Νευρο-Ασαφής Υπολογιστική
Neuro-Fuzzy Computing

Διδάσκων –
Δημήτριος Κατσαρός

@ Τμ. ΗΜΜΥ
Πανεπιστήμιο Θεσσαλίας

Διάλεξη 5η

Adaptive Linear Neuron (ADALINE)

A bit of history

- Bernard Widrow began working in neural networks in the late 1950s
  - at about the same time that Frank Rosenblatt developed the perceptron learning rule
- In 1960 Widrow, and his graduate student Marcian Hoff
  - introduced the ADALINE (ADAptive LInear NEuron) network, and
  - a learning rule which they called the LMS (Least Mean Square) algorithm
- ADALINE network is very similar to the perceptron
  - except that its transfer function is linear, instead of hard-limiting
- ADALINE and the perceptron suffer from the same inherent limitation: *they can only solve linearly separable problems*

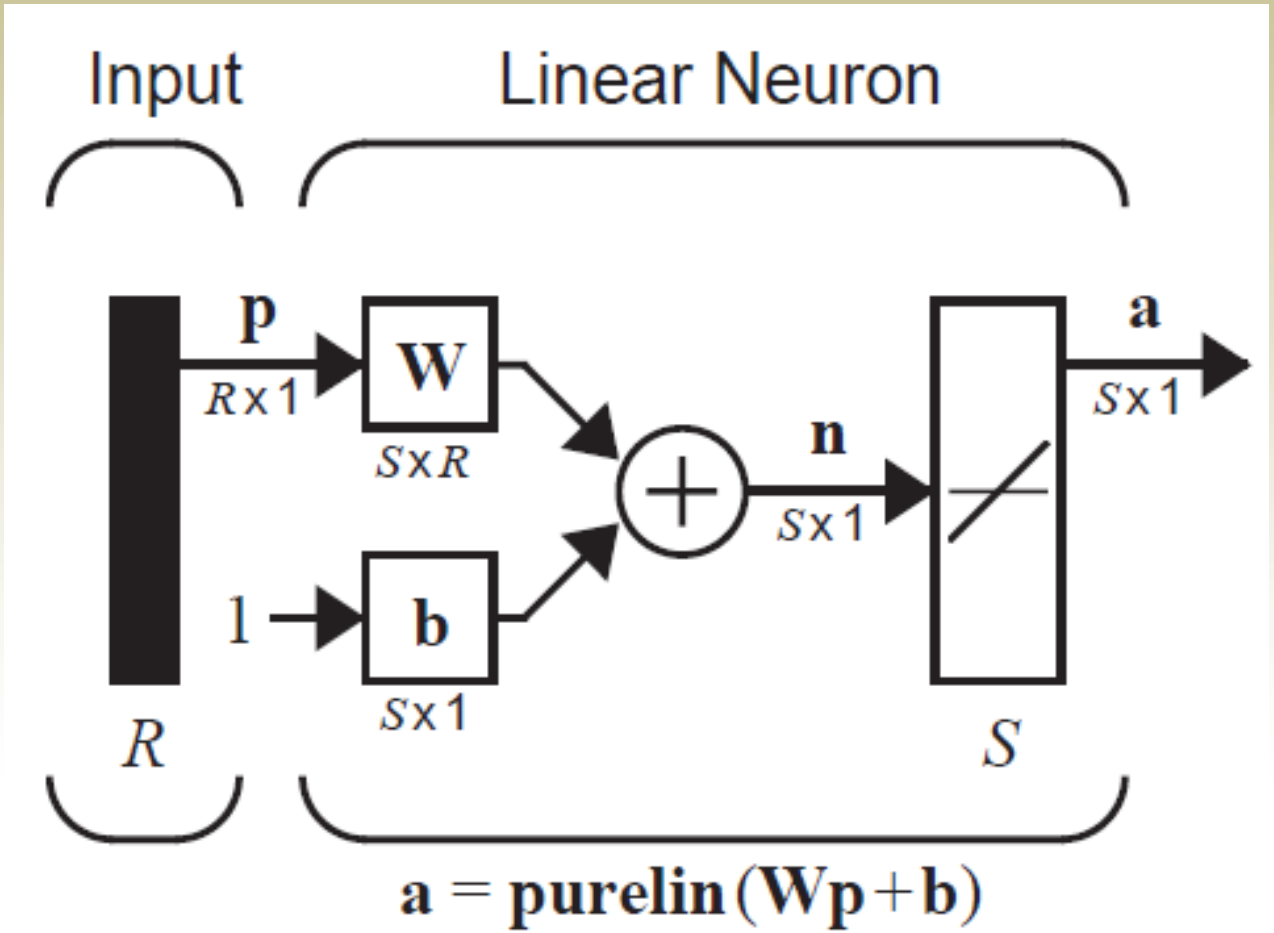
A bit of history

- ADALINE network versus perceptron
  - The LMS algorithm is more powerful than the perceptron learning rule
