CSCI567 Fall 2013 - Assignment 3
Due date: Oct. 18 , 2013

October 6, 2013

Algorithm Component

1 Logistic Regression

1.1 Question 1. (5 points)

MLaPP (the textbook) Exercise 8.6

1.2 Question 2. (6 points)

MLaPP (the textbook) Exercise 8.7

2 Bayesian approach: basic concepts

2.1 Question 3. (5 points)

MLaPP (the textbook) Exercise 3.4

2.2 Question 4. (5 points)

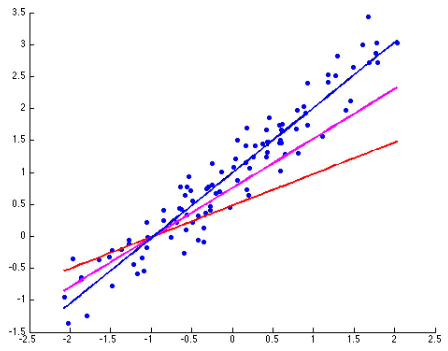
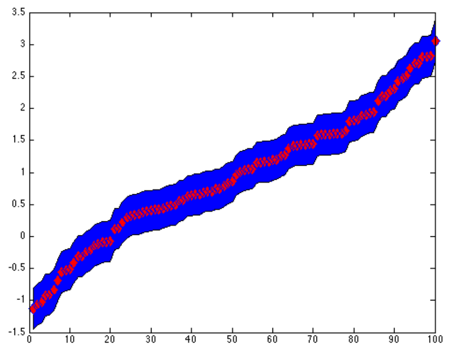
MLaPP (the textbook) Exercise 3.8

3 Bayesian linear regression

We have provided you with the file *experiment_ blr.m* that contains several functions, two of which with missing parts you should fill in:

- The function *computePosterior*() - computes the model's posterior (the posterior of w, after having seen the training data $\{X^{train}, y^{train}\}$)
- The function *predictiveDistribution*() - computes the predictive distribution.

The main function in that file, *experiment_ blr(),* contains a sequence of commands that create a mock dataset of $N = 100$ data points. A complete implementaion should produce figures similar (identical on Matlab R2011b) to those below [**hint**: you should first be able to reproduce these]:

	
N noised up samples + three models	Predictive distribution of the N points

Left hand side figure - The dots depict the randomly generated data (see $generateData()$). In red and magenta are two models produced by Bayesian linear regression each for a different *alpha* parameters. The blue line represents the Bayesian linear regression model, after estimating the β and α parameters from the data (see $estimateAlphaBetaViaEmpiricalBayes()$). Covered by the blue line is the ordinary linear regression model (in black).

Right hand side figure - The plot depicts the predicted value (red) of each data point, as well as an interval of one standard deviation around the it (blue).

3.1 Fill the blanks (5 points)

Study the code and the output of $experiment\_blr()$. Fill the blanks in the code. You will find that the Bayesian linear regression's output (blue line) covers the ordinary linear regression's output (black line). If you change $trueBeta$ in the code, would you see the difference between the two methods? Can you think of a reason to explain your observations? Please plot data and the predictions using both methods. Analyze the difference and give you explanation.

The following two questions ask you to use the provided dataset. Use the provided wine dataset to compare regularized linear regression (RLR) and Bayesian linear regression (BLR) as follows/

3.2 (2 Points)

Train the RLR model on the training set and find the regularizer $\lambda^{best} \in \{0, 0.5, 1, 1.5, 2\}$ that minimizes the development set average squared error. Evaluate and report the train/development/test average squared error as well as λ^{best}. (You will find your codes from Homework 2 are going to be handy.)

3.3 (10 Points)

Train the BLR model on **both** the training and development sets together.

1. First, we choose fixed pairs of hyperparameters α and β- try (α, β) in the following pairs

$$\{(50, 10), (50, 20), (20, 20), (10, 50), (10, 500), (30, 50), (30, 500)\}$$

   , use *computePosterior()* to compute model mean and variance. In regression use the output of *predictiveDistribution()*. Report train/test *average squared error* as well as the better pair of hyperparameters.

