Machine Learning from Data Assignment 10

Greg Stewart

November 19, 2018

Exercise 6.1

(a) *Give two vectors with high cosine similarity but low Euclidean distance similarity. Similarly, do the opposite.*

High cosine similarity; low distance similarity.

$$\mathbf{x}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \mathbf{x}_2 = \begin{pmatrix} 100.01 \\ 101.01 \end{pmatrix}$$

Low cosine similarity; high distance similarity

$$\mathbf{x}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \mathbf{x}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

(b) *If origin of coordinate system changes, which measure of similarity changes? How does this affect your choice of features?*

Euclidean distance is relative, so this measure of similarity **does not** change.

Cosine similarity, however, does change, because the angles of the vectors will change relative to the origin. What this change does depends on where the origin goes.

Exercise 6.2

Let

$$f(\mathbf{x}) = \begin{cases} +1 & \text{if } \pi(\mathbf{x}) \geq \frac{1}{2} \\ -1 & \text{otherwise} \end{cases}$$

Show that the probability of error on a test point $\mathbf{x}$ *is*

$$e(f(\mathbf{x})) = \mathbb{P}[f(\mathbf{x}) \neq y] = \min\{\pi(\mathbf{x}), 1 - \pi(\mathbf{x})\}$$

and $e(f(\mathbf{x}) \leq e(h(\mathbf{x}))$ *for any other hypothesis* h*.*

Take $\pi(\mathbf{x}) \geq 1/2$, giving $f(\mathbf{x}) = +1$. For y we have either +1 or -1, so we need the probability that $y = -1$, in this case, so the results don't match up. Given the definition of $\pi(\mathbf{x})$, for this case we have $\mathbb{P}[f(\mathbf{x}) \neq y] = 1 - \pi(\mathbf{x})$.

Now consider $\pi(\mathbf{x}) < 1/2$, so $f(\mathbf{x}) = -1$. Now we need the probability that $y = +1$, as this value means $f(\mathbf{x}) \neq y$. This is given by $\mathbb{P}[f(\mathbf{x}) \neq y] = \pi(\mathbf{x})$.

Thus, for any test point $\mathbf{x}$ we get the error

$$e(f(\mathbf{x})) = \mathbb{P}[f(\mathbf{x}) \neq y] = \min\{\pi(\mathbf{x}), 1 - \pi(\mathbf{x})\}$$

Now, $f(\mathbf{x})$ is the optimal prediction for the value of y as it is directly tied to $\pi(\mathbf{x})$. If any nondeterministic hypothesis h is chosen, then it can only be as good as $f(\mathbf{x})$, or worse. Deterministic hypotheses can likewise only be as good as $f(\mathbf{x})$ in the best case. Thus

$$e(h(\mathbf{x})) = \mathbb{P}[h(\mathbf{x}) \neq y] \geq \min\{\pi(\mathbf{x}), 1 - \pi(\mathbf{x})\}$$

