CS 5350/6350: Machine Learning Fall 2021

Homework 3

Handed out: 19 Oct, 2021
Due date: 11:59pm, 2 Nov, 2021

1 Paper Problems [36 points + 15 bonus]

1. [8 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).

   $dist(\mathbf{x}_i, h) = \frac{|\mathbf{w}^\top \mathbf{x}_i + b|}{||\mathbf{w}||}$

x_1	x_2	label	distance
1	1	1	$\frac{1}{\sqrt{13}}\|2*1+3*1-4\| = \frac{1}{\sqrt{13}}$
1	-1	-1	$\frac{1}{\sqrt{13}}\|2*1-3*1-4\| = \frac{5}{\sqrt{13}}$
0	0	-1	$\frac{1}{\sqrt{13}}\|2*0+3*0-4\| = \frac{4}{\sqrt{13}}$
-1	3	1	$\frac{1}{\sqrt{13}}\|2*-1+3*3-4\| = \frac{3}{\sqrt{13}}$

Table 1: Dataset 1, with distance calculations

(a) [4 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?)

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

Table 2: Dataset 2

Answer

Figure 1 demonstrates that this hyperplane does separate the data. Table 1 (updated with Distance calculations) gives the distance between each data point and the hyperplane. The minimum distance is $\frac{1}{\sqrt{13}}$, which means that is the margin.

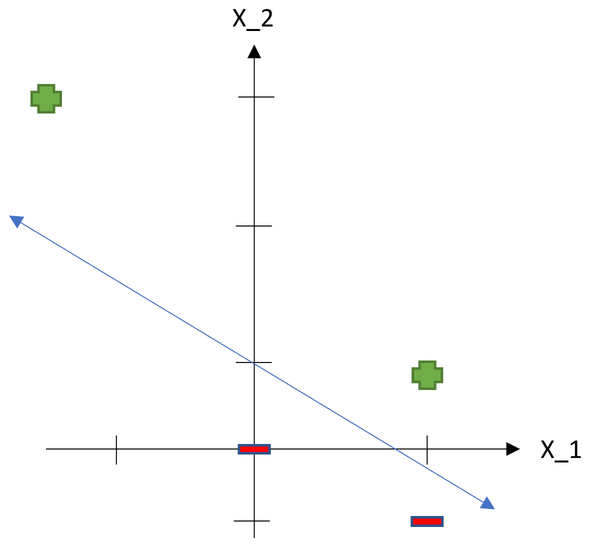

Figure 1: Points and linear classifier plotted for Part 1 Question 1a.

(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?

Answer

Figure 2 plots the hyperplane and the data points. Clearly, the hyperplane does not separate the data correctly, so there can be no margin.

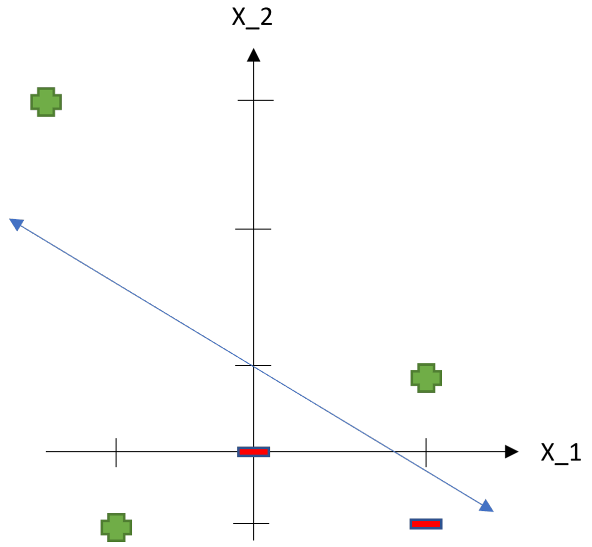

Figure 2: Points and linear classifier plotted for Part 1 Question 1b.

2. [8 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	distance
-1	0	-1	$\frac{1}{\sqrt{2}}\lvert -1+0\rvert = \frac{1}{\sqrt{2}}$
0	-1	-1	$\frac{1}{\sqrt{2}}\lvert 0-1\rvert = \frac{1}{\sqrt{2}}$
1	0	1	$\frac{1}{\sqrt{2}}\lvert 1+0\rvert = \frac{1}{\sqrt{2}}$
0	1	1	$\frac{1}{\sqrt{2}}\lvert 0+1\rvert = \frac{1}{\sqrt{2}}$

Table 3: Dataset 3, with distance calculations

(a) [4 points] Given the dataset in Table 3, can you calculate its margin? If you cannot, please explain why.

