Homework 2
A and B due in class October 4
C due at 11:59pm October 8 (Monday night!)

A. Mathematics:

We wish to identify “antique” furniture using Bayesian classification: furniture is either “old” (>100 years) or “new” (<=100 years). We use five features - $x_1$ (Material), $x_2$ (Color), $x_3$ (Type), $x_4$ (Weight), and $x_5$ (Smell). Each feature takes on one of a discrete number of values, shown below:

<table>
<thead>
<tr>
<th>Material</th>
<th>Wood</th>
<th>Plastic</th>
<th>Metal</th>
<th>Leather</th>
<th>Fabric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>Black</td>
<td>Blue</td>
<td>Brown</td>
<td>White</td>
<td>Red</td>
</tr>
<tr>
<td>Type</td>
<td>Couch</td>
<td>Bed</td>
<td>Chair</td>
<td>Ottoman</td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>Heavy</td>
<td>Light</td>
<td>MidWeight</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Smell</td>
<td>Mild</td>
<td>Strong</td>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We classify each user as either $y_i=$Old or $y_i=$New. Based on a large training set, we wish to estimate all joint probability likelihoods, e.g.,
P($x_1=$Metal, $x_2=$Blue, $x_3=$Couch, $x_4=$Light, $x_5=$None $|$ y=New),
P($x_1=$Fabric, $x_2=$Red, $x_3=$Ottoman, $x_4=$Heavy, $x_5=$None $|$ y=Old).

1. Assuming all five features are fully dependent (conditional on the class assignment Y), how many total parameters need to be estimated for the likelihood-based classification?

2. Assuming all five features are independent (conditional on the class assignment Y, Naïve assumption), how many total parameters need to be estimated for the likelihood-based classification?

3. Let us assume we alter our problem so we have four classes of subject: $y=$Old, $y=$Middle-Aged, $y=$New. How many parameters are required to learn a non-Naïve (non-independent features) Bayes posterior classifier? For this answer, include parameters for the class prior probabilities. (Due to an earlier typo, I will accept your answer for 3 or 4 classes.)

4. Identify two or more features that can be transformed to use a logistic classifier. Explain how one of these features can be converted from discrete values to positions on a continuous number line.
5. What are the magnitudes of each vector? Which of the following vectors has (approximately) unit magnitude? (At most 1.1, at least 0.9.)

(a) \[
\begin{bmatrix}
-0.4 \\
0 \\
0.1 \\
-0.5 \\
0.7 
\end{bmatrix}
\]
(b) \[
\begin{bmatrix}
0 \\
-1 \\
0 \\
1 \\
1 
\end{bmatrix}
\]
(c) \[
\begin{bmatrix}
3 \\
0 \\
2 \\
4 \\
0 
\end{bmatrix}
\]
(d) \[
\begin{bmatrix}
0 \\
-0.8 \\
0.3 \\
0.5 \\
-0.1 
\end{bmatrix}
\]

6. What is the dot product of the vectors below onto the vector (c) from question 5?

(a) \[
\begin{bmatrix}
2 \\
0 \\
1 \\
-4 \\
0 
\end{bmatrix}
\]
(b) \[
\begin{bmatrix}
-0.3 \\
0.2 \\
0 \\
0 \\
1 
\end{bmatrix}
\]
(c) \[
\begin{bmatrix}
0 \\
0 \\
0 \\
2 \\
-3 
\end{bmatrix}
\]

7. Let us assume we have a classifier with \( \mathbf{w} = \begin{bmatrix} -1 \\ 6 \\ 4 \end{bmatrix} \) and \( b = 2 \). Provide two distinct inputs \( \mathbf{x} \) that would be on the boundary between class 0 and 1, specifically find two inputs \( \mathbf{x} \) where \( \mathbf{w}^\top \mathbf{x} = 0 \). (I intended for you to provide answers such that \( \mathbf{w}^\top \mathbf{x} + b = 0 \); I will accept answers based on either \( \mathbf{w}^\top \mathbf{x} + b = 0 \) or \( \mathbf{w}^\top \mathbf{x} = 0 \) without incorporating \( b \).)

