Rutgers CS323 (04), Spring 2017, Homework 6

Due at 11:55pm on April 21, 2017, submitted via Sakai

L_2 Regularized Multi-Class Logistic Regression

1 Data

Denote the dataset by $\{x_i, y_i\}_{i=1}^n$, where x_i is a p-dimensional vector and $y_i \in \{0, 1, ..., K-1\}$ denotes the class label. The first example is the zipcode dataset. The following code displays a randomly selected subset of rows:

```
function DisplayZip

zip = feval('load', 'zip.train');
ind = randsample(size(zip,1),8);
figure;
for n = 1:length(ind);
    for i = 1:16;
        for j = 1:16;
            im(i,j) = zip(ind(n),1+(i-1)*16+j);
        end
    end
    subplot(2,4,n);
    imagesc(im);title(num2str(zip(ind(n),1)));axis off;
end
```

One execution generates the following plot:

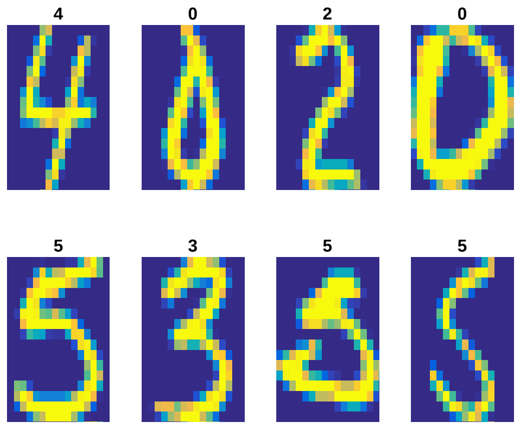

Figure 1:

2 Formulation

Following the lecture notes, L_2 regularized logistic regression aims to maximizing the following objective function:

$$l(\beta) = \sum_{i=1}^{n} \left\{ \sum_{k=0}^{K-1} r_{i,k} \log p_{i,k} \right\} - \frac{\lambda}{2} \sum_{k=0}^{K-1} \sum_{j=0}^{p} \beta_{k,j}^2$$

where $\lambda > 0$ and

$$r_{i,k} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise} \end{cases}$$

The Newton's updating formula is given in the lecture notes as

$$\beta_{\mathbf{k}}^{\mathbf{t}} = \beta_{\mathbf{k}}^{(\mathbf{t-1})} + \nu \left[\mathbf{X^T W_k^{(t-1)} X} + \lambda \mathbf{I} \right]^{\mathbf{-1}} \left[\mathbf{X^T} (\mathbf{r_k} - \mathbf{p_k^{(t-1)}}) - \lambda \beta_{\mathbf{k}}^{(\mathbf{t-1})} \right]$$

where:

- ν is a heuristic "shrinkage" factor, often $\nu = 0.1$ works well.
- β_k, $k = 0$ to $K-1$, is a weight vector of length $p+1$. Note that we always include the intercept $\beta_{k,0}$, i.e., we let $x_{i,0} = 1$. (Be careful that Matlab starts the index with 1 instead of 0.) In your implementation, β_k are initialized to be zero.
- p_k, $k = 0$ to $K-1$, is a probability vector of length n. Using the model $p_k = \frac{e^{x_i \beta_k}}{\sum_{s=0}^{K-1} e^{x_i \beta_s}}$ to compute p_k from β_k at each iteration.
- W_k is an $n \times n$ diagonal matrix $[p_{i,k} \times (1 - p_{i,k})]_{i=1}^{n}$. Note that naively materializing W_k may be too wasteful and the code may reach the memory limit on large datasets.

Note that this implementation slightly differs from the traditional method seen in textbooks. I believe this formulation is actually easier.

