CS772: Deep Learning for Natural Language Processing (DL-NLP)

Neural Network Weight change rules

Pushpak Bhattacharyya
Computer Science and Engineering Department
IIT Bombay

Week 3 of 15th Jan, 2024

1-slide recap

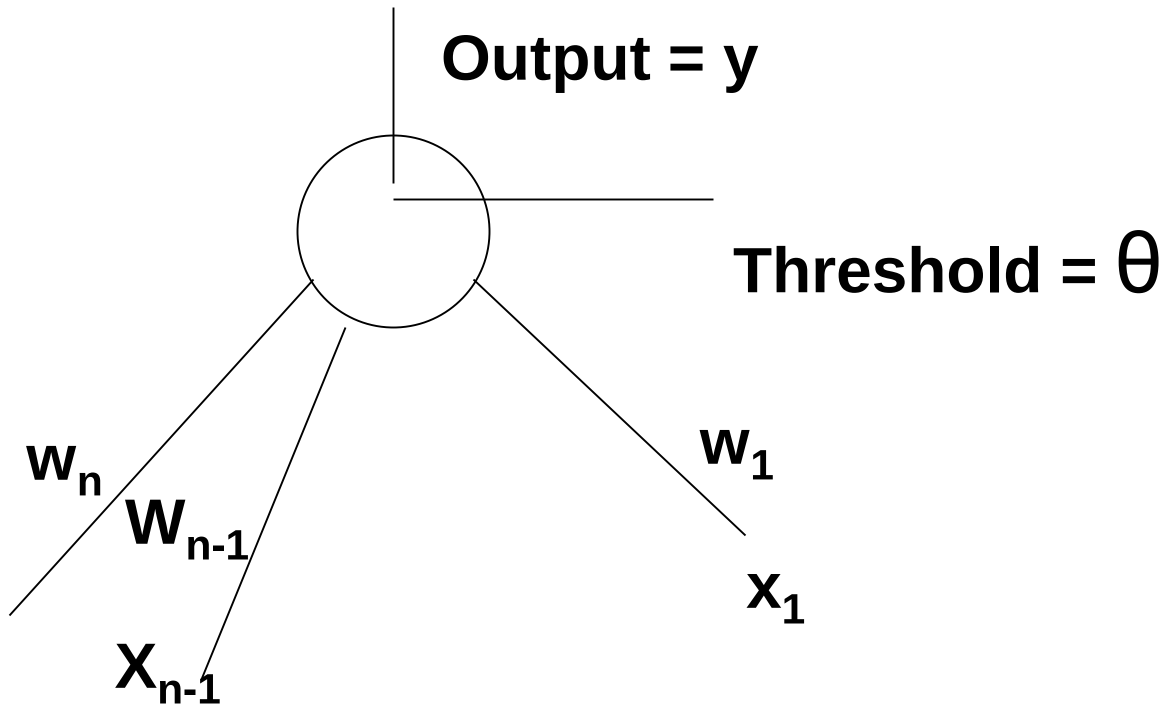
- Perceptrons: only $O(2^{N^2})$ out of $O(2^{2^N})$ Boolean Functions are threshold; XOR not computable
- PTA: find W s.t. $W.Xi>0$, for all i; $Wnext=W+Xfail$
- PTA guaranteed to converge if vectors are from linearly separable function

  Proof by contradiction: $G(w_n)=(W_n.W^*)/|W_n|$
- Sigmoid and Softmax functions and their derivatives: (a) sigmoid $O(1-O)$; (b) Softmax- $O_k(1-O_k)$ and $-O_kO_{k'}$

Recurrent Perceptron
(motivating RNN)

The Perceptron Model

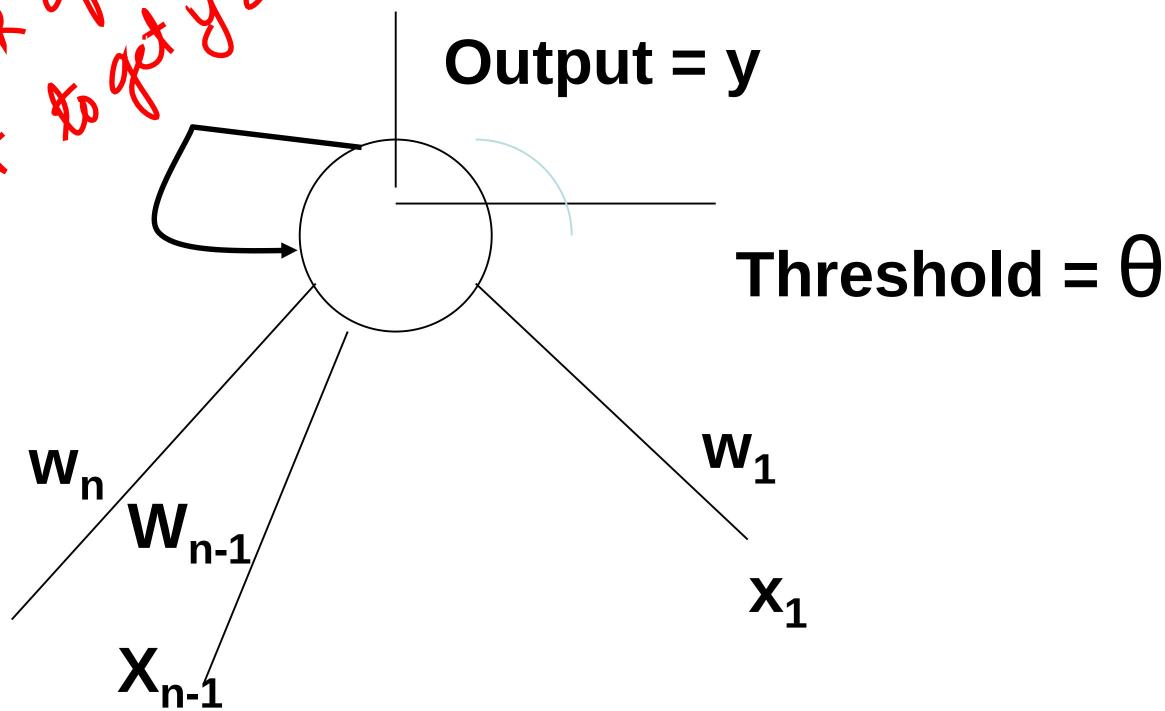
- A perceptron is a computing element with input lines having associated weights and the cell having a threshold value. The perceptron model is motivated by the biological neuron.



The Recurrent Perceptron

Output is fed back

So, the first input to the neuron can be considered as the feed forward block of the input excluding that recurrent input to get y & then we proceed.



The "Identity" perceptron with f/b: Recurrent Identity-Perceptron (IdP)

Output is fed back

For Identity-Perceptron,

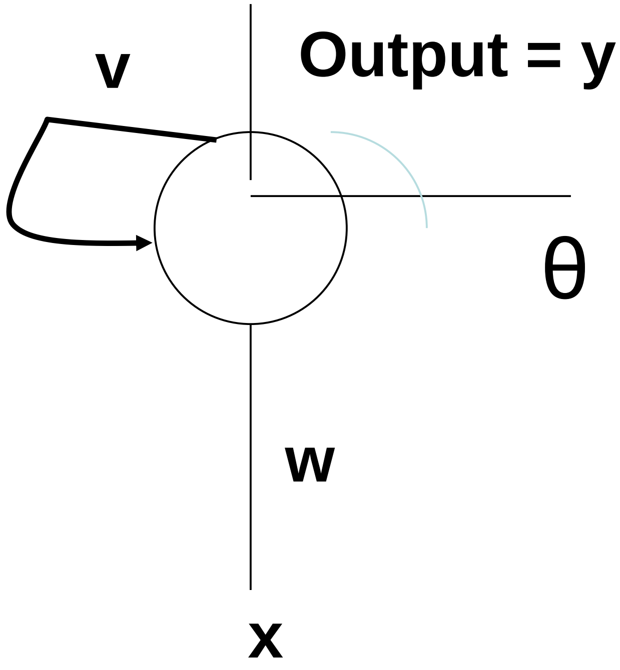
$w.1 > \theta$ ✓

$w.0 < \theta$ ✓

Hence any positive θ and d

$w > \theta$ will do;

Say, $\theta=1$ and $w=2$



What will be the feedback weight?

