Written assignment

1. Consider the following decision tree:

(a) Draw the decision boundaries defined by this tree. Each leaf of the tree is labeled with a letter. Write this letter in the corresponding region of input space.

(b) Give another decision tree that is syntactically different but defines the same decision boundaries. This demonstrates that the space of decision trees is syntactically redundant. Is this redundancy a statistical problem (i.e., does it affect the accuracy of the learned trees)? Is it a computational problem (i.e., does it increase or decrease the computational complexity of finding an accurate tree)?
This redundancy is not likely to be a statistical problem, because it has no effect on the expressive power of decision trees. That is, the same space of decision trees can be represented with or without the redundancy. The redundancy is likely to be a computational advantage, because it makes it easier for an imperfect greedy heuristic to find a good solution. For depth-first search methods (such as our greedy algorithm), the ideal search space is one such that every path leads to a solution. Redundancy increases the chance that any random sequence of node expansions will lead to a good tree.

2. In the basic decision tree algorithm, we choose the feature/value pair with the maximum mutual information as the test to use at each internal node of the decision tree. Suppose we modified the algorithm to choose at random from among those feature/value combinations that had non-zero mutual information, but that we kept all other parts of the algorithm unchanged.

(a) Prove that if a splitting feature/value combination has non-zero mutual information at an internal node, then at least one training example must be sent to each of the child nodes.

Proof by contradiction. Suppose there is a split with non-zero mutual information but that all of the training examples go to one of the child nodes. Let \((p, n)\) be the number of positive and negative examples at the internal node. Suppose as a result of the split, all of these examples go to the left child. Then it will also have \((p, n)\) examples, and the right child will receive \((0, 0)\). Now we plug into the formula for mutual information. The mutual information will be

\[
H(p, n) - [1.0H(p, n) + 0.0H(0, 0)] = 0
\]

This contradicts the assumption that the split had non-zero mutual information. Therefore, either such a split has zero mutual information or at least one of the training examples is sent to each child.

(b) What is the maximum number of leaf nodes that such a decision tree could contain if it were trained on \(m\) training examples?

The maximum number would be obtained if each training example were alone in its own leaf: \(m\) leaves. We can have no leaves with zero training examples, because each split has non-zero mutual information and therefore sends at least one example to each child. (Actually, let \(m'\) be the number of distinct training examples. Then no tree can have more than \(m'\) leaves.)
(c) What is the maximum number of leaf nodes that a decision tree could contain if it were trained on \( m \) training examples using the original maximum mutual information version of the algorithm? Is it bigger, smaller, or the same as your answer to (b)?

Using maximum mutual information, in the worst case we could also get one example in each leaf node. (or \( m' \) distinct examples and leaves).

(d) How do you think this change would affect the accuracy of the decision trees produced on average? Why?

Although in the worst case, both decision trees will have the same size, I would guess that in the average case, using randomized splits would give lower accuracy, particularly if there are irrelevant or noisy features in the data. A random split is more likely to split on an irrelevant or inappropriate feature. This will have the unfortunate result of subdividing the data randomly. This effectively creates two learning problems equivalent to the unsplit learning problem, but each has only half as much data. We know that the more data we have available, the more accurate the hypothesis will be on the average. Conversely, the less data we have available, the less accurate the hypothesis will be on the average.
3. Consider the following training set:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

a. Learn a Naive Bayes classifier by estimating all necessary probabilities. Make prediction for (A=1, B=0, C=0).

