

CS 480/680

Introduction to Machine Learning

Lecture 4

Non-Parametric Methods: KDE, k-NN, and k-Means

Kathryn Simone

19 September 2024



UNIVERSITY OF
WATERLOO

FACULTY OF
MATHEMATICS

Feedback from last class

	Changes
START Communicating expectations	Sample midterm will be made available mid-October Emphasize and clarify learning objectives
STOP Going through the math so fast	Incorporate pauses at the end of a section Make explicit references to the text ? Actively work out examples
CONTINUE Giving visual/Intuitive explanations and concrete examples	More examples of real-world applications of ML Write on tablet instead of blackboard ? Show code used to create visuals

The algorithms we have encountered so far involved assuming a hypothesis class and parametrization

Perceptron: Separating Hyperplane

$$\mathcal{H}_+ = \{x : w^T x > -b\}$$

$$\mathcal{H}_- = \{x : w^T x < -b\}$$

Linear Regression:

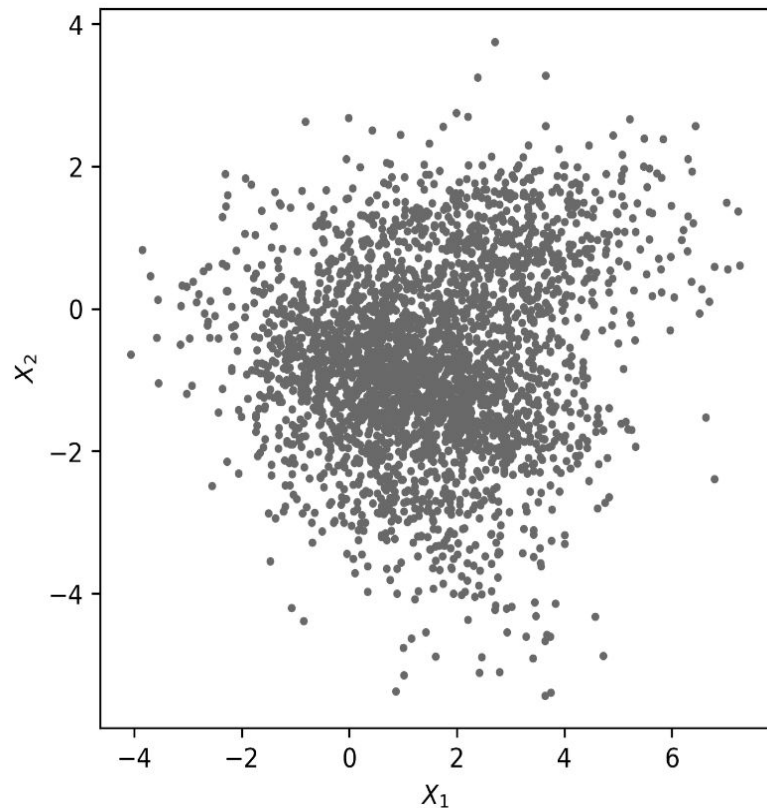
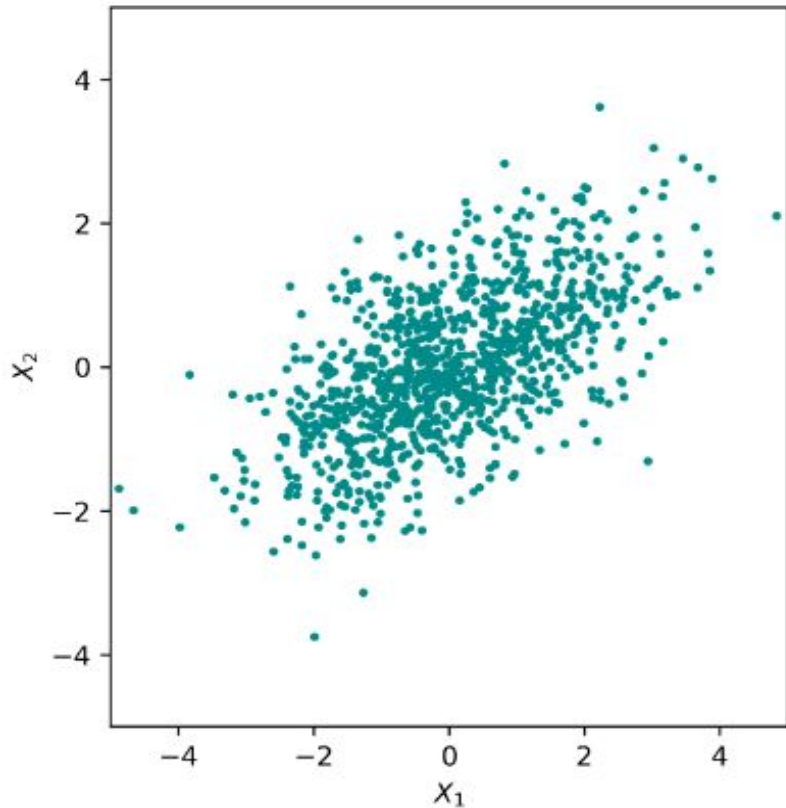
$$y(x) = w^T x + b$$

Gaussian distribution learning

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$



Strong assumptions limit the class of functions you can approximate well



Learning a distribution without a hypothesis class

Kernel (Parzen) Density Estimation

Given a set of observations $\{x_1, x_2, \dots, x_n\}, x_i \in \mathbb{R}^d$, the Parzen estimate for the probability density at some arbitrary point x_o is calculated as

$$\hat{p}_X(x_o) = \frac{1}{n\lambda} \sum_{i=1}^n K_\lambda(x_o, x_i)$$

Where: K_λ is the kernel function,
 λ is the bandwidth or length scale parameter.

