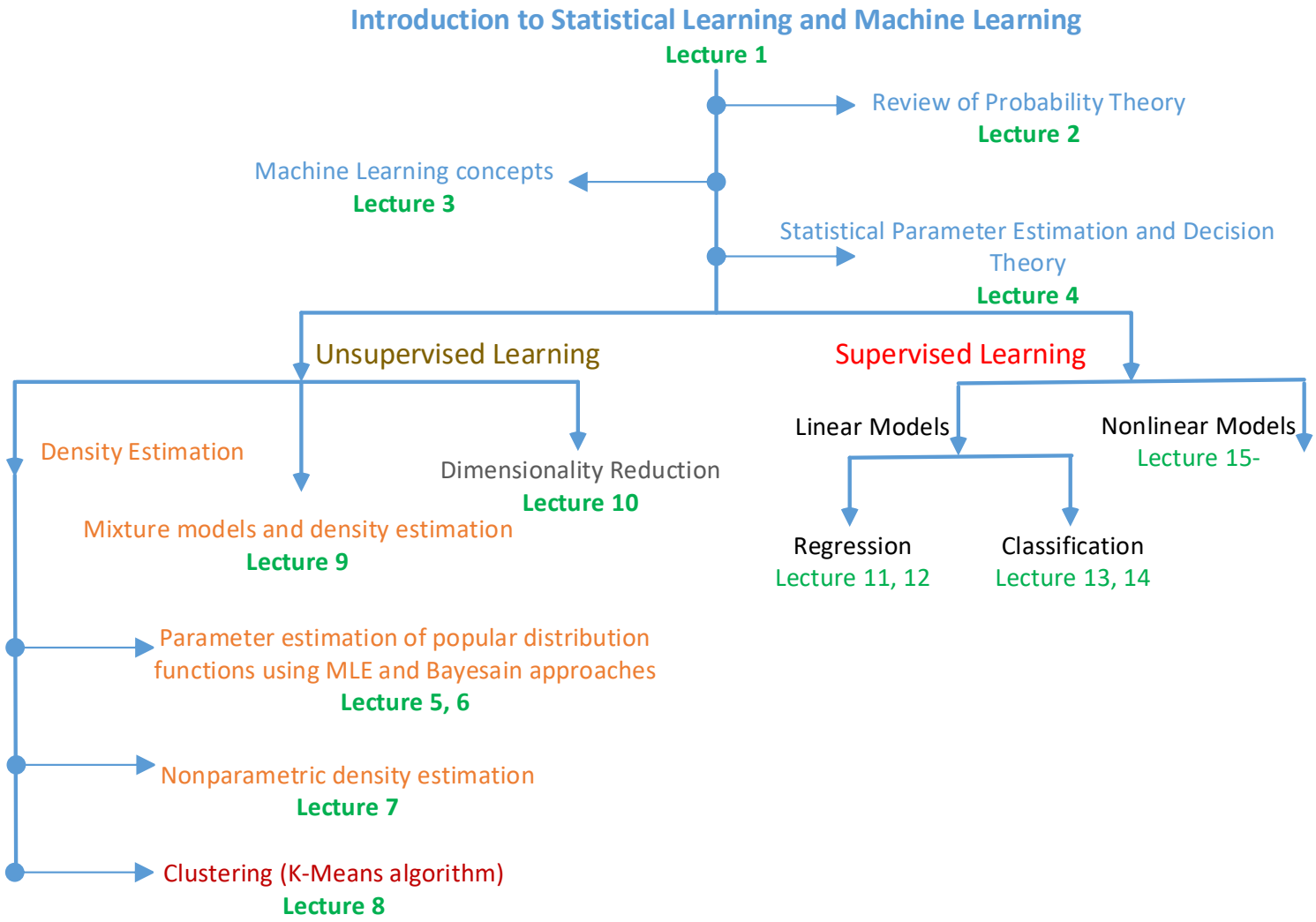
Statistical Learning and Machine Learning

Lecture 9 - Expectation Maximization

September 25, 2021

Course overview and where do we stand



Objectives of the lecture

- Review of mixture of Gaussians model
- Parameter estimation for mixture of Gaussians using Maximum Likelihood Estimation (and EM algorithm)
- Nearest Centroid vs K-Nearest Neighbour Classification

Mixture of Gaussians

Reminder: When using a superposition of K Gaussians, the resulting distribution has the form:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{9}$$

and is called mixture of Gaussian.

The parameters π_k are called *mixing coefficients* and satisfy:

$$0 \leq \pi_k \leq 1, \qquad \sum_{k=1}^{K} \pi_k = 1.$$

Thus, the mixing coefficients have the form of probabilities ($\pi_k = p(k)$).

Mixture of Gaussians: Latent variables

Let $\boldsymbol{z}$ be a binary random variable having a 1-of-K representation satisfying:

$$z_k \in \{0, 1\}, \qquad \sum_{k=1}^{K} z_k = 1.$$

$\boldsymbol{z}$ is associated with the marginal distribution $p(\boldsymbol{x})$, which is specified as:

$$p(z_k = 1) = \pi_k$$

Reminder: Since $\boldsymbol{z}$ is a 1-of-K vector:

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

Mixture of Gaussians: Latent variables

The conditional distribution of $\boldsymbol{x}$ for the k-th Gaussian is:

$$p(\boldsymbol{x}|z_k = 1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

and with respect to $\boldsymbol{z}$ is:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k}$$

We can now write $p(\boldsymbol{x})$:

$$p(\boldsymbol{x}) = \sum_{z} p(\boldsymbol{x}, \boldsymbol{z}) = \sum_{z} p(\boldsymbol{z}) p(\boldsymbol{x}|\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k) \tag{10}$$

Compare (10) with (9)!

