COMS 4771 SP21 HW4

Due: Fri Apr 09, 2021 at 11:59pm

This homework is to be done **alone**. No late homeworks are allowed. To receive credit, a typesetted copy of the homework pdf must be uploaded to Gradescope by the due date. You must show your work to receive full credit. Discussing possible solutions for homework questions is encouraged on piazza and with your peers, but you must write your own individual solutions and **not** share your written work/code. You must cite all resources (including online material, books, articles, help taken from specific individuals, etc.) you used to complete your work.

1 A question on active learning

Suppose we are trying to learn the correct "threshold" for data in $[0,1]$. Formally, let $\mathcal{D}$ be a distribution over $[0,1] \times \{0,1\}$, for $\alpha \in [0,1]$ define $f_\alpha : [0,1] \to \{0,1\}$ by $f_\alpha(x) = \mathbb{1}[x \geq \alpha]$, and let $\mathcal{F} = \{f_\alpha \mid \alpha \in [0,1]\}$ be the function class we are interested in learning. Furthermore, assume that there is a "true" target concept which always labels the data correctly (i.e. assume $\exists f_\alpha \in \mathcal{F} : \text{err}(f_\alpha) = 0$).

(a) Give a learning algorithm which given a sample $S = \{(x_i, y_i)\}_{i=1}^m$ drawn iid from $\mathcal{D}$ runs in time $O(m)$ and outputs a classifier $f_\alpha \in \mathcal{F}$ with the following property: for any $\epsilon, \delta > 0$ if $m \geq O\left(\frac{1}{\epsilon^2} \ln(1/\delta)\right)$ then with probability at least $1 - \delta$ over the draw of S, $\text{err}(f_\alpha) \leq \epsilon$. Make sure you prove that it has this property.

It is interesting to note that due to the assumption that there exists a true classifier in $\mathcal{F}$ which perfectly labels the points, it can actually be shown that only $O\left(\frac{1}{\epsilon} \ln(1/\delta)\right)$ points are required to PAC learn $\mathcal{F}$ (you do not need to prove this). However, this is essentially tight (it can be shown that there exist distributions on which that many points are needed). This is disappointing since the problem is simple enough that one could hope for a logarithmic dependence on $1/\epsilon$ only (i.e. exponentially better than what we have right now). To achieve something like this we can look towards **active learning**.

Active learning is a slightly different learning framework in which the learner is allowed to have some amount of say into which examples it gets (as opposed to the PAC learning framework seen in class where the learner is simply given a set of examples). Active learning is interesting in part because it better reflects the process of learning that we are familiar with. Indeed, in real life we as learners can ask questions and demand explanations. We are never expected to learn by simply being handed a set of examples. Furthermore, in many settings active learning can yield exponential improvements over passive PAC learning. Indeed:

(b) Suppose that $\mathcal{D}$ is uniform over $[0,1]$ and that the learner, rather than being given a set of points drawn iid from $\mathcal{D}$, is able to query for the true label of any point $x \in [0,1]$ (and gets the answer in $O(1)$ time). Give a learning algorithm which given $\epsilon > 0$ makes at most

$O(\ln(1/\epsilon))$ queries, runs in time $O(\ln(1/\epsilon))$, and returns some $f_\alpha \in \mathcal{F}$ such that $\text{err}(f) \leq \epsilon$ with probability 1. Make sure you prove that it has these properties.

Unfortunately there are two issues with an "active learning" solution of this type. First and foremost, $\mathcal{D}$ is usually unknown (above we assume it is uniform over $[0, 1]$) and it is not clear how to extend the above solution to work in such cases. Second, it is not always reasonable to assume that the algorithm can just ask for the label of ANY point in the input space. For example, in image classification most of the input space looks like nonsensical images to human beings. A nice way to deal with both issues is to assume that the algorithm gets points drawn iid from $\mathcal{D}$ as before but that they are **unlabelled**. The algorithm can then actively and sequentially query for the labels of these points, and do so only if the label is useful (i.e. if it brings more information). Formally, consider the following relatively general "active learning algorithm":

