ETC3555 2018 - Lab 3

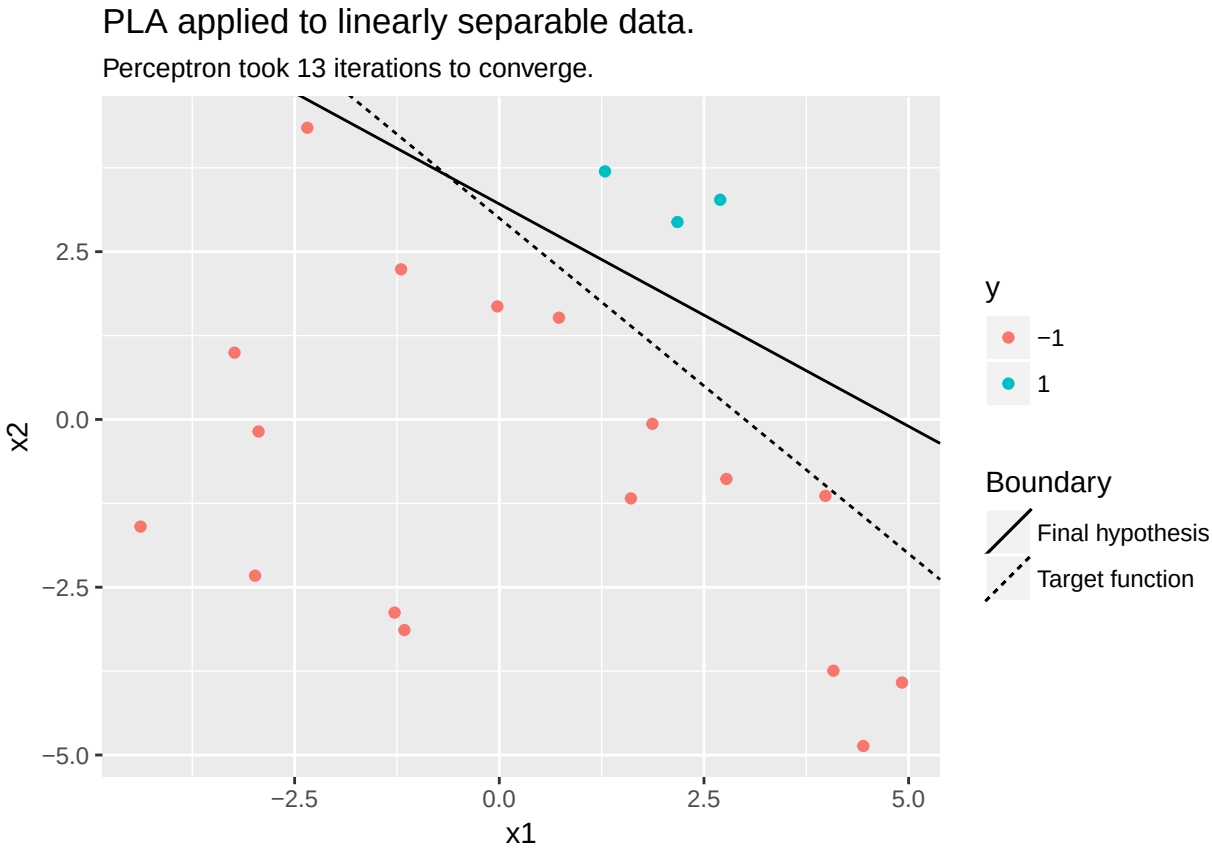
The learning problem and the perceptron algorithm

Cameron Roach and Souhaib Ben Taieb

04 September, 2018

Exercise 1

Solve exercise 1.4 in Learning From Data.



Assignment - Question 1

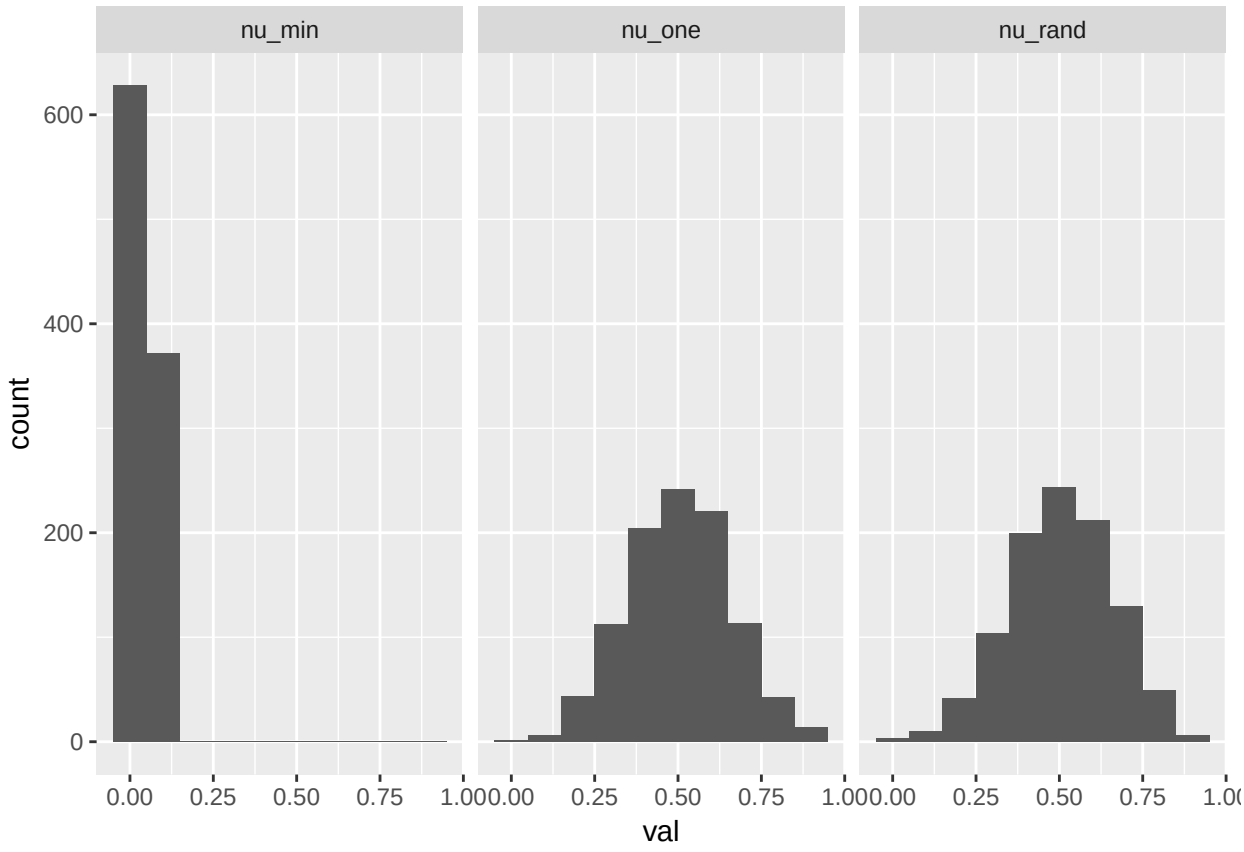
Solve exercise 1.10 in Learning From Data.

