

BRSU

Neural Networks Assignment 5

Bastian Lang

November 7, 2015

This report contains the summary, ex3.1 and ex3.2 without any programming/plotting.

1 SUMMARY

Please see figures 1.1 - 1.6.

2 Ex3.1

See figure 2.1

3 Ex3.2

3.1 A

See figure 3.1

3.2 B

See figure 3.2 for formulas.

Haykin Chapter 3

• Introduction (3.1)

- Formative year's outstanding researchers
 - McCulloch & Pitts: Introduction NN as computing machines
 - Hebb: First rule for learning
 - Rosenblatt: Perceptron first model for learning with a teacher
- Rosenblatt introduction of Perceptron for classification of linearly separable patterns
- Perceptron Convergence Theorem
- Use of perceptrons in adaptive filtering

• Adaptive Filtering Problem (3.2)

- Dynamical system that does input-output mapping from input vector to output scalar.
- Training data given
- Input data may
 - be a snapshot of data, where every element originates at a different point in space
 - represent uniformly distributed spaced in time
- How to design this model around a single neuron?
 - Start with arbitrary weight setting
 - Adjustments made to weights on continuous basis
 - Computations completed within one time interval
- ~~Neural~~ Neuronal model ~~refers~~ referred to as Adaptive Filter
- Two processes within adaptive filtering
 - Filtering
 - Compute $y(n)$ given $x(n)$
 - Compute error signal $e(n)$ given $y(n)$ and $d(n)$
 - Adaptive Process
 - Adjust weights in accordance with $e(n)$
- Linear Neuron \rightarrow Output equals induced local field

$$y(n) = u(n) = \sum_k x_k(n) w_k(n)$$

Figure 1.1: summary 1

Neural Networks

Unconstrained Optimisation Techniques

$E(w)$ = cost function, continuously differentiable,
for weight vector w .

Measure of how to choose w .

=> Minimise the cost function $E(w)$ w.r.t. w .

$$\nabla E(w^*) = 0,$$

$$\nabla \hat{=} \text{gradient operator} \quad \nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \dots, \frac{\partial}{\partial w_n} \right]^T$$

Three methods for UOTs relying on iterative descent.

Method of Steepest Descent

Adjustments to weight vector in the direction of
steepest descent. $g = \nabla E(w)$

$$\Rightarrow w(n+1) = \underbrace{\eta}_{\text{stepsize/learning rate}} g(n)$$

$$\Delta w(n) = w(n+1) - w(n) = -\eta g(n) \quad (\text{see error-correction rule})$$

Apply first order Taylor expansion to show that $E(w(n+1)) \leq E(w(n))$
holds

$$\begin{aligned} E(w(n+1)) &= E(w(n)) + g^T(n) \Delta w(n) \\ &= E(w(n)) - \eta g^T(n) g(n) \\ &= E(w(n)) - \eta \|g(n)\|^2 \end{aligned}$$

Depending on η the response of the algorithm can either
be damped, overdamped or unstable.

Figure 1.2: summary 2

Newton's Method

Minimize quadratic approximation of cost function $E(w)$

Using second order Taylor:

$$\begin{aligned}\Delta E(w(n)) &= E(w(n+1)) - E(w(n)) \\ &\approx g^T(n) \Delta w(n) + \frac{1}{2} \Delta w^T(n) H(n) \Delta w(n)\end{aligned}$$

$$H = \nabla^2 E(w)$$

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_1 \partial w_m} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \dots & \frac{\partial^2 E}{\partial w_2 \partial w_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_m \partial w_1} & \frac{\partial^2 E}{\partial w_m \partial w_2} & \dots & \frac{\partial^2 E}{\partial w_m^2} \end{bmatrix}$$

$$\Delta w(n) = -H^{-1}(n) g(n)$$

$$w(n+1) = w(n) + \Delta w(n) = w(n) - H^{-1}(n) g(n)$$

$H(n)$ has to be positive definite. Converges quickly and Q&A does not exhibit unstable behaviour.

