CS534 — Written Assignment 4 —

1. PAC learnability. Consider the concept class C of all conjunctions (allowing negations) over n boolean features. Prove that this concept class is PAC learnable. Note that we have gone through this example in class but not in full detail. For this problem, you should work out the full detail, including showing that there exists a poly-time algorithm for finding a consistent hypothesis.

   Solution: *Use the consistent-learn framework, we need to first show that $\log |C|$ is polynomial in n, which will allow us to have polynomial sample complexity. This is simple as there are three possible choices for each boolean feature, appearing positively, negatively, or not appearing in the conjunction. This gives $|C| = 3^n$ possible conjunctions. The sample complexity bound states that to achieve $1-\epsilon$ accuracy with $1-\delta$ probability, the number of training examples we need is:*

$$\frac{1}{\epsilon}(\log |C| + \log \frac{1}{\delta}) = \frac{1}{\epsilon}(n \log 3 + \log \frac{1}{\delta})$$

   which is linear in $n, \frac{1}{\epsilon}, \frac{1}{\delta}$. The next step is to prove that we have a polynomial time algorithm for finding a consistent hypothesis in C. Given m training examples, we can find a consistent hypothesis as follows. Begin with $h = x_1 \wedge \bar{x_1} \wedge \cdots \wedge x_n \wedge \bar{x_n}$, that is we include each boolean feature x_i and its negation $\bar{x}_i$. We then focus on only positive examples. For each training example, delete from h every literal that is false in the example. For example, if we observe a positive example 0100 (four features), we will remove x_1, $\bar{x_2}$, x_3, and x_4. After we finish processing all positive examples, the remaining h is guaranteed to be consistent with all training examples (both positive and negative). This is because literals that are in the target concept will never be deleted from h.

   Consider the following example, if we observe 3 positive training examples as show below:

x_1	x_2	x_3	x_4	x_5	y
0	*1*	*0*	*0*	*1*	*1*
1	*1*	*0*	*0*	*1*	*1*
0	*1*	*0*	*1*	*1*	*1*

   After processing all three training examples, we obtain $h = x_2 \wedge \bar{x}_3 \wedge x_5$. Clearly this can be done in time that is linear in n (the number of features) and m (the number of training examples).

   This completes the proof that the concept class of boolean conjunctions is PAC learnable.

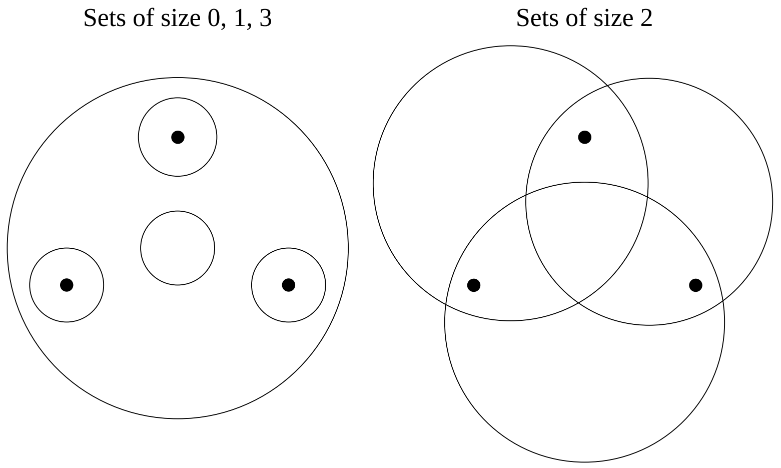
   Note that one might argue that we can learn a decision tree that is consistent with the training data in polynomial time for the second part of the proof. However, this is not a valid proof because the learned decision tree does not necessarily (actually most likely not) belong to the hypothesis space we are considering, i.e., boolean conjunctions. The requirement for our proof is that there exists an efficient algorithm (poly time) for finding a consistent $h \in C$.

2. VC dimension. Consider the hypothesis space H_c = circles in the (x, y) plane. Points inside the circle are classified as positive examples. What is the VC dimension of H_c? Please fully justify your answer.

   It takes only 3 parameters to specify a circle in 2-d space — the (x, y) coordinates of the center, and the radius. We can guess that the VC dimension is no larger than 3. The following 3 points can be shattered.

   With 4 points, you can't get all sets of size 2. Specifically, if we identify the two points that are furthest from one another, and label them positive, and the other two negative, we cannot possibly create a circle enclosing only the positive pairs.

3. Consider the class C of concepts of the form $(a \leq x \leq b) \wedge (c \leq y \leq d)$, where a, b, c, and d are integers in the interval $[0, 99]$. Note that each concept in this class corresponds to a rectangle with integer-valued boundaries on a portion of the (x, y) plane. Hint: Given a region in the plane bounded by the points $(0, 0)$ and $(n-1, n-1)$, the number of distinct rectangles with integer-valued boundaries within this region is $\left(\frac{n(n-1)}{2}\right)^2$.