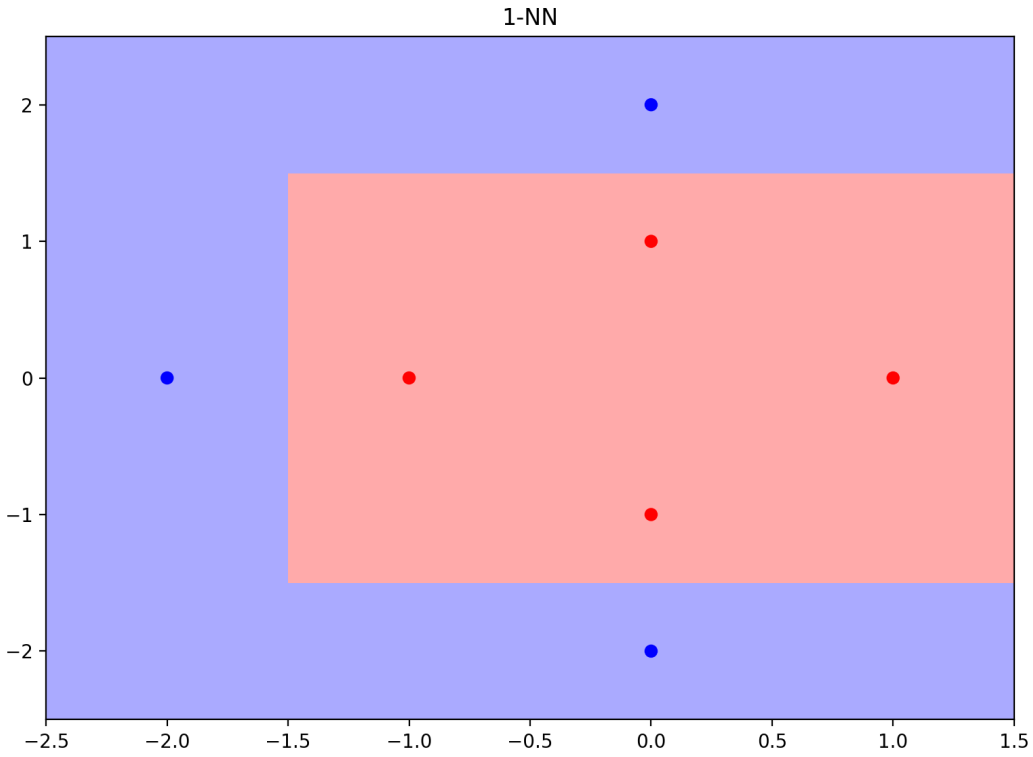
This implies that

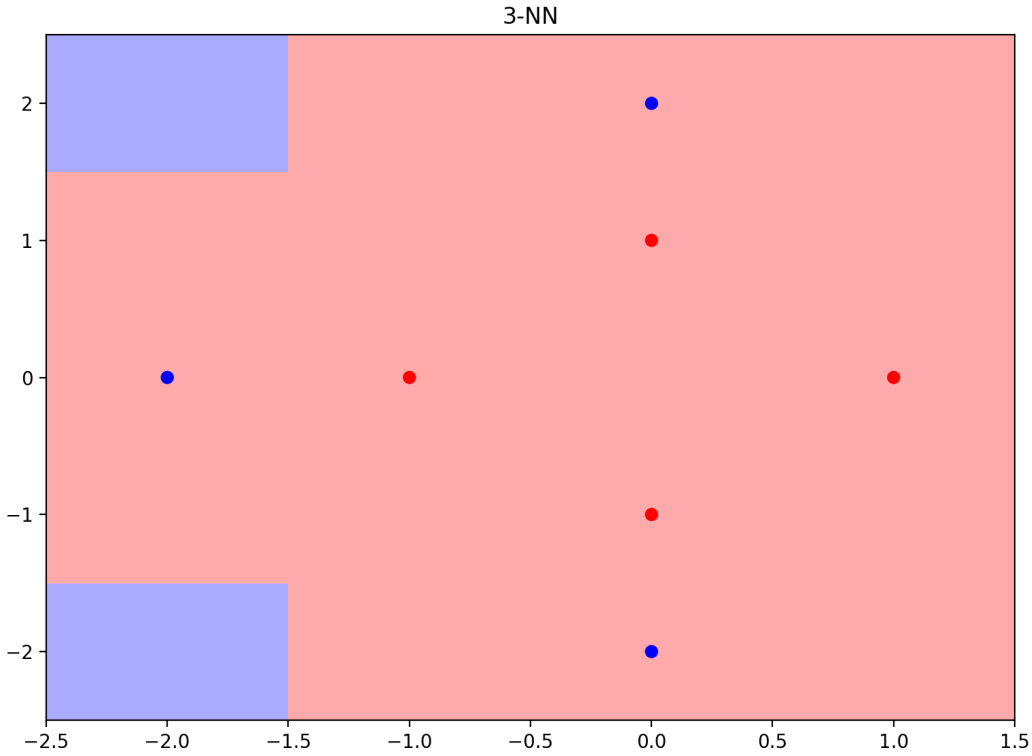
$$e(f(\mathbf{x})) \leq e(h(\mathbf{x}))$$

Problem 6.1

Consider the data set of 7 points given in the text.

