Midterm 1
Due: March 20th, 2018

There were six problems. Each problem was scored out of 10 points with a total target score of 50. We counted the lowest scoring problem as an extra credit of 1/10 of its scored value.
Problem 1 (algorithms)

For this problem, we are looking for responses that both indicate your assessment as to a possible accuracy change and your understanding of the algorithm that led to this assessment. Answers should be two or three sentences long and focus on the relevant and important issue.

a. We have trained a decision tree (binary vector input, binary label) on a data set. Then, we create a new data set that is identical to the original but it includes a new feature that is set randomly, with no particular correlation to any of the other features or the label and retrain. What would you expect to happen to the testing performance of the algorithm (assuming the new feature is available)?

   Given the newly generated feature is random, we would expect no information gain from splitting on this feature. The tree will either ignore this feature (no performance change) or split (no expected performance change).

   Also ok: Slightly negatively impact performance, essentially due to overfitting.

b. We have trained logistic regression (continuous vector input, binary label) on a data set. Then, we create a new data set that is identical to the original but includes a new attribute that is the Boolean negation of the label and retrain. What would you expect to happen to the testing performance of the algorithm (assuming the new feature is available)?

   Logistic regression will assign a large negative weight to the new feature, resulting in perfect performance.

Problem 2 (classifiers)

We trained a 1-NN classifier on a data set with a two-dimensional continuous input space. Here is a sample of the points the resulting classifier assigns positive labels to. Mark in the diagram where the two positive and two negative training points are.

Positives: (0.5, 0.5) and (0.7, 0.2). Negatives: (0.8, 0.4) and (0.5, 0.0).
Problem 3 (representation)

a. In a two-bit input space, a binary Naïve Bayes classifier can be represented by \( b \), the prior probability that the output class is 1 (\( \Pr(y = 1) \)), and \( b^y \), the probability that an example with output class \( y \) has bit \( i \) set to 1 (\( \Pr(x_i = 1|y) \)). Define \( b, b^{10}, b^{11}, b^{20} \) and \( b^{21} \) so the resulting classifier represents a disjunction—it assigns an output of 0 to input 00, and an output of 1 to 01, 10, and 11.

To correctly classify all four of possible inputs, we need to satisfy the following four inequalities:

\[
\begin{align*}
P(y = 1|x = 00) < P(y = 0|x = 00) & \implies b \cdot (1 - b^{11})(1 - b^{21}) < (1 - b)(1 - b^{10})(1 - b^{20}) \\
P(y = 1|x = 10) > P(y = 0|x = 10) & \implies b \cdot b^{11}(1 - b^{21}) > (1 - b)b^{10}(1 - b^{20}) \\
P(y = 1|x = 01) > P(y = 0|x = 01) & \implies b \cdot (1 - b^{11})b^{21} > (1 - b)(1 - b^{10})b^{20} \\
P(y = 1|x = 11) > P(y = 0|x = 11) & \implies b \cdot b^{11} \cdot b^{21} > (1 - b)b^{10} \cdot b^{20}.
\end{align*}
\]

Many possible answers can work. Here’s one:

\[
\begin{align*}
b &= 0.9 \\
b^{10} &= 0.1 \\
b^{11} &= 0.9 \\
b^{20} &= 0.1 \\
b^{21} &= 0.9.
\end{align*}
\]

b. Will the following training set produce the correct classifier? For smoothing purposes, any 0/1 probability is replaced by 0.1/0.9.

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

No, it returns the wrong answer for 00.

Our estimates (after smoothing) would be

\[
\hat{b} = 0.9, \quad \hat{b}^{10} = 0.1, \quad \hat{b}^{11} = 0.5, \quad \hat{b}^{20} = 0.1, \quad \hat{b}^{21} = 0.5.
\]

\[
\begin{align*}
P(y = 1|x = 00) &= 0.9 \cdot (1 - 0.5) \cdot (1 - 0.5) = 0.17 \\
P(y = 0|x = 00) &= 0.1 \cdot (1 - 0.1)(1 - 0.1) = 0.081
\end{align*}
\]

Therefore, since 0.17 > 0.081, we would classify \( x = 00 \) as 1, when it should be classified as 0.
Problem 4 (VC dimension)

What is the VC dimension of binary decision trees on \(d\)-dimensional binary data? Provide a complete proof.

The answer is \(2^d\). There are two parts to the proof: (1) Showing it can shatter at least one set of size \(2^d\) and showing it cannot shatter any set of \(2^{d+1}\).

1. The tree can assign each of the \(2^d\) distinct input strings to a unique leaf. We can then label each leaf completely independent of the others, resulting in all \(2^{2^d}\) possible assignments being covered.