    - While the perceptron rule is guaranteed to converge to a solution that correctly categorizes the training patterns, the resulting network can be sensitive to noise, since patterns often lie close to the decision boundaries
    - The LMS algorithm minimizes mean square error, and therefore tries to move the decision boundaries as far from the training patterns as possible
  - Widrow-Hoff learning is an approximate steepest descent algorithm, in which the performance index is mean square error
  - ADALINE is important for two reasons
    - First, it is widely used today in many signal processing applications
    - In addition, it is the precursor to the *backpropagation* algorithm for multilayer networks

ADALINE network

- The output of ADALINE is:

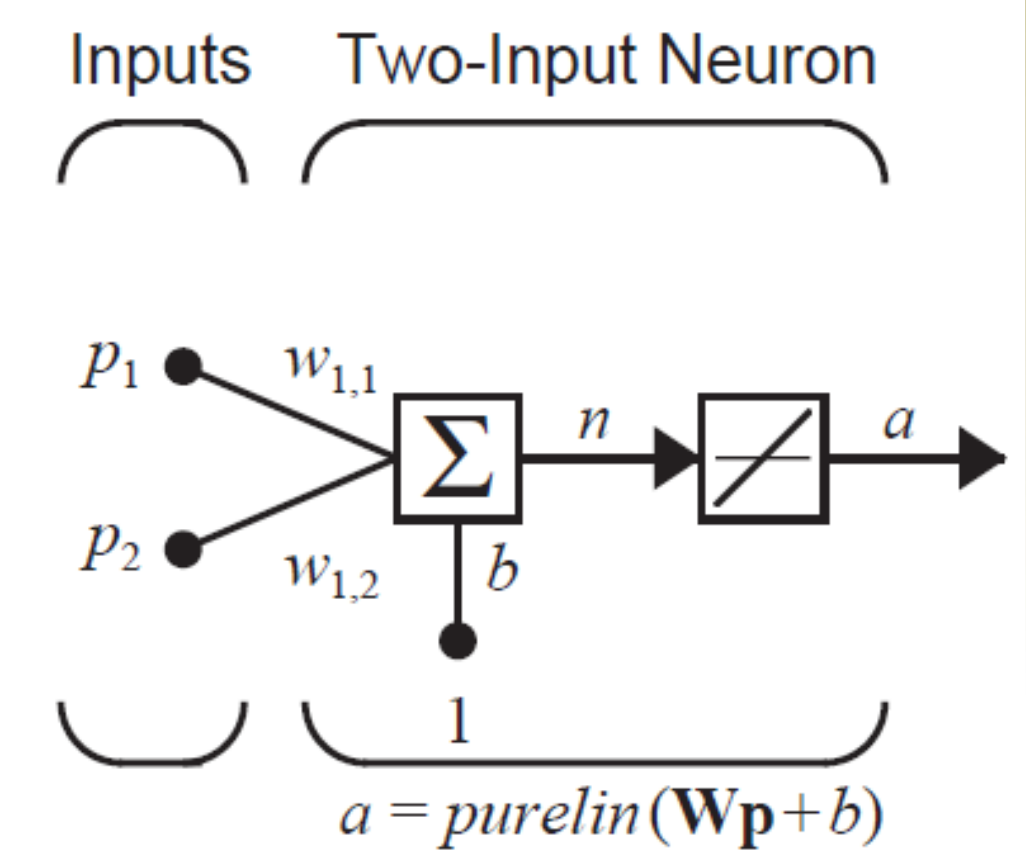
$$a = \text{purelin}(\mathbf{Wp}+\mathbf{b}) = \mathbf{Wp}+\mathbf{b}$$