Mixture of Gaussians: Latent variables

The conditional probability of $\boldsymbol{z}$ given $\boldsymbol{x}$ is (using Bayes' theorem):

$$\begin{aligned} \gamma(z_k) \equiv p(z_k = 1|\boldsymbol{x}) &= \frac{p(z_k = 1)p(\boldsymbol{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\boldsymbol{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \Sigma_j)} \end{aligned} \tag{11}$$

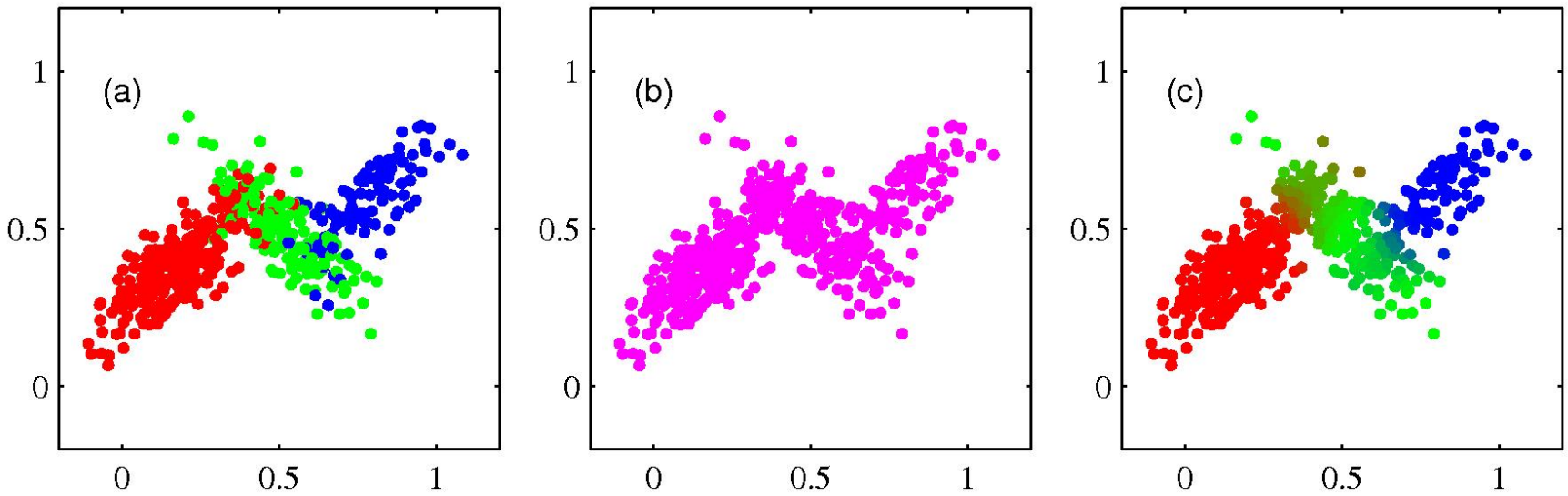
Thus, we consider π_k as the prior probability of $z_k = 1$ and $\gamma(z_k)$ as the corresponding posterior probability after observing $\boldsymbol{x}$ (also called *responsibility* of component k for 'explaining' $\boldsymbol{x}$).

Mixture of Gaussians: Latent variables

Technique to generate random samples described by the Gaussian mixture model:

1. Generate a random vector $\hat{\boldsymbol{z}}$ using $p(\boldsymbol{z})$ (multinomial distribution)
2. Generate $\boldsymbol{x}$ using $p(\boldsymbol{x}|\hat{\boldsymbol{z}}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \Sigma_k)^{\hat{z}_k}$

Use $p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})$, $p(\boldsymbol{x})$ and $\gamma(\boldsymbol{z})$ for visualization.



Mixture of Gaussians: Parameter estimation using MLE

Given a set of i.i.d. data points $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, we want to estimate the parameters of the mixture of Gaussians:

$$\{\boldsymbol{\mu}_k, \ \Sigma_k \ \pi_k\}, \quad k = 1, \ldots, K$$

The log-likelihood function is:

$$\begin{aligned} \ln p(\mathcal{D}|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) &= \ln \prod_{n=1}^{N} p(\boldsymbol{x}_n) \\ &= \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\} \end{aligned}$$

Maximization of the log-likelihood function is more complex than in the case of a single Gaussian. Why?

Mixture of Gaussians: Parameter estimation using MLE

Problem of singularity

- Consider the case where the covariance matrices of Gaussian mixture model are diagonal: $\Sigma_k = \sigma_k^2 \boldsymbol{I}$
- Further consider that $\boldsymbol{\mu}_j = \boldsymbol{x}_n$
- The data point will contribute the following term to the likelihood function:

$$\mathcal{N}(\boldsymbol{x}_n | \boldsymbol{x}_n, \sigma_j^2 \boldsymbol{I}) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sigma_j}$$

  For $\sigma_j \to 0$, the above term will o to infinity and thus maximization of the likelihood function will not make sense.

