CS 5350/6350: Machine Learning Spring 2019

Homework 3

Handed out: 25 Feb, 2019
Due date: 11:59pm, 9 Mar, 2019

- You are welcome to talk to other members of the class about the homework. I am more concerned that you understand the underlying concepts. However, you should write down your own solution. Please keep the class collaboration policy in mind.
- Feel free to discuss the homework with the instructor or the TAs.
- Your written solutions should be brief and clear. You do not need to include original problem descriptions in your solutions. You need to show your work, not just the final answer, but you do *not* need to write it in gory detail. Your assignment should be **no more than 15 pages**. Every extra page will cost a point.
- Handwritten solutions will not be accepted.
- *Your code should run on the CADE machines.* **You should include a shell script, `run.sh`, that will execute your code in the CADE environment. Your code should produce similar output to what you include in your report.**

  You are responsible for ensuring that the grader can execute the code using only the included script. If you are using an esoteric programming language, you should make sure that its runtime is available on CADE.
- Please do not hand in binary files! We will *not* grade binary submissions.
- The homework is due by **midnight of the due date**. Please submit the homework on Canvas.

1 Paper Problems [40 points + 10 bonus]

1. [7 points] Suppose we have a linear classifier for 2 dimensional features. The classification boundary, i.e., the hyperplane is $2x_1 + 3x_2 - 4 = 0$ (x_1 and x_2 are the two input features).

x_1	x_2	label
1	1	1
1	-1	-1
0	0	-1
-1	3	1

Table 1: Dataset 1

(a) [3 points] Now we have a dataset in Table 1. Does the hyperplane have a margin for the dataset? If yes, what is the margin? Please use the formula we discussed in the class to compute. If no, why? (Hint: when can a hyperplane have a margin?)

The margin for this dataset is $\frac{1}{\sqrt{13}}$, from the first data point.

x_1	x_2	label
1	1	1
1	-1	-1
0	0	-1
-1	3	1
-1	-1	1

Table 2: Dataset 2

(b) [4 points] We have a second dataset in Table 2. Does the hyperplane have a margin for the dataset? If yes, what is the margin? If no, why?

Because this hyperplane misclassifies the final data point there is no margin.

2. [7 points] Now, let us look at margins for datasets. Please review what we have discussed in the lecture and slides. A margin for a dataset is not a margin of a hyperplane!

x_1	x_2	label
-1	0	-1
0	-1	-1
1	0	1
0	1	1

Table 3: Dataset 3

(a) [3 points] Given the dataset in Table 3, can you calculate its margin? If you cannot, please explain why.

The hyperplane with the lowest margin is one that goes diagonally through the origin, which gives a margin of $\frac{1}{\sqrt{2}}$.

x_1	x_2	label
-1	0	-1
0	-1	1
1	0	-1
0	1	1

Table 4: Dataset 4

(b) [4 points] Given the dataset in Table 4, can you calculate its margin? If you cannot, please explain why.

Because this data is not linearly separable it does not have a margin.

3. [8 points] Let us review the Mistake Bound Theorem for Perceptron discussed in our lecture.

(a) [3 points] If we change the second assumption to be as follows: Suppose there exists a vector $\mathbf{u} \in \mathbb{R}^n$, and a positive γ, we have for each $(\mathbf{x}_i, y_i)$ in the training data, $y_i(\mathbf{u}^\top \mathbf{x}_i) \geq \gamma$. What is the upper bound for the number of mistakes made by the Perceptron algorithm? Note that $\mathbf{u}$ is unnecessary to be a unit vector.

If $\mathbf{u}$ is not a unit vector then the upper bound for mistakes is $(||\mathbf{u}||R/\gamma)^2$

(b) [3 points] Following (a), if we do NOT assume $\mathbf{u}$ is a unit vector, and we still want to obtain the same upper bound introduced in the lecture, how should we change the inequalities in the second assumption?

$\frac{y_i(\mathbf{u}^\top \mathbf{x}_i)}{||\mathbf{u}||} \geq \gamma$

(c) [2 points] Now, let us state the second assumption in another way: Suppose there is a hyperplane that can correctly separate all the positive examples from the negative examples in the data, and the margin for this hyper plane is γ. What is the upper bound for the number of mistakes made by Perceptron algorithm?

This is an equivalent statement to $\frac{y_i(\mathbf{u}^\top \mathbf{x}_i)}{||\mathbf{u}||} \geq \gamma$, so the upper bound for error is still $(||\mathbf{u}||R/\gamma)^2$.

