CS 165B – Machine Learning
Spring 2018, William Wang

Assignment #2
Due Tuesday, May 15 by 10:00am *PST*

This homework has 2 parts!

a. Written assignment to be turned into Gauchospace
b. Coding assignment to be submitted to CodaLab (Directions to follow)

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 question, or what lecture or reading material may be relevant), but the work you turn in must be solely your own.
- Be aware of the late policy in the course syllabus – i.e., you only have four late days for all the assignments,
- Justify every answer you give – **show the work** that achieves the answer or **explain** your response.
- Any updates or corrections will be posted on the Assignments page (of the course web site), so check there occasionally.
- Be sure to **include your name and perm number on the required place of the assignment (at the top of the 2nd page).**
- Be sure to **strictly stick to the format provided for answering all the questions**.
- **All assignments must be clear and legible.** It is recommended to type the solutions on this PDF directly. If you'll be submitting a handwritten assignment, please ensure that it's readable and neat. If your writing is not easily readable, your solution is not easy to follow on the page, or your PDF is not of very high quality, your assignment will not be graded. DO NOT submit picture of your written work. (If you must scan your written work in, use a high-quality scanner. Plan in advance.)
- Be sure to re-read the "Academic Integrity" on the course syllabus. You must complete the section below. If you answered Yes to either of the following two questions, give corresponding full details.

Did you receive any help whatsoever from anyone in solving this assignment? Yes No

Did you give any help whatsoever to anyone in solving this assignment? Yes No

---

Name:

Perm Number:

Problem #1 [8 points]
A ranking classifier ranks 25 training examples $\{x_i\}$, from highest to lowest rank, in the following order:

Highest Lowest

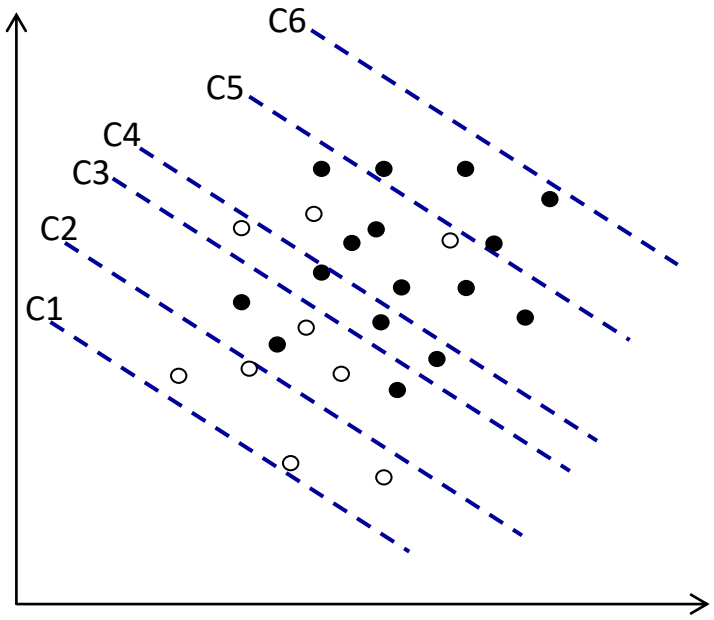
$x_2, x_3, x_1, x_5, x_{13}, x_6, x_8, x_7, x_9, x_{10}, x_{12}, x_{11}, x_{15}, x_4, x_{14}, x_{21}, x_{17}, x_{20}, x_{18}, x_{22}, x_{16}, x_{19}, x_{25}, x_{23}, x_{24}$

Examples x_1 through x_{12} are in the positive class (which should be ranked higher); examples x_{13} through x_{25} are in the negative class (which should be ranked lower).

(a) How many ranking errors are there?
(b) What is the ranking error rate?
(c) What is the ranking accuracy?
(d) Draw the coverage curve for the ranking classifier on this dataset.

Problem #2 [14 points]

The figure below shows training data with two features, with each example labeled as being in the positive (filled-in points) or negative (open points) class. Proposed linear discriminant functions (C1 through C6) are shown as dotted lines, each one indicating a different classifier for this data. Each classifier classifies points to the upper-right of its dotted line as positive and points to the lower-left of its dotted line as negative.



(a) Draw the coverage plot for this data and plot the different classifiers (and label them as C1, C2, etc.).

(b) Draw the ROC plot and label the classifiers on the plot.

(c) Which classifiers have the highest and lowest accuracy?

(d) Which classifiers have the highest and lowest precision?

(e) Which classifiers have the highest and lowest recall?

(f) Which classifiers (if any) are complete?

(g) Which classifiers (if any) are consistent?

Problem #3 [16 points]

The linear discriminant function for a binary classification problem with three features is the plane defined by the equation $3x_1 + 2x_2 + 4x_3 = 18$, where the features are x_1, x_2, and x_3. The discriminant plane separates the positive hypotheses ($3x_1 + 2x_2 + 4x_3 \geq 18$) from the negative hypotheses ($3x_1 + 2x_2 + 4x_3 < 18$).