- Remedy: Detecting the singularity and resetting the mean $\boldsymbol{\mu}_j$ to random value and resetting the covariance to some large value.

Mixture of Gaussians: Parameter estimation using MLE

We set the derivative of the log-likelihood function w.r.t. each parameter equal to zero.

$$
\begin{aligned}
0 &= \frac{\partial \ln p(\mathcal{D}|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\mu}_k} \\
0 &= -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{N} \pi_j \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} \Sigma_k(\boldsymbol{x}_n - \boldsymbol{\mu}_k) \qquad (12)
\end{aligned}
$$

By multiplying with Σ_k^{-1} (assuming Σ_k is nonsingular matrix):

$$
\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n
$$

with $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$ being the effective number of data points assigned to cluster k.

Mixture of Gaussians: Parameter estimation using MLE

We set the derivative of the log-likelihood function w.r.t. Σ_k equal to zero:

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T$$

Thus, Σ_k is the covariance matrix of a single Gaussian fitted to all data, with each data point weighted by the corresponding responsibility value and divided with the effective number of points.

Mixture of Gaussians: Parameter estimation using MLE

To optimize w.r.t. π_k, we need to also consider the constraint that $\sum_{k=1}^{K} \pi_k = 1$. To do so, we use a Lagrange multiplier and maximize for:

$$\ln p(\mathcal{D}|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right)$$

We set the derivative of the above Lagrangian function equal to zero and:

$$\sum_{n=1}^{N} \frac{\mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} + \lambda = 0$$

leading to $\pi_k = N_k/N$ (π_k is the average responsibility of the component to explain the data).

Mixture of Gaussians: Expectation-Maximization

EM for Gaussian Mixtures

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

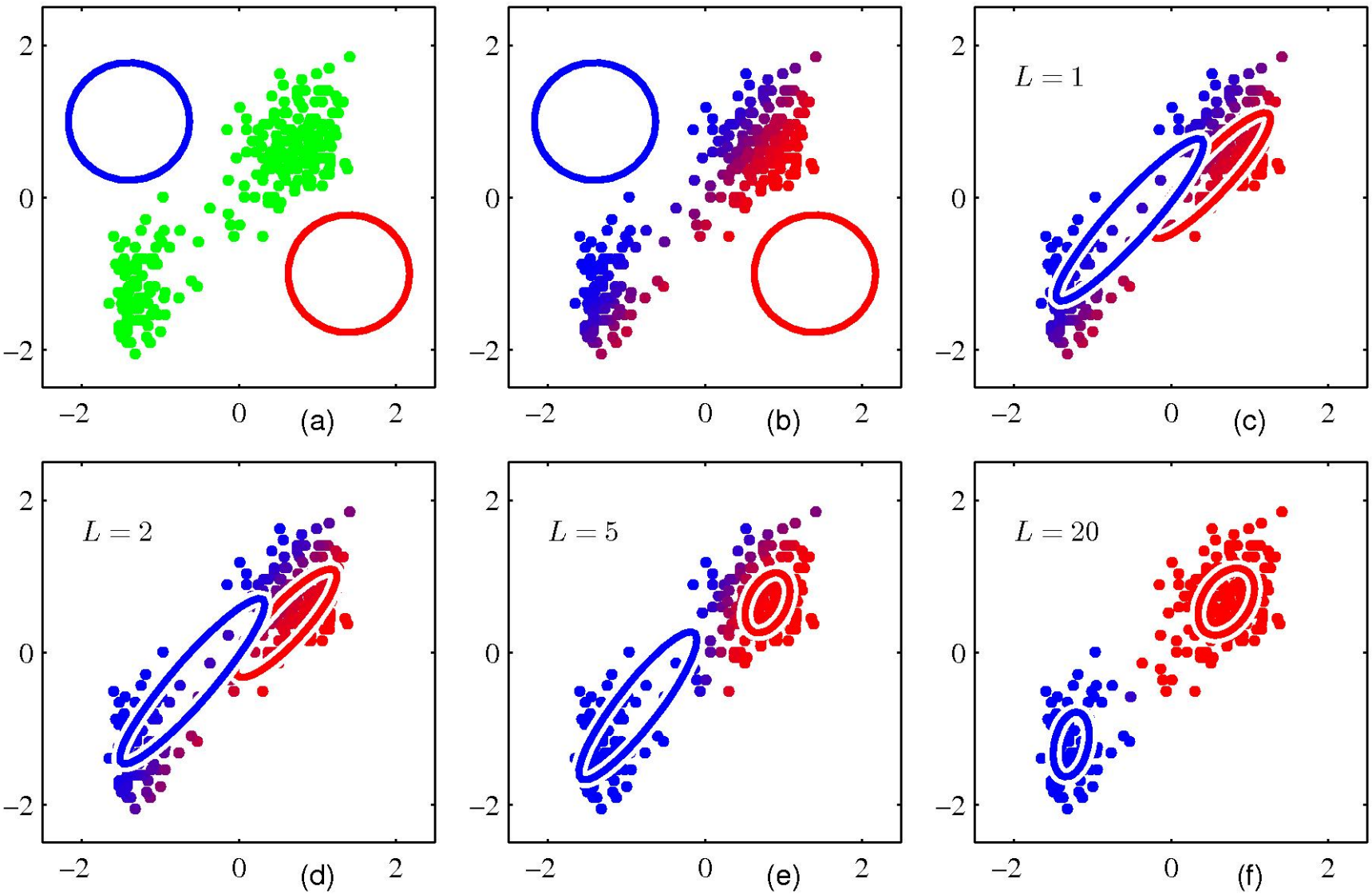
where $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$.

