Lecture 8

Announcements:

* Presentation schedule posted. Start early!

Recap: Rademacher complexity

Theorem (Excess risk bounds using Rademacher)

Assume that for all z and $h \in \mathcal{H}$ we have that $|\ell(h,z)| \leq c$. Then we probability at least $(1-\delta)$ over $S \sim D^n$,

(1) $$\sup_{h \in \mathcal{H}} \left(R(h) - \hat{R}_S(h) \right) \leq 2\, \mathbb{E}_{S'} R(\ell \circ \mathcal{H} \circ S') + c\sqrt{\frac{2\log(1/\delta)}{n}}.$$

(2) $$\sup_{h \in \mathcal{H}} \left(R(h) - \hat{R}_S(h) \right) \leq 2\, R(\ell \circ \mathcal{H} \circ S) + 3c\sqrt{\frac{2\log(2/\delta)}{n}}$$

(3) for $h^* = \operatorname{argmin}_{h \in \mathcal{H}} R(h)$

$$R(ERM_{\mathcal{H}}(S)) - R(h^*) \leq 2\, R(\ell \circ \mathcal{H} \circ S) + 4c\sqrt{\frac{2\log(4/\delta)}{n}}.$$

Lemma (Contraction Lemma)

For each $i \in [n]$, let $\phi_i : \mathbb{R} \to \mathbb{R}$ be a ρ-Lipschitz function i.e. $|\phi_i(x) - \phi_i(y)| \leq \rho |x-y| \quad \forall x, y \in \mathbb{R}$.

For any $a \in \mathbb{R}^n$ define $\phi(a) \in \mathbb{R}^n$ as $\phi(a) = (\phi_1((a)_1), \ldots, \phi_n((a)_n))$. For a set A, let $\phi \circ A = \{\phi(a) : a \in A\}$. Then

$$\mathcal{R}(\phi \circ A) \leq \rho \, \mathcal{R}(A).$$

Rademacher complexity of linear classes.

$$\mathcal{H}_1 = \{ h_w(x) = \langle w, x \rangle : \|w\|_1 \leq B_1 \}$$

$$\mathcal{H}_2 = \{ h_w(x) = \langle w, x \rangle : \|w\|_2 \leq B_2 \}$$

Lemma (ℓ_2 bounded linear predictor)

Let $S = (x_1, \ldots, x_n)$. Define $\mathcal{H}_2 \circ S = \{(\langle w, x_1 \rangle, \ldots \langle w, x_n \rangle) : \|w\|_2 \leq B_2\}$.

Then $\mathcal{R}(\mathcal{H}_2 \circ S) \leq B_2 \frac{\max_i \|x_i\|_2}{\sqrt{n}}$

Lemma (ℓ_1 bounded linear model)

Let $S = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}^d \ \forall\ i \in [n]$.

Then $$\mathcal{R}(\mathcal{H}_1 \circ S) \leq B_1 \max_i \|x_i\|_\infty \sqrt{\frac{2\log(2d)}{n}}.$$

Takeaways

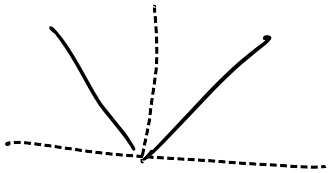
1. If the loss function $\ell(h, z)$ is 1-Lipschitz then by the contraction lemma,

$$\mathcal{R}(\ell \circ \mathcal{H}_2 \circ S) \leq \mathcal{R}(\mathcal{H}_2 \circ S).$$

Now using excess risk bound, we directly get generalization bound.

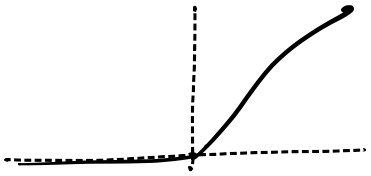
Examples of 1-Lipschitz losses

$\ell(h, z) = |h(x) - y|$



$\ell(h, z) = \max\{0, 1 - h(x)y\}$

(hinge loss)



2. Note that the VC-dimension bound for linear predictors in $\mathbb{R}^d$ is $O(d)$. The Rademacher bound does not directly depend polynomially on dimension, it depends on $B_2 \max_i \|x_i\|_2$ or $B_1 \max_i \|x_i\|_\infty$. Hence it could be much smaller than VC dimension.

3. This motivates the idea of <u>regularization</u>

* $\mathcal{H}_2$ bound depends on $B_2 \max_i \|x_i\|_2$.

