COMP 472 Artificial Intelligence
Machine Learning
Intro to Neural Networks & Perceptrons

Video #7

- Russell & Norvig: Sections 19.6, 21.1

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks

   a. Perceptrons
   b. Multi Layered Neural Networks

Neural Networks

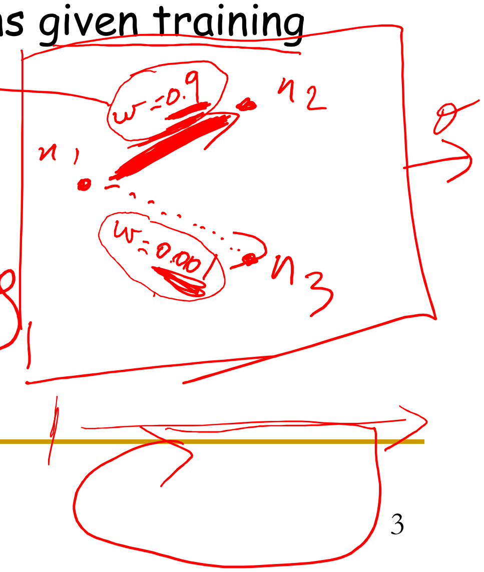
- Learning approach inspired by biology
  - the neurons in the human brain

- Set of many simple processing units (called neurons) connected together nodes
  - the behavior of each neuron is very simple
  - but a network of neurons can have sophisticated behavior and can be used for complex tasks
  - neurons are connected to each other to form a network
  - the strength of the connection between neurons are determined by weights
  - the network learns by learning the weights between neurons given training data

parameters that we will learn

- Different types of network architectures exist
  - feed forward neural networks (FFNN)
  - recurrent neural networks (RNN) NLP
  - convolutional neural networks (CNN) image
  - ... GAN

training data

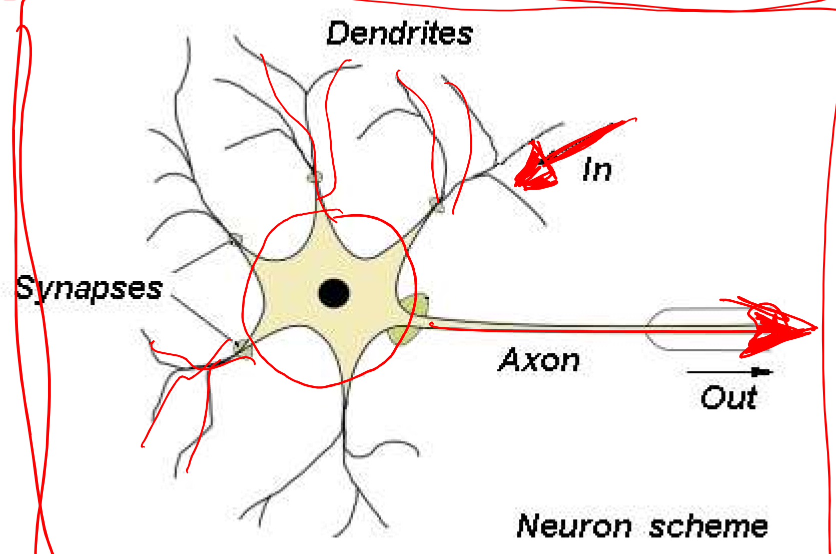


Biological Neurons

- Human brain =
  - 100 billion neurons
  - each neuron may be connected to 10,000 other neurons
  - passing signals to each other via 1,000 trillion **synapses**

- A neuron is made of:
  - Dendrites: filaments that provide input to the neuron
  - Axon: sends an output signal
  - Synapses: connection with other neurons – releases neurotransmitters to other neurons

