Neuronal Dynamics

Professor Wulfram Gerstner
Laboratory of Computational Neuroscience

Numerical simulation of Oja's hebbian learning rule

Exercise 1

Download `oja.py` from book's webpage. Create a script file `answers.py` in the same directory, and give it the following header:

```
from numpy import *
from scipy import *
from oja import *
```

You will type the code for your answers right below. When you want to execute your code, open ipython (pylab) in a terminal and type:

```
>> import answers (or reload answers if already imported)
```

The `oja` module contains two documented functions:

- `make_cloud(...)` which generates a 2D elliptic gaussian cloud of datapoints
- `learn(...)` which runs Oja's learning rule on the data and returns the time course of the weight vector

In your answers you will need to plot data. Plotting functions are already loaded with pylab, and you will find example-based tutorials at `http://matplotlib.sourceforge.net/gallery.html`

1.1 Run the learning rule on a *circular* data cloud (`ratio=1`). Plot the time course of both components of the weight vector. Do it many times. What happens?

1.2 Repeat the previous question with an *elongated* elliptic data cloud (e.g. `ratio=0.3`). Again, do it several times. What is the difference with a circular data cloud, in terms of learning curve?

1.3 Try to change the orientation of the ellipsoid (try several different angles). What does Oja's rule do? Hint: plot the learned weight vector in 2D space, and relate its orientation to that of the ellipsoid.

1.4 The above work assumes that the input activities can be negative (indeed the input were statistically centered). In real neurons, if we think of the "activity" as the firing rate, this cannot be less than zero. Try again the previous question, applying the learning rule on `5 + make_cloud(...)`, which centers the data around `(5,5)`. Draw your conclusions. Can you think of a modification of the rule?