4. [6 points] We want to use Perceptron to learn a disjunction as follows,

$$f(x_1, x_2, \dots, x_n) = \neg x_1 \lor \neg \dots \neg x_k \lor x_{k+1} \lor \dots \lor x_{2k} \quad \text{(note that } 2k < n).$$

The training set are all 2^n Boolean input vectors in the instance space. Please derive an upper bound of the number of mistakes made by Perceptron in learning this disjunction.

5. [6 points] Suppose we have a finite hypothesis space $\mathcal{H}$.

(a) [3 points] Suppose $|\mathcal{H}| = 2^{10}$. What is the VC dimension of $\mathcal{H}$?

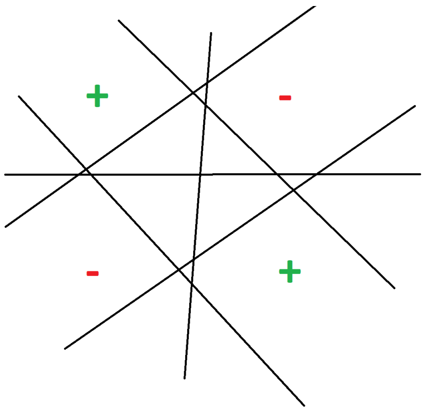
A hypothesis space of this size can represent any disjunction of up to 10 boolean variables. Therefore the VC dimension is 10.

(b) [3 points] Generally, for any finite $\mathcal{H}$, what is VC($\mathcal{H}$)?

VC($\mathcal{H}$) = log($|\mathcal{H}|$) for finite $\mathcal{H}$, generally.

6. [6 points] Prove that linear classifiers in a plane cannot shatter any 4 distinct points.

   The XOR function on 2 binary variables will give four points that are not linearly separable, and therefore cannot be shattered.



7. [**Bonus**] [10 points] Consider our infinite hypothesis space $\mathcal{H}$ are all rectangles in a plain. Each rectangle corresponds to a classifier — all the points inside the rectangle are classified as positive, and otherwise classified as negative. What is VC($\mathcal{H}$)?

2 Practice [60 points]

1. [2 Points] Update your machine learning library. Please check in your implementation of ensemble learning and least-mean-square (LMS) method in HW1 to your GitHub repository. Remember last time you created the folders "Ensemble Learning" and "Linear Regression". You can commit your code into the corresponding folders now. Please also supplement README.md with concise descriptions about how to use your code to run your Adaboost, bagging, random forest, LMS with batch-gradient and stochastic gradient (how to call the command, set the parameters, etc). Please create a new folder "Perceptron" in the same level as these folders.

2. We will implement Perceptron for a binary classification task — bank-note authentication. Please download the data "bank-note.zip" from Canvas. The features and labels are listed in the file "bank-note/data-desc.txt". The training data are stored in the file "bank-note/train.csv", consisting of 872 examples. The test data are stored in "bank-note/test.csv", and comprise of 500 examples. In both the training and testing datasets, feature values and labels are separated by commas.

   (a) [16 points] Implement the standard Perceptron. Set the maximum number of epochs T to 10. Report your learned weight vector, and the average prediction error on the test dataset.

   Learning Rate = 0.1

Learned **w** = [-4.14060202, -2.4885255, -2.63851262, -1.59607382]

Training Error: 0.0424311926606

Testing Error: 0.046

(b) [16 points] Implement the voted Perceptron. Set the maximum number of epochs T to 10. Report the list of the distinct weight vectors and their counts — the number of correctly predicted training examples. Using this set of weight vectors to predict each test example. Report the average test error.

Learning Rate = 0.1

Training Error: 0.0470183486239

Testing Error: 0.054

See `all_weights.txt` for all distinct weight vectors and their votes.

(c) [16 points] Implement the average Perceptron. Set the maximum number of epochs T to 10. Report your learned weight vector. Comparing with the list of weight vectors from (b), what can you observe? Report the average prediction error on the test data.

Learning Rate = 0.1

a: = [-32085.47564663041, -21283.06000449983, -19848.261108200008, -10992.40840545993]

Training Error: 0.0470183486239

Testing Error: 0.06

The weight vector **a** is roughly equal to the weighted sum of all weight vectors from voted perceptron, as is expected.

(d) [10 points] Compare the average prediction errors for the three methods. What do you conclude?

Standard perceptron achieved the best testing error followed by voted perceptron then averaged. This is likely because the standard version was able to converge somewhat in 10 epochs, while voted and averaged were not. It also appears that truncating individual guesses to just their sign in voted generalizes to the test data better than the averaged perceptron does.