8. In class, we define the logistic/sigmoid function as \( g(h) = \frac{1}{1+e^{-h}} \), which converts dot products from the range \((-\infty, +\infty)\) to pseudo-probabilities in the range \((0,1)\). Let us say that \( \mathbf{w} \) has been learned to properly separate the training data so that all data in class 0 satisfies \( \mathbf{w}^\top \mathbf{x} + b < 0 \) and all data in class 1 satisfies \( \mathbf{w}^\top \mathbf{x} + b > 0 \). Let us also say that we will only accept a data point as class 0 if \( g(\mathbf{x}^i; \mathbf{w}) < 0.2 \) and as class 1 if \( g(\mathbf{x}^i; \mathbf{w}) > 0.8 \). We find 10% of our class-0 and class-1 data produce sigmoid outputs that are between 0.2 and 0.8 --- they will not be properly classified into either class. What adjustments can be made to \( \mathbf{w} \) and \( b \) to ensure all data is properly classified --- forcing class 0 sigmoid outputs to fall below 0.2 and class 1 sigmoid outputs to rise above 0.8.

9. Which hyper-parameter controls the strength of the L1 prior assumption in the gradient ascent \( w_j \) update rule learned in class?
10. Consider the following function:

\[ f_j(x) = \frac{e^{x_j}}{\sum_k e^{x_k}} \]

which takes in a vector \( x \). It is used to determine the strength of element \( j \) \( (x_j) \) compared to all other elements in the vector \( x \) \( (x_k \text{ where } j \neq k) \).

(a) What is \( f_2(x) \) if \( x = \begin{bmatrix} 1 \\ 4 \\ 1 \end{bmatrix} \)?

(b) What is \( f_2(x) \) if \( x = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} \)?

(c) What is the largest possible value for \( f_2(x) \) to take? Provide a vector that will achieve this maximal value for \( f_2(x) \).

(d) What is the derivative of \( f_j(x) \) with respect to \( x_j \) ?

B. SVMs

1. (a) Let us use SVM to define a linear classifier with the following support vectors and \( \alpha \)'s:

\[
\begin{align*}
x^1 &= \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix}, & x^2 &= \begin{bmatrix} -4 \\ 1 \\ 0 \end{bmatrix}, & x^3 &= \begin{bmatrix} 0 \\ -2 \\ 4 \end{bmatrix}, & x^4 &= \begin{bmatrix} 0.5 \\ 1 \\ -2 \end{bmatrix}, & x^5 &= \begin{bmatrix} -4 \\ 2 \\ 3 \end{bmatrix}, & x^6 &= \begin{bmatrix} -2 \\ 0 \\ -0.5 \end{bmatrix} \\
y^1 &= +1, & y^2 &= +1, & y^3 &= +1, & y^4 &= -1, & y^5 &= -1, & y^6 &= -1 \\
\alpha^1 &= 0.6, & \alpha^2 &= 0.8, & \alpha^3 &= 1, & \alpha^4 &= 1.5, & \alpha^5 &= 0.3, & \alpha^6 &= 0.6
\end{align*}
\]

What is the resulting \( w \)?
(b) Consider the same support vectors as before but a different set of $\alpha$'s:
$\alpha^1 = 2.5$, $\alpha^2 = 1.5$, $\alpha^3 = 1$, $\alpha^4 = ?$, $\alpha^5 = 1$, $\alpha^6 = 2$

Presuming $x^1, x^2, x^3, x^4, x^5$, and $x^6$ are the only support vectors available, what value must $\alpha^4$ have?

What is the resulting $w$?

Consider the data and the three classification boundaries below.

![Graph showing three classification boundaries a, b, and c.]

2. Choosing from boundaries a, b, and c: Which boundary(ies) was/were most likely drawn by a Support Vector Machine (SVM)?

3. Select four data points likely to be a support vectors for the SVM? (Use your intuition, don’t worry about exact math here.)
C. Programming – due by 11:59pm on Monday October 8

In this question you will implement algorithms for learning parameters and classifying with Logistic Classification. We will use the “famous” Iris data set, available from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/datasets/Iris).

To submit Part C, create the directory HW2 inside your private/CIS5800 directory. Leave all relevant pieces of code in your private/CIS5800/HW2 directory. Matlab code for each function must be written in individually-named .m files (learnLogW.m for the learnLogW function). Python code for all functions must be written in the file hw2.py.

Accessing our data
The file hw2data.mat is available on our website (and on erdos using cp ~dleeds/MLpublic/hw2data.mat .) Load this file into your Matlab session to get access to the trainData and testData matrices. For each matrix, each row is one example flower. Columns 1 through 4 represent four features. The last column (column 5) represents the flower-type class $y^i$ – 1 (the Setosa Iris), 2 (the Versicolour Iris), or 3 (Virginica iris).

We will learn to discriminate between class $y^i=1$ and $y^i=2$ (Setosa vs. Versicolour).

