

# 1 K-means

## 1.1 What's the algorithm?

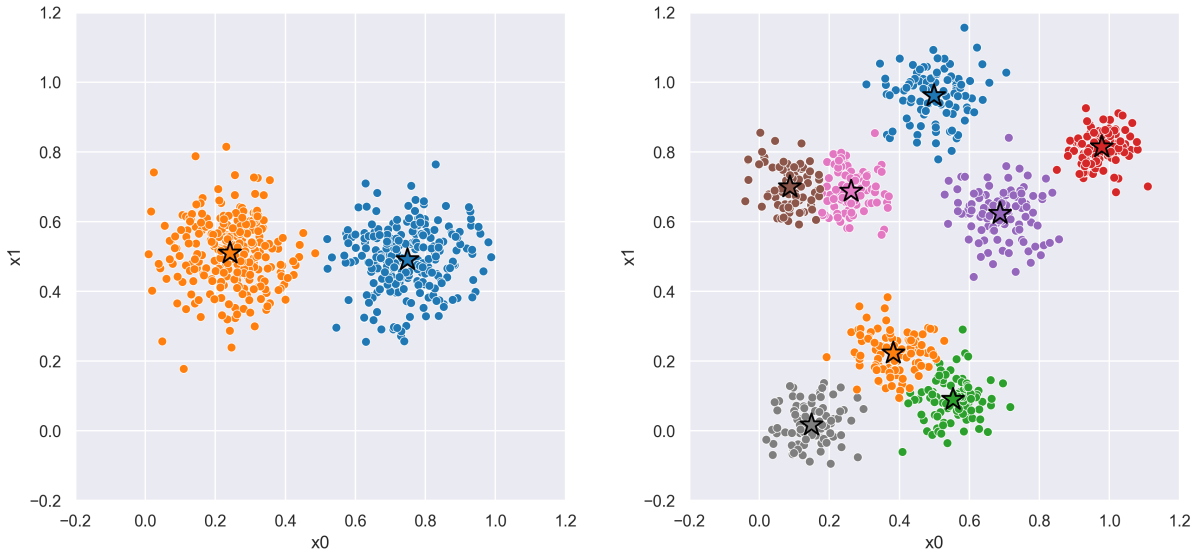
The K-means algorithm is an unsupervised learning algorithm that divides data points into K clusters. It is used to find groups which are not labeled. The algorithm consists of two parts, where the last part is run as many times as we, the user, specifies, or until the algorithm has converged:

1. Initialize **cluster centroids**  $\mu_1, \mu_2, \dots, \mu_k \in \mathbb{R}^d$  randomly.
2. Repeat:
  - (a) For every data point  $i$ , set  $c^{(i)} := \arg \min_j \|x^{(i)} - \mu_j\|^2$ , as the centroid closest to point  $i$ .
  - (b) For each centroid  $j$ , set  $\mu_{j,new} := \frac{\sum_{i=1}^n \mathbb{1}\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^n \mathbb{1}\{c^{(i)}=j\}}$ , as the new position for centroid  $j$ .

## 1.2 Inductive bias

The inductive bias in this problem is that we assume that there are K clusters that we want to find, and that the clusters are of similar size. In the second dataset, the ratios of the  $x_0$ 's and  $x_1$ 's were different than what was given in the first dataset. To fix this I divided the  $x_0$ 's by 100 and the  $x_1$ 's by 10 to get the ratios equal to one.

## 1.3 Some plots



## 1.4 Result/Preprocessing

For the first dataset I didn't have to do any preprocessing, but I matched the result given as an example. In the second dataset, the ratios of the  $x$ 's were a lot different than one, so to fix this I divided the  $x_0$ 's by 100 and the  $x_1$ 's by 10, to get the desired result as seen above, where all 8 clusters are found.

## 2 Logistic regression

### 2.1 What's the algorithm?

Logistic regression is used when we want to predict something that is categorical, i.e. true or false, and 1 or 0. It works by fitting  $\mathbf{u} = (\mathbf{w}, b)$  such that

$$J(\mathbf{u}) = \sum_{i=1}^N (-y_i \log(\sigma_i) - (1 - y_i) \log(1 - \sigma_i))$$

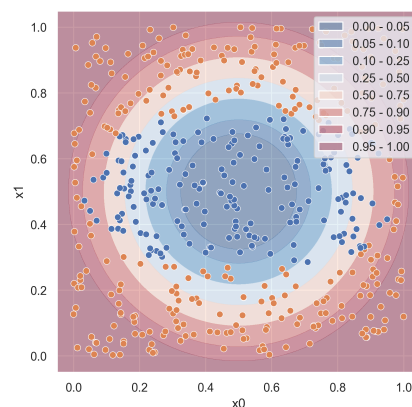
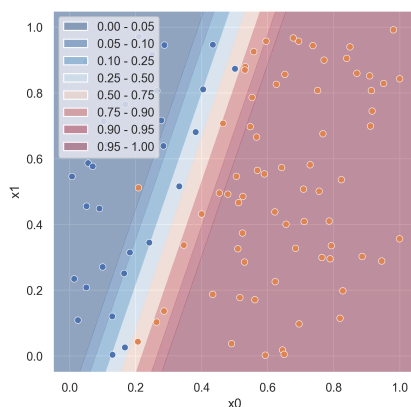
is minimized. Here  $\sigma_i = \sigma(\hat{y}_i)$ . This gets done using gradient descent:

1. Initialize  $\mathbf{u}$  randomly.
2. Repeat:
  - (a) Calculate the gradient  $\nabla = \frac{\partial J}{\partial \mathbf{u}}$
  - (b) Update  $\mathbf{u} = \mathbf{u} - \eta \nabla$ , where  $\eta$  is the gradient step size.
3. until  $\frac{\|\mathbf{u} - \mathbf{u}^{old}\|}{\|\mathbf{u}^{old}\|} < \epsilon$ .

### 2.2 Inductive bias

The inductive bias in this problem is that we assume that the data has two different classes, i.e.  $y = 1$  or  $y = 0$ , and we assume that there is a linear boundary between these two classes. The second dataset had the two classes separated in a circle, so to get around this problem I used the distance from the center to separate the two.

### 2.3 Some plots



### 2.4 Result/Preprocessing

In the first dataset I didn't have to do any preprocessing, but I exceeded the accuracy and cross entropy in the example. I got an accuracy of 0.980 and a cross entropy of 0.102. In the second dataset, there wasn't a linear barrier between the groups, so I used the distance from the middle to make such a barrier. The training data then gave an accuracy of 0.818 and a cross entropy of 0.365, and the testing data gave an accuracy of 0.828 and a cross entropy of 0.333.