1 neuron

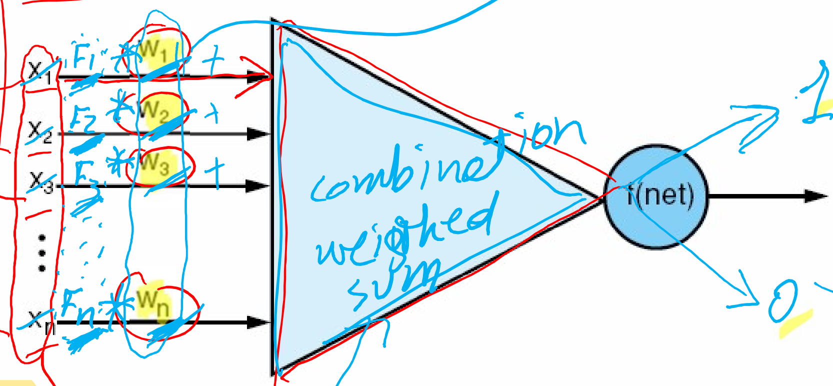


Source: http://www.human-memory.net/brain_neurons.html

A Perceptron = 1 single artificial neuron

will be learned

= n features



- A single computational neuron (no network yet…)
- Input:
  - input signals x_i
  - weights w_i for each feature x_i
    - represents the strength of the connection with the neighboring neurons

⚠ features x_n

- Output:
  - if sum of input weights >= some threshold, neuron fires (output=1)
  - otherwise output = 0
    - If $(w_1 x_1 + \ldots + w_n x_n) >= t$
    - Then output = 1
    - Else output = 0

threshold

- Learning :
  - use the training data to adjust the weights in the perceptron

source: Luger (2005)

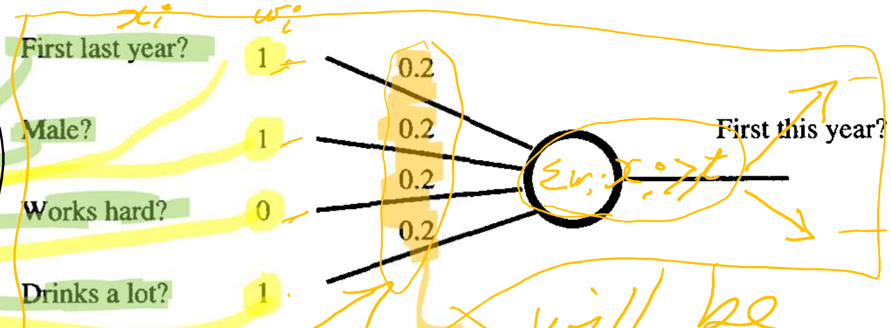
The Idea

4 features

2-class classification

Target class = expected class

Student	Features (x_i)				Output
	First last year?	Male?	Works hard?	Drinks ?	First this year?
Richard	Yes /1	Yes /1	No /0	Yes /1	No /0
Alan	Yes /1	Yes /1	Yes /1	No /0	Yes
… Jason					



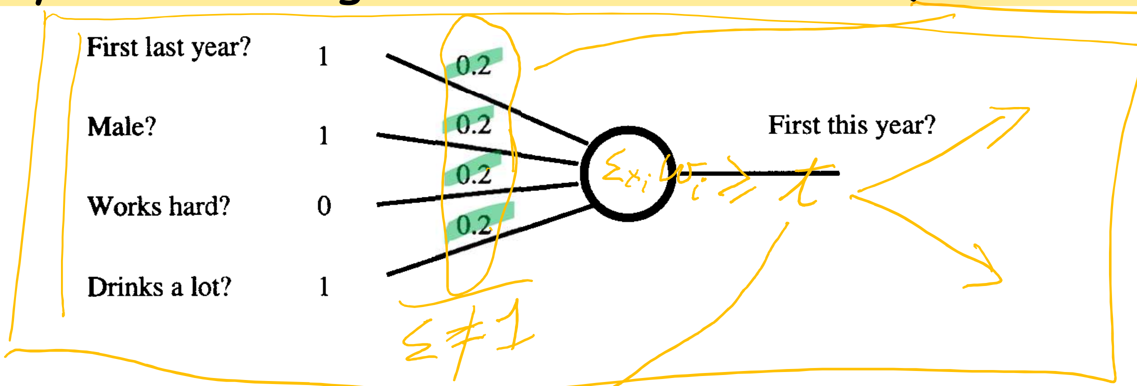
will be learned by the network using the training set

