innovate achieve lead

BITS Pilani

Machine Learning (IS ZC464) Session 10:

Artificial Neural Networks(ANN) – Perceptron and Linear decision Boundary, Pattern Recognition using ANN, Gradient Descent Algorithm

Learning

- The procedure that consists in estimating the parameters of neurons so that the whole network can perform a specific task.

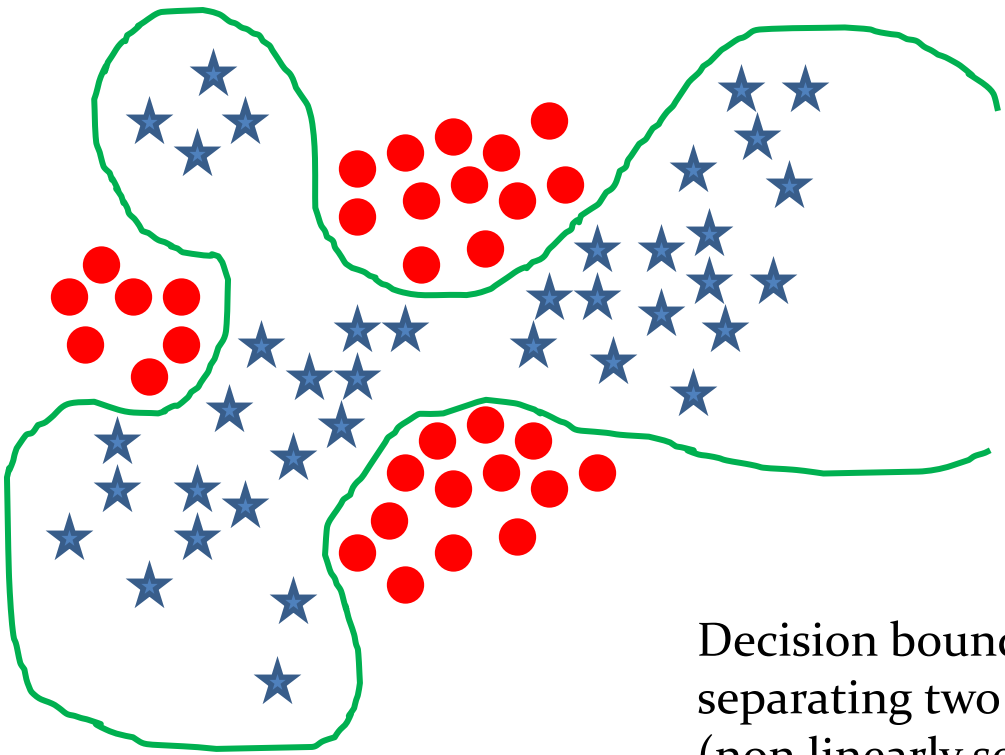
- Types of learning
  - ❑ The supervised learning
  - ❑ The unsupervised learning

- The Learning process (supervised)
  - ❑ Present the network a number of inputs and their corresponding outputs
  - ❑ See how closely the actual outputs match the desired ones
  - ❑ Modify the parameters to better approximate the desired outputs

Recall

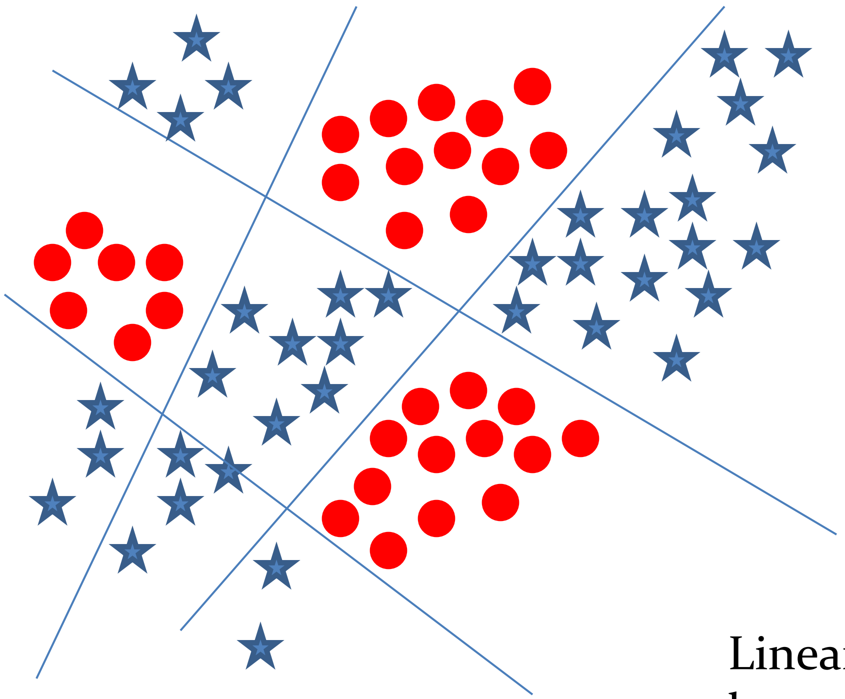
- Knowledge is acquired by the network through a learning process.
- Interconnection strengths known as synaptic weights are used to store the knowledge.

Classification Example: Draw decision boundaries



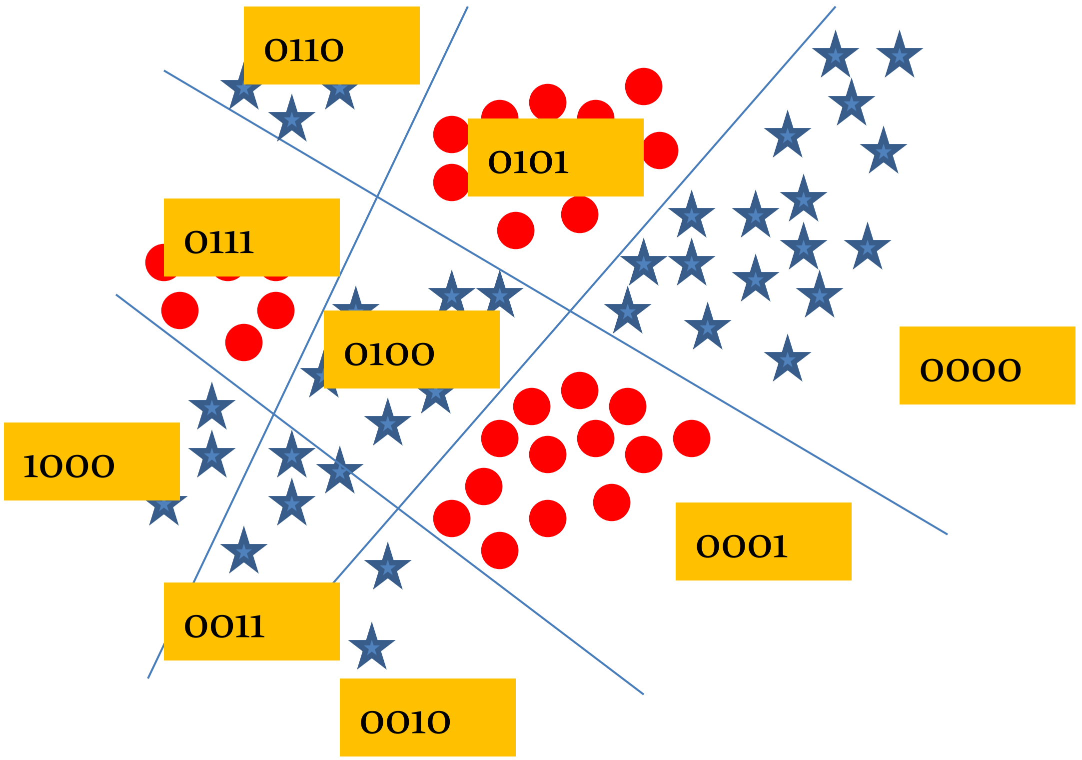
Decision boundary
separating two classes
(non linearly separable)

