Assignment 6
Student Name: Weining Hu
Student Number: 45606134

1 Multi-Class Logistic
1.1
1.3

1.1 Linear log-Odds Model

Given assumption 1, we have $\log\left(\frac{P(y_i=+1|w^Tx_i)}{P(y_i=-1|w^Tx_i)}\right) = w^Tx_i$

Apply 'exp' on both sides $\frac{P(y_i=+1|w^Tx_i)}{P(y_i=-1|w^Tx_i)} = e^{w^Tx_i}$ ...... ①

Using the fact that $P(y_i=+1|w^Tx_i) + P(y_i=-1|w^Tx_i) = 1$ ...... ②

Take ② into ① we now have $\frac{P(y_i=+1|w^TX)}{1-P(y_i=+1|w^TX)} = e^{w^TX}$

Then we would have $P(y_i=+1|w^TX) = \frac{e^{w^TX}}{1+e^{w^TX}}$, $P(y_i=-1|w^TX) = \frac{1}{1+e^{w^TX}}$

The loss function becomes $\arg\min_{w\in R^d} \sum_{i=1}^{n} -\log(P(y_i|w^Tx_i)) = \arg\min_{w\in R^d} -\log\left(\frac{e^{w^TX\cdot I(y_i=1)}}{1+e^{w^TX}}\right)$

1.3 Softmax Loss

$$P(y_i|W,x_i) = \frac{\exp(w_{y_i}^Tx_i)}{\sum_{c'=1}^{k}\exp(W_{c'}^Tx_i)}$$

The loss function: $J(w) = -\left[\sum_{i=1}^{n}\sum_{c'=1}^{k} I(y_i=k)\log\frac{\exp(w_{y_i}^Tx_i)}{\sum_{c'=1}^{k}\exp(W_{c'}^Tx_i)}\right]$

Take the gradient of loss function:

$$J(w) = -\left[\sum_{i=1}^{n}\sum_{c'=1}^{k} I(y_i=k)\left(\log(\exp(w_{y_i}^Tx_i)) - \log\left(\sum_{c'=1}^{k}\exp(W_{c'}^Tx_i)\right)\right)\right]$$

$$= -\left[\sum_{i=1}^{n}\sum_{c'=1}^{k} I(y_i=k)\left(w_{y_i}^Tx_i - \log\left(\sum_{c'=1}^{k}\exp(W_{c'}^Tx_i)\right)\right)\right]$$

$$\frac{\partial J(w)}{\partial W_{cj}} = -\sum_{i=1}^{n}\left[x_i\left(I(y_i=k) - \frac{\exp(W_{cj}^Tx_i)}{\sum_{c'=1}^{k}\exp(W_{c'}^Tx_i)}\right)\right]$$

1.2 One-vs-all Logistic Regression

Code:

```
function [model] = logLinearClassifier(X,y)
% Classification using one-vs-all least squares

% Compute sizes
[n,d] = size(X);
k = max(y);

W = zeros(d,k); % Each column is a classifier
for c = 1:k
    yc = ones(n,1); % Treat class 'c' as (+1)
    yc(y ~= c) = -1; % Treat other classes as (-1)
    W(:,c) = findMin(@logisticLoss,zeros(d,1),400,1,X,yc);
end
```

```
model.W = W;
model.predict = @predict;
end

function [yhat] = predict(model,X)
W = model.W;
   [~,yhat] = max(X*W,[],2);
end

function [f, g] = logisticLoss(w,X,y)
yXw = y.*(X*w);
f = sum(log(1 + exp(-yXw))); % Function value
g = -X'*(y./(1+exp(yXw))); % Gradient
end
```