1. Step 1: Set weights to random values
2. Step 2: Feed perceptron with an input
3. Step 3: Compute the network outputs
4. Step 4: Adjust the weights
   1. if output correct → weights stay the same
   2. if output = 0 but it should be 1 →
      - increase weights on *active* connections (i.e. input x_i =1)
   3. if output = 1 but should be 0 →
      - decrease weights on *active* connections (i.e. input x_i =1)
5. Step 5: Repeat steps 2 to 4 a large number of times until the network converges to the right results for the given training examples

source: Cawsey (1998)

A Simple Example

- Turn feature values into numerical values
  - yes -> 1   no -> 0
  - e.g. if x_1 = 1, then student got an A last year
  - e.g. if x_1 = 0, then student did not get an A last year
- Initially, set all weights to random values (all 0.2 here)



- Assume:
  - threshold = 0.55 #1
  - constant learning rate = 0.05 #2

source: Cawsey (1998)

A Simple Example (2)

Target

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
Richard	yes/ 1	1	no/ 0	1	no **0** target T
Alan	yes/ 1	1	1	no/ 0	yes **1**
Alison	0	0	1	0	**0**
Jeff	0	1	0	1	**0**
Gail	1	0	1	1	**1**
Simon	0	1	1	1	**0**

$\sum_{i=1}^{4} x_i w_i \geq t$

0 output

- Richard:
  - (1×0.2) + (1×0.2) + (0×0.2) + (1×0.2) = 0.6 >= 0.55 --> output is 1
    - w_1, w_2, w_3, w_4; non-active
  - ...but he did not get an A this year
  - So reduce weights of all active connections (with input 1) by 0.05.
    Do not change the weight of the inactive connections.

    learning rate
  - So we get w_1= 0.15, w_2= 0.15, w_3= 0.2, w_4= 0.15

Bcs w(3,richard) = 0

A Simple Example (3)

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
~~Richard~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~
Alan	1	1	1	0	1 Target
Alison	0	0	1	0	0
Jeff	0	1	0	1	0
Gail	1	0	1	1	1
Simon	0	1	1	1	0

w_1, w_2 w_3, w_4

- Alan:
  - w_1 w_2 w_3 w_4 $< t$ Output
  - $(1 \times 0.15) + (1 \times 0.15) + (1 \times 0.2) + (0 \times 0.15) = 0.5 < 0.55 \rightarrow$ output is 0
  - non-active
  - … but expected output is 1
  - So increase all weights of active connections by 0.05 constant learning rate
  - So we get $w_1 = 0.2$, $w_2 = 0.2$, $w_3 = 0.25$, $w_4 = 0.15$
- Alison…Jeff… Gail… Simon…

w_1, w_2 w_3 w_4 w_1, w_2 w_3 w_4

A Simple Example (4)

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
~~Richard~~	~~1~~	~~1~~	~~0~~	~~1~~	~~0~~
~~Alan~~	~~1~~	~~1~~	~~1~~	~~0~~	~~1~~
~~Alison~~	~~0~~	~~0~~	~~1~~	~~0~~	~~0~~
~~Jeff~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~
~~Gail~~	~~1~~	~~0~~	~~1~~	~~1~~	~~1~~
~~Simon~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~

- epoch 1: Richard, Alan, Alison, Jeff, Gail, Simon w_1, w_2, w_3, w_4 = 1 iteration over the training set
- epoch 2: Richard, Alan, Alison, Jeff, Gail, Simon
- ...
- epoch n... until all training data points are correctly classified
  - $\boxed{w_1 = 0.25\ w_2 = 0.1\ w_3 = 0.2\ w_4 = 0.1}$