Gauss-Newton Method

Applicable to a cost function expressed as the sum of error squares.

$$E(w) = \sum_{i=1}^n e^2(i)$$

scaling factor for simplification

$$e'(i, w) = e(i) + \left[\frac{\partial e(i)}{\partial w} \right]^T (w - w(n))$$

$$e'(n, w) = e(n) + J(n)(w - w(n)), \quad e(n) = [e(n_1), e(n_2), \dots, e(n_n)]^T$$

Figure 1.3: summary 3

$$J(n) = \begin{bmatrix} \frac{\partial e(n)}{\partial \omega_1} & \frac{\partial e(n)}{\partial \omega_2} & \dots & \frac{\partial e(n)}{\partial \omega_m} \\ \frac{\partial e(2)}{\partial \omega_1} & \frac{\partial e(2)}{\partial \omega_2} & & \frac{\partial e(2)}{\partial \omega_m} \\ \vdots & \vdots & & \vdots \\ \frac{\partial e(n)}{\partial \omega_1} & \frac{\partial e(n)}{\partial \omega_2} & & \frac{\partial e(n)}{\partial \omega_m} \end{bmatrix}$$

Jacobian is the transpose of the n-by-n gradient matrix $\nabla e(n)$.

$$\Rightarrow \omega(n+1) = \arg \min \left\{ \frac{1}{2} \|e'(n, \omega)\|^2 \right\}$$

$$\Rightarrow J^T(n) e(n) + J^T(n) J(n) (\omega - \omega(n)) = 0$$

$$\Rightarrow \omega(n+1) = \omega(n) - (J^T(n) J(n))^{-1} J^T(n) e(n)$$

Gauss-Newton only requires the Jacobian of the error vector $e(n)$.

Therefore $J^T(n) J(n)$ must be nonsingular. Q&A

\Rightarrow row rank of $J(n)$ must be $n \hat{=}$ n rows must be linearly independent.

A

To guarantee that, customary practice is to add diagonal matrix δI to ensure positive definite matrix.

$$\omega(n+1) = \omega(n) - (J^T(n) J(n) + \delta I)^{-1} J^T(n) e(n)$$

Linear Least-Squares Filter (3.4)

Two characteristics:

- (1) Single neuron is linear
- (2) Cost function $E(\omega)$ consists of sum of error squares

$$e(n) = d(n) - X(n) \omega(n)$$

$$\nabla e(n) = -X^T(n)$$

$$J(n) = -X(n)$$

Figure 1.4: summary 4

Applying Gauss-Newton

$\omega(n+1) = (X(n)X(n))^+ X(n)d(n)$, where $(X(n)X(n))^+ X(n)$ is the pseudoinverse of $X(n)$.

$$\Rightarrow \omega(n+1) = X(n)^+ d(n)$$

Wiener Filter: Limiting form of the Linear Least Squares Filter for an Ergodic Environment.

When $x(i)$ and $d(i)$ are drawn from an ergodic, the linear least squares filter asymptotically approaches the Wiener filter.

Requires knowledge: correlation matrix, cross-correlation vector.
Not always available \Rightarrow use of linear adaptive filter.

Least-Square algorithm (3.5)

$$E(n) = \frac{1}{2} e^2(n) \quad \frac{\partial E(n)}{\partial \omega} = e(n) \frac{\partial e(n)}{\partial \omega}$$

$$e(n) = d(n) - x(n)\omega(n)$$

$$\frac{\partial e(n)}{\partial \omega} = -x(n) \quad \frac{\partial E(n)}{\partial \omega(n)} = -x(n)e(n)$$

Estimate for gradient vector:

$$\hat{g}(n) = -x(n)e(n)$$

$$\hat{\omega}(n+1) = \hat{\omega}(n) + \eta x(n)e(n)$$

The smaller η , the more past data is remembered. The inverse of η may thus be seen as a measure of memory. Q&A

LMS algorithm does not require knowledge of the statistics of the environment.

Convergence Considerations of the LMS algorithm

Convergence ~~is~~ in the mean square

$$E[e^2(n)] \rightarrow \text{constant as } n \rightarrow \infty$$

Figure 1.5: summary 5

$0 < \eta < \frac{2}{\text{sum of mean-square values of the input}}$

Virtues and limitations of the
LMS Algorithm

+ simple + robust

- slow rate of convergence - sensitivity to variations

Figure 1.6: summary 6

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

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

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

$$g = \nabla E(w)$$

$$w(u+1) = w(u) - \eta g(u)$$

$$\Delta w(u) = -\eta g(u)$$

$$\nabla E(w) = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \dots, \frac{\partial E}{\partial w_m} \right]^T$$