Classification Example: Linear boundaries

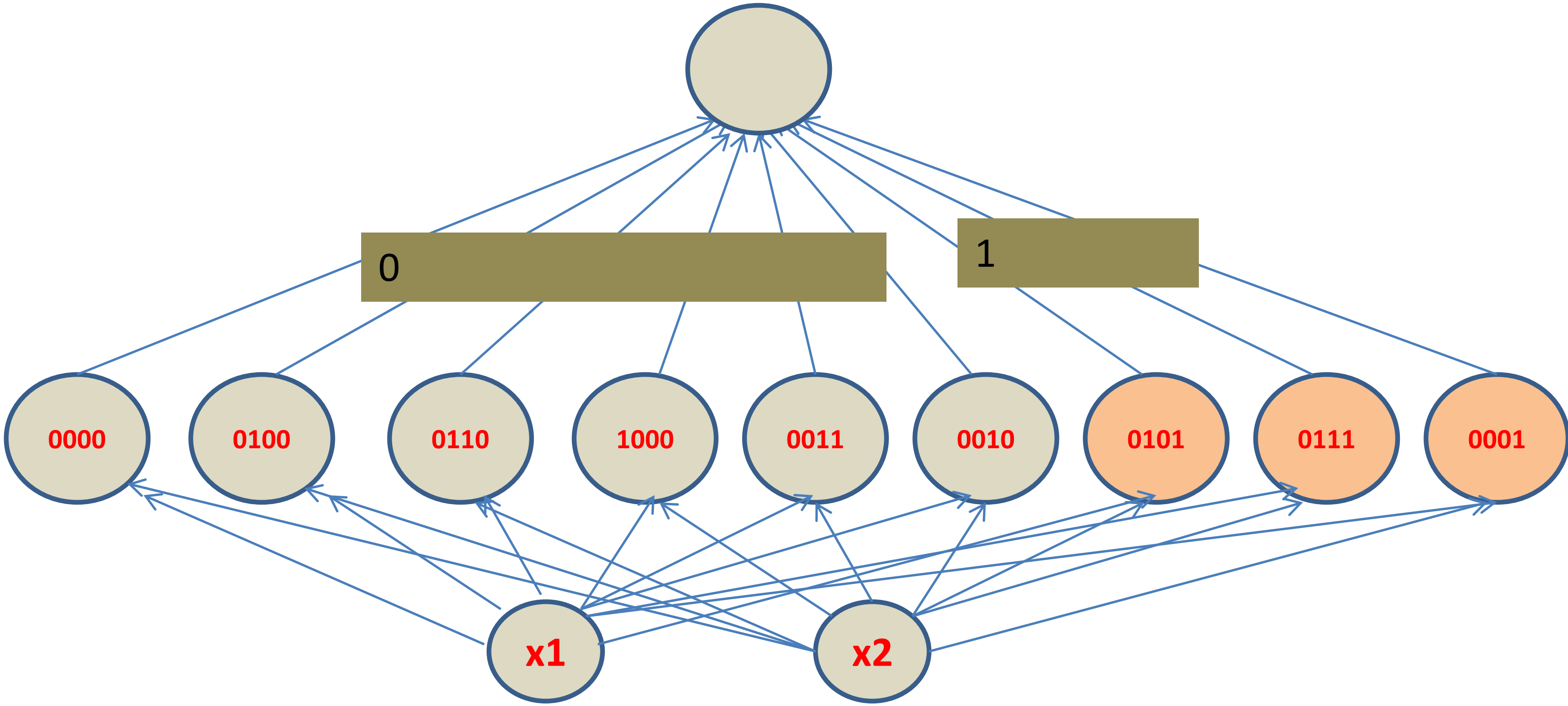


Linear Decision boundaries separating two classes

Example: give each region a label



Role of neurons in design of a neural network



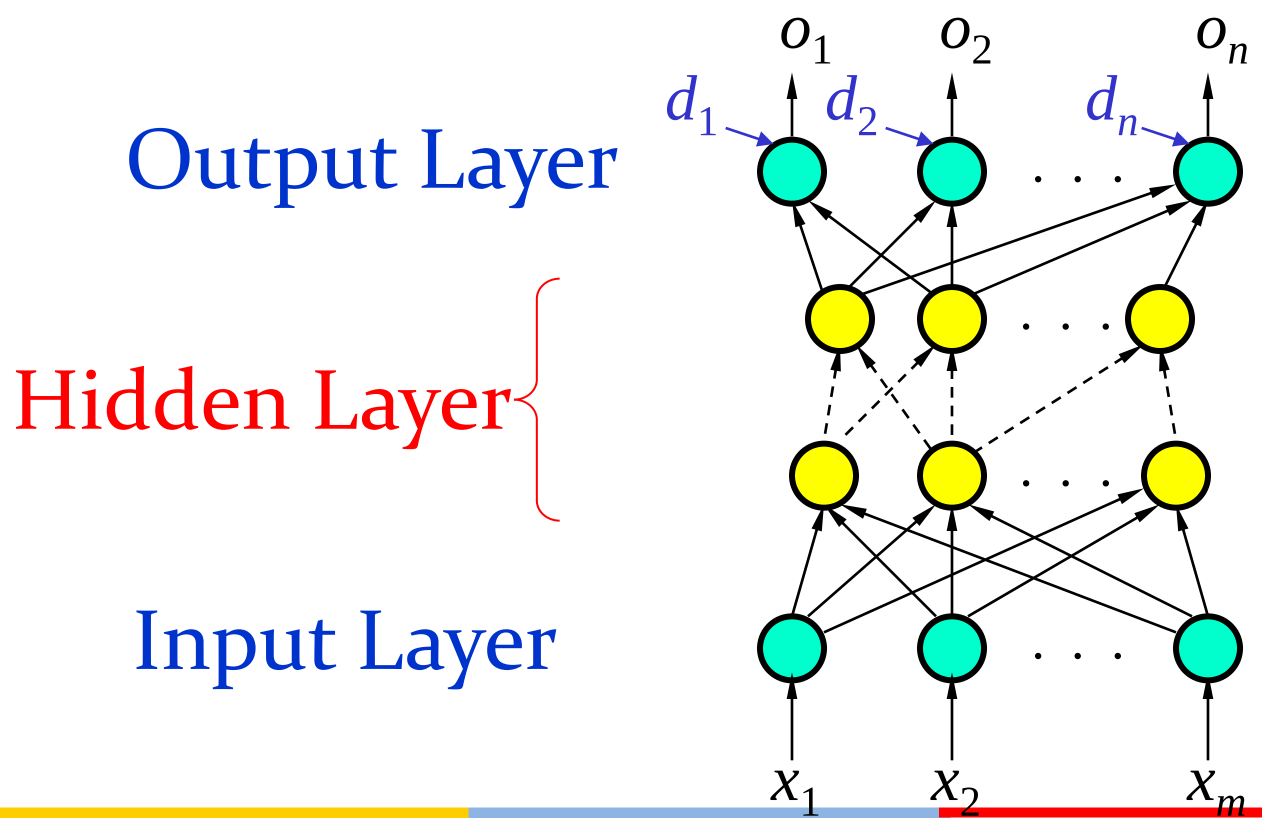
Input

Represent Training Data

- $\{x^{(i)}, d^{(i)}\}$ for i = 1,2,3,….,m
- Size of the training data = m
- $x^{(1)}$ is the feature vector corresponding to the first object
- $x^{(2)}$ is the feature vector corresponding to the first object
- $d^{(1)}$ is the class to which $x^{(1)}$ belongs
- $d^{(2)}$ is the class to which $x^{(2)}$ belongs
- And so on

Forward Learning

O1,o2,o3 etc are the output values produced by the NN



Goal

Sum of Squared Errors

$$E^{(l)} = \frac{1}{2}\sum_{j=1}^{n}\left[d_j^{(l)} - o_j^{(l)}\right]^2$$

Goal:

Minimize

$$E = \sum_{l=1}^{p} E^{(l)}$$

Learning Factors

- Initial Weights
- Learning Constant (η)
- Cost Functions
- Update Rules
- Training Data and Generalization
- Number of Layers
- Number of Hidden Nodes

Learning Phase

- During the learning phase the weights in the Feed Forward Neural Network are modified.
- All weights are modified in such a way that when a pattern is presented, the output unit with the correct category, hopefully, will have the largest output value.

