Homework 2
A and B due in class October 4
C due at 11:59 pm 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>Feature</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Material</td>
<td>Wood, Plastic, Metal, Leather, Fabric</td>
</tr>
<tr>
<td>Color</td>
<td>Black, Blue, Brown, White, Red, Yellow</td>
</tr>
<tr>
<td>Type</td>
<td>Couch, Bed, Chair, Ottoman</td>
</tr>
<tr>
<td>Weight</td>
<td>Heavy, Light, MidWeight</td>
</tr>
<tr>
<td>Smell</td>
<td>Mild, Strong, None</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., $P(x_1 = \text{Metal}, x_2 = \text{Blue}, x_3 = \text{Couch}, x_4 = \text{Light}, x_5 = \text{None} | y = \text{New})$, $P(x_1 = \text{Fabric}, x_2 = \text{Red}, x_3 = \text{Ottoman}, x_4 = \text{Heavy}, x_5 = \text{None} | y = \text{Old})$.

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?
   $2 \times (5 \times 6 \times 4 \times 3 \times 3 - 1) = 2158$

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?
   $2 \times ((5-1)+(6-1)+(4-1)+(3-1)+(3-1)) = 2 \times 16 = 32$

3. Let us assume we alter our problem so we have four three 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.)
   Four classes: $4 \times (5 \times 6 \times 4 \times 3 \times 3 - 1) + (4-1) = 4316 + 3 = 4319$
   Three classes: $3 \times (5 \times 6 \times 4 \times 3 \times 3 - 1) + (3-1) = 3237 + 2 = 3239$
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.

Weight and smell can be converted to continuous number line and used for logistic classifier. For example, weight can be -1 for light, 0 for mid-weight, 1 for heavy; smell could be 0 for none, 1 for mild, 2 for strong. All other features can be converted into multiple binary features. For example, type could be inverted into Type_couch=0 or 1, Type_bed=0 or 1, Type_chair=0 or 1, Type_ottoman=0 or 1.

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) \[
\begin{bmatrix}
0 \\
-0.8 \\
0.3 \\
0.5 \\
-0.1
\end{bmatrix}
\]

(a) 0.91 0.95 (b) 3 1.7 (c) 29 5.4 (d) 0.99

Vector (a) and (d) has near-unit magnitude.

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

(a) \[
\begin{bmatrix}
2 \\
0 \\
-4 \\
0
\end{bmatrix}
\]

(b) \[
\begin{bmatrix}
0.2 \\
0 \\
0 \\
1
\end{bmatrix}
\]

(c) \[
\begin{bmatrix}
0 \\
0 \\
2 \\
-3
\end{bmatrix}
\]

(a) 6+0+16+0 = 24 (b) -0.9+0+0+0+0 = -0.9 (c) 0+0+0+0+0 = 0

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

\( w^Tx+b \) Interpretation:

Examples: \[
\begin{bmatrix}
8 \\
6 \\
0
\end{bmatrix}
\]

\( w^Tx \) Interpretation:

Examples: \[
\begin{bmatrix}
8 \\
6 \\
4
\end{bmatrix}
\]

CORRECTED Oct 14, 10pm
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 \( w \) has been learned to properly separate the training data so that all data in class 0 satisfies \( w^T x + b < 0 \) and all data in class 1 satisfies \( w^T x + b > 0 \). Let us also say that we will only accept a data point as class 0 if \( g(x^i; w) < 0.2 \) and as class 1 if \( g(x^i; w) > 0.8 \).

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 \( w \) 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

We can multiply all elements of the \( w \) vector by a large positive constant \( k \). For example, multiply all values of \( w \) by 100. This will push small positive dot products to be larger positive dot products --- the resulting sigmoid output will be close to 1. It will also push small negative dot products to be large negative dot products --- the resulting sigmoid output will be close to 0.

9. Which hyper-parameter controls the strength of the L1 prior assumption in the gradient ascent \( w_j \) update rule learned in class? \( \lambda \)
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 \( x \) \((x_k) \) where \( j \neq k \).

