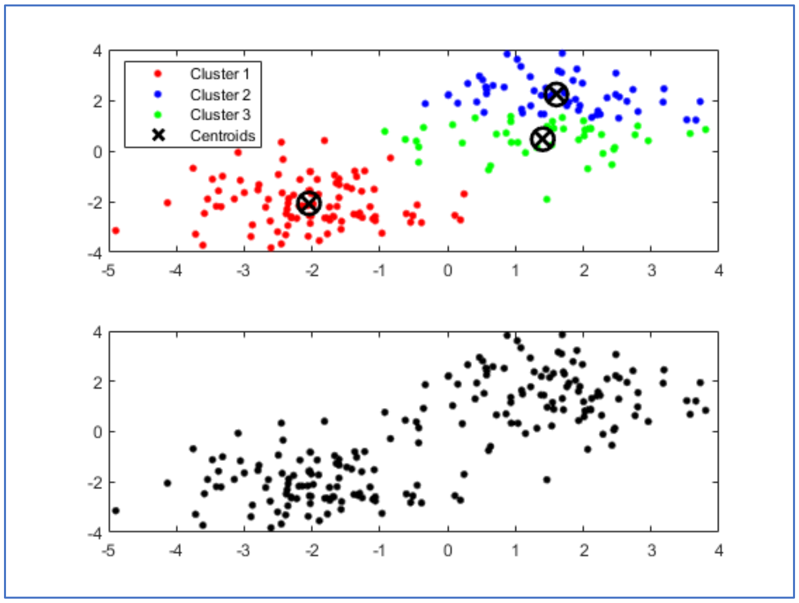
Assignment 6: Prof. Sjöström

Joon Hwan Hong; no collaborations to declare

Note: All figures are generated by running the *MASTER.m* script. An arbitrary RNG seed of 503 was chosen for the assignment for testing and reproducibility

Part 1) k-Means Clustering

A) Make two distributions but cluster with k=3

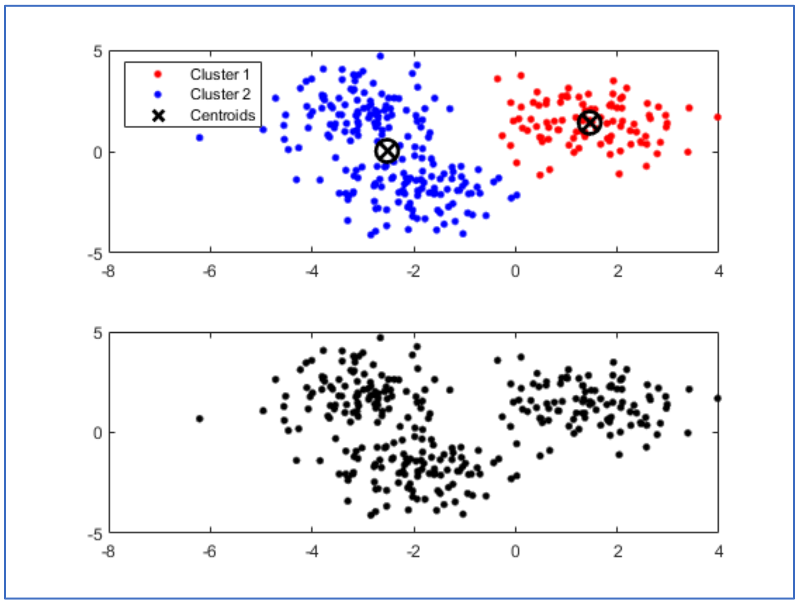


The kMeans script was converted into a function with arguments of: k_value, an integer and two_dist, a Boolean for two distribution or not.

Two clusters were made by not including the last randomly generated distributions in the X matrix.

