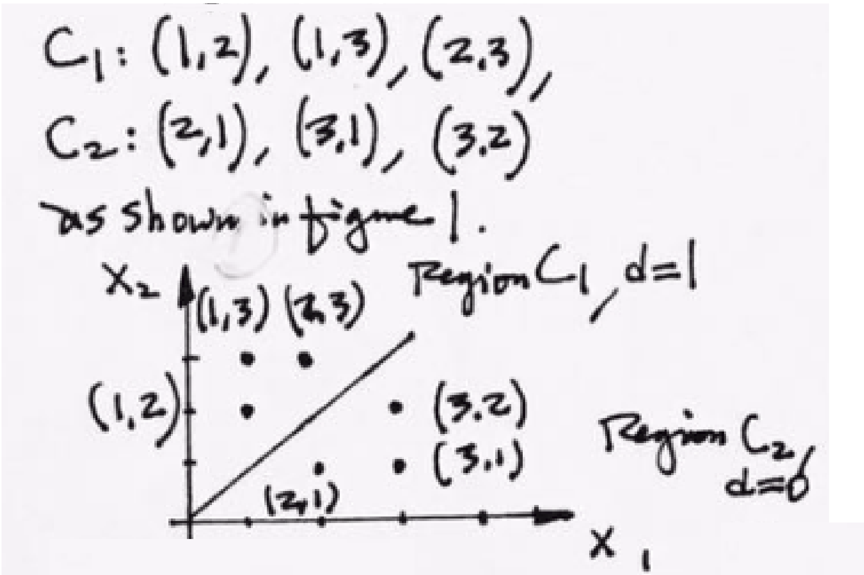
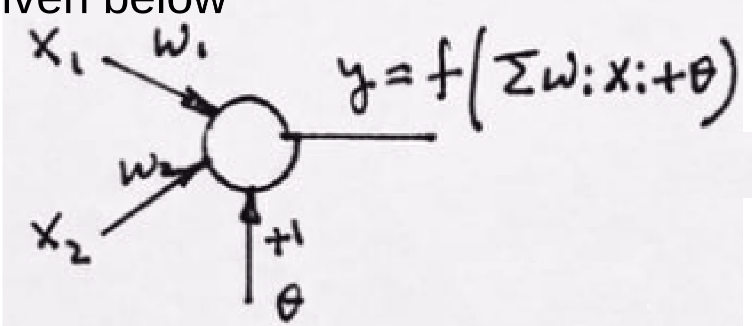
Example on Back Propagation (1)

Example: Given training data below



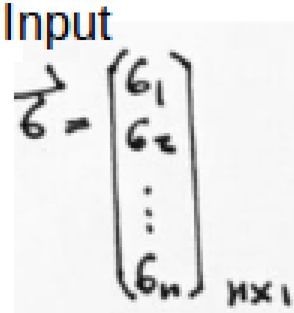
Suppose a single neuron network is given below



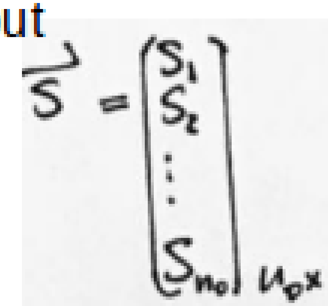
Now based on the notation in the class, see Figure 1. we have

1. input and output neurons

Input



Output



So we have input and output vector as follows

$$\vec{\sigma} = \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix} \quad \text{… (1)}$$