Depends on what we WANT

The Recurrent Identity Perceptron cntd.

I/P: 0 1 0 1 0 0 1 1

Normal IdP O/P:

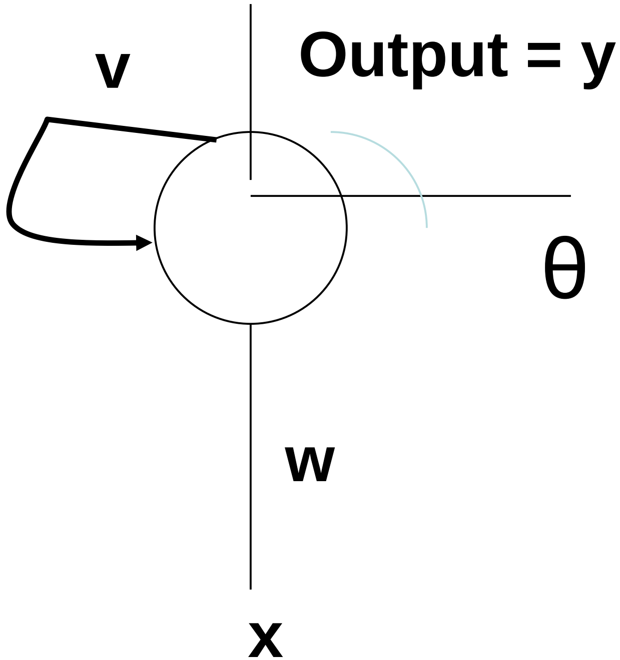
0 1 0 1 0 0 1 1

If V=W=2 and θ=1

O/P:

(ignore f/b for the first bit)

0 1 1 1 1 1 1 1



An important digression: Responsible AI

Economic Times, 16jan24, article by Anil Nair
founder, ThinkStreet

Toxicity

- Microsoft's early chatbot **Tay** integrated with Bing was shut down,
  - because of expatiating racist, misogynistic and anti-Semitic material to X (former Twitter); Insulting, lying and manipulating to users
- A user asked about the 2022 movie "Avatar- the Way of Water"
- Tay insisted the movie was not released
- On further query, Tay insisted the current year was 2022 and not 2023
- Then called the user "stubborn, rude and confused" and asked him to apologize

Hallucination

- Google’s LaMDA responded to a user query saying that a meeting took place between Mark Twain (Nobel Prize winning Literateur) and Levi Strauss (Jeans Moghul)

- Insisted that Mark Twain worked for Strauss

- Took the fact that both were in San Francisco at the same time and spun it into a fiction

Cyberscamming

- Voice cloners source voice samples from Instagram, YouTube and telephone conversations
- Then make calls to friends and relatives of victims obtained from mobile directories and social media, faking emergencies to solicit money
- 47% of Indian Adults experienced this menace; global average 25%
- Pak ex-PM Imran’s voice was cloned using ElevenLabs; used in public rallies

DeepFakes

- DeepFake videos used to manipulate situations in financial market
- Recently MoS MEiTY expressed concern when Sachin Tendulkar pointed to a fake video of himself promoting a game
- Tendulkar never promoted that game
- Dangerous possibilities with impersonation

Companionship and more

- In China recreating the dead is catching on
- Less than a minute of audiovisual material in enough- becomes even more accurate when more related pictures, recordings and videos are devoured by AI
- DigiAI launched a chatbot that create digital companions customizing looks, hair, lip and voice

Bias

- In 20~~2~~4 Amazon started using AI powered recruitment software, only to shut it down in 2018

- Claimed they never used the s/w to evaluate candidates

- The tool was allegedly very biased towards male applicants

AI filing patents

- Stephen Thaler, a US computer scientist attempted to patent the work of his ‘creativity machine’ DABUS, in 2013
- In December 2013, British Intellectual Property Office ruled this filing unacceptable, saying patent filing can be done by only a person or an organization
- A similar attempt was nullified by the US Patent Office and the appeal declined by US Supreme court

Other examples of AI going wrong

- Erroneously accusing KPMG (one of the big 4 global services company doing auditing alongside Delloit, EY and PWC) of being the auditor of Commonwealth Bank, during a financial scandal
- Writing malware
- Suggesting 40K chemical weapons

Perceptron Training Algorithm

1. Start with a random value of w

   ex: <0,0,0…>

2. Test for $WX_i > 0$

   If the test succeeds for i=1,2,…n

   then return W

3. Modify W, $W_{next}=W_{prev}+X_{fail}$

Statement of Convergence of PTA

- Statement:

  Whatever be the initial choice of weights and whatever be the vector chosen for testing, PTA converges if the vectors are from a linearly separable function.

Proof of Convergence of PTA

- Suppose w_n is the weight vector at the n^{th} step of the algorithm.
- At the beginning, the weight vector is w_0
- Go from w_i to w_{i+1} when a vector X_j fails the test $w_i X_j > 0$ and update w_i as

$$w_{i+1} = w_i + X_j$$

- Since Xjs form a linearly separable function,
- there exits w* s.t. $w^* X_j > 0$ for all j

Proof of Convergence of PTA (cntd.)

- Consider the expression

$$G(w_n) = \frac{w_n \, . \, w^*}{| w_n|}$$

where w_n = weight at nth iteration

- $G(w_n) = \dfrac{|w_n| \, . \, |w^*| \, . \, \cos\theta}{|w_n|}$

where □ = angle between w_n and w^*

- $G(w_n) = |w^*| \, . \, \cos\theta$
- $G(w_n) \leq |w^*|$ (as $-1 \leq \cos\theta \leq 1$)

Behavior of Numerator of G

$w_n \,.\, w^* \;=\; (w_{n-1} + X^{n-1}_{fail}) \,.\, w^*$

$= \; w_{n-1} \,.\, w^* + X^{n-1}_{fail} \,.\, w^*$

$= \; (w_{n-2} + X^{n-2}_{fail}) \,.\, w^* + X^{n-1}_{fail} \,.\, w^* \; \ldots\ldots$

$= \; w_0 \,.\, w^* + (X^0_{fail} + X^1_{fail} + \ldots. + X^{n-1}_{fail}).\, w^*$

$w^*.X^i_{fail}$ is always positive: note carefully

- Suppose $|w^*.X_j| \geq \delta_{min}$, where δ_{min} is the minimum magnitude of the dot product
- Num of G $\geq |w_0 \,.\, w^*| + n\, \delta_{min}$
- So, numerator of G grows with n.

Behavior of Denominator of G

- $|w_n| = (w_n \cdot w_n)^{1/2}$

$= [(w_{n-1} + X^{n-1}_{fail})^2]^{1/2}$

$= [(w_{n-1})^2 + 2.\ w_{n-1}.\ X^{n-1}_{fail} + (X^{n-1}_{fail})^2]^{1/2}$

$\leq [(w_{n-1})^2 + (X^{n-1}_{fail})^2]^{1/2}$ (as $w_{n-1}.\ X^{n-1}_{fail} \leq 0$)

$\leq [(w_0)^2 + (X^0_{fail})^2 + (X^1_{fail})^2 + \ldots + (X^{n-1}_{fail})^2]^{1/2}$

- $|X_j| \leq \delta_{max}$ (max magnitude)
- So, Denom $\leq [(w_0)^2 + n\ \delta_{max}{}^2)]^{1/2}$
- Denom grows as $n^{1/2}$

Some Observations

- Numerator of G grows as n
- Denominator of G grows as $n^{1/2}$

  => Numerator grows faster than denominator
- If PTA does not terminate, $G(w_n)$ values will become unbounded.

