Introduction to Machine Learning **Fall Semester**

Homework 1: Nov 16th, 2016

Due: Dec 7th (See the submission guidelines in the course web site)

Theory Questions

1. **Concentric Circles.** Let $X = \mathbb{R}^2$ and consider the set of concepts of the form:

$$c_r = \left\{(x, y) : x^2 + y^2 \leq r^2\right\}$$

for some non-negative real number r. Show that this class can be (ϵ, δ)-PAC-learned from training data of size $m \geq (1/\epsilon)\log(1/\delta)$. Do not use VC dimension in your proof.

2. **Rectangles.** Let $\mathcal{H}$ be the hypothesis class of axis-aligned rectangles, as described in class. What is the VC-dimension of $\mathcal{H}$?

3. **OR**. Let $\mathcal{X} = \{0, 1\}^n$, $T_1, T_2 \subseteq \{1, ..., n\}$. The hypothesis class $\mathcal{H}$ is

$$\mathcal{H} = \{h_{T_1,T_2} | T_1, T_2 \subseteq \{1, ..., n\}\},$$

where

$$h_{T_1,T_2}(\mathbf{x}) = (\vee_{i \in T_1} x_i) \vee (\vee_{j \in T_2} \bar{x_j}).$$

What is the VC-dimension of $\mathcal{H}$?

Programming Assignment

1. **Nearest Neighbor.** In this question, we will study the performance of the Nearest Neighbor (NN) algorithm on the MNIST dataset. The MNIST dataset consists of images of handwritten digits, along with their labels. Each image has 28 × 28 pixels, where each pixel is in grayscale scale, and can get an integer value from 0 to 255. Each label is a digit between 0 and 9. The dataset has 70,000 images. Althought each image is square, we treat it as a vector of size 784.

   The MNIST dataset can be loaded with `sklearn` as follows:

```
>>> from sklearn.datasets import fetch_mldata
>>> mnist = fetch_mldata('MNIST original')
>>> data = mnist['data']
>>> labels = mnist['target']
```

   Loading the dataset might take a while when first run, but will be immediate later. See `http://scikit-learn.org/stable/datasets/mldata.html` for more details. Define the training and test set of images as follows:

```
>>> import numpy.random
>>> idx = numpy.random.RandomState(0).choice(70000, 11000)
>>> train = data[idx[:10000], :]
>>> train_labels = labels[idx[:10000]]
>>> test = data[idx[10000:], :]
>>> test_labels = labels[idx[10000:]]
```

   It is recommended to use `numpy` where possible, for speed.

   (a) Write a function that accepts as input: (i) a set of images; (ii) a vector of labels, corresponding to the images (ii) a query image; and (iii) a number k. The function will implement the k-NN algorithm to return a prediction of the query image, given the given label set of images. The function will use the k nearest neighbors, using the Euclidean L2 metric. In case of a tie between the k labels of neighbors, it will choose an arbitrary option.

   (b) Run the algorithm using the first $n = 1000$ training images, on each of the test images, using $k = 10$. What is the accuracy of the prediction (measured by 0-1 loss; i.e. the percentage of correct classifications)? What would you expect from a completely random predictor?

   (c) Plot the prediction accuracy as a function of k, for $k = 1, \ldots, 100$. Discuss the results. What is the best k?

   (d) Now we fix k to be the best k from before, and instead vary the number of training images. Plot the prediction accuracy as a function of $n = 100, 200, \ldots, 5000$. Discuss the results.