The training set consists of the following labeled examples:

- (2, 2, 3) + (positive class)
- (3, 3, 2) +
- (1, 2, 3) +
- (1, 4, 1) +
- (4, 4, 4) +
- (2, 2, 2) +
- (3, 3, 1) – (negative class)
- (1, 1, 1) –
- (3, 3, 2) –
- (0, 4, 2) –
- (4, 0, 0) –
- (0, 0, 3) –

For each data point, give (a) its margin, (b) its 0-1 loss, (c) its hinge loss, and (d) its squared loss (clamped to zero above 1).

Problem #4 [10 points]
In a 5-class classification problem, 30 training examples are supplied that have the following class labels:

3	1	2	2	3	5
5	5	3	3	5	3
5	3	5	3	5	2
2	5	5	5	2	1
3	5	1	3	5	2

From the training data, we wish to estimate the probabilities of each class. Empirically estimate each class probability using (a) relative frequency, (b) Laplace correction, (c) m-estimate with m=5 and an even distribution of the pseudocounts, and (d) m-estimate with m=20 and an even distribution of the pseudocounts.

Plot these empirical probabilities, with class number on the x-axis and estimated probability on the y-axis – i.e., four plots (one for each approach), each of which consists of five connected points (the estimated class probabilities). Plot all four distributions on a single graph and label them.

Describe the trend as we go from (a) to (b) to (c) to (d) – that is, what does increasing the number of pseudocounts do (in general) to the probability distribution?

Problem #5 [10 points]
We'd like to learn a Boolean function that separates the people in the Hatfield family from people in the McCoy family. We know the following information about a given person:

Age status: { *Child, Adult, Elderly* }
Residency: { *West Virginia, Kentucky* }
Teeth: { *Few, Many* }
Sympathizes with: { *Union, Confederate, Neither* }
Occupation: { *Miner, Bootlegger, Other, None* }

(a) For this simple problem, if testing a classification hypothesis takes a nanosecond, how long would it take to test every possible hypothesis?

(b) How long would it take if we used a conjunctive hypothesis space representation?

(c) How long would it take if we used a conjunctive hypothesis space with internal disjunctions?

Programming Assignment

Problem #6 [40 points]
Write a **Python 2** program called **`hw2.py`** that implements a three-class linear classifier using the following method:

Training (using the training data set):

1. Compute the centroid of each class (A, B, and C).
2. Construct a discriminant function between each pair of classes (A/B, B/C, and A/C), halfway between the two centroids and orthogonal to the line connecting the two centroids. This is the "basic linear classifier" that we have discussed.

Testing (using the testing data set):

1. For each instance, use the discriminant function to decide "A or B" and then (depending on that answer) to decide "A or C" or "B or C." (Ties should give priority to class A, then B, then C.)
2. Keep track of true positives, true negatives, false positives, and false negatives.

The training and testing data sets are available on the resources page on piazza, along with a description of their formats. You are also provided a starter code file called **hw2.py**. In the hw2.py file you will find the function

```
% def run_train_test(training_input, testing_input)
```

This function is where you should implement the assignment. Feel free (it is strongly encouraged) to use additional functions, but **DO NOT** change this function signature and **DO NOT** change the hw2.py module name. We will call this as the entry point to your code.

As output, the program should return a dictionary of averages of the true positive rate, the false positive rate, the error rate, the accuracy, and the precision – e.g., as shown here:

```
% print(run_train_test(training_input, testing_input)
{
   "tpr": 0.80 # true positive rate
   "fpr": 0.27 # false positive rate
   "error_rate": 0.44
   "accuracy": 0.60
   "precision": 0.90
}
```

(Note: These numbers are made up, for purposes of illustration only.) The run_train_test function should **return results using the same keys as shown.**

Information on computing these averages is included in the hw2 starter package.

You can test your program with the training and testing sets provided on the Piazza.

CodaLab

Submission Instructions

You will need to create a zip file containing your submission. You should use the following command:

```
% zip –r submission.zip hw2.py
```

The **submission.zip** file is what will be uploaded to CodaLab where it is automatically tested. **DO NOT** put your code in a new directory, as our testing functions will not be able to import your code if you do.

CodaLab Competition Link

You will submit your code through this competition and get results for your homework. Below is the competition where you should submit

https://competitions.codalab.org/competitions/18902?secret_key=4255726d-ee46-49db-bb1a-60e274855965

Registration Instructions

Please follow the directions here to register and participate in our class competition: https://github.com/codalab/codalab-competitions/wiki/User_Participating-in-a-Competition

****Register with you UMail account!****

Remember to register a CodaLab Competitions account using your umail account so that the username will be your UCSBNetID. After that, log into CodaLab Competitions and set up your team name (whatever nickname you like). To protect your privacy, only the team names will be shown on the leader-board and your usernames will be anonymous. After your submission finishes running, please choose to submit it to the leader-board. Note that here team name is equivalent to your nickname, and it is still an independent homework assignment. You must implement the four algorithms according to the instructions above.