LFD Problem Set 10

John Cohen

November 18, 2024

Exercise 6.1

Deterministic noise depends on H, as some models approximate f better than others.

(a) Give two vectors with very high cosine similarity but very low Euclidean distance similarity. Similarly, give two vectors with very low cosine similarity but very high Euclidean distance similarity.

Cosine Similarity:

$$cos(\theta) = \frac{v_1 * v_2}{||v_1|| * ||v_2||}$$

Euclidean Distance:

$$||v_1 - v_2||$$

Case 1: High Cosine Similarity, Low Euclidean Similarity

$$v_1 = [1000, 1000], v_2 = [1, 1]$$

Cosine:

$$\frac{(1000 * 1) + (1000 * 1)}{\sqrt{1000^2 + 1000^2} * \sqrt{1^2 + 1^2}} = 1$$

Euclidean:

$$\sqrt{(1000 - 1)^2 + (1000 - 1)^2} = 1412.80$$

Case 2: Low Cosine Similarity, High Euclidean Similarity

$$v_1 = [0.001, 1], v_2 = [1, 0.001]$$

Cosine:

$$\frac{(0.001 * 1) + (1 * 0.001)}{\sqrt{0.001^2 + 1^2} * \sqrt{1^2 + 0.001^2}} \approx 0.002$$

Euclidean:

$$\sqrt{(0.001 - 1)^2 + (1 - 0.001)^2} \approx 1.4128$$

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

When the origin of the coordinate system changes (a translation), the Euclidean distance measure changes because it depends on the absolute positions of the vectors. In contrast, cosine similarity remains unchanged since it depends only on the angle between vectors, not their positions relative to the origin. This affects feature choice by necessitating data centering or normalization when using Euclidean distance to ensure meaningful comparisons after shifts in origin. If cosine similarity is used, you can focus on features that capture the directionality of data without concern for origin shifts. Therefore, understanding which similarity measure is affected guides how you preprocess data and select features for your analysis.

Exercise 6.2

Let .... Show that the probability of error on a test point x is

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

and $e(f(x)) \leq e(h(x))$ for any other hypothesis h (deterministic or not).

There are two cases: $\pi(x) \geq \frac{1}{2}$ and $\pi(x) \leq \frac{1}{2}$.
$\pi(x) < \frac{1}{2}$:

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

$\pi(x) \geq \frac{1}{2}$:

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

In both cases $e(f(x)) = P[f(x) \neq y] = \min\{\pi(x), 1-\pi(x)\}$. Now for $e(f(x)) \leq e(h(x))$ for any other hypothesis:

$$P[h(x) = -1] = \rho$$

$$P[h(x) = 1] = 1 - \rho$$

$$\mathrm{e}(h(x)) = P[h(x) = 1]P[y = -1|x] + P[h(x) = -1]P[y = 1|x]$$

$$\mathrm{e}(h(x)) = (1-\rho)(1-\pi(x)) + \rho\pi(x)$$

$$\mathrm{e}(h(x)) \geq (1-\rho)\min\{\pi(x), 1-\pi(x)\} + \rho\min\{\pi(x), 1-\pi(x)\}$$