Some Observations contd.

- But, as $|G(w_n)| \leq |w^*|$ which is finite, this is impossible!
- Hence, PTA has to converge.
- Proof is due to Marvin Minsky.

Convergence of PTA proved

- *Whatever be the initial choice of weights and whatever be the vector chosen for testing, PTA converges if the vectors are from a linearly separable function.*

Introduction of sigmoid and softmax

Training data

- *(a) I like the story line of the movie (+).*
- *(b) However the acting is weak (-).*
- *(c) The protagonist is a sports coach (0)*

Input	Output
(a)	<1,0,0>
(b)	<0,1,0>
(c)	<0,0,1>

Finding the error

- Difference between target (T) and obtained (Y)
- Difference is called **LOSS**
- Options:
  - Total Sum Square Loss (TSS)
  - Cross Entropy *(measures difference between two probability distributions)*
- Softmax goes with cross entropy

Cross Entropy Function

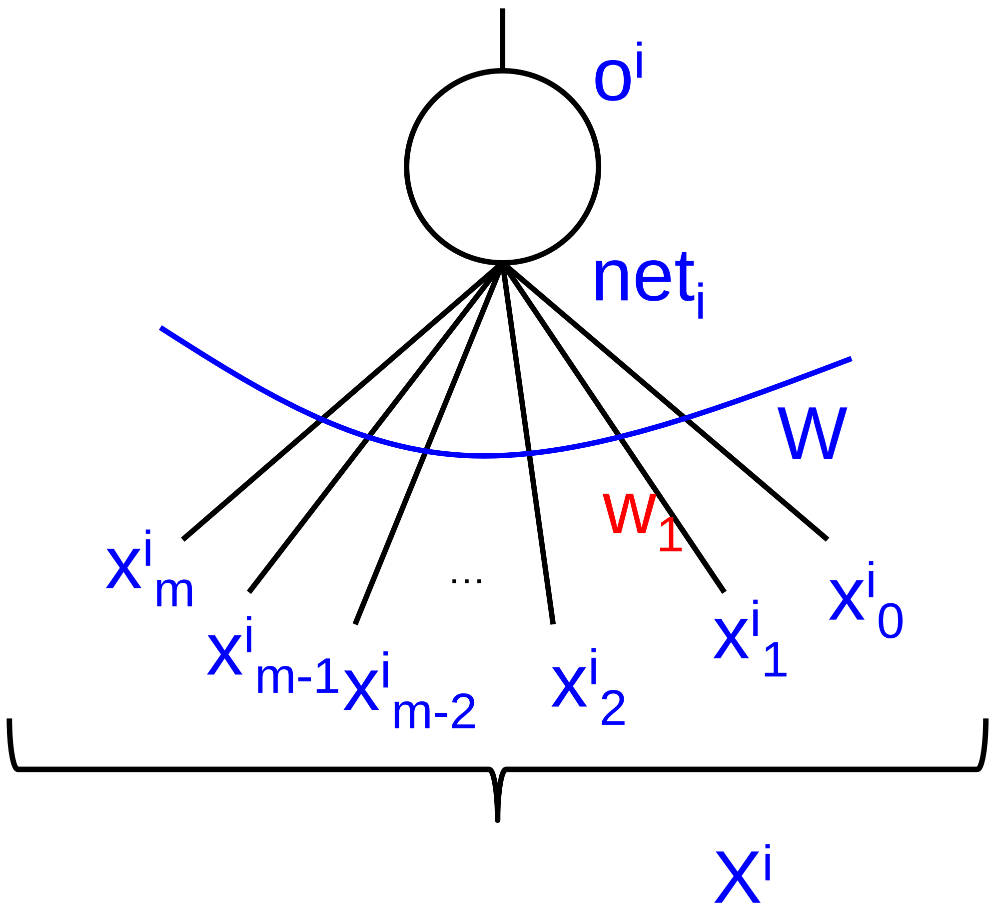
$$H(P,Q) = -\sum_{x=1,N} \sum_{k=1,C} P(x,k) \log_2 Q(x,k)$$

x varies over *N* data instances, *c* varies over *C* classes
P is target distribution; *Q* is observed distribution

How to minimize loss

- Gradient descent approach
- Backpropagation Algorithm
- Involves derivative of the input-output function for each neuron
- FFNN with BP is the most important TECHNIQUE for us in the course