In 2D space the line parameters are two

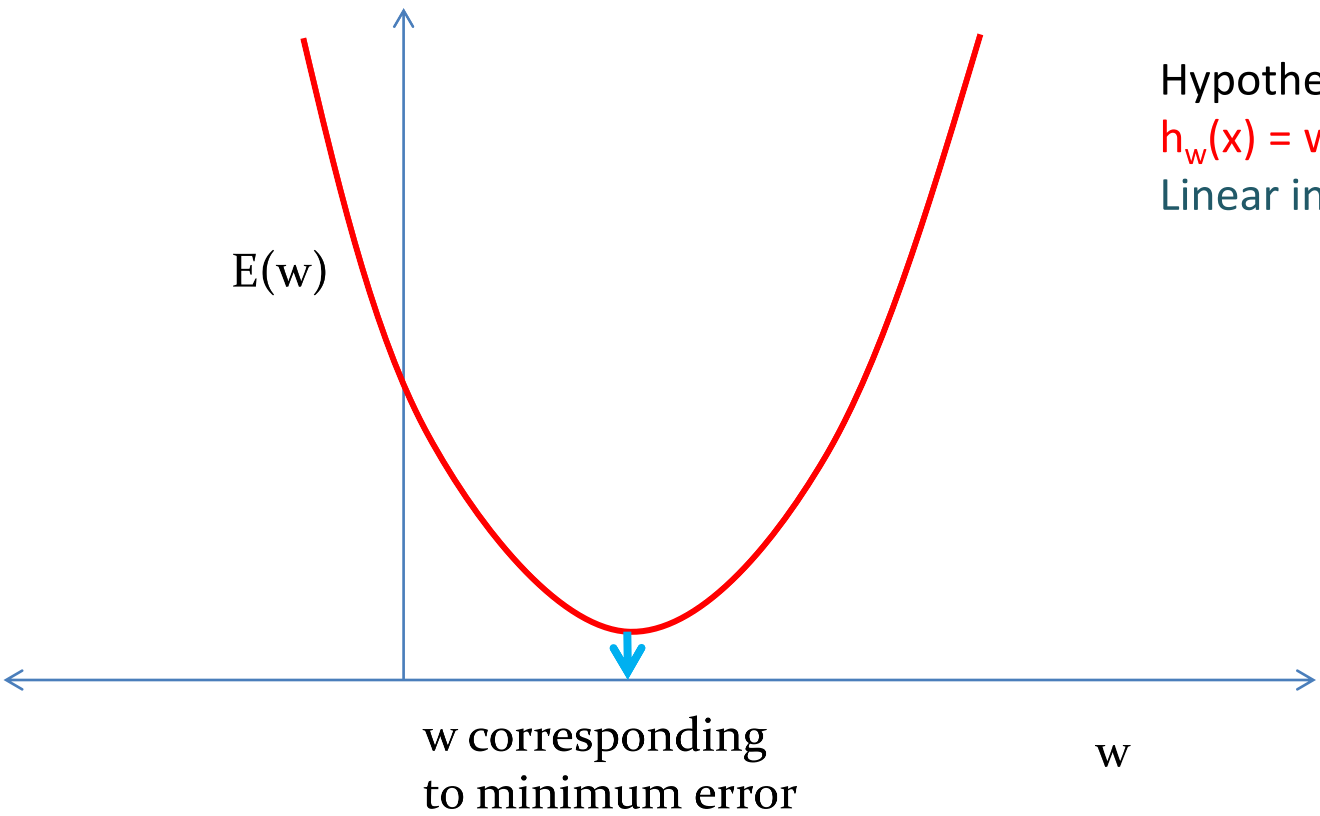
- Slope and intercept
- Can be called as w_1 and w_2
- In order to find a line that best fits the given data, we must find w1 and w2 in such a way that the sum of the squared error is minimum

Error surface for Neural Network based classification

- Consider m observations $<x^1,y^1>$, $<x^2,y^2>$, ….$<x^m,y^m>$.
- An hypothesis $h_w(x)$ that approximates the function that fits best to the given values of y
- There is likely to be some error corresponding to each observation (say i).
- The magnitude of such error is $y^i - h_w(x^i)$
- Objective is to find such w that minimizes the sum of squares of errors

$$E_{min}(w) = \text{Minimize}_w \sum_i (y^i - h_w(x^i))^2$$

Plotting error when y=f(x)