2. Second, let's try empirical Bayes way - use the *estimateAlphaBetaViaEmpiricalBayes()* function to estimate hyperparameters α and β and then do linear regression. Choose different initial value pairs of α and β as listed above. Report train/test average squared error. Compare the results with the previous ones.

Submission instruction

Your answers must be typeset with LaTeX and generated as a PDF file. No handwritten submission will be permitted. A template is provided for you. Please consult it for additional instructions. There are many free integrated LaTeXeditors that are convenient to use, please google search them and choose the one(s) you like the most. This `http://www.andy-roberts.net/writing/latex` seems like a good tutorial.

Due Date and Time

3:15pm, Friday, Oct. 18, 2013. You need to logon to Blackboard (for the time being) to submit your solutions. Your PDF file should be named as hw3Algorithm.pdf and packaged with other files, from the programming component in zip file called abcd_hw3_AB.zip where abcd is the last 4 digits of your USC ID and AB are the initials of your first and last names

A single two-day extension or two one-day extensions are allowed. You need to apply for it by Thursday Oct 17 2013, 3:15pm and get my approval. Please send your application to me via email. Late submissions will not be accepted if not previously approved. My email address is `feisha@usc.edu`

Collaboration

You may collaborate. However, you need to write your own solution and submit separately. You also need to list with whom you have discussed. For any team, at most 4 members can participate. You cannot participate more than 1 team.

Programming Component

4 Logistic regression

You are required to implement logistic regression to classify images of digits to their correct class labels. There are two approaches to this task - generative and discriminative. We will focus on the latter approach (the generative approach is left as **extra credit**).

For the discriminative approach you will implement a *binary* classifier, however, the underlying task is a multi-class classification problem with $C = 10$ classes. In order to perform multi-class classification, you will experiment with two techniques named *one-vs-one* and *one-vs-rest*. These techniques combine a set of binary predictions to form a final multi-class prediction.

4.1 Data

Handwritten digits data set is similar to the one in Assignment 2. The file *'HW3-USPS-split.mat'* contains a randomly partitioned and noised up version of the USPS dataset into training and test sets. Once you load the .mat file (using Matlab's 'load' command) your Matlab environment should contain two variables - X and y. The training, development and test sets are stored as $D\times$*(number of samples)* matrices in $X.train$ and $X.test$ with $D = 256$. The corresponding digit labels are similarly stored in y.

4.2 Implementation and Experiment

You should:

1. Implement a function *logreg_DISC* that returns the model parameters w. You should optimize the L_2-regularized logisitic regression function using regularization coefficient, chosen from cross-validation. Optimization should be done using the 2nd order technique discussed in the class slides (Newton's method). Note that the introduced regularizion term not only effects the objective function but also the computation of the gradient and Hessian matrix. You may also refer to wikipedia for details Newton's method.

   Your implementation should support a *shrinking step size* - Consider the following 2nd order update rule for w, performed at the tth iteration:

   $$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \gamma \boldsymbol{H}^{-1} \nabla \boldsymbol{f}$$

   where $\boldsymbol{H}$ and $\nabla \boldsymbol{f}$ are the Hessian and gradient of the objective function f, evaluated at w^{t-1}. Normally Newton's update is used with step size $\gamma = 1$, however, under some circumstances this may overshoot the location of the minimum. Selecting a small γ may help circumvent the problem. But which γ should we choose?

   A common solution is to search for an objective value shrinking γ by exponentation as follows:

   $$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \gamma^j \boldsymbol{H}^{-1} \nabla \boldsymbol{f}$$

   Specifically, at each iteration t, initialize $j = 0$, apply the modified rule and evaluate the objective function $f(\boldsymbol{w}^t)$. If the objective decreases, accept γ^j as the step size and move to the new $\boldsymbol{w}^t$. Otherwise increase j by 1 and repeat.

   It makes sense to bound the number of exponentations by some R (say $R = 15$), either because at the true minimum we don't expect the objective function can decrease, or because when j is large, the change to $\boldsymbol{w}$ is insignificant. In your implementation, use the starting $\gamma = \frac{1}{2}$.

   (a) Stopping conditions - We are minimizing a convex objective function (cross entropy). At each iteration, verify that the value of the objective function is non-increasing.

