10-315 Introduction to Machine Learning

Due:11:59 p.m. Monday, Feb 11, 2019

Instructions

- **Submit your homework on time electronically by submitting to Autolab**.
- **Late homework policy**: Homework is worth full credit if submitted before the due date. Up to 50 % credit can be received if the submission is less than 48 hours late. The lowest homework grade at the end of the semester will be dropped. Please talk to the instructor in the case of extreme extenuating circumstances. **Note that, your scores on the coding part will also be penalized if you choose to submit the writeup after the due date.**
- **Collaboration policy**: You are welcome to collaborate on any of the questions with anybody you like. However, you must write up your own final solution, and you must list the names of anybody you collaborated with on this assignment.
- **Writeup format**:
  - The written portion should be typeset (e.g. LaTeX).
  - The submission format should be PDF.
  - Every problem should be on a different page.

1 Linear Separators [30 pts]

Recall that a linear separator in 2 dimensions is given a function of the form

$$y = \phi(w_1 x_1 + w_2 x_2 + b)$$

Pictorially:

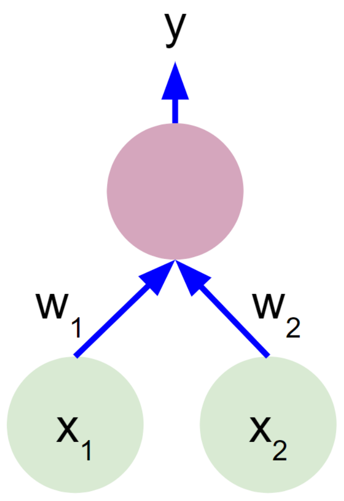

Figure 1: Linear Separator for question 1

where $x_i \in \{0, 1\}$, and transfer function

$$\phi(z) = \begin{cases} 0 & if\ z \leq 0 \\ 1 & otherwise \end{cases}$$

a. [4 pts] Complete the following truth table for the AND operation.

x_1	x_2	AND
0	0	
0	1	
1	0	
1	1	

b. [4 pts] Provide values of w_1, w_2 and b to use the linear separator for logical AND.

c. [4 pts] Complete the following truth table for the OR operation.

x_1	x_2	OR
0	0	
0	1	
1	0	
1	1	

d. [4 pts] Provide values of w_1, w_2 and b to use the linear separator for logical OR.

e. [4 pts] Complete the following truth table for the XOR operation.

x_1	x_2	XOR
0	0	
0	1	
1	0	
1	1	

f. [10 pts] Prove that the linear separator depicted in Figure 1 cannot be used to create logical XOR.

Early	**Finished hmk**	**Senior**	**Likes Coffee**	**Liked The Last Jedi**	**A**
1	1	0	0	1	1
1	1	1	0	1	1
0	0	1	0	0	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	1	1	1	1
1	0	0	0	1	0
0	1	0	1	1	1
0	0	1	0	1	1
1	0	0	0	0	0
1	1	1	0	0	1
0	1	1	1	1	0
0	0	0	0	1	0
1	0	0	1	0	1

Table 1: decision tree features and labels

2 Decision Trees [30 pts]

a. [5 pts] How many unique, perfect binary trees of depth 3 can be drawn if we have 5 attributes. By depth, we mean depth of the splits, not including the nodes that only contain a label. So a tree that checks just one attribute is a depth 1 tree. By perfect binary tree, we mean every node has either 0 or 2 children, and every leaf is at the same depth. Note also that a tree with the same attributes but organized at different depths are considered "unique". Do not include trees that test the same attribute along the same path in the tree.

(Bonus) [5 pts] In general, for a problem with A attributes, how many unique full D depth trees can be drawn? Assume $A >> D$

b. [15 pts] Consider the following dataset for this problem. Given the five attributes on the left, we want to predict if the student got an A in the course. Create 2 decision trees for this dataset. For the first, only go to depth 1. For the second go to depth 2. For all trees, use the ID3 entropy algorithm from class. For each node of the tree, show the decision, the number of positive and negative examples and show the entropy at that node.
Hint: There are a lot of calculations here. You may want to do this programatically. Table 1 is provided as a txt file on the website.

(Bonus) [5 pts] Make one more decision tree. Use the same procedure as in (b), but make it depth 3. Now, given these three trees, which would you prefer if you wanted to predict the grades of 10 new students who are not included in this dataset? Justify your choice.

c. [10 pts] Recall the definition of the "realizable case." For some fixed concept class C, such as decision trees, a realizable case is one where the algorithm gets a sample consistent with some concept $c \in C$. In other words, for decision trees, a case is realizable if there is some tree that perfectly classifies the dataset.

If the number of attributes A is sufficiently large, under what condition would a dataset not be realizable for decision trees of no fixed depth? Prove that the dataset is unrealizable if and only if that condition is true.