1. (a)

(1 mark) For each fair coin, we have $\mu = 0.5$.

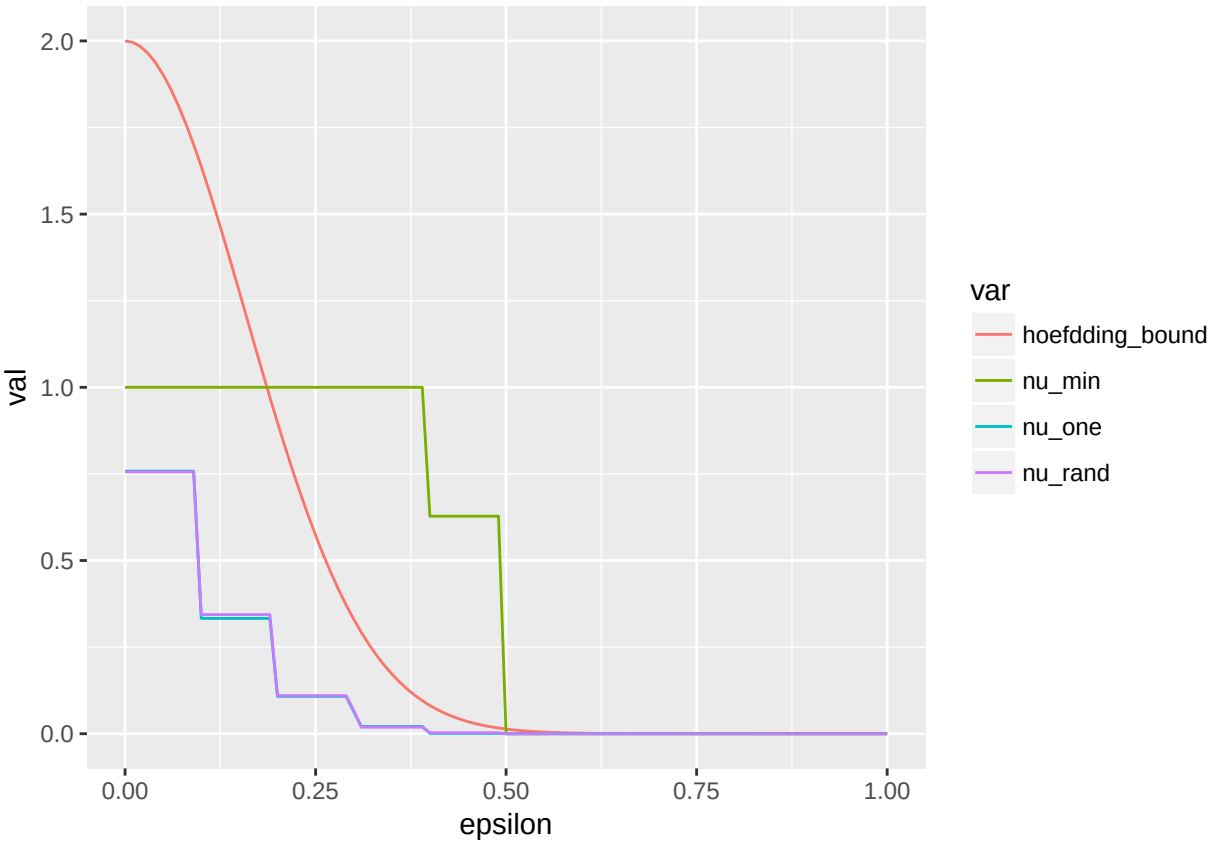
1. (b)

(1 mark)



1. (c)

(1 mark)



1. (d)

(1 mark) We see that c_{min} violates the Hoeffding bound because the coin we analyse is not fixed and is chosen after generating the data set. In other words, we allow our hypothesis to change based on the generated data which violates an assumption of the Hoeffding inequality. The Hoefdding inequality holds for c_{one} and c_{rand} as the coins are chosen before data is generated, hence the hypothesis remains fixed.

1. (e)

(.5 marks) Consider each coin as a bin and each coin flip as a marble drawn from a bin.

(.5 marks) Selecting our hypothesis from multiple bins is analogous to selecting our hypothesis from multiple coins, as when choosing c_{min}. As the Hoeffding bound does not hold for multiple bins, so too does it fail for c_{min}.