and the output vector

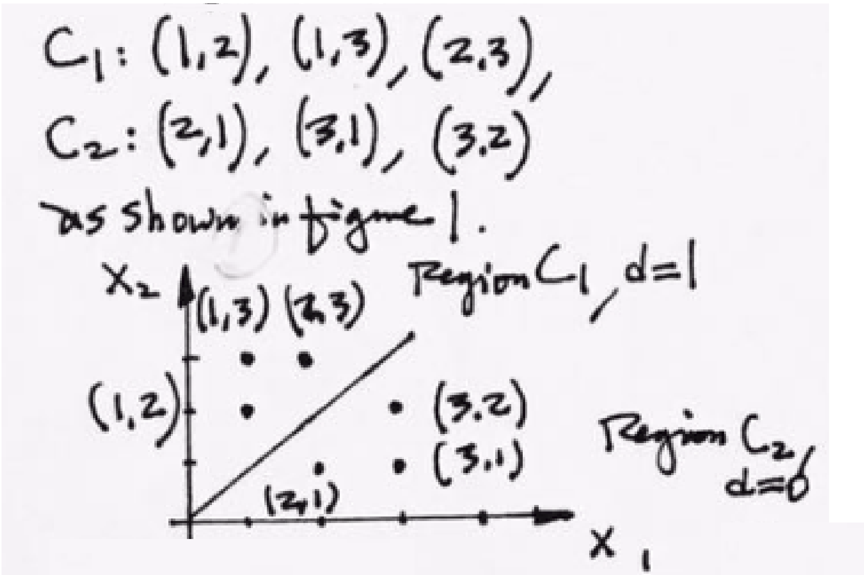
$$\vec{s} = (s_1) \quad \text{… (2)}$$

$$w_{i,k} \quad \text{… (3)}$$

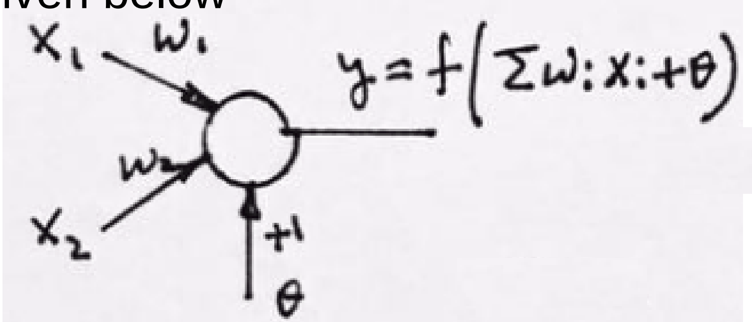
where i = 1, 2 for inputs and k =1 for the output

Example on Back Propagation (2)

Example: Given training data below



Suppose a single neuron network is given below



Now based on the notation in the class, see Figure 1. we have

Note: $(\sigma_1, \sigma_2) = (1, 2)$ for class C_1 for example.

Define activation function below

$$s_i = f(\sum_{i=1}^{2} w_{i,k} s_k + \phi) = f(\sum_{i=1}^{3} w_{i,k} s_k) \quad \ldots (4)$$

Note the summation upper bound is set to 2 for explicit offset phi, and the upper bound is changed to 3 when changed to generalized form by making s_3 = phi, and its weigh $w_{i,k}$ for i = 3 and k=1. So the notation is in the unified summation form.

Define transfer function as

$$h_i = \sum_{i=1}^{2} w_{i,k} s_k + \phi = \sum_{i=1}^{3} w_{i,k} s_k \quad \ldots (5)$$

Note subscript for h is index of the output i, in this example i = 1

Example on Back Propagation (3)

Define transfer function, in case of single layer NN, we use σ_k as input, and in case of multilayer NN, we use s_k as notation for the input from the previous output layer.

$$h_i = \sum_{i=1}^{2} w_{i,k}\sigma_k + \phi = \sum_{i=1}^{3} w_{i,k}\sigma_k \quad \text{...(5-1)}$$

So the neuron output i is

$$s_i = f(h_i) = f(h_i(w_{i,k})) \quad \text{… (6)}$$

Define an error as follows

$$\zeta_i{}^{\mu} - s_i{}^{\mu} \quad \text{… (7)}$$

note where $\zeta_i{}^{\mu}$ is desired (correct) output i at experiment μ, and $s_i{}^{\mu}$ is the actual output i at experiment μ.

Let's take a look at the hand calculation example, from the reference copied on the right, where desired output is d and actual output is y, and since in this hand calculation example it is only have one output, so i = 1 in eqn(7), and

4) Training algorithm

$$w_i^{+} = w_i^{-} + \eta(d-y)x_i \quad \text{...(2)}$$

where η is gain.

d : desired the output.

y: actual output.

The experiment μ in equation (7) refers to as the input data (1,2) from C1.

Example on Back Propagation (4)

5) Now, using data from C_1 and C_2 to perform training.

Choose (1, 2) from C_1.

$$w_1^+ = w_1^- + \eta(1-y)x_1 \Big|_{x_1=1}$$

$$= 0.5 + \eta(1-y)\cdot 1$$

where

$$y = f\left(\sum_{i=1}^{2} w_i x_i + \theta\right)$$

$$= f(w_1 x_1 + w_2 x_2 + \theta)$$

$$= f(0.5\times 1 + 0.5\times 2 + 0.5) = f(2)$$

$$= \mathrm{sgn}(2) = 1$$

hence, $w_1^+ = w_1^- + \eta(1-1)x_1 = w_1^- = 0.5$

See from hand calculation example,
The experiment μ_i in equation (7) refers to as the input data (1,2) from C1 as shown on the left.

Example on Back Propagation (5)

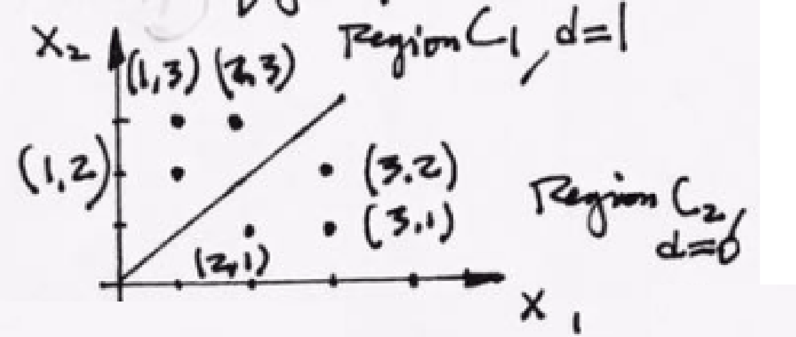
Definition 6. *Define total error for all neuron outputs and for all experiments*

$$D = \frac{1}{2} \sum_{\mu=1}^{6} \sum_{i=1}^{1} ({\zeta_i}^{\mu} - {s_i}^{\mu})^2 \qquad (8)$$

note where ${\zeta_i}^{\mu}$ for i equal to 1, for only 1 output neuron, from the hand calculation example, and and experiment μ is equal to 6 due to total number of 6 feature points, e.g., 6 experiments.