Two distributions but three clustering results in splitting one of the clusters into halves in cluster 2 and 3. (results in false clusters)

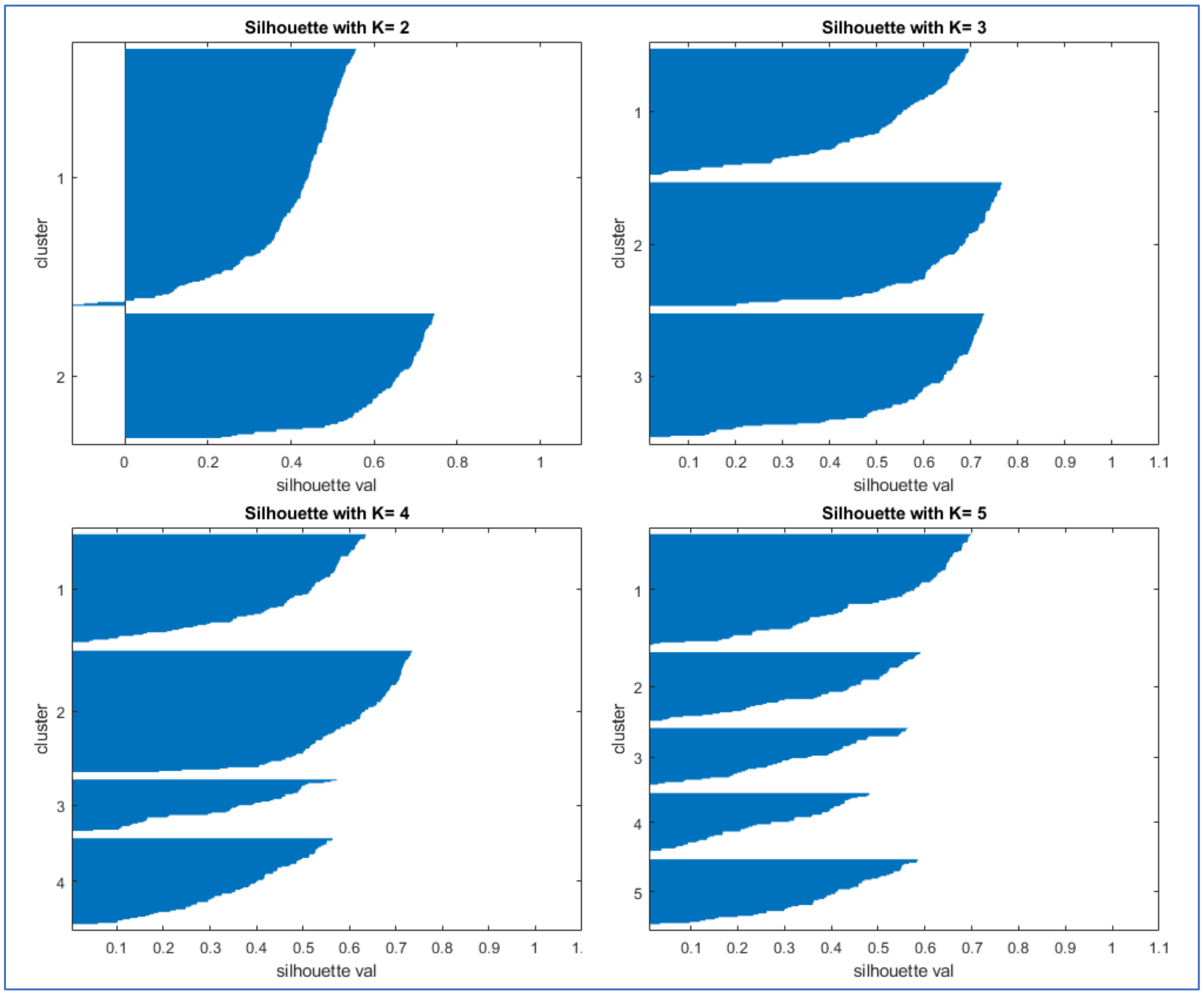
B) Make three distributions but cluster with k=2



In the opposite case, with two clusters when there are three distributions results in "merging" by the function. In rng seed 503, cluster 2 includes two distributions.

This behavior is general when K is set too low, resulting in less clusters generated than the distributions available.

C) Make three distributions, let your code determine k



The silhouette function provided different distance types. The Euclidean distance was chosen as the kMeans function utilised Euclidean distances.