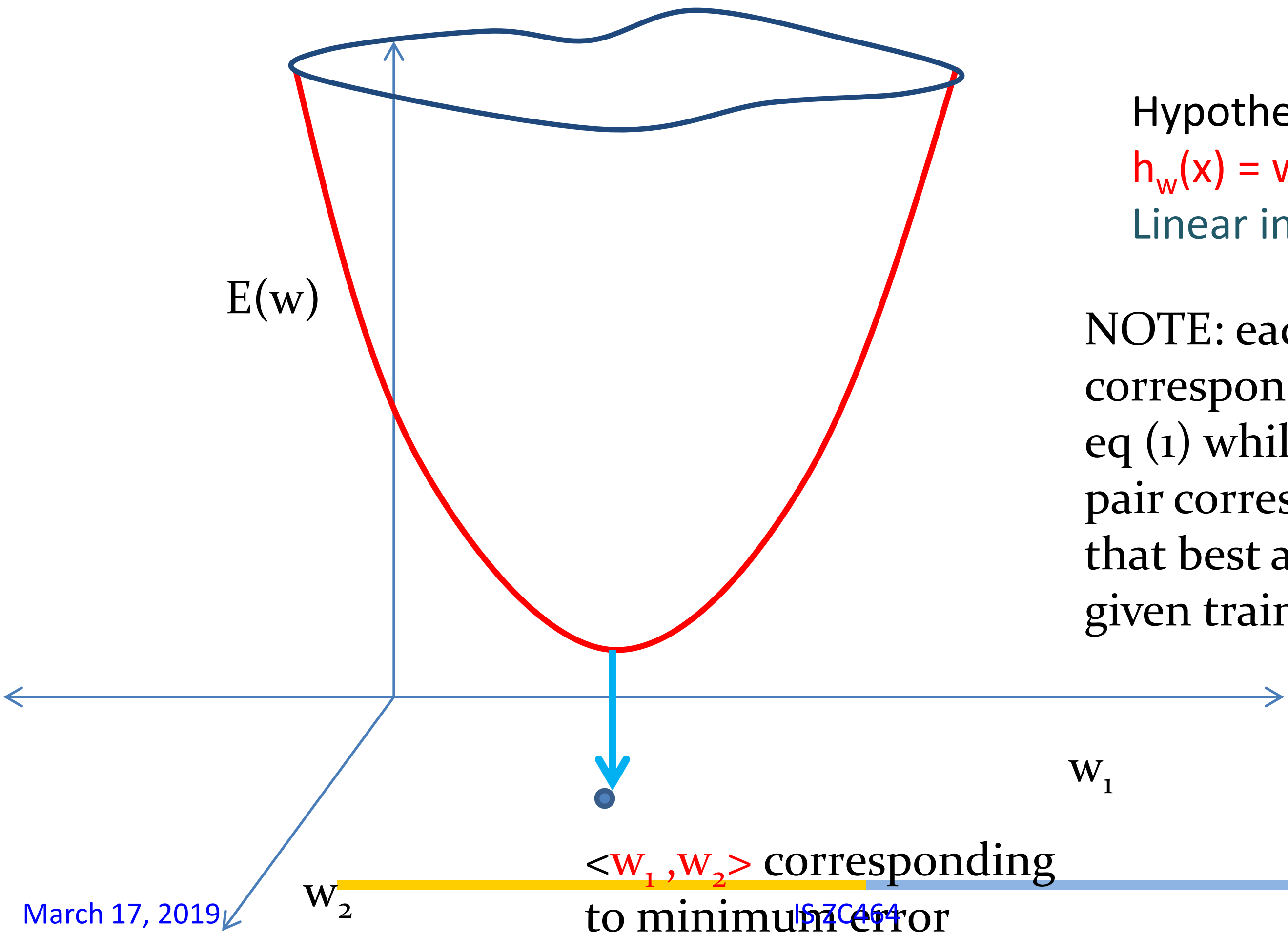


Hypothesis function

$h_w(x) = wx$

Linear in one variable

Plotting error when $y=f(x_1,x_2)$



Hypothesis function

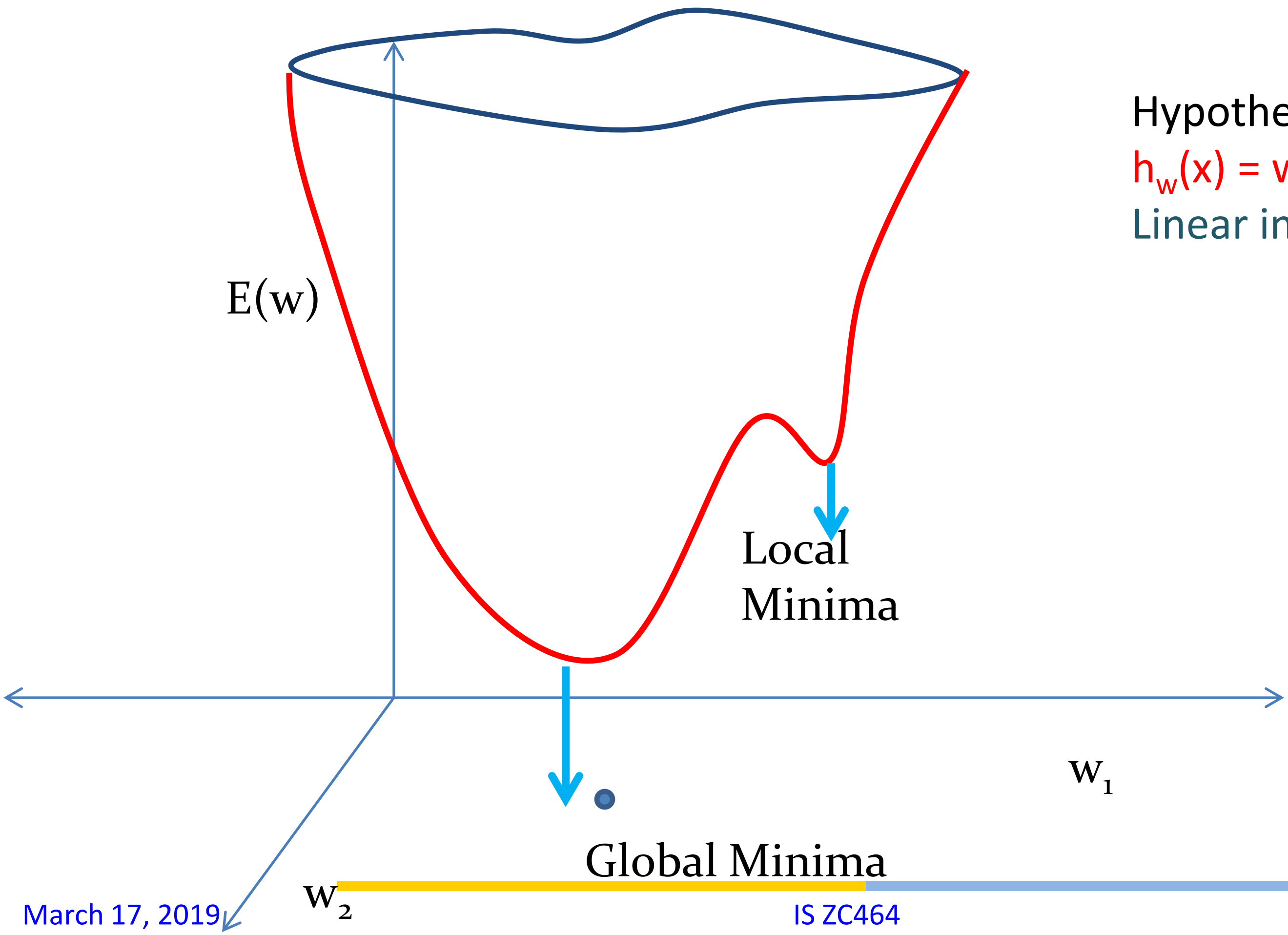
$h_w(x) = w_1x_1 + w_2x_2$ ....(1)

Linear in two variables

NOTE: each pair $< w_1, w_2>$ corresponds to a line given by eq (1) while only one such pair corresponds to the line that best approximates the given training data