C_1: (1,2), (1,3), (2,3),
C_2: (2,1), (3,1), (3,2)

as shown in figure 1.



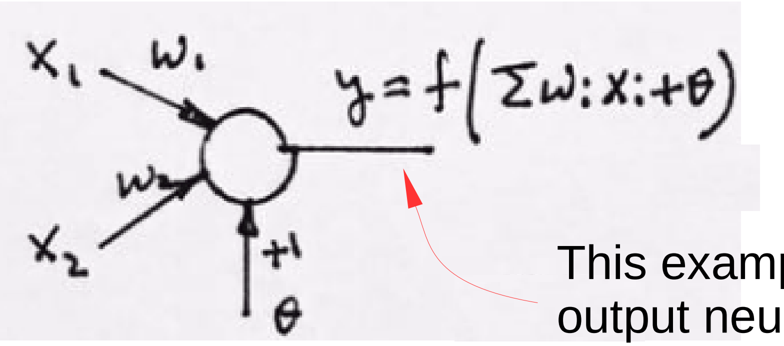
Example on Back Propagation (6)

Definition 7. *Minimize error function*

$$\frac{\partial D}{\partial w_{i,k}} = \frac{\partial}{\partial w_{i,k}} \frac{1}{2} \sum_{\mu=1}^{6} \sum_{i=1}^{1} ({\zeta_i}^{\mu} - {s_i}^{\mu})^2 = \sum_{\mu=1}^{6} ({\zeta_i}^{\mu} - {s_i}^{\mu}) f'({h_i}^{\mu}) \frac{\partial h_i}{\partial w_{i,k}} \quad \ldots (9)$$

So for the hand calculation example, we have 1 output, and input feature vector is 2 dimension,

$$\frac{\partial D}{\partial w_{1,k}} = \frac{\partial}{\partial w_{1,k}} \frac{1}{2} \sum_{\mu=1}^{6} \sum_{i=1}^{1} ({\zeta_1}^{\mu} - {s_1}^{\mu})^2 = \sum_{\mu=1}^{6} ({\zeta_1}^{\mu} - {s_1}^{\mu}) f'({h_1}^{\mu}) \frac{\partial h_1}{\partial w_{1,k}} \quad \ldots (10)$$



This example has 1 output neuron

Example on Back Propagation (7)

Note the derivative of the activation function

$$f'(h_1{}^{\mu}) \qquad (11)$$

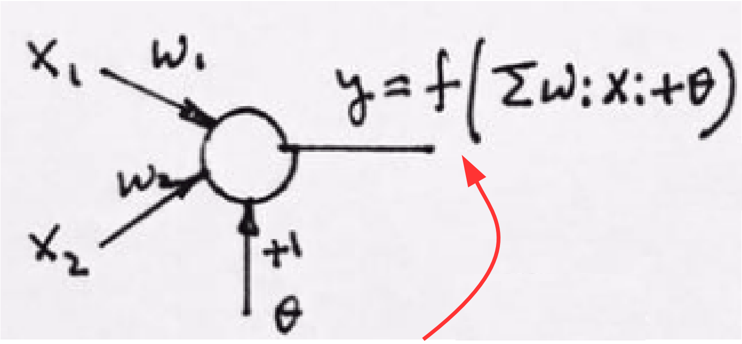
Suppose we choose RELU, then compute its derivative as

One Relu example

$$\boxed{f(x) = \ln(1 + e^x)} \qquad \text{… (12)}$$

Its derivative:

$$f'(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$



Activation function can be chosen as RELU for example

Example on Back Propagation (8)

Definition 8. *Learning by updating the weights*

$$w_{i,k}(t+1) = w_{i,k}(t) + \delta w_{i,k}(t) \tag{12}$$

where

$$\delta w_{i,k}(t) = -\epsilon \frac{\partial D}{\partial w_{i,k}} \tag{13}$$

for the given example, we have

$$\underline{w_{1,k}(t+1) = w_{1,k}(t) + \delta w_{1,k}(t)} \tag{14}$$

where

$$\delta w_{1,k}(t) = -\epsilon \frac{\partial D}{\partial w_{1,k}} \tag{15}$$

for k = 1, 2, see example feature vector dimension is 2

Note equation (14) now is employed to replace hand calculation equation (2)

From hand calculation example

4) Training algorithm

$$w_i^+ = w_i^- + \eta (d - y) x_i \quad \cdots (2)$$

where η is gain.

