CSE 455/555 Spring 2012 Homework 1

Jason J. Corso
Computer Science and Engineering
SUNY at Buffalo
jcorso@buffalo.edu

Date Assigned 31 Jan 2012
Date Due 27 Feb 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: Bayesian Decision Rule (25%)

Suppose the task is to classify the input signal x into one of K classes $\omega \in \{1, 2, \ldots, K\}$ such that the action $\alpha(x) = i$ means classifying x into class i. Based on zero-one loss function, the Bayesian Decision Rule is to maximize the posterior probability:

$$\alpha_{Bayes}(x) = \omega^* = \operatorname*{argmax}_{\omega} p(\omega|x)$$

We also define another two decision rules to maximize the prior and likelihood, respectively:

$$\alpha_p(x) = \omega_p^* = \operatorname*{argmax}_{\omega} p(\omega)$$

$$\alpha_L(x) = \omega_L^* = \operatorname*{argmax}_{\omega} p(x|\omega)$$

1. Suppose $K = 2$, $p(\omega = 1) = 0.7$, $p(\omega = 2) = 0.3$ and

$$p(x|\omega = 1) = \begin{cases} 0.5 & 0 \leq x \leq 2 \\ 0 & else \end{cases}$$

$$p(x|\omega = 2) = \begin{cases} 0.25 & 1 \leq x \leq 5 \\ 0 & else \end{cases}$$

   (a) Calculate the probability distribution of x as $p(x)$ given the information you have (i.e., the prior and the likelihood distributions and the knowledge that there are two possible classes). Hint: treat ω as a hidden variable.

   (b) Using zero-one loss function, calculate the overall risk of the three decision rules: Bayesian Decision Rule, Maximize Prior based Decision Rule, and Maximize Likelihood based Decision Rule.

   (c) Find out which decision rule achieves smallest overall risk and which one achieves largest overall risk.

2. Using zero-one loss function, prove that, for any distribution of x, the Bayesian Decision Rule can always achieve the minimum overall risk.

Problem 2: Bayesian Reasoning (25%)

This is a classical problem. Because you are studying so hard and staying up late, you need some caffeine. So, you run over to the Starbucks in the Commons and, to your surprise, you find a friendly greeter standing over three cups, say A, B, and C. The greeter explains to you that two of these cups have cold water in them and one has a fresh cup of hot coffee (the cups are opaque and you cannot see what is inside); you can select one and if it is the hot coffee then you win (and you get the hot coffee for free). So, you select cup A. Then, the greeter, who knows what is in each cup, opens cup C, which has cold water in it. The greeter then asks you whether or not you want to switch to cup B.

1. Formulate the problem using the Bayes rule, i.e. what are the random variables and the input data? What are the meanings of the prior and posterior probabilities in this problem?
2. What are the probability values for the prior?
3. What are the probability values for the likelihood?
4. Calculate the posterior probabilities (you need to derive the probability values with intermediate steps, not simply showing the final values).
5. What is the probability that cup A has hot coffee behind it after the greet opens cup C?
6. Should you switch the choice to cup B? Why?

Happy coffee drinking!

Programming Problem: Decision Trees and Random Forests (50%)

This is the first programming question of the term and it requires you to work with decision trees and random forests in Python.

For starters, you need to download and get familiar with the `prpy` package that has been developed for this course. Information on `prpy` is available on the course website. 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 source file `http://www.cse.buffalo.edu/~jcorso/t/CSE555/files/rfdigits_hw1.py`.

Finally, download the dataset from the website or locate it on the CSE network: `http://www.cse.buffalo.edu/~jcorso/t/CSE555/files/555_mnist_data.tar.bz`. The datafiles are also available on the CSE network at `/projects/jcorso/CSE555/555_mnist_data` that you can use without actually copying them locally into your account. This is a subset of the LeCun's MNIST dataset containing just the digits 0 through 5. The full dataset is available at `http://yann.lecun.com/exdb/mnist/`. The dataset is split into training and testing pictures. Do all training on the training pictures, and reserve the testing pictures for classification. I have preserved further testing images that I will use for evaluating the assignments.

I will not provide a tutorial on the prpy package or the assignment source files, other than what has been described in class. You can read the documentation in the source files to get a better idea of what is going on.

1. (10%) The first part of the assignment has you becoming acquainted with the existing Decision Tree code and tools. Use the prpy code and provided examples to generate a 4 class 2D dataset on the plane. Learn a decision tree. And compute the accuracy of the decision tree on your data set (see `pr.datatools.accuracy`). Plot the decision regions using the `pr.trees.testGrid` function. Provide screenshots of the data set you created, the accuracy score, and the decision regions as well as a discussion on why the method performed as it did.

2. (5%) Next, implement the multiclass variance impurity and retrain the decision tree. Compute the accuracy. Describe what has changed and also submit the py file with the impurity function.

3. (35%) Next, you are required to implement a Random Forest classifier for classifying grayscale images of handwritten digits. For simplicity, I have provided a `rfdigits_hw1.py` starter file that contains the full skeleton of the random forest code (both learning and classification) as well as the functionality to load in the data and compute features on it.

   You are required to implement the missing parts of the code as specified in the file. Two of the parts are for the classification side and one is for the learning side. You need to do all three.

   Just like in the Amit and Geman digits example given in class, the features that have been defined in this code are too many to enumerate. So, during the randomized learning of the tree, you will have to dynamically instantiate a random set of, say 100, features and learn the best query based on them. Then, at the next node, you instantiate a new random set of features and so on. The actual features that we will use are simple weighted sums of pixel values (all of the images are 28×28 grayscale). The class `KSum_Feature` has been provided for you to easily compute these features. Finally, the class `DTQuery_KSum` has been provided to encapsulate a query based on these features; you do not need to use this one if you don't want, but it is recommended.

   You need to complete the `rfdigits_hw1.py` file, and execute it (by typing, for example, `python rfdigits_hw1.py` assuming your `PYTHONPATH` is set properly. Running this script will load a subset of the data, train the tree, and then compute the accuracy on the training set as well as the accuracy on the provided testing set.

   You need to submit the following:

   (a) The completed `rfdigits_hw1.py` file. (We will run this separately with different data.)

   (b) A short description of how you implemented the training function for the randomized trees.

   (c) A short description of what happens when you run the `rfdigits_hw1.py` file; i.e., what are you accuracies? How do the accuracies vary as you change the parameters of the tree (what if you use a `featureDepth` of 10 or 100, how does this change the accuracy?).

   In addition to these requirements, you are encouraged to explore the classifier and the problem further to, for example, display images of digits that were misclassified or visualize the paths down a tree given the selected type of feature.

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.
