Plasticity in the Brain and Learning

Computational Neuroscience by University of Washington

Hebbian learning

Suppose we have a linear feedforward network with two input nodes and one output node. Let's say that we are learning our weight vector $\mathbf{w}$ and that we are using the Hebb rule.
Suppose the input correlation matrix Q is:
Q=[0.2 0.1; 0.1 0.15]
If we allow learning to go on for a long period of time, which of these could be a final weight vector $\mathbf{w}$?

Oja's Hebb rule for a single neuron

We will implement a neuron that will learn from two dimensional data that is given in the following Python pickle files containing 100 (x,y) data points:

Python 2.7:

c10p1.pickle

Python 3.4:

c10p1.pickle

Part A

Assume our neuron receives as input the two dimensional data provided in c10p1.pickle, but with the mean of the data subtracted from each data point (the mean of all x values should be subtracted from every x value and the mean of all y values should be subtracted from every y value). You should perform this zero-mean centering step and then display the points again to verify that the data cloud is now centered around $(0,0)$.
Implement the update rule derived in the previous question in Matlab or Octave. Let η=1, α=1, and Δt=0.01. Start with a random vector as $\mathbf{w}_0$. In each update iteration, feed in a data point $\mathbf{u}=(x,y)$ from c10p1. If you've reached the last data point in c10p1, go back to the first one and repeat.
Typically, you would keep updating $\mathbf{w}$ until the change in $\mathbf{w}$, given by norm(w(t+1) - w(t)), is negligibile (i.e., below an arbitrary small positive threshold), indicating that $\mathbf{w}$ has converged. However, since you are implementing this as an online learning algorithm, you may prematurely detect convergence using this method. Instead, you may just run the algorithm for 100,000 iterations.
Run your code multiple times. Observe the behavior of $\mathbf{w}$. Why does this happen?

Hint: Consider the eigenvectors of the correlation matrix of the mean-centered data. (The correlation matrix of a data matrix $\mathbf{X}$, where rows indicate separate samples, is $\mathbf{X}^T\mathbf{X}/N$, where N is the number of samples. You can calculate its eigenvalues using eig().) If the data is mean-centered, the correlation matrix will be the same as the covariance matrix.

Part B

What happens when the data is not zero-mean centered before the learning process?

In order to more fully explore the behavior of the Oja's rule when the data isn't mean centered, you should adjust the mean of the data a few times and observe the behavior of the learning rule. You can adjust the mean of the data by adding a constant to every x component of the data and a different constant to every y component of the data.

Part C

What happens when the pure Hebb rule is used instead of Oja's rule? You can explore what happens by removing the subtractive term $-\alpha v^2\mathbf{w}$ in your code and running the code.