Answer

Figure 3 plots Dataset 3 and a potential hyperplane that would separate the data. We can calculate R, which is the furthest point from the origin, but all points are equidistant from the origin, so they all have the same R. We see that $R = \sqrt{0^2 + 1^2} = 1$. If we use a simple hyperplane that goes through the origin and has the same distance to all four points, say $x_1 + x_2 = 0$, we see that $||\mathbf{w}|| = \sqrt{1^2 + 1^2} = \sqrt{2}$. Therefore, the margin, $\gamma = \frac{1}{\sqrt{2}}$.

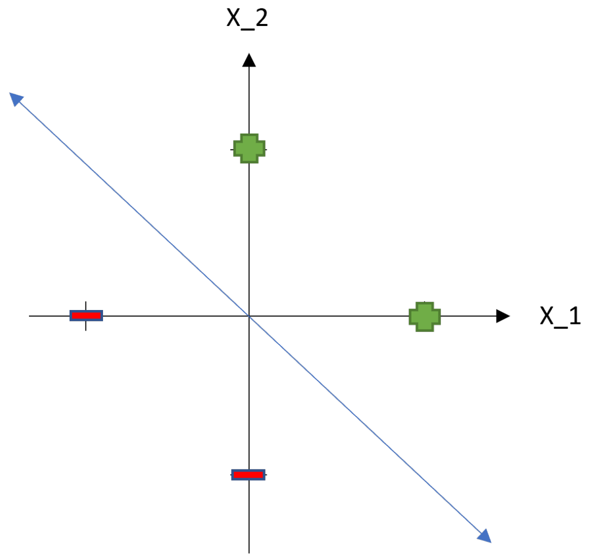

Figure 3: Points and likely linear classifier fitted to data set for Part 1 Question 2a.

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.

Answer

Figure 4 plots Dataset 4. Clearly, the data is not linearly separable, so we cannot calculate any margin.

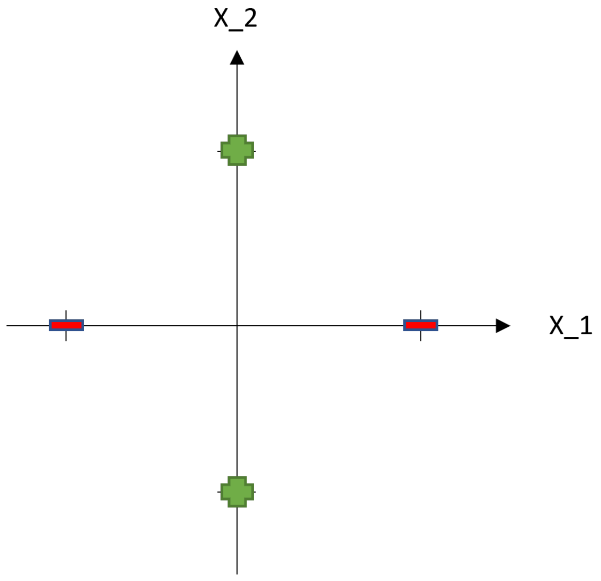

Figure 4: Points plotted for Part 1 Question 2b.

3. [**Bonus**] [5 points] Let us review the Mistake Bound Theorem for Perceptron discussed in our lecture. 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.

Answer

Comparing to the proof in the lecture slides (CLICK HERE), the only difference is that $\mathbf{u}$ is not *necessarily* a unit vector. We will note that normally, the upper bound is $\frac{R^2}{\gamma^2}$.

We will pick up at part 3/3 of the proof seen in the lecture slides. We know that $\mathbf{u}^\top \mathbf{w}_t \geq t\gamma$ and $tR^2 \geq ||\mathbf{w}_t||^2$, the latter of which can be rewritten as $\sqrt{t}R \geq ||\mathbf{w}_t||$. We can also rewrite the dot product $\mathbf{u}^\top \mathbf{w}$ as $||\mathbf{u}||||\mathbf{w}_t||cos\theta$, where θ is the angle between $\mathbf{u}$ and $\mathbf{w}_t$, and $cos\theta < 1$. The interesting case for us is when $||\mathbf{u}||||\mathbf{w}_t||cos\theta > ||\mathbf{w}_t||$, which means that $||\mathbf{u}|| > \frac{1}{cos\theta}$.

