Learning Concepts

AW

Lecture Overview

Can DTree classifier be learned?

- Recall, that for a class $\mathscr{H}$, a hypothesis $h \in \mathscr{H}$ and a training set S we define empirical risk as $L_S(h) \stackrel{\triangle}{=} \frac{|\{j \in [m] : h(x_i) \neq y_i\}|}{m}$ where $m = |S|$ and $[m] = \{1, \ldots, m\}$
- We agreed that a learner should choose h_S that classifies well on training set. In other words we should choose predictor h_S that belongs to a set of functions $ERM_{\mathscr{H}}(S) = \arg\min_{h \in H} L_S(h)$
- In binary classification with decision trees class $\mathscr{H}$ is a class of characteristic functions defined by decision trees over features that have finite domains (everything in computer is finite). Therefore $|\mathscr{H}| < \infty$.
- We opted for approximate algorithms since learning smallest optimal DTrees w.r.t. ERM is $\mathbf{NP}$-hard. But for a given S is there an algorithm (of any hardness) that learns ERM DTree exactly?

  We'll answer this question by answering more general one: given a training subset S of data D and predictor class $\mathscr{H}$ that contains true binary predictor for D, is there an algorithm that learns this optimal predictor?

Assumptions and their Implications

Recall that an error of a prediction rule h is $L_{(D,f)}(h) \stackrel{\triangle}{=} \mathbb{P}_{x\ D}(h(x) \neq f(x)) = D(\{x : h(x) \neq f(x)\})$ where f is a true classification function.

- The Realizability Assumption. There exists $h^* \in \mathscr{H}$ s.t. $L_{(D,f)}(h^*) = 0$. Note that this assumption implies that with probability 1 over random samples, S, where the instances of S are sampled according to D and are labeled by f, we have $L_S(h^*) = 0$.
- i.i.d Assumption. Any bound on the error with respect to the underlying distribution D for an algorithm that has access only to a sample S should depend on the relationship between D and S. We assume that the training sample S is generated by sampling points from the distribution D independently of each other

  Denote i.i.d assumption by $S \sim D^m$ where m is the size of S, and D^m denotes the probability over m-tuples induced by applying D to pick each element of the tuple independently of the other members of the tuple.
- $L_{(D,f)}(h_S)$ depends on the training set S which is picked by a random process, so there is randomness in the choice of the predictor h_S and, consequently, in the risk $L_{(D,f)}(h_S)$ which is then a random variable.

What are we estimating?

- Fix error ϵ admissible for a hypothesis in class $\mathscr{H}$. We interpret event $L_{(D,f)}(h_S) \geq \epsilon$ as failure of hypothesis h.
- The probability of $L_{(D,f)}(h_S) \geq \epsilon$ depends on how "good" S is: if it is picked by an adversary of h we may have big error even if h is good. But we agreed on $S \sim D^m$. So for a fixed probability δ that S is nonrepresentative we can examine probability of event $L_{(D,f)}(h_S) \geq \epsilon$ for chosen h
- $1 - \delta$ is called confidence parameter and ϵ is called accuracy (we use it already).
- We want to bound the probability $Pr_{S\sim D}(L_{(D,f)}(h_S) \geq \epsilon) = D^m(\{S : L_{(D,f)}(h_S) > \epsilon\})$ where $D^m(\{S : L_{D,f}(h_S) > \epsilon\})$ is a fraction of i.i.d drawn m-tuples (i.e. fraction of training sets of size m drawn w.r.t distribution D) on which a hypothesis h_S gives no errors, but it then gives more than ϵ-fraction of errors on all data.

