Introduction to Intelligent Systems
Unsupervised Learning: Vector Quantization

Implement **Winner-Takes-All unsupervised competitive learning** (VQ) as discussed in class and apply it to the data sets w6_1x.mat, w6_1y.mat, and w6_1z.mat.
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, *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 prototypes by random selection of *K* data points
- Repeat for epochs $t=1$ to $t=t_{max}$:
  - shuffle the data set by permuting the order of examples randomly (a useful command: randperm(P))
  - perform one epoch of training ($i = 1, ...P$). At every individual step present a single example to the system, evaluate the distances from all prototypes and update the *winner*
  - plot the data and prototype positions after each epoch, so you can observe 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

Perform experiments for $K = 2$ and $K = 4$. As an initial guess, use a learning rate $\eta = 0.1$, but you should try different values for comparison.
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.

Your report should contain at least:

– The learning curves for three different values of η for $K = 2$ and $K = 4$. You can select one of the data sets (your choice)
– A short discussion of your results, 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?

Remarks:
Do not use built-in matlab learning algorithms (e.g. from toolboxes) or code from the web. You should implement/code the competitive learning procedure yourself.
Of course you may use built-in, convenient matlab functions as "pdist" for the efficient computation of distances etc.
If your code uses special functions or "tricks", present and explain the corresponding lines of code also in the report.

You should observe the trajectories of prototypes in the course of learning, either 'live' on screen (as an animation) or as a plot showing the trace of the prototypes from initial to final position together with the data points. This kind of plot is not required in the report, but of course you may include it as bonus material.

Further possible **bonus** extensions (suggestions):

- Consider unfavorable 'stupd' initialization far away from the data
- Compare online VQ and K-means algorithm in terms of their convergence etc. You may use the built in K-means in the bonus, but it is also easy and more instructive to implement it yourself.
- Implement a 'rank-based' version that updates also the second, third, ... winners