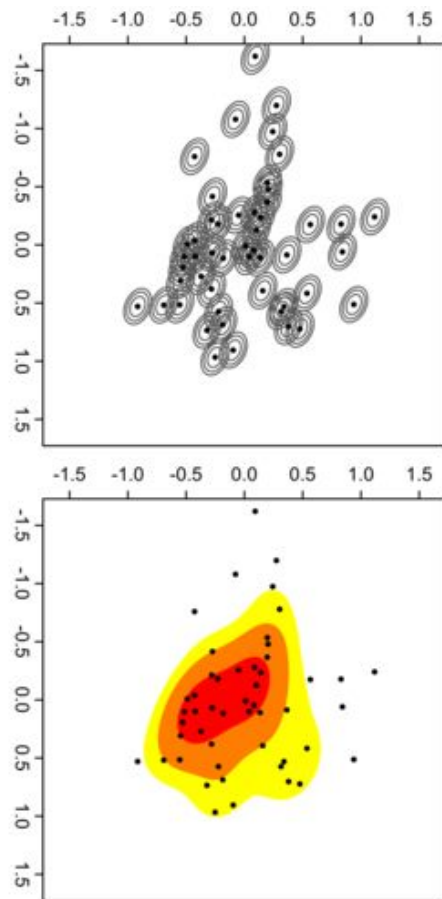
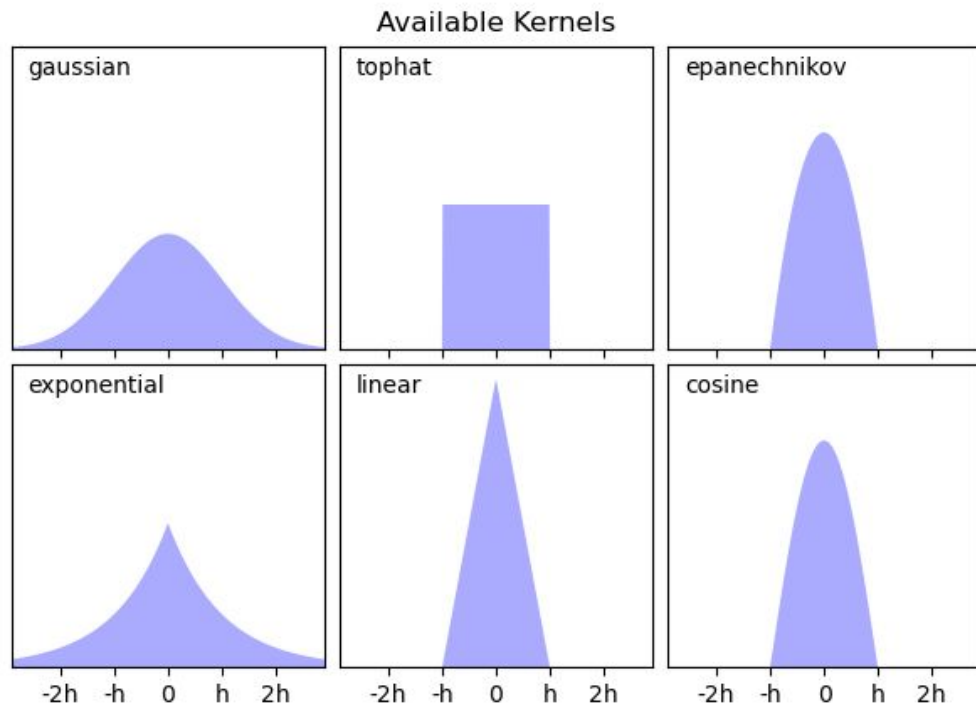


Figure: Wikipedia

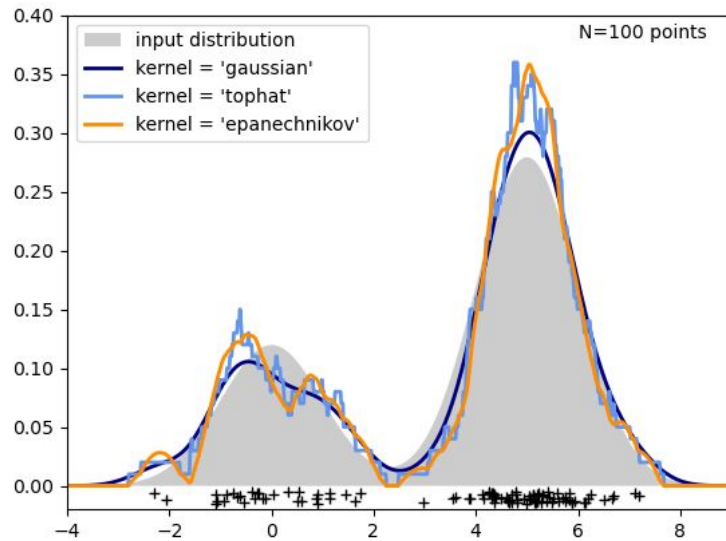
Elements of Statistical Learning, Section 6.6

Many admissible kernels for density estimation



Using KDE to approximate the reference in KL divergence

$$D_{\text{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \text{ (Continuous Random Variable)}$$



Today we'll look at examples of non-parametric algorithms

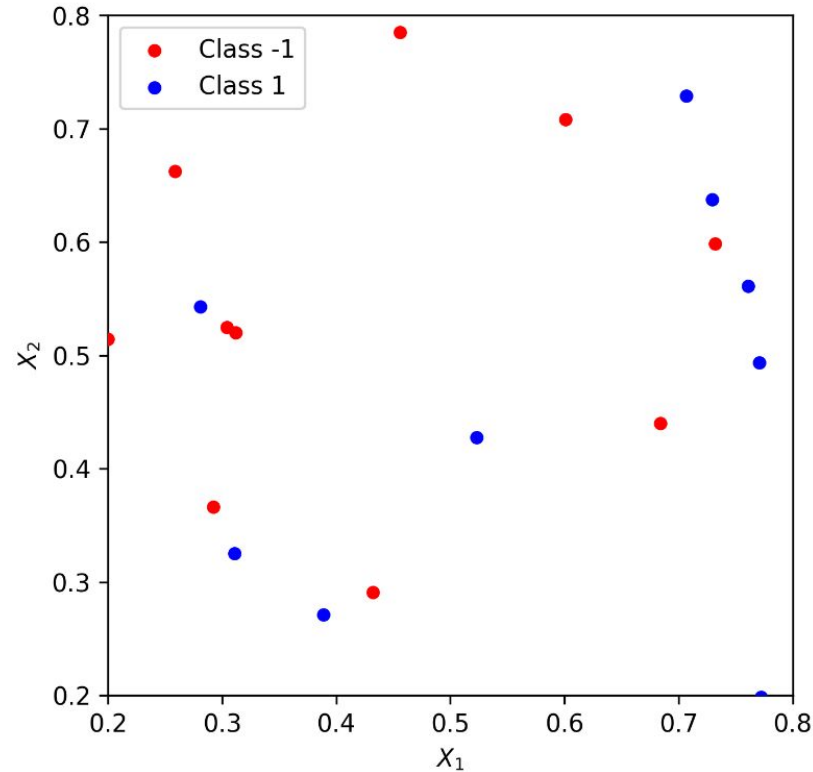
- I. Classification (k-Nearest Neighbors)
- II. Clustering (k-Means)



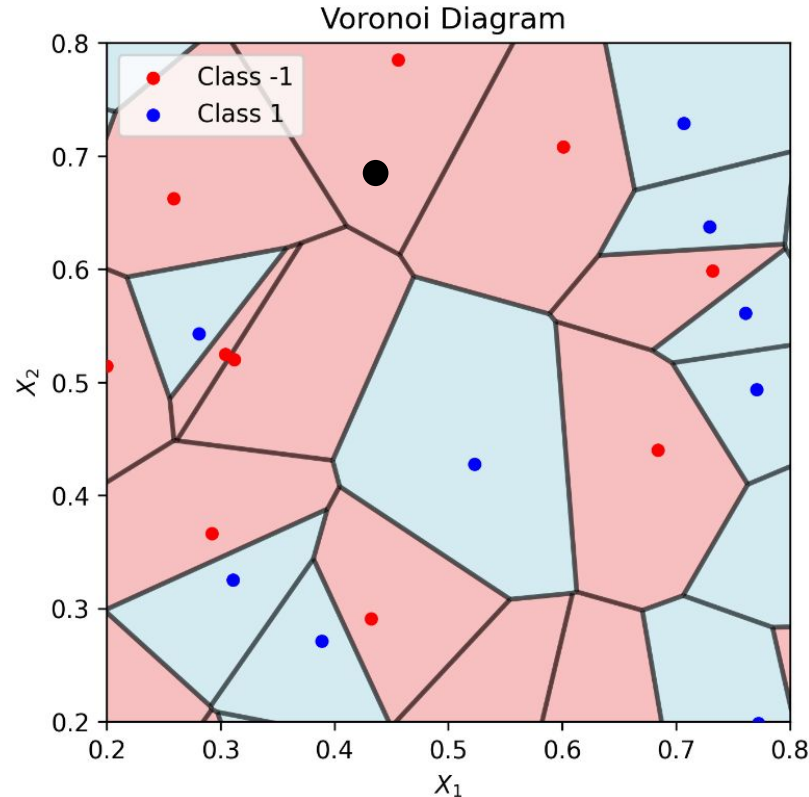
Non-Parametric Supervised Learning

K-Nearest Neighbors (k-NN)

