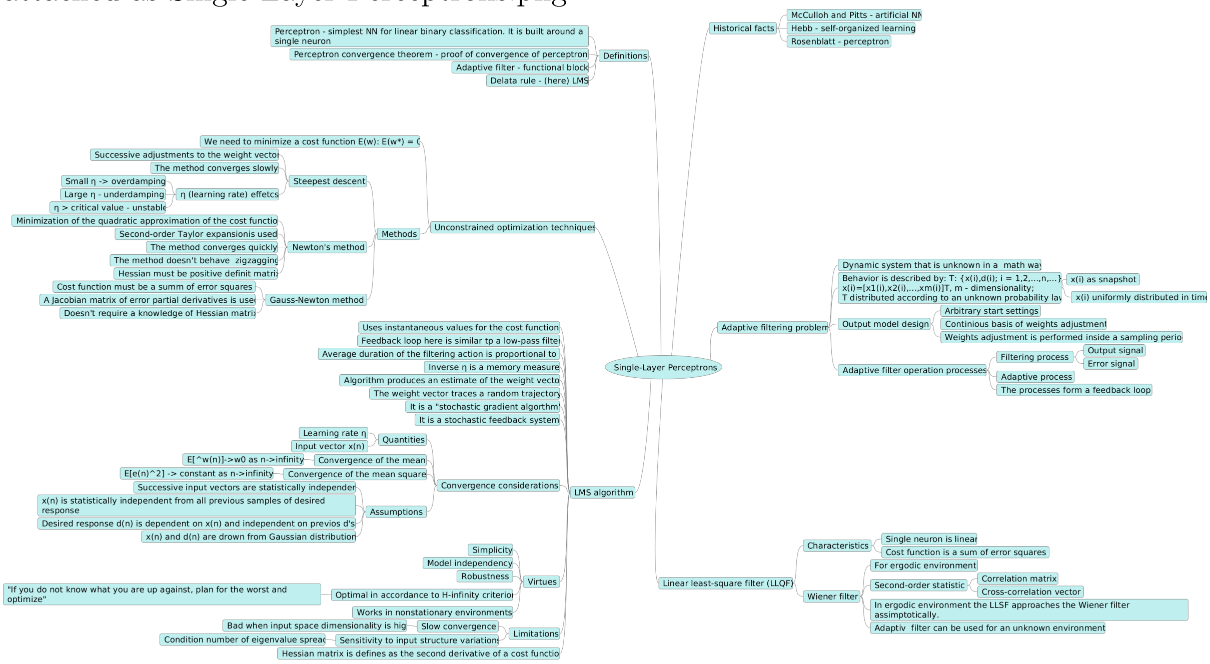
Neural Networks
- Homework 5 -

Petr Lukin, Ivan Vishniakou, Evgeniya Ovchinnikova

Lecture date: 31 October 2016

1 Mind map

Figure 1: Mind map. Chapter 3 (first part) from Haykins book. A zoomed version is attached as Single-Layer Perceptrons.png



2 Exercises

Exercise 3.1

Explore the method of steepest descent involving a single weight w by considering the following cost function:

$\varepsilon(w) = \frac{1}{2}\sigma^2 - r_{xd}w + \frac{1}{2}r_x w^2,$

where σ^2, r_{xd} and r_x are constants.

Solution:

The steepest descent algorithm is describes as following:

$w(n+1) = w(n) - \eta g(n),$
$\Delta w = -\eta g(n),$

where η is a learning rate that is a positive constant and $g(n)$ is a gradient vector evaluated in w(n) point:

$g(n) = \nabla \varepsilon(w) = [\frac{\partial \varepsilon}{\partial w_1}, \frac{\partial \varepsilon}{\partial w_2}, ..., \frac{\partial \varepsilon}{\partial w_m}]^T,$

where m is dimensionality of the input space.