Single ADALINE

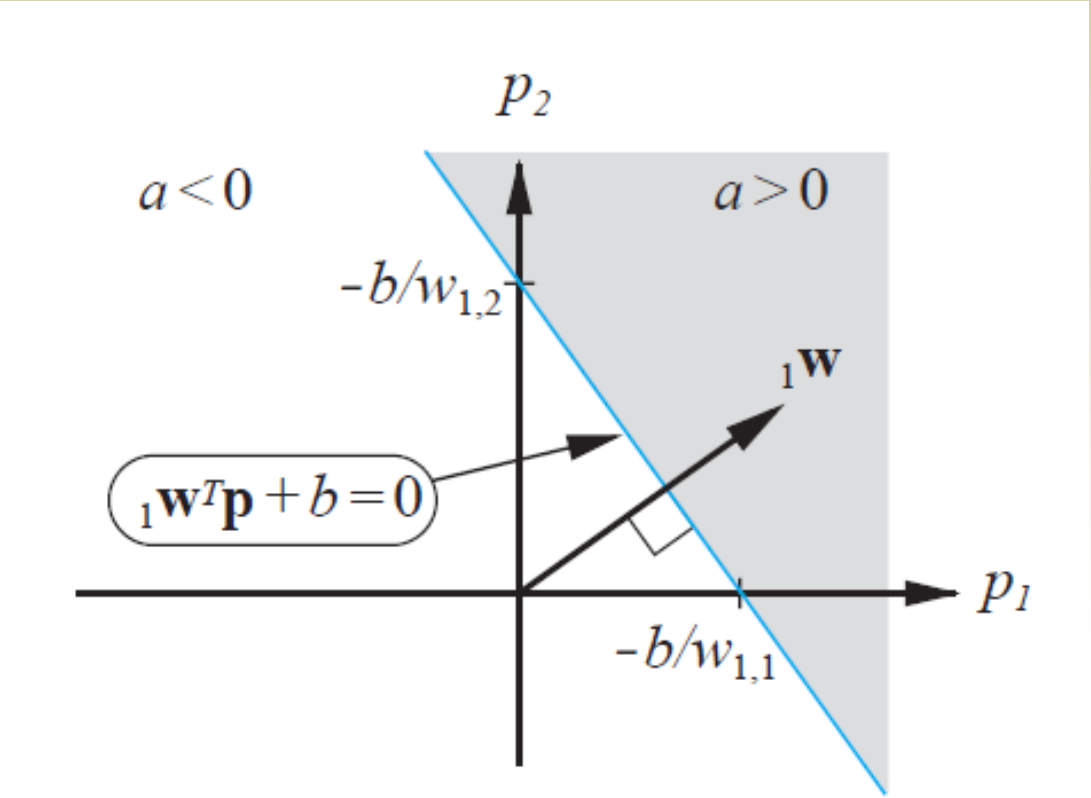
- The output of this ADALINE is:

$a = \text{purelin}(n) = \text{purelin}({}_1\mathbf{w}^T\mathbf{p}+\mathbf{b}) = {}_1\mathbf{w}^T\mathbf{p}+b = w_{1,1}p + w_{1,2}p + b$



The induced decision boundary

- By setting n=0, we get the equation of the line (decision boundary)
- ADALINE can be used to classify objects into two categories
  - However, it can do so only if the objects are linearly separable



ADALINE performance measure: Mean Square Error (MSE)

- ADALINE error: the difference between the target output and the network output
- Suppose we define $\mathbf{x}=({}_1\mathrm{w}\ \mathrm{b})^T$
  - and include the bias input '1' as a component of the input vector
- Now, instead of $\alpha={}_1\mathbf{w}^T\mathbf{p} + \mathrm{b}$, we have: $\alpha= \mathbf{x}^T\mathbf{z}$
- Thus, mean square error: $F(\mathbf{x})= E[e^2]= E[(t-\alpha)^2] = E[(t-\mathbf{x}^T\mathbf{z})^2]$, over all sets of input-output, and E denotes expectation
- We can expand the above as follows:

$$F(\mathbf{x})= E[(t^2-2t\mathbf{x}^T\mathbf{z}+\mathbf{x}^T\mathbf{z}\mathbf{z}^T\mathbf{x})^2]= E[t^2] - 2\mathbf{x}^T E[t\mathbf{z}] + \mathbf{x}^T E[\mathbf{z}\mathbf{z}^T]\mathbf{x}$$

- and in convenient form:

$$F(\mathbf{x})= c - 2\mathbf{x}^T\mathbf{h}+\mathbf{x}^T\mathbf{R}\mathbf{x}$$

where $c= E[t^2]$, $\mathbf{h}= E[t\mathbf{z}]$, and $\mathbf{R}= E[\mathbf{z}\mathbf{z}^T]$

- vector **h** gives the cross-correlation between the input vector and its associated target, while **R** is the input correlation matrix. The diagonal elements of this matrix are equal to the mean square values of the elements of the input vectors