Another possible surface

Plotting error when $y=f(x_1,x_2)$

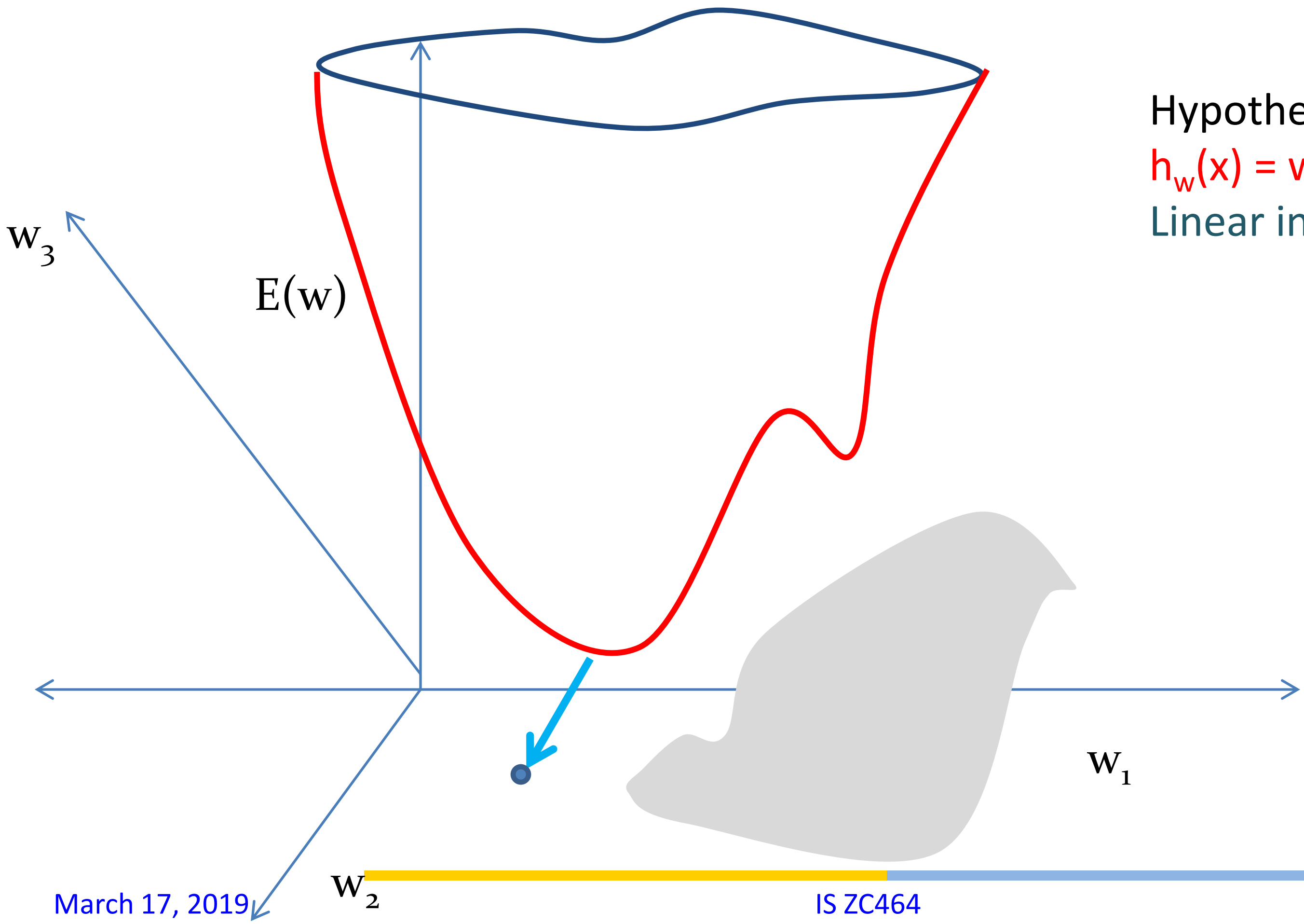


Hypothesis function

$h_w(x) = w_1x_1 + w_2x_2$ ....(1)

