CSE 455/555 Spring 2012 Homework 2

Jason J. Corso
TAs: Shujie Liu and Suxin Guo
Computer Science and Engineering
SUNY at Buffalo
jcorso@buffalo.edu
Date Assigned 20 March 2012
Date Due 19 April 2012
Homework must be submitted by midnight of the due-date, electronically (see below). No late work will be accepted.

Remember, you are permitted to discuss this assignment with other students in the class (and not in the class), but you must write up your own work from scratch.
I am sure the answers to some or all of these questions can be found on the internet. Copying from ***any*** *another source is indeed cheating. Obviously, it will undermine the primary purpose you are taking this course: to learn. This class has a zero tolerance policy toward cheaters and cheating. Don't do it.*

Problem 1: Parametric Methods (25%)

Suppose we have a random variable $x \in \{0, 1\}$ with 1 denoting the 'heads' and 0 denoting the 'tails' of the outcome from flipping a coin. We do not make any assumption on the fairness of the coin, instead, we assume the probability of $x = 1$ will be denoted by a parameter θ so that $p(x = 1|\theta) = \theta$. Thus the distribution of x_i can be denoted as $p(x|\theta) = \theta^x(1-\theta)^{(1-x)}$, which is known as the Bernoulli distribution.

Let $D = \{x_1, ..., x_n\}$ denoting n observed values drawn independently from the Bernoulli distribution.

1. Show that $p(D|\theta) = \theta^s(1-\theta)^{(n-s)}$ where $s = \sum_{i=1}^n x_i$.
2. Assuming a uniform prior distribution for θ and using the identity

$$\int_0^1 \theta^m(1-\theta)^n d\theta = \frac{m!n!}{(m+n+1)!} \quad ,$$

   show that the Bayes parameter estimation will have:

$$p(\theta|D) = \frac{(n+1)!}{s!(n-s)!}\theta_i^s(1-\theta_i)^{n-s} \quad .$$

3. Using Bayes parameter estimation of $p(\theta|D)$, integrate the product $P(x|\theta)p(\theta|D)$ over θ to obtain the desired conditional probability

$$p(x|D) = \left(\frac{s+1}{n+2}\right)^x \left(1 - \frac{s+1}{n+2}\right)^{1-x}$$

4. If we use maximum likelihood estimation to estimate θ, what is the estimated $p(x|D)$? What is the difference between maximum likelihood estimation result and Bayesian estimation result?

Problem 2: Nonparametric Methods (25%)

This is problem 6 in DHS Chapter 4.

Let $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of n independent labeled samples and let $\mathcal{D}_k(\mathbf{x}) = \{\mathbf{x}'_1, \ldots, \mathbf{x}'_k\}$ be the k nearest neighbors of $\mathbf{x}$. Recall that the k-nearest-neighbor rule for classifying $\mathbf{x}$ is to give $\mathbf{x}$ the label most frequently represented in $\mathcal{D}_k(\mathbf{x})$. Consider a two-category problem with $P(\omega_1) = P(\omega_1) = 1/2$. Assume further that the conditional densities $p(\mathbf{x}|\omega_i)$ are uniform within unit hyperspheres a distance of ten units apart.

1. Show that if k is odd, the average probability of error is given by

$$P_n(e) = \frac{1}{2^n} \sum_{j=0}^{(k-1)/2} \binom{n}{j} \tag{1}$$

2. Show that for this case the single-nearest neighbor rule has a lower error rate than the k-nearest-neighbor error rate for $k > 1$.
3. If k is allowed to increase with n but is restricted by $k < a\sqrt{n}$, show that $P_n(e) \to 0$ as $n \to \infty$.

Programming Problem: Support Vector Machines (50%)

This is the second programming question of the term and it requires you to work with linear support vector machines "from scratch".

As a starter, there is one package we need that Enthought Python does not contain. The package is for the quadratic optimization required in the SVM formulation. So, we will use the nicely done cvxopt package. Information about the package is at `http://abel.ee.ucla.edu/cvxopt/`. The CSE machines have the package avaiable (see `https://wiki.cse.buffalo.edu/services/content/cvxopt`). If you are working on your own machines, then you can also follow the instructions on the main cvxopt page to download, build and install (on some platforms, it is available as a binary package). I built it for my 32-bit EPD installation in about three minutes. Note, if you are building it, we only need the core functionality and none of the extra bells and whistles such as fftw.

It is assumed you have already been working with the `prpy` package. To prepare this assignment, be sure that you are using the current version of prpy when you are working on this assignment as some functions/protocols have changed while preparing the assignment materials.

Next, download the homework starter files `http://www.cse.buffalo.edu/~jcorso/t/CSE555/files/hw2.zip`.

The files in that zip are:

- `hw2_svm.py` This is the main source starter file. You will edit this file to complete the assignment. You will also `run hw2_svm.py` this file to produce the results for this assignment.
- `example_figout.pdf` This is an example figure output from the code, showing a linear SVM that is trained on a two-class data set. The circled green points are the support vectors and the boundary is draw in black.
- `hw2_dataX.npz` These are three ($X = \{1, 2, 3\}$) data files that are used in the assignment.

For this assignment, you do not need to work with the MNIST data set, and this assignment does not yet integrate with the first assignment. We leave that integration until the third assignment, which will have you combining both assignment 1 and 2 in an ensemble classifier scenario.

Similarly, for this assignment, you are encouraged to read the Burges SVM Tutorial briefly discussed in the course. The tutorial has all of the *answers* for this assignment.

1. (10%) You need to be acquainted with the methods in lindisc.py. So, you should take a dataset (e.g., one of those provided) and train a perceptron on it. Instantiate a `LinDisc` object and use the `testGrid` function to plot its output. Provide the code you wrote to do this and use the `pylab.savefig` command to write the figure to an image file. Submit it.

2. (40%) On the to linear no-slack SVM in `hw2_svm.py`. This code follows the Burges formulation. You are required to implement the missing parts of the code as specified in the file. The missing parts implement the elements of the quadratic program that is the base of the SVM as well as the ultimate solution for the weight vector, bias, and the margin. You need to implement all of them.

   Once done, you should `run hw2_svm.py` (make sure your `PYTHONPATH` and `LD_LIBRARY_PATH` are set properly). This will test the linear SVM on three data sets. It will also save figure outputs for all three. Submit the text output and the figure outputs.

   Provide an explanation of the outputs, the weight vector, bias term, and margin for all three cases. Pay specific attention to the uniqueness of the third data set over the first two. What is special about the third one? What would we need to do to handle this case?

   You need to submit the following:

   (a) The completed `hw2_svm.py` file. (We will run this separately with different data.)

   (b) A short description of how you implemented the missing elements from the file. I.e., how did you implement the constraints.

   (c) The output from running the script and the explanation required above.

   In addition to these requirements, you are encouraged to implement the necessary changes to solve the third data set.

Submission and Grading Information

You will be required to submit the assignment in electronic form only via the CSE department `submit` script. Information will be posted on the course website when available.

The non-programming problems are graded with partial credit for accuracy and completeness.

The programming problems are also graded with partial credit. If the programs that you provide do not run to completion (this is Python and there is no compilation), the maximum amount of credit you can get is 25% of the available points, which will be awarded for correctness of the code and any discussion. Otherwise, the programming assignments are weighted equally between correctness of the code, accuracy of the output, and written discussion/solution.