Recall Quadratic functions

- Generic quadratic function:

$$F(\mathbf{x}) = c + \mathbf{d}^T\mathbf{x} + \tfrac{1}{2}\,\mathbf{x}^T\mathbf{A}\mathbf{x}$$

- Thus, ADALINE's MSE is a quadratic function

where $\mathbf{d}=-2\mathbf{h}$ and $\mathbf{A}=2\mathbf{R}$

- The characteristics of the quadratic function depend primarily on the *Hessian* matrix
  - For example, if the eigenvalues of the Hessian are all positive, then the function will have one unique global minimum
- Here, the Hessian matrix is twice the correlation matrix **R**, and it can be shown that all correlation matrices are either positive definite or positive semidefinite, which means that they can never have negative eigenvalues

Recall Quadratic functions

- We are left with two possibilities:
  - If the correlation matrix has only positive eigenvalues, the performance index will have one unique global minimum
  - If the correlation matrix has some zero eigenvalues, the performance index will either have a *weak minimum* (explanation during the class lecture) or no minimum depending on the vector $\mathbf{d}=-2\mathbf{h}$

Investigation of the gradient

- Let's locate the *stationary* (where gradient equals zero) points:
  - grad $F(\mathbf{x})$ = grad[c+$\mathbf{d}^T\mathbf{x}$ + ½ $\mathbf{x}^T\mathbf{A}\mathbf{x}$] = $\mathbf{d} + \mathbf{A}\mathbf{x} \rightarrow$

$$\text{grad } F(\mathbf{x}) = -2\mathbf{h} + 2\mathbf{R}\mathbf{x}$$

  - Stationary points can be found by setting the above equal to zero $-2\mathbf{h} + 2\mathbf{R}\mathbf{x} = 0$
- If the correlation matrix is positive definite there will be a unique stationary point, which will be a *strong minimum*:

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$$

- It is worth noting here that the existence of a unique solution depends only on the correlation matrix **R**. Therefore the characteristics of the input vectors determine whether or not a unique solution exists

Least Mean Square (LMS) algorithm: The training algorithm of the ADALINE

- The next step is to design an algorithm to locate the minimum point
  - If we could calculate the statistical quantities **h** and **R**, we could find the minimum point directly
  - If we did not want to calculate the inverse of **R**, we could use the steepest descent algorithm, with the gradient calculated from previous slides
    - In general it is not desirable or convenient to calculate **h** and **R**
    - For this reason we will use an approximate steepest descent algorithm, in which we use an estimated gradient
- The key insight of Widrow and Hoff was that they could estimate the mean square error $F(\mathbf{x})$ by

$$\hat{F}(x) = (t(k) - a(k))^2 = e^2(k)$$

where the expectation of the squared error has been replaced by the squared error at iteration k

Least Mean Square (LMS) algorithm: The training algorithm of the ADALINE

- At each iteration we have a gradient estimate of the form:

$$\hat{\nabla} F(x) = \nabla e^2(k)$$

- This is sometimes referred to as the *stochastic gradient*. When this is used in a gradient descent algorithm, it is referred to as "on-line" or incremental learning, since the weights are updated as each input is presented to the network
- The first R elements of $\nabla e^2(k)$ are derivatives with respect to the network weights , while the $(R+1)^{st}$ element is the derivative with respect to the bias. Thus

$$[\nabla e^2(k)]_j = \frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k)\frac{\partial e(k)}{\partial w_{1,j}} \quad \text{for} \quad j = 1, 2, \ldots, R$$

and

$$[\nabla e^2(k)]_{R+1} = \frac{\partial e^2(k)}{\partial b} = 2e(k)\frac{\partial e(k)}{\partial b}$$

Least Mean Square (LMS) algorithm: The training algorithm of the ADALINE

Now consider the partial derivative terms at the ends of these equations. First evaluate the partial derivative of $e(k)$ with respect to the weight $w_{1,j}$:

$$\frac{\partial e(k)}{\partial w_{1,j}} = \frac{\partial (t(k) - \alpha(k))}{\partial w_{1,j}} = \frac{\partial}{\partial w_{1,j}} \Big(t(k) - ({}_1w^T p(k) + b) \Big)$$

$$= \frac{\partial}{\partial w_{1,j}} \left[t(k) - \left(\sum_{i=1}^{R} w_{1,i} p_i(k) + b \right) \right]$$

where $p_i(k)$ is the i-th element of the input vector at the k-th iteration. This simplifies to

$$= \frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k)$$

Least Mean Square (LMS) algorithm: The training algorithm of the ADALINE

- Similarly, $$\frac{\partial e(k)}{\partial b} = -1$$
- Notice that $p_j(k)$ and 1 are the elements of the input vector **z**, so the gradient of the squared error at iteration k can be written as:

$$\hat{\nabla} F(x) = \nabla e^2(k) = -2e(k)z(k)$$

- This approximation to grad $F(\mathbf{x})$ can be used in the steepest descent algorithm. The steepest descent, with constant learning rate, is

$$x_{k+1} = x_k - \alpha \nabla F(x)\big|_{x=x_k}$$

Least Mean Square (LMS) algorithm: The training algorithm of the ADALINE

- By substitution from previous slide, we get

$$x_{k+1} = x_k + 2\alpha e(k) z(k)$$

- or $${}_1w(k+1) = {}_1w(k) + 2\alpha e(k) p(k)$$

- and $$b(k+1) = b(k) + 2\alpha e(k)$$

- These last two equations make up the least mean square (LMS) algorithm
- This is also referred to as the *delta rule* or the *Widrow-Hoff learning algorithm*

Least Mean Square (LMS) algorithm: The training algorithm of the ADALINE

- The preceding results can be modified to handle the case where we have multiple outputs, and therefore multiple neurons
- To update the i-th row of the weight matrix use

$$_i\mathbf{w}(k+1) = {}_i\mathbf{w}(k) + 2\alpha e_i(k)\mathbf{p}(k)$$

where $e_i(k)$ is the i-th element of the error at iteration k

- To update the i-th element of the bias, we use

$$b_i(k+1) = b_i(k) + 2\alpha e_i(k)$$

The LMS algorithm

The LMS algorithm can be written in matrix notation as follows:

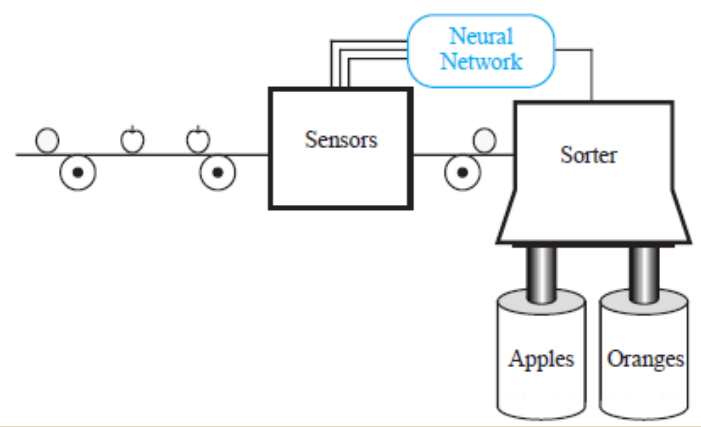
$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha\mathbf{e}(k)\mathbf{p}^T(k)$$

and

$$\mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha\mathbf{e}(k)$$

- *What about the convergence of the LMS algorithm?* (next lecture)
- *Are there any bounds (e.g., maximum value) on the learning rate?* (next lecture)

Exercise-01: Example execution of LMS



$$\mathbf{p} = \begin{bmatrix} shape \\ texture \\ weight \end{bmatrix}$$

orange

$$\mathbf{p}_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

apple

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Target output for oranges: -1

Target output for apples: 1

$\mathbf{W}(0) = [0\ 0\ 0]$

learning rate= 2

Problem: Which vector is the solution (convergence vector)?

Exercise-02: Investigation of MSE surface

Exercise. Suppose that we have the following input/target pairs:

$$\left\{\mathbf{p}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_1 = 1\right\}, \left\{\mathbf{p}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, t_2 = -1\right\}$$

These patterns occur with *equal probability*, and they are used to train an ADALINE network with no bias.

- What does the mean square error (MSE) performance surface look like?

Solution. We need to calculate the various terms of the quadratic function.

Recall that: $F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x}$

Therefore, we need to calculate: c, **h** and **R**

Exercise-02: Investigation of MSE surface

These patterns occur with *equal probability*, so the respective targets occur with equal probability.