Lecture Overview

1. What are we trying to learn when constructing classifier?
2. Learning Theory - PAC

The bound

- We need to estimate the probability that we picked a bad hypothesis h that gives lots of errors $L_{(D,f)}(h) \geq \epsilon$
- How could we pick it? By realizability we only consider predictors that give no errors on training sets $L_S(h_S) = 0$.
- So a training set S is bad for us if there is a bad hypothesis h that gives no errors on S. If we sampled S we can pick bad learner h

The bound

- We need to estimate the probability that we picked a bad hypothesis h that gives lots of errors $L_{(D,f)}(h) \geq \epsilon$
- How could we pick it? By realizability we only consider predictors that give no errors on training sets $L_S(h_S) = 0$.
- So a training set S is bad for us if there is a bad hypothesis h that gives no errors on S. If we sampled S we can pick bad learner h
- Consider a family M of all bad training sets. Note that if $\{S : L_{(D,f)}(h_S) > \epsilon\} \subseteq M$:
  - By realizability predictor h must yeild no error on training set, i.e. $L_S(h_S) = 0$, so $L_{(D,f)}(h_S) > \epsilon$ can only be true if for some bad h (that become h_s when we choose it to be a predictor), we have $L_S(h) = 0$.
  - If we take all bad hypothesis h and collect all sets on which they behave perfectly we get M, which means $M = \bigcup_{h \in \mathscr{H}_{bad}} \{S | L_S(h) = 0\}$. Here $\mathscr{H}_{bad}$ is a set of all bad hypothesis.

The bound

- We need to estimate the probability that we picked a bad hypothesis h that gives lots of errors $L_{(D,f)}(h) \geq \epsilon$
- How could we pick it? By realizability we only consider predictors that give no errors on training sets $L_S(h_S) = 0$.
- So a training set S is bad for us if there is a bad hypothesis h that gives no errors on S. If we sampled S we can pick bad learner h
- Consider a family M of all bad training sets $M = \bigcup_{h \in \mathscr{H}_{bad}} \{S \,|\, L_S(h) = 0\}$.

$$
\begin{aligned}
Pr_{S\sim D}(L_{(D,f)}(h_S) \geq \epsilon) &= D^m(\{S \,|\, L_{(D,f)}(h_S) \geq \epsilon\}) \leq D^m(M) \\
&= D^m(\bigcup_{h \in \mathscr{H}_{bad}} \{S \,|\, L_S(h) = 0\} \\
&\leq \sum_{h \in \mathscr{H}_{bad}} D^m(\{S \,|\, L_S(h) = 0\})
\end{aligned}
$$

Also

$$
\begin{aligned}
D^m(\{S \,|\, L_S(h) = 0\}) &= D^m(\{S \,|\, \forall i \in [m] \;\; h(x_i) = f(x_i)\}) \\
&\overset{\text{i.i.d.}}{=} \prod_{i=1}^{m} D(\{x_i \,|\, h(x_i) = f(x_i)\})
\end{aligned}
$$

The bound

- We need to estimate the probability that we picked a bad hypothesis h that gives lots of errors $L_{(D,f)}(h) \geq \epsilon$
- How could we pick it? By realizability we only consider predictors that give no errors on training sets $L_S(h_S) = 0$.
- So a training set S is bad for us if there is a bad hypothesis h that gives no errors on S. If we sampled S we can pick bad learner h

  So

$$\begin{aligned} Pr_{S\sim D}(L_{(D,f)}(h_S) \geq \epsilon) &= D^m(\{S \mid L_{(D,f)}(h_S) > \epsilon\}) \\ &\leq \textstyle\sum_{h\in\mathscr{H}_{bad}} D^m(\{S \mid L_S(h) = 0\}) \\ &\leq |\mathscr{H}_{bad}|\, D^m(\{S \mid L_S(h) = 0\}) \\ &\leq |\mathscr{H}| D^m(\{S \mid L_S(h) = 0\}) \end{aligned}$$

  and

$$\begin{aligned} D^m(\{S \mid L_S(h) = 0\}) &= \textstyle\prod_{i=1}^m D(\{x_i \mid h(x_i) = f(x_i)\}) \\ &= \textstyle\prod_{i=1}^m (1 - L_{(D,f)}(h)) \\ &\leq \textstyle\prod_{i=1}^m (1-\epsilon) \;=\; (1-\epsilon)^m \;<\; e^{-\epsilon m} \end{aligned}$$

  because $L_{(D,f)}(h)) \geq \epsilon$, so $1 - L_{(D,f)}(h) \leq 1 - \epsilon$, and $1 - x < e^{-x}$