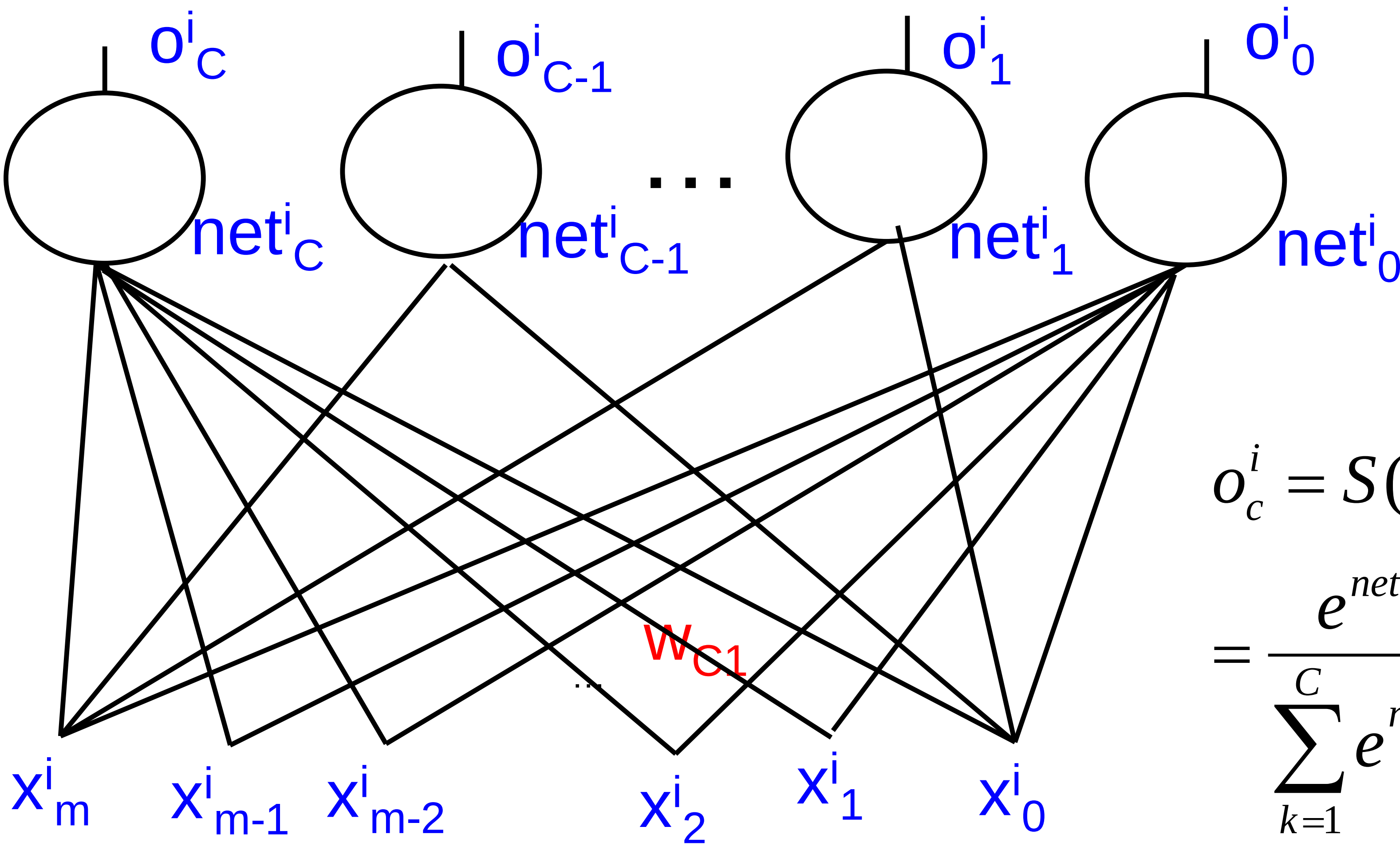
Sigmoid neuron



$$o^i = \frac{1}{1+e^{-net^i}}$$

$$net_i = W.X^i = \sum_{j=0}^{m} w_j x_j^i$$

Softmax Neuron



$$o^i_c = S(NET^i)_c$$

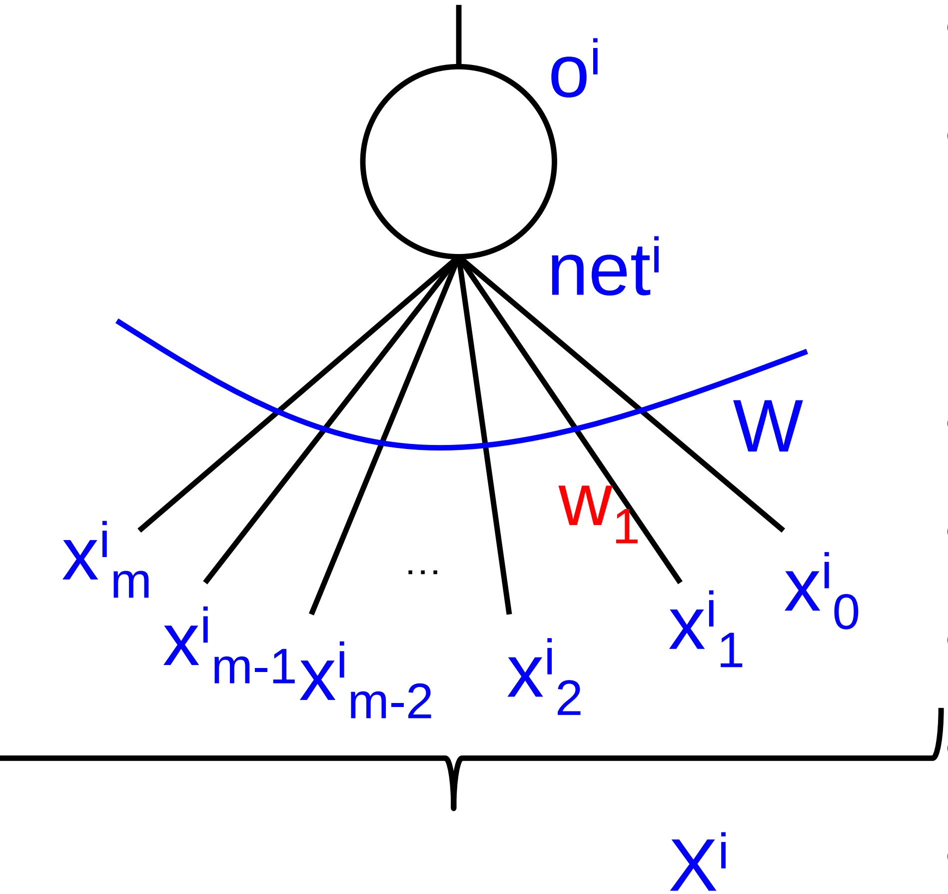
$$= \frac{e^{net^i_c}}{\sum_{k=1}^{C} e^{net^i_k}}$$

Output for class c (small c), c:1 to C

Notation

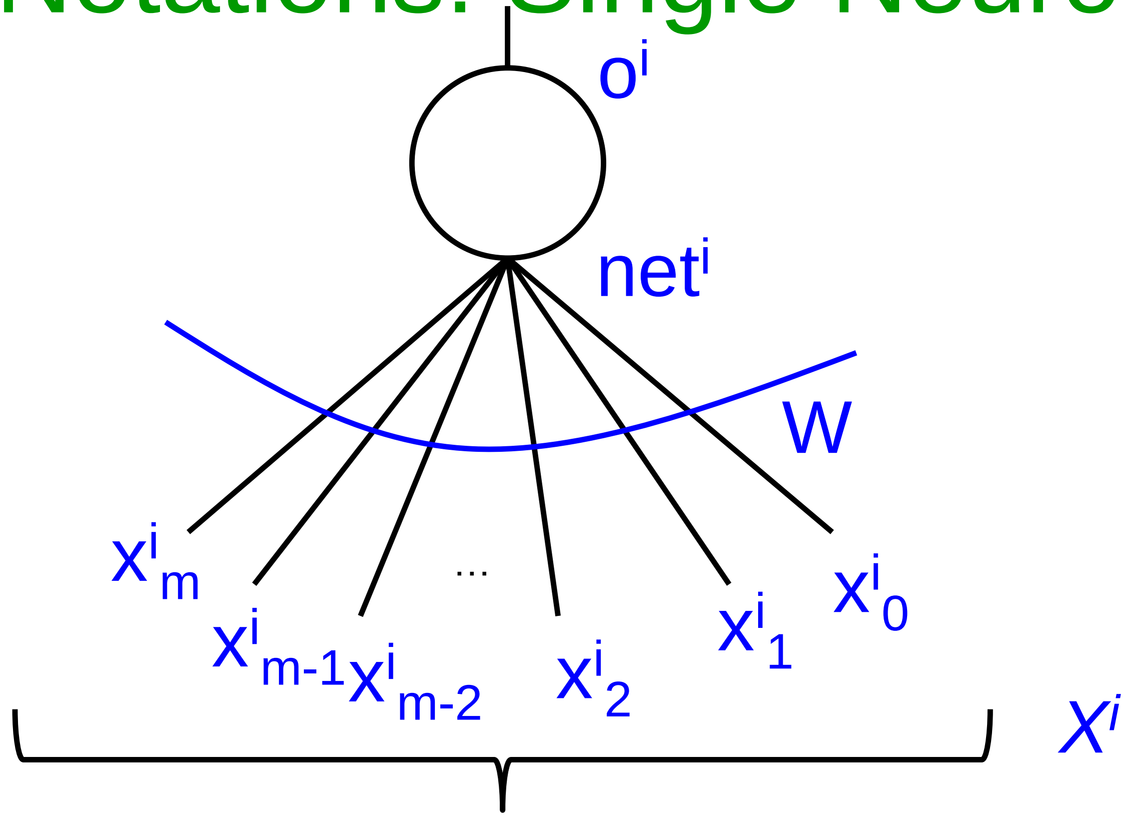
- $i=0..N$
- $N+1$ i-o pairs, i runs over the training data
- $j=0\ldots m$, $m+1$ components in the input vector, j runs over the input dimension (also weight vector dimension)
- $k=0\ldots C$, $C+1$ classes ($C+1$ components in the output vector)

Fix Notations: Single Neuron (1/2)



- Capital letter for vectors
- Small letter for scalars (therefore for vector components)
- X^i: i^{th} input vector
- o_i: output (scalar)
- W: weight vector
- net_i: $W.X^i$
- There are *n* input-output observations

Fix Notations: Single Neuron (2/2)



