Machine Learning from Data HW10

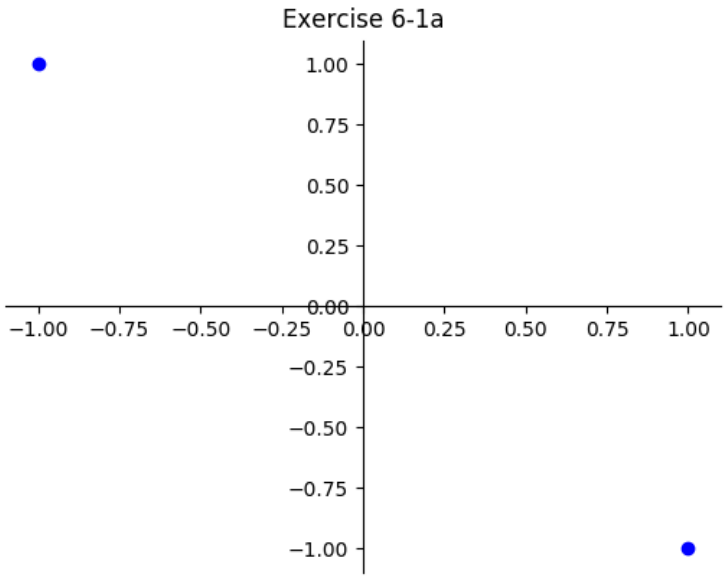
Shane O'Brien

November 2017

Exercise 6.1

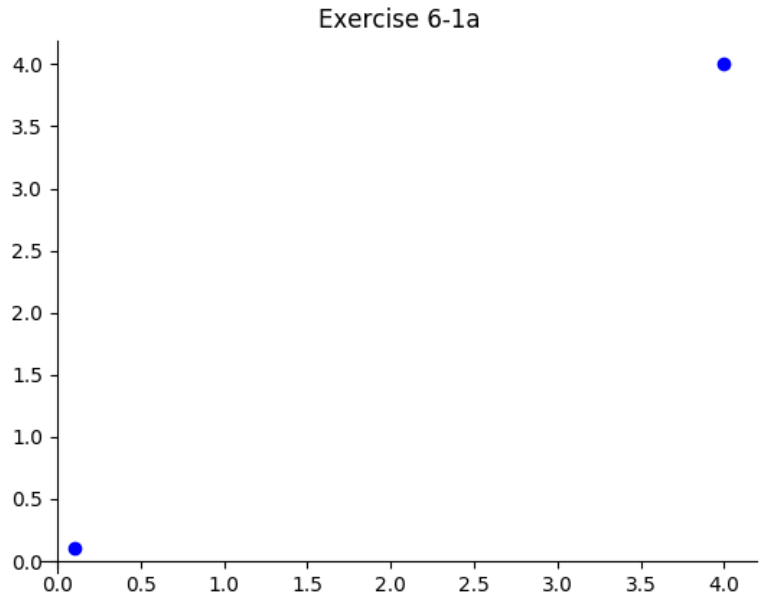
a

A good example of vectors with very high cosine similarity but very low Euclidean distance similarity is as follows:



This is because the two points have the same angle from the x axis, but the Euclidean distance is very far.

A good example of vectors with very low cosine similarity but very high Euclidean distance similarity is as follows:



This is because the cosine similarity is as far as possible. The two points could be brought in to almost touching to shrink the Euclidean distance.

b

If the origin changes, the cosine similarity changes. That is because cosine similarity is based off of the angle from the origin. Euclidean is just the raw distance from the two points, regardless of angles.

Exercise 6.2

a

f(x) picks what seems to be the most optimal, as indicated from $\pi(x)$. If $\pi(x) \geq \frac{1}{2}$, then P[Error] $= 1 - \pi(x)$. If $\pi(x) < \frac{1}{2}$, then P[error] $= \pi(x)$.

