4 Practical: Vector Quantization

Implement the **Winner-Takes-All unsupervised competitive learning (VQ)** as discussed in class and apply it to the data set simplevqdata (provided in *.mat and in *.csv format). It contains 1000 unlabeled two-dim. data points. Use the (squared) Euclidean distance measure. Your code should have roughly this structure:

- Read in the file containing the data, determine the following:
  N (the dimension of input vectors) and *P* (the number of examples)
- Set the parameters
  - K: the number of prototypes,
  - η: the learning rate (step size),
  - t_{max}: maximum number of epochs (sweeps through the data set)
- Initialize the prototype, e.g. by random selection of K data points
- Repeat for epochs $t = 1$ to $t = t_{max}$:
  - In each epoch present all of the data set in randomized order (every example is presented exactly once, the order of examples is different in every epoch). In matlab this can be done conveniently by permuting the examples with the randperm(P) command
  - perform an epoch of training using all of the P examples. At every individual step present a single example to the system, evaluate the distances from all prototypes and update the *winning* prototype.
  - plot the data and prototype positions after each epoch, so you can observe / show how they approach their final positions
  - evaluate the quantization error H_{VQ} after each epoch (not after each individual update step!)
- After t_{max} epochs: plot the learning curve, i.e. H_{VQ} as a function of t

Remarks: You should implement the competitive learning procedure yourself. You may use available routines for the (squared) Euclidean distance etc., but the basic VQ should be coded by you.

Implementation and parameter setting: Implement the VQ training as described above and perform experiments for $K = 2$ and $K = 4$. As an initial guess, use a learning rate $\eta = 0.1$, but you should try different (smaller) values for comparison.
For a given learning rate, determine a reasonable value of t_{max} for which H_{VQ} seems to approach a minimum. Note that on-line VQ might need many epochs for successful training.

Report You should hand in a structured report comprising:

- **(1 point)** A brief **introduction**.
- **(3 points)** A **methods** section in which you explain your approach in a general manner and highlight specifics of your implementation or code if needed. Also motivate your choice of t_{max} and η. Code and implementation itself will also be taken into account in the grading of this section.
- **(2 points)** The **learning curves** (H_{VQ} vs. t) for three different values of η for $K = 2$ and $K = 4$, with prototypes initialized in randomly selected data points.
- **(2points)** Plots displaying the **trajectories of prototypes** in the course of learning for $K = 4$, showing the trace of the prototypes from initial to final position together with the data points. Choose a learning rate that gives non-trivial, meaningful results. Perform the training from at least two different intializations (one of them in randomly selected data points, one of them *"stupid"*).
- **(2points)** A **discussion** of the results, in in particular with respect to the role of the learning rate, addressing the following questions:
  - How does the final value of the cost function change with η?
  - What happens if η is too large or too small?

Bonus suggestions

Select one of the following to obtain up to **1 point**. If you have an alternative idea of your own, discuss it first with the teaching assistants.

- Compare online VQ and K-means algorithm in terms of their convergence. You may use a buil-in or publicly available K-means implementation in the bonus, but it is also easy and more instructive to implement it yourself.
- Implement and run a 'rank-based' version for $K = 4$ or larger that updates also the second, third, ... winners.
- Perform experiments for $K = 1, 2, 3, 4, 5,$ and plot the final value of H_{VQ} as a function of K. Discuss the result in terms of the elbow method.