$\varepsilon(w(n+1)) \simeq \varepsilon(w(n)) - \eta ||g(n)||^2$

Here:

$g(n) = -r_{xd} + r_x w,$

so:

$\Delta w = \eta(r_{xd} - r_x w),$

$w(n+1) = w(n) + \eta(r_{xd} - r_x w),$

$\varepsilon(w(n+1)) \simeq \varepsilon(w(n)) - \eta ||g(n)||^2 = \frac{1}{2}\sigma^2 - r_{xd}w + \frac{1}{2}r_x w^2 - \eta ||(r_x w - r_{xd})||^2$

To plot it we need to choose some values for the constants: $\sigma^2 = 2, r_{xd} = 3$ and $r_x = 4$.

To plot $\varepsilon(w)$ and paths with different η's (0.1, 0.01, 0.45) we used the following python code:

```
import numpy as np
import matplotlib.pyplot as plt
from numpy import linalg as LA

%matplotlib inline

#constants:

sigma = 2
r_xd = 3
r_x = 4
eta = 0.1
eta_small = 0.01
eta_large = 0.45

w = np.arange(-10., 10, 0.2)
plt.ylabel('E')
plt.xlabel('weights')
```

```
plt.plot(w, 0.5*sigma**2 - r_xd*w + 0.5*r_x*w**2, 'go')
plt.show()

def steepest_descent(eta):
    err = 100
    #estimate from plot above:
    w_init = 2
    w = w_init
    weights = np.array([])
    Es = np.array([])
    iterations = 0

    while(abs(err) > 0.0001):
        iterations += 1
        E = 0.5*sigma**2 - r_xd*w + 0.5*r_x*w**2
        g = r_x*w - r_xd
        E_upd = E - eta * (LA.norm(g))**2
        w_upd = w - eta * g
        weights = np.append(weights, w)
        Es = np.append(Es,E)
        err = w_upd - w
        w = w_upd
    print "minimum weight"
    print w
    print "number of iterations"
    print iterations

    w = np.arange(-1., 2.5, 0.05)
    plt.ylabel('E')
    plt.xlabel('weights')
    plt.plot(w, 0.5*sigma**2 - r_xd*w + 0.5*r_x*w**2, 'go',
        weights, Es, 'bd', weights, Es, 'k')
    plt.show()

steepest_descent(eta)
steepest_descent(eta_small)
steepest_descent(eta_large)
```

The $\varepsilon(w)$ is shown in Fig. 2 and the results are depicted in Fig.3 - 5.
As we can see, the smoothest trajectory is with the smallest η - the transient response is overdamped, it converges quite slow. Large η shows zigzag behavior and if we continue to increase the η, the oscillations and a number of iterations will increase (see Fig.6). Finally, if we try to plot the path with $\eta \geqslant 0.5$ the algorithms diverges.
So, small ηs give a better result, but they might require too many iterations and one should be careful with large ηs because they can start oscillate.

Figure 2: $\varepsilon(w)$

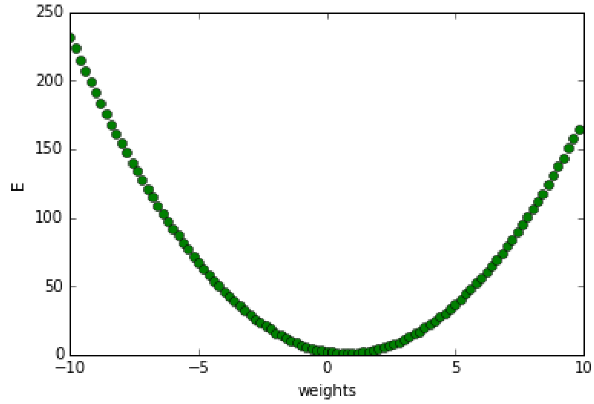


Figure 3: $\varepsilon(w)$ (green circles) and a weight path (blue diamonds connected with a line), $\eta = 0.1$. Minimum weight = 0.750126949946, number of iteration = 18

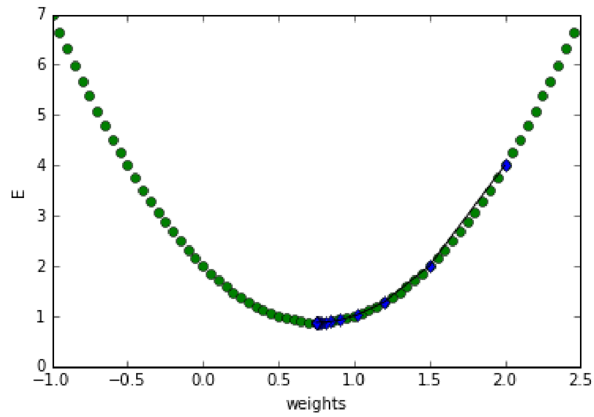


Figure 4: $\varepsilon(w)$ (green circles) and a weight path (blue diamonds connected with a line), $\eta = 0.01$. Minimum weight = 0.752326375959, number of iteration = 154