∴ If we choose to learn over small l_2 norm ball (i.e. small B_2), we have better generalization. Now if best w^* also has $\|w^*\|_2 < B_2$, we also get small approximation error. We want to choose best B_2 (an example of regularization).

* $\mathcal{H}_1$ bound depends on $B_1 \max_i \|x_i\|_\infty$

Say $x_i \in \{\pm 1\}^d$ ∴ $\max_i \|x_i\|_\infty = 1$,

$\max_i \|x_i\|_2 = \sqrt{d}$

Say w is in $\{-1, 0, +1\}^d$ & w is also k-sparse.

$\|w\|_2 = \sqrt{k}$ $\|w\|_1 = k$.

$B_2 \cdot \max_i \|x_i\|_2 = \sqrt{R} \cdot \sqrt{d}$

$B_1 \quad \max_i \|x_i\|_\infty = R \cdot 1$

Now, if $R << \sqrt{R} \cdot \sqrt{d}$ (or $R << d$) working over ℓ_1 ball could be much better than working over ℓ_2 ball.

This concludes our study of the statistical complexity of learning.

Computational complexity of learning

RECAP

Definition (PAC Learnability):
A hypothesis class $\mathcal{H}$ is PAC-learnable if $\exists$ a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$, every distribution D over X and every $h \in \mathcal{H}$, when the algorithm is given $n_{\mathcal{H}}(\epsilon, \delta)$ samples drawn from D & labelled by h, the alg. produces a hypothesis $\hat{h}$ s.t. with probability $1-\delta$, $P(\hat{h}) \leq \epsilon$.
(The probability is over randomness in training set, and any internal algorithmic randomness.

What about the running time of the algorithm?
What can we learn "in polynomial time"?
First, what does it mean to learn "in polynomial time"?

→ Polynomial in size of training set?
Not such a good metric, what if we give the algorithm much more data than it needs to learn, it then just uses a small subset of the data it actually needs, but it runs in time polynomial (size of original data)?

Account for cost of training data.

Define (Example Oracle)

For any distribution D over X & hypothesis $h(\cdot): x \to y$, we define $Ex(h, D)$ as an example oracle which draws $x \sim D$, labels $y = h(x)$, outputs $(x, h(x))$

Sample complexity : # calls to oracle $Ex(h, D)$.

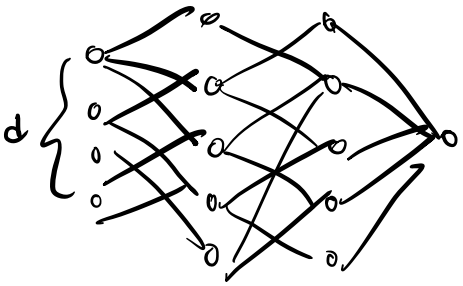
Right notion: Polynomial in instance size.

Typically we consider $X = \{0,1\}^d$ or $X = \mathbb{R}^d$, d is the instance size.

Aside: We also want polynomial in representation size of hypothesis class.

What is representation size of hypothesis class?

$\cong$ # bits required to write down any hypothesis in hypothesis class. For e.g. a neural network



Representation size = (# edges) · (# bits required to store each weight)

For all hypothesis classes we consider,

(representation size) = poly (instance size of the data points)

∴ For instances we consider,

poly (representation size, instance size) = poly (instance size).

Therefore we will only consider polynomial in instance size.

Definition (Efficient PAC learning)

A hypothesis class $\mathcal{H}$ is PAC-learnable if $\exists$ a learning algorithm A with the following property: for every $\epsilon, \delta \in (0,1)$, every distribution D over $\{X\}^d$ (where X is typically $\{0,1\}$ or $\mathbb{R}$), and every $h^* \in \mathcal{H}$, if A is given access to example oracle $Ex(h^*, D)$ and ϵ, δ, with probability $1-\delta$, it outputs a hypothesis $h \in \mathcal{H}$ with $R(h) \leq \epsilon$.

$\mathcal{H}$ is <u>efficiently</u> PAC learnable if running time of A is polynomial in $d, \frac{1}{\epsilon}, \frac{1}{\delta}$.

Example: learning Boolean functions.

$X^d = \{0,1\}^d$

<u>CONJUNCTIONS</u>: The class of all conjunctions on d Boolean literals $(x_1, x_2, \ldots, x_d)$

e.g. $x_1 \wedge \bar{x}_3 \wedge x_4$

represents the hypothesis class which is 1 if and only if $x_1 = 1$ & $x_3 = 0$ & $x_4 = 1$.

