CSCI699: Theory of Machine Learning Fall 2021

Problem Set 2

Due: October 20 by 11:59 pm

Discussion is allowed and encouraged but everyone should write solutions on their own. Please also mention any collaborators you had substantial discussions with. You are also allowed to consult general resources on the internet (such as for example whiteboard notes from our own lectures or one of the books), but you should not search for any the solutions themselves online. Homeworks should be written in Latex and submitted via Gradescope.

If you use the LaTeX template, then please only keep your answers and remove the questions before submitting.

When you submit on Gradescope, make sure to mark the page which contains each answer.

Problem 1: Rademacher Complexity Bounds for Neural Networks

In this problem, we will derive bounds on the Rademacher Complexity for some simple neural networks.

(a) (5pts) First, consider the following class of 'neural networks' with no hidden layers and a ReLU activation:

$$\mathcal{C}_0 = \{x \mapsto \max\{0, w^T x\} : w \in \mathbb{R}^d, \|w\|_2 \leq B_2\}.$$

Consider a set of unlabelled datapoints $S = (x_1, \ldots, x_n)$, where $x_i \in \mathbb{R}^d$, $\|x_i\|_2 \leq C$. Bound the Rademacher Complexity $\mathcal{R}(\mathcal{C}_0 \circ S)$.

(b) (8pts) Now consider the following class of neural networks with one hidden layer with m hidden units,

$$\mathcal{C}_1 = \{x \mapsto \sum_{j=1}^{m} \alpha_j \max\{0, w_j^T x\} : \sum_{j=1}^{m} |\alpha_j| \leq B_1 \;\&\; \forall\, j \in [m], w_j \in \mathbb{R}^d, \|w_j\|_2 \leq B_2\}.$$

Consider a set of unlabelled datapoints $S = (x_1, \ldots, x_n)$, where $x_i \in \mathbb{R}^d$, $\|x_i\|_2 \leq C$. Bound the Rademacher Complexity $\mathcal{R}(\mathcal{C}_1 \circ S)$.

To do this, you will likely find it useful to bound the Rademacher complexity of an absolute value of a function composition. Define $\mathcal{R}'(A)$ as the Rademacher complexity of a set A, but with an absolute value:

$$\mathcal{R}'(A) = \frac{1}{n}\mathbb{E}_{\sigma \sim \{\pm\}^n}\left[\sup_{a \in A}\left|\sum_{i=1}^{n} \sigma_i (a)_i\right|\right],$$

where $(a)_i$ is the ith coordinate of the vector a. This definition is identical to the one we used in class, except for the additional absolute value. In fact, many papers consider this alternative definition of Rademacher complexity [1]. It can be shown $\mathcal{R}'(A)$ satisfies a function composition property very similar to the contraction lemma we stated in class for the original definition of Rademacher complexity.

Lemma 1. *[1] Let $\phi : \mathbb{R} \to \mathbb{R}$ be a ρ-Lipschitz function which satisfies $\phi(0) = 0$. For any $a \in \mathbb{R}^n$, define $\phi(a) \in \mathbb{R}^n$ as the function ϕ applied to very coordinate of a, i.e. $\phi(a) = (\phi((a)_1), \ldots, \phi((a)_n))$. Let $\phi \circ A = \{\phi(a), a \in A\}$. Then,*

$$\mathcal{R}'(\phi \circ A) \leq 2\rho\mathcal{R}'(A).$$

Try to use the above result to simplify your calculations. In the end, you should get a bound which does not explicitly depend on m or d. Therefore, in contrast to the VC dimension bound we got in the last homework, the Rademacher complexity bound only depends on some appropriate norms of the parameters of the neural network, not the number of parameters itself.

Problem 2: PAC Learning with 2-sided Oracles

(15pts) As we mentioned in class, one of the advantages of defining the example oracle $\mathrm{EX}(c, D)$ is that we can now just think of access to a randomly drawn, labelled example as a resource/oracle that the learner has. The example oracle is just one possible kind of oracle access the learner could have, and this question will explore a different **two-oracle** model. For a target concept $c \in \mathcal{C}$, define two separate distributions, D_c^+ over the positive examples of c, and D_c^- over the negative examples of c. In other words, D_c^+ is the distribution of x conditioned on $c(x) = 1$, and similarly for D_c^- (where as usual, we say that label 1 is a positive label and label 0 is a negative label). The learning algorithm now has access to two oracles $\mathrm{EX}(c, D_c^+)$ and $\mathrm{EX}(c, D_c^-)$ that return a random positive or negative example in unit time. For error parameter ϵ, the learning algorithm must find a hypothesis $h \in \mathcal{H}$ satisfying $\Pr_{x \in D_c^+}[h(x) = 0] \leq \epsilon$ and $\Pr_{x \in D_c^-}[h(x) = 1] \leq \epsilon$. Thus, the learning algorithm may now explicitly request either a positive or negative example, but must find a single hypothesis with small error on both distributions.