3 Programming Assignment: Perceptrons [40 pts]

3.1 Implementation [30 pts]

Algorithm In this part, you will be implementing the perceptron algorithm for classifying handwritten digits.

Recall the perceptron algorithm we learned in class for $x_i \in \mathcal{R}^d$ and $y_i \in \{-1, 1\}$

- Initialize the weight vector $w \in \mathcal{R}^d$
- Iterate until convergence:
    - For $i = 1...n$
        - $\hat{y_i} = \phi(w^T x_i)$
        - If: $\hat{y_i} \neq y_i$; Then $w = w + y_i x_i$

Where we use a slightly altered definition from Problem 1:

$$\phi(z) = \begin{cases} -1 & if\ z \leq 0 \\ 1 & otherwise \end{cases}$$

Note that this is an iterative algorithm and there are different ways to define "convergence" In this assignment we provide the convergence rule for you. It stops that algorithm when $\hat{y} = \hat{y}^{(prev)}$. Note that $\hat{y}$ and $\hat{y}^{(prev)}$ are both vectors. This condition means that the algorithm is no longer changing the predicted values of $\hat{y}$.

Code structures Two files are provided:

- data.py: it provides functions to load the dataset. Do not change this file.
- perceptron.py: it provides a template for implementing the perceptron algorithm.

You should complete the following function:

- Complete the `standardize` and the `standardize_by_mean_and_std` function: compute the mean and std in the `standardize` function and then use function `standardize_by_mean_and_std` to standardize features to mean 0 and variance 1 in this function. In addition, append a constant with value 1 to the feature. You will find `numpy.mean` and `numpy.std` useful.
- Complete the `perceptron_pred` function: the function `preceptron_pred(x, w)` takes in a weight vector w and a feature vector x, and outputs the predicted label based on $\phi(z)$. You will find `numpy.dot` useful.
- Complete the `perceptron` function: the function `perceptron(X, Y, w)` runs the perceptron training algorithm taking in an initial weight vector w, feature matrix X, and vector of labels Y. It outputs a learned weight vector w.

Software versions Make sure that you are using Python 3 and numpy 1.13.3. Otherwise, when running your program on Autolab, it may produce a result different from the result produced on your local computer.

Testing on Datasets

(1) [10 pts] First let's try your implementation on a very simple dataset. The dataset is a simple 2D dataset with a cluster of positive and a cluster of negative examples. Your algorithm should easily separate the two clusters. You can find the dataset in the handout named simple_dataset.mat. You can test your implementation's accuracy using function `test_simple_dataset`.

You will get full score if your implementation can get 100% accuracy on this dataset. Otherwise, your score would be calculated by:

$$\exp((\text{acc} - 1) * 10) * 10 \tag{1}$$

where acc is your implementation's accuracy.

(2) [20 pts] Then, test your algorithm on a subset of the MNIST dataset. You can find this in the handout as perceptron_train.mat and perceptron_test.mat. Run perceptron.py to perform training and see the performance on training and test sets. Record your accuracy on both datasets. For reference, we achieved a 97.6% test accuracy. You can test your implementation's accuracy using function `test_mnist`.

You will get full score if your implementation can get 96.0% accuracy on this dataset. Otherwise, your score would be calculated by:

$$\exp((\text{acc} - 96.0) * 10) * 20 \tag{2}$$

where acc is your implementation's accuracy.

Submission Instructions ***DO NOT*** change the name of any of the files or folders in the handout. In other words, your submitted files should have exactly the same names as those in the submission template. Do not modify the directory structure. Do not change functions other than the four previously mentioned functions.

Run the following command to generate a tarfile:

```
$ tar -cvf src.tar src
```

Then submit your tarfile to Autolab. (***DO NOT*** change this command.) After submission, if you do not get a full score, you can click your score and see the logs of the autograding. We recommend debugging your code locally since autograding is very slow.

3.2 Visualization [10 points]

Write a visualization function to visualize the decision boundary on the simple dataset. The visualization should look something like Figure 2.

Once you have done that, plot the decision boundary for the first 5 iterations of the algorithm on the simple dataset, and then once more at convergence. You should have 6 figures in your writeup. Even if the convergence criterion described in Section 3.1 happens before 5 iterations, on this dataset please still run the algorithm for enough iterations to generate these plots.

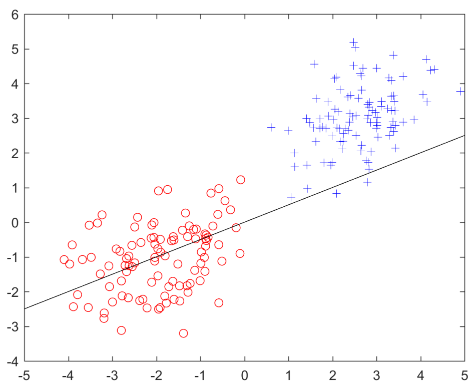

Figure 2: Plot of perceptron decision boundary