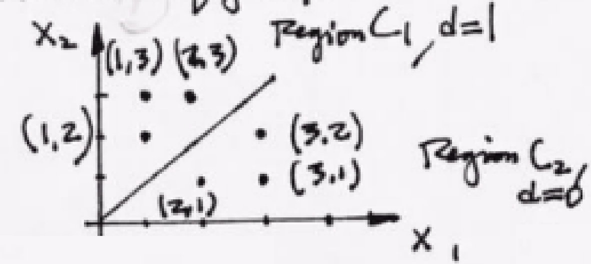
d : desired the output.

y: actual output.

C_1: (1,2), (1,3), (2,3),

C_2: (2,1), (3,1), (3,2)

as shown in figure 1.



Harry Li, Ph.D.

Example on Back Propagation (9)

So the gradient is defined as follows for the given example

$$\begin{pmatrix} \frac{\partial D}{\partial w_{i,1}} \\ \frac{\partial D}{\partial w_{i,2}} \end{pmatrix} = \begin{pmatrix} ({\zeta_i}^{\mu} - {s_i}^{\mu}) f'({h_i}^{\mu}) \frac{\partial h_i}{\partial w_{i,1}} \\ ({\zeta_i}^{\mu} - {s_i}^{\mu}) f'({h_i}^{\mu}) \frac{\partial h_i}{\partial w_{i,2}} \end{pmatrix} \qquad (16)$$

Since in our example, we have only 1 output, so

$$\begin{pmatrix} \frac{\partial D}{\partial w_{1,1}} \\ \frac{\partial D}{\partial w_{1,2}} \end{pmatrix} = \begin{pmatrix} ({\zeta_1}^{\mu} - {s_1}^{\mu}) f'({h_1}^{\mu}) \frac{\partial h_1}{\partial w_{1,1}} \\ ({\zeta_1}^{\mu} - {s_1}^{\mu}) f'({h_1}^{\mu}) \frac{\partial h_1}{\partial w_{1,2}} \end{pmatrix} \qquad (17)$$

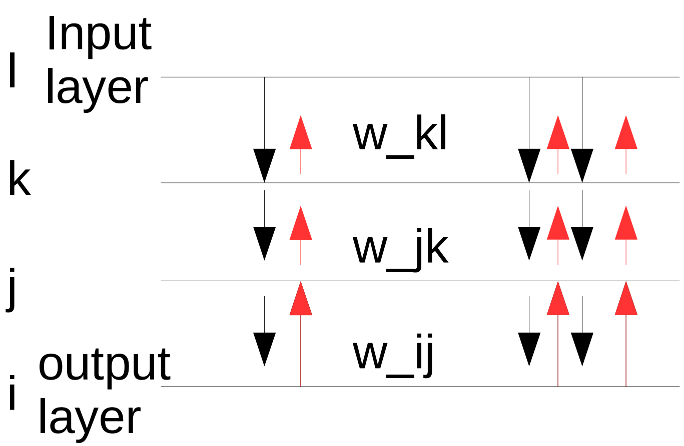
Hence, the following gradient for n inputs is now replaced by equation (17)

$$\begin{bmatrix} \frac{\partial D}{\partial w_{i1}} \\ \frac{\partial D}{\partial w_{i2}} \\ \vdots \\ \frac{\partial D}{\partial w_{in}} \end{bmatrix} = \begin{bmatrix} [\zeta_i^{\mu} - f(h_i^{\mu})] f' \frac{\partial h_i^{\mu}}{\partial w_{i1}} \\ [\zeta_i^{\mu} - f(h_i^{\mu})] f' \frac{\partial h_i^{\mu}}{\partial w_{i2}} \\ \vdots \\ [\zeta_i^{\mu} - f(h_i^{\mu})] f' \frac{\partial h_i^{\mu}}{\partial w_{in}} \end{bmatrix}$$

Example on Back Propagation (10)

Now generalize the discussion, we can extend the mathematical formulation to multiple layers, for example in this lecture, we have 4 layers

10. feed forward NN (4 layers)



Black for the input; red for the back prop training direction

Update sequence back to the front:

w_ij → w_jk → w_kl

The state of the output functions for the layers i (output) and j input:

$$S_i = f(h_i) \qquad h_i = \sum_j w_{ij} S_j - \theta_i$$

Back Propagation (1)

1. input and output neurons

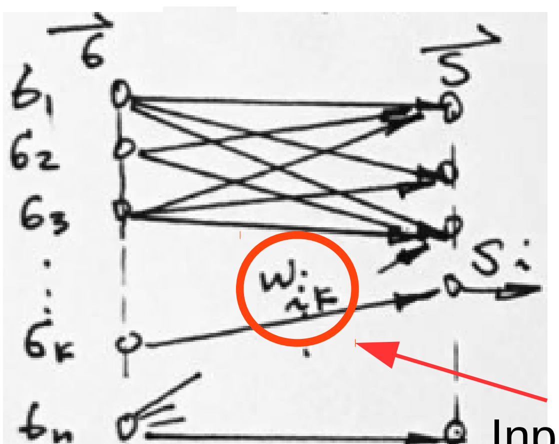
Input

$$\vec{\sigma} = \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_n \end{bmatrix}_{n \times 1}$$