Classification with non-linearly separable data



Approach: assume that points that are near to one another are likely to have the same label

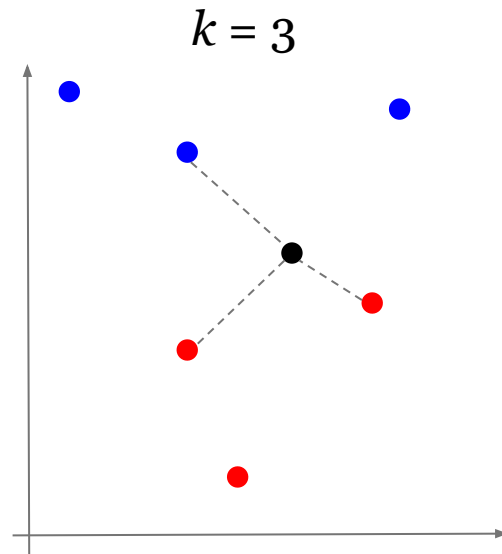


k-Nearest Neighbors (k-NN) Classification

Algorithm 1 k Nearest Neighbors (binary classification)

Input:

- 1: Dataset $\{(x_1, y_1), \dots, (x_n, y_n)\}$, $x_i \in \mathbb{R}^d$, $y_i \in \{\pm 1\}$
 - 2: Number of neighbors k
 - 3: New instance $x \in \mathbb{R}^d$
 - 4: **for** $i = \{1, 2, \dots, n\}$ **do**
 $d_i = \text{dist}(x, x_i)$
 - 5: **end for**
 - 6: Find smallest k indices of d
 - 7: **return** Majority vote of $\{y_i, \dots, y_k\}$
-



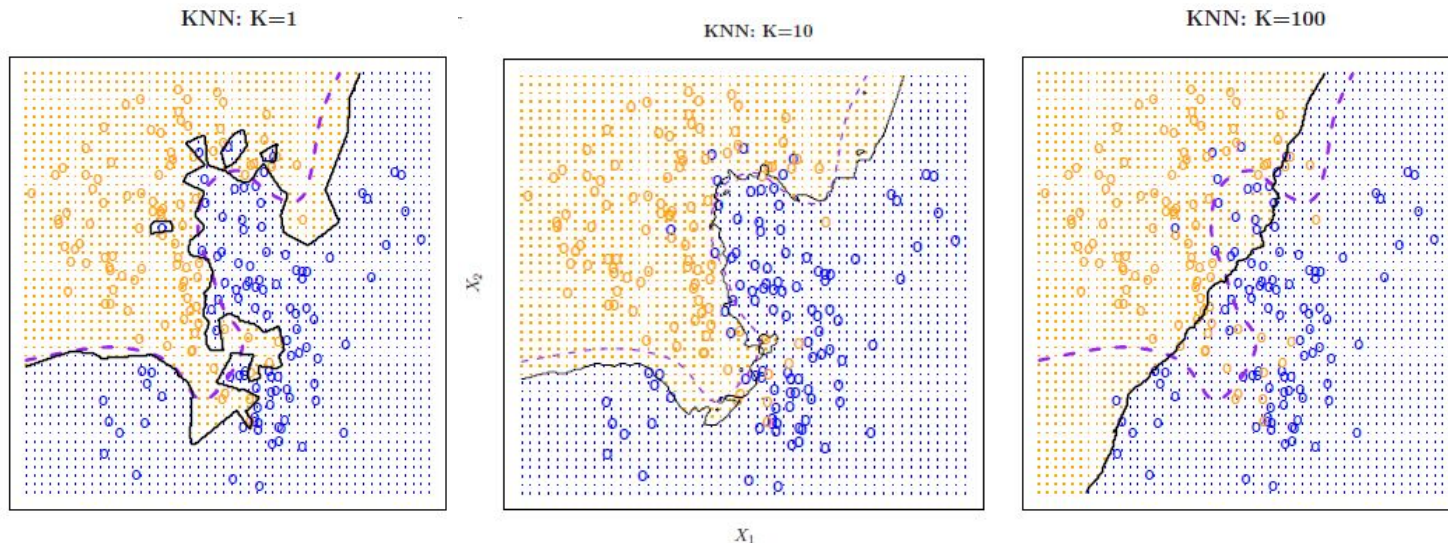
KNN requires a distance function

$$d : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$$

- Euclidean: $d(x, x') = \sqrt{\sum_{m=1}^d (x_m - x'_m)^2}$
- Manhattan: $d(x, x') = \sum_{m=1}^d \|x_m - x'_m\|_1$



The bias-variance tradeoff and the role of hyperparameter k



Variable:

Predictions depend strongly on training dataset

Biased:

systematic error due to simplified model

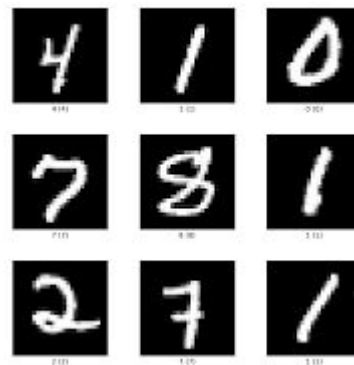
Computational complexity of KNN

	Time	Space
Train	0	$O(nd)$
Test	$O(ndk)$	$O(nd)$



KNN on handwritten digit recognition (MNIST)

CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)
Linear Classifiers		
linear classifier (1-layer NN)	none	12.0
linear classifier (1-layer NN)	deskewing	8.4
pairwise linear classifier	deskewing	7.6
K-Nearest Neighbors		
K-nearest-neighbors, Euclidean (L2)	none	5.0
K-nearest-neighbors, Euclidean (L2)	none	3.09
K-nearest-neighbors, L3	none	2.83
K-nearest-neighbors, Euclidean (L2)	deskewing	2.4
Convolutional nets		
Convolutional net LeNet-1	subsampling to 16x16 pixels	1.7
Convolutional net LeNet-4	none	1.1
Convolutional net LeNet-4 with K-NN instead of last layer	none	1.1
Convolutional net LeNet-4 with local learning instead of last layer	none	1.1
Convolutional net LeNet-5, [no distortions]	none	0.95
Convolutional net LeNet-5, [huge distortions]	none	0.85
Convolutional net LeNet-5, [distortions]	none	0.8
Convolutional net Boosted LeNet-4, [distortions]	none	0.7
Trainable feature extractor + SVMs [no distortions]	none	0.83
Trainable feature extractor + SVMs [elastic distortions]	none	0.56



An upper bound on KNN error rate (Cover and Hart, 1967)