Thus the expected value of the square of the targets is:

$$E[t^2] = (1)^2(0.5) + (-1)^2(0.5) = 1.$$

In a similar way, the cross-correlation between the input and the target can be calculated:

$$\mathbf{h} = E[t\mathbf{z}] = (0.5)(1)\begin{bmatrix} 1 \\ 1 \end{bmatrix} + (0.5)(-1)\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

The input correlation matrix $\mathbf{R}$ is

$$\mathbf{R} = \mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}] = \mathbf{p}_1\mathbf{p}_1^{\mathrm{T}}(0.5) + \mathbf{p}_2\mathbf{p}_2^{\mathrm{T}}(0.5) \Rightarrow$$

$$\mathbf{R} = (0.5)\left[\begin{bmatrix} 1 \\ 1 \end{bmatrix}[1\ 1] + \begin{bmatrix} 1 \\ -1 \end{bmatrix}[1\ -1]\right] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Exercise-02: Investigation of MSE surface

Therefore the MSE index is:

$F(\mathbf{x}) = c - 2\mathbf{x}^T\mathbf{h} + \mathbf{x}^T\mathbf{R}\mathbf{x} =$

$$= 1 - 2\begin{bmatrix} w_{1,1} & w_{1,2} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} w_{1,1} & w_{1,2} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} w_{1,1} \\ w_{1,2} \end{bmatrix} =$$

$$= 1 - 2w_{1,2} + w_{1,1}^2 + w_{1,2}^2$$

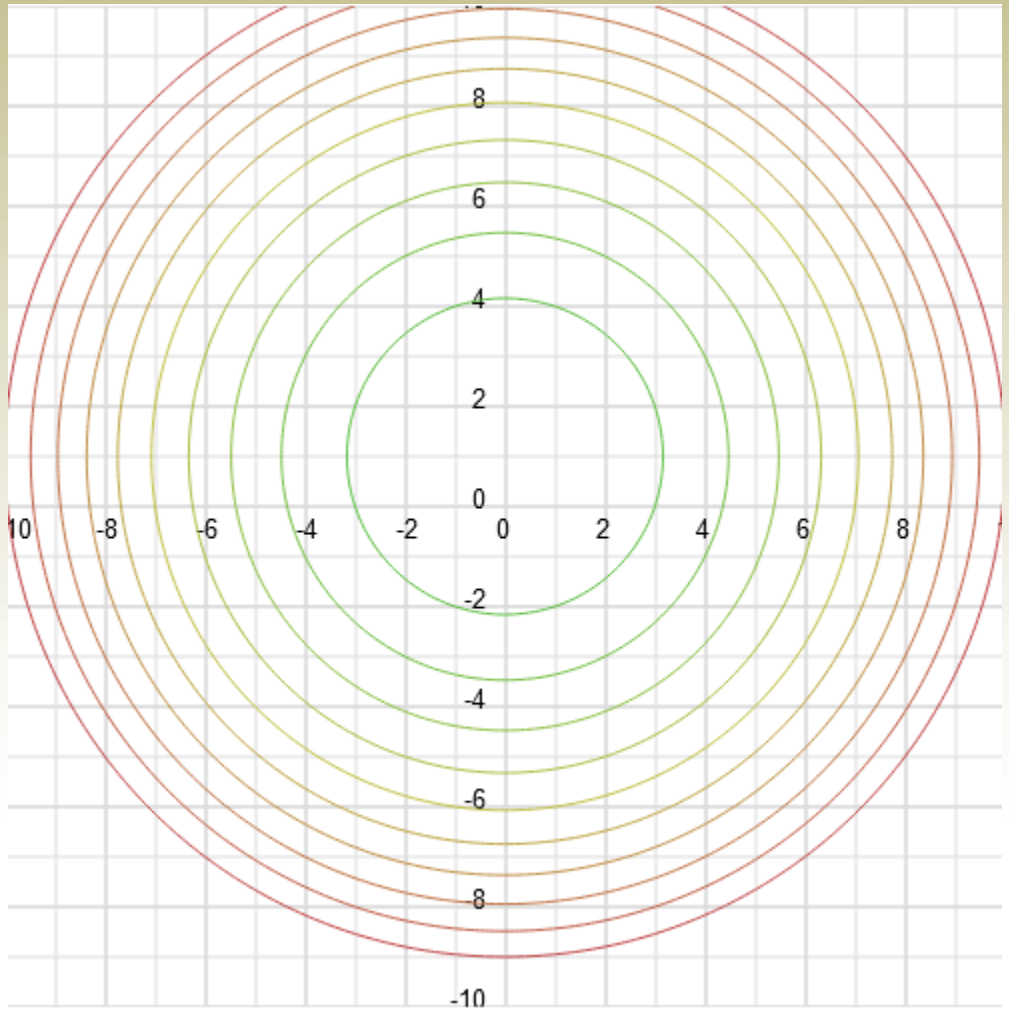
The Hessian matrix of $F(\mathbf{x})$, which is equal to $2\mathbf{R}$, has both eigenvalues at 2. Therefore, the contours are circular. To find the center of the contours (the minimum point), we need to solve:

$$\mathbf{x}^\star = \mathbf{R}^{-1}\mathbf{h} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