Assignment - Question 2

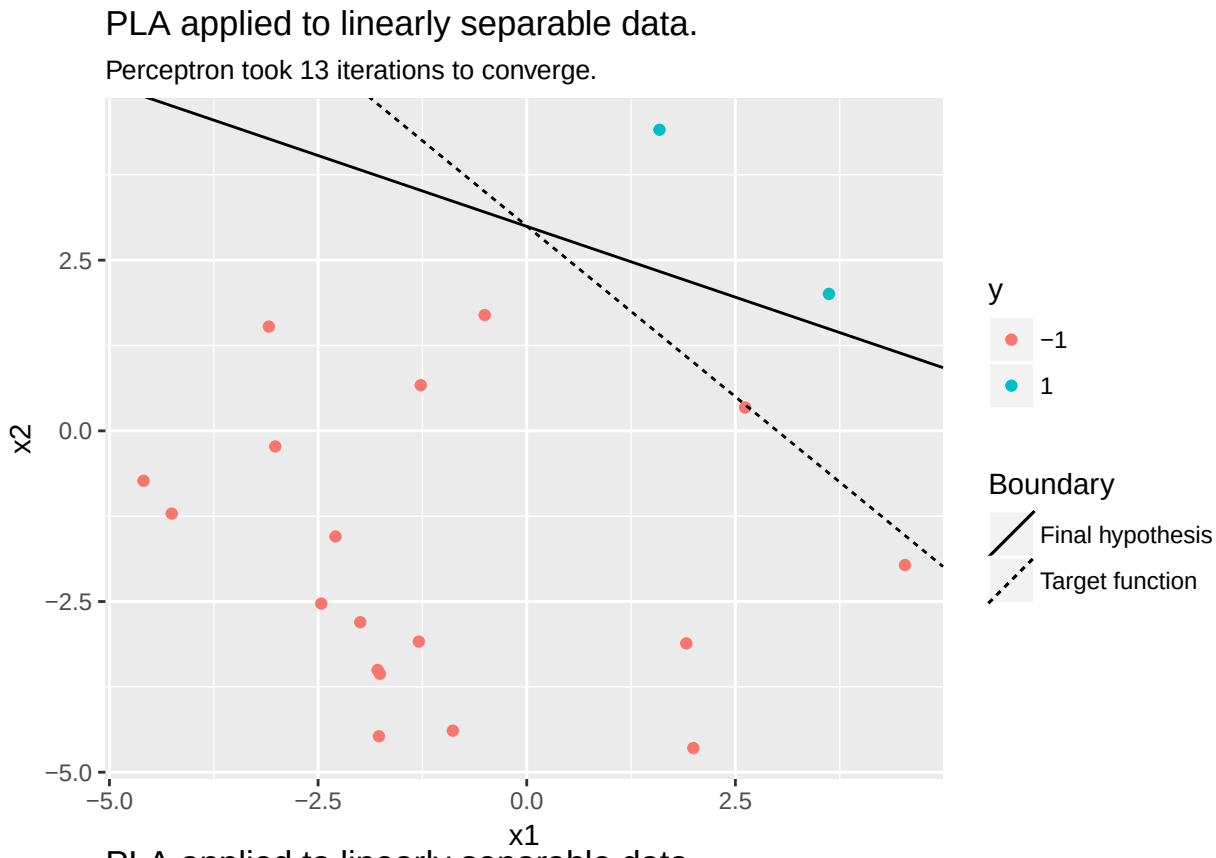
Solve problem 1.4 in Learning From Data.

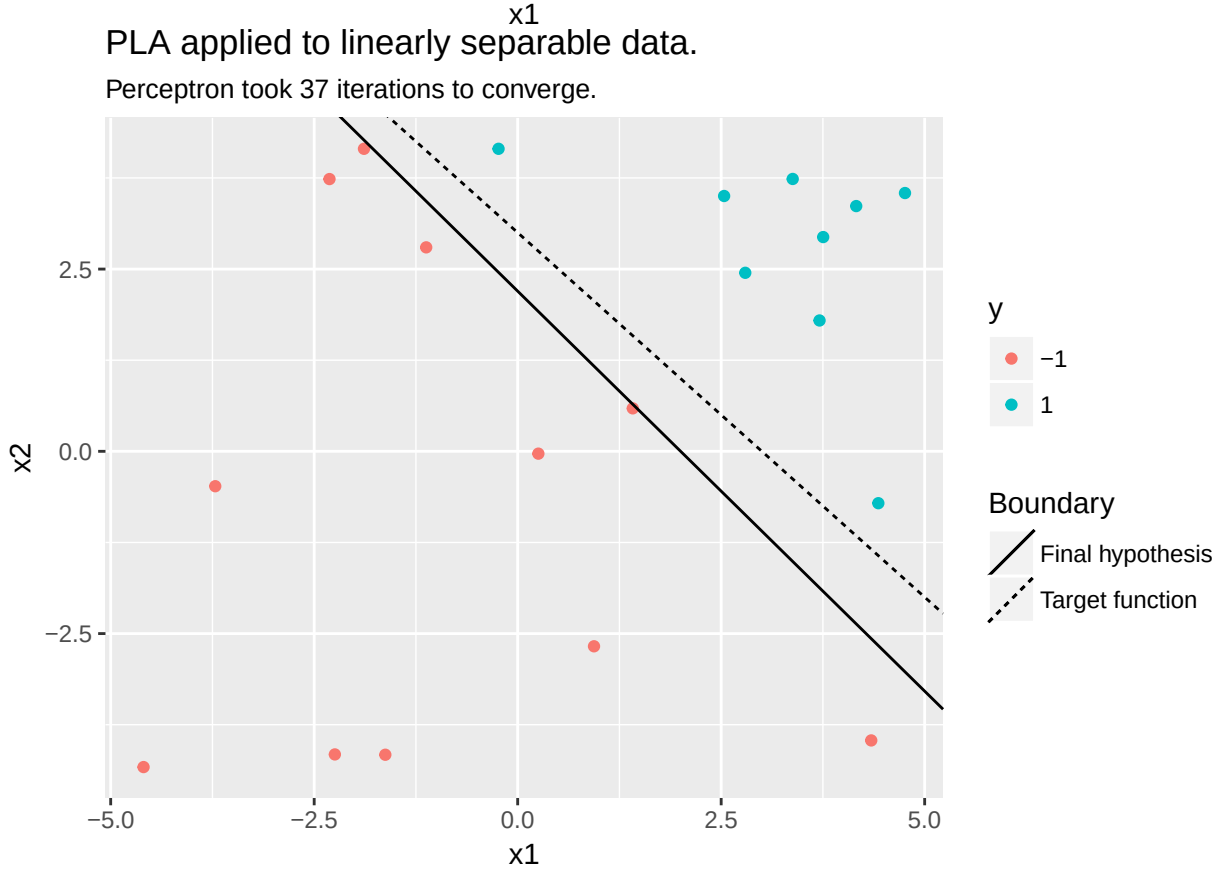
2. (a)