The Bayes optimal classifier outputs the most likely class under a known distribution P :

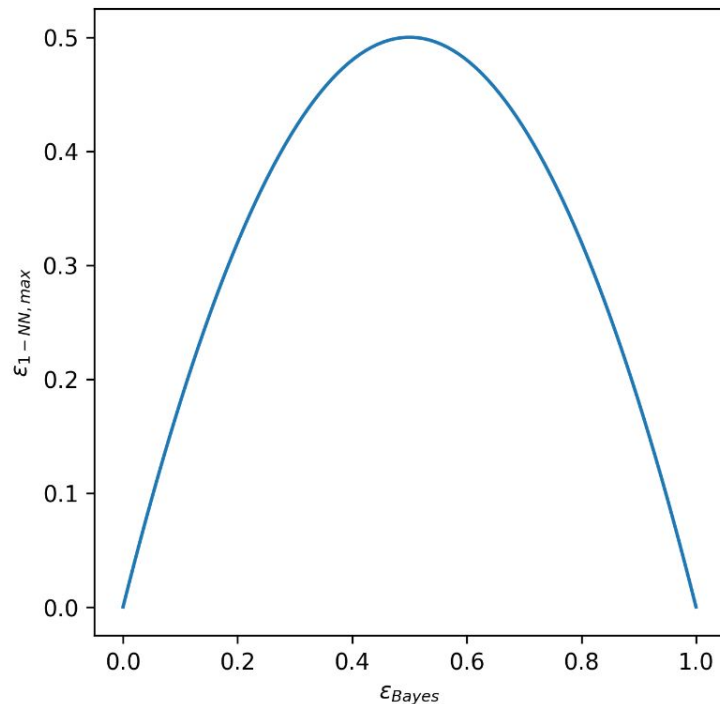
$$\hat{y}(x) = \operatorname{argmax}_c \Pr[y = c|x, P]$$

- Given a point x , look at the distribution of labels given that feature vector
- Pick whichever label is most likely to be generated
- Error rate $\epsilon : E[1 - \max_c \Pr[y = c|x, P]]$

Suppose $KNN(x)$ is a Nearest Neighbor binary classifier with $k=1$, then its error rate ϵ_h

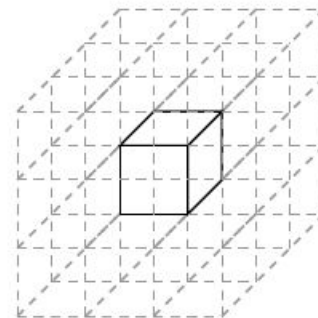
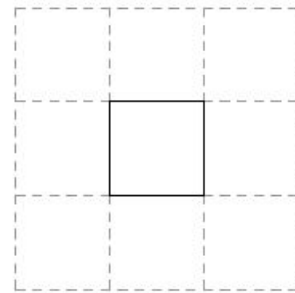
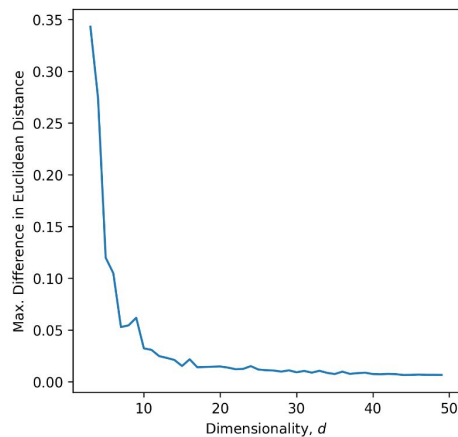
$$\lim_{n \rightarrow \infty} \epsilon_{KNN} < 2\epsilon_{Bayes}(1 - \epsilon_{Bayes})$$

Where ϵ_{Bayes} is the error rate of the Bayes optimal classifier, and n is the number of samples.



Caveat of the Cover-Hart Theorem

The “curse of dimensionality” (Bellman, 1961)



The number of samples needed to train the KNN classifier to reach the lower bound on the error rate grows exponentially with the number of features.

Right: Michael Betancourt, 2018

Discussion: Elements of Statistical Learning, Section 2.5,

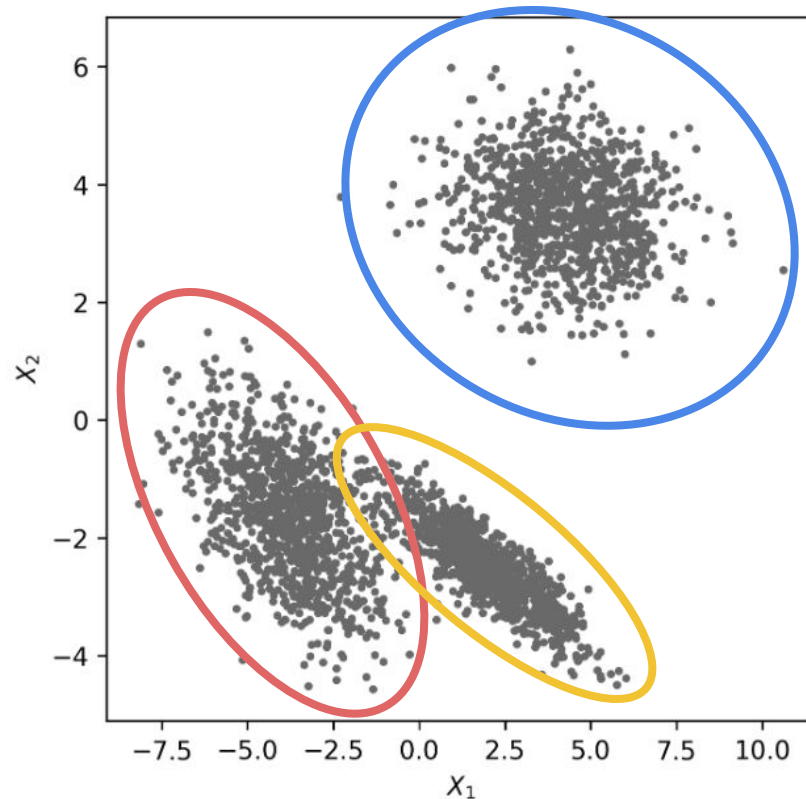
Proof: Understanding Machine Learning, Section 19.2.2