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

*If we interpret denominator to sum ALL \( x_k \) including \( x_j \):*

- If \( x_2 \) is 2\(^{nd} \) feature: \( 0.9 \)

*If we interpret denominator to sum every \( x_k \) EXCEPT \( x_j \):*

- If \( x_2 \) is 2\(^{nd} \) feature: \( 9.8 \)

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

*If we interpret denominator to sum ALL \( x_k \) including \( x_j \):*

- If \( x_2 \) is 2\(^{nd} \) feature: \( 0.4 \)

*If we interpret denominator to sum every \( x_k \) EXCEPT \( x_j \):*

- If \( x_2 \) is 2\(^{nd} \) feature: \( 0.7 \)

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

Actually, there is no truly maximal input.

*If we interpret denominator to sum ALL \( x_k \) including \( x_j \):*
1. A vector that will come CLOSE to 1 is: \[
\begin{bmatrix}
-10 \\ 2 \\ -10
\end{bmatrix}
\]

If we interpret denominator to sum every \(x_k\) EXCEPT \(x_j\):

No maximal value – it will approach infinite. A vector that will have a very high value is: \[
\begin{bmatrix}
-100 \\ 100 \\ -100
\end{bmatrix}
\]
make high value as high as possible, low values as low as possible

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

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

We actually never learned the technique to take the derivative of one function \(f(x)\) divided by another \(g(x)\) in class. I meant for you to take the derivative of the log, but I did not specify this.

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

\[
d/d x_j = 1 - \frac{e^{x_j}}{\sum_k e^{x_k}}
\]

If we interpret denominator to sum ALL \(x_k\) including \(x_j\):

\[
d \log f_j(x) \ / d x_j = 1 - \frac{e^{x_j}}{\sum_k e^{x_k}} = 1 - f_j(x)
\]

If we interpret denominator to sum every \(x_k\) EXCEPT \(x_j\):

\[
d \log f_j(x) / d x_j = 1
\]
B. SVMs

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

\[
\begin{align*}
    x^1 &= \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix},
    x^2 &= \begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix},
    x^3 &= \begin{bmatrix} -2 \\ 0 \\ 4 \end{bmatrix},
    x^4 &= \begin{bmatrix} 0.5 \\ 1 \\ -2 \end{bmatrix},
    x^5 &= \begin{bmatrix} -4 \\ 2 \\ 0 \end{bmatrix},
    x^6 &= \begin{bmatrix} -2 \\ 3 \\ -0.5 \end{bmatrix},
\end{align*}
\]

\[
\begin{align*}
    y^1 &= +1,
    y^2 &= +1,
    y^3 &= +1,
    y^4 &= -1,
    y^5 &= -1,
    y^6 &= -1,
\end{align*}
\]

$\alpha^1 = 0.6$, $\alpha^2 = 0.8$, $\alpha^3 = 1$, $\alpha^4 = 1.5$, $\alpha^5 = 0.3$, $\alpha^6 = 0.6$

What is the resulting $w$?

\[
\begin{align*}
    0.6 \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix} + 0.8 \begin{bmatrix} -4 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \\ 0 \end{bmatrix} - 1.5 \begin{bmatrix} 0.5 \\ 1 \\ -2 \end{bmatrix} - 0.3 \begin{bmatrix} -4 \\ 2 \\ 0 \end{bmatrix} - 0.6 \begin{bmatrix} -2 \\ 3 \\ -0.5 \end{bmatrix}
    &= \begin{bmatrix} -0.6 - 3.2 + 0 - 0.75 + 1.2 + 1.2 \\ -1.8 + 0.8 - 2 - 1.5 - 0.6 - 1.8 \\ -1.2 + 0 + 4 - 4.5 - 0.9 - 1.2 \end{bmatrix}
    &= \begin{bmatrix} -2.1 \\ -6.9 \\ -3.8 \end{bmatrix}
\end{align*}
\]

(b) Consider the same support vectors as before but a different set of $\alpha$'s:

$\alpha^1 = 2.5$, $\alpha^2 = 1.5$, $\alpha^3 = 1$, $\alpha^4 = 2$, $\alpha^5 = 1$, $\alpha^6 = 2$

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?

\[
\sum_{} \alpha^i y^i = 0
\]

\[
2.5 + 1.5 + 1 - \alpha^4 - 1.2 = 0 \rightarrow \alpha^4 = 2
\]

What is the resulting $w$?