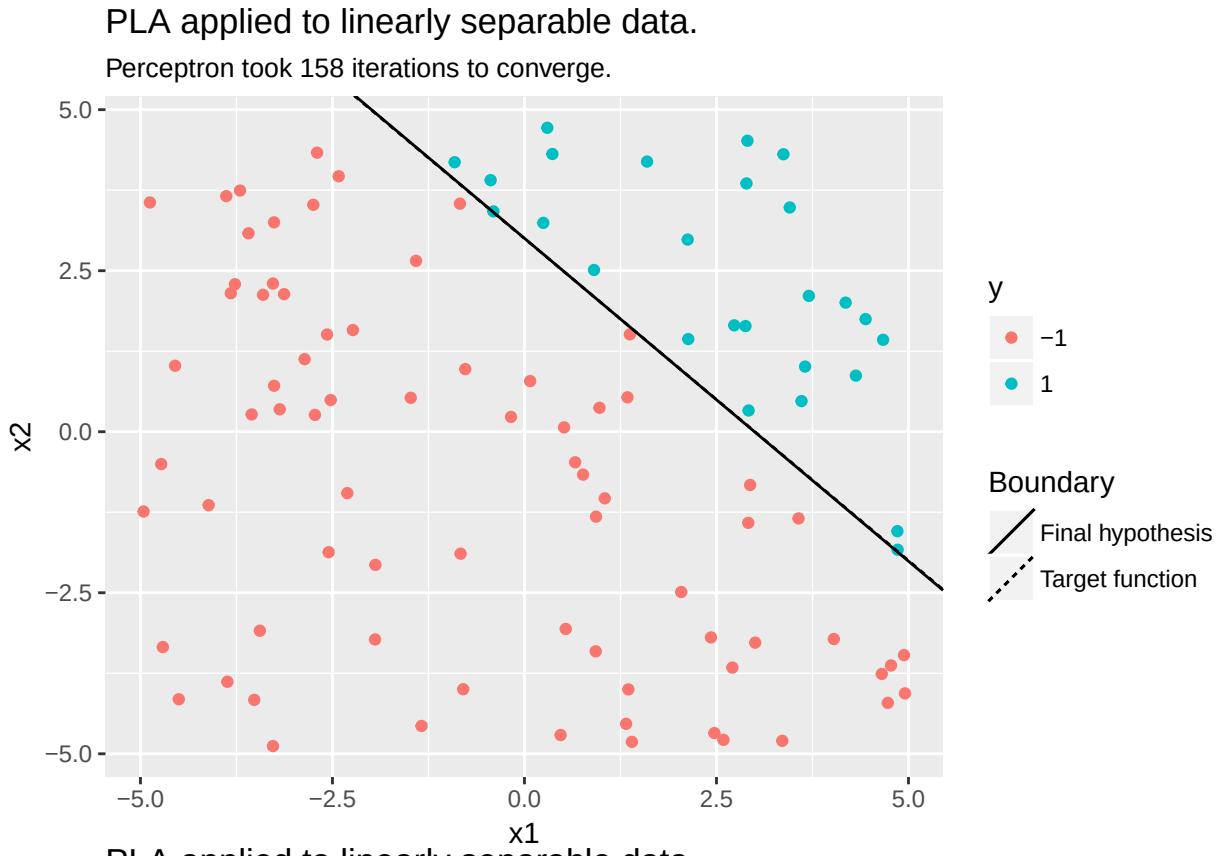
(1 mark) See Exercise 1.

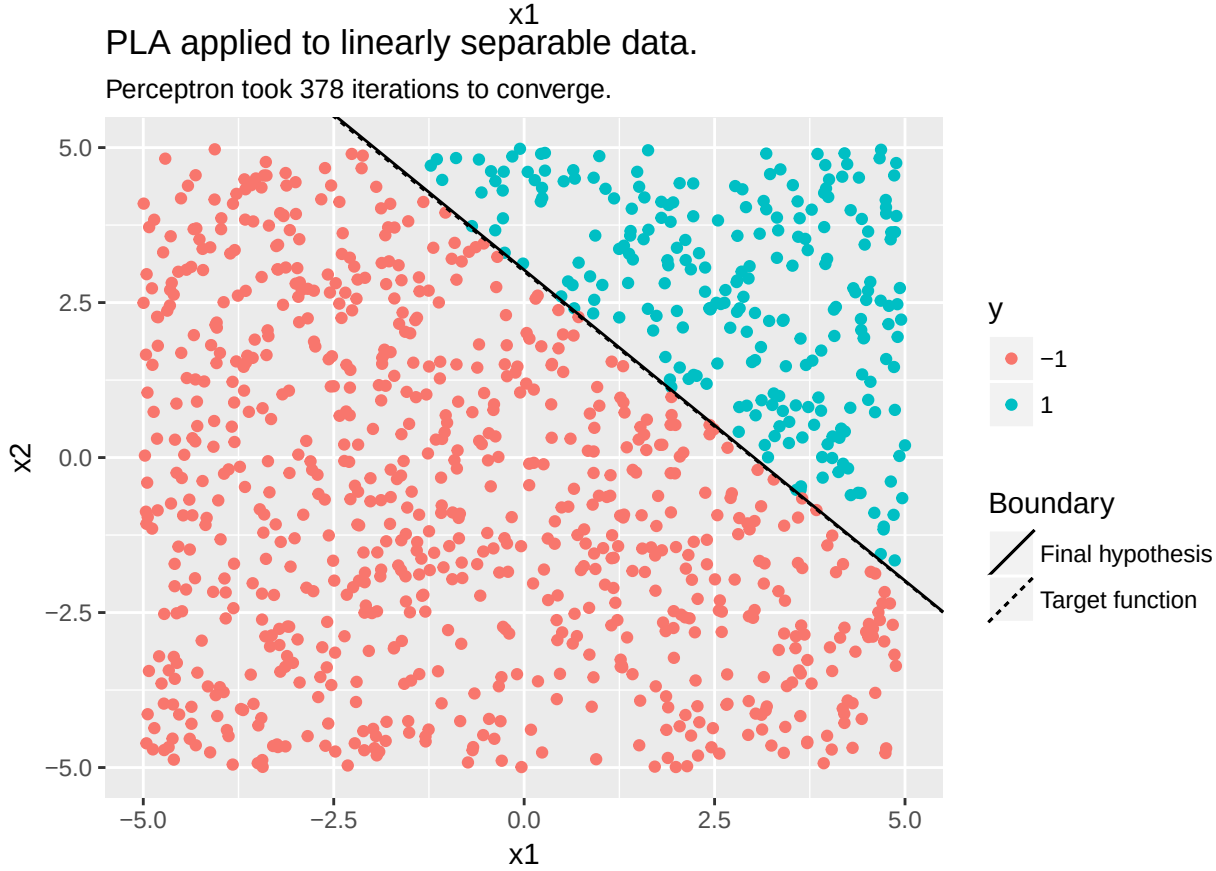
2. (b), (c), (d), (e), (f) and (g)