$$w(i) = [w_1(i), w_2(i), \dots, w_m(i)]^T$$

so

$$\frac{\partial E}{\partial w}$$

For a single weight w

$$\nabla E(w) = \frac{\partial E}{\partial w} = -r_{xd} + r_x w$$

$$\Rightarrow \Delta w(u) = -\eta (-r_{xd} + r_x w)$$

$$w(u+1) = w(u) - \eta g(u) = w(u) - \eta (-r_{xd} + r_x w)$$

$$= w(u) + \eta r_{xd} - \eta r_x w(u)$$

$$= w(u)(1-\eta) + \eta r_{xd}$$

Figure 2.1: ex3.1

8.2) Consider the cost-function

$$E(w) = \frac{1}{2} d^2 - r_{rd}^T w + \frac{1}{2} w^T R_x w$$

where d is some constant and

$$r_{rd} = \begin{bmatrix} 0.8182 \\ 0.354 \end{bmatrix}$$

$$R_x = \begin{bmatrix} 1 & 0.8182 \\ 0.8182 & 1 \end{bmatrix}$$

(a) Find the optimum value w^* for which $E(w)$ reaches its minimum value

$$\begin{aligned} E(w) &= \frac{1}{2} d^2 - \begin{bmatrix} 0.8182 \\ 0.354 \end{bmatrix}^T \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 & 0.8182 \\ 0.8182 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \\ &= \frac{1}{2} d^2 - 0.8182 w_1 - 0.354 w_2 + \frac{1}{2} \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} w_1 + 0.8182 w_2 \\ 0.8182 w_1 + w_2 \end{bmatrix} \\ &= \frac{1}{2} d^2 - 0.8182 w_1 - 0.354 w_2 + \frac{1}{2} (w_1^2 + 0.8182 w_1 w_2 + 0.8182 w_1 w_2 + w_2^2) \\ &= \frac{1}{2} d^2 - 0.8182 w_1 - 0.354 w_2 + \frac{1}{2} w_1^2 + \frac{1}{2} w_2^2 + 0.8182 w_1 w_2 \end{aligned}$$

$$\frac{\partial E}{\partial w_1} = -0.8182 + w_1 + 0.8182 w_2$$

$$\frac{\partial E}{\partial w_2} = -0.354 + w_2 + 0.8182 w_1$$

$$\Delta w_1(n) = -\eta (-0.8182 + w_1 + 0.8182 w_2)$$

$$\Delta w_2(n) = -\eta (-0.354 + w_2 + 0.8182 w_1)$$

$$\nabla E(w^*) = 0 \quad w_1 = -0.8182 w_2 + 0.8182 \quad (1)$$

$$w_2 = -0.8182 w_1 + 0.354 \quad (2)$$

$$w_1 = -0.8182 (-0.8182 w_1 + 0.354) + 0.8182 \quad (2) \rightarrow (1)$$

$$w_1 = +0.8182^2 w_1 \leftarrow (0.8182 \cdot 0.354) + 0.8182$$

$$0.331 w_1 = 0.52856$$

$$w_1 \approx 1.597$$

$$w_2 = -0.8182 \cdot 1.597 + 0.354$$

$$= -0.9527$$

Figure 3.1: ex3.2a

(b) Use the method of steepest descent to compute w 's for the following two values of learning-rate parameter

(i) $\eta = 0.3$

(ii) $\eta = 1.0$

(1) $\Delta w_1(n) = +0.2455 - 0.3w_1 - 0.2455w_2$
 $\Delta w_2(n) = 0.1062 - 0.3w_2 - 0.2455w_1$
 $w_1(n+1) = \cancel{0.2455 - 0.3w_1 - 0.2455w_2} w_1(n) + \Delta w_1(n)$
 $= w_1(n) + 0.2455 - 0.3w_1 - 0.2455w_2$
 $w_2(n+1) = w_2(n) + 0.1062 - 0.3w_2 - 0.2455w_1$

(2) $\Delta w_1(n) = -\cancel{0.8182} (-0.8182 + w_1 + 0.8182w_2) = 0.8182 - w_1 - 0.8182w_2$
 $\Delta w_2(n) = -\cancel{0.354} (-0.354 + w_2 + 0.8182w_1) = 0.354 - w_2 - 0.8182w_1$

Figure 3.2: ex3.2b