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

Mixture of Gaussians: Expectation-Maximization



Classification: Formal Definition

- Given a training set
  1. Observations in the form of data features
  2. class labels corresponding to each set of observation(s)
- Generate a function/classifier
  1. input: observation/features
  2. output: class label
- The classifier used for classifying previously unseen records

Nearest Centroid-based classification

- Training: compute center or centroid for each of K classes
  1. center of all points of that class
  2. can be viewed as centroid for a cluster as in K-means
- Classification:
  1. assign each data point to the class corresponding to the nearest centroid
- How is this different from k-nearest neighbour classifier?

Nearest Centroid-based classification

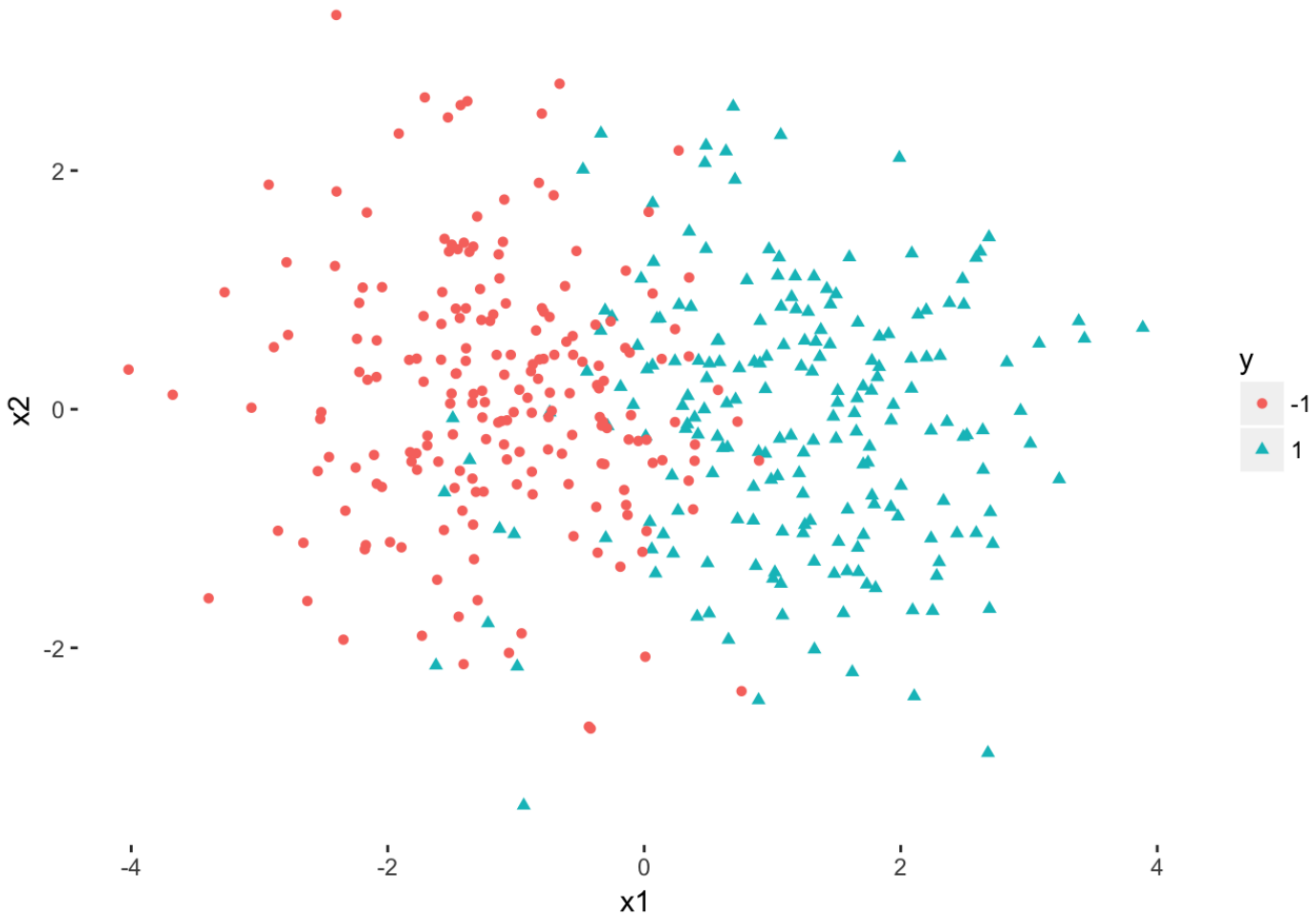


Figure: 2D data points generated from Gaussian distribution with identity covariance matrix and means $(-1, 0)$ and $(1, 0)$.
Source: STOR 390: Introduction to Data Science by Iain Carmichael.