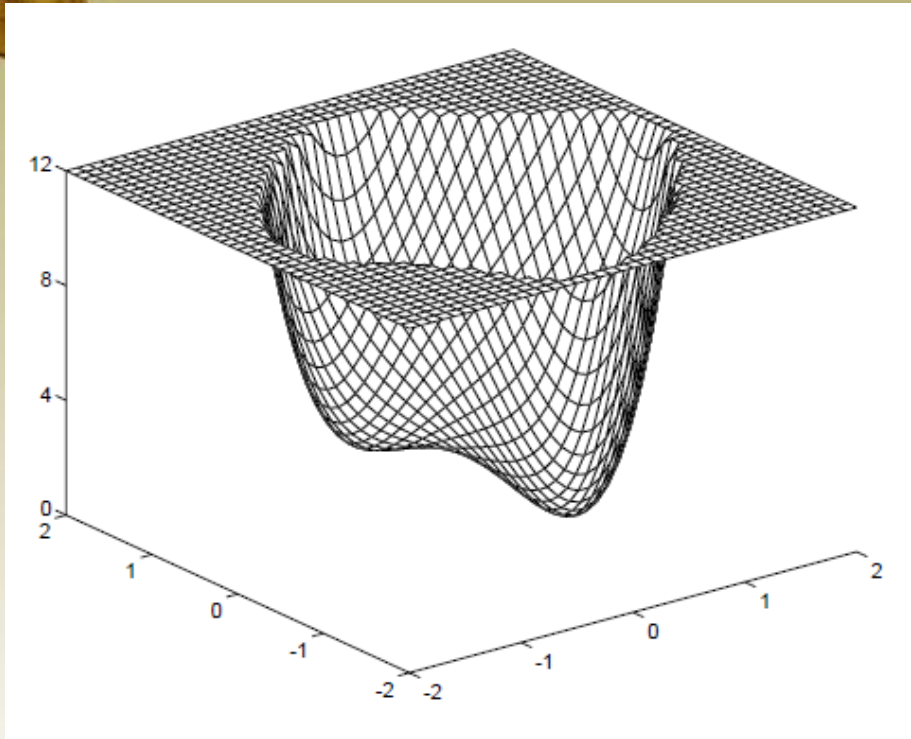
Thus, we have a minimum at $w_{1,1} = 0$, $w_{1,2} = 1$

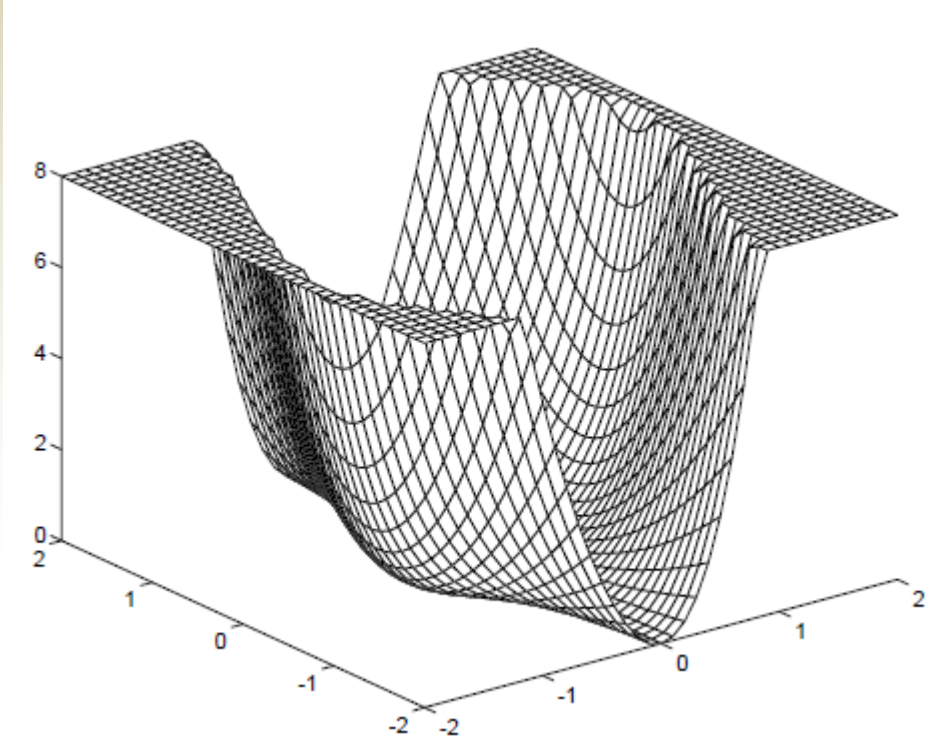
Exercise-02: Investigation of MSE surface

