Introduction to Intelligent Systems
Vector Quantization

Unsupervised Learning - VQ

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
  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 (useful command: `randperm(P)`)
  - perform one epoch of training ($i = 1, ...P$), present single examples to the system, evaluate the distances from all prototypes, and update the *winner*
  - plot the data and prototype positions after each epoch, observe how they approach their final positions
  - evaluate the *quantization error* H_{VQ} after each epoch
- 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 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 more *epochs* than K-means needed iterations for successful training.

Produce several learning curves for different choices of η. Observe how the behavior depends on the learning rate. How does the final value of the cost function change with η? What happens if η is too large (too small)?

You should hand in (at least):

- Your MatLab code
- The learning curve for at least three different values of η for $K = 2$ and $K = 4$.
  for one of the data sets (your choice)
- A short discussion of your results, in particular with respect to the role of the learning rate