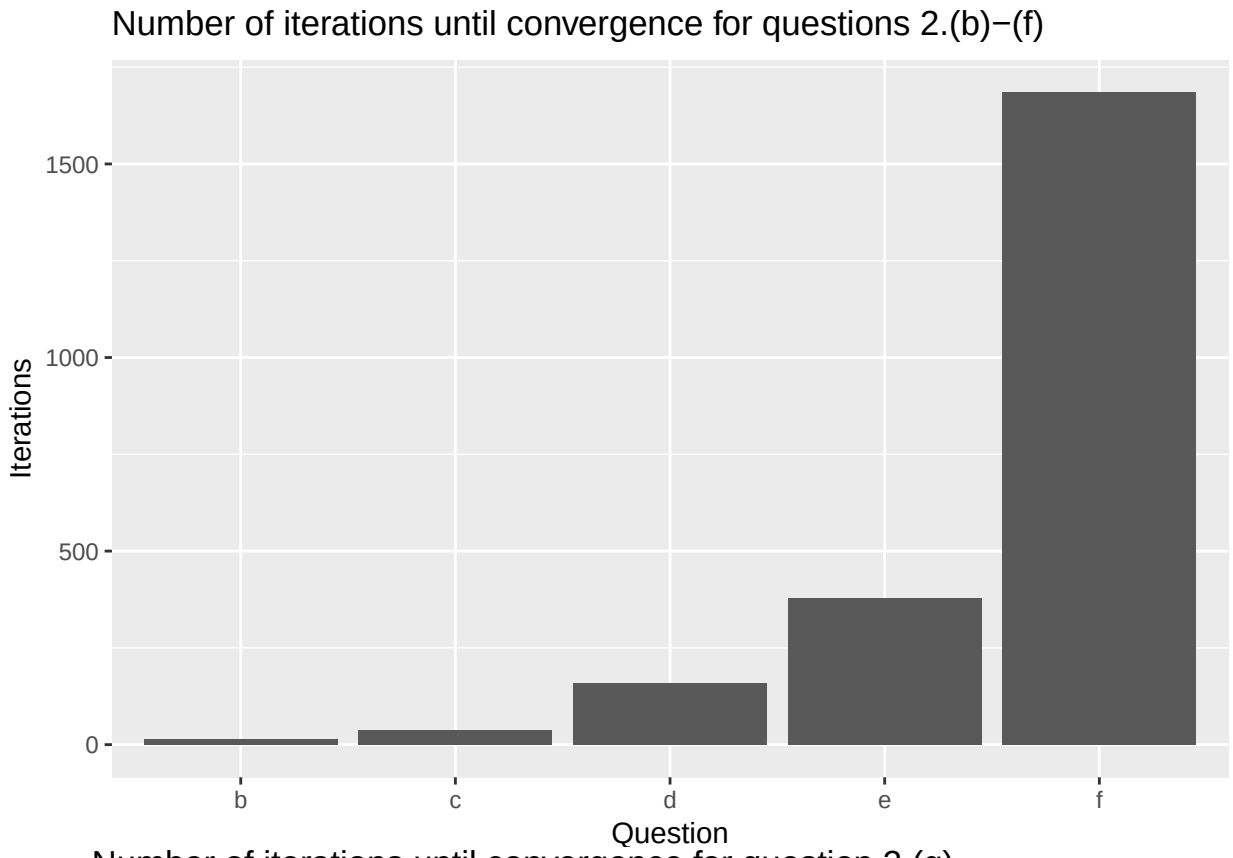
(1 mark each, total of 6)