The bound

- We need to estimate the probability that we picked a bad hypothesis h that gives lots of errors $L_{(D,f)}(h) \geq \epsilon$
- How could we pick it? By realizability we only consider predictors that give no errors on training sets $L_S(h_S) = 0$.
- So a training set S is bad for us if there is a bad hypothesis h that gives no errors on S. If we sampled S we can pick bad learner h

So

$$\begin{aligned} Pr_{S\sim D}(L_{(D,f)}(h_S) \geq \epsilon) &= D^m(\{S \,|\, L_{(D,f)}(h_S) > \epsilon\}) \\ &\leq \textstyle\sum_{h\in\mathscr{H}_{bad}} D^m(\{S \,|\, L_S(h) = 0\}) \\ &\leq |\mathscr{H}_{bad}|\, D^m(\{S \,|\, L_S(h) = 0\}) \\ &\leq |\mathscr{H}| D^m(\{S \,|\, L_S(h) = 0\}) \\ &< |\mathscr{H}| e^{-\epsilon m} \end{aligned}$$

The last line because of

$$D^m(\{S \,|\, L_S(h) = 0\}) = \textstyle\prod_{i=1}^m D(\{x_i \,|\, h(x_i) = f(x_i)\}) < e^{-\epsilon m}$$

Meaning of the Bound

- We have shown that for each bad hypothesis H at most $(1-\epsilon)^m < e^{-\epsilon m}$ fraction of training sets is misleading, so probability to choose bad hypothesis is at most $\Pr(L_{(D,f)}(h_S)) > \epsilon) < |\mathscr{H}|e^{-\epsilon m}$
- We want to choose m so that the probability that S is non-representative for D is at most δ. So for a given ϵ, we'd like to find such m that $\Pr(L_{(D,f)}(h_S)) > \epsilon) < |\mathscr{H}|e^{-\epsilon m} \leq \delta$, then we have $\Pr(L_{(D,f)}(h_S)) < \epsilon) \geq 1-\delta$.
- But then for any given δ and ϵ if we choose $m > \frac{ln(|\mathscr{H}|/\delta)}{\epsilon}$, then $\Pr(L_{(D,f)}(h_S)) < \epsilon) \geq 1-\delta$ as we desired!

The meaning of this bound: For any labeling function f and probability distribution D over data if we choose sufficiently large i.i.d. sample S (of size $m > \frac{ln(|\mathscr{H}|/\delta)}{\epsilon}$), and follow the $ERM_{\mathscr{H}}$ rule in choosing predictor h_S from a finite realizable hypothesis class $\mathscr{H}$, then our predictor choice is probably (with confidence $1-\delta$) approximately (up to an error of ϵ) correct.

Meaning of the Bound

The meaning of this bound: For any labeling function f and probability distribution D over data if we choose sufficiently large i.i.d. sample S (of size $m > \frac{ln(|\mathscr{H}|/\delta)}{\epsilon}$), and follow the $ERM_{\mathscr{H}}$ rule in choosing predictor h_S from a finite realizable hypothesis class $\mathscr{H}$, then our predictor choice is probably (with confidence $1-\delta$) approximately (up to an error of ϵ) correct.
So why are we using approximate algorithms? Instead compute m from input (δ, ϵ), design an efficient algorithm for DTree learning from size m sample using ERM, sample data, and learn the tree. What could be problematic in this plan?

