CS 5350/6350: Machine Learning Spring 2021

Homework 3

Handed out: 11 Mar, 2021
Due date: 11:59pm, 25 Mar, 2021

- 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 hyperplane does have a margin. The distance from the hyperplane to a feature point is $dist(x_0, h) = y_i(u\top x_i)$

Then we simply need to find the distance of every feature point from the origin. The smallest distance will be our margin.

Remember that in the augmented feature space, there is a one in every feature vector.

$$dist(x_0, h) = y_i(u\top x_i)$$

$w =< 2, 3, -4 >$

$$u = \frac{w}{||w||}$$

$$u = \frac{< 2, 3, -4 >}{\sqrt{2^2 + 3^2 + 4^2}}$$

$$u = \frac{< 2, 3, -4 >}{\sqrt{29}}$$

$x_0 =< 1, 1, 1 >= y_i(u\top x_i) = (1)\frac{<2,3,-4>}{\sqrt{29}} \cdot < 1, 1, 1 > = \frac{1}{\sqrt{29}}$
$x_1 =< 1, -1, 1 >= y_i(u\top x_i) = (-1)\frac{<2,3,-4>}{\sqrt{29}} \cdot < 1, -1, 1 > = \frac{-5}{\sqrt{29}}$
$x_2 =< 0, 0, 1 >= y_i(u\top x_i) = (-1)\frac{<2,3,-4>}{\sqrt{29}} \cdot < 0, 0, 1 > = \frac{-4}{\sqrt{29}}$
$x_3 =< -1, 3, 1 >= y_i(u\top x_i) = (1)\frac{<2,3,-4>}{\sqrt{29}} \cdot < -1, 3, 1 > = \frac{3}{\sqrt{29}}$

x_0 is the closest to the separating hyperplane therefore x_0 is the margin.

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?
The data does have a separating hyperplane. The hyperplane $x_1 - x_2 = 0$

Dataset 2 does not have a margin because the hyperplane does not even separate the data into positive and negative values the correctly.

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.

To find the margin of the data set γ we need to find the shortest distance from the separating hyperplane to all points in the data set.
There are many potential separating hyperplanes however if we plot the points on the xy-plane, we can see that the unit vector $u =< 1, 1 >$ separates the dataset.
Also notice that every point in the dataset has the same distance to the separating hyperplane. Therefore if we find the distance from the hyperplane to any of the points in the dataset, we will get the margin γ of the dataset.

$dist(x_0, h) = y_i(u \top x_i)$
$dist(x_0, h) = (1)(< 1, 1 > \top < 1, 0 >)$
$dist(x_0, h) = 1 + 0$
$dist(x_0, h) = 1$
$\gamma = 1$

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.

We cannot calculate the margin for Dataset 4 because there isn't even a hyperplane that separates positive examples from negative examples.

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.

Given:

$$u \top w_t \geq t\gamma$$

If we replace u with w then this inequality turns into:

$$w \top w_t \geq t\gamma$$

The proof of this equation turns into:

$$w \top w_{t+1} = w \top w_t + y_i w \top x_i$$

$$w \top w_{t+1} \geq w \top w_t + \gamma$$

Everything remains the same we continue with the proof.

Next we look at the following inequality.

$$||w_t||^2 \leq tR$$

Notice that this inequality does not have u in it so we don't have to modify it to include a non unit vector w.

We then know that:

$$w \top w_t = ||w||||w_t||cos\theta$$

Notice that if we scale $||w_t||$ by $||w||$ then we know that:

$$||w||||w_t|| \geq ||w||||w_t||cos\theta$$

Because, $cos\theta$ will always be ≤ 1.

Also notice that we need to scale $R\sqrt{t}$, therefore we have:

$$||w||R\sqrt{t} \geq ||w||||w_t|| \geq w \top w_t \geq t\gamma$$

Finally we we can simply say:

$$||w||R\sqrt{t} \geq t\gamma$$

With simple algebra we can rewrite this to:

$$\left(\frac{||w||R}{\gamma}\right)^2 = t$$

(b) [3 points] Following (a), if we do NOT assume **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?

To keep the same upper bound of:

$$\left(\frac{R}{\gamma}\right)^2 = t$$

and by not assuming our separating vector is a unit vector, the γ inequality will look as follows:

$$y_i(\mathbf{u}^\top \mathbf{x}_i) \geq \gamma -> y_i(\mathbf{w}^\top \mathbf{x}_i) \geq \gamma$$

Where w is a separating non-unit vector.
We need to notice that in the inequality:
w is simply u scaled by a factor of a, therefore we can say that $w = au$
Now, notice that in the inequality: $y_i(\mathbf{w}^\top \mathbf{x}_i) \geq \gamma$
w is inner producted with x_i, therefore we need to scale down x_i by the same factor of a that we used to scale the u unit vector up to equal the separating vector w.

Therefore we finally have:

$$y_i(\mathbf{w}^\top \frac{\mathbf{x}_i}{a}) \geq \gamma$$

(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?

If we assume the only difference from this assumption and the original second assumpiton is that γ does not necessarily have to be greater than 0. Then this means that the perceptron algorithm has no upper bound. When $\gamma = 0$, the upper bound equation will look like:

$$t = \frac{R^2}{0}$$

Notice that 0 is in the denominator, therefore $t = \infty$

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

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

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.

To find the upper bound, we need to figure out what γ and R are for the dataset.

$R = ||x_i||$, where x_i is the feature vector furthest from the separating hyperplane.

Notice that the point furthest from the hyperplane is the point where every feature in the input feature vector has a value of 1. The described feature vector would look as follows:

$$f(x_1, x_2, \ldots, x_n) =< 1, 1, 1, ...n, 1 >$$

$$||x_i|| = \sqrt{1^2 + 1^2 ... n^2 + 1^2}$$

$$||x_i|| = \sqrt{n+1}$$

$$R = \sqrt{n+1}$$

Remember that we are in the augmented feature space so there is always a 1 added to every input feature vector to match the b in the separating weight vector.

Now to find the γ for the dataset:

$\neg x_1 \vee \neg \ldots \neg x_k \vee x_{k+1} \vee \ldots \vee x_{2k} = 1$ when:

$\neg x_1 + \neg \ldots \neg x_k + x_{k+1} + \ldots + x_{2k} \geq 1$

Therefore, the separating hyperplane is:

$$\neg x_1 + \neg \ldots \neg x_k + x_{k+1} + \ldots + x_{2k} - 1 = 0$$

Howevere, $\gamma > 0$ so our real hyperplane will be:

$$\neg x_1 + \neg \ldots \neg x_k + x_{k+1} + \ldots + x_{2k} - \frac{1}{2} = 0$$

This hyperplane normalized will be:

$$u = \frac{1}{\sqrt{2k + \frac{1}{4}}} < -1, \ldots - x_k, 1, \ldots, x_{2k}, -\frac{1}{2} >$$

And the input vector x:

$$x = \neg x_1 + \neg \ldots \neg x_k + x_{k+1} + \ldots + x_{2k} + 1$$

Notice that point closest to the separating hyperplane is when every attribute is 0 execept 1 or whene all values are zero.

Finally, γ will be:

$$\gamma = y_i u \top x = \frac{\frac{1}{2}}{\sqrt{2k + \frac{1}{4}}}$$

After calculating R and γ for the dataset, the upper bound is :

$$t \leq \left(\frac{R}{\gamma}\right)$$

$$t \leq \left(\frac{\sqrt{n+1}}{\frac{\frac{1}{2}}{\sqrt{2k+\frac{1}{4}}}}\right)^2$$

$$t \leq 2(n+1)(2k + \frac{1}{4})$$

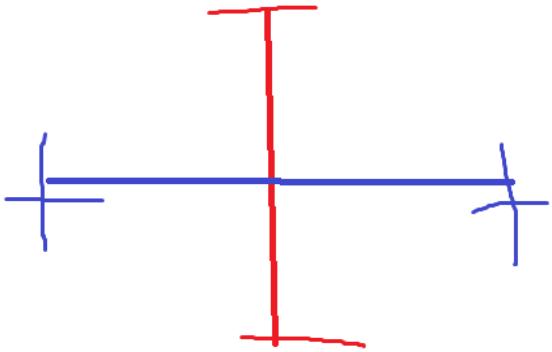
5. [6 points] Suppose we have a finite hypothesis space $\mathcal{H}$.

   (a) [3 points] Suppose $|\mathcal{H}| = 2^{10}$. What is the VC dimension of $\mathcal{H}$?
   VC of $|\mathcal{H}| = log(|\mathcal{H}| = log(2^{10}) = 3.01$

   (b) [3 points] Generally, for any finite $\mathcal{H}$, what is VC($\mathcal{H}$) ?
   Using the definition of VC dimensions in a finite space:
   The VC dimension of hypothesis space H over instance space X is the size of the largest finite subset of X that is shattered by H
   That is if we can at least find a any subset of size d that can be shattered, then VC(H) ¿= d.

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

   Imagine there are 2 positive examples in a straight horizontal line and 2 negative examples in a vertical line that intersects the horizontal line:



To separate these examples, you must find a separating hyperplane that does not cross the blue line or the red line. It is impossible to separate these points without crossing either of the lines therefore the VC dimensions = 3

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.

The complete data is printed to the console on the python file: StandardPerceptron.

Unit vector separating hyperplane:

[-0.6556223971796259, -0.48948690948971185, -0.5186570271695159, -0.2481062794899229]

Success percent: 0.95

(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.

The following data is simply the last 6 examples before the perceptron converged. I added these examples so I can compare them on question c.

The complete data is printed to the console on the python file: VotedPerceptron.

———— Success percent: 0.954————

m = 1120

Weighted vector:

[-0.6611471690026989, -0.49161118846291607, -0.5180039868679581, -0.22994505840895174]

Correct Predictions: 3

m = 1121

Weighted vector:

[-0.6598463969533983, -0.48837666946810054, -0.5220621886222617, -0.2313915131498944]

Correct Predictions: 5

m = 1122

Weighted vector:

[-0.6596212134572851, -0.4883369898943353, -0.5224300391104921, -0.23128703658676666]

Correct Predictions: 1

m = 1123

Weighted vector:

[-0.6583420898361476, -0.48491572454340975, -0.5265002392814951, -0.2328946776490226]

Correct Predictions: 21

m = 1124

Weighted vector:

[-0.6586458488955418, -0.48616917029337753, -0.525258367939705, -0.2322258178976605]

Correct Predictions: 4

m = 1125

Weighted vector:

[-0.6587365164327136, -0.4856246232680955, -0.525960103357338, -0.23151867499221715]

Correct Predictions: 9

Success percent: 0.954

(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.

What I can observe is that towards the end of the voted perceptron, the weigthed vectors get closer and closer to the average separating hyperplane. This means that Both voted perceptron are roughly equivalent but the average separating hyperplane has the advantage of being much faster because it does not have to keep track of all of the previous weigthed vectors.

The complete data is printed to the console on the python file: AveragePerceptron.

Average separating hyperplane a:
[-0.7213766109841521, -0.49133765788479156, -0.4220215146036088, -0.24515491486197616]
Success percent: 0.938

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

The standard perceptron is trying to find a separating hyperplane that will completely separate the data set given $t = \left(\frac{R}{\gamma}\right)^2$ trails. This would work however real data has noise in it so there doesn't exist a true separating hyperplane for most real data. This means that the percetron algorithm will be in an infinite loop unless we stop it. The final separating hyperplane for the standard perceptron is calculated after stopping the algorithm after 10 epochs. This will be our standard to variants of the perceptron algorithm.

The voted perceptron algorithm tries to predict examples by using a sort of ensemble learning. The voted weighted voted prediction will always be better than the standard perceptron algorithm because it doesn't only take into account the final w separating hyperplane, but also uses every single previous w to get a more robust prediction. We can see this idea in our results as the voted perceptron success percent (.954) > the standard perceptron (.95)

Finally the average perceptron algorithm is much more simplified version of the voted perceptron algorithm. The average perceptron tries to emulate the voted perceptron algorithm but from the calculated results, we can see that the average perceptron prediction success is slightly lower (.938) than the voted perceptron (.954). The average perceptron isn't outshined by the voted perceptron because the voted perceptron is incredibly slow if the list of features or the number of examples increases.