Obviously, based on Proof 3/3 from the lecture slides, if $||\mathbf{u}|| \leq \frac{1}{cos\theta}$, then we get $\frac{R^2}{\gamma^2}$.

If $||\mathbf{u}|| > \frac{1}{cos\theta}$,

$$=> ||\mathbf{u}||||\mathbf{w}_t||cos\theta \geq t\gamma$$

$$=> ||\mathbf{w}_t|| \geq \frac{t\gamma}{||\mathbf{u}||cos\theta}$$

From here, we add in that $\sqrt{t}R \geq ||\mathbf{w}_t||$ to yield,

$$R\sqrt{t} \geq ||\mathbf{w}_t|| \geq \frac{t\gamma}{||\mathbf{u}||cos\theta}$$

$$=> R^2t \geq \frac{t^2\gamma^2}{||\mathbf{u}||^2cos^2\theta}$$

$$=> ||\mathbf{u}||^2cos^2\theta\frac{R^2}{\gamma^2} \geq t$$

So, it is the same value as the upper bound for when $\mathbf{u}$ is a unit vector, but scaled by the square of the norm of $\mathbf{u}$ and the cosine of its angle with $\mathbf{w}_t$. Obviously, this means it is simpler to compute the upper bound if we just keep $\mathbf{u}$ as a unit vector.

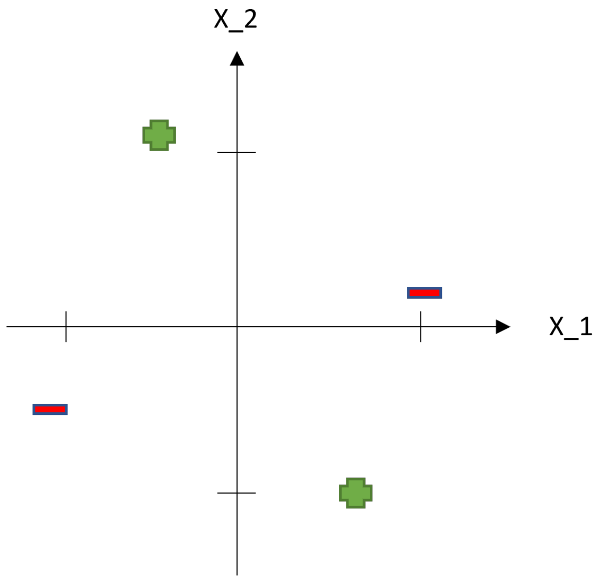

Figure 5: Proof that linear classifiers cannot shatter 4 points.

4. [10 points] We want to use Perceptron to learn a disjunction as follows,

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

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

Answer

Figure 5 shows a case with four points that cannot be shattered by any linear classifier.

6. [**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($\mathcal{H}$)?

   Answer

   Figure 6 shows how rectangles can shatter 1 to 2 points, because an infinitesimally small rectangle can be drawn over any single point. Assuming that not all 3 points are co-linear, we also see that 3 points can be shattered because we can draw a rectangle around any 2 adjacent points. For a similar reason (under similar conditions where any 3 points cannot be co-linear) we can also shatter any 4 points. However, we cannot shatter 5 points, as indicated when $d = 5$. So, because $\mathcal{H}$ can shatter 3 points but not 4, we can conclude VC($\mathcal{H}$) $= 4$.

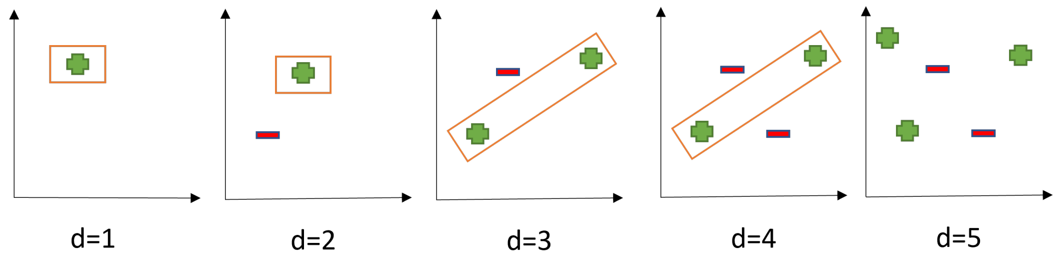

Figure 6: Proof that rectangles in a plane can shatter 1 to 4 points, but cannot shatter 5 points.