Output

$$\vec{S} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{n_o} \end{bmatrix}_{n_o \times 1}$$

The architecture



Input: k
output: i

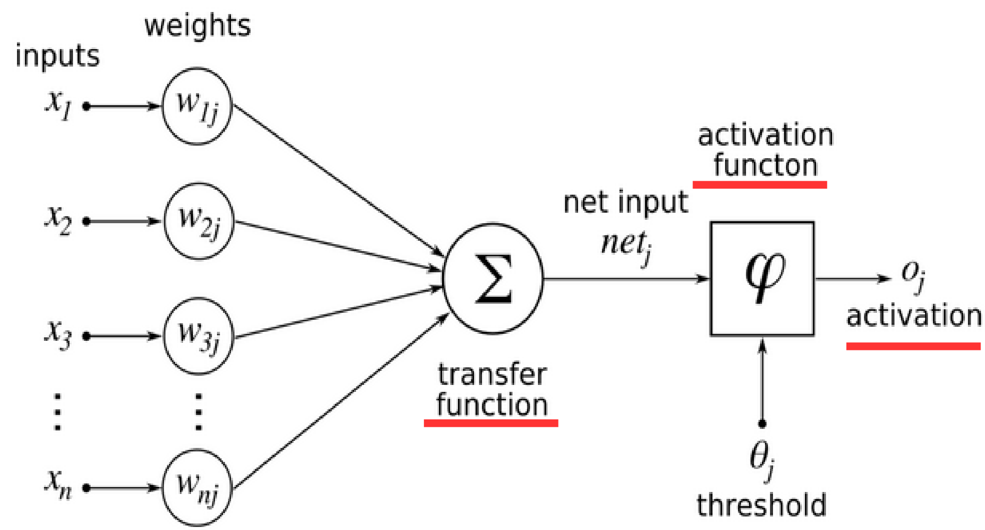
2. weights w_ik

i for output

k for input

Other popular notation

$$Y = \sum (weight * input) + bias$$



3. Activation function

$$\boxed{S_i = f\left(\sum_k \omega_{ik} \sigma_k\right)} \quad \ldots (1)$$

Note the activation function f(.) and the bias (offset can be written in the unified summation term)

Harry Li, Ph.D.

Back Propagation (2)

4. Transfer function h_i

let

$$h_i = \sum_k w_{ik} \sigma_k \quad \text{... (2)}$$

Index i for the output neuron

Neuron output function S_i

$$S_i = f(h_i) = f[h_i(w_{ik})] \quad \text{... (2-1)}$$

Neuron output is tied to activation function f(.), transfer function h_i and weights w_ik

5. Error at each neuron output (the difference between the true output Zeta (desired true output) and the current output S_i) at the experiment Mu

$$\zeta_i^\mu - S_i^\mu \quad \text{... (2-2)}$$

6. total error for all output neuron i and all experiments Mu

$$D = \frac{1}{2} \sum_\mu \sum_i (\zeta_i^\mu - S_i^\mu)^2 \quad \text{... (3)}$$

7. minimize the error wrt to w_ik

$$\frac{\partial D}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}} \cdot \frac{1}{2} \sum_\mu \sum_i [\zeta_i^\mu - f(h_i^\mu)]^2$$

$$= \sum_\mu [\zeta_i^\mu - f(h_i^\mu)] f'(h_i^\mu) \frac{\partial h_i}{\partial w_{ik}}$$

8. Training the NN by updating the weights

$$w_{ik}(t+1) = w_{ik}(t) + \delta w_{ij}(t) \quad \text{...(4)}$$

where

$$\delta w_{ij}(t) = -\epsilon \frac{\partial D}{\partial w_{ij}} \quad \text{...(5)}$$

Back Propagation (3)

9. computation of the derivatives to update the weight

$$\begin{bmatrix} \frac{\partial D}{\partial w_{i1}} \\ \frac{\partial D}{\partial w_{i2}} \\ \vdots \\ \frac{\partial D}{\partial w_{in}} \end{bmatrix} = \begin{bmatrix} [\zeta_i^{\mu} - f(h_i^{\mu})] f' \frac{\partial h_i^{\mu}}{\partial w_{i1}} \\ [\zeta_i^{\mu} - f(h_i^{\mu})] f' \frac{\partial h_i^{\mu}}{\partial w_{i2}} \\ \vdots \\ [\zeta_i^{\mu} - f(h_i^{\mu})] f' \frac{\partial h_i^{\mu}}{\partial w_{in}} \end{bmatrix}$$