2. There are only \(2^d\) distinct input patterns. Therefore, a dataset of size \(2^d + 1\) must have two identical data points. These two identical points will be classified by the same leaf in the decision tree. We cannot shatter the set since we cannot satisfy the case in which the identical points are labeled differently.
Problem 5 (loss)
Here’s how Wikipedia defines the **Sleeping Beauty Problem**:

*Sleeping Beauty volunteers to undergo the following experiment and is told all of the following details: On Sunday she will be put to sleep. Once or twice, during the experiment, Beauty will be awakened, interviewed, and put back to sleep with an amnesia-inducing drug that makes her forget that awakening. A fair coin will be tossed to determine which experimental procedure to undertake:

- If the coin comes up heads, Beauty will be awakened and interviewed on Monday only.
- If the coin comes up tails, she will be awakened and interviewed on Monday and Tuesday.*

*In either case, she will be awakened on Wednesday without interview and the experiment ends. Any time Sleeping Beauty is awakened and interviewed she will not be able to tell which day it is or whether she has been awakened before. During the interview Beauty is asked: “What is your credence now for the proposition that the coin landed heads?”.*

We will attack this problem by defining a loss function and asking Sleeping Beauty to declare her “credence” the coin landed heads as by reporting the value that minimizes the loss. Note that she can’t distinguish the different awakenings, so she must report one value, which we’ll call $p$, no matter when she is asked to report her credence.

a. Each time she is awakened and it is Monday, her loss is increased by $p$ if the coin is tails and $1 - p$ if the coin is heads. Her loss is unchanged on Tuesday regardless of what she reports. What should she report to minimize her loss and why?

The loss function is $\frac{1}{2}p + \frac{1}{2}(1 - p) = 1/2$, so it doesn’t matter what she reports.

b. Each time she is awakened, we flip a second weighted coin that comes up heads with probability $p$. If, at the end of the experiment, all coins flipped during the experiment match (all heads or all tails), no loss is incurred. Otherwise, a loss of 1 is incurred. What should she report to minimize her loss and why?

The loss function becomes: $\frac{1}{2} + \frac{1}{2}p - \frac{1}{2}p^2$. Taking the derivative and solving for when it equals zero gives $p = \frac{1}{2}$, however, this value represents the maximum loss. Checking the two boundary conditions reveals that the minimum loss of $\frac{1}{2}$ is achieved for both $p = 0$ and $p = 1$. 
Problem 6 (optimizers)

For binary classifiers of \( n \)-bit strings, consider the hypothesis class of \( n \)-bit string equality, \( H = \{ h_b : b \in \{0,1\}^n \} \) where:

\[
h_b(x) = \begin{cases} 
1 & \text{if } x = b, \\
0 & \text{otherwise.}
\end{cases}
\]


a. Explain how to compute the ERM for this class in the realizable case. State the computational complexity of the algorithm in the context of a data set of size \( m \). (You can assume there is at least one positive example in the training set.) Example dataset with \( n = 3 \) and \( m = 4 \): \( \{(001, 1), (010, 0), (111, 0), (101, 0)\} \). ERM is \( h_{001} \).

Loop through all data \( (x, y) \in S \) looking for a positive label \( y = 1 \) \( (O(m)) \). Since we’re in the realizable case, this positive example must correspond to the true hypothesis. Set \( b = x \) and return \( b \) \( (O(n)) \). Total time is \( O(m + n) \).

b. Explain how to compute the ERM for this class in the agnostic case. State the computational complexity of the algorithm in the context of a data set of size \( m \). Example dataset with \( n = 3 \) and \( m = 5 \): \( \{(001, 1), (010, 0), (111, 1), (001, 0), (101, 0)\} \). ERM is \( h_{111} \).

First, note that we can evaluate the loss for any \( h_b \) in time \( O(mn) \). This calculation involves checking, for each \( (x, y) \in S \) whether \( b = x \) and adding \( 1 - y \) to the loss if so, and \( y \) to the loss otherwise.

Then, we can check the loss of all of the hypotheses \( h_x \) where \( x \) is one of the training examples and chose the \( h_x \) with minimum loss. The time is \( m \) times the loss calculation, resulting in \( O(m^2n) \).

We also need to check one other case, which is the loss for \( h_b \) where \( b \) is not one of the input examples. The reason is that all such \( b \) have the same loss and, in the weird case where all \( x \) appearing in the training set have more negative labels than positive, such a hypothesis can minimum loss. Finding such a \( b \) can be done in time \( O(m^2n) \) by enumerating each bit pattern and checking to see if each one is already in the data set. As soon as it finds one that is not in there, it can halt (so, not all \( 2^n \) patterns need to be considered).

There is an even more efficient algorithm that is possible. Using an associative array (hash table), run through the data and compute, for each \( x \), the number of times it appears with a positive label \( (p(x)) \) and a negative label \( (n(x)) \). Also track the total number of positives \( p \) and negatives \( n \) in the entire data set. This computation can be done in time \( O(mn) \) assuming we can hash each training example in time \( O(n) \).

With this information stored, we can find a \( b \) not in the training set in time \( O(m) \). We can evaluate the loss of any \( h_b \) in time \( O(1) \) via \( n(b) + (p - p(b)) \) since choosing \( b \) as the hypothesis will be wrong whenever \( b \) has a negative label and it will be wrong whenever any other hypothesis in the set has a positive label. We can return the \( b \) with minimum loss in linear time \( (O(mn)) \).