Information Retrieval - Sheet 06

Bernd Menia & Julia Wanker

May 14, 2019

1 Exercise 2b

Look into how the DBSCAN and k-Means algorithms actually work.

1.1 k-Means

One major difference between k-Means and other clustering algorithms like DBSCAN is that you have to specify the amount of clusters before running k-Means. Then the algorithm works like this:

1. Choose how many clusters k we want to create. Choose how many iterations n we want k-Means to perform.
2. Randomly select k points in the data set and make each of them a cluster C.
3. For each other point P calculate the distances from it to every cluster C. Incorporate P to the cluster which is closest.
4. Calculate the mean of each cluster. Repeat step 3 for each point in the data set with the means being the new initial clusters. Repeat step 4 until no more changes are made.
5. For each cluster take every point and sum up its distance to its mean. This sum is called the **Variation** of cluster C.
6. Repeat steps 2 to 5 n-1 times and keep track of the variations in each repetition.
7. Select the clustering which has the lowest variation.

Note that with increasing k and theoretically infinite n the variation would decrease towards 0. Since this is obviously not wanted we have to choose a k which is most fitting for clustering our data. We can run k-Means with different k and plot each run with its minimal variation. Most likely this will result in a plot that mimics a logarithmic plot. By looking at this plot we can find out which k has the sharpest curve and select it as the most fitting value.

1.2 Density Based Spatial Clustering of Applications with Noise (DBSCAN)

Before applying the algorithm we first have to specify the 2 parameters that are needed besides the data set itself:

- The neighborhood N (epsilon in the slides) specifies how far a point Q is maximally allowed to be distant from a point P such that P and Q are considered to be in the same cluster.
- The minimum number of points M that is used to classify points. A **Core Point** is a point that has at least M points in its neighborhood N. A **Border Point** is a point that is not a core point, but has at least one Core Point in its neighborhood. Finally a **Noise Point** is a point that is neither a Core Point, nor a Border Point, i.e. it has no core points in its neighborhood.

Next we look at the algorithm and how it makes use of the given parameters:

1.
2.
3.
4.

2 Exercise 2d

How does PCA work and how can we interpret its results?

Plotting a 1d or 2d graph is quite simple by putting values on 1-axis and 2-axis respectively. Doing the same for a 3d graph is also possible, but it is already harder to do so because of the flattened nature of a screen / book. From that point onwards going above 3 dimensions cannot be plotted accurately anymore. However what if we still want to graphically represent data that has more than 3 variables / dimensions? This is were **P**rincipal **C**omponent **A**nalysis (PCA) comes into play.

PCA works independent on the number of dimension a plot has. For each dimension PCA does the following:

1. Draw a line through the origin and call it PC n (where n is the n-th dimension).
2. For each value in the dataset calculate the squared distance from the origin to the point on the line that is perpendicular to the value. Squaring the distance is important because otherwise negative points (distances) would cancel out positive ones.

3. Rotate the line until the combined sum of all squared distances of the points to the origin is maximal. Said maximum value is called the **Eigenvalue** of PC n and its square root is called **Singular Value** of PC n. This line is now representative for PCA n.

4. Calculate the slope of PCA n which gives a ratio about how the data is spread out on which axis. For example if N = 2 and PCA 1 has a slope of 0.25 then for each 4 units on first dimension (original axis) we go up 1 unit on the second dimension, i.e. the data is mostly spread out on the first dimension.

5. Calculate the length of the vector that represents the line that we just maximized by using the pythagorean theorem. Since PCA is scaled the length of the vector has to be scaled to be of length 1. To do this simply divide each element of the vector by its distance. This new vector is called the **Eigenvector** of PCA n and its values are called **Loading Scores**. Note that the ratio between the dimensions has not changed in this process.

6. Repeat the same process for all dimensions.

7. Divide each Eigenvalue by the amount of samples in the data set to calculate how much variation (information) each PC holds and order the results.

8. Take 2 PCs that most accurately describe the plot and rotate them so that one PC is horizontal. From that point we can directly plot the results on these new x- and y-axis. However it is most certain that information will get lost by downsizing the dimensions to just 2, so it can be that the data is misrepresented.