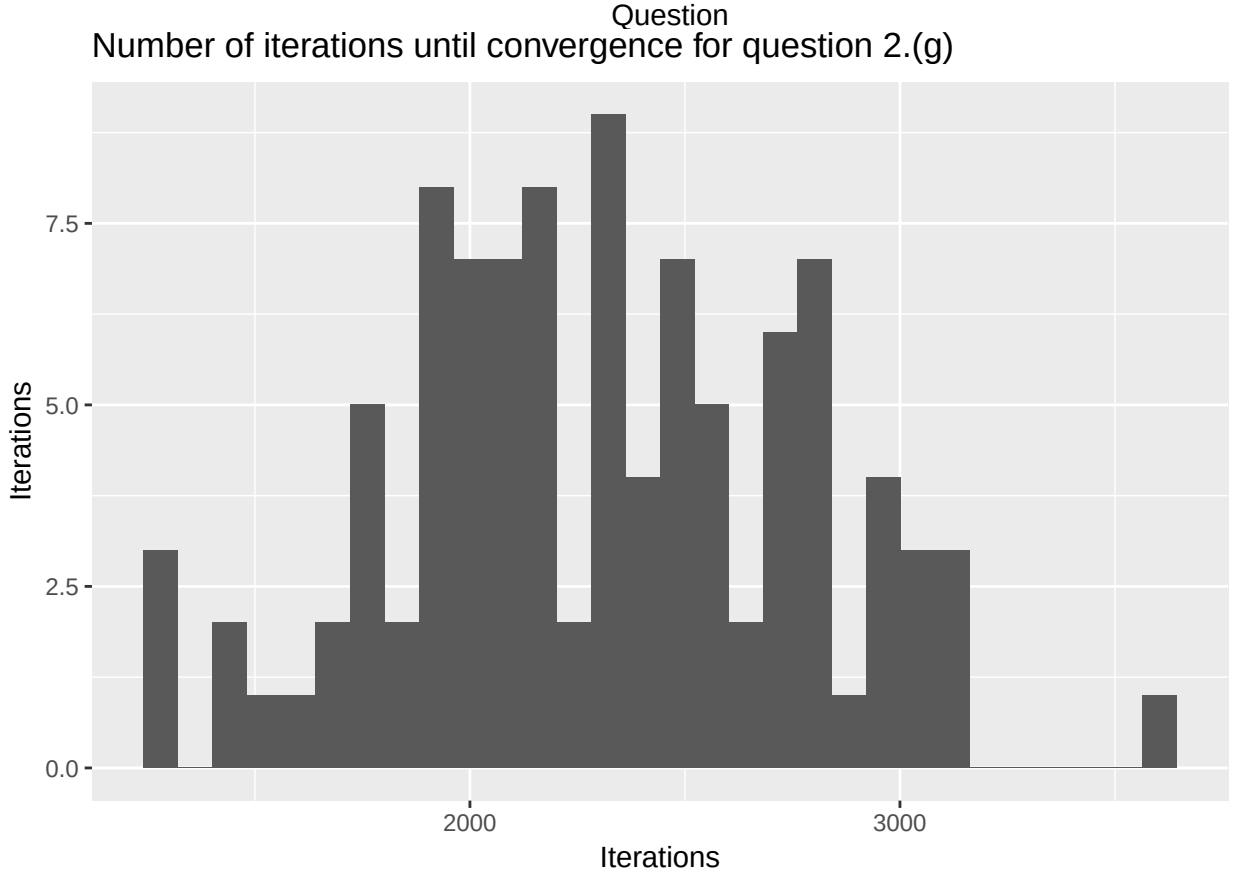

PLA applied to linearly separable data.
Perceptron took 13 iterations to converge.
x2
x1
2.5
0.0
−2.5
−5.0
−5.0
−2.5
0.0
2.5
y
−1
1
Boundary
Final hypothesis
Target function


PLA applied to linearly separable data.
Perceptron took 37 iterations to converge.
x2
x1
2.5
0.0
−2.5
−5.0
−2.5
0.0
2.5
5.0
y
−1
1
Boundary
Final hypothesis
Target function


PLA applied to linearly separable data.
Perceptron took 158 iterations to converge.
5.0
2.5
0.0
−2.5
−5.0
x2
−5.0
−2.5
0.0
2.5
5.0
x1
y
−1
1
Boundary
Final hypothesis
Target function


PLA applied to linearly separable data.
Perceptron took 378 iterations to converge.
5.0
2.5
0.0
−2.5
−5.0
x2
−5.0
−2.5
0.0
2.5
5.0
x1
y
−1
1
Boundary
Final hypothesis
Target function


Number of iterations until convergence for questions 2.(b)−(f)
1500
1000
500
0
Iterations
b
c
d
e
f
Question

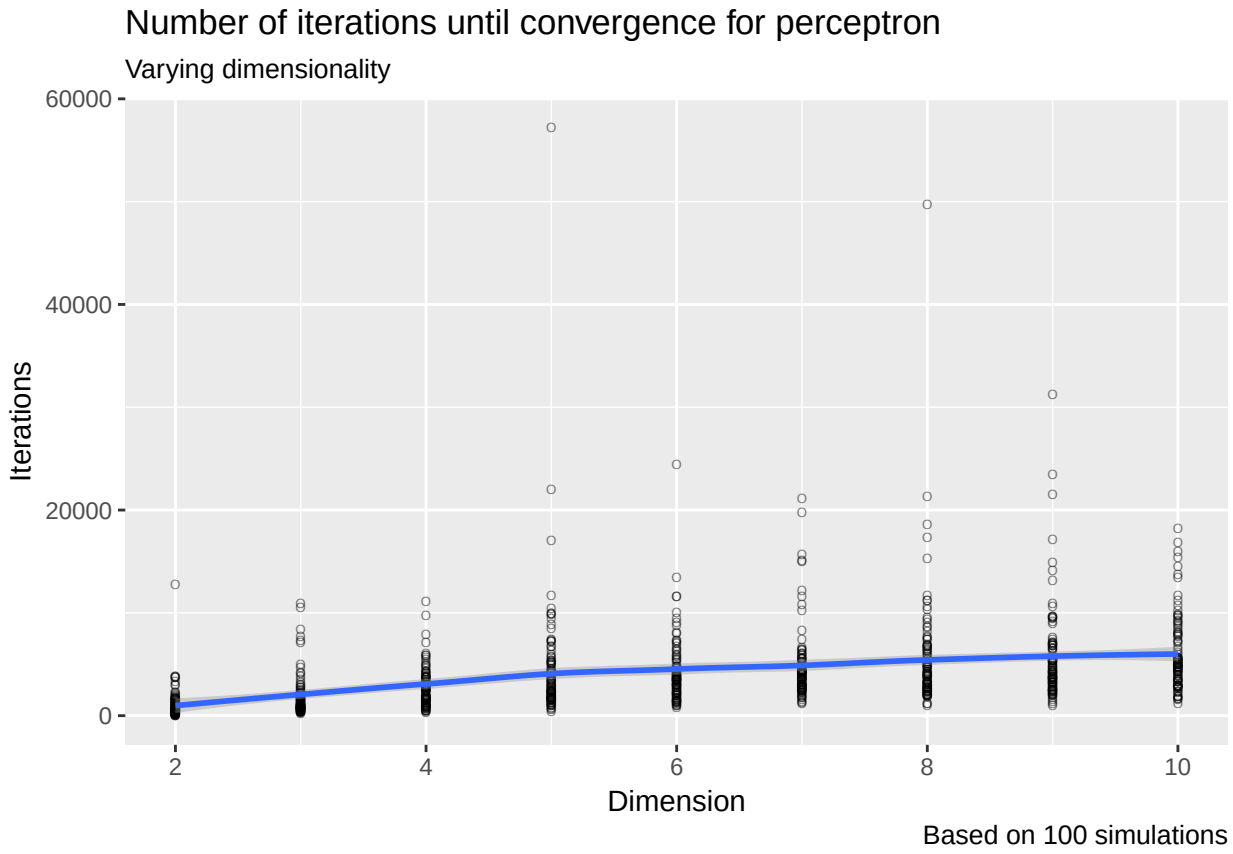

Number of iterations until convergence for question 2.(g)
7.5
5.0
2.5
0.0
Iterations
2000
3000
Iterations