$$\mathrm{e}(h(x)) \geq \min\{\pi(x), 1-\pi(x)\}$$

$$\mathrm{e}(h(x)) \geq \mathrm{e}(f(x))$$

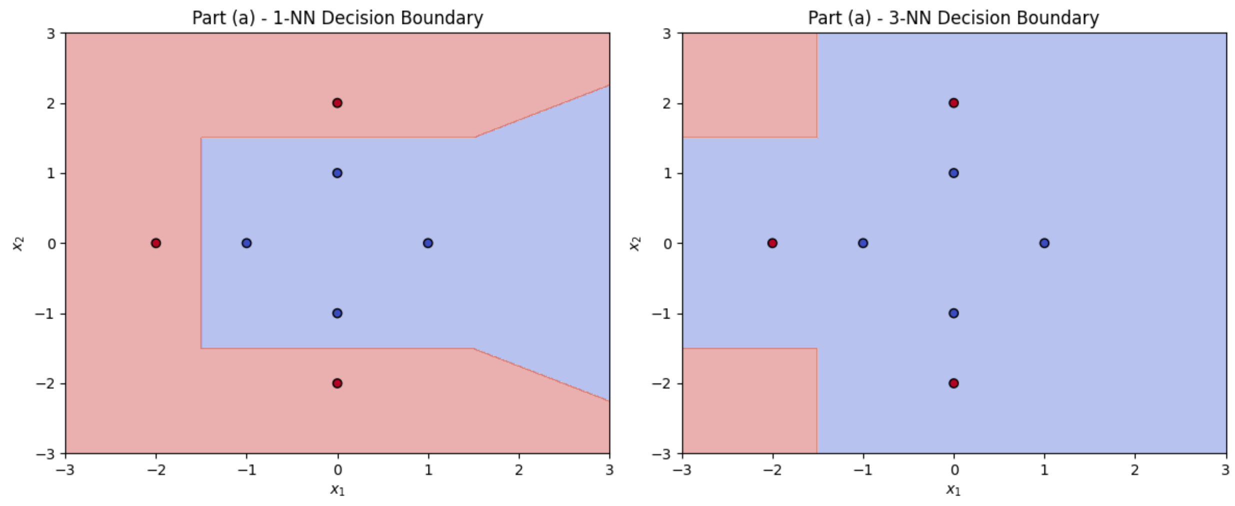
Problem 6.1

Consider the following data set with 7 data points

$$(\begin{bmatrix}1\\0\end{bmatrix}, -1)(\begin{bmatrix}0\\1\end{bmatrix}, -1)(\begin{bmatrix}0\\-1\end{bmatrix}, -1)(\begin{bmatrix}-1\\0\end{bmatrix}, -1)$$

$$(\begin{bmatrix}0\\2\end{bmatrix}, +1)(\begin{bmatrix}0\\-2\end{bmatrix}, +1)(\begin{bmatrix}-2\\0\end{bmatrix}, +1)$$

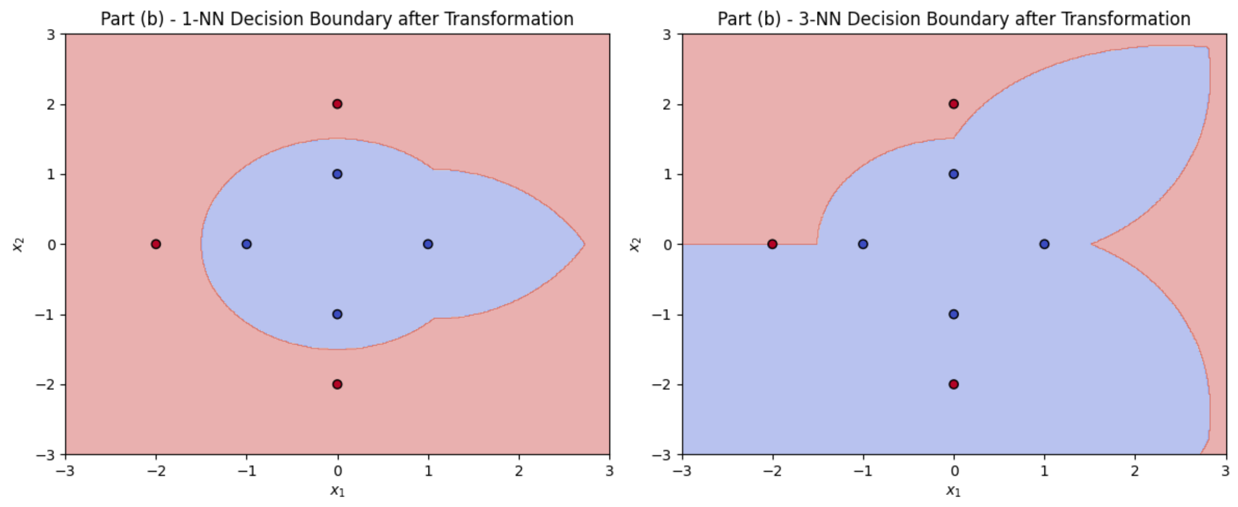
(a) Show the decision regions for the 1-NN and 3-NN rules.



