Week 9 | Solve With Instructors

Q1) We know that the decision boundary learned by the perceptron is a line in $\mathbb{R}^2$. We also observed that it divides the entire space of $\mathbb{R}^2$ into two regions, suppose that the input vector $\mathbf{x} \in \mathbb{R}^5$, then the perceptron decision boundary will divide the whole $\mathbb{R}^5$ space into how many regions?

- ◯ 2
- ◯ 3
- ◯ 4
- ◯ 5

Solution: The perceptron decision boundary is determined by the hyperplane $\mathbf{w}^T\mathbf{x} = 0$ where $\mathbf{w} \in \mathbb{R}^n$ is the weight vector and $\mathbf{x} \in \mathbb{R}^n$ is the input vector.

In $\mathbb{R}^5$, this hyperplane is a 4-dimensional subspace that divides the 5-dimensional space into **two regions**:

1. The region where $\mathbf{w}^T\mathbf{x} > 0$, corresponding to one class.
2. The region where $\mathbf{w}^T\mathbf{x} < 0$, corresponding to the other class.

Thus, **the perceptron decision boundary divides the space $\mathbb{R}^5$ into exactly two regions**.

Q2) Consider the following table, where x_1 and x_2 are features (packed into a single vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$) and y is label

x_1	x_2	y
0	0	-1
0	1	1
1	0	1
1	1	1

Suppose that the perceptron model is used to classify the data points. Suppose further that the weight $\mathbf{w}$ are initialized to $\mathbf{w} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The following rule is used for classification,

$$\widehat{y} = \begin{cases} 1 & \mathbf{w}^T\mathbf{x} > 0 \\ -1 & \mathbf{w}^T\mathbf{x} \leqslant 0 \end{cases}$$

The perceptron learning algorithm is used to update the weight vector $\mathbf{w}$. Then, how many times the weight vector $\mathbf{w}$ will get updated during the entire training process? Cycle through the data-points in the given order.

Solution: The perceptron update rule says for any iteration, if the model makes a mistake in prediction, then update the weight vector as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{x}_i y_i$$

where $(\mathbf{x}_i, y_i)$ is a mistake with respect to $\mathbf{w}^{(t)}$.

Data points	True label	Predicted label ($\widehat{y}$)	Update ?
$\begin{bmatrix}0 & 0\end{bmatrix}^T$	-1	$\begin{bmatrix}1 & 1\end{bmatrix}^T\begin{bmatrix}0\\0\end{bmatrix} = 0, \widehat{y} = -1$	No
$\begin{bmatrix}0 & 1\end{bmatrix}^T$	1	$\begin{bmatrix}1 & 1\end{bmatrix}^T\begin{bmatrix}0\\1\end{bmatrix} = 1, \widehat{y} = 1$	No
$\begin{bmatrix}1 & 0\end{bmatrix}^T$	1	$\begin{bmatrix}1 & 1\end{bmatrix}^T\begin{bmatrix}1\\0\end{bmatrix} = 1, \widehat{y} = 1$	No
$\begin{bmatrix}1 & 1\end{bmatrix}^T$	1	$\begin{bmatrix}1 & 1\end{bmatrix}^T\begin{bmatrix}1\\1\end{bmatrix} = 2, \widehat{y} = 1$	No

Hence, the weight vector will not update.

Q3) Which of the following options are correct with respect to Perceptron and Logistics regression algorithm?

A) The perceptron algorithm is robust to outliers because it only updates weights for misclassified points.
B) Logistic regression assumes that the data is perfectly linearly separable for effective training.
C) For a linearly separable dataset, the perceptron algorithm always makes $\frac{R^2}{\gamma^2}$ mistakes where the R and γ have the same meaning as it was in the lecture.
D) Depending on how you cycle through the data-points, perceptron may return a different weight vector.
E) In perceptron, as the margin decreases, the number of updates increases.
F) The logistics regression uses cross entropy loss.
G) The perceptron algorithm guarantees that the learned weights will be optimal for linearly separable data.
H) The gradient of the logistic regression loss with respect to the weights $\mathbf{w}$ is proportional to the difference between predicted probabilities and the actual labels y
I) For logistic regression, as $\mathbf{w}^T\mathbf{x} \to \infty$, the predicted probability tends to 0.