(a) Give an upper bound on the number of randomly drawn training examples sufficient to assure that for any target concept c in C, any consistent learner using $H = C$ will, with probability 95%, output a hypothesis with error at most 0.15.

We will compute the size of this hypothesis space and then apply the bound

$$m \leq \frac{1}{\epsilon}\left[\ln|H| + \log\frac{1}{\delta}\right]$$

For $a \leq x \leq b$ and $x \in \{0, 1, \ldots, 99\}$, there are $\frac{100 \cdot 99}{2} = 4950$ legal combinations. The same argument holds for $c \leq y \leq d$. So the total number of axis-parallel rectangles is 24,502,500. The problem specifies $\epsilon = 0.15$ and $\delta = 0.05$, so plugging in, we get (in lisp notation):

```
(* (/ 1.0 0.15) (+ (log 24502550) (log (/ 1.0 0.05)))) =
(* 6.666666666666667 (+ 17.01428775173149 2.995732273553991)) =
133.4001335019032
```

So we need at least 134 examples.

(b) Now suppose the rectangle boundaries a, b, c, and d take on *real* values instead of integer values. Update your answer to the first part of this question.

We will compute the VC dimension of the space and apply the bound

$$m \leq \frac{1}{\epsilon}\left[4\lg(2/\delta) + 8d\lg(13/\epsilon)\right]$$

From the previous problem, the VC dimension is 4. So this becomes

```
(* (/ 1.0 0.15)
(+ (* 4 (log (/ 2.0 0.05) 2))
(* 8 4 (log (/ 13.0 0.15) 2.0)))) =
(* 6.666666667
(+ 21.28771237954945 205.99696999383357)) =
1515.2312
```

So we need 1,516 examples.

Tom Mitchell's ML textbook gives a nice analysis of the following algorithm. Let h be the smallest axis-parallel rectangle that covers all of the positive training examples. If there is no noise in the training data, then we know that $h \subseteq t$, where t is the true axis parallel rectangle. We can then bound the error between h and t by the probability that a new data point will fall outside of h but inside t. The resulting bound is

$$m \geq (4/\epsilon)\log(4/\delta).$$

The question said you should bound the error rate of a learning algorithm that chooses any consistent rectangle. But if we restrict the algorithm to choosing the smallest consistent rectangle, then we get a bound of 117 examples!

4. In the discussion of learning theory, a key assumption is that the training data has the same distribution as the test data. In some cases, we might have noisy training data. In particular, we consider the binary classification problem with labels $y \in \{0, 1\}$, and let $\mathcal{D}$ be a distribution over $\{x, y\}$ that we think of as the original uncorrupted distribution. However, we don't directly observe from this distribution for training. Instead, we observe from $\mathcal{D}_\tau$, a corrupted distribution over $\{x, y\}$, which is the same as $\mathcal{D}$, except that the label y has some probability $0 < \tau < 0.5$ to be flipped. In other words, our training data is generated by first sampling from $\mathcal{D}$, then with probability τ (independent of the observed x and y) replace y with $1 - y$. Note that $\mathcal{D}_0 = \mathcal{D}$.

   The distribution $\mathcal{D}_\tau$ models the setting in which an unreliable labeler is labeling the training data for you, and each example has a probability τ of being mislabeled. Even though the training data is corrupted, we are still interested in evaluating our hypotheses w.r.t. the original uncorrupted distribution $\mathcal{D}$.

   We define the generalization error with respect to $\mathcal{D}_\tau$ to be:

$$\epsilon_\tau = P_{(x,y)\sim\mathcal{D}_\tau}[h(x) \neq y]$$

   Note that ϵ_0 is the generalization error with respect to the clean distribution. It is with respect to ϵ_0 that we wish to evaluate our hypothesis.

   a. For any hypothesis h, the quantity $\epsilon_0(h)$ can be calculated as a function of $\epsilon_\tau(h)$ and τ. Write down a formula for $\epsilon_0(h)$ in terms of $\epsilon_\tau(h)$ and τ.
   If h is correct on an instance from $\mathcal{D}_0$, it will be correct for for $\mathcal{D}_\tau$ with probability $1 - \tau$, and be incorrect with probability τ. If h is incorrect on an instance from $\mathcal{D}_0$, with probability τ it remain incorrect for $\mathcal{D}_\tau$, and correct with probability $1 - \tau$. Thus we have:

$$\epsilon_\tau = (1 - \epsilon_0) * \tau + \epsilon_0 * (1 - \tau)$$

   Solving ϵ_0, we have:

$$\epsilon_0 = \frac{\epsilon_\tau - \tau}{1 - 2\tau}$$

   b. Let H be finite and suppose our training set $\mathcal{S} = \{(x^i, y^i) : i = 1, ..., m\}$ is obtained by sampling IID from $\mathcal{D}_\tau$. Suppose we pick the hypothesis $h \in H$ that minimizes the training error: $\hat{h} = \arg\min_{h\in H} \hat{\epsilon}_\tau(h)$. Also, let $h^* = \arg\min_{h\in H} \epsilon_0(h)$, i.e., the optimal hypothesis in H. Let any $\delta, \gamma > 0$ be given, prove that for

