# CS445 Machine Learning Midterm #2

Practice

Name: ______________________________

| Question | Points | Score |
|---|---|---|
| 1 | 15 | |
| 2 | 16 | |
| 3 | 10 | |
| 4 | 10 | |
| 5 | 14 | |
| Total: | 65 | |

*I have neither given nor received unauthorized assistance on this examination.*

______________________

1. **Ensemble Methods**

   We've discussed two general approaches to ensemble learning: *bagging* and *boosting*. Briefly describe the two approaches.

   (a) (5 points) Bagging

   (b) (5 points) Boosting

   (c) (5 points) Briefly describe the random forest algorithm. Does it make use of bagging, boosting, or neither?

2. **Naive Bayes Classifier**

Recall that the decision rule for a naive Bayes classifier can be expressed as follows:

$$\hat{y} = \underset{y \in Y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^{d} P(X_i|y)$$

Assume that we are developing a classifier to determine whether a child will go out to play ($y = yes$) under a given set of weather conditions. We have three attributes:

- $x_1 \in \{sunny, overcast, rainy\}$
- $x_2 \in \{hot, mild, cool\}$
- $x_3 \in \{high, normal\}$

We have the following priors and class-conditional probabilities:

| | |
|---|---|
| $P(y = no)$ | $\frac{3}{7}$ |
| $P(y = yes)$ | $\frac{4}{7}$ |

| | |
|---|---|
| $P(x_1 = sunny \mid y = no)$ | $\frac{2}{3}$ |
| $P(x_1 = overcast \mid y = no)$ | $\frac{0}{3}$ |
| $P(x_1 = rainy \mid y = no)$ | $\frac{1}{3}$ |
| $P(x_2 = hot \mid y = no)$ | $\frac{2}{3}$ |
| $P(x_2 = mild \mid y = no)$ | $\frac{0}{3}$ |
| $P(x_2 = cool \mid y = no)$ | $\frac{1}{3}$ |
| $P(x_3 = high \mid y = no)$ | $\frac{2}{3}$ |
| $P(x_3 = normal \mid y = no)$ | $\frac{1}{3}$ |

| | |
|---|---|
| $P(x_1 = sunny \mid y = yes)$ | $\frac{0}{4}$ |
| $P(x_1 = overcast \mid y = yes)$ | $\frac{2}{4}$ |
| $P(x_1 = rainy \mid y = yes)$ | $\frac{2}{4}$ |
| $P(x_2 = hot \mid y = yes)$ | $\frac{1}{4}$ |
| $P(x_2 = mild \mid y = yes)$ | $\frac{1}{4}$ |
| $P(x_2 = cool \mid y = yes)$ | $\frac{2}{4}$ |
| $P(x_3 = high \mid y = yes)$ | $\frac{2}{4}$ |
| $P(x_3 = normal \mid y = yes)$ | $\frac{2}{4}$ |

(a) (6 points) Use the naive Bayes classification algorithm to predict whether the child will go out to play given $x = \langle rainy, cool, high \rangle$. Show your work.

(b) (2 points) Based on the classifier results, what probability would we assign to the "no" prediction in this case?

(c) (4 points) What is the purpose of Laplace Smoothing? Was Laplace smoothing used to determine the class-conditional probabilities above? Justify your answer.

(d) (4 points) From an implementation standpoint, what problem arises when we attempt to apply the naive Bayes formula above to classifications tasks with many (e.g. hundreds) of attributes? How can we address this problem? For this question you should assume that we are using Laplace smoothing.

3. **Deriving Learning Rules**

We've discussed squared loss (L2 loss) and absolute value loss (L1 loss) for linear regression. Another option is quartic loss:

$$L(\boldsymbol{w}) = \frac{1}{4}(y - \boldsymbol{w}^T\boldsymbol{x})^4$$

(This is the error associated with a single training sample with input $\boldsymbol{x}$ and target value $y$.)

(a) (4 points) Find $\dfrac{\partial L(\boldsymbol{w})}{\partial w_i}$ for the loss function above. Show your work.