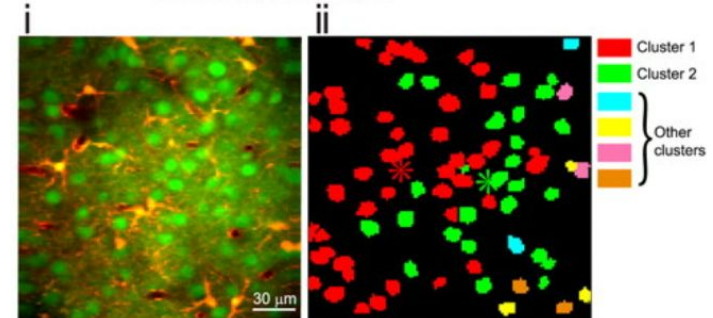
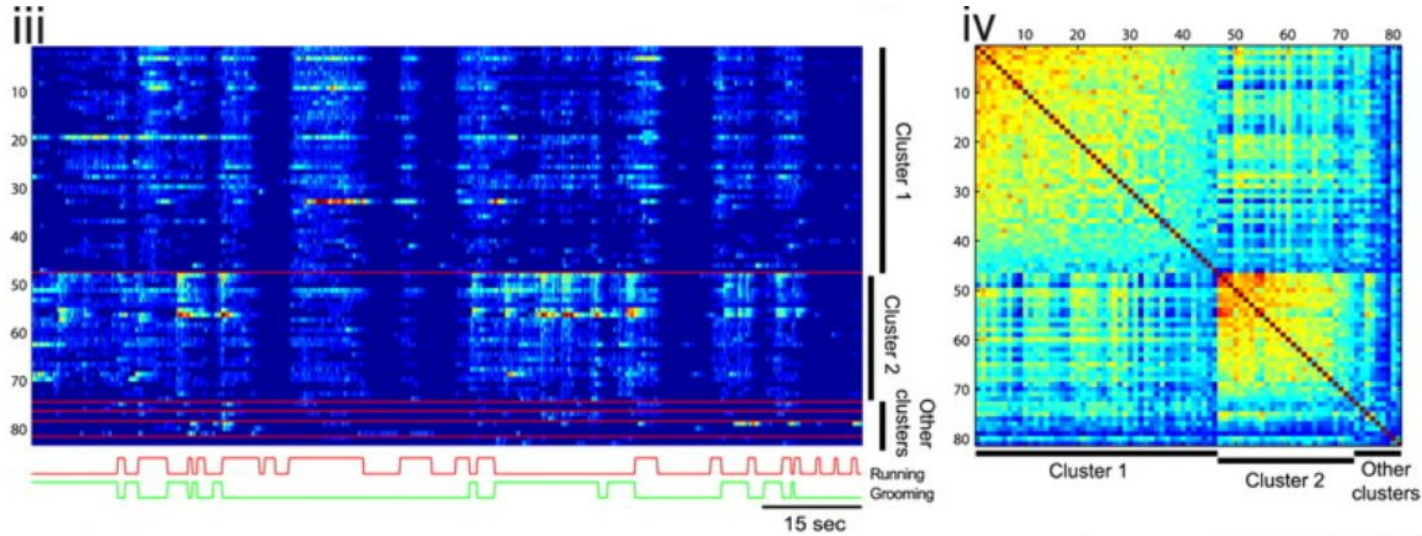
K-Means Clustering

Non-Parametric Clustering

The clustering task concerns grouping unlabelled data



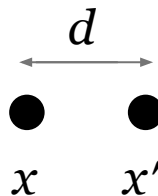
Example: Clustering applied to neural data in motor cortex



A common clustering problem

Given: A dataset of n observations, $D = \{x_1, x_2, \dots, x_n\}$, $x_i \in \mathbb{R}^d$, and function $f : \mathbb{R}^d \rightarrow \mathbb{R}_+$

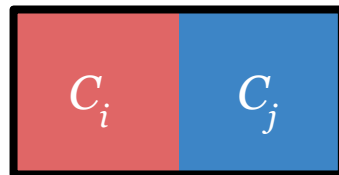
- f is symmetric: $f(x, x') = f(x', x) \quad \forall x, x'$
- If f is a distance function, $f(x, x) = 0 \quad \forall x$,
and $f(x, z) \leq f(x, y) + f(y, z)$
- If f is a similarity measure,
 $f : \mathbb{R}^d \rightarrow [0, 1]$, $f(x, x) = 1 \quad \forall x$



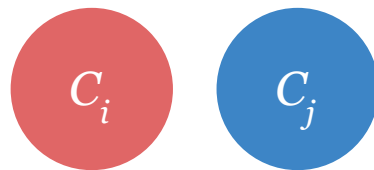
Goal: A partition of the set into a set of subsets
 $C = \{C_1, C_2, \dots, C_k\}$.

- Each observation belongs to a subset:
 $\cup_{i=1}^k C_i = D$
- Each observation belongs to only one subset:
 $C_i \cap C_j = \emptyset, \forall i \neq j$

