PAC learnable.

Proof

- Start with the hypothesis $h = x_1 \wedge \bar{x_1} \wedge \ldots x_n \wedge \bar{x_n}$.
- Ignore negative examples generated by $Oracle(c, D)$.
- For a positive example $(a_1, \ldots, a_n)$, if $a_i = 1$ remove $\bar{x}_i$, otherwise remove x_i from h.

Lemma

At any step of the algorithm the current hypothesis never errs on negative example. It may err on positive examples by not removing enough literals from h*.*

Proof.

Initially the hypothesis has no satisfying assignment. It has a satisfying assignment only when no literal and its complement are left in the hypothesis. A literal is removed when it contradicts a positive example and thus cannot be in c. Literals of c are never removed. A negative example must contradict a literal in c, thus is not satisfied by h. □

Analysis

- The learned hypothesis h can only err by rejecting a positive examples. (it rejects an input unless it had a similar positive example in the training set.)
- If h errs on a positive example then in has a literal that is not in c.
- Let z be a literal in h and not c. Let

$$p(z) = Pr_{a \sim D}(c(a) = 1 \text{ and } z = 0 \text{ in } a).$$

- A literal z is"bad" If $p(z) > \frac{\epsilon}{2n}$.
- Let $m \geq \frac{2n}{\epsilon} \ln(2n) + \ln \frac{1}{\delta}$. The probability that after m samples there is any bad literal in the hypothesis is bounded by

$$2n(1 - \frac{\epsilon}{2n})^m \leq \delta.$$

Two fundamental questions:

- What concept classes are PAC-learnable with a given number of training (random) examples?
- What concept class are efficiently learnable (in polynomial time)?

A complete (and beautiful) characterization for the first question, not very satisfying answer for the second one.

Some Examples:

- Efficiently PAC learnable: Interval in R, rectangular in R^2, disjunction of up to n variables, 3-CNF formula,...
- PAC learnable, but not in polynomial time (unless $P = NP$): DNF formula, finite automata, ...
- Not PAC learnable: Convex body in R^2, $\{\sin(hx) \mid 0 \leq h \leq \pi\}$,...