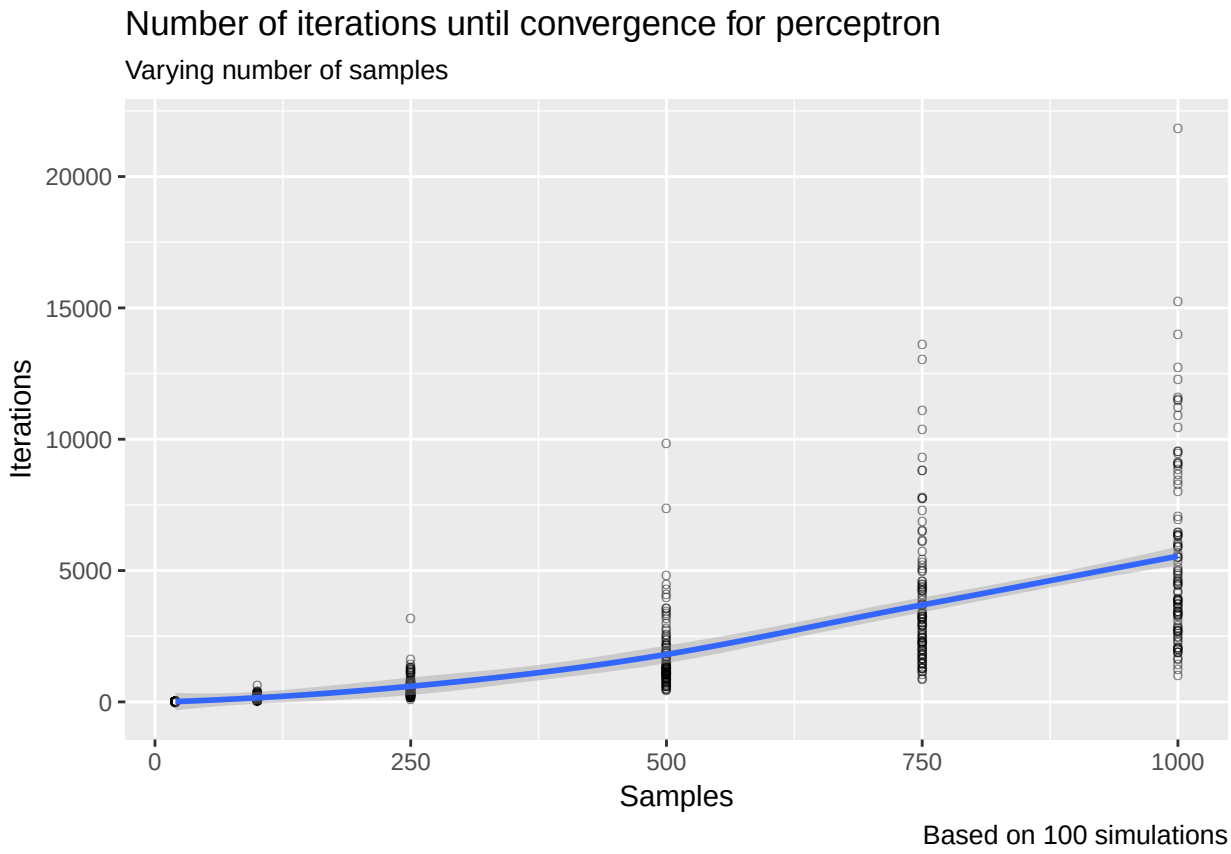
2. (h)

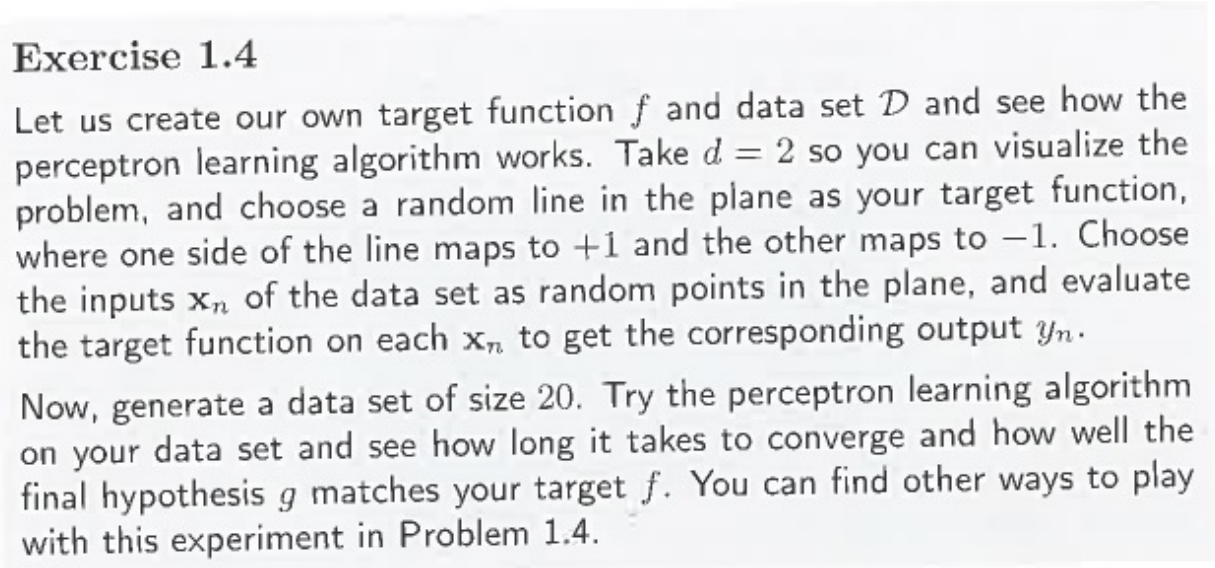
(.5 marks) Either:

1. The classification accuracy doesn't change with respect to N given convergence. PLA always converge to a linear separator with linearly separable data.
2. Final hypothesis approaches target function as N increases.

(.5 marks) Number of iterations required for convergence increases with N and d.