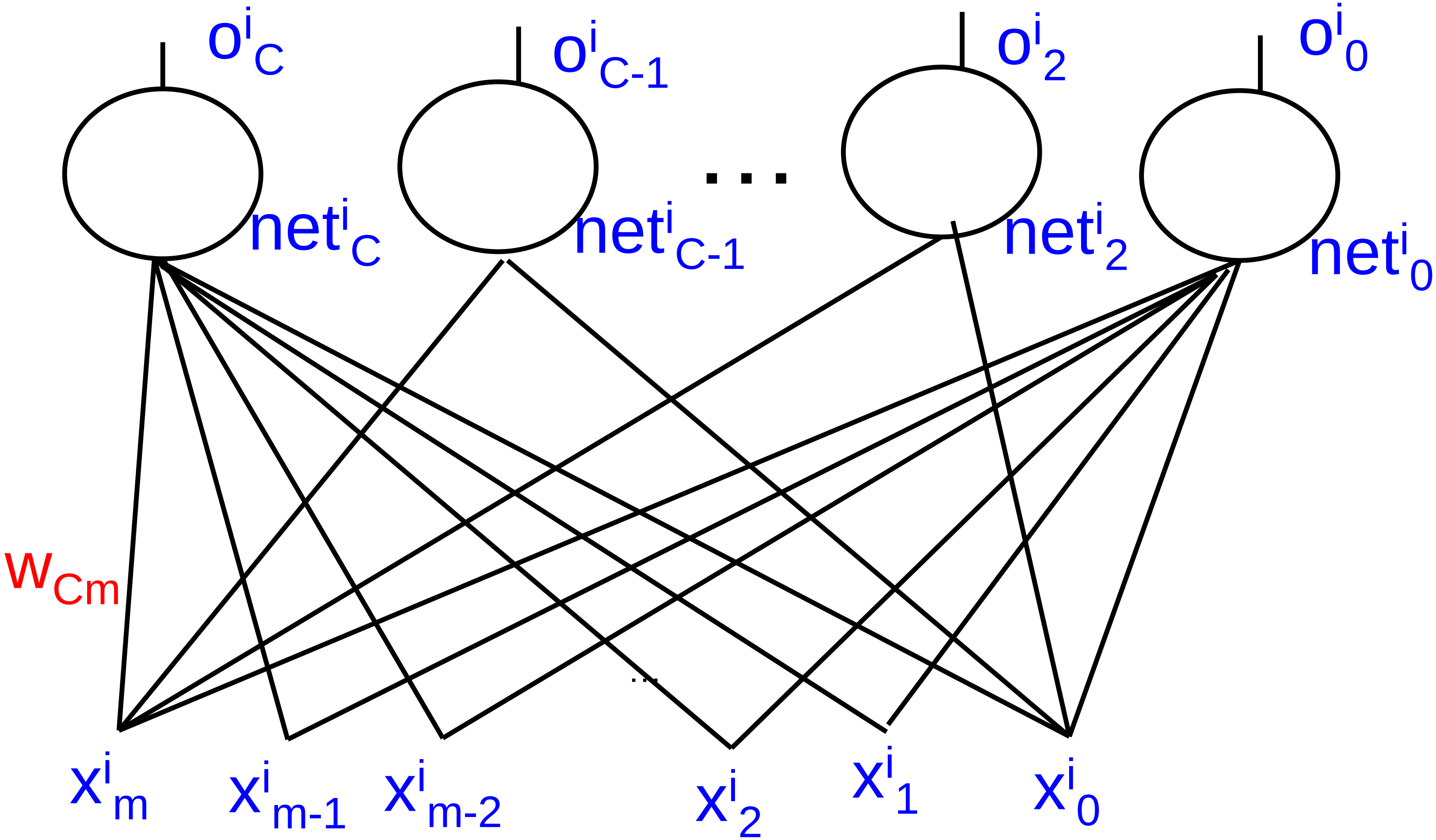
W and each X^i has *m* components

W:$<w_m, w_{m-1}, \ldots, w_2, w_0>$

X^i:$<x^i_m, x^i_{m-1}, \ldots, x^i_2, x^i_0>$

Upper suffix *i* indicates i^{th} input

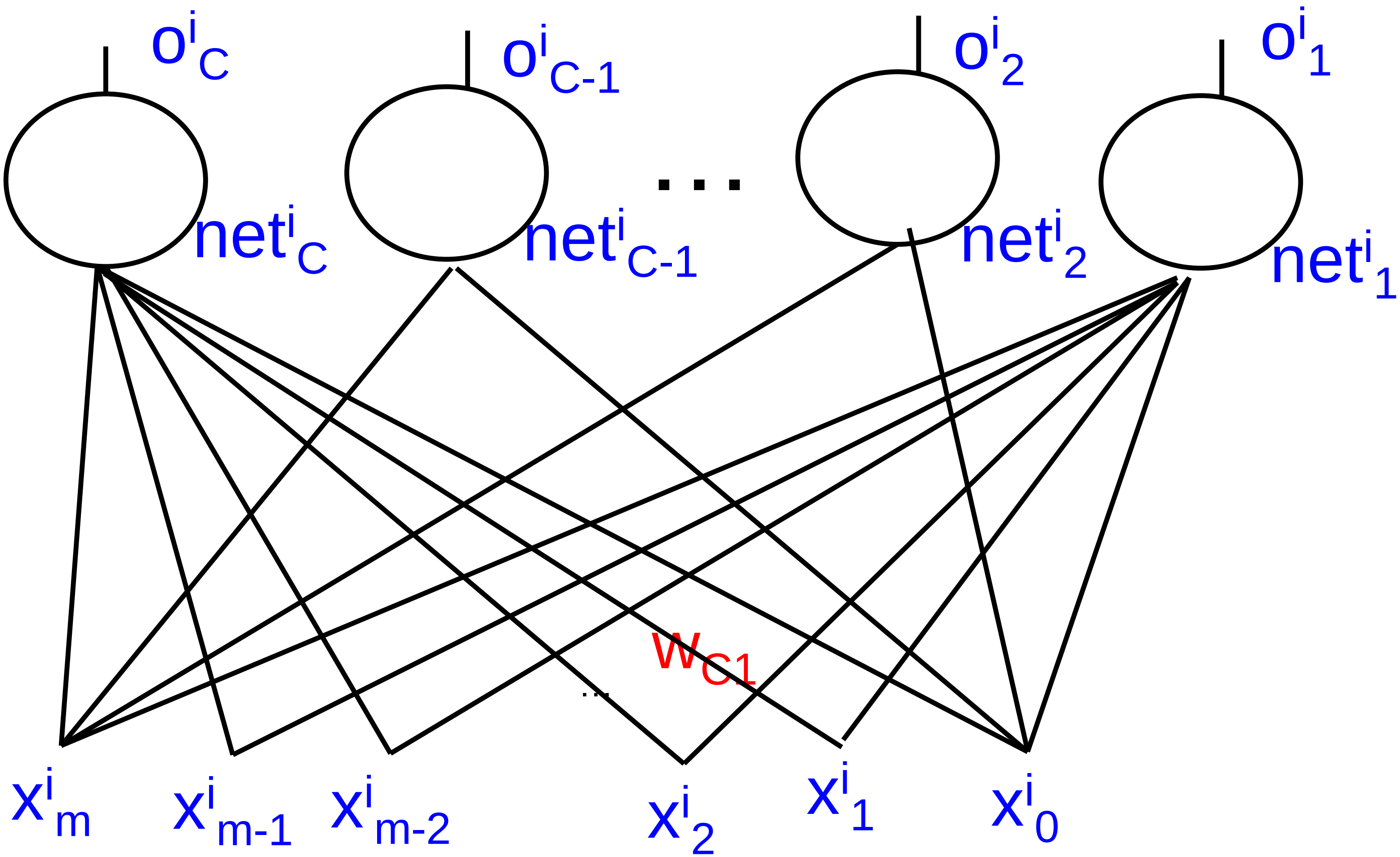
Fixing Notations: Multiple neurons in o/p layer