Nearest Centroid-based classification

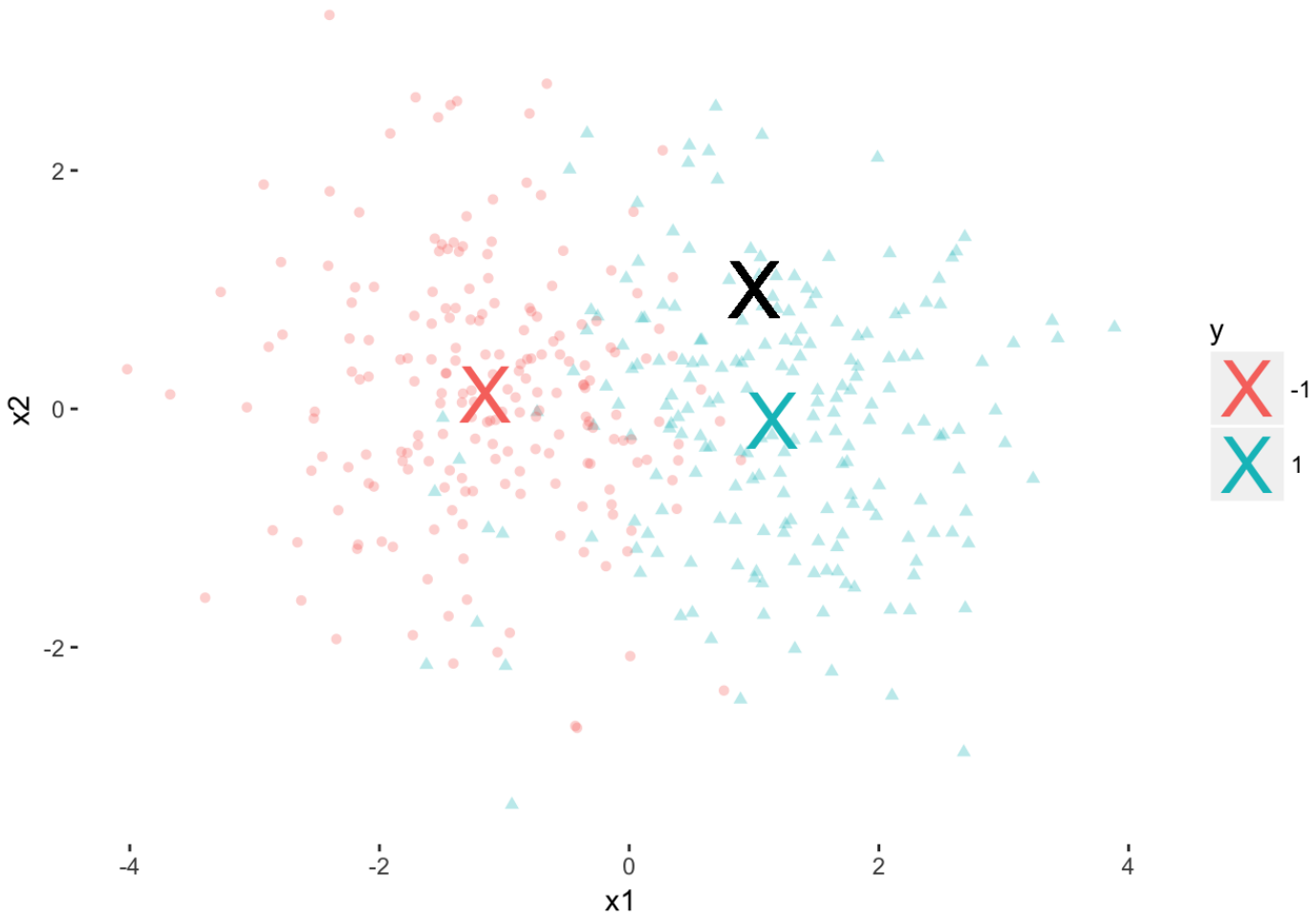


Figure: Generation of centroids of two classes by taking means of the training data points corresponding to the classes.
Source: STOR 390: Introduction to Data Science by Iain Carmichael.