Below are the list of probabilities that are learned from the training data.
Class prior: \( p(y = 1) = \frac{1}{2} \)

Class conditional probability for \( y = 1 \):
\( p(A = 0|y = 1) = \frac{1}{3}; p(B = 0|y = 1) = \frac{1}{3}; p(C = 0|y = 1) = \frac{2}{3} \)

Class conditional probability for \( y = 0 \):
\( p(A = 0|y = 0) = \frac{2}{3}; p(B = 0|y = 0) = \frac{1}{3}; p(C = 0|y = 0) = \frac{1}{3} \)

Prediction for (1,0,0):
\[
p(y = 1|X) = \frac{p(y = 1) \cdot P(A = 1|y = 1) \cdot P(B = 0|y = 1) \cdot P(C = 0|y = 1)}{Z} = 0.5 \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} = \frac{2}{27}
\]
\[
p(y = 0|X) = \frac{p(y = 0) \cdot P(A = 1|y = 0) \cdot P(B = 0|y = 0) \cdot P(C = 0|y = 0)}{Z} = 0.5 \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{54}
\]

Note that \( p(y = 1|X) + p(Y = 0|X) = 1 \), thus
\[
Z = \frac{2}{27} + \frac{1}{54} = \frac{5}{54}
\]

This gives us:
\[
p(y = 1|X) = \frac{4}{5}; p(y = 0|X) = \frac{1}{5}
\]

Note that you may notice that one can also directly make prediction \( y = 0 \) without calculating the normalization factor \( Z \) by simply noticing that no matter what \( Z \) value is, \( p(y = 1|X) \) is greater than \( p(y = 0|X) \).

b. Suppose we know that A, B and C are independent random variables, can we say that the Naive Bayes assumption is valid? (Note that the particular data set is irrelevant for this question). If your answer is yes, please explain why; if you answer is no please give an counter example.

No. \( p(A, B, C) = p(A)p(B)p(C) \) does not imply \( p(A, B, C|y) = p(A|y)p(B|y)p(C|y) \).

d. Learn a decision tree from the training set shown above using the Mutual Information criterion.
Since $A$ and $C$ give the same amount of information gain, we randomly pick $C$ as the root node test.

For the next step, we focus on the left branch. See the following figure. It is clear from the figure without computation that $B$ is a better choice.
We can continue and finally reach the following decision tree.

Note that if you happen to choose A as the root node, the resulting decision tree will be bigger, but correct none-the-less.

5. Let’s consider an alternative neural network diagram different from the lectures. In this dia-
gram, a neural network uses a softmax activation function for its output layer. Its outputs can be interpreted as posterior probabilities $P(y|x)$ for a categorical target variable $y$. Consider the following neural network with three output units. The softmax activation function is defined as: 

$$\hat{y}_i = \frac{\exp(x_i)}{\sum_{j=1}^{3} \exp(x_j)}$$

(as opposed to what we saw in class $\hat{y}_i = \frac{1}{1 + \exp(-x_i)}$), where $x_i$ is the net input input the activation function of the output node $i$. Note that $\hat{y}_1 + \hat{y}_2 + \hat{y}_3 = 1$, making it a valid posterior probability. In this problem, we will compute the derivatives needed for the backpropagation algorithm for this kind of network.

![Neural Network Diagram](image)

a. Write down the log likelihood objective function $J(w)$ for this network, where $w$ is the concatenation of $W_6, W_7, W_8, W_9, W_{10},$ and $W_{11}$. You may assume that each training example has the form $(x, y)$, where $x = (1, x_1, x_2, x_3, x_4)$ and $y = (y_1, y_2, y_3)$. There are only three possible $y$ values: $y = (1, 0, 0)$, $y = (0, 1, 0)$, and $y = (0, 0, 1)$.

$$J(w) = \sum_i \sum_k -y_{i,k} \log P(y_{i,k}|x_i)$$

where $P(y_{i,k}|x) = \hat{y}_{i,k}$ is the predicted probability of class $k$ for training example $i$. I have included a minus sign so that our goal is to minimize $J$, just as it was for squared error loss.

