

هنتكلم انهارد عن النويرال نتورك ده هيبقا انتروودكشن سريعه وقدام هنتكلم عنها اكثر ... وانهارده هنتكلم علي نوع بدائي جداً وهو ال perceptron .. مالوش اي hidden layer .. مجرد أكتيفيشن و شوية ويتس .. الدكتور بيتكلم شويه عن المخ .. وان النويرال نتوركس اللي موجوده بتستخدم نفس اللي بيحصل في المخ .. ما علينا ☺ .. محاضرة انهارد خفيفه يعني مفهائش كلام كثير بيحري ورا بعضو فمش كاتب كثير ..

## Comparison

- Brain
  - Network of neurons
  - Nerve signals propagate in a neural network
  - Parallel computation
  - **Robust (neurons die everyday without any impact)**
- Computer
  - Bunch of gates
  - Electrical signals directed by gates
  - Sequential and parallel computation
  - **Fragile (if a gate stops working, computer crashes)**

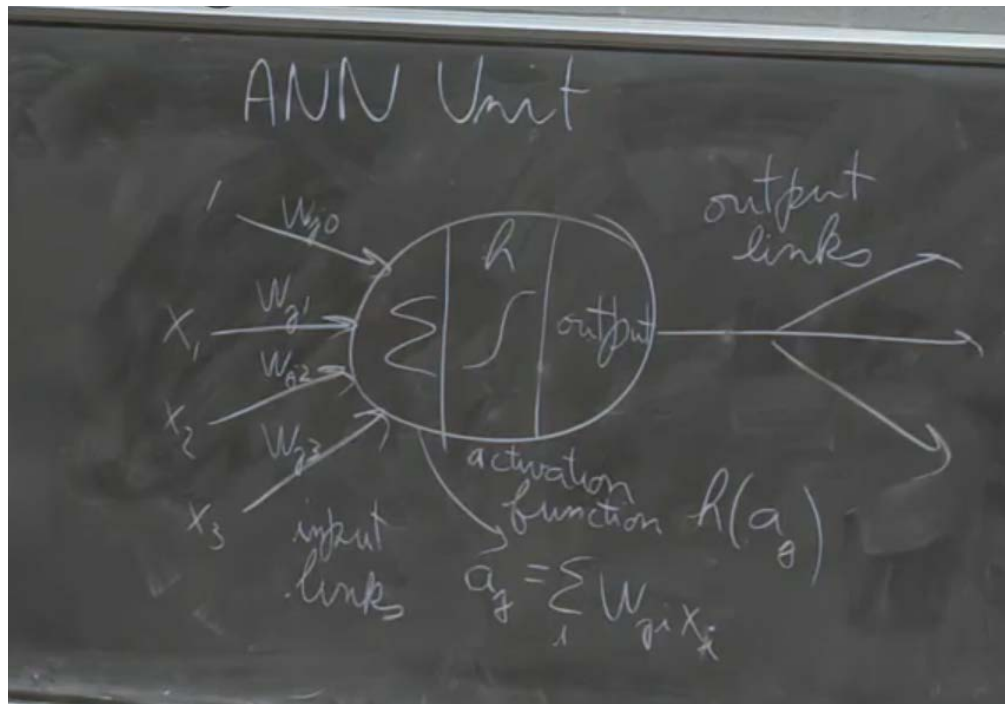
طيب .. الاساس بتاع النويرال نتورك هو النوود .. النتورك هيبقا ليها يونتس كتيره .. كل يونت ليها اندكس  $i$  .. في يونيت اسمها  $i$  و في يونت اسمها  $j$  وهكذا ... عندي شوية انبوتس  $x_i$  وبطلع signal جديده اللي هي  $a_j$  ...

## ANN Unit

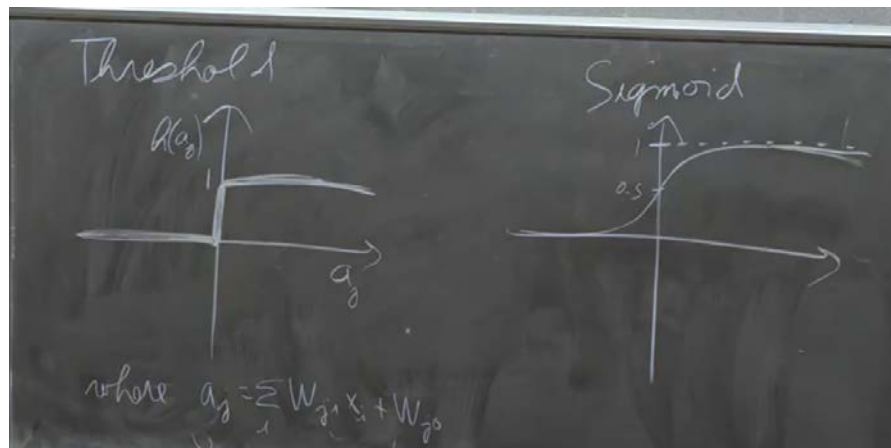
- For each unit  $i$ :
- **Weights:  $W$** 
  - Strength of the link from unit  $i$  to unit  $j$
  - Input signals  $x_i$  weighted by  $W_{ji}$  and linearly combined:  

$$a_j = \sum_i W_{ji} x_i + w_0 = W_j \bar{x}$$
- **Activation function:  $h$** 
  - Numerical signal produced:  $y_j = h(a_j)$