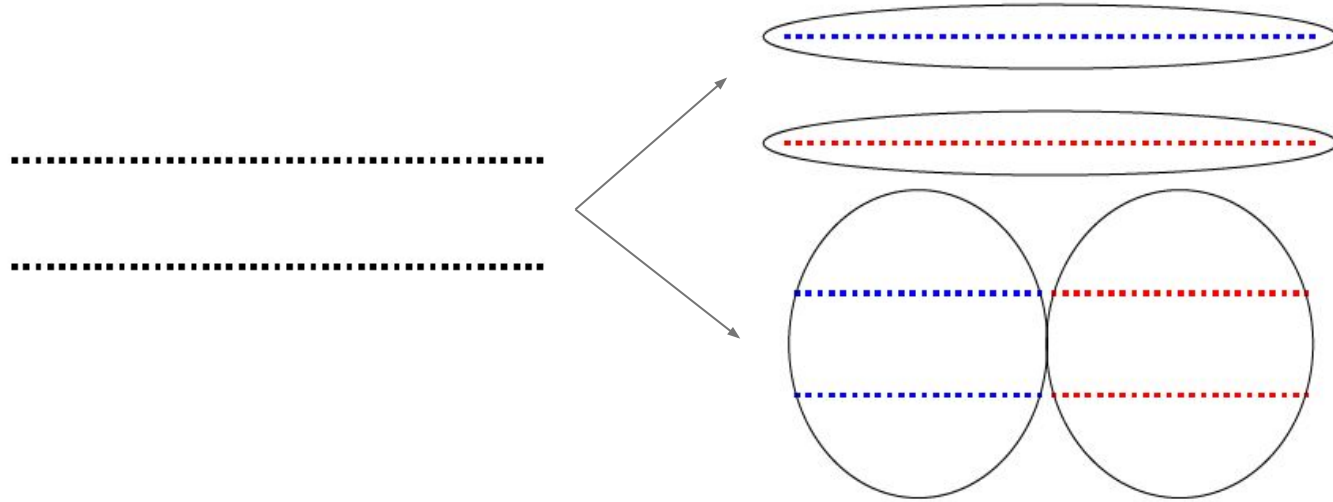
Collectively
Exhaustive



Disjoint or
mutually exclusive



Clustering output depends on what we mean by similarity



The k-Means clustering objective

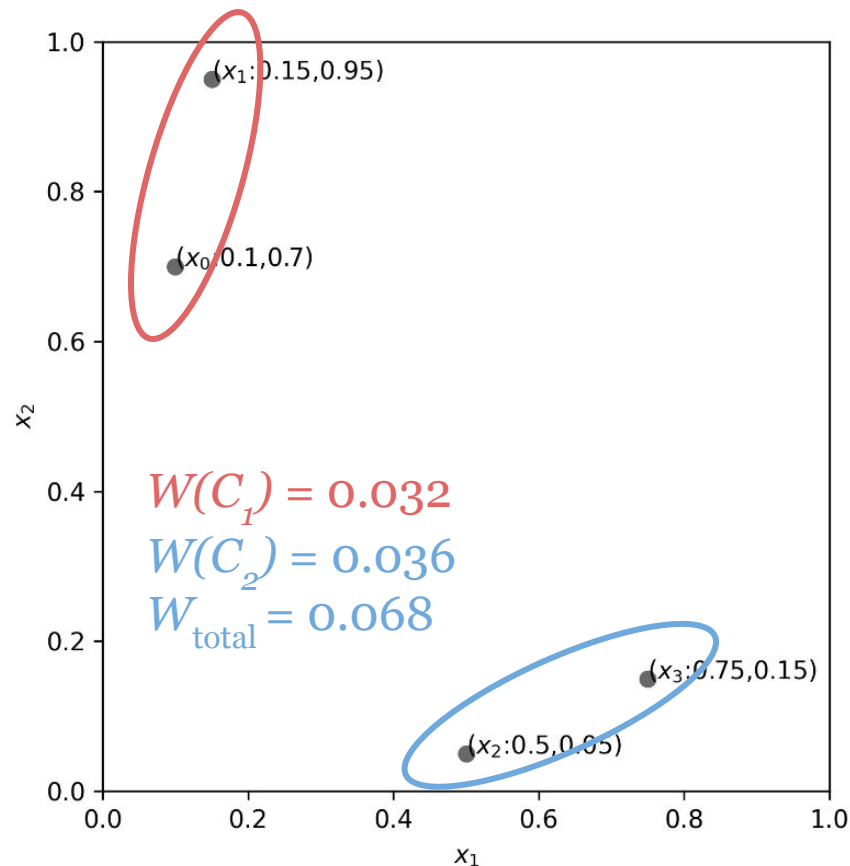
Objective:

Cost of C_j , $W(C_j)$

$$\min_{C_1, \dots, C_k} \sum_{j=1}^k \frac{1}{|C_j|} \sum_{x_i, x'_i \in C_j} \|x_i - x'_i\|_2^2,$$

Where:

$|C_j|$ is the number of points in cluster j



The k-Means clustering objective

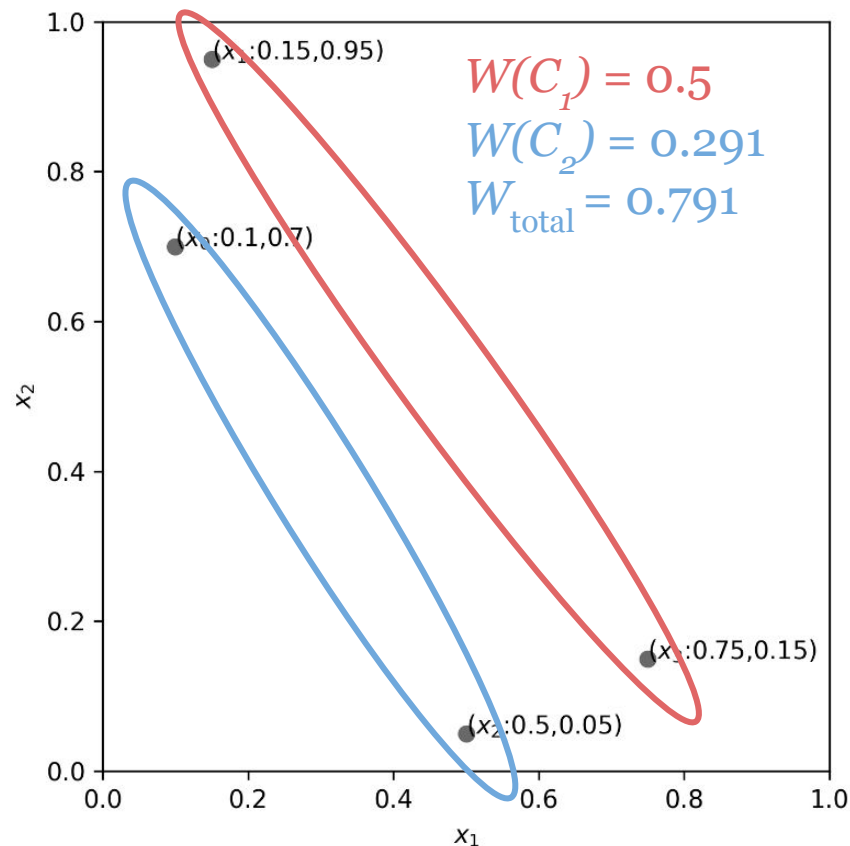
Objective:

Cost of C_j , $W(C_j)$

$$\min_{C_1, \dots, C_k} \sum_{j=1}^k \frac{1}{|C_j|} \sum_{x_i, x'_i \in C_j} \|x_i - x'_i\|_2^2,$$

Where:

$|C_j|$ is the number of points in cluster j



Reformulating the k -Means objective

The j^{th} cluster has a centroid $\mu_j \in \mathbb{R}^d$:

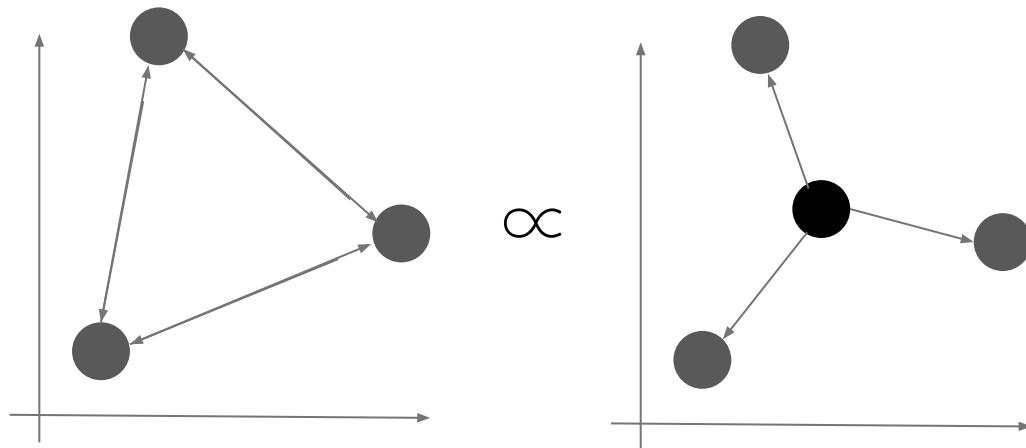
$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

It can be shown that

$$\sum_{x_i, x'_i \in C_j} \|x_i - x'_i\|_2^2 \propto \sum_{x_i \in C_j} \|x_i - u'_j\|_2^2,$$

therefore, the objective can be rewritten as:

$$\min_{C_1, \dots, C_k} \sum_{j=1}^k \frac{1}{|C_j|} \sum_{x_i \in C_j} \|x_i - \mu_j\|_2^2.$$



How to minimize the objective function?