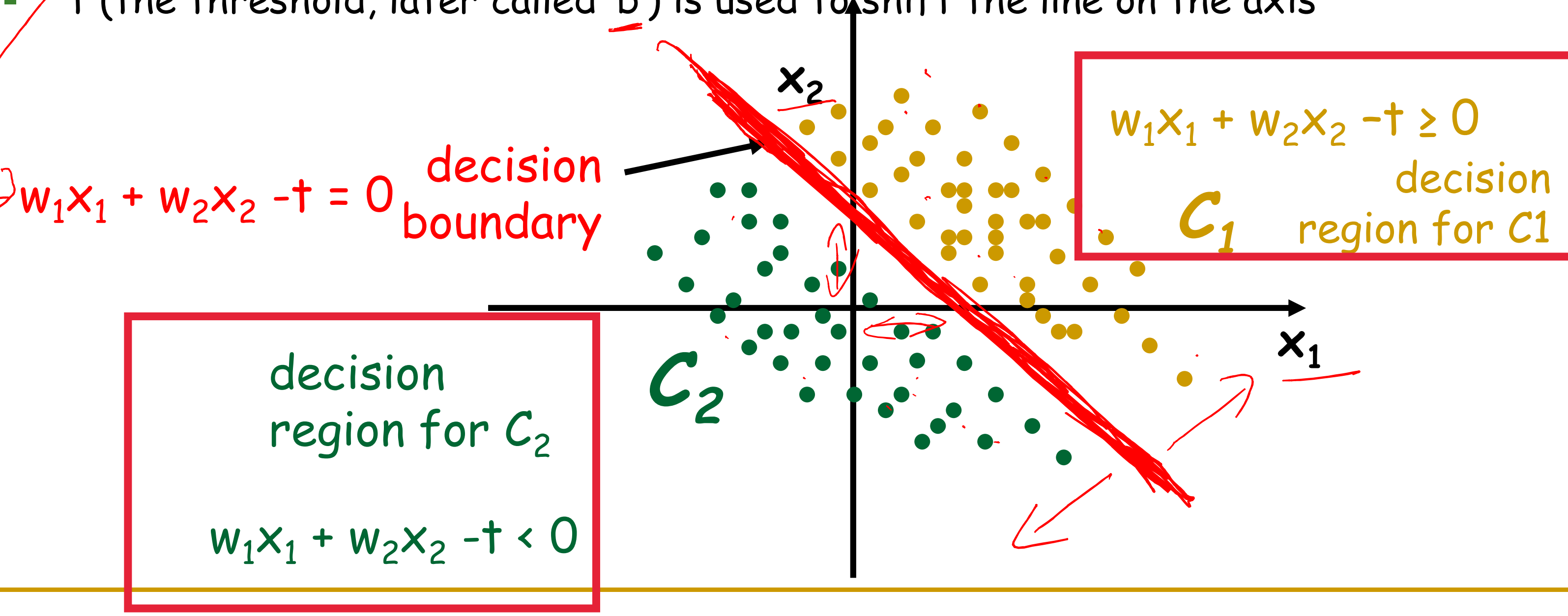
A Simple Example (5)

	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
Richard	1	1	0	1	**0**
Alan	~~1~~ 100	1	~~1~~ 5	0	**1**
Alison	0	0	1	0	**0**
Jeff	0	1	0	1	**0**
Gail	1	0	1	1	**1**
Simon	0	1	1	1	**0**

- Let's check… $(w_1 = 0.2\ w_2 = 0.1\ w_3 = 0.25\ w_4 = 0.1)$
  - Richard: $(1\times0.2) + (1\times0.1) + (0\times0.25) + (1\times0.1) = 0.4 < 0.55$ -> output is 0 ✓
  - Alan: $(1\times0.2) + (1\times0.1) + (1\times0.25) + (0\times0.1) = 0.55 \geq 0.55$ -> output is 1 ✓
  - Alison: $(0\times0.2) + (0\times0.1) + (1\times0.25) + (0\times0.1) = 0.25 < 0.55$ -> output is 0 ✓
  - Jeff: $(0\times0.2) + (1\times0.1) + (0\times0.25) + (1\times0.1) = 0.2 < 0.55$ -> output is 0 ✓
  - Gail: $(1\times0.2) + (0\times0.1) + (1\times0.25) + (1\times0.1) = 0.55 \geq 0.55$ -> output is 1 ✓
  - Simon: $(0\times0.2) + (1\times0.1) + (1\times0.25) + (1\times0.1) = 0.45 < 0.55$ -> output is 0 ✓