2 Practice [64 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.

   Click here for the repository for HW3. It is not the same as the repository for HW1 or HW2.

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.

Answer

WEIGHT VECTOR

WaveletVariance -5.777450

WaveletSkew -3.204904

WaveletCurtosis -4.753204

ImageEntropy -0.603608

MODEL_BIAS -5.300000

ERRORS

TRAIN ERROR: 0.04243

TEST ERROR: 0.05000

(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.

Answer

Figure 7 shows how the weight vector changes with each update, and Figure 8 shows the votes assigned to each weight vector. The votes correspond to how many data points that each weight vector managed to accurately predict. The votes can be seen in Section 3.1, and the updated weights array can be seen in Section 3.2. The CSV's can also be found in the GitHub repo, though the convergence of weights CSV is formatted with ' & ' instead of ',' and lacks a header. This was done to make it easier to copy values into a table in this document (see Appendix).

CLICK HERE FOR VOTES CSV LINK

CLICK HERE FOR WEIGHTS CONVERGENCE CSV LINK

TRAIN ERROR: 0.01261

TEST ERROR: 0.01400

(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.

Answer

WEIGHT VECTOR WaveletVariance -37108.434439

WaveletSkew -25185.487162

WaveletCurtosis -25383.376175

ImageEntropy -7638.087261

MODEL_BIAS -34243.400000

ERRORS

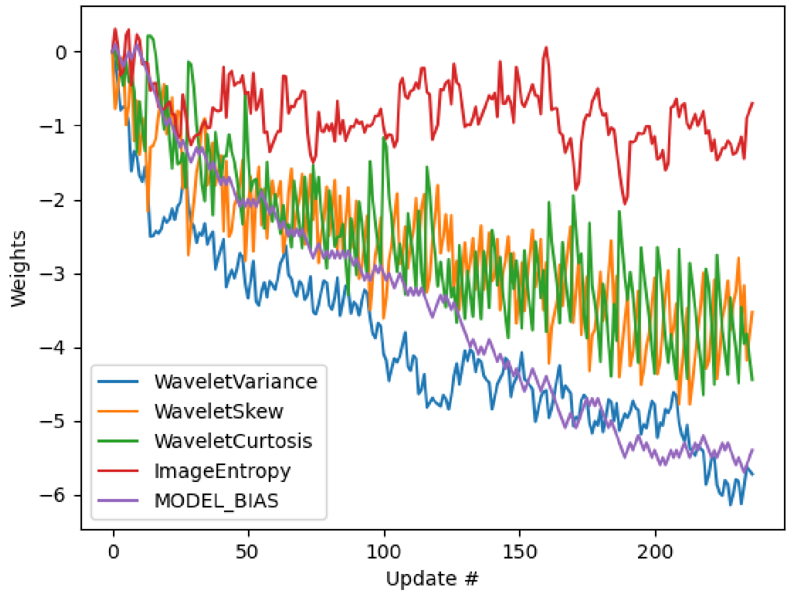

Figure 7: Weight vector change per update.

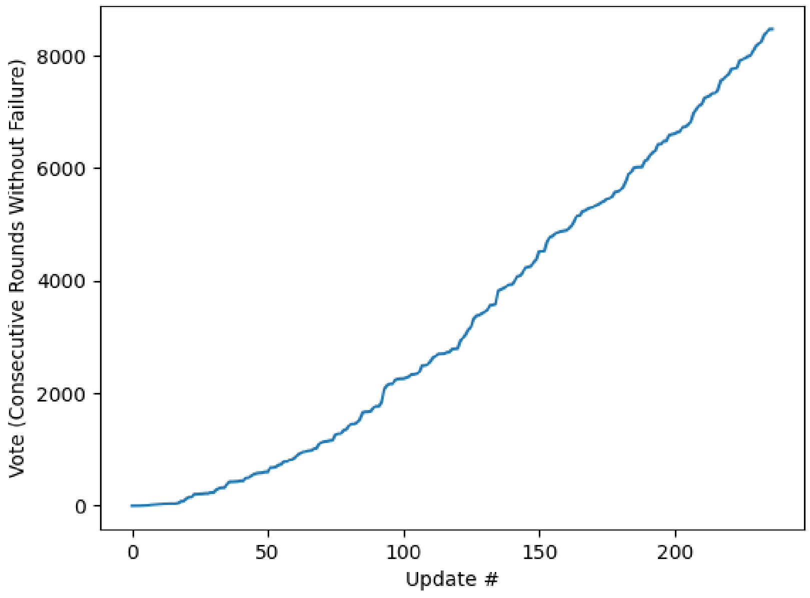

Figure 8: Votes for each weight vector.