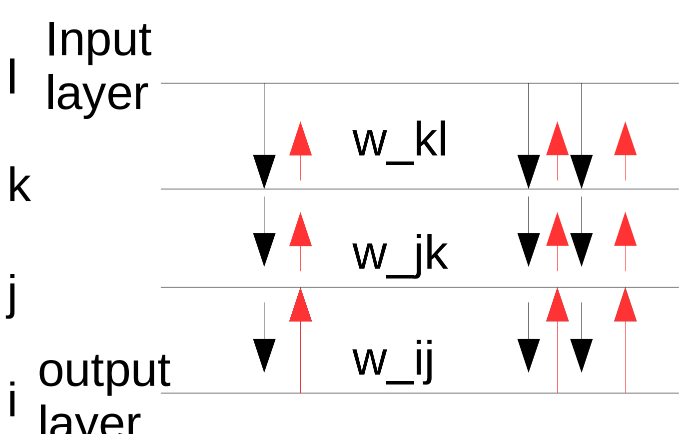
Update sequence back to the front:

w_ij w_jk w_kl

The state of the output functions for the layers i (output) and j input:

$S_i = f(h_i)$ $h_i = \sum_j w_{ij} S_j - \theta_i$

10. feed forward NN (4 layers)



Black for the input; red for the back prop training direction

Back Propagation (4)

11. chain rule, updating the weights (training)

For layer i

$$\frac{\partial D}{\partial w_{ij}} = \sum_{\mu} \left[\zeta_i^{\mu} - f(h_i^{\mu}) \right] f'(h_i^{\mu}) \frac{\partial h_i^{\mu}}{\partial w_{ij}}$$

For layer j

$$\frac{\partial D}{\partial w_{jk}} = \sum_{\mu} \sum_{i} \left[\zeta_i^{\mu} - f(h_i^{\mu}) \right] f'(h_i^{\mu}) \frac{\partial h_i^{\mu}}{\partial S_j} \cdot \frac{\partial S_j}{\partial w_{jk}}$$

For layer k

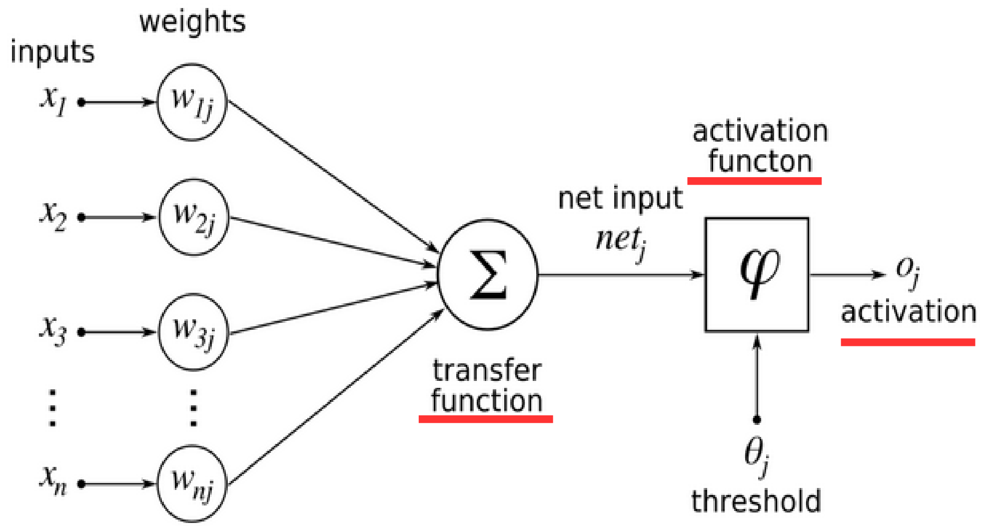
$$\frac{\partial D}{\partial w_{kl}} = \sum_{\mu} \sum_{i} \sum_{j} \left[\zeta_i^{\mu} - f(h_i^{\mu}) \right] f'(h_i^{\mu}) \frac{\partial h_i^{\mu}}{\partial S_j} \cdot \frac{\partial S_j}{\partial S_k} \cdot \frac{\partial S_k}{\partial w_{kl}}$$

Note: the training described here all related to the derivative to the activation function, f'(.). So selection of the activation function is important and will be discussed in details next.

Activation Functions

Definition: for single neuron for the purpose of generating the output of the neuron.

$$Y = \sum (weight * input) + bias$$



Type: over hundreds proposed, we will focus On the following 4 types, Sigmoid, Softmax, Tanh, ReLU, (SSTR)

Characteristics and Comparison:

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Tanh

$$tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$

ReLU

$$f(x) = max(0, x)$$

Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, j = 1, 2, \ldots, K$$

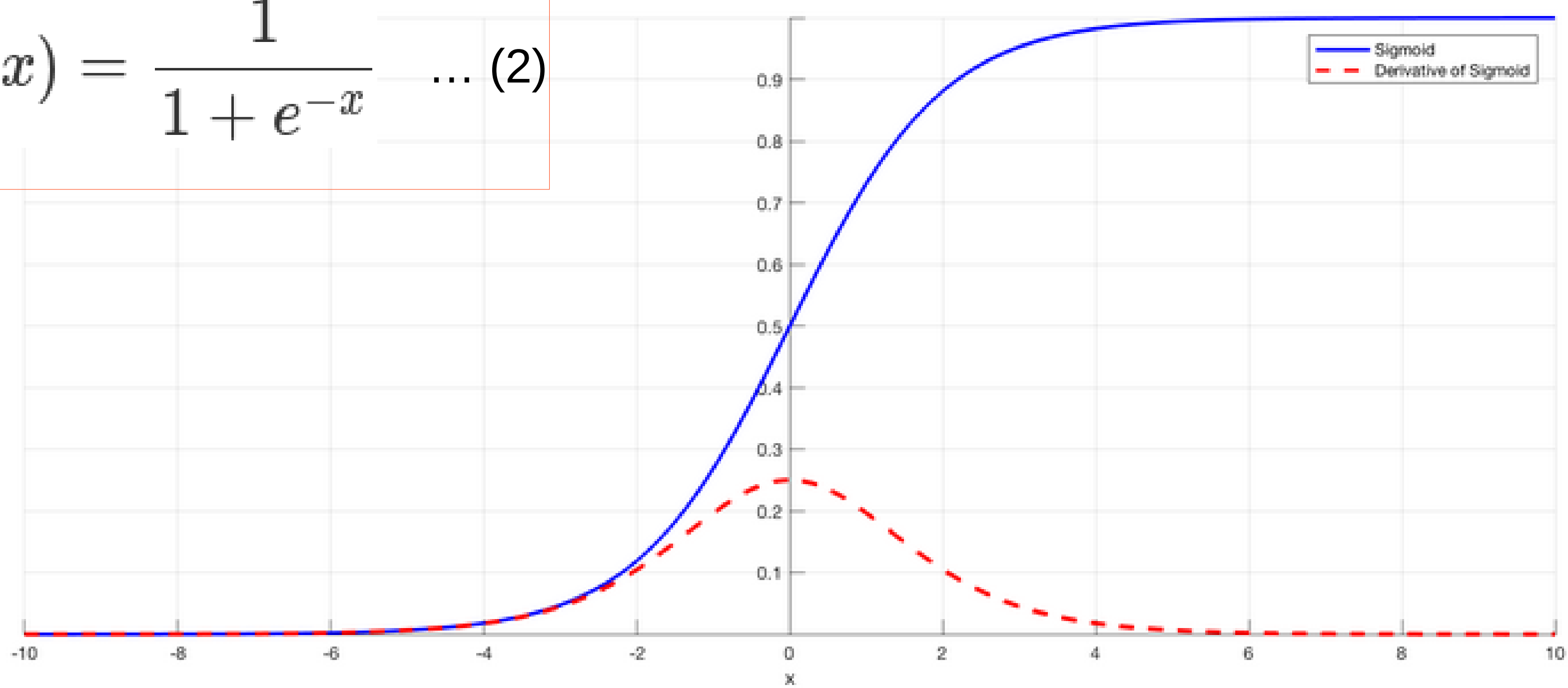
Sigmoid Functions

https://isaacchanghau.github.io/post/activation_functions/

$$\boxed{\sigma(x) = \frac{L}{1 + e^{-k(x-x_0)}}} \quad \ldots (1)$$

Logistic function in general as in equation (1)

