CPSC 340: Assignment4

Yiwei Hou(84435156)
Qiong Zhang(85896158)

1 Logistic Regression with Sparse Regularization

1.1 L2-Regularization

```
function [model] = logRegL2(X,y)
[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(@logisticLoss,w0,maxFunEvals,verbose,X,y);
model.predict = @(model,X)sign(X*model.w); % Predictions by taking sign
end

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

Number of non-zeros: 101, validation error is 0.0740 with $\lambda = 1$.

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
```

Number of non-zeros: 71, validation error is 0.0520 with $\lambda = 1$.

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;
            minScore = score;
            minInd=ind_new;
        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
```

Number of non-zeros: 23, validation error is 0.0380 with $\lambda = 1$.

2 Convex Functions and MLE/MAP Loss Functions

2.1 Showing Convexity from Definitions

1. $f(w) = aw^2 + bw$, then $f'(w) = 2aw + b$, $f''(w) = 2a > 0$, by the definition of convex function, quadratic function with $a > 0$ is convex.

2. $f(w) = -\log(aw)$, then $f'(w) = -w^{-1}$, $f''(w) = w^{-2} > 0$, by the definition of convex function, Negative logarithm is convex.

3. We already know norms are convex and the composition of a convex function and a linear function is convex, thus $||Xw - y||_p$ is convex. Since $||w||_q$ is convex, $\lambda \geq 0$, and a convex function multiplied by non-negative constant is convex, we have $\lambda||w||_q$ is convex. We also know that the sum of convex functions is a convex function, thus we have shown the objective function for regularized regression is convex.

4. $g(w) = \log(1 + \exp(-y_i w^T x_i))$ are convex. The sum of convex functions is convex, thus the objective function for logistic regression is convex.

5. $f(x) = |x|$ is a convex function by definition. The composition of a convex function and a linear function is convex, thus we have $|w^T x_i - y_i| - \epsilon$ are convex. The max of convex functions in convex, then $\max\{0, |w^T x_i - y_i| - \epsilon\}$ are convex. The sum of convex functions is convex, thus $\sum_{i=1}^n \max\{0, |w^T x_i - y_i| - \epsilon\}$ is convex. Similar as 3, $\frac{\lambda}{2}||w||_2^2$ is convex. Thus, we have the objective function for support vector regression is convex.

2.2 MAP Estimation

1. Maximize the posterior is equivalent to minimize the negative log of the posterior, thus we minimize $-\log p(w|x,y) = -\sum_{i=1}^n \log p(y_i|x_i,w) - \sum_{j=1}^d \log p(w_j) = \sum_{i=1}^n \frac{(y_i - w^T x_i)^2}{2} + \sum_{j=1}^d \lambda|w_j| + \text{constant}$, this lead to the standard L1-regularized least squares objective function $f(w) = \frac{1}{2}||Xw - y||^2 + \lambda||w||_1$

2. $-\log p(w|x,y) = -\sum_{i=1}^n \log p(y_i|x_i,w) - \sum_{j=1}^d \log p(w_j) = \sum_{i=1}^n |y_i - w^T x_i| + \frac{\lambda}{2}\sum_{j=1}^d w_j^2 + \text{constant}$, this lead to an objective function $f(w) = ||Xw - y||_1 + \frac{\lambda}{2}||w||^2$

3. $-\log p(w|x,y) = -\sum_{i=1}^n \log p(y_i|x_i,w) - \sum_{j=1}^d \log p(w_j) = \sum_{i=1}^n \frac{(y_i - w^T x_i)^2}{2\sigma^2} + \frac{\lambda}{2}\sum_{j=1}^d w_j^2 + \text{constant}$, this lead to an objective function $f(w) = \frac{1}{2\sigma^2}||Xw - y||^2 + \frac{\lambda}{2}||w||^2$

4. $-\log p(w|x,y) = -\sum_{i=1}^n \log p(y_i|x_i,w) - \sum_{j=1}^d \log p(w_j) = \sum_{i=1}^n \frac{(y_i - w^T x_i)^2}{2\sigma_i^2} + \frac{\lambda}{2}\sum_{j=1}^d w_j^2 + \text{constant}$, this lead to an objective function $f(w) = \sum_{i=1}^n \frac{(y_i - w^T x_i)^2}{2\sigma_i^2} + \frac{\lambda}{2}||w||^2$

5. $-\log p(w|x,y) = -\sum_{i=1}^n \log p(y_i|x_i,w) - \sum_{j=1}^d \log p(w_j) = -\frac{\nu+1}{2}\sum_{i=1}^n \log\left[1 + \frac{(y_i - w^T x_i)^2}{\nu}\right] + \frac{\lambda}{2}\sum_{j=1}^d w_j^2 + \text{constant}$, this lead to an objective function $f(w) = -\frac{\nu+1}{2}\sum_{i=1}^n \log\left[1 + \frac{(y_i - w^T x_i)^2}{\nu}\right] + \frac{\lambda}{2}||w||^2$. We say an objective for regression is robust when the function put less focus on large errors. In 5, even if $(y_i - w^T x_i)^2$ is huge, the objective function put less focus on thus terms by taking the logarithm of squared error, and thus makes the estimation of regression coefficients more robust.

3 Multi-Class Logistic

3.1 One-vs-all Logistic Regression

```
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)
    model=logReg(X,yc);
   W(:,c) = model.w;
end

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

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

The validation error is 0.07.

3.2 Softmax Classification

Since the probabilities of the three categories share the same denominator, we only need to consider the numerators. As the exponential function is strictly increasing, we have $\hat{x}W_1 = 1$, $\hat{x}W_2 = 4$, $\hat{x}W_3 = 2$, and $\hat{y} = \arg\max_i \hat{x}W_i = 2$

3.3 Softmax Loss

The loss funcion is given by

$$f(W) = \sum_{i=1}^{n}\left[-w_{y_i}^T x_i + \log\left(\sum_{c=1}^{k}\exp(w_c^T x_i)\right)\right]$$

Take derivative with respect to W_{jc}, we have

$$\frac{\partial f(W)}{\partial W_{jc}} = -\sum_{i=1}^{n} x_{ij} I(y_i = c) + \sum_{i=1}^{n}\frac{\exp(w_c^T x_i)x_{ij}}{\sum_{c'=1}^{k}\exp(w_{c'}^T x_i)}$$

3.4 Softmax Classifier

The validation error is 0.008.

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

% Compute sizes
[~,d] = size(X);
```

```
k=max(y);

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

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

function [f,g] = softmaxLoss(w,X,y)
k = max(y);
[~,d]=size(X);
W=reshape(w,d,k); %reshape by column
value=exp(X*W);
f = sum(diag(-X*W(:,y)))+sum(log(sum(value,2))); % Function value
g = zeros(d,k); % Gradient
for i = 1: d
    for j = 1:k
        denom = sum(value,2);
        num = exp(X*W(:,j));
        g(i,j)= -sum(X(:,i)'*(y==j))+sum(X(:,i).*num./denom);
    end
end
g = reshape(g,d*k,1);
end
```