CS 165B – Machine Learning, Spring 2016

Assignment #5
Due Friday, June 3 by 4:30pm

Notes:

- *This assignment is to be done individually. You may discuss the problems at a general level with others in the class (e.g., about the concepts underlying the problem, or what lecture or reading material may be relevant), but the work you turn in must be solely your own.*
- Be sure to re-read the "Policy on Academic Integrity" on the course syllabus.
- Be aware of the late policy in the course syllabus – i.e., *late submissions will not be accepted*, so turn in what you have by the due time.
- Any updates or corrections will be posted on the Assignments page (of the course web site), so check there occasionally.
- Your code must be well commented – graders should not have to work to figure out the purpose or method of any section or line or code.
- You may use Java, Python, C++, or Matlab in your programming solution.
- Turning in the assignment:
  - There is no hardcopy to turn in for this homework.
  - Turn in your source code and output files (as described in the problems) using the **`turnin`** command on CSIL. You'll turn in files in a single directory like this:

```
% turnin hw5@cs165b hw5
```

Problem #1 [90 points]
Create a binary classifier using bagging, based on the simple linear classifier (similar to what you implemented in HW2, problem 6, but with two classes instead of three). You may use code from the posted solutions or from your own solution to HW2, problem 6.

Write a program called **`bagit`** that implements the basic linear classifier (where the discrimination surface is half-way between the class sample means, perpendicular to the vector connecting the means) and uses this in a bagging approach to classify the testing data. The syntax of your program should be:

```
% bagit [-v] T size <train> <test>
```

where **`–v`** is an optional "verbose" flag, **`T`** is number of classifiers (and bootstrap sample sets) created, **`size`** is the number of data points in each bootstrap sample set, and **`<train>`** and **`<test>`** are filenames containing the training and testing data sets, respectively. **`T`** and **`size`** can be any non-zero integers.

Each of the **`T`** bootstrap sample sets is created by randomly sampling (with replacement) the training instances – i.e., randomly choosing **`size`** training data points to be included in the bootstrap sample set. (Since the sampling is "with replacement," some of the data points may be repeated.) Each training data point should have an equal probability of being selected

From each bootstrap sample set, create a simple linear classifier. Once all **`T`** classifiers have been created, run them on the testing data set and combine the outputs with majority vote. (Using $\boldsymbol{w}^T\boldsymbol{x} > 0$ to test for membership in the True class, points exactly on the boundary should be classified as False.) Ties in the voting (from outputs of the multiple classifiers) should assign a point to the True class.

As output, the program should print out the numbers of positives and negatives in the testing data and the numbers of false positives and false negatives in the testing output. If the "verbose" flag is set (**`–v`**), then additional information should be printed out: the composition of the **`T`** bootstrap sample sets and the classification results on the testing data. For example:

```
% bagit –v 3 6 training_file testing_file
Positive examples: 10
Negative examples: 9
False positives: 4
False negatives: 4

Bootstrap sample set 1:
3.4 5.63 -12.2 – True
2.0 4.2 -10.0 – True
1.9 -2.3 4.8 - False
2.0 4.2 -10.0 – True
-0.33 0.0 1.0 – False
3.4 5.63 -12.2 – True
```

(… etc. for two more bootstrap sample sets, then…)

```
Classification:
0.0 1.0 2.0 - False (correct)
-2.3 1.8 2.1 – False (correct)
3.6 4.5 -3.17 – True (correct)
1.0 0.0 -1.0 – True (false positive)
```

(… etc. for the remainder of the testing set)

Only one of the bootstrap sample sets is shown in this example printout, but all three sets (in this example) should be printed if the "**`-v`**" flag is used. Then each instance in the testing set should be printed along with your classifier's output (**`True`** or **`False`**) and an evaluation of the classification (**`correct`**, **`false positive`**, or **`false negative`**).

The training and testing data files have the same format as in HW2, problem 6, except there will always be two classes: the True class followed by the False class:

```
D N_true N_false
P11 P12 … P1D
P21 P22 … P2D
…
```

where **`D`** is the dimensionality of the data points, **`N_true`** is the number of points in the True class, **`N_false`** is the number of points in the False class, and **`Pij`** is real-valued, describing the **`j`**th component of point **`i`**. The values of **`D`** must be the same in the training and testing files. The True points are listed first, then the False points.

Run the program on the provided data sets, and save the output from these examples in different text files to be turned in with your code:

`HW5-output1.txt`: Run on example 1 with the **`–v`** flag, **`T=3`**, and **`size=10`**
`HW5-output2.txt`: Run on example 2 with the **`–v`** flag, **`T=5`**, and **`size=6`**

Although not part of the assignment, you are encouraged to compare this bagging classifier with the original linear classifier (on the provided data sets and/or on your own).

Problem #2 [10 points]
Run your **`bagit`** program from Problem #1 on the new datasets provided for this problem and report error rates (on the test data) for **`T=1`**, **`T=3`**, **`T=5`**, **`T=7`**, and **`T=9`** and **`size=|D|`** (the size of the training data set). For convenience of grading, provide this information in a file called **`Problem2Results.txt`** (or .doc or .docx or .pdf) in the same directory as problem 1.