Let $Z = \exp a_9 + \exp a_{10} + \exp a_{11}$ be the normalizer of the softmax. Then we can rewrite $J$ as

$$J(w) = \sum_i -y_{i,1} \log \hat{y}_1 - y_{i,2} \log \hat{y}_2 - y_{i,3} \log \hat{y}_3$$

$$= \sum_i -y_{i,1} \log e^{a_9} Z - y_{i,2} \log e^{a_{10}} Z - y_{i,3} \log e^{a_{11}} Z$$

$$= -y_{i,1} a_9 - y_{i,2} a_{10} - y_{i,3} a_{11} + (y_{i,1} + y_{i,2} + y_{i,3}) \log Z$$

$$= -y_{i,1} a_9 - y_{i,2} a_{10} - y_{i,3} a_{11} + \log Z$$

b. Compute the partial derivative

$$\frac{\partial J(w)}{\partial w_{9,6}}$$
Let’s consider just one training example \((x, y)\) and drop the subscript \(i\). The objective function becomes

\[
J(w) = -y_1a_9 - y_2a_{10} - y_3a_{11} + \log Z
\]

where exactly one of the \(y_k\) is non-zero.

\[
\frac{\partial J(w)}{\partial w_{9,6}} = -y_1 \frac{\partial a_9}{\partial w_{9,6}} - y_2 \frac{\partial a_{10}}{\partial w_{9,6}} - y_3 \frac{\partial a_{11}}{\partial w_{9,6}} + \frac{\partial \log Z}{\partial w_{9,6}}
\]

Now \(a_{10}\) and \(a_{11}\) do not involve \(w_{9,6}\), so these partial derivatives are zero. Hence we just have

\[
\frac{\partial J(w)}{\partial w_{9,6}} = -y_1 \frac{\partial a_9}{\partial w_{9,6}} + \frac{\partial \log Z}{\partial w_{9,6}}
\]

Now \(a_9 = W9 \cdot A\), where \(A\) is the vector of hidden unit activations. Expanding the dot product, \(a_9 = \sum_u w_{9,u}a_u\). From this, we obtain:

\[
\frac{\partial J(w)}{\partial w_{9,6}} = -y_1 a_6 + \frac{\partial \log Z}{\partial w_{9,6}}
\]

Now let’s compute \(\partial \log Z/\partial w_{9,6}\):

\[
\frac{\partial \log Z}{\partial w_{9,6}} = \frac{1}{Z} \frac{\partial Z}{\partial w_{9,6}} = \frac{1}{Z} \left( \frac{\partial \exp a_9 + \exp a_{10} + \exp a_{11}}{\partial w_{9,6}} \right)
\]

Only the first term \((a_9)\) depends on \(w_{9,6}\), so we obtain

\[
\frac{\partial \log Z}{\partial w_{9,6}} = \frac{1}{Z} \frac{\partial \exp a_9}{\partial w_{9,6}} = \frac{\exp a_9}{Z} \frac{\partial a_9}{\partial w_{9,6}} = \hat{y}_1 a_6
\]

Plugging this back into the original derivation, we have

\[
\frac{\partial J(w)}{\partial w_{9,6}} = -y_1 a_6 + \frac{\partial \log Z}{\partial w_{9,6}} = -y_1 a_6 + \hat{y}_1 a_6 = a_6(\hat{y}_1 - y_1)
\]

As with sigmoidal neural networks, it is useful to define

\[
\delta_9 = \hat{y}_1 - y_1
\]

c. Compute the partial derivative

\[
\frac{\partial J(w)}{\partial w_{6,3}}
\]
This is the same as for squared error loss, assuming that we have redefined $\delta$ as above:

$$\delta_6 = a_6(1 - a_6) \sum_{u=9}^{11} w_{u,6}\delta u.$$ 

d. Generalize your answers to (b) and (c) and write the pseudo-code for the backpropagation algorithm using them.