(a) *Show the decision regions for the 1-NN and 3-NN rules.*



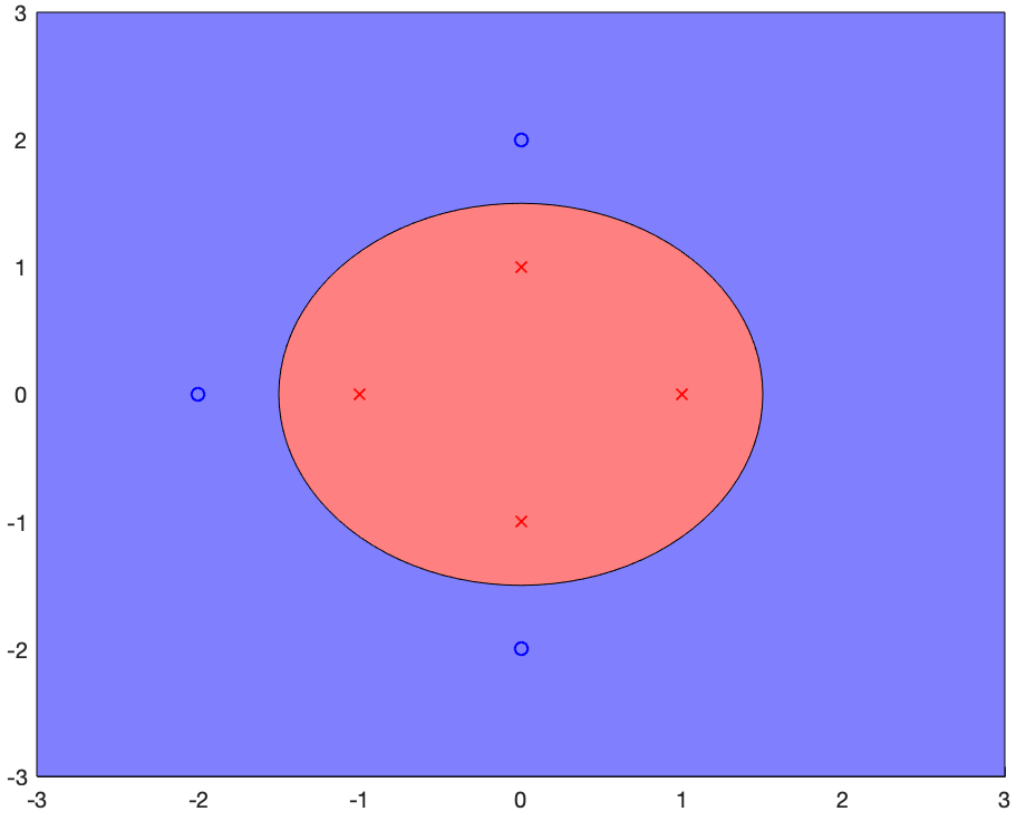


(b) *Consider the nonlinear transform*

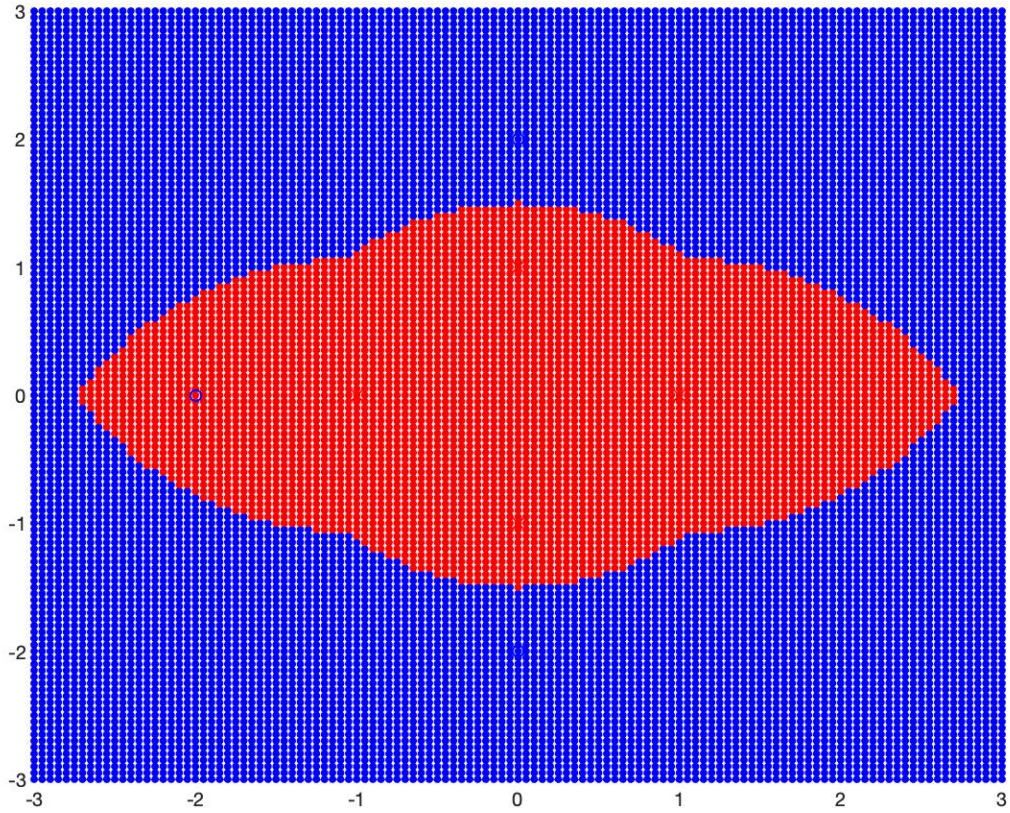
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ \arctan(x_2/x_1) \end{bmatrix},$$