Solution:

A) **Incorrect**. The Perceptron algorithm is not robust to outliers. While it only updates weights for misclassified points, if an outlier is misclassified, the perceptron will adjust the weights significantly for that outlier, leading to poor generalization. Outliers can heavily influence the model in the case of Perceptron.

B) **Incorrect.** Logistic regression does not require the data to be linearly separable. It can still be trained on non-separable data, unlike the Perceptron, which is guaranteed to work only if the data is linearly separable. Logistic regression optimizes a probabilistic cost function (cross-entropy loss), which works even in the presence of noisy data.

C) **Incorrect.** This is an upper bound on the number of mistakes. The algorithm can converge before too!

D) **Correct.** The Perceptron algorithm is sensitive to the order of the data points. If you present the data in a different order, the algorithm may end up with a different weight vector, even though it will always converge to a solution if the data is linearly separable. This is because the updates happen iteratively for misclassified points, and the update sequence can affect the final weight vector.

E) **Correct.** We know that for a linearly separable dataset, perceptron makes at most $\frac{R^2}{\gamma^2}$ mistakes. Now, as the margin (γ) decreases, $\frac{R^2}{\gamma^2}$ increases that is the number of mistake that can happen increases, hence the number of updates increases.

F) **Correct.** Logistic regression uses cross-entropy loss (also known as log loss) as its objective function. The loss function measures the difference between the predicted probabilities and the true binary labels, and the goal is to minimize this loss during training.

G) **Incorrect.** The Perceptron algorithm does not guarantee optimal weights. It will eventually converge to a solution if the data is linearly separable, but the solution may not be unique or optimal.

H) **Correct.** The gradient of the logistic regression loss (cross-entropy loss) with respect to the weights $\mathbf{w}$ is indeed proportional to the difference between the predicted probabilities and the actual labels. Specifically, the gradient is $\sum_{i=1}^{n}(y_i - \widehat{y})\mathbf{x}_i$.

I) **Incorrect.** For logistics regression, as $\mathbf{w}^T\mathbf{x} \to \infty$, the predicted probability tends to 1.

Q4) Consider a logistic regression model that is trained on videos to detect objectionable content. Videos with objectionable content belong to the positive class (label 1). Harmless videos belong to the negative class (label 0).

A good detector should be able to correctly identify almost all videos that are objectionable. If it incorrectly classifies even a single video that has inappropriate content in it, that could have serious consequences, as millions of people might end up watching it. In this process the detector may classify some harmless videos as belonging to the positive class. But that is a price we are willing to pay.

How should we choose the threshold (for inference) of this logistic regression model?

A) The threshold should be a low value.
B) The threshold should be a high value.
C) The performance of the classifier is independent of the threshold.

Solution:

To detect objectionable content, the threshold should be set low to minimize the chance of predicting objectionable video to harmless video, ensuring almost all objectionable videos are identified. While this increases harmless videos flagged as objectionable, it is acceptable since the consequences of missing objectionable content are severe.

$Q5$) If we use gradient ascent to find (x, y) such that $f(x, y) = 3 + x + y - x^2 - y^2$ is maximum, then after how many iterations will the algorithm stop? Use $(2, 2)$ as initialization point and 0.5 as learning rate.

ANSWER : 2

$\nabla f(x, y) = (1 - 2x, 1 - 2y)$

$w = (2, 2)$

At $(t + 1)^{th}$ iteration,

$$w_{t+1} = w_t + \eta \nabla f(w_t)$$

ITERATION 1

$$w_1 = w_0 + \eta \nabla f(w_0)$$
$$w_1 = (2, 2) + (0.5) \nabla f(2, 2)$$
$$w_1 = (2, 2) + (0.5) \times (-3, -3)$$
$$w_1 = (0.5, 0.5)$$

ITERATION 2

$$w_2 = w_1 + \eta \nabla f(w_1)$$
$$w_2 = (0.5, 0.5) + (0.5) \nabla f(0.5, 0.5)$$
$$w_2 = (0.5, 0.5) + (0.5) \times (0, 0)$$
$$w_2 = (0.5, 0.5)$$

$w_2 = w_1$, converged after two iterations.

$Q6$) For which dataset, is the perceptron algorithm likely to take less number of iterations to converge?