Example :

$X^d = \{0,1\}^d$

$d = 5$

For every datapoint $a_1 = (0, 0, 1, 1, 1)$

& its label is given by some ground-truth conjunction on coordinates.

If $h^* = x_1 \wedge \bar{x}_3 \wedge x_4$

label of a_1 is 0

$a_2 = (1, 1, 0, 1, 1)$

label of a_2 is 1.

Q) Can we design a polynomial $(d, \frac{1}{\varepsilon}, \frac{1}{\delta})$ time algorithm for learning conjunctions?

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

Proof

ALGORITHM:

1. Set $h = x_1 \wedge \bar{x}_1 \wedge x_2 \wedge \bar{x}_2 \ldots \wedge x_d \wedge \bar{x}_d$
2. For $i = 1, \ldots, n$ {
3. $(a_i, y_i) \leftarrow EX(h^*, D)$
4. if $(y_i = 1)$ {
5. Drop each $\bar{x}_j$ from h if $(a_i)_j = 1$
6. Drop each x_j from h if $(a_i)_j = 0$
7. }
8. }
9. Return h

Claim: The above algorithm is an ERM over the class of conjunctions.

Proof We show that the algorithm gets 0 misclassification error over training examples $\{(a_1, y_1), \ldots, (a_n, y_n)\}$

Claim The set of literals appearing in h at any time, contains the set of literals in target hypothesis h^*.

Proof In the beginning, h contains all possible conjunctions, we remove a literal from h if it was set to 0 in an example. Such a literal cannot appear in h^*. ∎

This implies: $h(a) = 1 \Rightarrow h^*(a) = 1 \quad \forall\, a \in \{0,1\}^d$.

∴ h correctly classifies all training data labelled as 0.

Moreover, after getting any new data point a_i with $y_i = 1$, h updates to predict 1 on the datapoint (and will never be updated to predict 0 on a_i, as literals are only removed). ∎

Now we use ERM result for finite hypothesis classes.

$$|\mathcal{H}| \leq 2^{2d}$$

∴ result for learnability of finite hypothesis classes implies that we can learn with error ϵ with failure probability δ, with

$$O\left(\frac{\log(|H|/\delta)}{\varepsilon}\right) = O\left(\frac{d\log(1/\delta)}{\varepsilon}\right) \text{ samples.}$$

■

Intractability of learning 3-Term DNF

what is a 3-term DNF (Disjunctive Normal Form) ?

3-Term-DNF$_d$ = $\{ T_1 \vee T_2 \vee T_3 \mid T_i$ is a conjunction on $\{x_1, \ldots, x_d\}\}$

Theorem: 3-term DNF formulae are <u>not</u> efficiently PAC learnable unless RP=NP.

Complexity review

* A decision problem C is in NP if there exists a poly-time algorithm A s.t.

  for every instance x of C,

  → if x evaluates to "yes", then $\exists\, y$, $|y| \leq \text{poly}(|x|)$, $A(x, y) = 1$.

  → if x evaluates to "no", then $\forall\, y$, $|y| \leq \text{poly}(|x|)$, $A(x, y) = 0$.

Intuition

A: verifier y : certificate / witness

Consider 3SAT

$$3SAT_d = (x_1 \vee \bar{x_2} \vee x_5) \wedge (x_5 \vee x_6 \vee \bar{x_4}) \wedge \vdots \wedge (x_{d-2} \vee \bar{x_{d-1}} \vee x_d)$$

Certificate y: satisfying assignment

Easy to check if an assignment is valid.

There always exists a certificate if instance is satisfiable, no certificate if not.

* A decision problem C is in RP, if there exists a randomized poly-time algorithm s.t. for every instance x of C,
  - → if x evaluates to "yes", A outputs "yes" w.p. $\geq 2/3$.
  - → if x evaluates to "no", A outputs "no" w.p. 1.

Intuition: C is in RP if $\exists$ poly-time algo. with one-sided error.

3SAT instance is satisfiable, output TRUE w.p. $\geq 2/3$.

" not satisfiable, always output FALSE.

Widely believed that RP ≠ NP.
(no randomized poly-time algorithm for 3SAT).

* A decision problem C is NP-complete
  → if C is in NP
  → every decision problem C' in NP can be reduced to C in polynomial time.