(b) Consider the non-linear transform

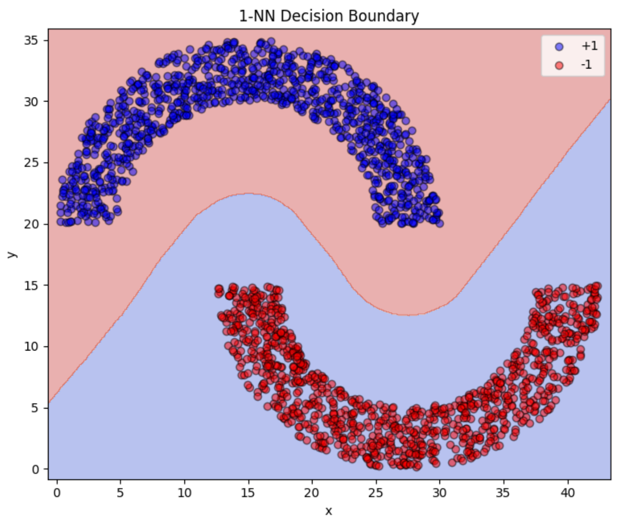
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \Longrightarrow \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} \sqrt{x_1^2 + x_2^2} \\ \arctan(x_2/x_1) \end{bmatrix}$$

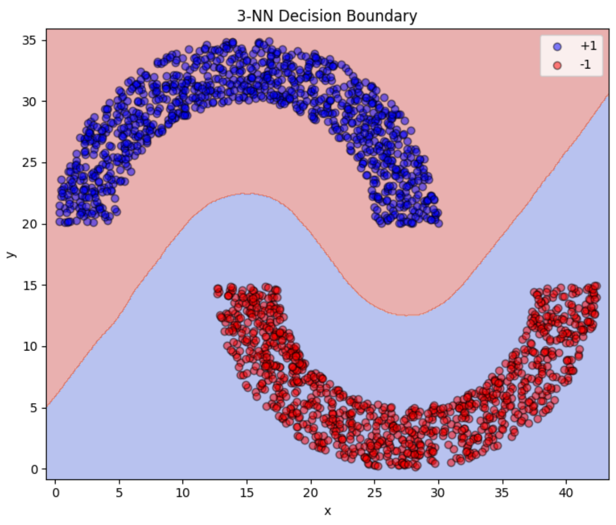
which maps x to z. Show the classification regions in the x-space for the 1-NN and 3-NN rules implemented on the data in the z-space.