Nearest Centroid-based classification

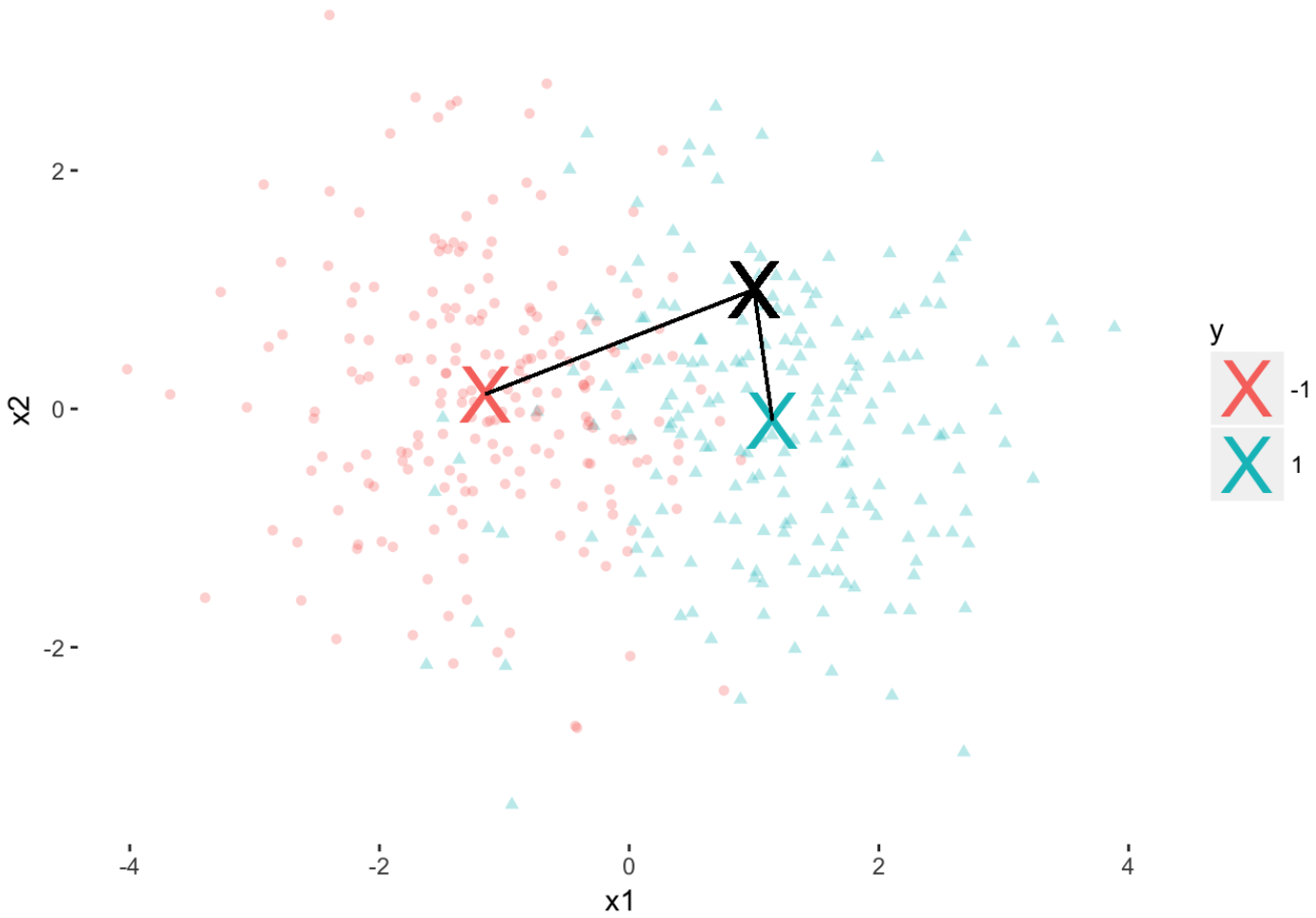


Figure: Find the distances of the test data point with the two means. Assign the test data point (black cross) to the class 1 since the distance of the test data point to the mean of class 1 is smaller.

Source: STOR 390: Introduction to Data Science by Iain Carmichael.