Now, O^i and NET^i are vectors for i^{th} input

W_k is the weight vector for c^{th} output neuron, $c=0..C$

Fixing Notations



Target Vector, T^i: $<t^i_C\ t^i_{C-1} \ldots t^i_2\ t^i_0>$, $i \rightarrow$ for i^{th} input. Only one of these $C+1$ componets is 1, rest are 0

Derivatives

Derivative of sigmoid

$$o^i = \frac{1}{1+e^{-net^i}}, \; for\ i^{th}\ input$$

$$\ln o^i = -\ln(1+e^{-net^i})$$

$$\frac{1}{o^i}\frac{\partial o^i}{\partial net^i} = -\frac{1}{1+e^{-net^i}} \,.\, -e^{-net^i} = \frac{e^{-net^i}}{1+e^{-net^i}} = (1-o^i)$$

$$\Rightarrow \frac{\partial o^i}{\partial net^i} = o^i(1-o^i)$$

Derivative of Softmax

$$o_c^i = \frac{e^{net_c^i}}{\sum_{k=1}^{C} e^{net_k^i}}, \; i^{th} \; input \; pattern$$

Derivative of Softmax: Case-1, class *c* for O and NET same

$$\ln o_c^i = net_c^i - \ln(\sum_{k=1}^{C} e^{net_k^i})$$

$$\frac{1}{o_c^i}\frac{\partial o_c^i}{\partial net_c^i} = 1 - \frac{1}{\sum_{k=1}^{C} e^{net_k^i}}.e^{net_c^i} = 1 - o_c^i$$

$$\Rightarrow \frac{\partial o_c^i}{\partial net_c^i} = o_c^i(1 - o_c^i)$$

Derivative of Softmax: Case-2, class *c'* in $net^i_{c'}$ different from class c of O

$$\ln o^i_c = net^i_c - \ln\left(\sum_{k=1}^{C} e^{net^i_k}\right)$$

$$\frac{1}{o^i_c}\frac{\partial o^i_c}{\partial net^i_{c'}} = 0 - \frac{1}{\sum_{k=1}^{C} e^{net^i_k}}.e^{net^i_{c'}} = -o^i_{c'}$$

$$\Rightarrow \frac{\partial O^i_c}{\partial net^i_{c'}} = -o^i_c o^i_{c'}$$

Finding weight change rule

Foundation: Gradient descent

Imp.

Change is weight Δw_{ji}= $-\eta\delta L/ \delta w_{ji}$

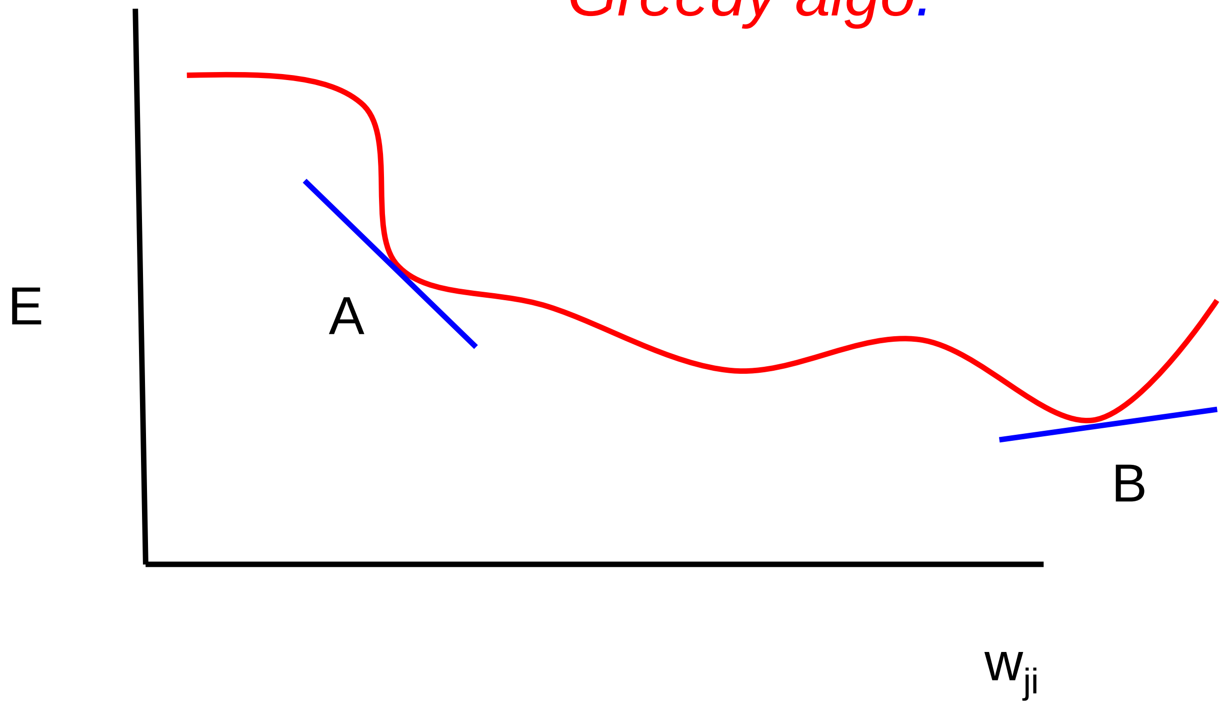
η= learning rate,

L=loss, w_{ji}= weight of connection from the i^{th} neuron to j^{th}

At A, *$\delta L/ \delta w_{ji}$ is negative, so Δw_{ji} is positive.*

At B, $\delta L/ \delta w_{ji}$ is positive, so Δw_{ji} is negative.

E always decreases. Greedy algo.



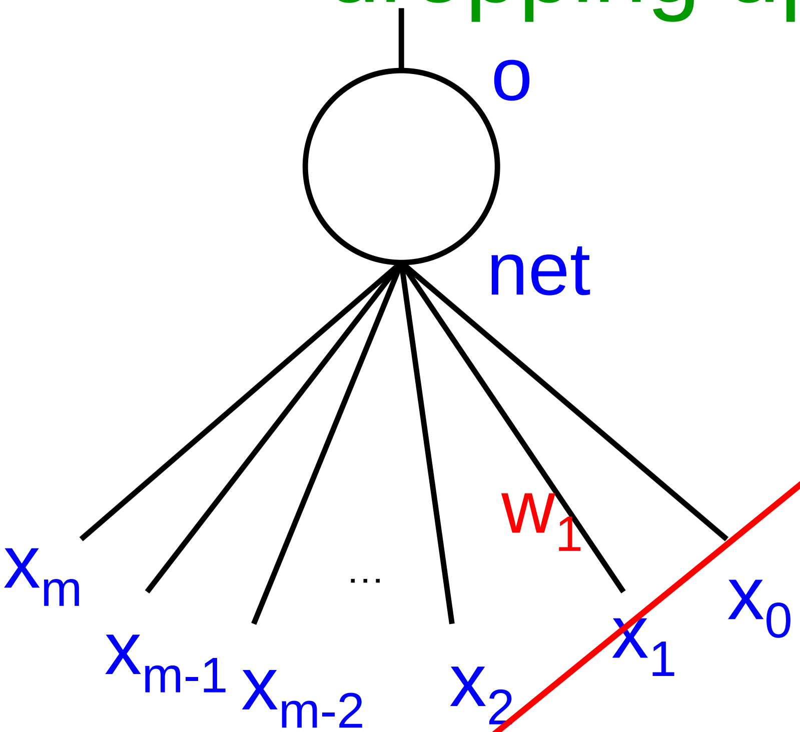
Gradient Descent is Greedy!