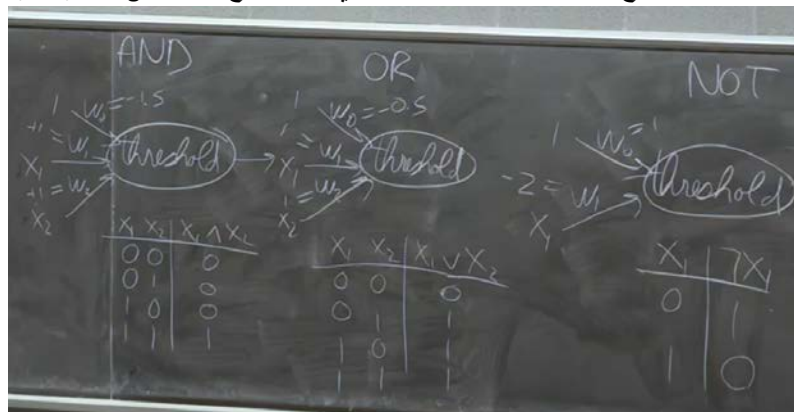
عندي node .. هيبقا ليها شوية inputs .. كل انبوت هيبقا rescaled with a weight ... اللي هيحصل ان النوود هتاخذ ال linear combination of inputs according to the weight .. هتجيب  $a_j$  هتساوي ال Sum over  $i$  of  $w_{ji} x_i$  ... فيبعد ما حسبنا ال linear combination هن apply the activation function ... فانت هتجيب  $h(a_j)$  .... وده هيطلعلك أوتبوت هيروح لل next nodes



افضل بقا حط nodes كثيره مع بعض .. ال unit بتحسب non-linear function ل linear combinations based on weights we have ... ال activation هتبقا non-linear .. لو معندكش activation .. هيبقا عندك حاجه linear و ده مش هيبقا expressive أوي .. ال activation هنا دورها انها ت fire .. اللي هو ان ال unit is activated and outputs something near to 1 .. ال firing لو الأوتبوت مكنش activated .. في الحاله ديه انت مطلع أوتبوت قرب الصفر ... هنتكلم علي نوعين من الأكتيفيشن فانكشنز اللي موجودين .. threshold and sigmoid .. ال unit step function هي sigmoid فانكشن هي smooth version of thresholding function .. مشكله ال threshold انها not smooth and continus فلو حبيت تحسب ال gradient هيطلعلك مشكله ...



طيب تعال نشوف ال thresholding .. حاجه بتطلع صفر و واحد .. هنشوف ازاى بنتعامل مع ال and, or, not gates ... هنعلمهم ال units بتاعتهم



في نوعين من النوركس .. اول واحد هو ال feed-forward .. و ده بببقا acyclic .. رايح في اتجاه واحد بس.. النوع الثاني هو ال RNN .. ديه حاجه مهمه في ال NLP .. ده عشان انت عندك انبوتس الطول بتاعها بيتغير ..فكر في جمله .. فيها كلمات كتيره فال RNN هيبقا مناسبه لحاجه زي كذا .. وديه هتبقا cyclic ..عندك جزء بيهاندل كلمه وبعدين يرجع ياخد ثاني كلمه و بعدين يرجع وياخد ثالث كلمه .. بتقدر ت memorize information ...

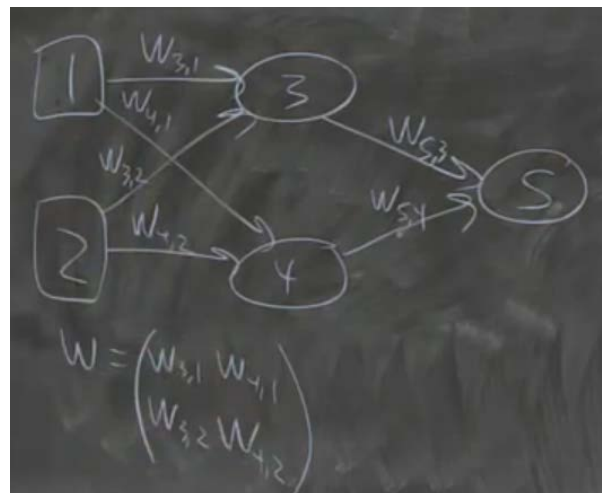
# Network Structures

- **Feed-forward network**

- Directed **acyclic** graph
- No internal state
- Simply computes outputs from inputs

- Recurrent network

- Directed **cyclic** graph
- Dynamical system with internal states
- Can memorize information




طیب تعال نشوف التریننج هیقا عامل ازای ..