Linear in two variables

Difficult to visualize when $y=f(x_1, x_2, x_3)$

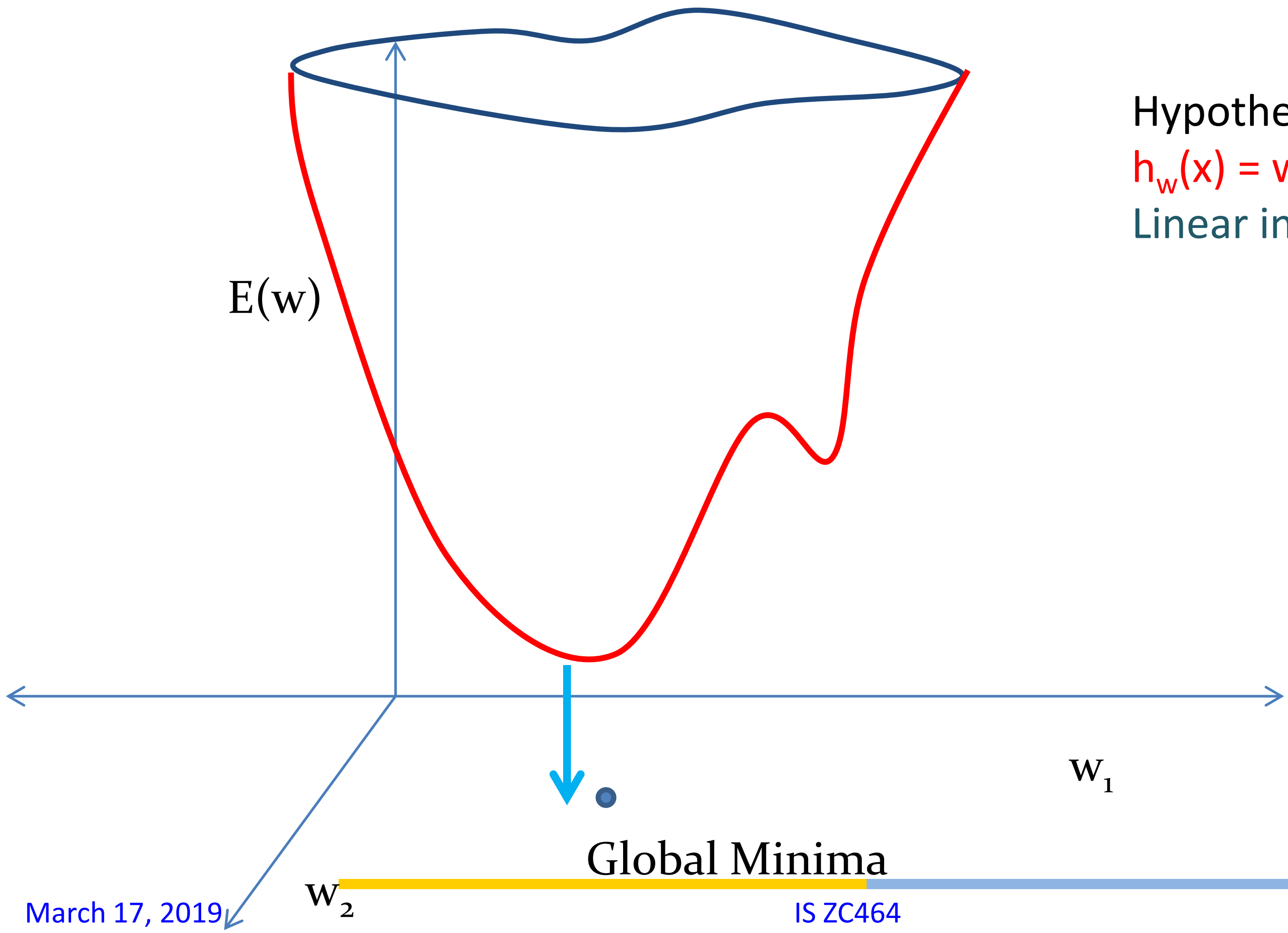


Hypothesis function

$h_w(x) = w_1x_1 + w_2x_2 + w_3x_3$

Linear in three variables

We will visualize in 2D but will extend the concept to n dimensions

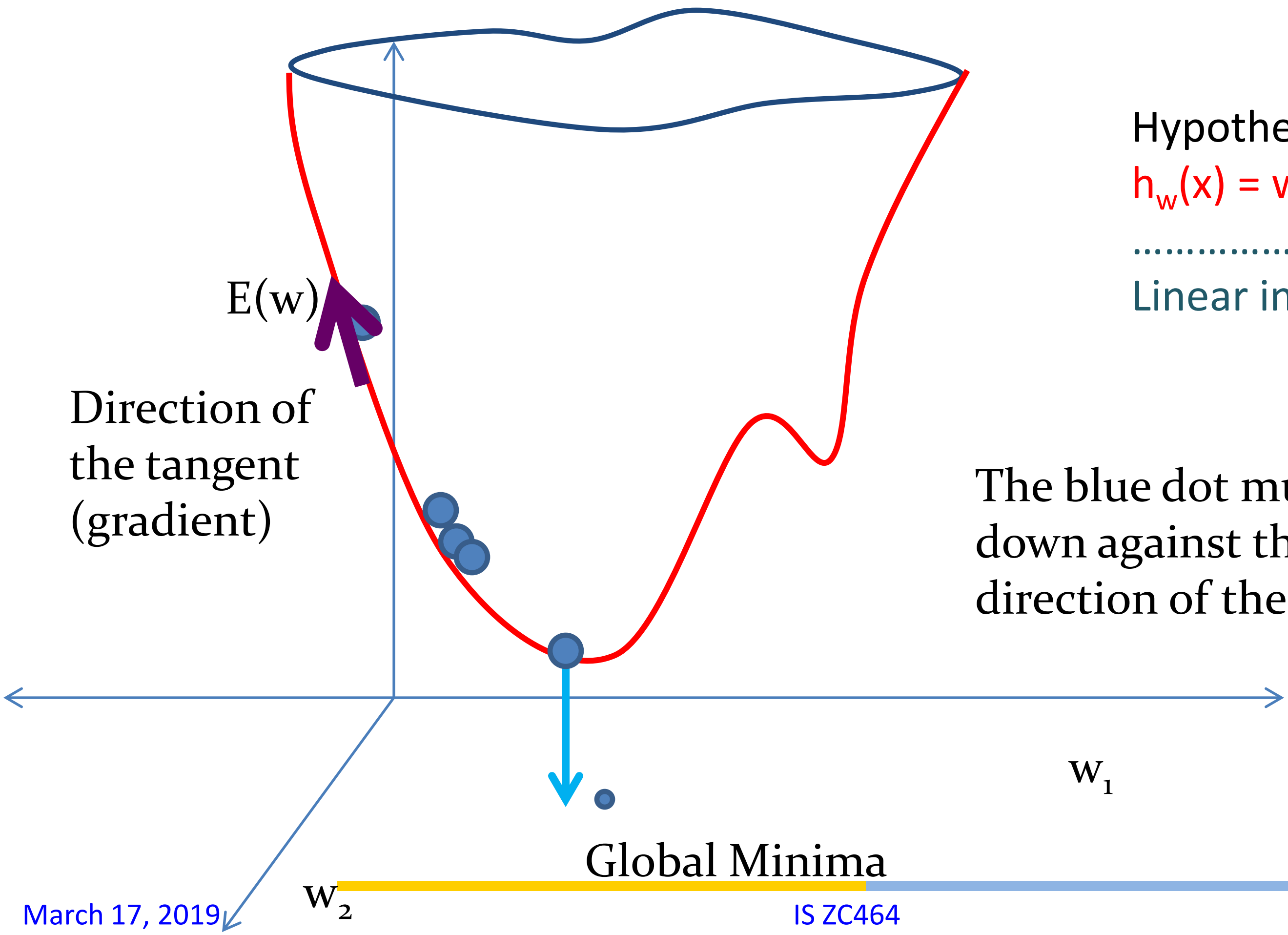


Hypothesis function

$h_w(x) = w_1x_1 + w_2x_2 + \dots w_nx_n$

Linear in n variables

Initial weight



Hypothesis function

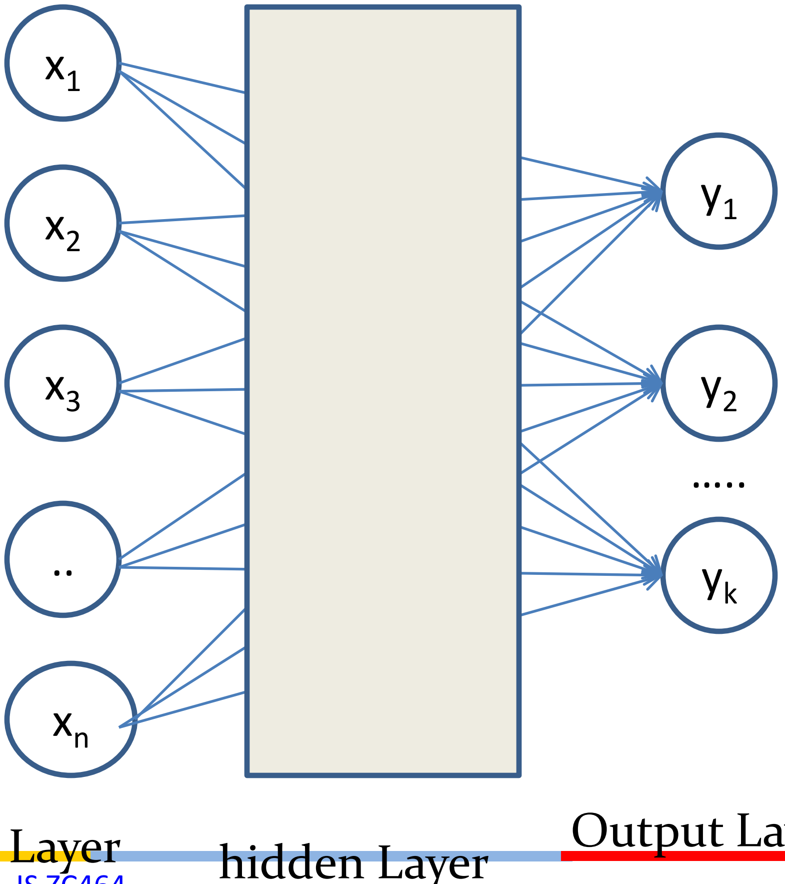
$h_w(x) = w_1x_1 + w_2x_2 + \dots w_nx_n$

……………………equation 2

Linear in n variables

Visualization of n-dimensional data

- $y_1 = f_1(x_1,x_2,\ldots..x_n)$
- $y_2 = f_2(x_1,x_2,\ldots..x_n)$
- $y_3 = f_3(x_1,x_2,\ldots..x_n)$
- $y_4 = f_4(x_1,x_2,\ldots..x_n)$
- …..
- $y_k = f_k(x_1,x_2,\ldots..x_n)$

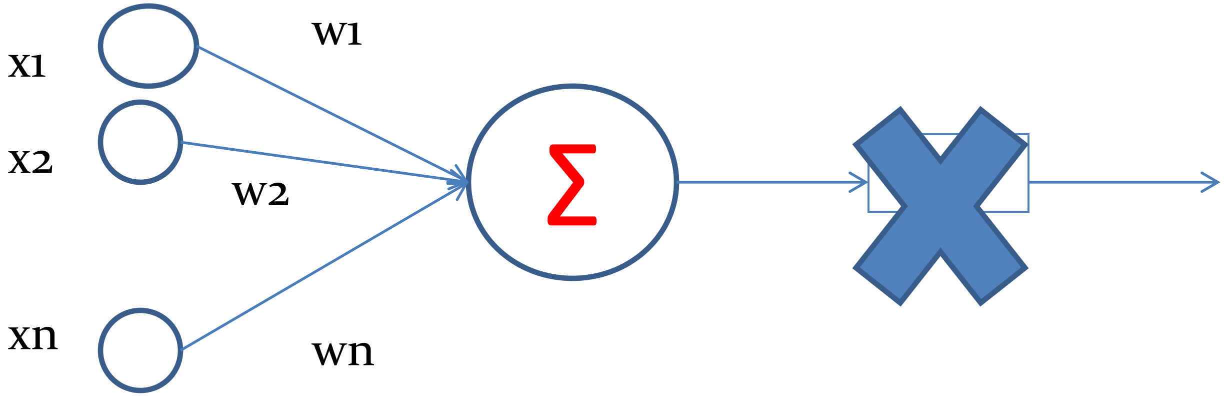


Computing gradient for the simple one perceptron model of neural network

Output of the neural network

$h_w(x) = w_1x_1 + w_2x_2 + \ldots. w_nx_n$

Human supervised output = y



Error = $y - h_w(x)$

Understanding symbols

- Assume that the feature vector is $(x_1,x_2,x_3,\ldots x_n)$
- There are m observations whose feature vectors are written as follows

$$(x^i_1, x^i_2, x^i_3, \ldots x^i_n)$$

For i = 1,2,3,…m

Note: Here the superscript 'i' represents the 'i'th observation and NOT the power of x.