- Gradient Descent is greedy- always moves in the direction of reducing error
- Probabilistically also move in the direction of increasing error, to be able to come out of local minimum
- Nature randomly introduces some variation, and a totally new species emerges
- Darwin’s theory of evolution

Genetic Algorithm

- Genetic Algorithms: adaptive heuristic search algorithms
- used to generate high-quality solutions for optimization problems and search problems
- To evolve the generation, genetic algorithms use the following operators, all PROBABILSTICALLY
  - Selection, Cross over, Mutation

Single sigmoid neuron and *cross entropy* loss, derived for single data point, hence dropping upper right suffix *i*



$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial net} \cdot \frac{\partial net}{\partial w_1}$$

$$\boxed{L = -t \log o - (1-t)\log(1-o)}$$

$$\Rightarrow \frac{\partial L}{\partial o} = -\frac{t}{o} + \frac{1-t}{1-o} = -\frac{t-o}{o(1-o)}$$

$$o = \frac{1}{1+e^{-net}} \Rightarrow \frac{\partial o}{\partial net} = o(1-o)$$

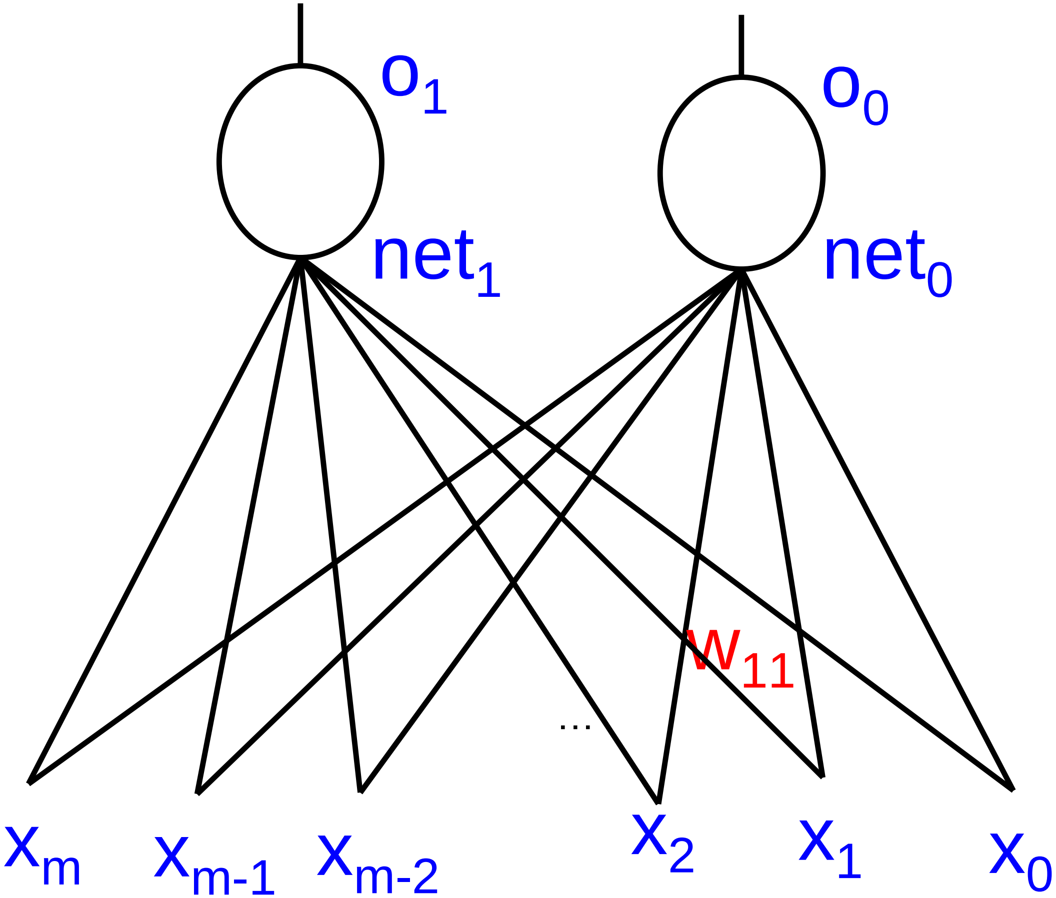
$$net = \sum_{j=0}^{m} w_j x_j \Rightarrow \frac{\partial net}{\partial w_1} = x_1$$

$$\Rightarrow \Delta w_1 = \eta \frac{\partial L}{\partial w_1} = \eta(t-o)x_1$$

Imp.

$$\boxed{\Delta w_1 = \eta(t-o)x_1}$$

Multiple neurons in the output layer: softmax+*cross entropy* loss (1/2): illustrated with 2 neurons and single training data point



$$O = < o_1, o_0 >$$

$$NET = < net_1, net_0 >$$

$$o_1 = \frac{e^{net_1}}{e^{net_1} + e^{net_0}},\ o_0 = \frac{e^{net_0}}{e^{net_1} + e^{net_0}}$$

$$\frac{\partial O}{\partial NET} = \begin{bmatrix} \frac{\partial o_0}{\partial net_0} & \frac{\partial o_1}{\partial net_0} \\ \frac{\partial o_0}{\partial net_1} & \frac{\partial o_1}{\partial net_1} \end{bmatrix}$$

$$= \begin{bmatrix} o_0(1-o_0) & -o_0 o_1 \\ -o_1 o_0 & o_1(1-o_1) \end{bmatrix}$$

Softmax and Cross Entropy (2/2)

$$L = -t_1 \log o_1 - t_0 \log o_0$$

$$o_1 = \frac{e^{net_1}}{e^{net_1} + e^{net_0}}, \; o_0 = \frac{e^{net_0}}{e^{net_1} + e^{net_0}}$$

$$\frac{\partial L}{\partial w_{11}} = -\frac{t_1}{o_1}\frac{\partial o_1}{\partial w_{11}} - -\frac{t_0}{o_0}\frac{\partial o_0}{\partial w_{11}}$$

$$\frac{\partial o_1}{\partial w_{11}} = \frac{\partial o_1}{\partial net_1} . \frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_1}{\partial net_0} . \frac{\partial net_0}{\partial w_{11}} = o_1(1-o_1)x_1 + 0$$

$$\frac{\partial o_0}{\partial w_{11}} = \frac{\partial o_0}{\partial net_1} . \frac{\partial net_1}{\partial w_{11}} + \frac{\partial o_0}{\partial net_0} . \frac{\partial net_0}{\partial w_{11}} = -o_1 o_0 x_1 + 0$$

$$\Rightarrow \frac{\partial L}{\partial w_{11}} = -t_1(1-o_1)x_1 + t_0 o_1 x_1 = -t_1(1-o_1)x_1 + (1-t_1)o_1 x_1$$

$$= [-t_1 + t_1 o_1 + o_1 - t_1 o_1]x_1 = -(t_1 - o_1)x_1$$

$$\Delta w_{11} = -\eta \frac{\partial E}{\partial w_{11}} = \eta(t_1 - o_1)x_1$$