# Threshold Perceptron Learning

- Learning is done separately for each unit  $j$ 
  - Since units do not share weights
- Perceptron learning for unit  $j$ :
  - For each  $(x, y)$  pair do:
    - Case 1: correct output produced
$$\forall_i W_{ji} \leftarrow W_{ji}$$
    - Case 2: output produced is 0 instead of 1
$$\forall_i W_{ji} \leftarrow W_{ji} + x_i$$
    - Case 3: output produced is 1 instead of 0
$$\forall_i W_{ji} \leftarrow W_{ji} - x_i$$
  - Until correct output for all training instances

بالنسبة لل perceptron learning الذي يستخدم ال threshold فانكشن ك activation .. كل unit هتستخدم ال threshold function كأكتنفیشن ...  
و هنا عندك ألجوردم بسيط .. لووب واحد عشان تعدل الویتس .. حتي لما كان البرسيبترون لیه اکثر من أوتوت .. هتعامل كل أوتوت لوحده .. "كل كيلو  
ف كيس" .. كل یونیت لیه شویة إیدجز الی هی الویتس .. وكل یونیت لیه الاإیدجز الی إندبت علی بقیت الیونیتس .. فالألجوردم ده هیبقا لكل أوتوت یونیت  
لوحدها .. مش بالإجماع .. مفیث dependency ...

## Threshold Perceptron Learning


- Learning is done separately for each unit  $j$ 
  - Since units do not share weights
- Perceptron learning for unit  $j$ :
  - For each  $(x, y)$  pair do:
    - Case 1: correct output produced (✓) 

$$\forall_i W_{ji} \leftarrow W_{ji}$$
    - Case 2: output produced is 0 instead of 1 ✗
$$\forall_i W_{ji} \leftarrow W_{ji} + x_i$$
    - Case 3: output produced is 1 instead of 0 ✗
$$\forall_i W_{ji} \leftarrow W_{ji} - x_i$$
  - Until correct output for all training instances

## Threshold Perceptron Learning

- Dot products:  $\bar{x}^T \bar{x} \geq 0$  and  $-\bar{x}^T \bar{x} \leq 0$
- Perceptron computes
  - 1 when  $\mathbf{w}^T \bar{x} = \sum_i x_i w_i + w_0 > 0$
  - 0 when  $\mathbf{w}^T \bar{x} = \sum_i x_i w_i + w_0 < 0$
- If output should be 1 instead of 0 then
$$\mathbf{w} \leftarrow \mathbf{w} + \bar{x} \text{ since } (\mathbf{w} + \bar{x})^T \bar{x} \geq \mathbf{w}^T \bar{x}$$
- If output should be 0 instead of 1 then
$$\mathbf{w} \leftarrow \mathbf{w} - \bar{x} \text{ since } (\mathbf{w} - \bar{x})^T \bar{x} \leq \mathbf{w}^T \bar{x}$$

## Alternative Approach

- Let  $y \in \{-1, 1\} \forall y$  
- Let  $M = \{(x_n, y_n)_{\forall n}\}$  be set of misclassified examples
  - i.e.,  $y_n \mathbf{w}^T \bar{x}_n < 0$
- Find  $\mathbf{w}$  that minimizes misclassification error
$$E(\mathbf{w}) = - \sum_{(x_n, y_n) \in M} y_n \mathbf{w}^T \bar{x}_n$$
- Algorithm: gradient descent
$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla E$$

learning rate  
or step length

# Sequential Gradient Descent

- Gradient:  $\nabla E = -\sum_{(x_n, y_n) \in M} y_n \bar{x}_n$

- <sup>1 exmp @ a time</sup> Sequential gradient descent:
  - Adjust  $\mathbf{w}$  based on one example  $(x, y)$  at a time

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y \bar{\mathbf{x}}$$

- When  $\eta = 1$ , we recover the threshold perceptron learning algorithm

## Threshold Perceptron Hypothesis Space

- Hypothesis space  $h_{\mathbf{w}}$ :
  - All binary classifications with parameters  $\mathbf{w}$  s.t.  
 $\mathbf{w}^T \bar{\mathbf{x}} > 0 \rightarrow +1$   
 $\mathbf{w}^T \bar{\mathbf{x}} < 0 \rightarrow -1$
- Since  $\mathbf{w}^T \bar{\mathbf{x}}$  is linear in  $\mathbf{w}$ , perceptron is called a **linear separator**
- **Theorem:** Threshold perceptron learning converges iff the data is linearly separable