\[
\begin{align*}
    2.5 \begin{bmatrix} -1 \\ 0 \\ -2 \end{bmatrix} + 1.5 \begin{bmatrix} -4 \\ 2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -2 \\ 4 \end{bmatrix} - 2 \begin{bmatrix} 0.5 \\ 1 \\ -2 \end{bmatrix} - \begin{bmatrix} -4 \\ 2 \\ 0 \end{bmatrix} - 2 \begin{bmatrix} -2 \\ 3 \\ -0.5 \end{bmatrix}
    &= \begin{bmatrix} -2.5 - 6 + 0 - 1 + 4 + 4 \\ -7.5 + 1.5 - 2 - 2 - 2 - 6 \\ -5 + 0 + 4 - 6 - 3 - 4 \end{bmatrix}
    &= \begin{bmatrix} -1.5 \\ -18 \\ -14 \end{bmatrix}
\end{align*}
\]

Consider the data and the three classification boundaries below.
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.)

Best choices in solid red, decent choices in dotted red. (Updated Thursday night!)

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' – 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).

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 $\text{sig}(w^T x + b) = \frac{1}{1+e^{-(w^T x + b)}}$.

Specifically: you will be able to call the function as
\[
\text{sigOut} = \text{mySigmoid}(x, w, b);
\]
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.)

```matlab
function out = mySigmoid(x,w,b)

h=0;
for i=1:length(w)
    h = h+ x(i)*w(i);
end;
out = 1/(1+exp(-h+b));
```

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 Using 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:
```
for dataPoint in data :
    for feature in dataPoint:
        UPDATE w
```

Or

Matlab:
```
for i=1:numDataPoints,
    for j = 1:numFeats,
        UPDATE w
    end;
end;
```
Specifically: you will be able to call the function as
\[ \text{finalW}=\text{learnLogW}(w_0, x, y); \]
w0 is that initial guessed values w with b at the end: \[ [w_1, w_2, \ldots, w_n, b] \]. 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 \[ [w_1, w_2, \ldots, w_n, b] \] after a single loop of gradient ascent across all training data input into x.

\begin{verbatim}
function finalW=learnLogW(w0,x,y)
y=y-1;
numDataPts=size(x,1);
for i=1:numDataPts
    sigOut = mySigmoid(x(i,:),w(1:end-1),w(end));
    for j=1:(length(w)-1)
        w(j) = w(j) + 0.01*x(i,j)*(y(i)-sigOut);
    end;
    w(end) = w(end) + 0.01*(y(i)-sigOut);
end;
end;
\end{verbatim}

3. Write a function \textit{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
\[ \text{classLabels}=\text{logClassify}(x, w); \]
w is that parameters w with b at the end: \[ [w_1, w_2, \ldots, w_n, b] \]. 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.

\begin{verbatim}
function classLabels=logClassify(x,w)
numDataPts=size(x,1);
for i=1:numDataPts
    if (mySigmoid(x(i,:),w(1:end-1),w(end))>0.5)
        classLabels(i)=2;
    else
        classLabels(i)=1;
    end;
end;
\end{verbatim}
4. Write a function \textbf{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:

```python
for dataPoint in data :
    if logClassify(dataPoint,w) is ??? # how do you code this?
        correctCount = correctCount+1
```

Or

Matlab:

```matlab
for i=1:numDataPoints,
    if logClassify(x(i,:),w) is ??? # how do you code this?
        correctCount = correctCount+1;
    end;
end;
```

Specifically: you will be able to call the function as

```matlab
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 \( x, y \).

```matlab
function accuracy=logTest(x,w,y)

    numDataPoints=size(x,1);
    numCorrect=0;
    for i=1:numDataPoints
        if (y(i) == logClassify(x(i,:),w))
            numCorrect = numCorrect+1;
        end;
    end;

    accuracy = numCorrect/numDataPoints;
```

5. Write a function \textbf{smartLoop} that repeated calls 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

```matlab
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 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].

function finalW=smartLoop(xTrain,xTest,yTrain,yTest)

w = [0 0 0 0 0];
oldAcc=-1; newAcc=-0.5;
k=1;

while (newAcc>oldAcc)
    oldAcc=newAcc;
    w=logLearnW(w,xTrain,yTrain);
    newAcc=logTest(xTest,w,yTest);
    k=k+1;
end;

finalW = [w, k];