The resulting MSE surface is shown below (plot done online with https://academo.org/demos/contour-plot/):



Appendix: strong, global, weak minimum





Appendix: Hessian

$$\nabla^2 F(x) = \begin{pmatrix} \frac{\partial^2}{\partial x_1^2} F(x) & \frac{\partial^2}{\partial x_1 \partial x_2} F(x) & \dots & \frac{\partial^2}{\partial x_1 \partial x_n} F(x) \\ \frac{\partial^2}{\partial x_2 \partial x_1} F(x) & \frac{\partial^2}{\partial x_2^2} F(x) & \dots & \frac{\partial^2}{\partial x_2 \partial x_n} F(x) \\ \frac{\partial^2}{\partial x_n \partial x_1} F(x) & \frac{\partial^2}{\partial x_n \partial x_2} F(x) & \dots & \frac{\partial^2}{\partial x_n^2} F(x) \end{pmatrix}$$

In the next lecture, an example of quadratic function optimization with Steepest Descent using Hessian. Differs from what we have presented in λ_k. Now it is:

$$\lambda_k = \frac{||\nabla f(x_k)||^3}{\nabla^T f(x_k) \mathbf{H} \nabla f(x_k)}$$

Analysis of convergence

- Note that in the LMS algorithm, $\mathbf{x}_{k+1}$ is a function only of z(k-1), z(k-2), …
- If we assume that successive input vectors are statistically independent, then **x** is independent of **z**. We will show in the following development that for stationary input processes meeting this condition, the expected value of the weight vector will converge to

$$\mathbf{x}^* = \mathbf{R}^{-1}\mathbf{h}$$

- This is the minimum mean square error
- Recall the LMS algorithm

$$\mathbf{x}_{k+1} = \mathbf{x}_k + 2\alpha e(k)\mathbf{z}(k)$$

- Taking expectations

$$E(\mathbf{x}_{k+1}) = E(\mathbf{x}_k) + 2\alpha E[e(k)\mathbf{z}(k)]$$

Analysis of convergence

- Substitute $t(k)$-$\mathbf{x}_k^T\mathbf{z}(k)$ for the error to give

$$E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha\{E[t(k)\mathbf{z}(k)] - E[(\mathbf{x}_k^T\mathbf{z}(k))\mathbf{z}(k)]\}$$

- Finally, substitute $\mathbf{z}^T(k)\mathbf{x}_k$ for $\mathbf{x}_k^T\mathbf{z}(k)$ and rearrange terms to give

$$E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha\{E[t_k\mathbf{z}(k)] - E[\mathbf{z}(k)\mathbf{z}^T(k)\mathbf{x}_k]\}$$

- Since $\mathbf{x}_k$ is independent of $\mathbf{z}(k)$

$$E[\mathbf{x}_{k+1}] = E[\mathbf{x}_k] + 2\alpha\{\mathbf{h} - \mathbf{R}E[\mathbf{x}_k]\}$$

- This can be written as

$$E[\mathbf{x}_{k+1}] = [\mathbf{I} - 2\alpha\mathbf{R}]\, E[\mathbf{x}_k] + 2\alpha\mathbf{h}$$

- This dynamic system will be stable if all of the eigenvalues of $[\mathbf{I} - 2\alpha\mathbf{R}]$ fall inside the unit circle
- It is known that the eigenvalues of $[\mathbf{I} - 2\alpha\mathbf{R}]$ will be $1\text{-}2\alpha\lambda_i$, where the λ_i are the eigenvalues of $\mathbf{R}$.

Analysis of convergence

- Therefore the system will be stable if

$$1-2\alpha\lambda_i > -1$$

- Since $\lambda_i>0$, $1-2\alpha\lambda_i$ is always less than 1. The condition on stability is therefore

$$\alpha < 1/\lambda_i, \text{ for all i}$$

- or

$$0 < \alpha < 1/\lambda_{max}$$

- If this condition on stability is satisfied, the steady state solution is

$$E[\mathbf{x}_{ss}] = [\mathbf{I}-2\alpha\mathbf{R}]\, E[\mathbf{x}_{ss}] + 2\alpha\mathbf{h}$$

- or

$$E[\mathbf{x}_{ss}] = \mathbf{R}^{-1}\mathbf{h} = \mathbf{x}^*$$