FIGURE 1

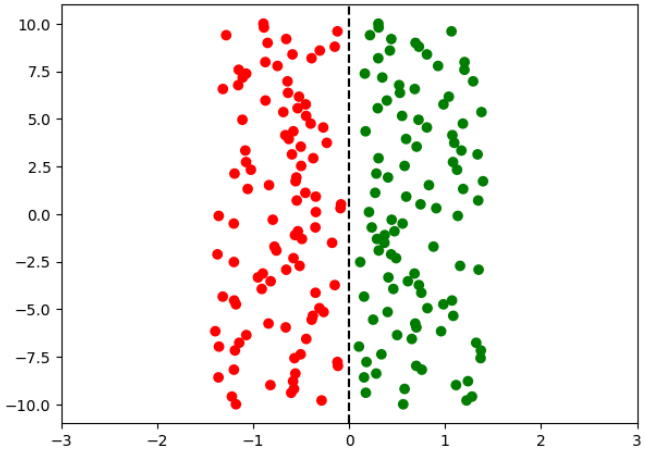
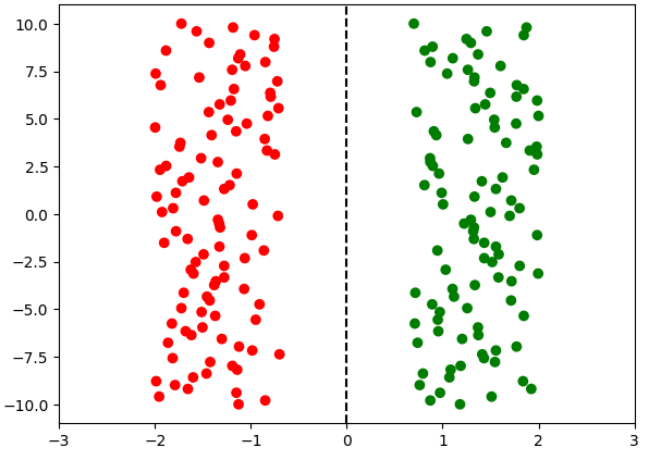


FIGURE 2



ANSWER : FIGURE 2

Figure 1 took 2367 rounds to converge, while Figure 2 took 264 rounds to converge.

REASON

Figure 1 has a lesser gamma margin when compared to Figure 2. Any hyper-plane within the gamma margin can be a boundary.

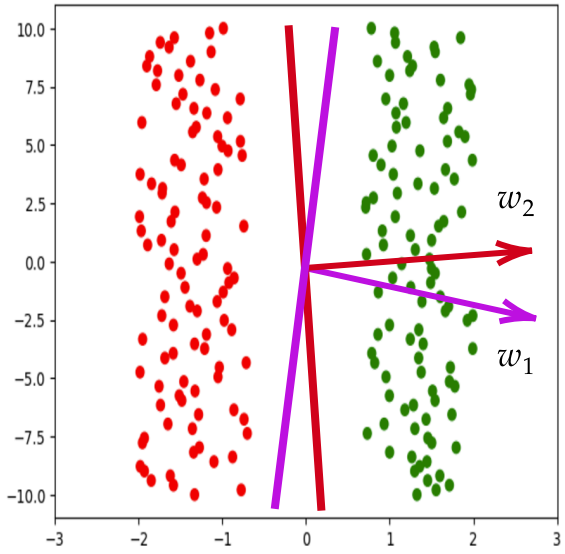
$Q7$) Choose the correct options

A. Initializing at 0 is compulsory in order to implement perceptron
B. Irrespective of the choice of initialization, perceptron will converge to a unique w
C. Logistic regression assumes linear decision boundary
D. There is no possibility to make any adjustments and apply perceptron algorithm on a dataset which is linearly separable with gamma margin but by a hyperplane which does not pass through the origin.

ANSWER : C

A. Initializing at zero is not compulsory. We can randomly choose any vector. Think about what would happen if our random choice turns out to be the choice for which all the data points are correctly predicted.