(b) (2 points) Use your answer to part (a) to develop a learning rule of the form $\boldsymbol{w_i} \leftarrow ??$

(c) (4 points) Compete the following function using Numpy operations so that it performs one step of batch gradient descent using your newly discovered learning rule:

```
def batch_gd(self, old_w, X, y, lr=.01):
     """Perform one epoch of batch gradient descent and update weights.

     Parameters:
         old_w - length-d numpy array representing the old weight vector.
         X - n x d numpy array, n points dimensionality d (with 1's in col 0)
         y - length-n numpy array of target values
         lr - learning rate
     Returns:
         The updated weight vector
     """
     # YOUR CODE HERE
```

4. **Logistic Regression**

As a reminder, the binary cross-entropy loss function can be expressed as follows:

$$-\mathcal{LL}(\mathbf{w}) = -\sum_{i=1}^{n} y_i \log(\sigma(\mathbf{w}^T\mathbf{x} + b)) + (1 - y_i)\log(1 - \sigma(\mathbf{w}^T\mathbf{x} + b))$$

where $\sigma$ represents the logistic function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

(a) (6 points) Consider a logistic regression classifier with the following weights and biases:

$$\mathbf{w} = \begin{bmatrix} .2 \\ 1.5 \end{bmatrix}, b = -1.2$$

Given these weights, what is the cross-entropy loss for the following input and label:

$$\mathbf{x} = \begin{bmatrix} -2.0 \\ 1.0 \end{bmatrix}, y = 0$$

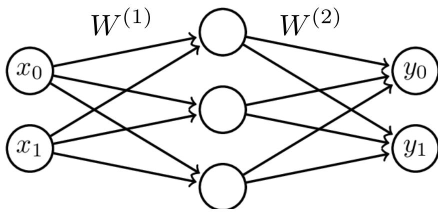
Show all steps of your calculation.

(b) (2 points) Would the classifier correctly classify this data point? Justify your answer.

(c) (2 points) One advantage of a logistic regression classifier is that the weights tell us something about the predictive usefulness of the corresponding attributes: a large positive weight indicates an attribute that is highly predictive of the positive class. However, this only really works if the attributes have been **normalized**: re-scaled to have zero mean and unit variance. Why is this?

5. **MLP**

(a) Consider the following three-layer neural network for a two-class classification problem:



In a three-layer neural network, the activations at the hidden layer can be expressed as:

$$a(\mathbf{x}) = h(\mathbf{x}^{\mathsf{T}} W^{(1)} + \mathbf{b}^{(1)})$$

where $\mathbf{x}$ is a column vector of input values, $W^{(1)}$ is a weight matrix, $h$ is the hidden-unit nonlinearity that is applied element-wise, and $\mathbf{b}^{(1)}$ represents the vector of bias values for the hidden units (not shown in the figure).

For this network, the activations at the output layer can then be represented as:

$$\sigma\left(a(\mathbf{x})W^{(2)} + \mathbf{b}^{(2)}\right)$$

Where $\sigma$ is the softmax function:

$$\sigma(\mathbf{a})_i = \frac{e^{a_i}}{\sum_{j=1}^{K} e^{a_j}}$$

i. (3 points) Assuming,

$$\mathbf{x} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad W^{(1)} = \begin{bmatrix} 1 & .2 & 0 \\ .5 & -1 & 3 \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} 1.5 & .5 & 1 \end{bmatrix}$$

What are the activations at the hidden layer <u>before</u> the nonlinearity has been applied? Show your work.

ii. (2 points) Assuming that the hidden layer uses a ReLU nonlinearities, what are the final activations at the hidden layer <u>after</u> the nonlinearity has been applied?

iii. (3 points) Assuming,

$$W^{(2)} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ .5 & 4 \end{bmatrix}, \quad \mathbf{b}^{(2)} = \begin{bmatrix} -1 & -5.7 \end{bmatrix}$$

what is the activation of the output units <u>before</u> the softmax nonlinearity has been applied? Show your work.

iv. (2 points) What is the activation at the output unit <u>after</u> the softmax nonlinearity has been applied? Assuming that the true one-hot encoded class label is $\mathbf{y} = \begin{bmatrix} 0 & 1 \end{bmatrix}$, does this network correctly classify this point?

(b) (4 points) Consider a fully-connected neural network with the following configuration:

- Layer 1 (Input) : 100 units
- Layer 2 (Hidden): 1000 units
- Layer 3 (Hidden): 100 units
- Layer 4 (Output): 4 units

How many overall weights does this network have, including bias weights? Show your work.

(c) What is the advantage of an MLP of the sort described in this question over simple logistic regression? In particular, are there problems that fundamentally can't be solved using logistic regression that can be solved using an MLP?