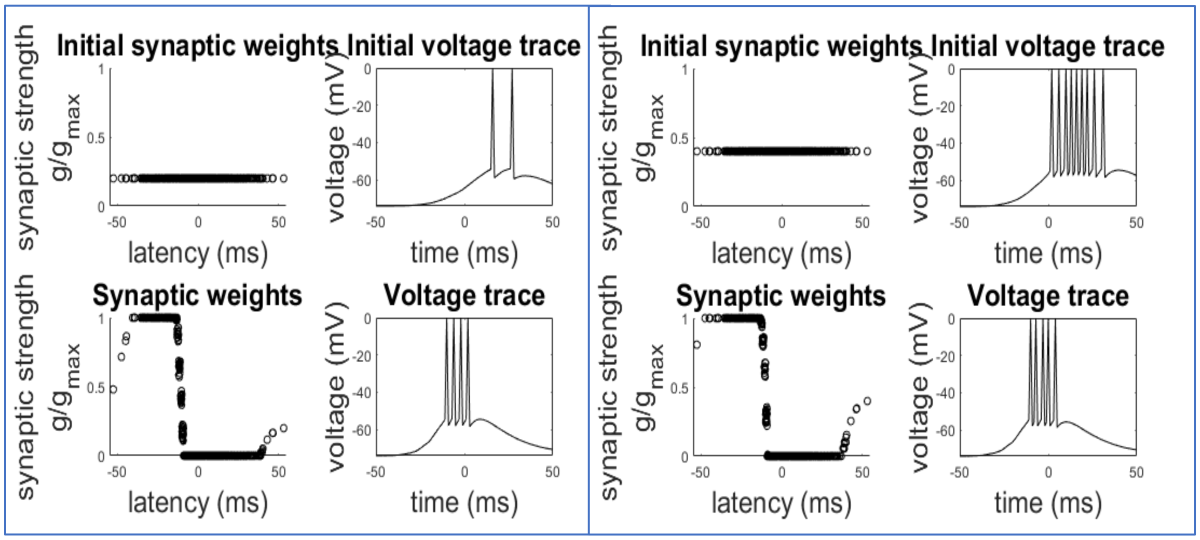
The silhouette plot displays how distant each point in a given cluster is to points in neighbouring clusters. In other words: -1 suggests the point is wrongly classified; 0 suggests that there is no distinction with neighbouring cluster(s); 1 suggests very distant from neighbouring cluster(s).

Obviously, k=2 contains negative values, while the others do not, indicating “potential” misclassifications. K=4 and k=5 contain multiple clusters with silhouette values that are “low”, ranging from 0.4 to 0.6, while in comparison, the k=3 clusters have consistently higher silhouette values (all > 0.7), indicating that they are more distinct from neighbouring clusters.

This result indicates that k=3 is the ideal k to use for the three randomly generated distributions.

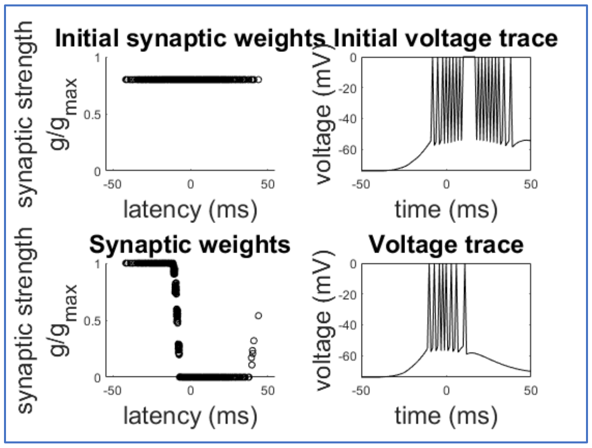
Part 2) STDP

A) STDP reduces latencies and sharpens responses



On *simSTDPlatencies.m*, the g value was changed from 0.003 to 0.006. The figures above are for g values of 0.003 and 0.006 respectively. After doubling the initial conductance, an obvious difference is that more action potentials are fired (visible in the Initial voltage trace). In addition, the first synapse response occurs quicker when conductance is doubled & duration increased.

B) STDP provides a degree of stability



The starting value multiplication factor was incrementally increased by 5 until postsynaptic neuron saturation was detected (seen by the "depolarization block" in the middle of Initial voltage trace). The factor value in this figure is 120.