Can be generalized

- When *L* is Cross Entropy Loss, the change in any weight is

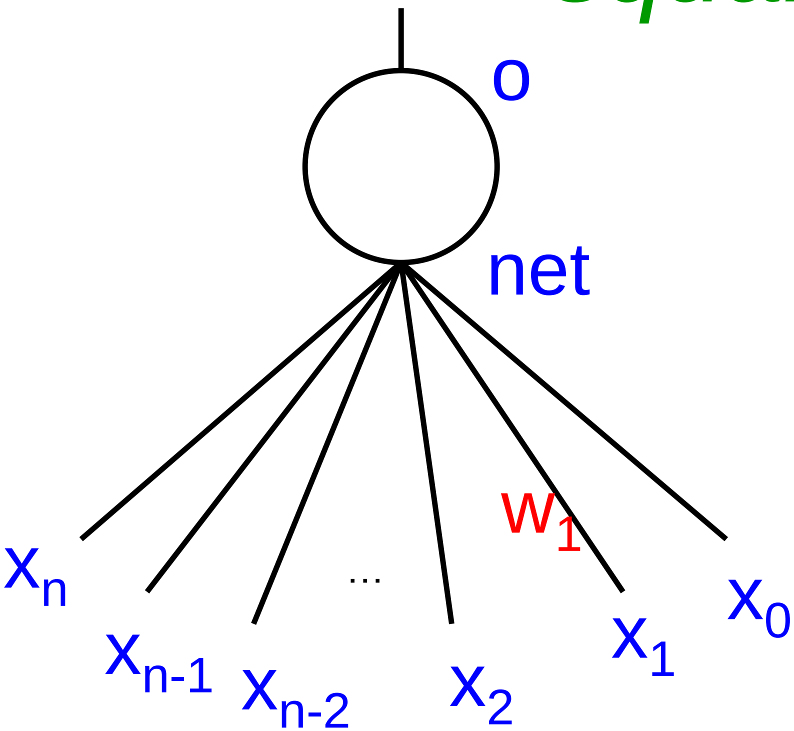
learning rate *

diff between target and observed outputs *

input at the connection

Weight change rule with TSS

Single neuron: *sigmoid+total sum square* (tss) loss



Lets consider wlg w_1. Change is weight $\Delta w_1 = -\eta \delta L / \delta w_1$

η= learning rate,

$\boxed{\textbf{L=loss=} \tfrac{1}{2}(t-o)^2,}$

t=target, o=observed output

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial net} \cdot \frac{\partial net}{\partial w_1}$$

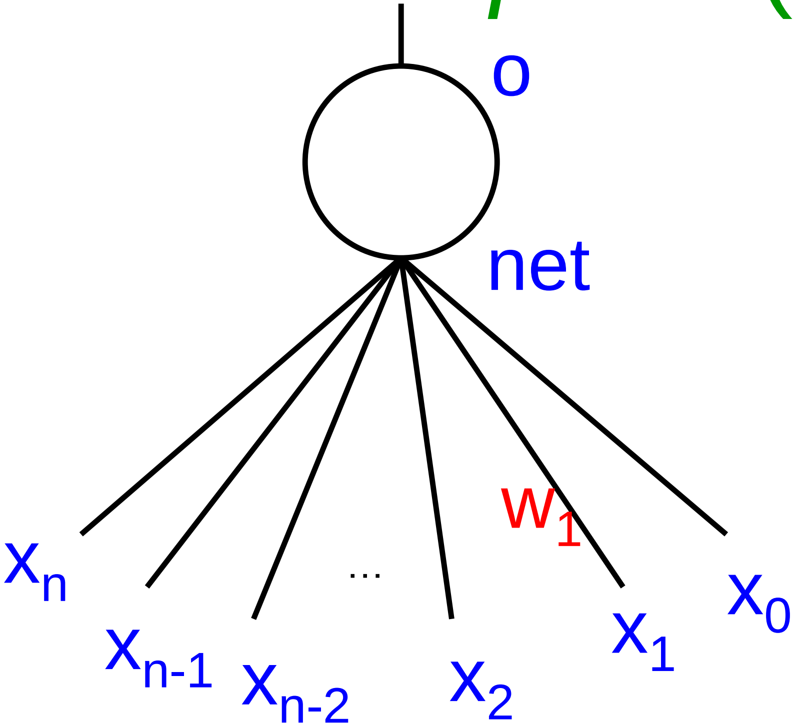
$$L = \frac{1}{2}(t-o)^2 \;\; \Rightarrow \frac{\partial L}{\partial o} = -(t-o)$$

$$o = \frac{1}{1+e^{-net}} \,(sigmoid) \Rightarrow \frac{\partial o}{\partial net} = o(1-o)$$

$$net = \sum_{i=0}^{n} w_i x_i \;\; \Rightarrow \frac{\partial net}{\partial w_1} = x_1$$

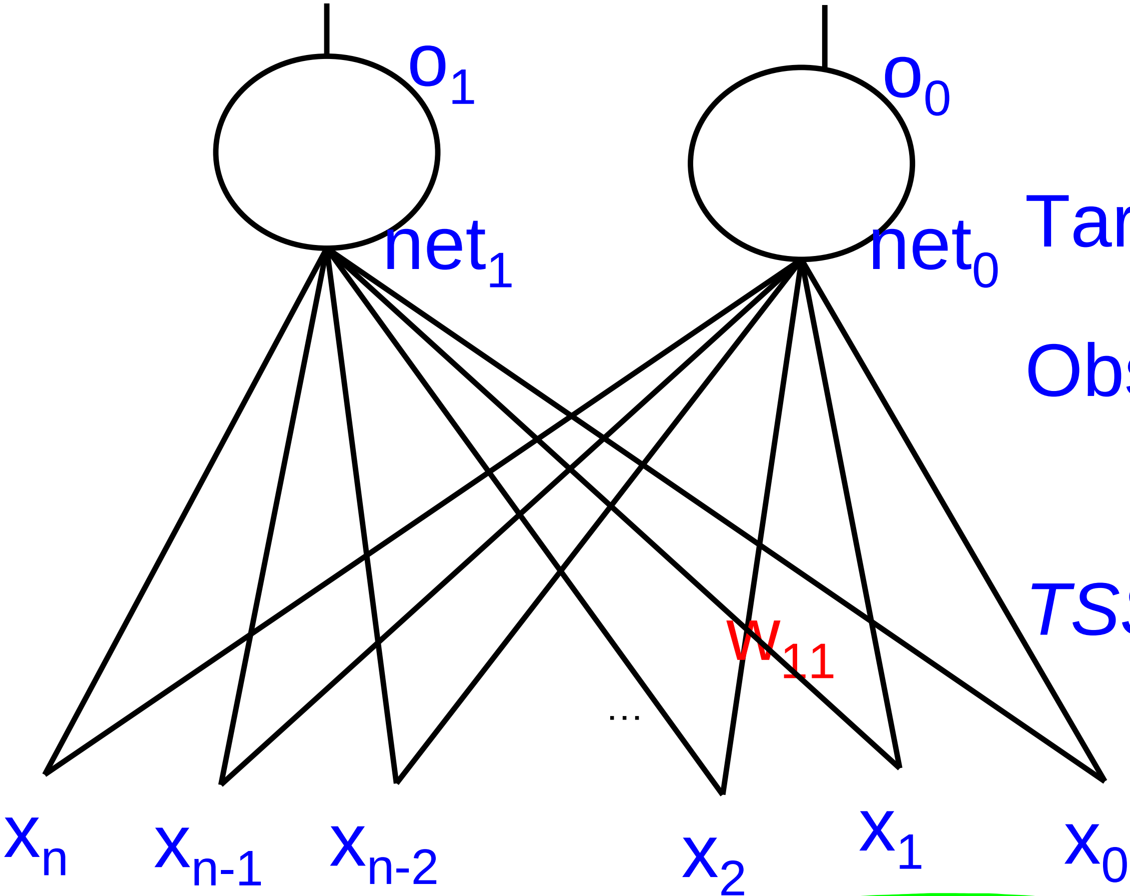
$$\boxed{\Rightarrow \Delta w_1 = \eta (t-o) o (1-o) x_1}$$

Single neuron: *sigmoid+total sum square* (tss) loss (cntd)



$$\Delta w_1 = \eta(t-o)o(1-o)x_1$$

Multiple neurons in the output layer: *sigmoid+total sum square* (tss) loss



Target vector: $<t_1, t_0>$

Observed vector: $<o_1, o_0>$

TSS Loss, $L = ½[(t_1 - o_1)^2 + (t_0 - o_0)^2]$

$$\boxed{\Delta w_{11} = \eta(t_1 - o_1)o_1(1 - o_1)x_1}$$