Problem 6.4

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





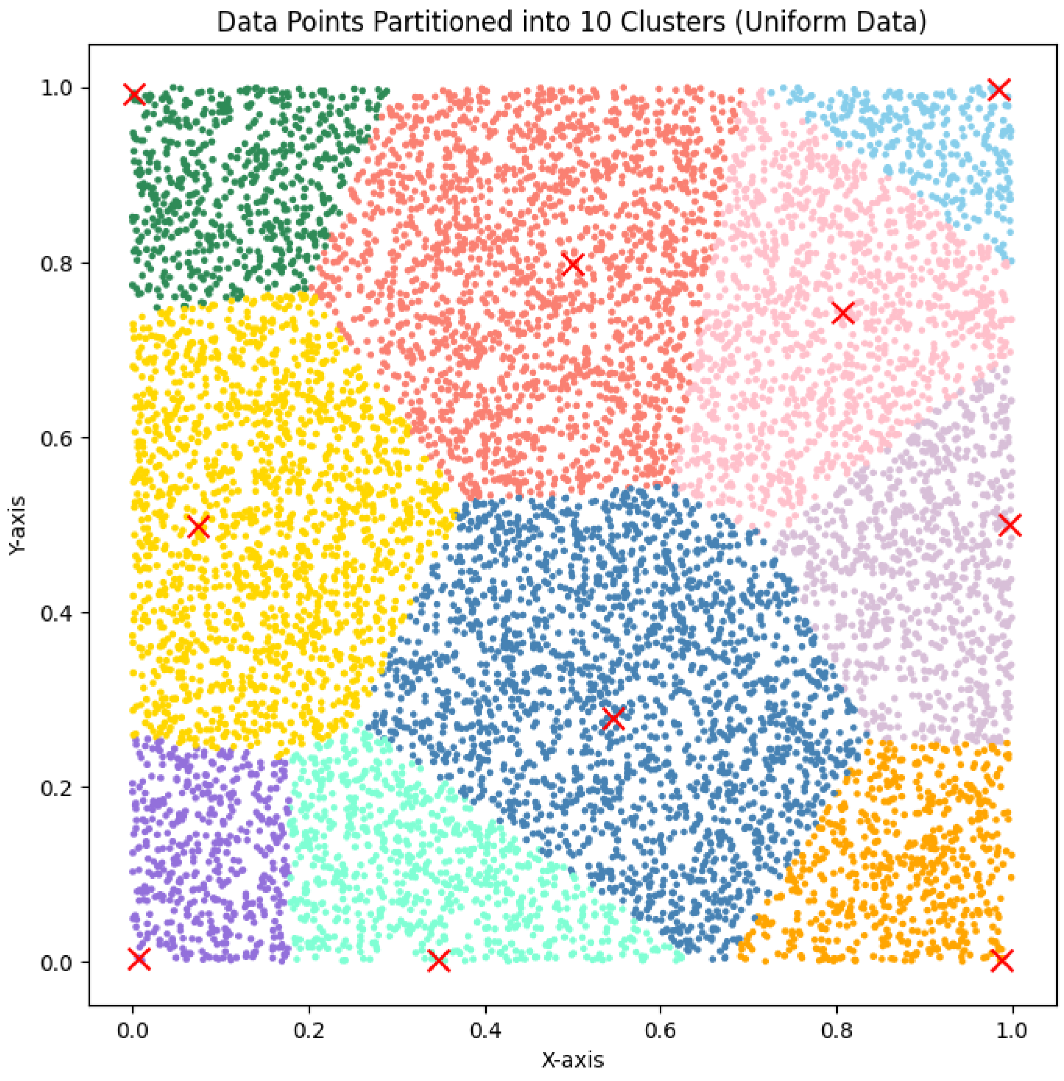
Problem 6.16

Deterministic noise depends on H, as some models approximate f better than others.

