Assignment 2
CSE471 : Statistical Methods in Artificial Intelligence (SMAI)
Spring 2017
Submission Deadline : 23.55 Hrs. 22/02/2017

General Instructions

1. Assignment can be implemented in Matlab/Octave, Python, C/C++, R .
2. Ensure that submitted assignment is your original work. Please do not copy any part from any source including your friends, seniors and/or the internet. If any such attempt is caught then serious actions including an F grade in the course is possible.
3. A single pdf file needs to be uploaded to the Courses Portal. The file should contain your answers as well as the code you have written and its output. Include the assignment number, your name and roll number at the top-left of the first page of your submission.
4. Your grade will depend on the correctness of answers and output. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.

Problem 1: Perceptron Learning [5 Marks]

Write a program to implement online (single-sample) Perceptron algorithm.

1. Use the data given in Table 1. Starting with $\mathbf{w} = [0 \quad 0]^T$ and $b = 0$, apply your program to learn a linear classifier to separate classes C_1 and C_2. Plot the data points for C_1 and C_2. Plot the final classifier learnt at the convergence. Note the number of iterations required for convergence.
2. Use the data given in Table 1. Run the same algorithm for building a classifier to separate C_2 and C_3. Again, plot the data points for C_2 and

samples	C_1		C_2		C_3	
	x_1	x_2	x_1	x_2	x_1	x_2
1	0.1	1.1	7.1	4.2	-3.0	-2.9
2	6.8	7.1	-1.4	-4.3	0.5	8.7
3	-3.5	-4.1	4.5	0.0	2.9	2.1
4	2.0	2.7	6.3	1.6	-0.1	5.2
5	4.1	2.8	4.2	1.9	-4.0	2.2
6	3.1	5.0	1.4	-3.2	-1.3	3.7
7	-0.8	-1.3	2.4	-4.0	-3.4	6.2
8	0.9	1.2	2.5	-6.1	-4.1	3.4
9	5.0	6.4	8.4	3.7	-5.1	1.6
10	3.9	4.0	4.1	-2.2	1.9	5.1

Table 1: Data for Problem 1

C_3. Plot the final classifier. Note the number of iterations required for convergence.

3. Comment on the difference on the number of iteration required for convergence in above two cases and give reasons.

Problem 2: Voted Perceptron [5 Marks]

The Perceptron algorithm converges only when the data is linearly separable. It does not converge when the data is non-separable. Also, the performance of Perceptron on unseen test data is not that amazing. In order to make it more competitive with other learning algorithms, it needs to be modified a bit to get better generalization.

Consider a data set with 10,000 examples. Suppose that after the first 100 examples, the Perceptron has learned a really good classifier. Its so good that it goes over the next 9899 exam- ples without making any updates. It reaches the 10,000th example and makes an error. It updates. For all we know, the update on this 10,000th example completely ruines the weight vector that has done so well on 99.99%. What we would like is for weight vectors that survive a long time to get more say than weight vectors that are overthrown quickly. One way to achieve this is by voting. As the perceptron learns, it remembers how long each hyperplane survives. At test time, each hyperplane encountered during training votes on the class of a test example. If a particular hyperplane survived for 20 examples, then it gets a vote of 20. If it only survived for one example, it only gets a vote of 1. In particular, let $(\mathbf{w}_1, b_1), ..., (\mathbf{w}_K, b_K)$ be the K weight vectors encountered during training, and $c_1, ..., c_K$ be the survival times for each of these weight vectors. (A weight vector that gets immediately updated gets $c = 1$; one that survives another round gets $c = 2$ and so on.)

Then the prediction on a test point is:

$$\hat{y} = \text{sign}\Big\{ \sum_{k=1}^{K} c_k \text{sign}(\mathbf{w}_k^T \mathbf{x} + b_k) \Big\}$$

This algorithm is known as **voted Perceptron** works quite well in practice, and there is some nice theory showing that it is guaranteed to generalize better than the vanilla Perceptron.

Algorithm 1: Voted Perceptron

Input: $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
Output: $(\mathbf{w}_1, b_1, c_1), \ldots, (\mathbf{w}_n, b_n, c_n)$
begin
  Initialization $n = 1,\ \mathbf{w}_1 = \mathbf{0}, b_1 = 0, c_1 = 1;$
  for *iter* $\leftarrow 1$ **to** *Num_Epochs* **do**
    for $i \leftarrow 1$ **to** m **do**
      if $y_i(\mathbf{w}_n^T \mathbf{x}_i + b_n) \leq 0$ **then**
        $n = n + 1;$
        $\mathbf{w}_n = \mathbf{w}_{n-1} + y_i \mathbf{x}_i;$
        $b_n = b_{n-1} + y_i;$
        $c_n = 1$
      end
      else
        $c_n = c_n + 1$
      end
    end
  end
end

Assignment Problem

1. Write the program to implement Voted Perceptron.
2. Consider the following UCI Datasets for classification
   - Breast Cancer Dataset: `http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/`
   - Ionosphere Dataset: `http://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/`

   Remove the rows corresponding to missing values in Breast Cancer Dataset.
3. Run the vanilla Perceptron as well as Voted Perceptron on both the datasets.
4. Report the following:

(a) Take different values for number of Epochs, say 10,15, 20, 25, 30, 35, 40, 45, 50. One Epoch is a complete pass of the training set from the algorithm.

(b) For each Epoch value in the above, report 10-fold cross validation accuracies of both Perceptron and Voted Perceptron (for both the datasets). (For those who don't know cross validation accuracy, can read the link below: `http://scikit-learn.org/stable/modules/cross_validation.html`)

(c) Plot the number of Epochs versus Cross validation accuracy for both approaches and for both the datasets.

(d) Comment on the performance of two approaches. Give reasons.

Problem 3: Least Square Approach [5 Marks]

samples	C_1		C_2	
	x_1	x_2	x_1	x_2
1	3	3	-1	1
2	3	0	0	0
3	2	1	-1	-1
4	0	2	1	0

Table 2: Dataset 1 for Problem 3

samples	C_1		C_2	
	x_1	x_2	x_1	x_2
1	3	3	-1	1
2	3	0	0	0
3	2	1	-1	-1
4	0	1.5	1	0

Table 3: Dataset 2 for Problem 3

Table 2 and Table 3 show two different datasets for binary classification problem. Do the following:

1. Write the program to find the linear classifier using least square approach.
2. Write the program to find the linear classifier using Fisher's linear discriminant.
3. Plot the data points in Table 2 with different colors for different classes. Find the classifiers learnt using both least square approach as well as Fisher's linear discriminant analysis. Plot both the classifiers on the same graph.

4. Plot the data points in Table 3 with different colors for different classes. Find the classifiers learnt using both least square approach as well as Fisher's linear discriminant analysis. Plot both the classifiers on the same graph.

5. Comment on the difference in the classifier learnt in the above two cases. Give reasons.

Problem 4: Relation between Least Squares and Fisher's linear discriminant [5 Marks]

Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, y_m)\}$. Let m_1 be the number of points in class C_1 and m_2 be the number of points in class C_2. Thus, if we take the class labels as follows:

$$y_i = \begin{cases} \frac{m}{m_1} & \text{if } \mathbf{x}_i \in C_1 \\ -\frac{m}{m_2} & \text{if } \mathbf{x}_i \in C_2 \end{cases}$$

Show that with these class labels, linear classifier learnt using least squares is same the Fisher's linear discriminant.