Number of iterations until convergence for perceptron
Varying number of samples
Iterations
20000
15000
10000
5000
0
0
250
500
750
1000
Samples
Based on 100 simulations



Exercise 1.4

Let us create our own target function f and data set $\mathcal{D}$ and see how the perceptron learning algorithm works. Take $d = 2$ so you can visualize the problem, and choose a random line in the plane as your target function, where one side of the line maps to $+1$ and the other maps to -1. Choose the inputs $\mathbf{x}_n$ of the data set as random points in the plane, and evaluate the target function on each $\mathbf{x}_n$ to get the corresponding output y_n.

Now, generate a data set of size 20. Try the perceptron learning algorithm on your data set and see how long it takes to converge and how well the final hypothesis g matches your target f. You can find other ways to play with this experiment in Problem 1.4.

Figure 1: Source: Abu-Mostafa et al. Learning from data. AMLbook.

Exercise 1.10

Here is an experiment that illustrates the difference between a single bin and multiple bins. Run a computer simulation for flipping $1,000$ fair coins. Flip each coin independently 10 times. Let's focus on 3 coins as follows: c_1 is the first coin flipped; c_{rand} is a coin you choose at random; $c_{\min}$ is the coin that had the minimum frequency of heads (pick the earlier one in case of a tie). Let ν_1, ν_{rand} and $\nu_{\min}$ be the fraction of heads you obtain for the respective three coins.

(a) What is μ for the three coins selected?

(b) Repeat this entire experiment a large number of times (e.g., $100,000$ runs of the entire experiment) to get several instances of ν_1, ν_{rand} and $\nu_{\min}$ and plot the histograms of the distributions of ν_1, ν_{rand} and $\nu_{\min}$. Notice that which coins end up being c_{rand} and $c_{\min}$ may differ from one run to another.

(c) Using (b), plot estimates for $\mathbb{P}[|\nu - \mu| > \epsilon]$ as a function of ϵ, together with the Hoeffding bound $2e^{-2\epsilon^2 N}$ (on the same graph).

(d) Which coins obey the Hoeffding bound, and which ones do not? Explain why.

(e) Relate part (d) to the multiple bins in Figure 1.10.

Figure 2: Source: Abu-Mostafa et al. Learning from data. AMLbook.

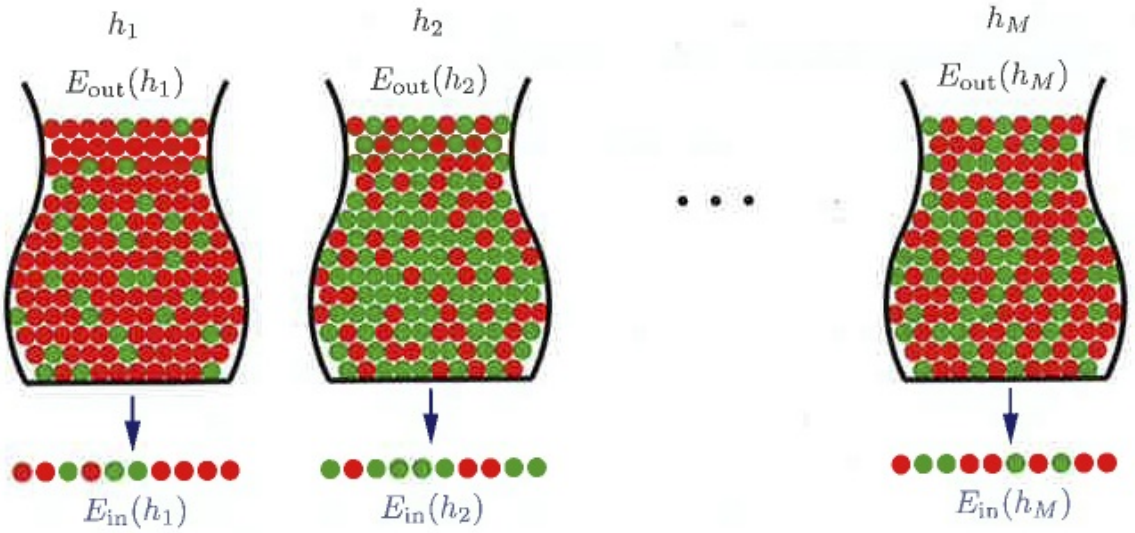


Figure 1.10: Multiple bins depict the learning problem with M hypotheses

Figure 3: Source: Abu-Mostafa et al. Learning from data. AMLbook.

Problem 1.4 In Exercise 1.4, we use an artificial data set to study the perceptron learning algorithm. This problem leads you to explore the algorithm further with data sets of different sizes and dimensions.

(a) Generate a linearly separable data set of size 20 as indicated in Exercise 1.4. Plot the examples $\{(\mathbf{x}_n, y_n)\}$ as well as the target function f on a plane. Be sure to mark the examples from different classes differently, and add labels to the axes of the plot.

(b) Run the perceptron learning algorithm on the data set above. Report the number of updates that the algorithm takes before converging. Plot the examples $\{(\mathbf{x}_n, y_n)\}$, the target function f, and the final hypothesis g in the same figure. Comment on whether f is close to g.

(c) Repeat everything in (b) with another randomly generated data set of size 20. Compare your results with (b).

(d) Repeat everything in (b) with another randomly generated data set of size 100. Compare your results with (b).

(e) Repeat everything in (b) with another randomly generated data set of size $1,000$. Compare your results with (b).

(f) Modify the algorithm such that it takes $\mathbf{x}_n \in \mathbb{R}^{10}$ instead of $\mathbb{R}^2$. Randomly generate a linearly separable data set of size $1,000$ with $\mathbf{x}_n \in \mathbb{R}^{10}$ and feed the data set to the algorithm. How many updates does the algorithm take to converge?

(g) Repeat the algorithm on the same data set as (f) for 100 experiments. In the iterations of each experiment, pick $\mathbf{x}(t)$ randomly instead of deterministically. Plot a histogram for the number of updates that the algorithm takes to converge.

(h) Summarize your conclusions with respect to accuracy and running time as a function of N and d.

Figure 4: Source: Abu-Mostafa et al. Learning from data. AMLbook.