CPSC 340 Assignment 4

Tristan Rice, q7w9a, 25886145

1 Logistic Regression with Sparse Regularization

1.1 L2-Regularization

```
function [model] = logRegL2(X,y,lambda)

[n,d] = size(X);

maxFunEvals = 400; % Maximum number of evaluations of objective
verbose = 1; % Whether or not to display progress of algorithm
w0 = zeros(d,1);
model.w = findMin(@logisticLossL2,w0,maxFunEvals,verbose,X,y,lambda);
model.predict = @(model,X)sign(X*model.w); % Predictions by taking sign
end


function [f,g] = logisticLossL2(w,X,y,lambda)
yXw = y.*(X*w);
% Add an L2 regularizer.
f = sum(log(1 + exp(-yXw))) + w'*w*lambda/2; % Function value
g = -X'*(y./(1+exp(yXw))); % Gradient
end
```

```
At maximum number of function evaluations
numberOfNonZero =  101
trainingError = 0
validationError =  0.086000
```

1.2 L1-Regularization

```
function [model] = logRegL1(X,y,lambda)

[n,d] = size(X);

maxFunEvals = 400; % Maximum number of evaluations of objective
verbose = 1; % Whether or not to display progress of algorithm
w0 = zeros(d,1);
model.w = findMinL1(@logisticLoss,w0,lambda, maxFunEvals,verbose,X,y);
model.predict = @(model,X)sign(X*model.w); % Predictions by taking sign
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
```

```
Problem solved up to optimality tolerance
numberOfNonZero =  71
trainingError = 0
validationError =  0.052000
```

1.3 L0-Regularization

```
function [model] = logRegL0(X,y,lambda)

[n,d] = size(X);
maxFunEvals = 400; % Maximum number of evaluations of objective
verbose = 0; % Whether or not to display progress of algorithm
w0 = zeros(d,1);
oldScore = inf;

% Fit model with only 1 variable,
% and record 'score' which is the loss plus the regularizer
ind = 1;
w = findMin(@logisticLoss,w0(ind),maxFunEvals,verbose,X(:,ind),y);
score = logisticLoss(w,X(:,ind),y) + lambda*length(w);
minScore = score;
minInd = ind;

while minScore ~= oldScore
    oldScore = minScore;
    fprintf('\nCurrent set of selected variables (score = %f):',minScore);
    fprintf(' %d',ind);

    for i = 1:d
        if any(ind == i)
            % This variable has already been added
            continue;
        end

        % Fit the model with 'i' added to the features,
        % then compute the score and update the minScore/minInd
        ind_new = union(ind,i);

        w = findMin(@logisticLoss,w0(ind_new),maxFunEvals,verbose,X(:,ind_new),y);
        score = logisticLoss(w,X(:,ind_new),y) + lambda*length(w);
        if score < minScore
            minInd = ind_new;
            minScore = score;
        end
    end
    ind = minInd;
end

model.w = zeros(d,1);
model.w(minInd) = findMin(@logisticLoss,w0(minInd),maxFunEvals,verbose,X(:,minInd),y);
model.predict = @(model,X)sign(X*model.w); % Predictions by taking sign
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
```

```
nZero =  24
trainingError = 0
validationError =  0.018000
```

2 Convex Functions and MLE/MAP Loss Functions

2.1 Showing Convexity from Definitions

2.1.1 Quadratic

$$f(w) = aw^2 + bw$$
$$f'(w) = 2aw + b$$
$$f''(w) = 2a > 0$$

Since the second derivative is positive, $f(w)$ is convex.

2.1.2 Negative logarithm

$$f(w) = -\log(aw)$$
$$f'(w) = -\frac{a}{aw} = -\frac{1}{w}$$
$$f''(w) = \frac{1}{w^2}$$

Since $w > 0$, $f''(w) > 0$ and thus $f(w)$ is convex.

2.1.3 Regularized regression (arbitrary norms)

L1 norms are convex and the summation of two convex functions is also convex. $Xw - y$ is a linear function, and the composition of a convex function and a linear function is convex. Thus, the whole function is convex.

2.1.4 Logistic regression

2.1.5 Support vector regression

2.2 MAP Estimation

3 Multi-Class Logistic

3.1 One-vs-all Logistic Regression

```
function [model] = logLinearClassifier(X,y)
% Classification using one-vs-all with logistic loss.

% 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) = (X'*X)\(X'*yc);

    maxFunEvals = 400; % Maximum number of evaluations of objective
    verbose = 1; % Whether or not to display progress of algorithm
    w0 = zeros(d,1);
    W(:,c) = findMin(@logisticLoss,w0,maxFunEvals,verbose,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
```

```
errors =  0.070000
```

3.2 Softmax Classification

$$max_{y \in 1,2,3} p(y|W, \hat{x})$$

$$\sum_{c=1}^{k} \exp(w_c^T x_i)$$

is constant for all y_i, thus don't need to compute it. Just take the max of all the results.

$$p(1|W, \hat{w}) = \exp(\begin{bmatrix} +2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = \exp(2-1) = \exp(1)$$

$$p(2|W, \hat{w}) = \exp(\begin{bmatrix} +2 & +2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = \exp(2+2) = \exp(4)$$

$$p(3|W, \hat{w}) = \exp(\begin{bmatrix} +3 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}) = \exp(3-1) = \exp(2)$$

Since $p(2|W, \hat{w})$ is the highest, the class label would be 2.

3.3 Softmax Loss

3.4 Softmax Classifier