which maps $\mathbf{x}$ *to* $\mathbf{z}$*. Show the classification regions in the* $\mathbf{x}$ *space for the 1-NN and 3-NN rules implemented on teh data in the* $\mathbf{z}$*-space.*

First, we have the 1-NN boundary using the transformation.

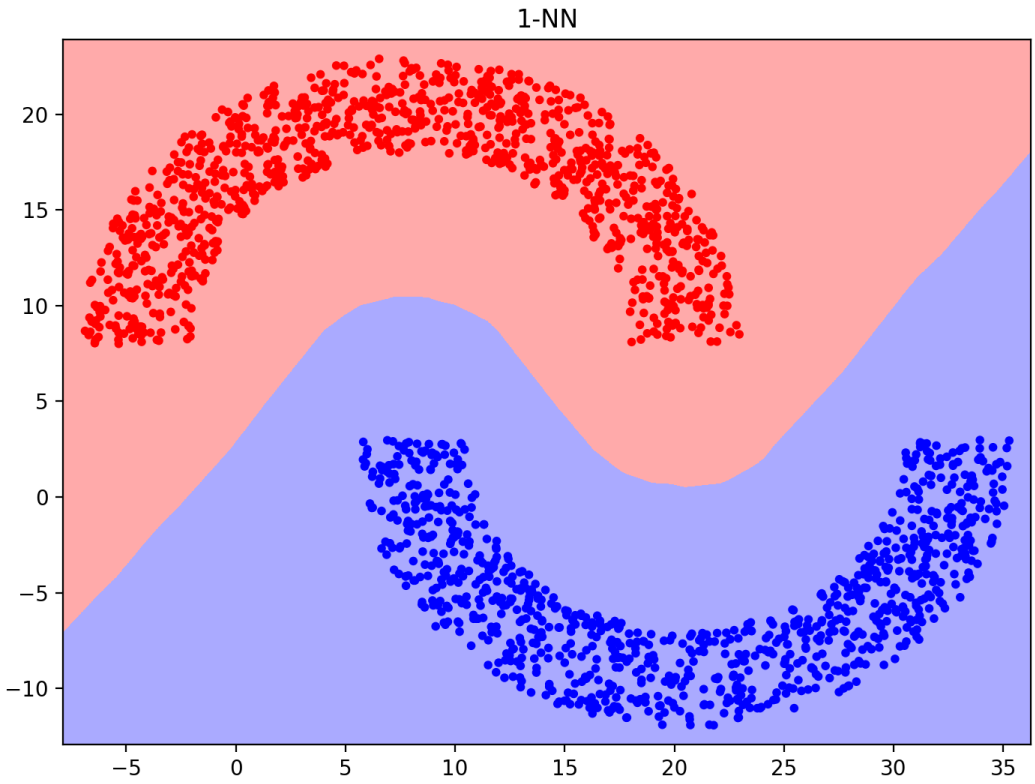


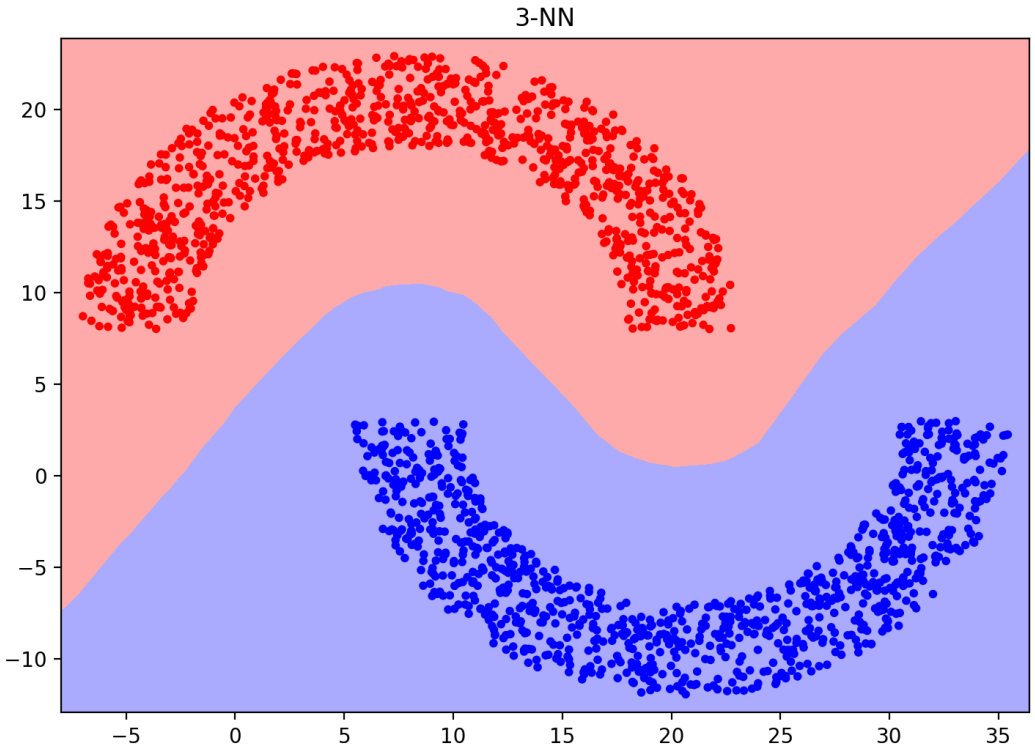
Then, we have the 3-NN boundary under this transformation (excuse the poor plotting; couldn't figure out something better in MATLAB).