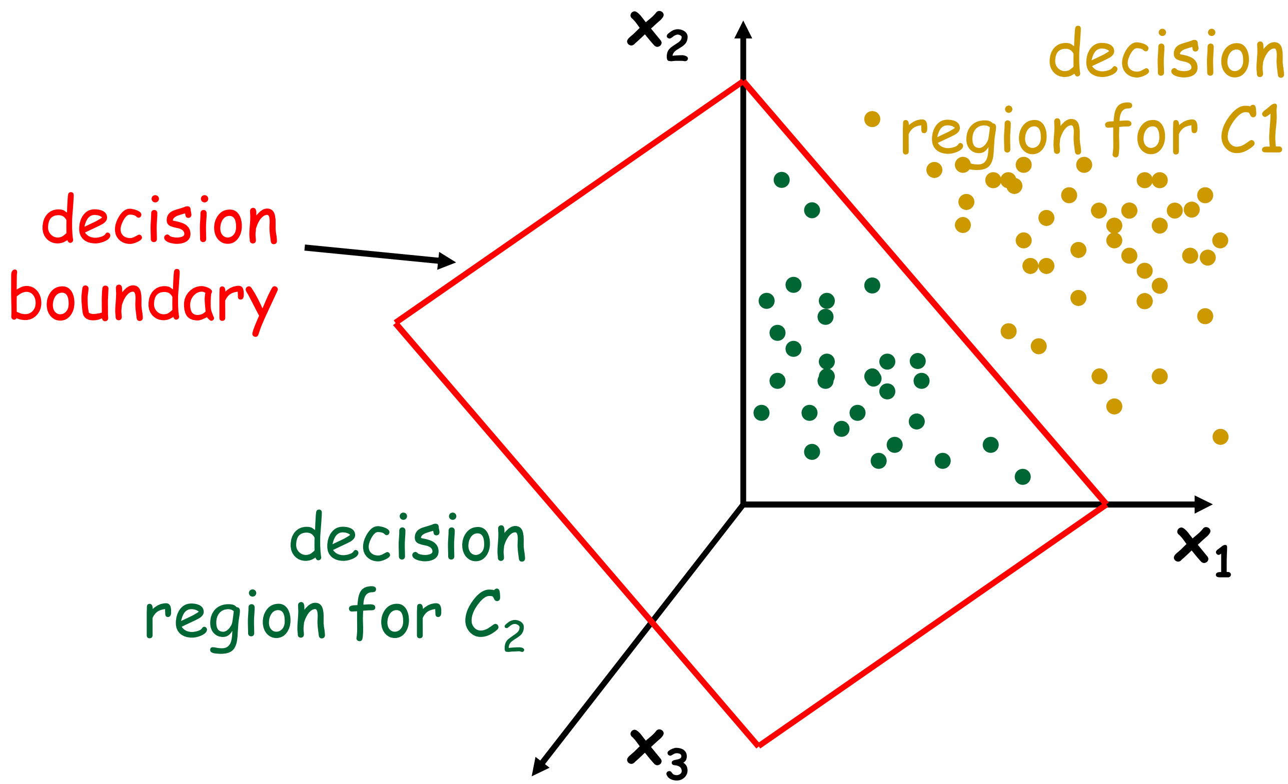
Decision Boundaries of Perceptrons

- So we have just learned the function: t class 1 class 0
  - If $(0.2x_1 + 0.1x_2 + 0.25x_3 + 0.1x_4 \geq 0.55)$ then 1 otherwise 0
  - If $(0.2x_1 + 0.1x_2 + 0.25x_3 + 0.1x_4 - 0.55 \geq 0)$ then 1 otherwise 0
- Assume we only had 2 features:
  - If $(w_1x_1 + w_2x_2 - t >= 0)$ then 1 otherwise 0
  - The learned function describes a line in the input space
  - This line is used to separate the two classes C1 and C2
  - t (the threshold, later called 'b') is used to shift the line on the axis



Decision Boundaries of Perceptrons

- More generally, with n features, the learned function describes a hyperplane in the input space.