So then, we put these together. If we are given an x, the probability we get a wrong result is the minimum of the probabilities from above. That is because, by definition, f(x) picks the maximum chance, determined by $\pi(x)$

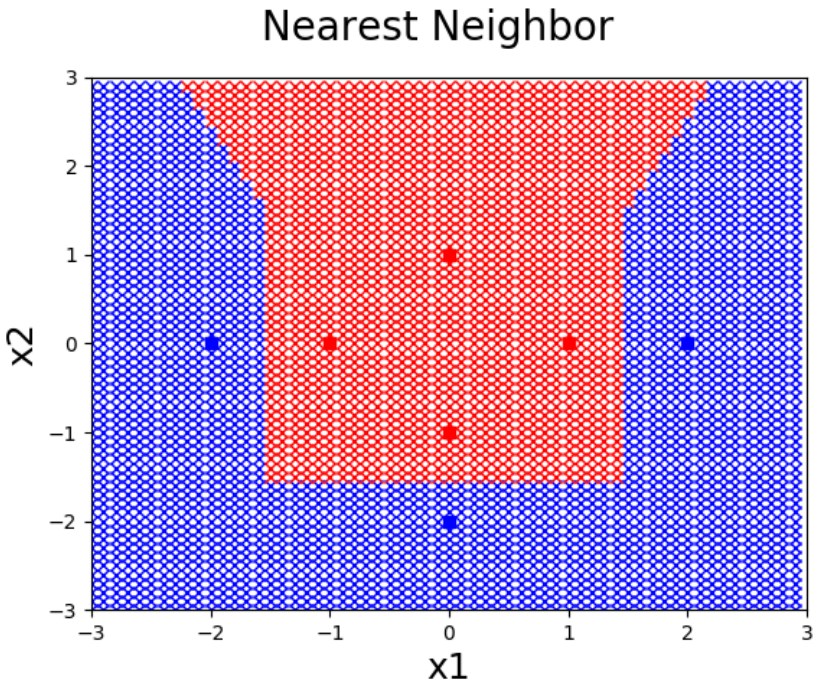
b

By definition, $e(f(x))$ is minimized. In the creation of $f(x)$, we pick the options that minimize error on each point x. If we disagree with $f(x)$ on any point to create $h(x)$, then that error is higher.

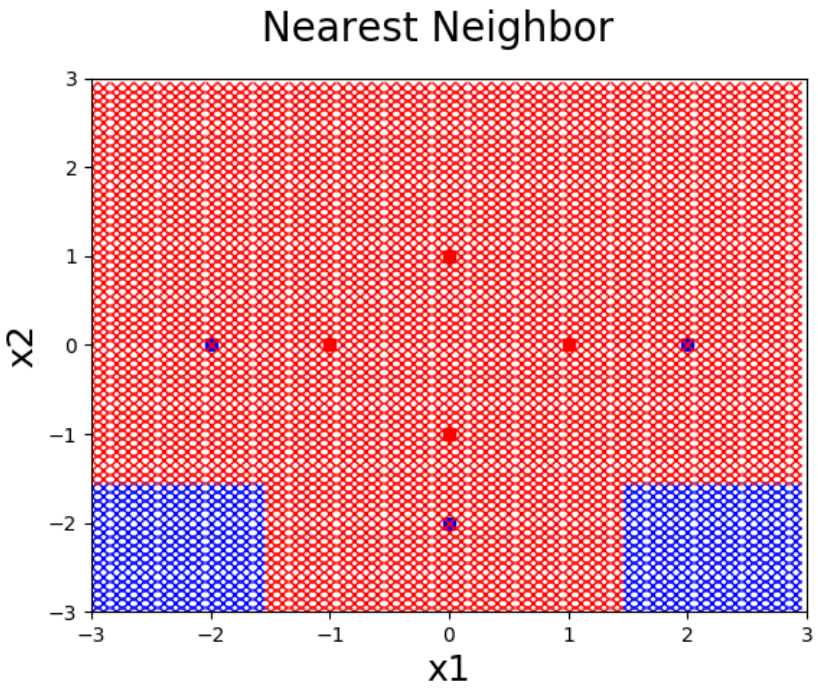
Problem 6.1

a

Below is the decision region for the 1-NN:

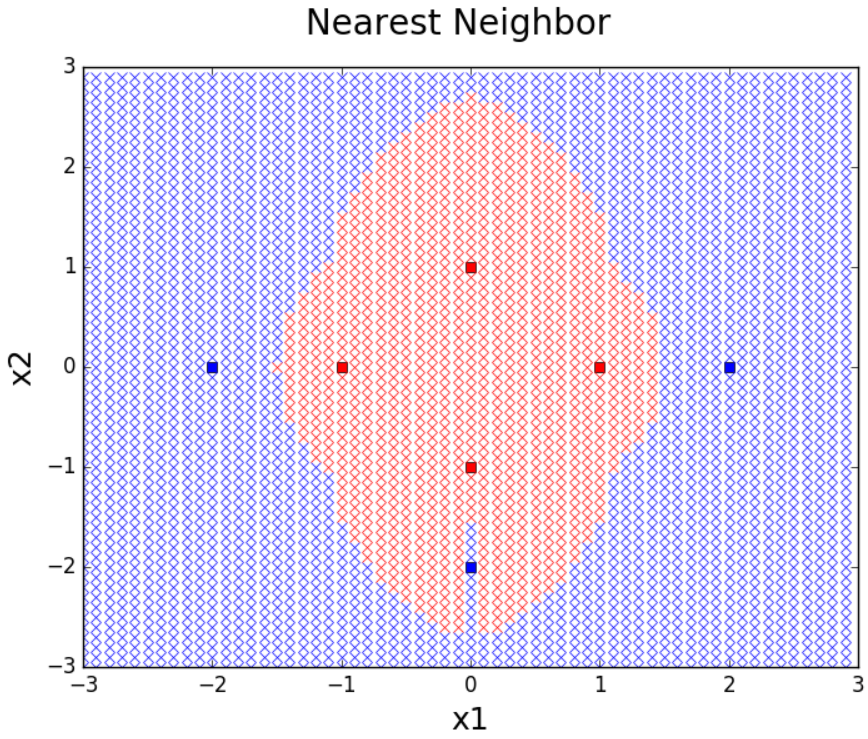


Below is the decision region for the 3-NN:

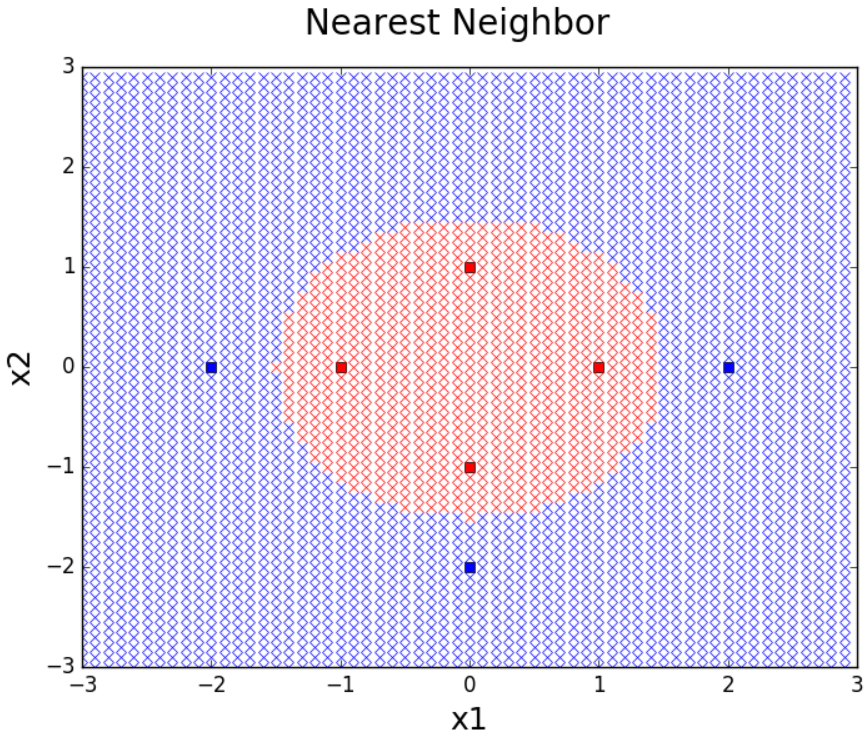


b

Below is the decision region in the x-space for the data in the z-space for the 1-NN:

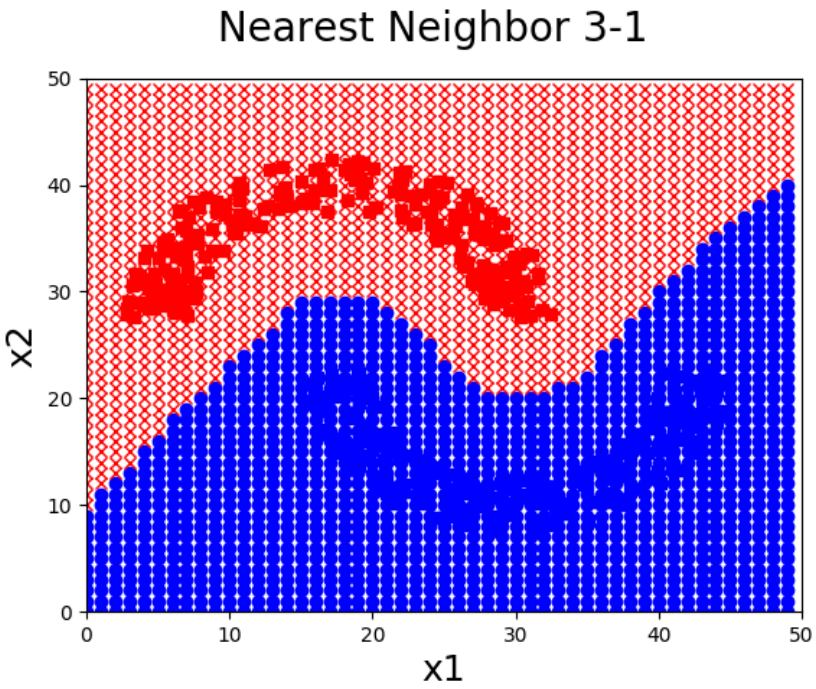


Below is the decision region in the x-space for the data in the z-space for the 3-NN:

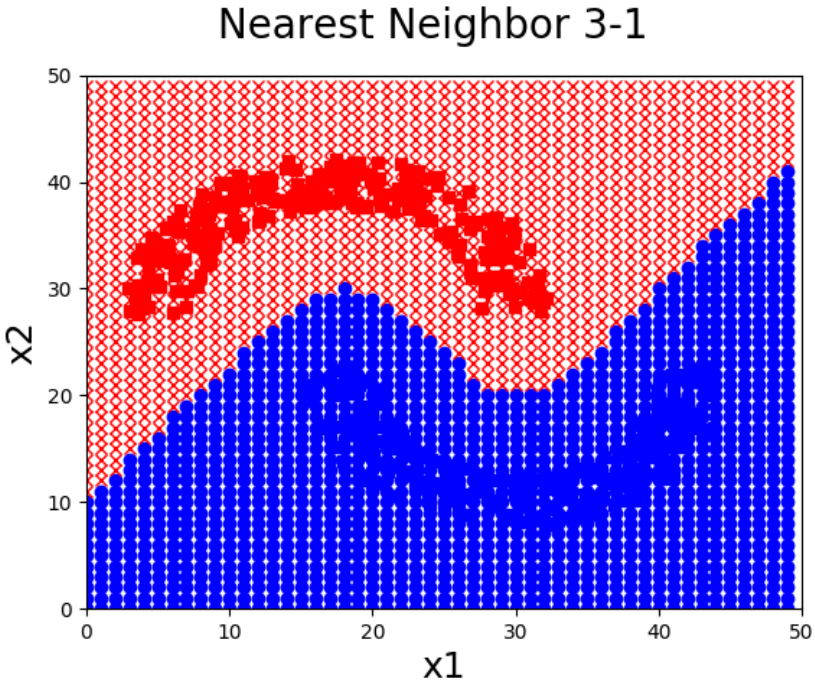


Problem 6.4

The data for this problem was achieved from problem 3.1. The picture below is the decision region for 1-NN:



The picture below is the decision region for 3-NN:



These two results are very similar, which makes sense. The similar data is grouped together, making for very few regions that change. The only difference is that 3NN is a bit more rigid and rocky looking.

Problem 6.16

a

After running the two different algorithms, I achieved these running times with the uniformly generated data:

Branch and Bound: 452.87 Seconds
Brute-Force: 1091.11 Seconds

There is a pretty large improvement for the uniformly generated data.

b

After running the two different algorithms, I achieved these running times with the Gaussian data:

Branch and Bound: 867.34 Seconds
Brute-Force: 1065.14 Seconds

There is a noticeable, but smaller improvement for the Gaussian data

c

This result makes sense because the uniformly generated data can be split into very equal groups. Then, once you try to run Branch and Bound, it is very possible that you satisfy the bound. On the other hand, the Gaussian data isn't separated very well. In the 10 partitions, it looks like I get about 90 percent of my points in just two partitions. So when you run Branch and Bound, you almost always fail the bound and must continue with the brute-force algorithm.

In a bigger sense, these results make sense with the textbook. Even though the Branch and Bound technique can work with varying degrees, it can never make our run time worse. The Gaussian data makes it very hard to utilize Branch and Bound, but it still improves our time by around 20 percent.

d

No, it does not depend on how many test points we will need to evaluate. As I stated above, Branch and Bound cannot make our run time worse, so we should ALWAYS use it. But anyway, If you want to guess how well Branch and Bound will perform on your data, then the number of test points won't tell you much. It'd be more effective to analyze the distribution of the data.