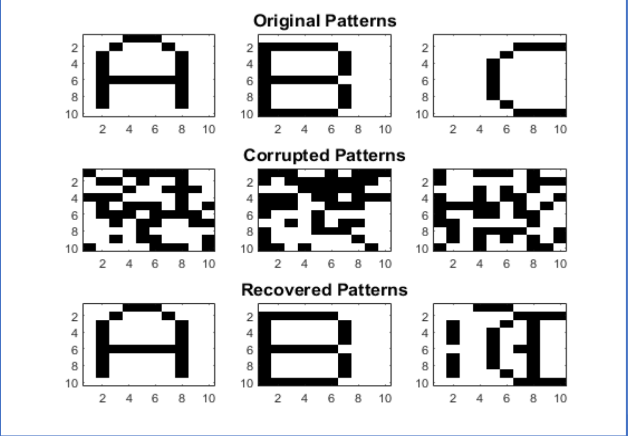
It appears that STDP ensures a stable distribution in synaptic weights. The postsynaptic neuron is still sensitive on presynaptic action potentials.

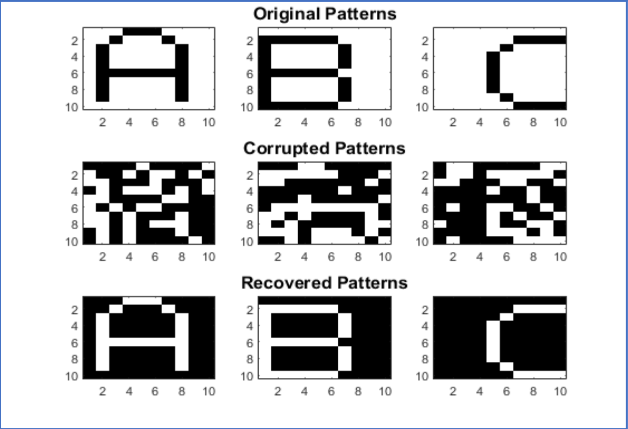
C) STDP can pick up on correlations in inputs

As a general definition, unsupervised learning includes clustering or data classification. The STDP separates synaptic inputs into strong and weak clusters and shape their conductance values (classification). This results in different classification of synapses that are correlated in firing & those succeed in starting postsynaptic action potential from the synapses that failed in contrast. Thus, STDP classifies without a predefined labeling; it is unsupervised learning.

Part 3) The Hopfield Network

A) Experiment with noise levels to find local minima and ghost states





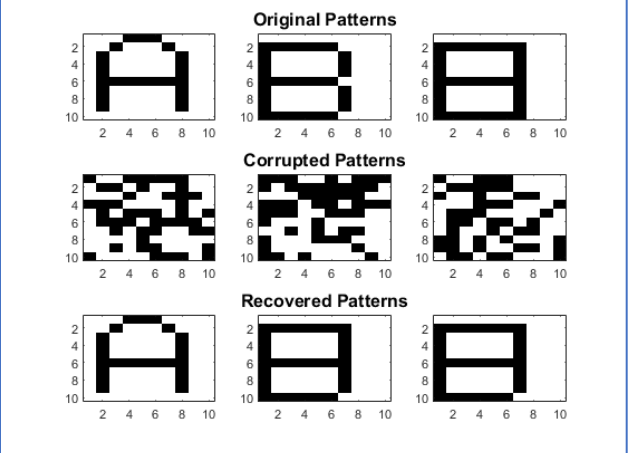
The following figures were generated with noise level of 42 and 69 respectively.

Overall finding: noise levels < 42 are able to recover the patterns; noise levels from 42-68 are unable to correctly recover the patterns; noise levels > 68 result in ghost states (the inverse of the original patterns).

The ghost states with high noise level occurs probably because the noise eventually saturates the 10x10 field too much, making it "more similar" to the ghost states. It changes too many pixels into the inverse, favoring the ghost state.

When it incorrectly recovered the pattern in the upper figure, this indicates that the local minimum reached resulted in the image (recovered pattern) and stabilized there.