Let $\mathcal{C}$ be any concept class and $\mathcal{H}$ be any hypothesis class. Let h_0 and h_1 be representations of the identically 0 and identically 1 functions, respectively (i.e. $h_0(x) = 0 \; \forall \; x \in \mathcal{X}$, analogously for h_1). Prove that:

(a) If $\mathcal{C}$ is efficiently PAC learnable using $\mathcal{H}$ in the original one-oracle model, then $\mathcal{C}$ is efficiently PAC learnable using $\mathcal{H}$ in the two-oracle model.

(b) If $\mathcal{C}$ is efficiently PAC learnable using $\mathcal{H}$ in the two-oracle model, then $\mathcal{C}$ is efficiently PAC learnable using $\mathcal{H} \cup \{h_0, h_1\}$ in the one-oracle model.

Problem 3: Learning Halfspaces

In this problem, we will establish some learnability and hardness results for halfspaces, and functions of halfspaces.

(a) (7pts) Consider the concept class of halfspaces

$$\mathcal{C} = \{x \mapsto \mathbf{1}(\theta^T x + b > 0) : \theta \in \mathbb{R}^d, b \in \mathbb{R}\},$$

here $\mathbf{1}(\cdot)$ denotes the indicator function which takes the value 1 if the input is true, and 0 otherwise. Show that $\mathcal{C}$ is efficiently PAC learnable . You might find it useful to use a routine for solving *linear programs* as part of your learning algorithm. A linear program (LP) solver takes as input $u \in \mathbb{R}^d, A \in \mathbb{R}^{n \times d}$ and $v \in \mathbb{R}^m$, and outputs $w \in \mathbb{R}^d$ which is the solution to:

$$\begin{aligned} \max_{w \in \mathbb{R}^d} \quad & w^T u \\ \text{such that} \quad & Aw \geq v. \end{aligned}$$

It is known that there LP solvers which run in time polynomial in n, d [2].

(b) (15pts) Now we consider a slightly more general hypothesis class, which is an intersection (conjunction) of *two* halfpspaces. More formally, define

$$\mathcal{C}_2 = \{x \mapsto \mathbf{1}(\theta_1^T x + b_1 > 0) \cdot \mathbf{1}(\theta_2^T x + b_2 > 0) : \theta_1, \theta_2 \in \mathbb{R}^d, b_1, b_2 \in \mathbb{R}\}.$$

Show that it is not possible to properly PAC learn $\mathcal{C}_2$ in polynomial time, unless NP=RP .

To do this, you will find it helpful to consider the *hypergraph 2-coloring* problem.

Definition 2. *A hypergraph G is a set V of d vertices (denoted as $1, 2, \ldots, d$) and m subsets $s_1, \ldots, s_m \subseteq V$, these subsets are also known as hyperedges. The hypergraph 2-coloring problem is the problem of determining if there is a valid 2-coloring of the vertices (i.e. an assignment of a color, Red or Blue, to every vertex of the graph) such that no subset s_j is monochromatic (i.e. every s_j has at least one Red vertex and one Blue vertex). It is known that hypergraph 2-coloring is NP-Complete.*

You can use the following construction to create a set of labelled examples based on an input hypergraph G.

(a) First, create a datapoint $(\mathbf{0}, +1)$, where the input $\mathbf{0}$ is the origin, and the label is $+1$.

(b) Next, for every vertex i, create a datapoint $(e_i, -1)$, where the input e_i is the i-th basis vector (having a 1 in the i-th coordinate and 0 everywhere else), and the label is -1.

(c) Finally, for every hyperedge s_j, create a datapoint $(u_j, +1)$ where u_j is the indicator vector for the subset s_j (having a 1 in coordinate i if $i \in s_j$, and 0 otherwise).

Hint: A figure always helps, try to draw even a simple three vertex graph example. If you like, you can look at the hints given for Exercise 3.2 in Chapter 8 of the Understanding Machine Learning book, which shows a similar hardness result.

References

[1] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[2] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.