2. **Union of intervals.** In this question, we will study the hypothesis class of a finite union of disjoint intervals, discussed in class. To review, let the sample space be $\mathcal{X} = [0,1]$ and assume we study a binary classification problem, i.e. $\mathcal{Y} = \{0,1\}$. We will try to learn using an hypothesis class that consists of k intervals. More explicitly, let $I = \{[l_1,u_1],\ldots,[l_k,u_k]\}$ be k disjoint intervals, such that $0 \leq l_1 \leq u_1 \leq l_2 \leq u_2 \leq \ldots \leq u_k \leq 1$. For each such k disjoint intervals, define the corresponding hypothesis as

$$h_I(x) = \begin{cases} 1 & \text{if } x \in [l_1,u_1],\ldots,[l_k,u_k] \\ 0 & \text{otherwise} \end{cases}$$

Finally, define $\mathcal{H}_k$ as the hypothesis class that consists of all hypotheses that correspond to k disjoint intervals:

$$\mathcal{H}_k = \{h_I | I = \{[l_1,u_1],\ldots,[l_k,u_k]\},\ 0 \leq l_1 \leq u_1 \leq l_2 \leq u_2 \leq \ldots \leq u_k \leq 1\}$$

We note that $\mathcal{H}_k \subseteq \mathcal{H}_{k+1}$, since we can always take one of the intervals to be of length 0 by setting its endpoints to be equal. We are given a sample of size $m = \langle x_1, y_1 \rangle, \ldots, \langle x_n, y_m \rangle$. Assume that the points are sorted, so that $0 \leq x_1 < x_2 < \ldots < x_m \leq 1$.

We will now study the properties of the ERM algorithm for this class. Under

`~schweiger/courses/ML2016-7/intervals.py`

you will find a function that implements an ERM algorithm for $\mathcal{H}_k$. Given a sorted list `xs`=$[x_1,\ldots,x_m]$, the respective labeling `ys`=$[y_1,\ldots,y_m]$ and k, the given function `find_best_interval` returns a list of up to k intervals and their error count on the given sample. These intervals have the smallest empirical error count possible from all choices of k intervals or less.

(a) Assume that the true distribution $\mathcal{D}(x,y) = \Pr(y|x) \cdot \Pr(x)$ is: x is distributed uniformly on the interval $[0,1]$, and

$$\Pr(y=1|x) = \begin{cases} 0.8 & \text{if } x \in [0,0.25] \text{ or } x \in [0.5,0.75] \\ 0.1 & \text{if } x \in [0.25,0.5] \text{ or } x \in [0.75,1] \end{cases}$$

and $\Pr(y=0|x) = 1 - \Pr(y=1|x)$.

Write a function that draws m pairs of (x,y) according to the distribution $\mathcal{D}$. Use it to draw a sample of size $m = 100$ and create a plot:

i. Plot the points and their label (have the y axis in range $-0.1, 1.1$ for clarity of presentation).

ii. Mark the lines $x = 0.25, 0.5, 0.75$ clearly on the plot.

iii. Run the `find_best_interval` function on your sample, and plot the intervals clearly.

(b) Note that here, we know the true distribution $\mathcal{D}$, so for every given hypothesis $h \in \mathcal{H}_k$, we can calculate $error(h)$ precisely. What is the hypothesis with the smallest error?

(c) Write a function that, given a list of intervals, calculates the error of the respective hypothesis. Then, for $k = 2$, $m = 10, 15, 20, \ldots, 100$, perform the following experiment $T = 100$ times: (i) Draw a sample of size m and run the ERM algorithm on it; (ii) Calculate the empirical error for the returned hypothesis; (iii) Calculate the true error for the returned hypothesis. Plot the average empirical and true errors, averaged across the T runs, as a function of m. Discuss the results - is each error seems to be decreasing or increasing in m? Why?

(d) Draw a data set of $m = 50$ samples. Find the best ERM hypothesis for $k = 1, 2, \ldots, 20$, and show the empirical error as a function of k. How does the error behaves? What is k^*, defined as the k with the smallest error? Does this mean the hypothesis with k^* intervals is a good choice?

(e) Repeat the above procedure $T = 100$ times. Plot the average empirical error across runs, and the average true error across runs.

(f) Given a new data set with $m = 50$ samples, how would you perform cross-validation to choose the best hypothesis? Demonstrate.