TRAIN ERROR: 0.01491

TEST ERROR: 0.01400

The primary observation I would make is that the final weight vector for the Average Perceptron is massively larger in scale than any single weight vector in (b), but it would make sense because the Voted Perceptron will multiply each weight vector by its vote, which in some cases is as high as 8000 (see Figure 8). However, in terms of scale relative to each other, the individual weights seem to be on the same relative scale to each other as they were in (a) (and individually in (b)). For example, ImageEntropy is about 10 times smaller than the other weights in both the Average Perceptron, Standard Perceptron, and individually in the Voted Perceptron.

(d) [14 points] Compare the average prediction errors for the three methods. What do you conclude?

Obviously, the standard Perceptron is the worst performer. But, Voted and Average Perceptrons perform about the same. Because the Voted Perceptron stores more data, the Average Perceptron is likely a more performative model at higher epochs since it only needs to compare to evaluate with one weight vector. In any case, increasing the margin certainly has a significant effect on the error rate.

3 Appendix

3.1 Part 2 Question 2b VOTES

The following array shows the number of votes per updated weight array. The array is 237 values long.

[1 1 3 3 5 5 10 18 22 26 32 34 36 38 39 39 39 48 81 83 124 155 155 209 209 214 216 222 222 236 236 284 307 320 320 387 427 427 430 433 443 443 496 499 538 562 577 584 587 600 600 680 681 694 727 736 787 788 812 821 857 898 931 957 963 978 984 1013 1021 1102 1130 1142 1150 1162 1169 1262 1281 1282 1341 1359 1425 1452 1455 1486 1530 1657 1670 1670 1678 1746 1767 1769 1842 2074 2137 2166 2166 2237 2255 2260 2260 2275 2295 2329 2335 2348 2382 2492 2493 2511 2565 2635 2661 2699 2703 2705 2725 2734 2783 2789 2792 2930 2984 3040 3134 3183 3326 3377 3388 3413 3446 3483 3559 3567 3583 3820 3844 3867 3900 3929 3931 3989 4075 4088 4144 4230 4246 4259 4318 4371 4518 4521 4525 4695 4771 4788 4837 4858 4875 4884 4895 4923 4970 5061 5151 5153 5232 5244 5273 5295 5310 5334 5358 5390 5408 5450 5464 5489 5571 5583 5610 5657 5759 5891 5927 6005 6016 6017 6019 6130 6148 6231 6284 6310 6426 6428 6478 6481 6587 6602 6610 6637 6652 6727 6736 6775 6820 6971 7051 7105 7130 7246 7264 7283 7331 7334 7387 7553 7581 7628 7674 7763 7770 7782 7910 7931 7951 7988 8001 8079 8169 8210 8243 8360 8415 8470 8471]

3.2 Part 2 Question 2b Weights

Update No.	WaveletVariance	WaveletSkew	WaveletCurtosis	ImageEntropy	MODEL_BIAS
0	0.000	0.000	0.000	0.000	0.000
1	-0.193	-0.776	0.017	0.308	0.100
2	-0.409	-0.473	-0.224	0.065	0.000
3	-0.795	-0.121	-0.263	-0.321	-0.100
4	-0.744	-0.073	-0.461	-0.263	-0.200
5	-0.771	-0.995	-0.132	0.221	-0.100
⋮	⋮	⋮	⋮	⋮	⋮
231	-5.831	-2.799	-4.487	-1.273	-5.500
232	-6.126	-3.626	-3.461	-1.156	-5.600
233	-5.886	-3.170	-3.959	-1.446	-5.700
234	-5.622	-4.184	-3.826	-0.899	-5.600
235	-5.674	-3.858	-4.135	-0.801	-5.500
236	-5.726	-3.531	-4.444	-0.702	-5.400

Table 5: Convergence of Weights as calculated in Part 2 Question 2b. Values removed to show beginning and ending values.