3 Coding Instructions

TestMultiLogit.m is written almost entirely for you.

```
function TestMultiLogit(v,Lam,numIter)

data = feval('load','zip.train');
X = data(:,2:end); Y = data(:,1)+1;  X = [X(:,1)*0+1 X];

data = feval('load','zip.test');
Xt = data(:,2:end); Yt = data(:,1)+1;  Xt = [Xt(:,1)*0+1 Xt];

m = 1:numIter;
for lam = Lam
    [err, loglik, err_t]  = FitMultiLogit(X,Y,Xt,Yt,numIter,v,lam);

    figure(1);
    plot(m,err*100,'linewidth',2); hold on; grid on; set(gca,'FontSize',20);
    xlabel('Iteration'); ylabel('Mis-Classification Error (%)');
    % write the corresponding  text at the end of the curve.
    axis([1 numIter 0 16]);
    title(['Train Mis-Classfication Error: \nu = ' num2str(v)]);

    figure(2);
    plot(m,loglik,'linewidth',2); hold on; grid on; set(gca,'FontSize',20);
    xlabel('Iteration'); ylabel('Log likelihood');
    % write the corresponding text at the end of the curve.
    axis([1 numIter -15000 1000]);
    title(['Train Log Likelihood: \nu = ' num2str(v)]);

    figure(3);
    plot(m,err_t*100,'linewidth',2); hold on; grid on; set(gca,'FontSize',20);
    xlabel('Iteration'); ylabel('Mis-Classification Error (%)');
    % write the corresponding text at the end of the curve.
    axis([1 numIter 5 30]);
    title(['Test Mis-Classfication Error: \nu = ' num2str(v)]);
end
```

You just need need to complete the code for writing, for example, $\lambda = 1$, at the end of the each curve, as shown in the figure. Note that, when plotting the log likelihood, you are asked not to include the regularization part. That is, you only plot $\sum_{i=1}^{n}\left\{\sum_{k=0}^{K-1} r_{i,k}\log p_{i,k}\right\}$, which will be returned from

```
[err, loglik, err_t]  = FitMultiLogit(X,Y,Xt,Yt,numIter,v,lam);
```

We will execute your code for any “numIter” value and “Lam” vector.

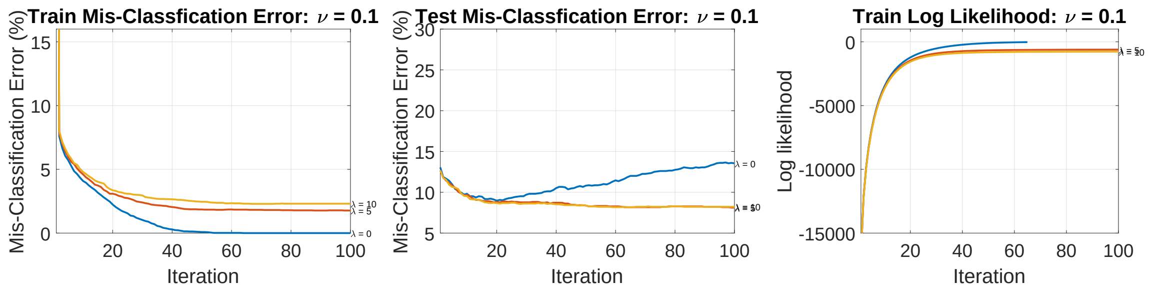

Figure 2: Fitting multi-class logistic regression on the *zipcode* dataset for $\nu = 0.1$ for 3 λ values: 0, 5, 10.

Efficiency requirement: Because we will run your code, there will be basic requirement for code efficiency. One iteration should not take more than 1/3 second. Nevertheless, we allow 1 second per iteration. That is, if we test your code for 3 λ values and 100 iterations, the total execution time should not exceed $1 \times 100 \times 3 = 300$ seconds. If your code is correct but it runs too slow, we will test your code on a smaller number of iterations and you will receive partial credits.

4 Submission

You should submit two .m files: FitMultiLogit.m and TestMultiLogit.m in one zipped file to Sakai.