B. Not necessary. Consider two different initializations for perceptron. One is given by w_1 and the other is given by w_2. Both w_1 and w_2 predict all the points correctly. Therefore, the w we get after convergence for 1^{st} initialization is w_1 and after convergence for 2^{nd} initialization is w_2



C. True

D. We can use $sign\left(w^T x + b\right)$, b will take care of shifting the hyperplanes.

Question:

Consider a linearly separable binary classification dataset with 500 data points and 50 features. Assume that there exists a $\mathbf{w}$ such that $\|\mathbf{w}\| = 1$ and $y_i(\mathbf{w}^\top \mathbf{x}_i) \geq 0.25 \; \forall i$. Also, assume that $\|\mathbf{x}\|_2 \leq 1 \; \forall i$.

What is the maximum number of mistakes that the Perceptron algorithm can make on this dataset?

Answer:

The maximum number of mistakes that the Perceptron algorithm can make is given by:

$$M \leq \frac{R^2}{\gamma^2}$$

where:

- R is the maximum norm of any data point, i.e., $R = \max_i \|\mathbf{x}_i\|_2$
- γ is the margin, i.e., the minimum value of $y_i(\mathbf{w}^\top \mathbf{x}_i)$

From the question:

$$R \leq 1,$$
$$\gamma \geq 0.25.$$

Substituting these values:

$$M \leq \frac{R^2}{\gamma^2} = \frac{1^2}{(0.25)^2} = \frac{1}{0.0625} = 16.$$

Thus, the maximum number of mistakes the Perceptron algorithm can make is:16

Question

Consider a data set:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \mathbf{x}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix},$$

with corresponding class labels:

$$y_1 = -1, \quad y_2 = +1, \quad y_3 = +1, \quad y_4 = -1.$$

Assume you are using the Perceptron algorithm to find a weight vector $\mathbf{w}$ (no bias). You decide to cycle through the points in the order:

$$\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_2, \mathbf{x}_1$$

repeatedly until a linear separator is found.
Question: What is the number of mistakes?
Question: What is the squared length of the weight vector $\mathbf{w}$ corresponding to the final linear separator your algorithm outputs?

Detailed Answer

We initialize the weight vector $\mathbf{w}$ as:

$$\mathbf{w} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The Perceptron update rule is:

$$\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{x}_i,$$

where y_i is the label of the misclassified point $\mathbf{x}_i$.

The **prediction** is determined as:

$$\text{prediction} = \text{sign}(\mathbf{w}^\top \mathbf{x}_i),$$

where:

$$\text{sign}(z) = \begin{cases} +1 & \text{if } z > 0, \\ -1 & \text{if } z < 0. \end{cases}$$

If the prediction does not match y_i, the update is applied.

—

First Cycle:

1. Point $\mathbf{x}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, $y_4 = -1$:

$$\mathbf{w}^\top \mathbf{x}_4 = \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = 0, \quad \text{prediction} = \text{sign}(0) = +1.$$

Since prediction $\neq y_4$, update:

$$\mathbf{w} \leftarrow \mathbf{w} + y_4\mathbf{x}_4 = \begin{bmatrix}0\\0\end{bmatrix} + (-1)\begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}1\\-1\end{bmatrix}.$$

2. Point $\mathbf{x}_3 = \begin{bmatrix}-1\\-1\end{bmatrix}$, $y_3 = +1$:

$$\mathbf{w}^\top\mathbf{x}_3 = \begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}-1\\-1\end{bmatrix} = -1+1 = 0, \quad \text{prediction} = \text{sign}(0) = +1.$$

Since prediction $= y_3$, no update.

3. Point $\mathbf{x}_2 = \begin{bmatrix}1\\-1\end{bmatrix}$, $y_2 = +1$:

$$\mathbf{w}^\top\mathbf{x}_2 = \begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix} = 1+1 = 2, \quad \text{prediction} = \text{sign}(2) = +1.$$

Since prediction $= y_2$, no update.

4. Point $\mathbf{x}_1 = \begin{bmatrix}1\\1\end{bmatrix}$, $y_1 = -1$:

$$\mathbf{w}^\top\mathbf{x}_1 = \begin{bmatrix}1 & -1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = 1-1 = 0, \quad \text{prediction} = \text{sign}(0) = +1.$$

Since prediction $\neq y_1$, update:

$$\mathbf{w} \leftarrow \mathbf{w} + y_1\mathbf{x}_1 = \begin{bmatrix}1\\-1\end{bmatrix} + (-1)\begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}0\\-2\end{bmatrix}.$$

—

1. Point $\mathbf{x}_4 = \begin{bmatrix}-1\\1\end{bmatrix}$, $y_4 = -1$:

$$\mathbf{w}^\top\mathbf{x}_4 = \begin{bmatrix}0 & -2\end{bmatrix}\begin{bmatrix}-1\\1\end{bmatrix} = 0-2 = -2, \quad \text{prediction} = \text{sign}(-2) = -1.$$

Since prediction $= y_4$, no update.

