TDT4173 - Individual assignment

Andreas B. Berg

17.09.2021

1 K-Means

How K-Means works

K-Means is a relatively simple algorithm. It works by first initializing centroids using a random selection - or, as I will explain later, a specific algorithm - then alternating two steps until convergence:

1. Assign each data point to its closest centroid
2. Compute new centroids based on the geometric mean of its assigned data points

By alternating these two steps, each centroid will shift towards a stable location within its cluster. When it stabilizes, each centroid represents its cluster, and each datapoint is assigned to its closest centroid. New datapoints can then be categorized by adding them to the cluster of its closest centroid.

Machine learning problems K-Means is suited for

K-Means, being a simple unsupervised clustering algorithm, is suited for unsupervised machine learning problems, where the goal is to catch tendencies or patterns in the data, such as during data exploration where the goal is to learn more about the dataset. As the clustering only uses distance between datapoints, the only requirements is that the data has computable distances, as well as a way to calculate the mean of each position.

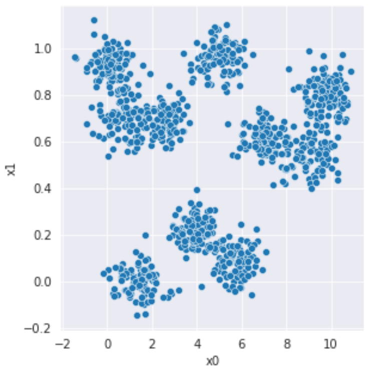
Bias - Assumptions made about the data

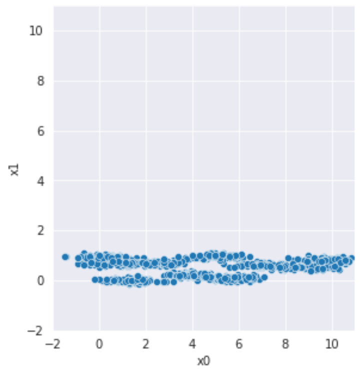
The K-Means algorithm assumes a couple things about the data:

- The clusters are isotropic, meaning they are approximately uniform in each orientation
- The number of clusters are known beforehand
- The clusters are approximately the same size and density

About the second dataset

The second dataset contained 10 non-isotropic clusters. This is especially visible when looking at an isotropic plot of the dataset:





While this represented the biggest challenge with the dataset, it is also worth noting that some clusters were close together, almost clumped up into one larger cluster, while some where further apart. This made the clustering more challenging, as well.

Handling the second dataset

To handle the second dataset, I implemented two things in my algorithm:

Initialization of centroids

While random selection of datapoints to use as centroids worked, it was very unreliable. To improve this, I opted to use the *k-means++*-algorithm to initialize the centroids. *k-means++* works by:

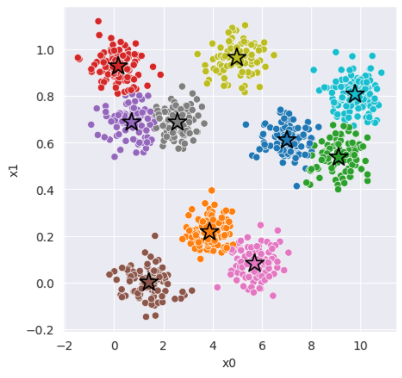
1. Randomly select one initial centroid
2. Repeat until k centroids:
   (a) Compute the distance from each datapoint to its closest (already selected) centroid
   (b) Select a new centroid from all datapoints with probability proportional with the distance two the datapoints' closest centroid

By using *k-means++* to initialize the centroids, I achieved a much more stable clustering.

Preprocessing of data

In order to deal with the non-isotropic dataset, I preprocessed the data by normalizing it along each axis, transforming the data to be in the area given by $x \in [0, 1]$ and $y \in [0, 1]$.

Combining these two methods, I achieved an algorithm that found at least 9/10 clusters each time, and all 10 clusters most of the time.



2 Logistic Regression

How Logistic Regression works

Logistic regression works by assigning each input dimension its own weight (known as the model's parameters), then computing the logistic function of the product.

In mathematical terms, logistic regression finds a prediction $h_\theta(x)$, where x is the input vector, $h_\theta = \frac{1}{1+e^{-\theta^T x}}$ is the logistic regression model and θ is a vector of the parameters (weights) of the model.

To train the parameters of the model, the following method is used:

1. Initialize random parameters
2. Repeat for set number of epochs:
   (a) Predict results for all inputs
   (b) Calculate how much the model missed by
   (c) Update the weighs by $\theta_j = \theta_j + \alpha(actual\ result - predicted\ result) * x_j$, where α is the learning rate of the model.

Machine learning problems Logistic Regression is suited for

Because the logistic function always gives results in the range $(0, 1)$, the logistic regression model is well suited for predicting probabilities. As such, it is useful in situations like predicting illness or risks from a disease, predicting successes when starting new projects, or predicting the likelihood of a stock going up.

The model can also be used for classification (such as is the case in this task) by splitting the results by probability: If the probability of a datapoint belonging to class 1 is less than 0.5, then the datapoint is assigned to class 0, and it the probability is higher than or equal 0.5, the datapoint is assigned to class 1.

Logistic regression is also heavily used in neural networks, where the logistic function is one possible activation function hidden layers can use. While this is not a direct application of Logistic Regression, it indicates that the model can be used as one part of a solution to a multitude of problems.

Bias - Assumptions made about the data

The Logistic Regression algorithm assumes a couple things about the data:

- No significant outliers
- No collinearity
- Linearly separable data. The model gives a straight separation of the data.

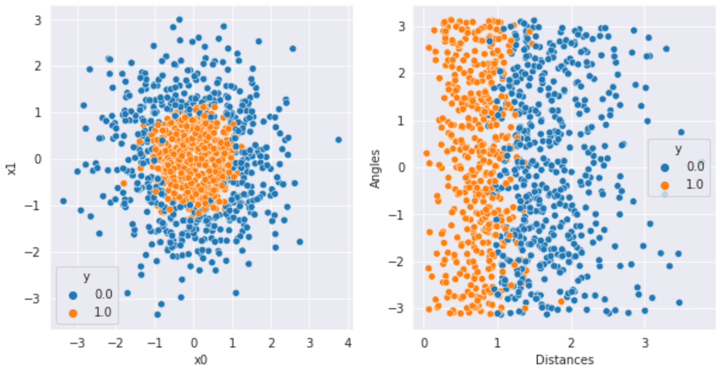
About the second dataset

The first dataset was a straight-forward linearly separable dataset.

The second dataset contained data that was not directly linearly separable. This presented a challenge as it goes directly agains the assumption that the dataset is linearly separable.

Handling the second dataset

By looking at a plot of the second dataset, it became clear that the output class depended - at least partially - on the distance from origo. Based on this, an attept was made to convert the cartesian coordinates of the original dataset to polar coordinates, and use those as input to the model. Below, the left plot is the full original dataset, and the right plot is the same dataset converted to polar coordinates.



Using this preprocessing step, the data became linearly separable, and the Logistic Regression model could do its work. The result was as follows:

Dataset	**Accuracy**	**Cross entropy**
Train	0.906	0.218
Test	0.910	0.195

With an accuracy of over 90 %, this indicates that the model works as expected, and that the choice of pre-processing was correct.