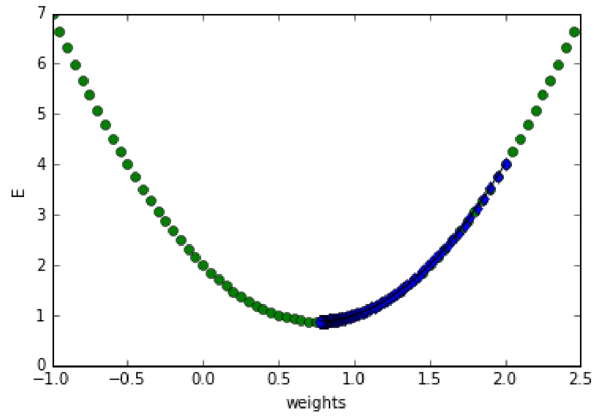


Figure 5: $\varepsilon(w)$ (green circles) and a weight path (blue diamonds connected with a line), $\eta = 0.45$. Minimum weight = 0.750043556143, number of iteration = 46

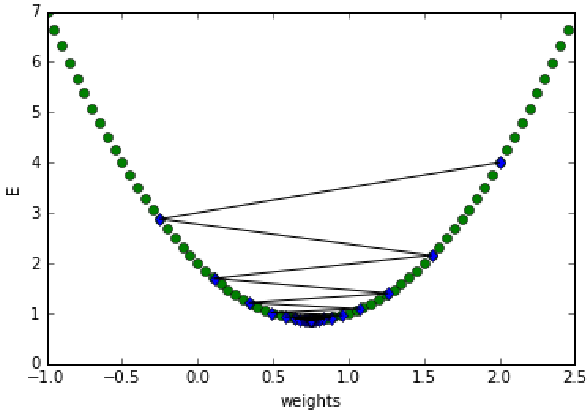
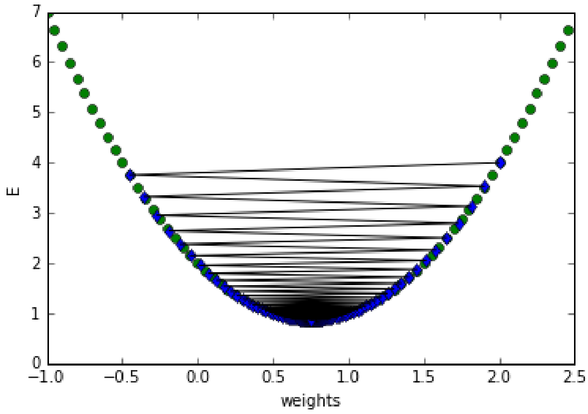


Figure 6: $\varepsilon(w)$ (green circles) and a weight path (blue diamonds connected with a line), $\eta = 0.49$. Minimum weight = 0.749951866545, number of iteration = 249



Exercise 3.4

The correlation matrix R_x of the input vector $x(n)$ in the LMS algorithm is defined by

$$R_x = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

Define the range of values for the learning-rate parameter η of the LMS algorithm for it to be convergent in the mean square.

Solution:

The algorithm converges iff:

$0 < \eta < \frac{2}{\lambda_{max}}$,
where λ_{max} is the larges eigenvalue of the correlation matrix R_x.

$$R_x - \lambda I = \begin{vmatrix} 1-\lambda & 0.5 \\ 0.5 & 1-\lambda \end{vmatrix} = \lambda^2 - 2\lambda + 0.75,$$

so $\lambda_1 = 1.5$ and $\lambda_2 = 0.5$. Therefore, $\lambda_{max} = 1.5$ and:
$0 < \eta < \frac{2}{1.5} \Rightarrow 0 < \eta < 1.3333$