Nearest Centroid-based classification

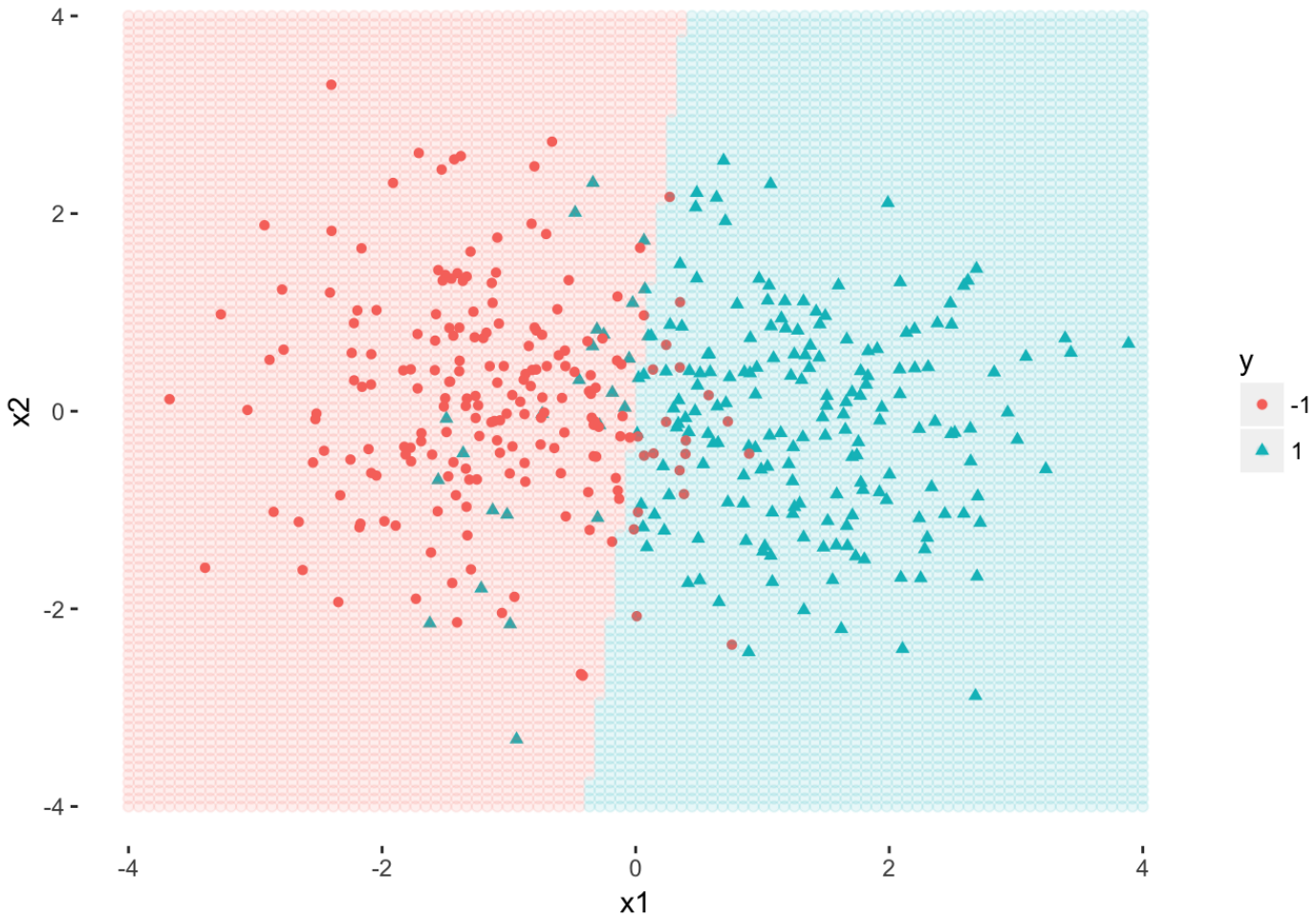


Figure: Nearest Centroid-based classification is an example of Linear classification since the the 2 classes are separated by a line.
Source: STOR 390: Introduction to Data Science by Iain Carmichael.

K-Nearest Neighbour Classification

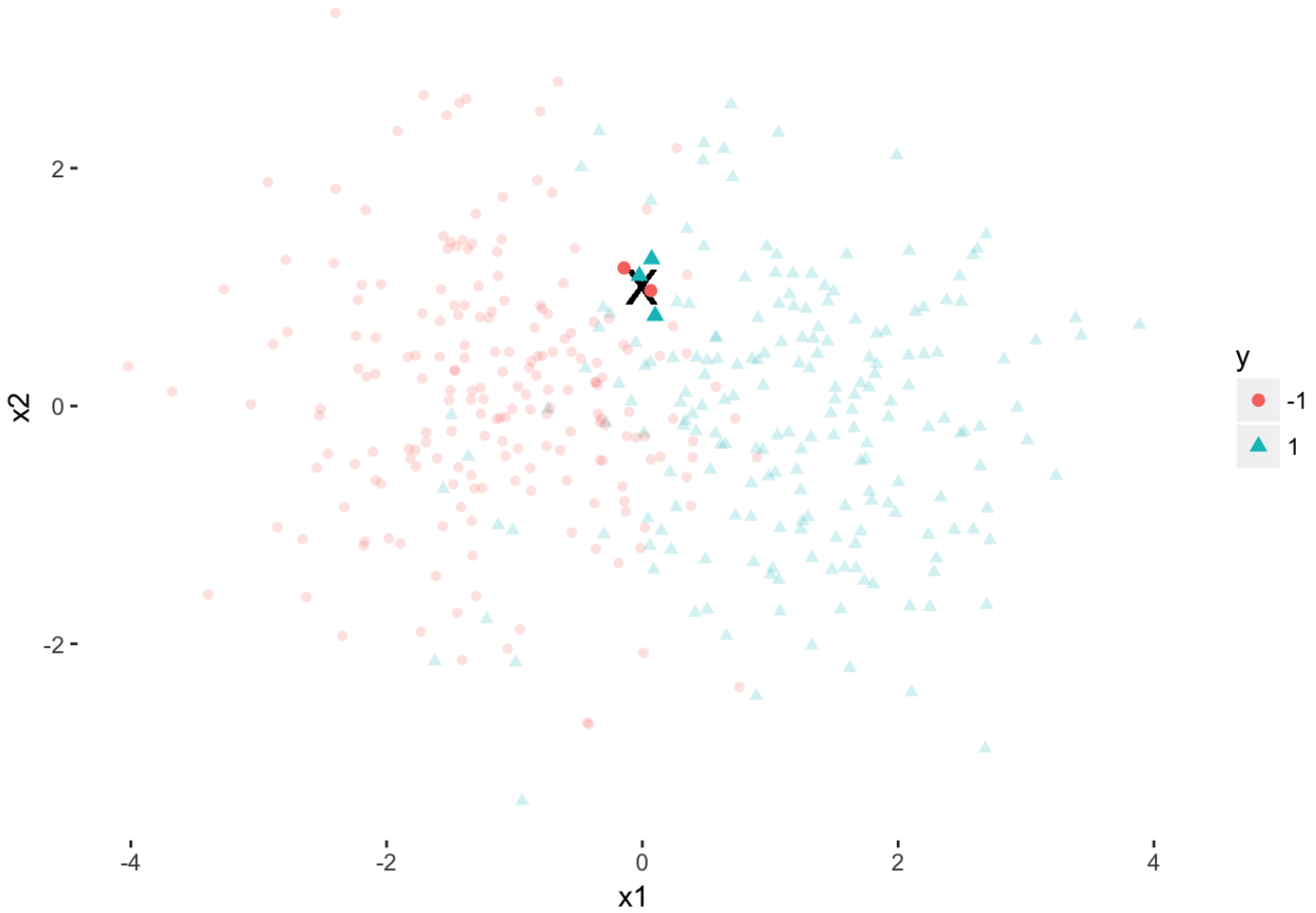


Figure: K-Nearest Neighbour Classification on the previous data set.
Source: STOR 390: Introduction to Data Science by Iain Carmichael.