Algorithm 1 Active learning algorithm

```
Input: A set of samples $S = \{x_i\}_{i=1}^m$ drawn iid from $\mathcal{D}$, the function class $\mathcal{F}$, and access to a query
  oracle $\mathcal{O}$ which on input $x_i \in S$ outputs $y_i$. It must be true that $\exists f \in \mathcal{F}$ such that err$(f) = 0$.
  Let $V = \mathcal{F}$
  for $x_i \in S$ do
    if $\exists f_1, f_2 \in V$ such that $f_1(x_i) \neq f_2(x_i)$ then
      Get $y_i$ by making the query $\mathcal{O}(x_i)$
      Set $V$ to be $\{f \in V : f(x_i) = y_i\}$
    end if
  end for
  return Any $f \in V$.
```

The issue with this algorithm is that it is often difficult to check the condition of the `if` statement and update V efficiently (i.e. without using too much running time). Also it is not clear in which order one should iterate through S. But for some simple problems there are solutions to these issues, in which case this active learning algorithm can give a much better label complexity (number of labeled points it needs) than the standard PAC alternatives.

(c) Going back to our problem from (a) and (b) rewrite Algorithm 1 so that:

- Your algorithm runs in time $O(m \log(m))$ (you can assume each call to $\mathcal{O}$ takes $O(1)$ time).
- Your algorithm makes $O(\log m)$ queries (calls to $\mathcal{O}$).
- If $m \geq O\left(\frac{1}{\epsilon^2}\ln(1/\delta)\right)$ then with probability at least $1 - \delta$ over the draw of S your algorithm returns f_α such that err$(f_\alpha) \leq \epsilon$.

As always make sure to prove that your algorithm has each of these properties.

Note that the label complexity of your new algorithm (i.e. how many examples it needs to have labelled) is exponentially smaller than that of the algorithm from part (a).

2 BER / application of large deviation theory

Suppose we have a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$, a sample $S = \{(x_i, y_i)\}_{i=1}^m$ drawn iid from $\mathcal{D}$, and a classifier $f : \mathcal{X} \to \{0, 1\}$. We have usually looked at the 0-1 error of f which is given by $\mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y]$ and which can be estimated via $\frac{1}{m}\sum_{i=1}^m \mathbb{1}[f(x_i) \neq y_i]$.

Sometimes, it is more reasonable to look at the **Balanced Error Rate** (BER) of f which is defined as:

$$\text{BER}(f) = \frac{1}{2}\left(\mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y \mid y = 1] + \mathbb{P}_{(x,y)\sim\mathcal{D}}[f(x) \neq y \mid y = 0]\right)$$

This is useful when one needs a classifier that has good accuracy on both classes (points where $y = 1$ and points where $y = 0$) despite having a distribution with very different priors (i.e. when one class is much more prevalent than the other).

Propose an estimator $\overline{\text{BER}}(f, S)$ of $\text{BER}(f)$ with the following property: for any $\varepsilon \in (0, 1)$, if $m \geq \frac{1000}{\varepsilon^2 \min(\mathbb{P}[y=1], \mathbb{P}[y=0])}$ then with probability at least 0.99 (over the draw of S),

$$\left|\overline{\text{BER}}(f, S) - \text{BER}(f)\right| \leq \varepsilon$$

Then show that your estimator has this property.

Hint: Use the Chernoff-Hoeffding bound seen in class.

3 Studying k-means

Recall that in k-means clustering we attempt to find k cluster centers $c_j \in \mathbb{R}^d$, $j \in \{1, \ldots, k\}$ such that the total (squared) distance between each datapoint and the nearest cluster center is minimized. In other words, we attempt to find $c_1, \ldots, c_k$ that minimizes

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|x_i - c_j\|^2, \tag{1}$$

where n is the total number of datapoints. To do so, we iterate between assigning x_i to the nearest cluster center and updating each cluster center c_j to the average of all points assigned to the jth cluster (aka Lloyd's method).