Your program is only responsible to learn and classify data in classes 1 and 2. You can write your program to either ignore data in class 3 or you can remove the class 3 data from the data set.

1. Write a function called mySigmoid that takes in the features $x^i$, and a weights vector $w$ and the offset $b$. The function returns the output of $\text{sig}(w^Tx+b)=\frac{1}{1+e^{-(w^Tx+b)}}$.

Specifically: you will be able to call the function as
```
sigOut=mySigmoid(x, w, b);
```
x is a vector containing n elements, w is a vector containing n elements, b is a single number, and $y$ is the corresponding output of the sigmoid function (a number between 0 and 1).

(You may NOT use the pre-defined sigmoid/logistic function in Matlab and Python toolboxes. However, you CAN use the dot multiplication function defined in Matlab and Python.)
2. Write a function called `learnLogW` that takes in the initial weight vector $w^0$, the training data $x$ and their labels $y$, and the number of learning loops $K$. The function outputs the new weights $w$. Assume the step size is 0.01.

**Note:** For each “loop,” learnLogisticWeights will loop through each data point in the training set and use gradient ascent to update $w$ for each data point. Use 1 loop, learnLogW will visit each data point to update $w$, then visit each data point a second time to perform a second update, then a third time, ... up to an $k$th time.

It will follow a structure:

**Python:**
```python
for dataPoint in data :
    for feature in dataPoint:
        UPDATE w
```

**Or**

**Matlab:**
```matlab
for i=1:numDataPoints,
    for j = 1:numFeats,
        UPDATE w
    end;
end;
```

Specifically: you will be able to call the function as
```
finalW=learnLogW (w0, x, y);
```

`w0` is that initial guessed values $w$ with b at the end: $[w_1, w_2, ..., w_n, b]$. $x$ is a matrix where each row is a data point (a company) and there are $n$ columns corresponding to $n$ features. $y$ is a vector of 1's and 2's corresponding to the class of each data point (flower); if there are $m$ rows in $x$, there are $m$ entries in $y$. finalW will be a vector with $n+1$ elements containing the parameters $[w_1, w_2, ..., w_n, b]$ after a single loop of gradient ascent across all training data input into $x$.

In order to make the update function from class work, you will have to treat classes 1 and 2 as if they were classes 0 and 1. At the start of your function, you could just execute something like: $y( :) = y( :) - 1$

3. Write a function `logClassify` that takes in the feature values for multiple data points and the parameters $w$ and $b$ for the separator plane. The function returns the 1 or 2 label for each data point.

Specifically: you will be able to call the function as
```
classLabels=logClassify(x, w);
```
w is that parameters w with b at the end: \([w_1, w_2, ..., w_n, b] \). x is a matrix where each row is a data point (a company) and there are n columns corresponding to n features. classLabels is a vector of 1’s and 2’s corresponding to the class of each data point (flower) based on the logistic classifier.

4. Write a function \texttt{logTest} that finds and reports the testing-set accuracy of your classifier for a given set of parameters \(w\).

Some partial code you can use:
Python:

\[
\text{for dataPoint in data :} \\
\quad \text{if logClassify(dataPoint,w) is ??? # how do you code this?} \\
\quad \text{correctCount = correctCount+1}
\]

Or

Matlab:

\[
\text{for i=1:numDataPoints,} \\
\quad \text{if logClassify(x(i,:),w) is ??? # how do you code this?} \\
\quad \text{correctCount = correctCount+1;}
\]

Specifically: you will be able to call the function as 
\[
\text{accuracy=logTest(x,w,y)};
\]

w is that parameters w with b at the end: \([w_1, w_2, ..., w_n, b] \). x is a matrix where each row is a data point (a flower) and there are n columns corresponding to n features. y is a vector of 1’s and 2’s corresponding to the true class of each data point (flower). accuracy is a single number indicating the fraction of correctly-labeled data points in the input x, y.

5. Write a function \texttt{smartLoop} that repeated calls \texttt{learnLogW} until overfitting occurs (until the testing error increases from loop k to loop k+1)

Specifically: you will be able to call the function as 
\[
\text{finalW=smartLoop(xTrain,xTest,yTrain,yTest);} \\
\]

w is that parameters w with b at the end: \([w_1, w_2, ..., w_n, b] \). xTrain and xTest are matrices where each row is a data point (a flower) and there are n columns corresponding to n features. yTrain and yTest are a vectors of 1’s and 2’s corresponding to the true class of each data point (flower). finalW will be a vector with n+2 elements containing the fully learned w and b parameters, followed by the number of iterations completed \([w_1, w_2, ..., w_n, b, k] \).