- Designing an efficient algorithm for DTree learning from size m sample using ERM? Recursive partitioning with ERM can be modified to take m as parameter so that resulting algorithm is optimal (from greedy to dynamic programming)
- finding m? indeed we need to estimate size $|\mathscr{H}|$ and solve the respective equation for m. The class of DTrees is 'almost infinite' - wont work! But we could limit the class to Dtrees of degree b and depth d. We'll come back to it later

PAC learning

When we showed that a finite class $\mathscr{H}$ is learnable with confidence δ and accuracy ϵ, we introduced new paradigm of learning:
PAC Learnability: A (binary) hypothesis class $\mathscr{H}$ defined on domain X is Probably Approximately Correctly (**PAC**) learnable if there exist a function $m_H : (0,1)^2 \to \mathscr{N}$ and a learning algorithm $\mathscr{A}$ with the following property. For every

- $\epsilon,\ \delta \in (0,1)$,
- distribution D over data X,
- labeling function $f : X \to \{0,1\}$ that belongs to $\mathscr{H}$

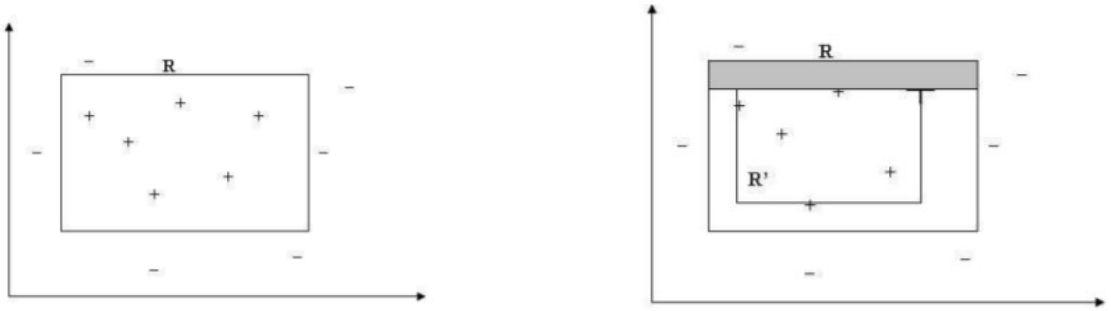
running the learning algorithm $\mathscr{A}$ on $m \geq m_H(\epsilon, \delta)$ i.i.d. examples generated by D and labeled by f, result in $\mathscr{A}$ returning a hypothesis $h : X \to \{0,1\}$ that with probability of at least $1 - \delta$ (over the choice h from $\mathscr{H}$ determined by D via ERM) has error $L_{(D,f)}(h) \leq \epsilon$.

Note that both realizability assumption and i.i.d. assumption are embedded into this definition

The class is said to be efficiently learnable if $X < \infty$, $m_H(\epsilon, \delta) = poly(\frac{1}{\epsilon}, \frac{1}{\delta}, |X|)$ and $\mathscr{A}$ runs in time bounded by time polynomial in $|X|$.
Does any class like that exists? Why do we expect these assumptions to hold?

Learnable rectangles

Does any class like that exists? Let $X = \{x, y, z | \underline{x} \leq x \leq \overline{x}, \underline{y} \leq y \leq \overline{y}, z \in \{-, +\}\}$, so $X = R^+ \cup R^-$ with rectagle R^+ and R^- its complement.



Algorithm to construct hypothesis for R^+ (see on the left fig.) - decision tree that fits the tightest rectangle.

- Error probability=area of the frame (on right fig), must be $< \epsilon$;
- The error event - no example gets into frame.
- The area of one stripe $= \epsilon/4$. The probability of one example not to get into it is $(1 - \epsilon/4)$. None of m i.i.d. samples getting-into-stripe probability $(1 - \epsilon/4)^m$. Frame of 4 stripes - probability no examples in frame $\leq 4(1 - \epsilon/4)^m$ (union bound).
- Require $4(1 - \epsilon/4)^m < \delta$. Since $1 - x < e^{-x}$ require $4(1 - \epsilon/4)^m < 4e^{-m\epsilon/4} < \delta$, so when $m > (4/\epsilon) \ln 4/\delta$ tree learns it.

Reading

In Ben-David, Shalev Schwartz. Understanding Machine Learning, supplemental book (online-see syllabus section):
Chapter 2, 3 and 4.1