   You may stop the algorithm after either a fixed number of steps T (say $T = 50$) or once it seems the objective function stops decreasing - e.g., once $[f(w_{t-1}) - f(w_t)]/f(w_{t-1}) < \epsilon$ for some $\epsilon > 0$ (say $\epsilon = 10^{-5}$).

2. For multi-class classification, use two common techniques:

   (a) one-vs-one: For each $j, k \in \{1, ..., C\}$ where $j \neq k$, train a logistic regression model w_{jk} that discriminates between data-points from the classes C_j and C_k only. Then combine the nns of the $\binom{C}{2}$ classifiers to a single, multi-class prediction per data-point.

   (b) one-vs-rest: For each $j \in \{1, ..., C\}$, train a logisitic regression model w_j that discriminates between data-points from class C_j and data points from the remaining training sets. Combine the predictions of the C classifiers to a single, multi-class prediction per data-point.

   Note that for either approach, you should choose the regularization coefficients independently for each binary classifier. You should use the binary classification accuracy on holdout datasets to choose the coefficient.

Extra Credit: Generative approach with common covariance Gaussian class conditionals: Here, you should use your solution for the theoretical question 4 as well as read the class slides and the reference to Ng's note regarding Gaussian discriminative analysis.

Implement a function *logreg_ MLE* that returns $W = [w_1, \ldots, w_C]$, a matrix containing all the C hypothesis.

Here you first need to estimate a set of parameters $\{\pi_k, \mu_k\}_{k=1}^{C}$ and a shared covariance matrix Σ. These parameters are then used to form $\{w_k\}$.

At classification time, a data-point x is classified to $y = argmax_j p(j|x, w_j)$ by applying Bayes' rule .

This may seem complicated, however, notice that contrary to the discriminative approach, obtaining the solution W does not involve any numerical optimization - all the parameters being estimated have a closed form solution (thus we expect to produce relatively simple matlab code).

What to submit for Linear Classification

1. (30+10 points) **Report** - report your results *(HW3Programming.pdf)*.

   (a) (4 points) For one-vs-the-rest, plot the regularized cross entropy as a function of the iterations. Also list the number of iterations T or ϵ parameter you choose. Note that you should have 10 plots, one for each binary classifier. For each binary classifier, you only need to plot the curve with the optimal coefficient chosen from the cross-validation. Please also report the cross-validation coefficient you have chosen.

   (b) (10 points) Report the training/test accuracy for one-vs-one and one-vs-rest.

   (c) (8 points) Explain how you combined the predictions in the one-vs-one setting as well as the rational behind it.

   (d) (8 points) Same as (c), but for one-vs-rest.

   (e) (10 points) **Extra Credit**: report the accuracy of your generative model. List the closed form solution for the parameters $\{\pi_k, \mu_k\}_{k=1}^{C}$ the shared covariance Σ, as well as the generated model w_k and their biases w_{k0}.

2. **Code** (5 points)- Your .m code files - your code should be runnable using a function named *run_ 1_ logreg.m.* This function estimates the the discriminative, one-vs-rest model. The resulting hypothesis should be arranged in a matrix W^{DISC} whose columns are the C binary models (similarly W^{MLE} if you implemented the extra credit generative model).

Blackboard Submission Instructions

You are required to submit both runnable Matlab code files, result mats (.mat) and a formatted report (PDF) via the blackboard system.

Upload a single zip archive file *containing a single folder*. **The name of the zip file should be your abcd_hw3_AB.zip** where abcd is the last 4 digits of your USC ID and AB is your initials. **The folder should be named as results**. If you are working in a group, only one student from the group should upload the files. Please use the uploading student's ID number for the folder name.

The folder should contain all your .m files and a single PDF report named *HW3Progmming.pdf.* The report should contain the names and USC student IDs of all the group members. It should answer all programming questions — make you mark clearly each question you are answering. Since you will have plots, you will need to refer to the plots (such as "Please see Fig. 1" — LaTeX does not place plots next to texts so you need to number the plots and refer that way.)

Due Date and Time

The programming assignment is due on blackboard the same way as the algorithm component.

Collaboration

For the coding part of the assignment, you may collaborate in groups of up to 4. Each group needs to submit to blackboard only once. Group members share the same credits. Changing group membership throughout the semester is strongly discouraged.