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \ldots (2)$$

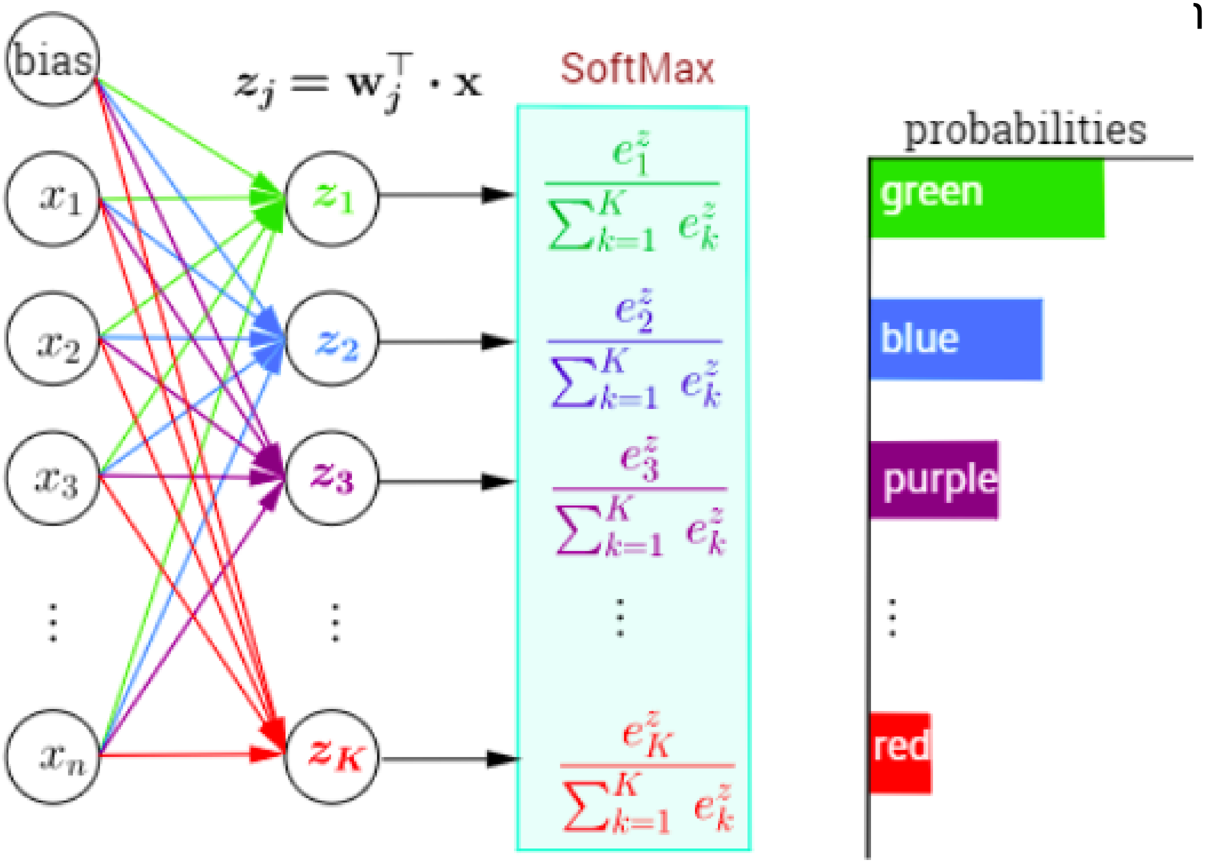


Softmax Functions

https://isaacchanghau.github.io/post/activation_functions/

$$\boxed{\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, j = 1, 2, \ldots, K} \quad \ldots (1)$$

used in various multiclass classification methods, such as multinomial logistic regression, multiclass linear discriminant analysis, naive Bayes classifiers, ıd artificial neural networks.



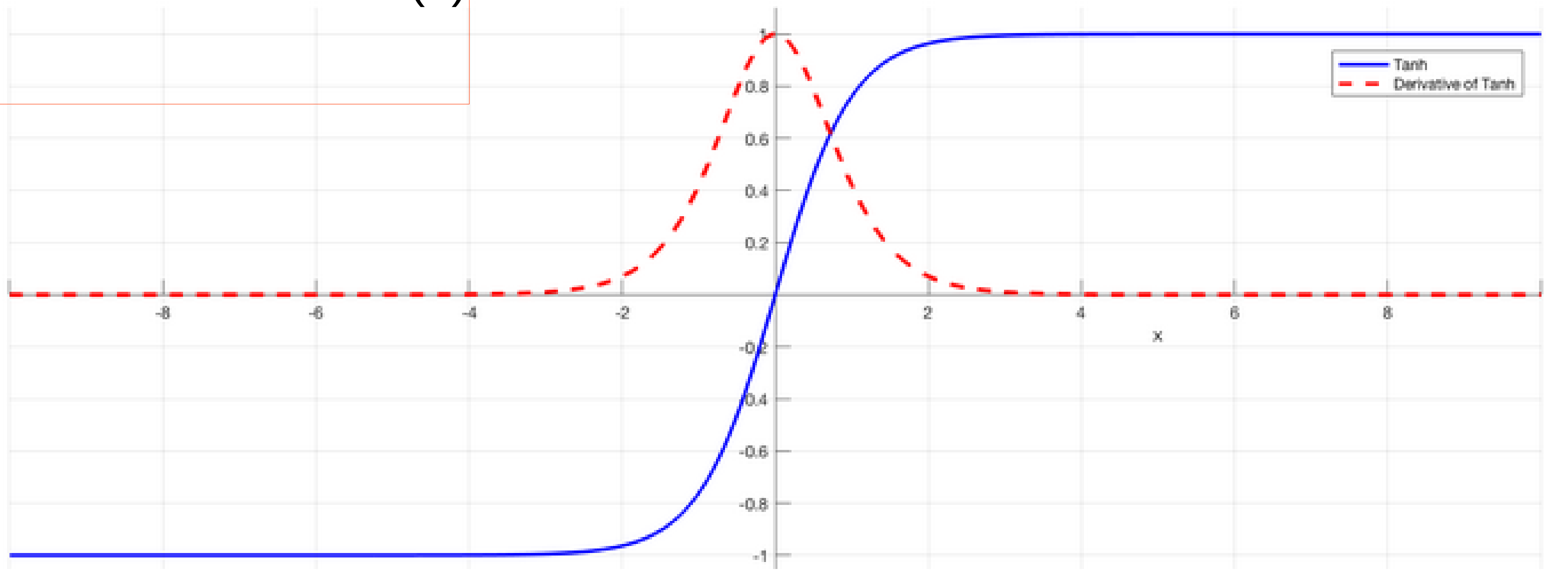
tanh Functions

https://isaacchanghau.github.io/post/activation_functions/

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad \text{… (1)}$$

The derivative (red curve)

… (2)



Relu Functions

https://isaacchanghau.github.io/post/activation_functions/

$$f(x) = max(0, x)$$ … (1)

One Relu example (green)

$$f(x) = \ln(1 + e^x)$$ … (2)

Its derivative:

$$f'(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$