$$\epsilon_0(\hat{h}) \leq \epsilon_0(h^*) + 2\gamma$$

   to hold with probability $1 - \delta$, we need to have training data size

$$m \geq \frac{1}{2(1 - 2\tau)^2\gamma^2} \log \frac{2|H|}{\delta}$$

   Hint: you will need to use the following fact derived from Hoeffding bound:

$$\forall h \in H, |\epsilon_\tau(h) - \hat{\epsilon}_\tau(h)| \leq \hat{\gamma} \text{ with probability } (1 - \delta), \delta = 2|H| \exp(-2\hat{\gamma}^2 m)$$

   where $\hat{\epsilon}_\tau(h)$ is the training error of h on training data $\mathcal{S}$ generated from corrupted distribution D_τ. You will also need to use the answer to subproblem (a).
   we begin by note that

$$\forall h \in H, |\epsilon_\tau(h) - \hat{\epsilon}_\tau(h)| \leq \hat{\gamma} \textit{ with probability } (1 - \delta), \delta = 2|H| \exp(-2\hat{\gamma}^2 m)$$

Using results from the last subproblem, we have:

$$\begin{aligned}
\epsilon_0(\hat{h}) &= \frac{\epsilon_\tau(\hat{h}) - \tau}{1-2\tau} \Rightarrow \\
\epsilon_0(\hat{h}) &\le \frac{\hat{\epsilon}_\tau(\hat{h}) - \tau + \hat{\gamma}}{1-2\tau} \textit{ with } (1-\delta) \textit{ probability} \Rightarrow \\
\epsilon_0(\hat{h}) &\le \frac{\hat{\epsilon}_\tau(h^*) - \tau + \hat{\gamma}}{1-2\tau} \textit{ with } (1-\delta) \textit{ probability} \Rightarrow \\
\epsilon_0(\hat{h}) &\le \frac{\epsilon_\tau(h^*) - \tau + 2\hat{\gamma}}{1-2\tau} \textit{ with } (1-\delta) \textit{ probability} \Rightarrow
\end{aligned}$$

Plug in $\epsilon_\tau(h^*) = (1-2\tau)\epsilon_0(h^*) + \tau$, we have:

$$\begin{aligned}
\epsilon_0(\hat{h}) &\le \frac{(1-2\tau)\epsilon_0(h^*) + 2\hat{\gamma}}{1-2\tau} \textit{ with } (1-\delta) \textit{ probability} \Rightarrow \\
\epsilon_0(\hat{h}) &\le \epsilon_0(h^*) + \frac{2\hat{\gamma}}{1-2\tau} \textit{ with } (1-\delta) \textit{ probability} \Rightarrow
\end{aligned}$$

Now do a variable exchange: $\gamma = \frac{\gamma}{1-2\tau}$, we have:

$$\epsilon_0(\hat{h}) \le \epsilon_0(h^*) + 2\gamma \textit{ with } (1-\delta) \textit{ probability, where } \delta = 2|H|\exp(-2(\gamma(1-2\tau))^2 m$$

Let $2|H|\exp(-2(\gamma(1-2\tau))^2 m \le \delta$, we have:

$$m \ge \frac{1}{2(1-2\tau)^2\gamma^2}\log\frac{2|H|}{\delta}$$

***Remark**: This suggests that m training examples that are corrupted by the noise will be worth about $(1-2\tau)^2 m$ uncorrupted training examples. This is a useful rule of thumb to know if you want to decide how much to pay for a more reliable source of training data. Also note that when τ approaches 0.5, the labels become more and more random and the training examples eventually becomes useless as we will need infinite amount of data in order to have any guarantee.*

5. Boosting. Please show that in each iteration of Adaboost, the weighted error of h_i on the updated weights D_{i+1} is exactly 50%. In other words, $\sum_{j=1}^{N} D_{i+1}(j) I(h_i(X_j) \ne y_j) = 50\%$.

   ***Solution:** Let ϵ_i be the weighted error of h_i, that is $\epsilon_i = \sum_{j=1}^{N} D_i(j) I(h_i(X_j) \ne y_j)$, where $I(\cdot)$ is the indicator function that takes value 1 if the argument is true, and value 0 otherwise. Following the update rule of Adaboost, let's assume that the weights of the correct examples are multiplied by $e^{-\gamma}$, and those of the incorrect examples are multiplied by e^{γ}. After the updates, to make sure that h_i has exactly 50% accuracy, we only need to satisfy the following:*

$$\begin{gathered}
\epsilon e^{-\gamma} = (1-\epsilon)e^{\gamma} \Rightarrow \\
\frac{\epsilon}{1-\epsilon} = e^{2\gamma} \Rightarrow \\
\log\frac{\epsilon}{1-\epsilon} = 2\gamma \Rightarrow \\
\gamma = \frac{1}{2}\log\frac{\epsilon}{1-\epsilon}
\end{gathered}$$

   which is exactly the value Adaboost uses.