Problem 6.4

For the double semi-circle problem in 3.1, plot the decision regions for the 1-NN and 3-NN rules.

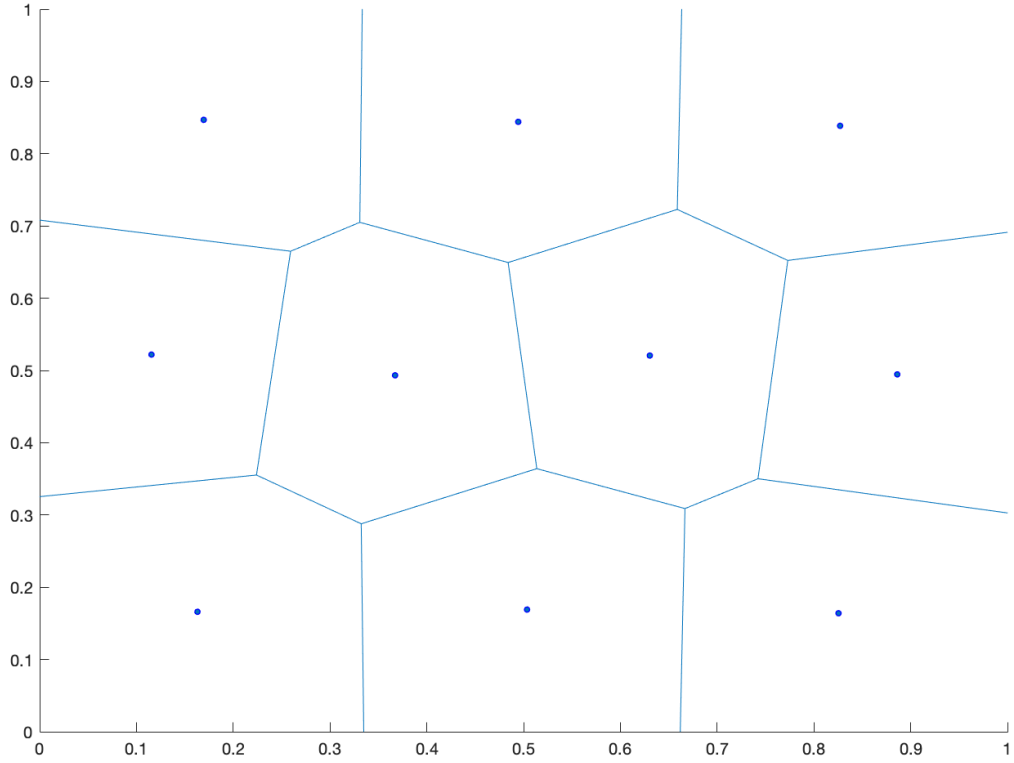

1-NN
20
15
10
5
0
−5
−10
−5
0
5
10
15
20
25
30
35



Problem 6.16

(a) *Generate a data set of 10,000 data points uniformly in the unit square* $[0,1]^2$ *to test the performance of branch and bound method.*

i. *Construct a 10-partition for the data using the simple greedy heuristic described in the text.*