K-Nearest Neighbour Classification

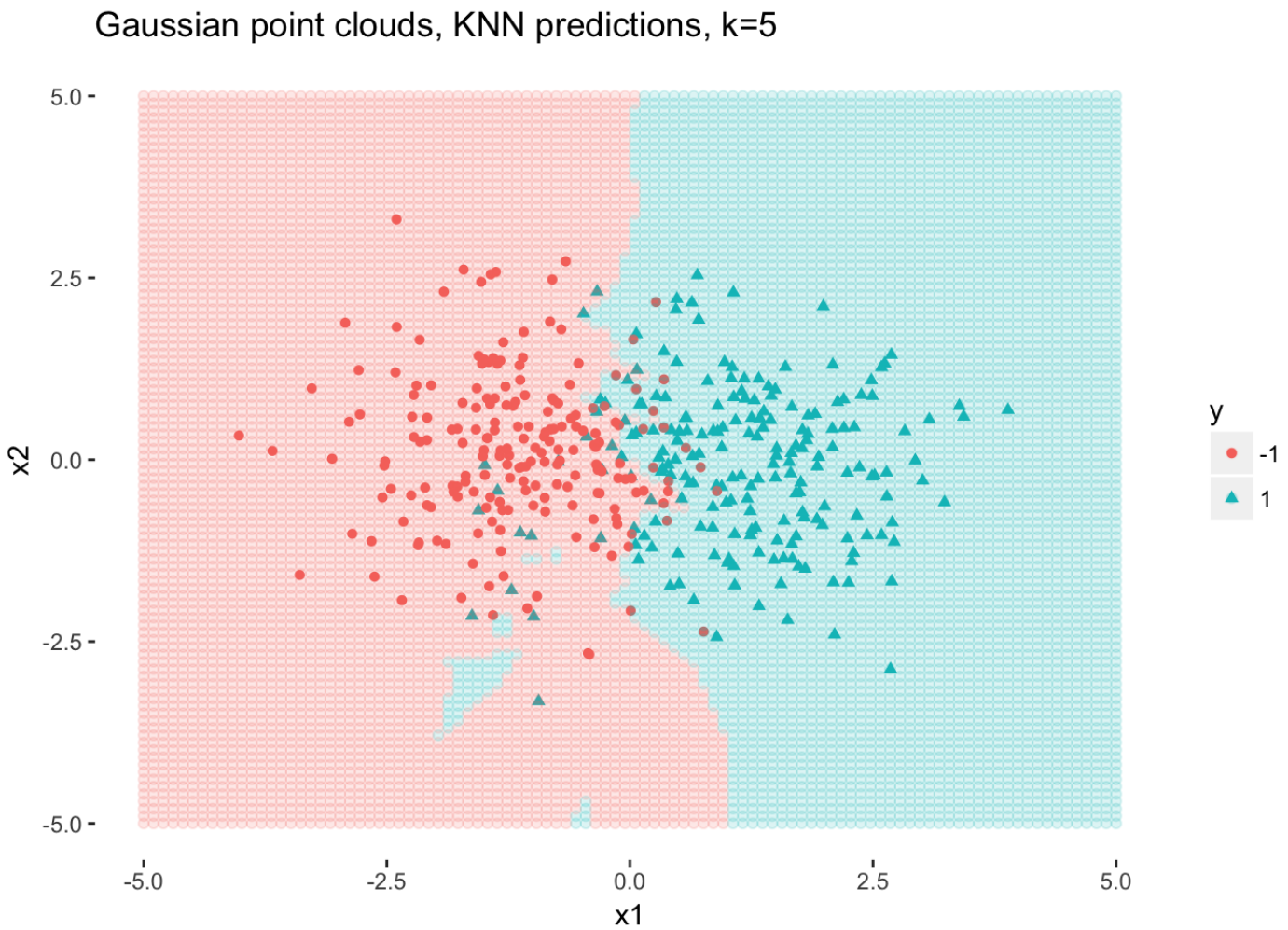


Figure: K-Nearest Neighbour Classification on the previous data set (K=5).
Source: STOR 390: Introduction to Data Science by Iain Carmichael.

Nearest Centroid vs K-Nearest Neighbour

Which of the two classifiers do you think will perform better in the following cases?

- Nonlinear classification
- Label noise (some training data points are mislabelled)
- Presence of outliers in data

Nearest Centroid vs K-Nearest Neighbour

- Nearest Centroid
  1. requires a lot of memory (all training data must be stored at all times)
  2. slow to compute
  3. robust to outliers
- K-Nearest Neighbour
  1. requires little memory
  2. fast
  3. robust to class label noise