Validation error: errors =0.0700

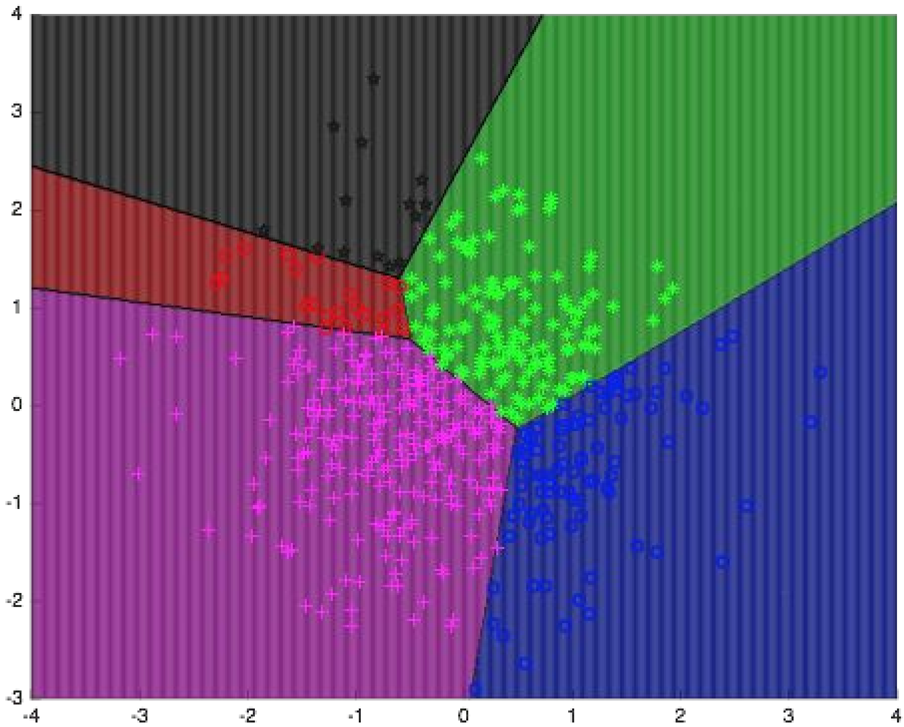
1.4Softmax Classifier

code

```
function [model] = softmaxClassifier(X,y)
% Classification using one-vs-all least squares

% Compute sizes
[n,p] = size(X);
k = max(y);

W = zeros(p,k); % Each column is a classifier
maxFunEvals=400;
verbose=1;
W(:) = findMin(@softLoss,W(:),maxFunEvals,verbose,X,y,k);

model.W = W;
model.predict = @predict;
end

function [f,g]=softLoss(w,X,y,k)
[n,d] = size(X);
```

```
W = reshape(w, [d k]);

f=sum(-sum(X.*W(:,y).',2)+log(sum(exp(X*W),2)));

g = zeros(d,k);
for c = 1:k
    for j =1:d
        gval=0;
        for i=1:n
            if (y(i)==c)
                indi=1;
            else indi=0;
            end
            minus=-X(i,j)*indi;
            den=sum(exp(X*W),2);
            nom=exp(X(i,:)*W(:,c))*X(i,j);
            gval=gval+sum(minus+nom/den(i));
        end
        g(j,c)=gval;
    end
end
g = reshape(g, [d*k 1]);

end

function [yhat] = predict(model,X)
W = model.W;
    [~,yhat] = max(X*W,[],2);
end
```

errors =0.0240



1.5
what is the cost of training the softmax classier? What is the cost of classifying the test examples?
Cost of training the softmax classifier: **O(T(nk+nkd+kd))**

Compute loss: O(nk), n examples of k classes
Compute gradient: O(nkd), n examples of k classes, d features
Update w(j,k) using softmax_gradient(j,k): O(kd), k class and d features to update
Since we have T iteration, cost of training the softmax classifier is O(T(nk+nkd+kd))

Cost of classifying the test examples: **O(tkd)**

Classify one test example: O(kd), k classes, each has d features.
Since we have t test examples, cost of classifying the test examples: O(tkd)

2 Random walk

code:

```
function [label] = runRandomWalk(A,labelList,v)

n = length(A);
labels = zeros(n, 1);
% Copy the initial labels from labellist
for i=1:length(labelList)
   labels(labelList(i, 1)) = labelList(i, 2);
end
e = zeros(n, 1);
num = 0;
while 1
   k = 0;
   % Store all the neighbor nodes
   for i = 1 : n
      % If meet a node that is connected? 1 in the adjacency matrix
      if A(v, i) == 1
         k = k + 1;
         e(k) = i;
      end
   end
   % if you meet a node that has label, either 1 or -1
   if labels(v) ~= 0
      num = randi(k + 1);
      % If you happened to pick the 'node' with the label
      if num == k + 1
         label = labels(v);
         return;
      end
   else
      num = randi(k);
   end
   % Assign to the new node
   v = e(num);
end
```

probability:

```
probabilities =

   0.2400    0.7600
   0.1900    0.8100
   0.1900    0.8100
   0.3400    0.6600
   0.3200    0.6800
   0.2300    0.7700
```

0.2700	0.7300
0.2700	0.7300
0.2700	0.7300
0.4400	0.5600
0.2400	0.7600
0.6900	0.3100
0.7000	0.3000
0.7200	0.2800
0.7200	0.2800
0.7300	0.2700
0.7600	0.2400
0.8400	0.1600
0.7600	0.2400
0.7800	0.2200
0.7200	0.2800
0.7700	0.2300
0.7600	0.2400