Problem Set 4

⋄ Please hand in the written part in class on the due date. For the programming part, please email your code and report to `gavinrbrown@gmail.com` by 23:59PM on the due date.

⋄ Late policy: there will be a penalty of 10% per day, up to three days late. After that no credit will be given.

1. **(30 points) Written Problems**

(a) (10 points) Bishop 6.2

(b) (10 points) Bishop 7.3

(c) (10 points) Bishop 7.4

2. **(70 points) Programming**

Download and save the attached data set, images of handwritten digits: `MNISTdata.mat`, `prog2`. The data represents a subsampling of the full MNIST data set available in `http://yann.lecun.com/exdb/mnist/`

1. Please submit both report and Matlab codes

2. Your report must provide an analysis of each methods performance and reasoning behind the analysis.

3. Compare in your report the relative strengths and weaknesses of the methods based on the experimental results and your understanding of each algorithm.

(a) (35 points)

Develop code for training and testing an SVM classifier with nonlinear kernel. You are welcome to use either formulation described in the textbook (chapter 7). You cannot use an SVM library to complete this assignment. You can use quadratic programming library if you like. Using your implementation of the SVM classifier, compare multi-class classification performance of two different voting schemes:

    i. one versus the rest

    ii. one versus one

Be sure to specify your voting scheme using a method described in the book . To analyze accuracy, you will find it helpful to produce and analyze the multiclass confusion matrix (`http://en.wikipedia.org/wiki/Confusion_matrix`), in addition to examining the overall error rate.

(b) (25 points) Use the same one versus one classifiers from the previous problem in a DAGSVM approach. A paper describing the approach, DAGSVM.pdf, is attached. Compare multi- class classification performance with the other two voting schemes.

(c) (c) (10 points) A baseline implementation of the DAGSVM with 6th degree polynomial kernels achieves 95% accuracy on the test set. See if you can do better than this baseline, using the DAGSVM approach. `baseline-CM.pdf` contains the confusion matrix of the baseline implementation: