CS 5350/6350: Machine Learning Spring 2019

Homework 3

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

Author: Cade Parkison
Github: https://github.com/c-park/Machine-Learning-Library
Note:

To get the results of the Perceptrion algorithm, you may run "run.sh", "test.py", or you can see the results in the perceptron.ipynb notebook file. "perceptron.py" is the file that contains all three algorithms.

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 indeed have a margin for the dataset becase the data is linearly separable. The distances to the hyperplane are now computed for each example in the dataset, and the margin is taken as the smallest distance.

$$\gamma = \min_{\mathbf{x}_i, y_i} \frac{y_i |b + w_1 x_{1i} + w_2 x_{2i}|}{\sqrt{w_1^2 + w_2^2}}$$

Finding minimum distance:

$$\begin{aligned} d_1 &= \frac{|b + w_1 x_1 + w_2 x_2|}{\sqrt{w_1^2 + w_2^2}} \\ &= \frac{|-4 + 2(1) + 3(1)|}{\sqrt{2^2 + 3^2}} \\ &= \frac{\sqrt{13}}{13} \\ &= 0.27735 \end{aligned}$$

$$\begin{aligned} d_2 &= \frac{|-4 + 2(1) + 3(-1)|}{\sqrt{2^2 + 3^2}} \\ &= 1.3868 \end{aligned}$$

$$\begin{aligned} d_3 &= \frac{|-4 + 2(0) + 3(0)|}{\sqrt{2^2 + 3^2}} \\ &= 1.1094 \end{aligned}$$

$$\begin{aligned} d_4 &= \frac{|-4 + 2(-1) + 3(3)|}{\sqrt{2^2 + 3^2}} \\ &= 0.8321 \end{aligned}$$

Example 1, $\mathbf{X}_1 = [1, 1]$, is the closest to the hyperplane, so this point will give us a margin of:

$$\begin{aligned} \gamma &= \frac{y_1|b + w_1 x_{11} + w_2 x_{21}|}{\sqrt{w_1^2 + w_2^2}} \\ &= \frac{1|-4 + 2(1) + 3(1)|}{\sqrt{2^2 + 3^2}} \\ &= 0.27735 \end{aligned}$$

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?

This new dataset is not linearly separable by the hyperplane, the 5th data point is classified as 1 but lies on the wrong side of the hyperplane.

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.

Yes, it is possible to calculate this margin because the data is linearly separable. The best hyperplane for this data is $-x_1 + x_2 = 0$, and this gives the maximum margin possible of $\sqrt{\frac{1}{2}}$.

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.

This data is not linearly separable, so it is not possible to calculate its margin in this way.

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 necessarily a unit vector, then the mistake bound would change to $\left(\frac{||\mathbf{u}||R}{\gamma}\right)^2$

(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?

The two inequalities in the second assumption would change to the following:

$$\frac{\gamma}{||\mathbf{u}||} > 0$$

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

(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?

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.

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 $\mathcal{H}$?

The VC dimension of H is:

$$VC(2^{10}) \leq log_2(2^{10}) = 10$$

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

In general for finite H, $VC(H) \leq log_2|H|$.

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

As a proof by contradiction, we can take the binary XOR function in two dimensions. The four points associated with this are $[0,0][0,1][1,0]and[1,1]$. Both [0, 0] and [1, 1] are labeled 1, while the other two are labeled 0. There is no such linear classifier that would shatter these 4 points. Because of this, the proof is complete.

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($\mathcal{H}$)?

A set of four points arranged in a diamond can be shattered by axis-aligned rectangles. That is, there is an axis-aligned rectange that contains any given subset of the points, but not the rest. This means that VC($\mathcal{H}$) $\geq$ 4. If we supposed we had 5 points, the shattering must allow all 5 points to be selected and also 4 points to be selected without the 5th. If our rectangle just barely enclosed all 5 points, where 4 points are on the edges of the rectangle, the 5th point must either lie on an edge or inside the rectangle. This would prevent selecting only 4 points without the fifth. This means that VC($\mathcal{H}$) = 4

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.

   I managed to get very good results when I implemented the standard Perceptron. When using a maximum number of epochs of 10, my average prediction errors were very small every time I ran the algorithm. The learned weight vector after 10 epochs was the following:

$$\mathbf{w} = [0.53, -0.5881, -0.3583, -0.3462, -0.0775]$$

   When evaluating this weight vector on the test dataset, the average prediction error was 4.2%. For the test dataset of 500 examples, this means it only predicted the wrong label 21 times out of 500. Even when I increased the number of epochs to 100, the result improvements were negligible. However, increasing the epochs does make the weight vector change considerably, but the resulting training error remains the same. An interesting point is that when running this algorithm multiple times, the resulting errors fluctuate up and down. To account for this, I ran my algorithm 100 times and took the average of the errors for more consistent results. The average test error of the 100 standard perceptrons was 2.31%

   (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 Voted Perecptron algorithm gave similar, but slightly better results. The variance in the errors each time the algorithm is ran is greatly decreased. The list of distinct weight vectors contained 247 weights. Shown below are a few of the weight vectors and their corresponding counts of correctly predicted training examples.

$$\mathbf{w}_1 = [0, -0.0615, -0.0364, -0.02500, 0.0504]$$
$$C_1 = 8$$

$$\mathbf{w}_2 = [0.04, -0.1217723, -0.1124037, -0.10197537, -0.05279357]$$
$$C_2 = 17$$

$$\mathbf{w}_3 = [0.14, -0.14897254, -0.155197, -0.16077387, -0.04530147]$$
$$C_3 = 30$$

$$\mathbf{w}_4 = [0.41, -0.45830366, -0.3029646, -0.31770652, -0.02693296]$$
$$C_4 = 111$$

$$\mathbf{w}_5 = [0.51, -0.59840946, -0.4135217, -0.29767352, -0.12966562]$$
$$C_5 = 1$$

$$\mathbf{w}_6 = [0.53, -0.52431746, -0.4178887, -0.40093652, -0.11616962]$$
$$C_6 = 78$$

The six weights and counts above are only a few of the 247 that my algorithm produced, the rest were omitted for clarity. As can be seen above, there is a large distribution of counts even for small changes in the weights. Using the whole set of 247 weights and counts, I can accurately predict the outcomes of the test data. Running my algorithm 100 times and averaging the results, I get the following average test error:

$$\mathbf{e}_{mean} = 1.38\%$$

This corresponds to mislabeling about 7 out of the 500 testing examples, which is an improvement over the standard Perceptron algorithm.

(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.

The average Perceptron performed very well. The learned weight vector is shown below:

$\mathbf{w} = [3437.8202, -3980.81967769, -2591.51978412, -2681.63009231, -752.43529268]$

As can be seen, the weight vector is much larger due to the $\mathbf{a} = \mathbf{a} + \mathbf{w}$ part of the algorithm where the weights are all summed. This has the effect of making the error rates much more stable each time the algorithm is ran. With the above weights, I get an average prediction error of 1.386% on the test data.

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

The average prediction error of voted and average Perceptron were almost identical and extremely low. The standard Perceptron error was only 0.93% worse. Both voted and standard Perceptron suffer from high variance in errors each time the algorithm is ran, where average Perceptron is much more stable.