For these exercises, it will be helpful to review the notes on Linear Classifiers and the Perceptron. You may also find it helpful to write some test code with a local python installation or in a google colab notebook.

1) Classification

Consider a linear classifier through the origin in 4 dimensions, specified by

$$\theta = (1, -1, 2, -3)$$

Which of the following points x are classified as positive, i.e. $h(x; \theta) = +1$?

1. $(1, -1, 2, -3)$
2. $(1, 2, 3, 4)$
3. $(-1, -1, -1, -1)$
4. $(1, 1, 1, 1)$

Enter a Python list with a subset of the numbers 1, 2, 3, 4: [1, 3]

100.00%

You have infinitely many submissions remaining.

Solution: `[1, 3]`

Explanation:

In a classification problem, a point is considered positive if $sign(\theta \cdot x)$ is positive and otherwise negative. Note that $\theta \cdot x$ is equal to

$$\sum_i x_i \theta_i$$

Therefore, we have that

$$\theta \cdot x_1 = (1, -1, 2, -3) \cdot (1, -1, 2, -3) = 15$$

So the first point x_1 is classified positively. Similar computations show that x_3 classified positively.

2) Classifier vs Hyperplane

Consider another parameter vector

$$\theta' = (-1, 1, -2, 3)$$

Ex2a

Does θ' represent the same hyperplane as θ does? yes

100.00%

You have infinitely many submissions remaining.

Solution: yes

Explanation:

Note that the hyperplane defined by a norm vector θ is the set of points x such that $\theta \cdot x = 0$. Now, θ' determines the same hyperplane as θ. This is because a hyperplane H is defined as the set of points x such that $x \cdot \theta$ is equal to 0 (x is perpendicular to θ) for some arbitrary θ. For our specific θ in this problem, note that

$$\theta \cdot x = 0 = -\theta \cdot x$$

Which shows that the set of points perpendicular to $-\theta$ is equivalent to the points perpendicular to θ.

Ex2b

Does θ' represent the same classifier as θ does? no

100.00%

You have infinitely many submissions remaining.

Solution: no

Explanation:

Note that hyperplanes specified by norm vectors θ' and θ are the same. However, because θ' and θ point in opposite directions, θ' determines a different classifier than θ since the $sign(\theta \cdot x)$ is different.

3) Linearly Separable Training

As discussed in lecture and in the lecture notes, note that $\mathcal{E}_n(\theta, \theta_0)$ refers to the training error of the linear classifier specified by θ, θ_0, and $\mathcal{E}(\theta, \theta_0)$ refers to its test error. What does the fact that the training data are *linearly separable* imply?

Select "yes" or "no" for each of the following statements:

Ex3a

There must exist θ, θ_0 such that $\mathcal{E}(\theta, \theta_0) = 0$ no

100.00%

You have infinitely many submissions remaining.

Solution: no

Explanation:

There doesn't necessarily exist a hyperplane such that $\mathcal{E}(\theta, \theta_0) = 0$ since just because the training data is separable by a specific hyperplane doesn't mean the entire underlying data distribution will be separable by a hyperplane

Ex3b

There must exist θ, θ_0 such that $\mathcal{E}_n(\theta, \theta_0) = 0$ yes

Submit View Answer 100.00%

You have infinitely many submissions remaining.

Ex3c

A separator with 0 training error exists yes

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3d

A separator with 0 testing error exists, for all possible test sets no

Submit View Answer **100.00%**

You have infinitely many submissions remaining.

Ex3e

The perceptron algorithm will find θ, θ_0 such that $\mathcal{E}_n(\theta, \theta_0) = 0$ yes

100.00%

You have infinitely many submissions remaining.

Solution: yes

Explanation:

If the data is linearly separable, then perceptron will find a separator.

4) Separable Through Origin?

Provide two points, (x_0, x_1) and (y_0, y_1) in two dimensions that are linearly separable but not linearly separable through the origin. If you get stuck try drawing a picture and review the notes on offsets.

Enter a Python list with two entries of the form `[[ x0, x1], label]` where label is `1` or `-1`. (So each entry represents a point with 2 dimensions and its label) [[[1, 1], +1], [[2,2], -1]]

Submit View Answer **100.00%**

You have infinitely many submissions remaining.