(a) **[it is unclear how to find the best k, i.e. estimate the correct number of clusters!]** Instead of holding the number of clusters k fixed, one can think of minimizing (1) over both k and c. Show that this is a bad idea. Specifically, what is the minimum possible value of (1)? what values of k and c result in this value?

(b) **[suboptimality of Lloyd's method]** For the case $d = 1$ (and $k \geq 2$), show that Lloyd's algorithm is *not* optimal. That is, there is a suboptimal setting of cluster assignment for some dataset (with $d = 1$) for which Lloyd's algorithm will not be able to improve the solution.

(c) **[improving k-means quality]** k-means with Euclidean distance metric assumes that each pair of clusters is linearly separable (see part ii below). This may not be the desired result. A classic example is where we have two clusters corresponding to data points on two concentric circles in the $\mathbb{R}^2$.

  (i) Implement Lloyd's method for k-means algorithm and show the resulting cluster assignment for the dataset depicted above. Give two more examples of datasets in $\mathbb{R}^2$, for which optimal k-means setting results in an undesirable clustering. Show the resulting cluster assignment for the two additional example datasets.

  (ii) Show that for $k = 2$, for any (distinct) placement of centers c_1 and c_2 in $\mathbb{R}^d$, the cluster boundary induced by minimizing the k-means objective (i.e. Eq. 1) is necessarily linear.

One way to get a more *flexible* clustering is to run k-means in a transformed space. The transformation and clustering is done as follows:

- Let G_r denote the r-nearest neighbor graph induced on the given dataset (say the dataset has n datapoints), that is, the datapoints are the vertices (notation: v_i is the vertex associated with datapoint x_i) and there is an edge between vertex v_i and v_j if the corresponding datapoint x_j is one of the r closest neighbors of datapoint x_i.
- Let W denote the $n \times n$ edge matrix, where

$$w_{ij} = \mathbf{1}[\exists \text{ edge between } v_i \text{ and } v_j \text{ in } G_r].$$

- Define $n \times n$ diagonal matrix D as $d_{ii} := \sum_j w_{ij}$, and finally define $L = D - W$.
- Compute the *bottom* k eigenvectors/values of L (that is, eigenvectors corresponding to the k smallest eigenvalues). Let V be the $n \times k$ matrix of of the bottom eigenvectors. One can view this matrix V as a k dimensional representation of the n datapoints.
- Run k-means on the datamatrix V and return the clustering induced.

We'll try to gain a better understanding of this transformation V (which is basically the lower order spectra of L).

(iii) Show that for any vector $f \in \mathbb{R}^n$,

$$f^\mathsf{T} L f = \frac{1}{2}\sum_{ij} w_{ij}(f_i - f_j)^2.$$

(iv) L is a symmetric positive semi-definite matrix.

(v) Let the graph G_r have k connected components $C_1, \ldots, C_k$. Show that the $n \times 1$ indicator vectors $\mathbb{1}_{C_1}, \ldots, \mathbb{1}_{C_k}$ are (unnormalized) eigenvectors of L with eigenvalue 0. (the ith component of an indicator vector takes value one iff the vertex v_i is in the connected component)

Part (v) gives us some indication on why the transformation V (low order spectra of L) is a reasonable representation. Basically: (i) vertices belonging to the same connected component/cluster (ie, datapoints connected with a "path", even if they are located far away or form odd shapes) will have the same value in the representation $V = [\mathbb{1}_{C_1}, \ldots, \mathbb{1}_{C_k}]$, and (ii) vertices belonging to different connected component/cluster will have distinct representation. Thus making it easier for a k-means type algorithm to recover the clusterings.

(vi) For each of the datasets in part (i) (there are total three datasets), run this flexible version of k-means in the transformed space. Show the resulting clustering assignment on all the datasets. Does it improve the clustering quality? How does the choice of r (in G_r) affects the result?

(You must submit your code for parts (i) and (vi) to receive full credit.)