- **Forward pass.** Compute $a_u$ and $\hat{y}_u$ for hidden units $u$ and output units $v$.
- **Compute Errors.** Compute $\delta_v = (\hat{y}_v - y_v)$ for each output unit.
- **Compute Hidden Unit Deltas.** Compute $\delta_u = a_u(1 - a_u)\sum_v w_{v,u}\delta_v$.
- **Compute Gradient.** Compute $\frac{\partial J}{\partial w_{u,i}} = \delta_u x_{ij}$ for input-to-hidden weights. Compute $\frac{\partial J}{\partial w_{v,u}} = \delta_v a_{iu}$ for hidden-to-output weights.
- **Take Gradient Descent Step**

$$W := W - \eta\nabla_W J(x_i)$$

6. [6] Cubic Kernels. In class, we showed that the quadratic kernel $K(x_i, x_j) = (x_i \cdot x_j + 1)^2$ was equivalent to mapping each $x$ into a higher dimensional space where

$$\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

for the case where $x = (x_1, x_2)$. Now consider the cubic kernel $K(x_i, x_j) = (x_i \cdot x_j + 1)^3$. What is the corresponding $\Phi$ function (again, for the special case where $x = (x_1, x_2)$)?

Let $x_i = (x_{i1}, x_{i2})$ and $x_j = (x_{j1}, x_{j2})$. Then the kernel is

$$K(x_i, x_j) = (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^3$$

$$= (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^2 \cdot (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)$$

$$= (x_{i1}^2 x_{j1} + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 1) \cdot (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)$$

$$= x_{i1}^3 x_{j1} + 3x_{i1}^2 x_{j1}^2 + 3x_{i1} x_{j1}^3 +$$

$$3x_{i1}^2 x_{j1} x_{i2} x_{j2} + 6x_{i1} x_{i2} x_{j1} x_{j2} + 3x_{i1} x_{j1} x_{i2}^2 x_{j2}^2 +$$

$$3x_{i2} x_{j2} + 3x_{i2}^2 x_{j2}^2 + x_{i2}^3 x_{j2}^3 + 1$$

$$= (x_{i1}^3, \sqrt{3}x_{i1}^2, \sqrt{3}x_{i1}, \sqrt{3}x_{i1} x_{i2}, \sqrt{6}x_{i1} x_{i2}, \sqrt{3}x_{i1} x_{i2}^2, \sqrt{3}x_{i2}, \sqrt{3}x_{i2}^2, x_{i2}^3, 1) \cdot$$

$$\cdot (x_{j1}^3, \sqrt{3}x_{j1}^2, \sqrt{3}x_{j1}, \sqrt{3}x_{j1} x_{j2}, \sqrt{6}x_{j1} x_{j2}, \sqrt{3}x_{j1} x_{j2}^2, \sqrt{3}x_{j2}, \sqrt{3}x_{j2}^2, x_{j2}^3, 1)$$

Hence, the function $\Phi(x) = (x_1^3, \sqrt{3}x_1^2, \sqrt{3}x_1, \sqrt{3}x_1 x_2, \sqrt{6}x_1 x_2, \sqrt{3}x_1 x_2^2, \sqrt{3}x_2, \sqrt{3}x_2^2, x_2^3, 1)$

**Implementation assignment**

I. Implement the Naive Bayes classifier for categorical features with and without Laplace smoothing. Test your implementation on the provided data sets (train on the training data and test with the testing data sets).

**Data set information:** This data set is extracted from the UCI zoo data set. Note that there are 16 features (the first 16 columns) and the class labels are in the last column. There
are 7 classes (numerically specified as class 1 to 7). All features are binary except for feature 13, which is a categorical variable with possible values 0, 2, 4, 5, 6, 8.

**Need to report:**
1. Report the parameters that are learned by your classifiers (with and without Laplace smoothing respectively).
2. Report the test set accuracy for both cases.

II. Implement a fixed depth decision tree algorithm. In particular, the input to your algorithm will include the training data set and the maximum depth of the tree. For example, if the depth is set to one, you will learn a decision tree with one test node, which is also called a decision stump. Test your implementation, with depth=1, and 2 respectively, on the same data set as described above (train on the training data and test with the testing data set).

**Need to report:**
1. Report the learned decision tree (depth 1 and depth 2)
2. Report the test set accuracy for both trees.