Neural Computation (VS 265), Problem Set 3 - Unsupervised learning

Due date: October 10, 3:30pm

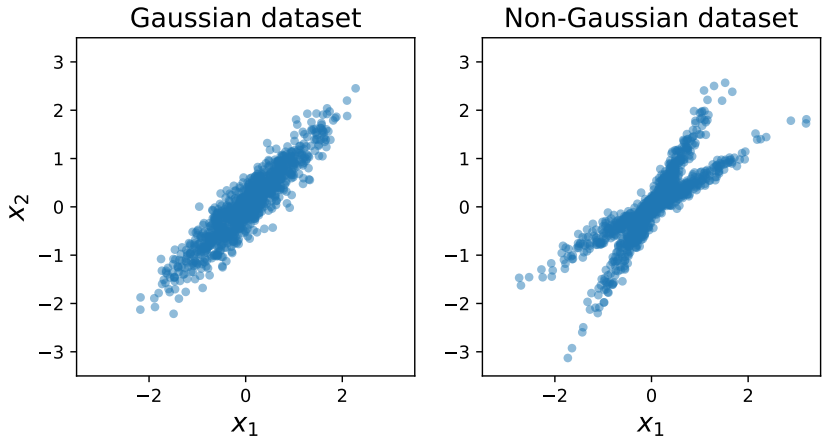
Fall 2024

General guidelines:

- We are grading problem sets anonymously. **Include your student ID in the submission, but do not include your name.**
- You may work in small groups of 2-3. Note that you are responsible for writing up and submitting your submission individually.
- You are expected to attach any code you used for this assignment but will be evaluated primarily on the writeup.
- A link to download the problem set datasets can be found on the course website.

Part 1: PCA and linear Hebbian learning

As discussed in class, a linear neuron undergoing Hebbian learning via Oja's rule will learn to represent the first principal component of the data. Sanger's rule extends this to multiple units - and hence multiple principal components - by serially subtracting out the previously learned components from the input. Here we will explore the consequences of these learning rules for capturing the structure contained in different types of data.



i. **Gaussian dataset.** Train a network consisting of two linear neurons, using Sanger's rule, on the simple 2D Gaussian dataset. Visualize the solution learned by the network by superimposing the weight vectors on a scatterplot of the data points. Make an animation showing the evolution of the weight vectors starting from random initial conditions (or alternatively, plot the solution at different stages of learning - initial condition, partway through learning, and fully converged).

ii. **Non-Gaussian dataset.** Now train the network on the non-Gaussian dataset and visualize the learned solution similar to above. Visualize how well this network captures the structure of the data by synthesizing a dataset using the learned vectors and the corresponding variances on each unit. How do the two distributions compare?

iii. **Faces dataset.** Finally, train a network of five units on the dataset of face images and show the learned weight vectors as a set of five images.

   a) Examine how well the network's 5-D representation captures these images by attempting to reconstruct any one of the images from these five components.
   b) Visualize the structure learned by the network by synthesizing images from the learned weight vectors and the corresponding variances of each unit.

Part 2: Winner-take-all learning

One way of learning non-Gaussian structure in data – for example, when there are distinct clusters corresponding to different objects – is via a simple competitive learning rule such as winner-take-all (WTA) learning.

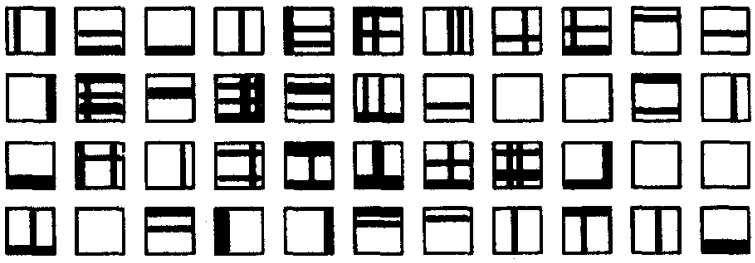
i. Train a WTA network (as discussed in class) on the non-Gaussian 2D dataset as in ii. above, and visualize the learned solution superimposed on a scatter plot of the data. Show the learned solution for networks containing different numbers of neurons - e.g., 3, 4 and 5 neurons.

ii. Next train a WTA network with 5 neurons on the faces dataset as in iii. above and visualize the learned weight vectors as a set of images. How does the solution differ and why?

Part 3: Foldiak's sparse coding model

WTA learning corresponds to a simple form of clustering called 'k-means'. It is useful when the data contain well-defined clusters, and you know how to set k, but otherwise it is a very lossy representation as it forces any given data point to be represented by a single weight vector. One possible improvement would be to allow a data point to be represented as a combination of a small number of weight vectors, rather than just one. This is the idea behind sparse coding.

In 1990 Peter Foldiak proposed a simple neural network model for sparse coding that combines Hebbian learning, thresholding and lateral inhibition to learn a sparse encoding of data. As in the paper we will utilize a toy dataset consisting of 8×8 pixel images containing a small number of randomly placed vertical and horizontal bars:



Clearly it would be impossible to capture the structure of this dataset with WTA learning since there are many different ways the bars could be combined to generate an image that won't coalesce into a small number of clusters. Yet, the structure is still very simple in that any given image is composed of a small number of bars.

i. Implement Foldiak's network dynamics and learning rules for feedforward and lateral connections and train the network on the bars dataset. It makes the most sense here to use a network of 16 neurons. Plot the learned feedforward weights, thresholds, and lateral connection weights to see how they change throughout learning. It is also helpful to show alongside this the average activation for each unit and their correlations to see how the constraints are satisfied.

ii. **(Optional)** How would linear Hebbian learning (i.e., PCA) handle this dataset? Try training your network from part 1 on the bars data and examine the solution and its ability to properly represent the data.