2. Point $\mathbf{x}_3 = \begin{bmatrix}-1\\-1\end{bmatrix}$, $y_3 = +1$:

$$\mathbf{w}^\top\mathbf{x}_3 = \begin{bmatrix}0 & -2\end{bmatrix}\begin{bmatrix}-1\\-1\end{bmatrix} = 0+2 = 2, \quad \text{prediction} = \text{sign}(2) = +1.$$

Since prediction $= y_3$, no update.

3. Point $\mathbf{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $y_2 = +1$:

$$\mathbf{w}^\top \mathbf{x}_2 = \begin{bmatrix} 0 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 0 + 2 = 2, \quad \text{prediction} = \text{sign}(2) = +1.$$

Since prediction $= y_2$, no update.

4. Point $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $y_1 = -1$:

$$\mathbf{w}^\top \mathbf{x}_1 = \begin{bmatrix} 0 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0 - 2 = -2, \quad \text{prediction} = \text{sign}(-2) = -1.$$

Since prediction $= y_1$, no update.

—

Final Weight Vector: After these iterations, the weight vector converges to:

$$\mathbf{w} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}.$$

Squared Length of the Weight Vector:

$$\|\mathbf{w}\|^2 = 0^2 + (-2)^2 = 4.$$

Answer:
a)2 b)4

Question

Consider a logistic regression model that has been trained for a binary classification problem on a dataset in R^2. The weight vector is:

$$\mathbf{w} = \begin{bmatrix} \frac{1}{2} \\ \frac{2}{2} \\ \frac{2}{3} \end{bmatrix}.$$

Given a test data-point as input to the model, it returns 1 as the predicted label if the probability output by the model is greater than 0.60 and 0 otherwise. What is the predicted label for the test data-point:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}?$$

Note that the probability output by a logistic regression model is:

$$P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x}),$$

where $\sigma(z)$ is the sigmoid function defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Answer

The weight vector is:

$$\mathbf{w} = \begin{bmatrix} \frac{1}{2} \\ \frac{2}{3} \end{bmatrix},$$

and the test data-point is:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

The dot product $\mathbf{w}^\top \mathbf{x}$ is calculated as:

$$\mathbf{w}^\top \mathbf{x} = \begin{bmatrix} \frac{1}{2} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{2} \cdot 1 + \frac{2}{3} \cdot 0 = \frac{1}{2}.$$

The sigmoid function is:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Substituting $z = \frac{1}{2}$:

$$\sigma\left(\frac{1}{2}\right) = \frac{1}{1 + e^{-\frac{1}{2}}}.$$

Numerically, $e^{-\frac{1}{2}} \approx 0.60653$, so:

$$\sigma\left(\frac{1}{2}\right) = \frac{1}{1 + 0.60653} \approx \frac{1}{1.60653} \approx 0.622.$$

Since $0.622 > 0.60$, the predicted label is 1.

Final Answer:

$$\boxed{1}$$

Question Consider a logistic regression model trained for a binary classification problem with features in R^2 and labels in $\{1, 0\}$. The probability that the test point

$$\begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

belongs to class 1 is equal to 0.75. What is the probability of the test point

$$\begin{bmatrix} -1 \\ -3 \end{bmatrix}$$

belonging to class 0?

The logistic regression model assigns probabilities to classes based on a linear decision boundary. The test point

$$\begin{bmatrix}1\\3\end{bmatrix}$$

has a probability of 0.75 of belonging to class 1, which implies a probability of:

$$P(\text{class 0 for } \begin{bmatrix}1\\3\end{bmatrix}) = 1 - P(\text{class 1}) = 1 - 0.75 = 0.25.$$

Assuming symmetry in the logistic regression model, the probabilities flip for the test point

$$\begin{bmatrix}-1\\-3\end{bmatrix},$$

so it has a probability of 0.75 of belonging to class 0. Thus, the answer is:

$$\boxed{0.75}.$$