- If you do brute-force search, you'll have a bad time
 - There are K^n possible clusters (an NP-hard problem)
 - Would only work for small k and n
- Partial derivative won't be as helpful here
 - Hard-membership means the gradient is non-smooth
 - Multiple local minima



Lloyd's algorithm for K-Means clustering


Algorithm 1 Lloyd’s k -Clustering method

Input:

1: Dataset $D = \{x_1, x_2 \dots, x_n\}$ $x_i \in \mathbb{R}^d$

2: Number of clusters k

Output: Partition $C = \{C_1, C_2, \dots, C_k\}$

3: Randomly initialize partition $C^0 = \{C_1, C_2, \dots, C_k\}$ 4: **while** $\exists i C_i^{(t)} \neq C_i^{(t-1)}$ **do**5: **for** $C^t = \{C_1^t, C_2^t, \dots, C_k^t\}$ **do**

$$\mu_j = \frac{1}{|C_j^t|} \sum_{x_i \in C_j^t} x_i$$

6: \triangleright Compute centroid of cluster j \blacktriangleleft

7: end for

8: **for** $\{x_1, x_2 \dots, x_n\}$ **do**

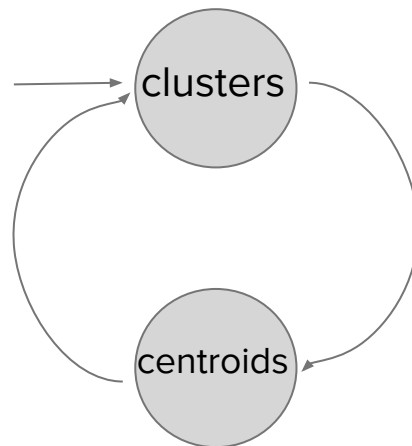
$$j = \operatorname{argmin}_j \|x_i - \mu_j\|_2^2$$

$$C_i^{t+1} \cup \{x_i\}$$

▷ Reassign sample i to cluster j ◀

9: end for

```
10: end while
```

11: **return** Partition $C = \{C_1, C_2, \dots, C_k\}$ 

Lloyd's algorithm in action

Algorithm 1 Lloyd’s k -Clustering method

Input:

1: Dataset $D = \{x_1, x_2 \dots, x_n\}$ $x_i \in \mathbb{R}^d$

2: Number of clusters k

Output: Partition $C = \{C_1, C_2, \dots, C_k\}$

3: Randomly initialize partition $C^0 = \{C_1, C_2, \dots, C_k\}$

4: **while** $\exists i C_i^{(t)} \neq C_i^{(t-1)}$ **do**5: **for** $C^t = \{C_1^t, C_2^t, \dots, C_k^t\}$ **do**

$$\mu_j = \frac{1}{|C_j^t|} \sum_{x_i \in C_j^t} x_i$$


6: ▷ Compute centroid of cluster j ◀◀

7: end for

8: **for** $\{x_1, x_2 \dots, x_n\}$ **do**

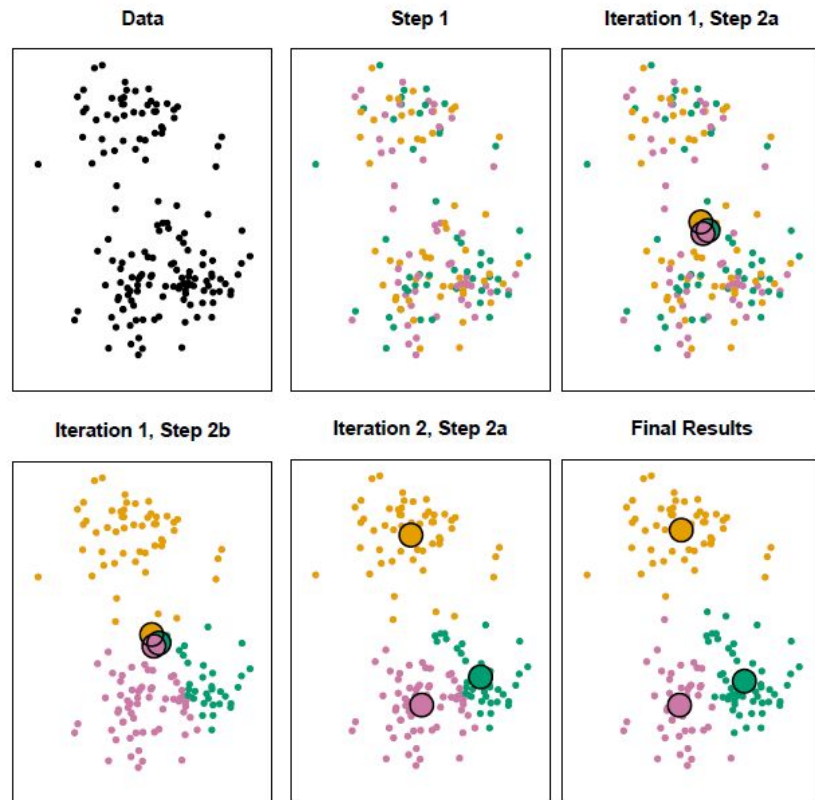
$$j = \operatorname{argmin}_j \|x_i - \mu_j\|_2^2$$