- Let y^i be the output (human supervised)
- Let T_i be the error (note the use of subscript 'i' instead of superscript—That is just my way of representing)

Gradient Descent

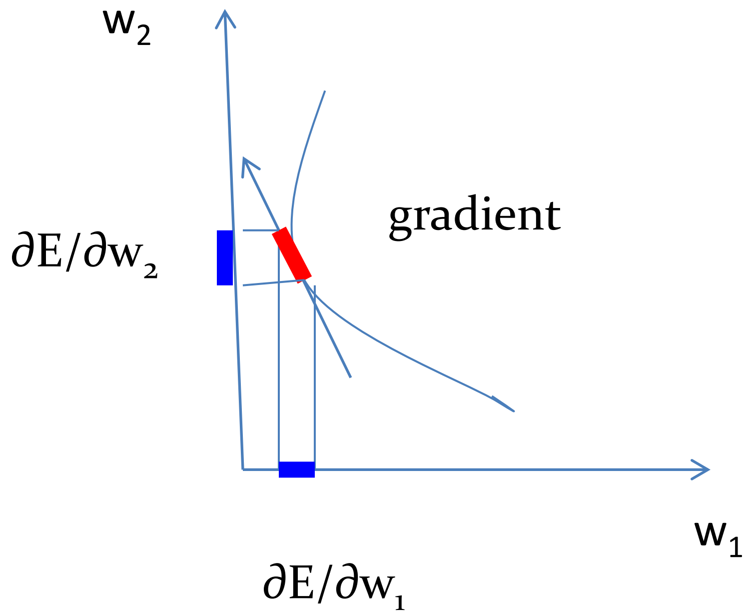
- This technique is used to reduce the squared error by calculating the partial derivative of E with respect to each weight.
- $E(w) = \sum_i T_i^2 = \sum_i (y^i - h_w(x^i))^2$

Replace $h_w(x)$ with the expression given in eq 2

$E(w) = (½)\sum_i (y - (w_1 x^i_1 + w_2 x^i_2 + \ldots w_n x^i_n))^2$

- Observe that E is a function of $w_1, w_2, \ldots w_n$
- Note that the effort is towards finding the equation of a line in n-dimensional space that best fits n dimensional data.
- Normalization with ½ is for computational convenience

Computing gradient



Computing Gradient

$E(w) = \sum_i T_i^2$

$= \sum_i (y^i - h_w(x^i))^2$

where T_i is the error term for the i^{th} observation and is given by the difference between the desired output (y^i) value and the estimated value ($h_w(x^i)$)of the output

$T_i = y^i - h_w(x^i)$

$h_w(x^i)$ is the hypothesis function given by

$h_w(x^i) = w_1 x^i_1 + w_2 x^i_2 + \ldots . w_n x^i_n$

Observe: E is a function of w.

Note: Here the superscript 'i' represents the 'i'th observation and NOT the power of x.

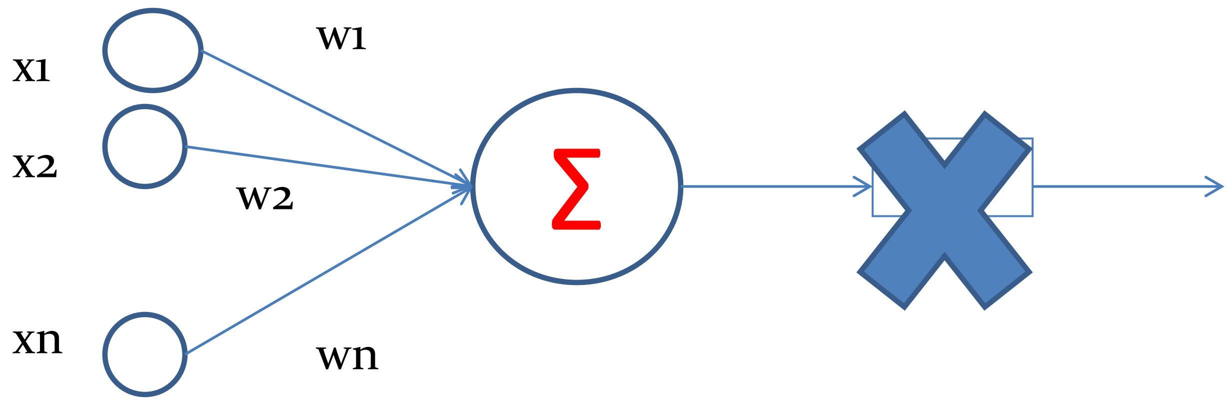
Computing gradient for the simple one perceptron model of neural network

Output of the neural network

$h_w(x) = g\ (w_1x_1\ + w_2x_2 + \ldots. w_nx_n\ )$

Where g is the activation function

Human supervised output $= y$



Error = $y - g(h_w(x))$

Computing Gradient

$E(w) \quad = \sum_i T_i^2$

$\quad\quad\quad = \sum_i (y^i - g(h_w(x^i)))^2$

where T_i is the error term for the i^{th} observation and is given by the difference between the desired output (y^i) value and the estimated value ($h_w(x^i)$)of the output

$$T_i = y^i - g(h_w(x^i))$$

$h_w(x^i)$ is the hypothesis function given by

$$h_w(x^i) = w_1 x^i_1 + w_2 x^i_2 + \ldots . w_n x^i_n$$

Observe: E is a function of w.

Note: Here the superscript 'i' represents the 'i'th observation and NOT the power of x.

Observe

- E is the function of T_i
- Ti is the function of g (assuming y as constant)
- g is the function of h
- h is the function of w

- Chain rule of Differentiation

$$\partial E/\partial w_k = \sum_i \partial E/\partial T_i * \partial T_i/\partial g * \partial g/\partial h * \partial h/\partial w_k$$

Equation 1

Observe

Since

$$E(w) = \sum_i T_i^2$$

$$\partial E/\partial T_i = 2{*}T_i$$

Chain rule of Differentiation

$$\partial E/\partial w_k = 2{*}\sum_i T_i * \partial T_i/\partial g * \partial g/\partial h * \partial h/\partial w_k$$

Also since

$$T_i = y^i - g(h_w(x^i))$$

Equation 2

Therefore

$$\partial T_i/\partial g = 0 - 1 = -1$$

Working with derivatives

Equation 2 now becomes

$$\partial E/\partial w_k = 2 * \sum_i T_i * (-1) * \partial g/\partial h * \partial h/\partial w_k$$