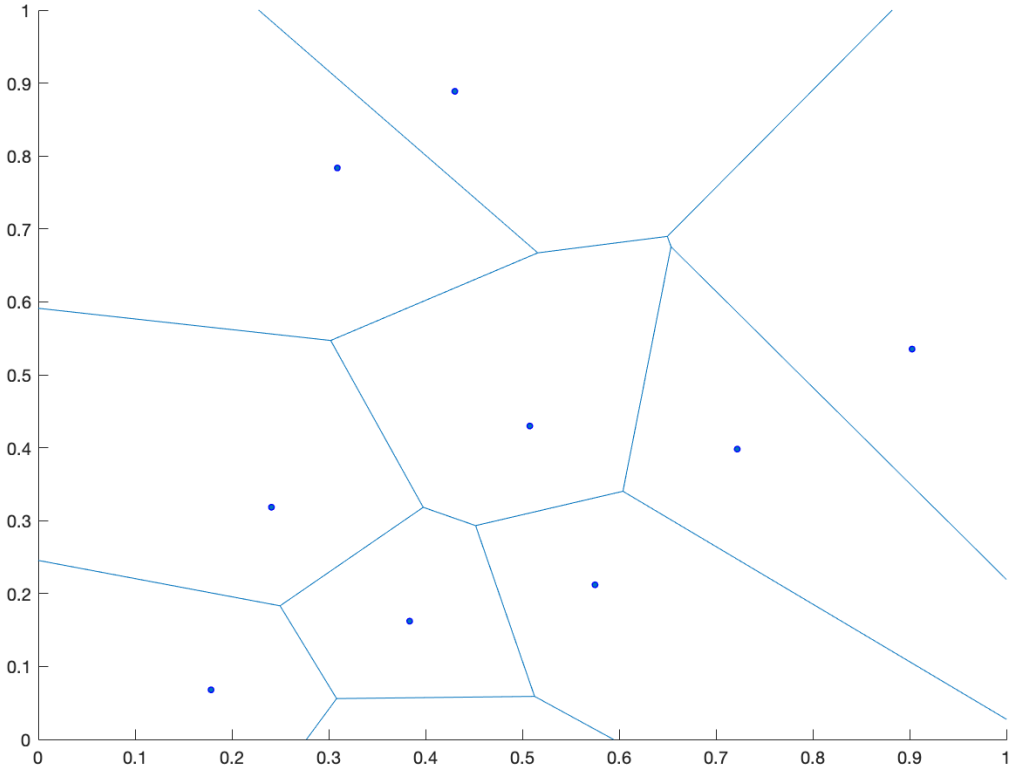


ii. *Generate 10,000 random query points and compare the running time of obtaining the nearest neighbor using partition with branch and bound versus brute force approach which does not use partition.*

The brute force approach (finding the minimum of the distances from each query point to all other points) takes 3.165 seconds.

The branch and bound method takes only .397 seconds to find the nearest neighbor for all query points.

(b) *Repeat (a) instead generating the data from a mix of 10 guassians with centers randomly distributed in* $[,1]^2$ *and identical covariances for each bump equal to* σI *where* $\sigma = 0.1$.



The brute force method takes about the same amount of time, at 3.214 seconds for all query points' nearest neighbors.

The branch and bound method took slightly less time, at .345 seconds to find all nearest neighbors.

(c) *Explain your observations.*

The guassian distributions lead to much less regular cluster calculations, as you can see from the voronoi diagrams. This is because the clusters of points are much more localized, and only the centers are randomly distributed, while in part (a) all the points are uniformly distributed, leading to fairly uniform cluster regions.

The brute force method should take the same amount of time to finish, since it does all the same calculations. It would make sense for the branch and bound method to take less time to complete for the guassian distributions, since the bound is more likely to confirm the guess for the nearest neighbor when the clusters are tight and more distant. In the uniform distribution, it is much less likely for the bound to hold on the first try, so other clusters will have to be analyzed for a nearer neighbor.

(d) *Does your decision to use the branch and bound technique depend on how many test points you need to evaluate?*

Yes. For only a few test points, the brute force method may be the faster approach because it takes a fairly significant amount of computation to calculate all the clusters and perform the bound test. However, as the number of test points increases, the branch and bound method will outpace brute force.