$$C_i^{t+1} \cup \{x_i\}$$

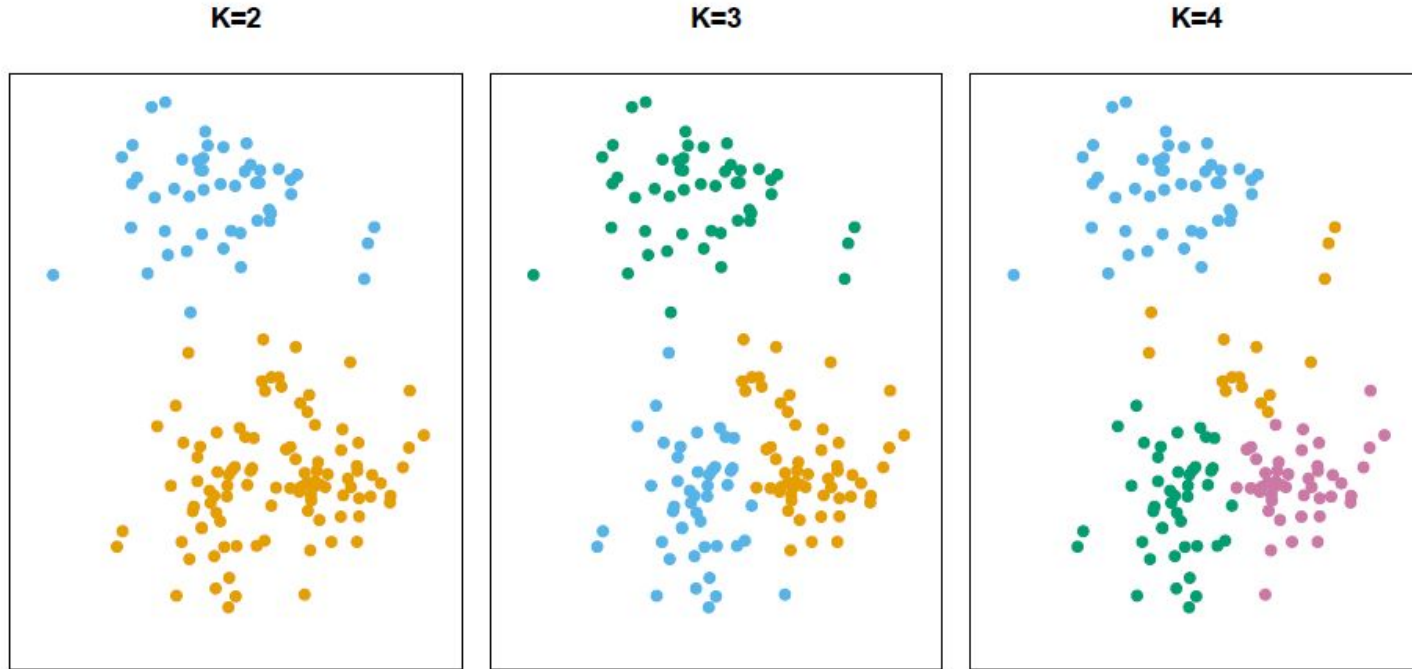
- ▷ Reassign sample i to cluster j 

9: end for

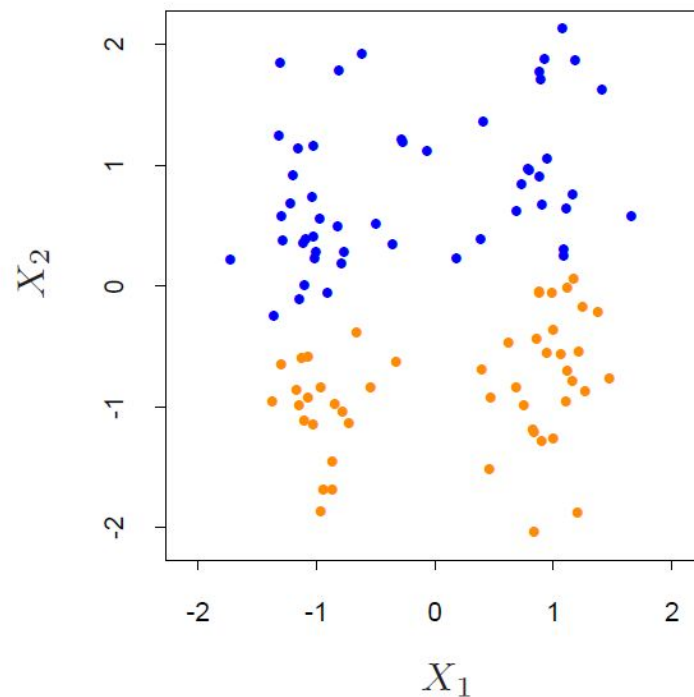
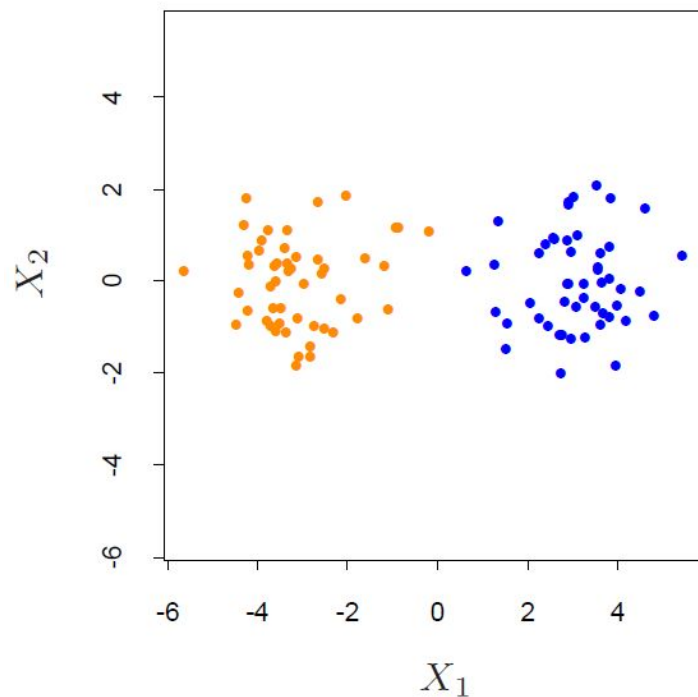
```
10: end while
```

11: **return** Partition $C = \{C_1, C_2, \dots, C_k\}$ 

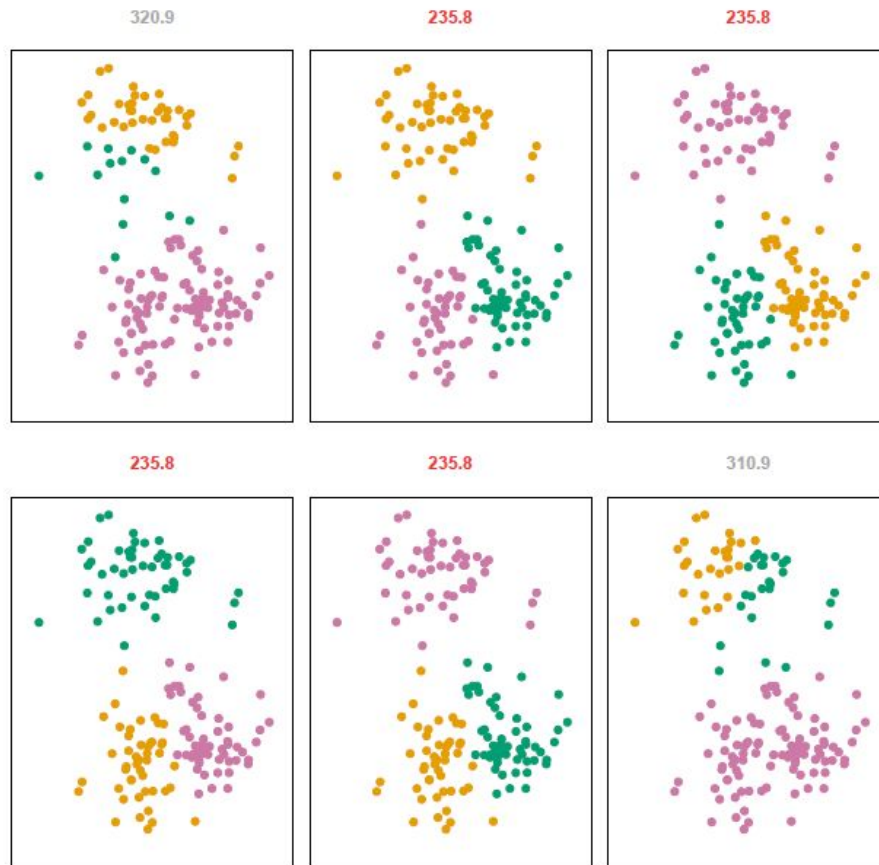
Limitations of k -Means



Limitations of k -Means



Limitations of Lloyd's Algorithm



Summary

Parametric vs. Non-parametric Methods

	Parametric	Non-Parametric
Assumptions about the Data		
Complexity of the model		
Adaptability to new data		
Data Efficiency		



Now that we're at the end of the lecture, you should be able to...

- ★ Differentiate **parametric and non-parametric algorithms**, and give examples of algorithms in unsupervised and supervised settings.
- ★ Defend k -NN on both empirical and theoretical grounds, with reference to the **Bayes optimal classifier**.
- ★ State the Cover-Hart theorem and anticipate its implications on real datasets, with reference to the **curse of dimensionality**.
- ★ Characterize ML algorithms by their compromise of the **bias-variance** tradeoff.
- ★ Use kernel density estimation (KDE) to fit an empirical distribution.
- ★ Apply KDE to assess a learned parametric model with KL divergence.
- ★ State the clustering problem and differentiate it from classification.
- ★ Implement Lloyd's algorithm for K-Means clustering.
- ★ Implement k -NN classification with $O(ndk)$ at test.
- ★ Design k -NN algorithms and justify choices against alternatives.
- ★ Recommend k -NN in appropriate supervised learning settings.