Also since

$$\partial g/\partial h = \partial\, g(h_w(x^i)) /\partial h = g'$$

And

$$h_w(x^i) = w_1 x^i_1 + w_2 x^i_2 + \dots w_n x^i_n$$

Therefore

$$\partial h/\partial w_k = x^i_k$$

Hence equation 3 is simplified as

$$\partial E/\partial w_k = -\, 2 * \sum_i T_i * g' * x^i_k$$

Equation 3

Equation 4

Computing gradient in the direction of w_k

- Substitute expression for T_i in equation 4

$$\partial E/\partial w_k = -\ 2 * \sum_i (y^i - g(h_w(x^i))) * g' * x^i_k$$

Equation 5

- The Weight update in the direction of w_k

$$\Delta w_k = -\ 2 * \sum_i (y^i - g(h_w(x^i))) * g' * x^i_k$$

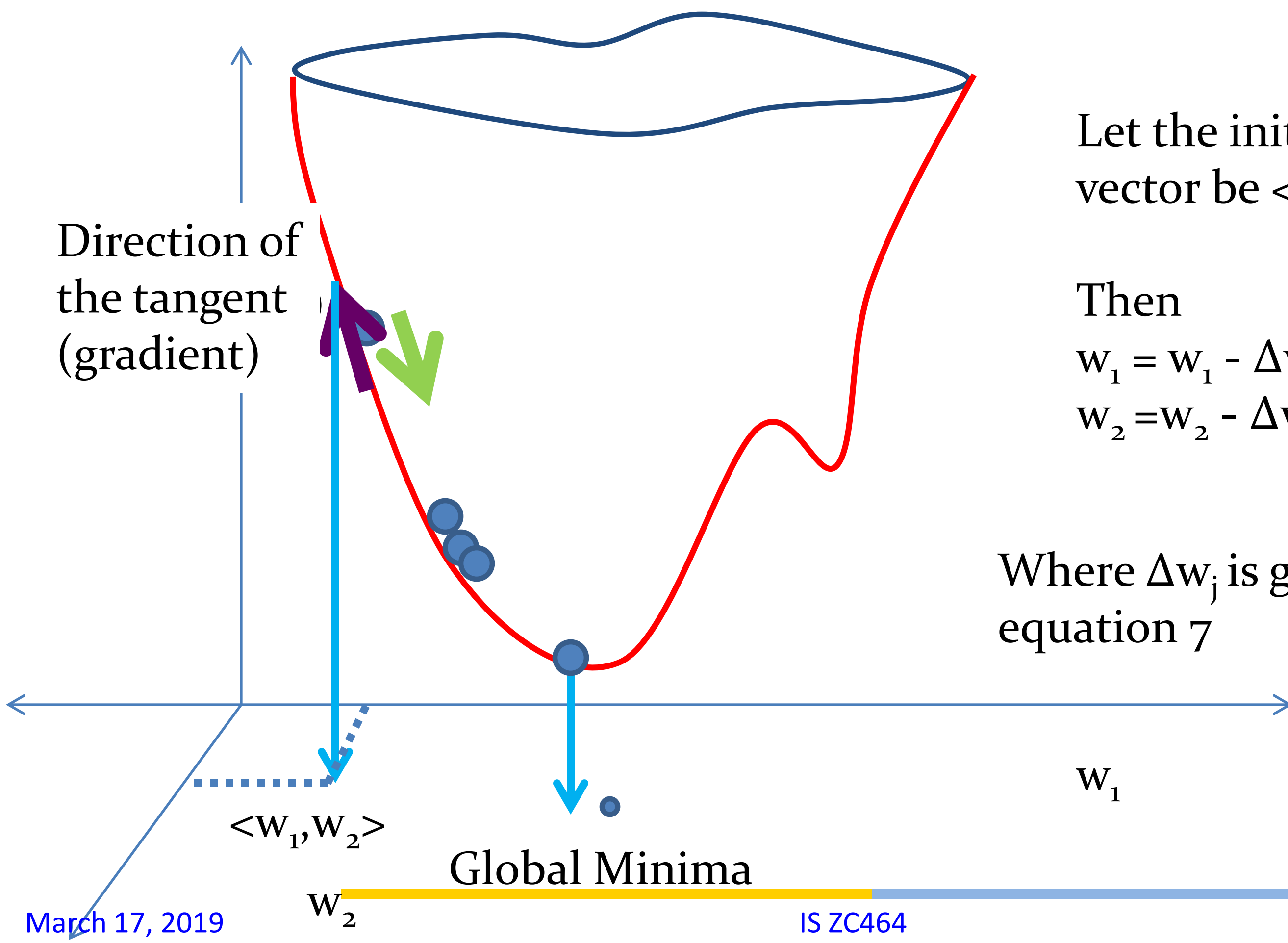
Equation 6

Where 2 can be dropped to bring normalization.

$$\Delta w_k = -\ \sum_i (y^i - g(h_w(x^i))) * g' * x^i_k$$

Equation 7

Delta Learning: Modification of the Initial weight



Let the initial weight vector be $<w_1,w_2>$

Then

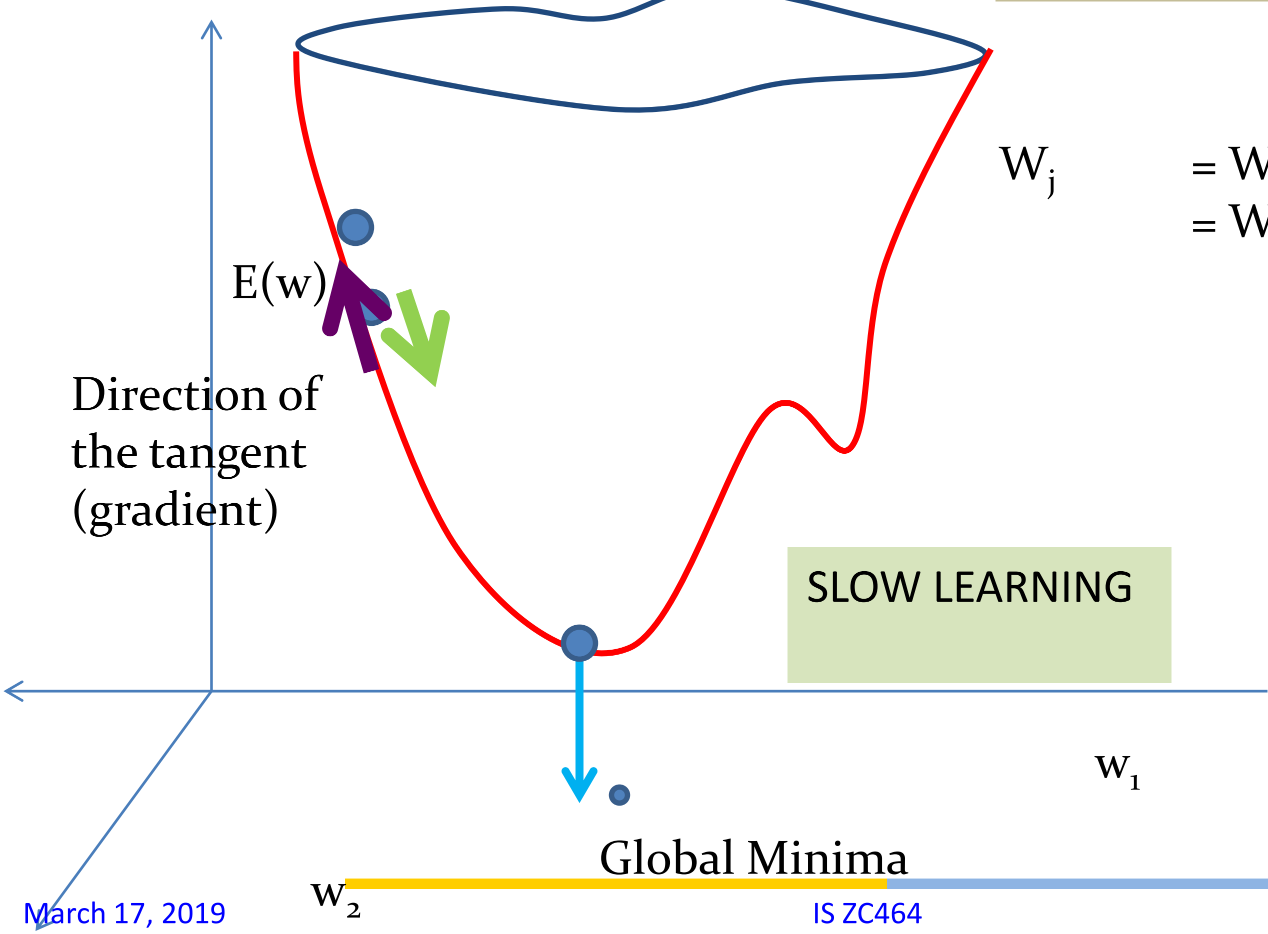
$w_1 = w_1 - \Delta w_1$

$w_2 = w_2 - \Delta w_2$

Where Δw_j is given by equation 7

Learning rate: fast or slow learning

$$W_j = W_j + \alpha * \sum_i (y^i - g(h_w(x^i)))_* g'_* x^i_k$$



$$W_j = W_j - \Delta w_j$$
$$= W_j + \sum_i T_{i*} g'_* x^i_k$$

α

α ranges from 0 to 1
If it is 0 , there is no learning
If it is large the learning rate is high

Multilayer Feed Forward neural network

- These represent the class of networks which approximate the complex functions.
- The network has one or more hidden layers.
- The neuron ‘i’ of layer ‘L’ is connected by a synaptic weight w_{ki} to the ‘k’th neuron of layer ‘L+1’

Weight Terminology