# Pattern Recognition
# K-means

**Useful matlab functions:**

`sort`, `scatter`, `unique`, `tcdf`, `randsample`, `realmax`, `randperm`, `voronoi`

**Guidelines for lab reports:**

- Always give a (short) explanation of what you are doing.
- Do not forget to include your Matlab programs. Present and discuss the results of your programs, be it a number, a matrix or an image.
- Put large pieces of Matlab code in an appendix.
- One should be able to understand plots independently, be sure to label axes, add a legend for colors, etc.
- Refer to all plots, tables, code blocks, etc. in your report.
- If you print gray-scale make sure the colors used in the plots are distinguishable.

**Assignment 1:** *k-means clustering, quantization error, gap statistic*

1. The file `kmeans1.mat` contains 2D feature vectors. Apply the *k-means* clustering algorithm with Euclidian distance to this data for the values of $k$ given below. As a stop criterion use the condition that none of the data points are re-assigned to a different cluster. After completion of the algorithm for a given value of $k$, make a scatter plot of the data points, taking care that the different clusters can easily be distinguished, e.g. by rendering points that belong to different clusters in different colors or shapes if you print in black and white. Add the final cluster means to the same plot, making sure they are distinguishable from the data points. Make another plot that shows all intermediate positions of the cluster means computed at the different iterations of the algorithm, indicationg which are the starting and ending position of each cluster mean.

   Make the steps described above for `k = 2, 4, 8` means and include the resulting images in your report.

*hint:* Write your algorithm as a function that takes the 2D array of feature vectors and the number $k$ of means as arguments so that you need not change your code to use it for a different number of means or a different dataset. Initialize the $k$ means as $k$ distinct points randomly chosen *from the dataset*. You can make use of the provided `plot_arrow` function to illustrate the movement of the means.

2. Upon completion of the algorithm for a given value of $k$, compute the quantization error $J(k)$ as a function of $k$, i.e. $J(k)$, for $k = 1, 2, 3, \ldots, k_{max}$ where $k_{max}$ is some reasonable number, not larger than the total number of points in the data set. Now, consider a reference function $R(k) = J(1)k^{\frac{-2}{d}}$, where $d = 2$ is the number of dimensions. Note, that for $k = 1$ the functions $J$ and $R$ have equal values, $J(1) = R(1)$. Next, consider the function $D(k) = \frac{R(k)}{J(k)}$.

   *Remark:* The term $k^{\frac{-2}{d}}$ in the function $R(k)$ mimics the behavior of the quantization error for a data set that is uniformly distributed in the domain occupied by the real data set. The coefficient $J(1)$ in front of it in the expression for $R(k)$ makes sure that $J(1) = R(1)$. For those who want to learn more on this topic, here is a reference: R. Tibshirani, G. Walther, T. Hastie. Estimating the number of clusters in a data set via the gap statistic, *Journal of the Royal Statistical Society*, Series B 63 (2000), 411–423.

   a) Plot the function $D(k)$ in a figure and determine the value $k_{opt}$ of $k$ for which $D(k)$ reaches its maximum, $k_{opt} = \arg\max_k D(k)$ – this is the optimal number of clusters to be used. Mark the position of $k_{opt}$ in the figure.

   *hint:* See the lecture slides for the formula of the quantization error. Pay attention to the summation domain and the fact that each data point contributes to the quantization error with (the square of) its distance to *only one* cluster mean, the one which is nearest to that point. Since the quantization error can vary between runs, depending on the initialization of the cluster means, run the k-means algorithm multiple times and compute $J(k)$ as the average quantization error over different runs with the same value of $k$.

   b) Plot the functions $J(k)$ and $R(k)$ simultaneously in one figure using different colors and mark the position of $k_{opt}$ in that figure.

3. The k-means algorithm can get stuck in local minima, and so the quality of its results depends heavily on the initialization of the cluster-means. The k-means++ algorithm includes a specific procedure for initializing the prototypes of the k-means algorithm such that its solution is close to the optimal k-means solution.

   The k-means++ algorithm is as follows:

```
1. Choose one point at random from the set of all data points
   and let this point be the first prototype.
2. For each data point x, compute D(x), the distance between x and
   the nearest prototype. (In the beginning, there is only one
   prototype but later other prototypes are being added, see below.)
3. Choose one new data point at random as a new prototype,
   using a probability distribution where a point x is chosen with
   probability proportional to D(x)^2. Thus, a point x is more likely
   to be selected as a new prototype if this point is far from the
```

```
   already existing protptypes.
4. Repeat Steps 2 and 3 until k prototoypes have been chosen.
5. Now that the initial prototypes have been chosen, proceed
    with the standard k-means clustering algorithm.
```

a) Modify your k-means function to require a third parameter signifying whether k-means++ initialization should be used. Implement k-means++ initialization in your k-means function.

b) The file `checkerboard.mat` contains `3700` 2D feature vectors. For this data and using $k = 100$, compute the mean of the quantization error and its standard deviation averaged over 20 runs of the k-means algorithm, once with and a second time without k-means++ initialization.

c) Does the use of k-means++ initialization for this dataset lead to a statistically significant improvement?
The context of this question is as follows: for each run of the k-means and the k-means++ algorithm you may get different values of the quantization error. In general, we expect that the k-means++ algorithm will yield a smaller quantization error but it may happen that for some run the quantization error obtained with the simple k-means algorithm is smaller than the quantization error achieved with the k-means++ algorithm for the same or for some other run. If the average value of the quantization error obtained with the k-means++ algorithm over the considered 20 runs is smaller than the corresponding averrage value obtianed with the k-means algorithm, does this mean that this will still be the case if we perform another 20 runs with the two algorithms? The certainty with which we can say that k-means++ algorithm outperforms the k-means algorithm depends on the difference of the two average values in relation to the corresponding standard deviations (spreads). It can be computed and expressed as a p-value using an unpaired one-tailed two-sample t-test (e.g. Welch's t-test). Perfom such a test and determine the concerned p-value.

**Assignment 2:** *Batch Neural gas vs k-means*

In this assignment we compare the (standard) k-means and the batch Neural gas algorithms.

1. Complete the implementation of the **batch** Neural gas algorithm in the file `batchNG.m`.

2. The file `checkerboard.mat` contains `3700` 2D feature vectors. Apply (a) the `k-means` and (b) the `batch neural gas` clustering algorithms to this dataset. Use 100 prototypes initialized from the file `clusterCentroids.mat`, which contains 100 distinct data points randomly selected form the original dataset. Plot the data points, the prototypes and the cluster boundaries for `batch neural gas` after 20, 100, 200, and 500 epochs and for `k-means` after termination.

   Note that for a fair comparison we take the same initial prototypes for both algorithms, so you need to modify your k-means function so that it initializes the cluster means from `clusterCentroids.mat`.

3. Comment on the results and the advantages and disadvantages of the two algorithms.