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

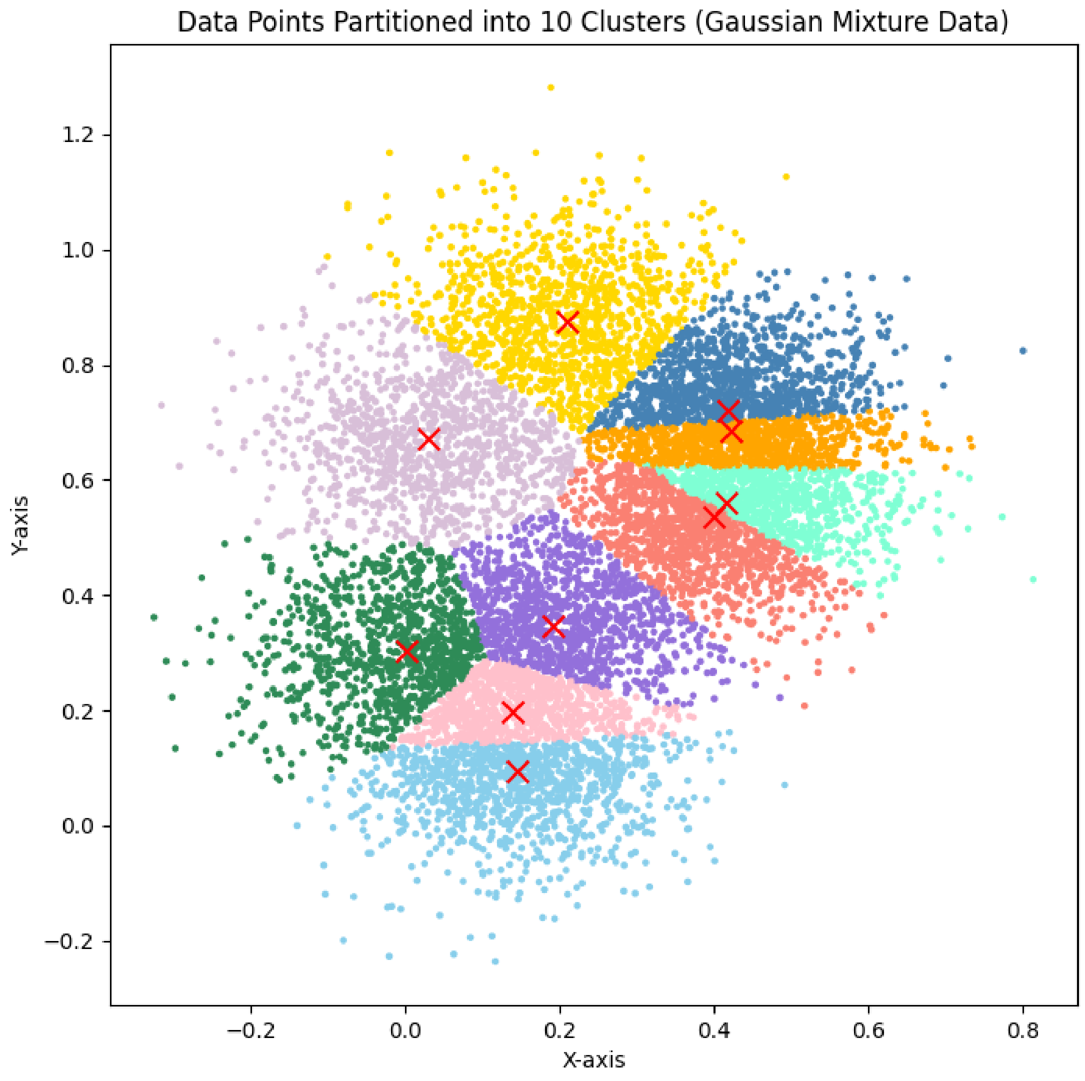
(i) Construct a 10-partition for the data using the simple greedy heuristic described in the text.

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



Branch and Bound: 37.18 seconds
Brute Force: 7.17 seconds

(b) Repeat (a) but instead generate the data from a mixture of 10 gaussians with centers randomly distributed in $[0, 1]^2$ and identical covariances for each bump equal to $\sigma^2 I$ where $\sigma = 0.1$.



Branch and Bound: 41.12 seconds
Brute Force: 19.21 seconds

(c) Explain your observations.

The branch and bound method significantly reduced the running time compared to the brute force approach in both the uniform and Gaussian mixture datasets. The speedup was more pronounced in the uniform data (about 5 times faster) than in the Gaussian mixture data (about 2 times faster). This is because, in the uniform data, clusters are well-separated, allowing the branch and bound method to effectively prune distant clusters and minimize unnecessary distance calculations. In contrast, the Gaussian mixture data has overlapping clusters due to the nature of the Gaussians with $\sigma = 0.1$, reducing the efficiency of pruning and requiring more computations within multiple clusters.

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

Yes, the decision depends on the number of test points. When evaluating a large number of test points, the initial overhead of constructing partitions is offset by the cumulative time saved during nearest neighbor searches, making the branch and bound technique advantageous. If the number of test points is small, the overhead may not be justified, and the brute force method might be more efficient overall. Therefore, the branch and bound technique is more beneficial when dealing with large datasets and numerous queries, where the per-query time savings significantly impact total performance.