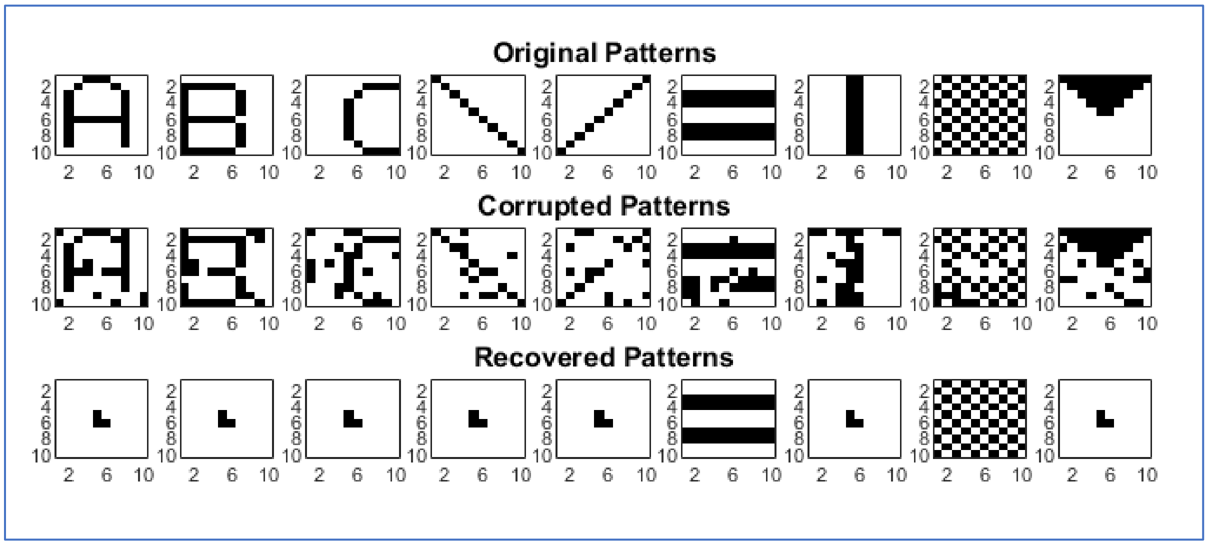
B) Alter one memory



Changed the C into an 8 pattern by coping the B battern and replacing the edge 0s with 255s (mem_ABC_mod.txt).

This results in 2 very similar memories. The converged attractor state for both B and 8 results in a "merged" recovered pattern. This indicates that the recovered state is more stable than either patterns when recovering from the corrupted patterns for that instance.

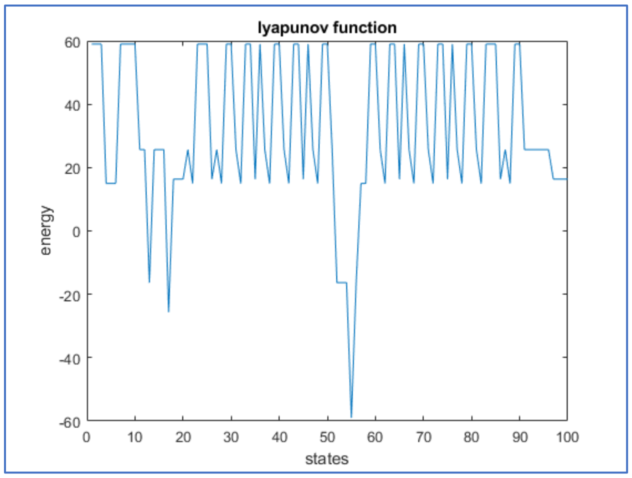
C) The information storage limit of a Hopfield network



The memories are available in mem_ABC_mod2.txt. The figure above indicates that besides two patterns, all other resulted into the same minimum state. Perhaps the noise level was too little for the two horizontal bars and checkered pattern to be disturbed away from its local minima, but the fact that the other seven patterns resulted in the same recovered pattern after their corruption imply that there exists a “global” minimum which all of these patterns gravitate towards.

The theoretical storage limit of the network is given in the lecture, where p is the maximum number of stored patterns and N is the number of nodes: $p = \frac{N}{4logN}$. This assumes that the patterns are well distinct from each other. In this instance, since there are 100 neurons, this results in 12.5 patterns stored → 12 patterns stored using 100 neurons.

D) Calculate numerically the network energy as it converges



The equation was given in the lecture:

$$E(\bar{s}) = -\frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{N} w_{ji} s_j s_i$$

Which is equivalent to the weight matrix (100x100) multiplying with a vector of all states (1x100). The ½ was just considered as a scalar factor which was irrelevant for this question in determining which state (x coordinate). As the hopfield_net function conveniently kept track of the weight matrix and state vector, they were used.

Thus, given 100 neurons, noise level of 10 like the base example, and the original memory text, the global minimum which is converged to was determined to be at x=55. The Hopfield network can be used as a classifier to cluster data into predefined categories/patterns as it measures the similarity between patterns by converging similar memories towards the same attractor state, clustering into groups.