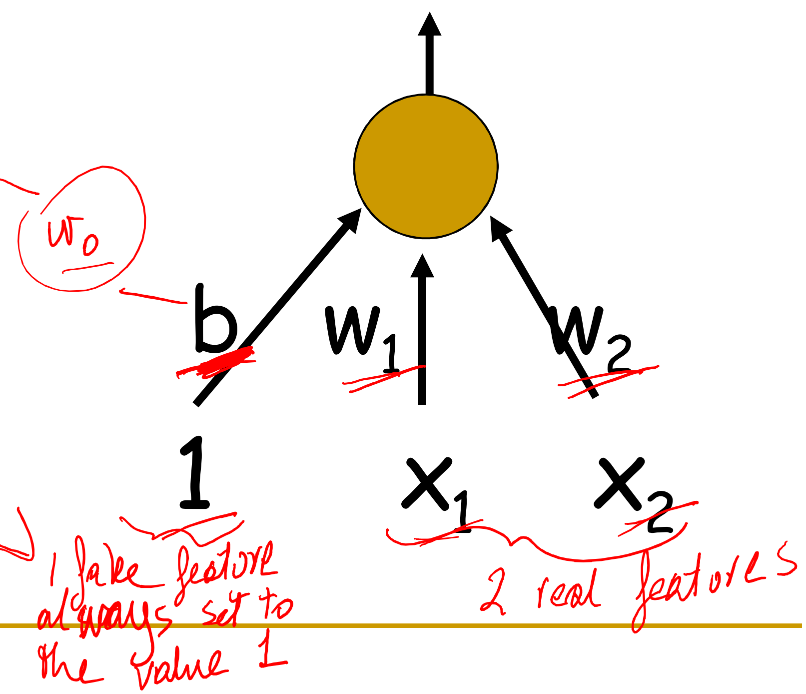


Adding a Bias #1

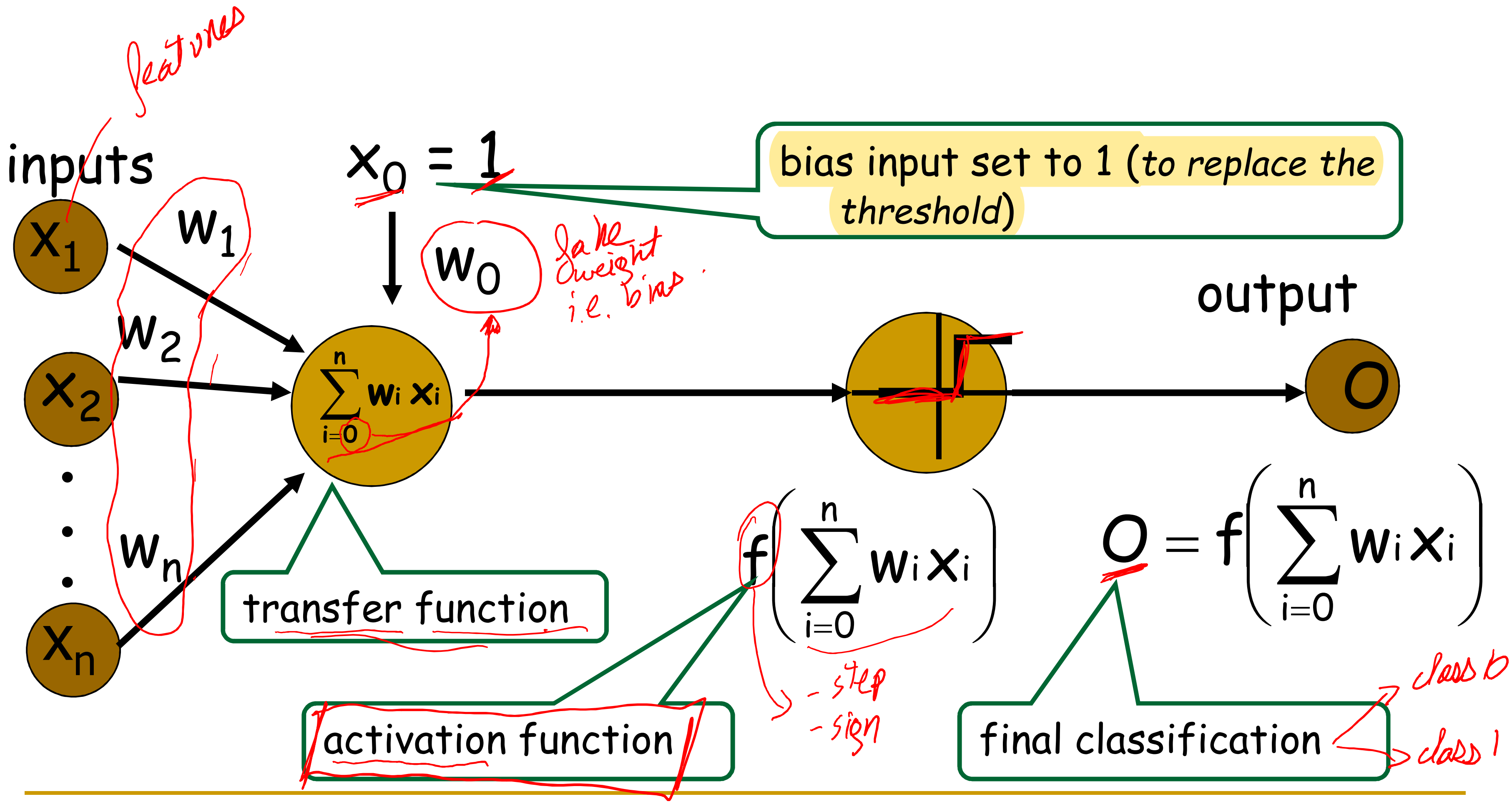
- We can avoid having to figure out the threshold by using a "bias"

$$b + \boxed{\sum_i x_i w_i}$$

- A bias is equivalent to a weight on an extra input feature that always has a value of 1.



Perceptron - More Generally



Common Activation Functions

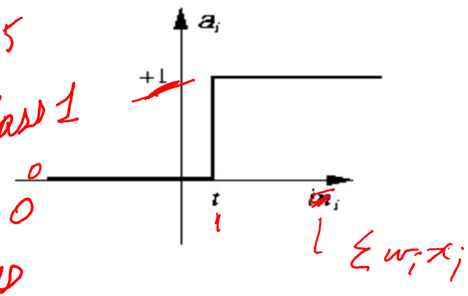
threshold

- **step**

$$O = \begin{cases} if \left(\sum_{i=1}^{n} w_i x_i \right) \geq t \rightarrow 1 & \text{// class 1} \\ otherwise \rightarrow 0 & \text{// class 0} \end{cases}$$

eg. 0.55

// real features

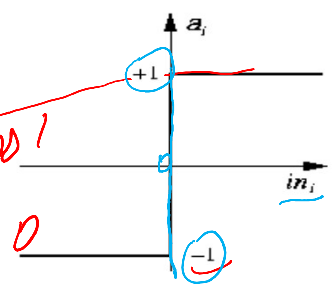


(a) Step function

value is not important

- **sign**

$$O = \begin{cases} if \left(\sum_{i=0}^{n} w_i x_i \right) \geq 0 \rightarrow 1 & \text{// class 1} \\ otherwise \rightarrow -1 & \text{// class 0} \end{cases}$$



(b) Sign function

merged together into

$$O = \begin{cases} if \left(\sum_{i=0}^{n} w_i x_i \right) > 0) \rightarrow 1 \\ if \left(\sum_{i=0}^{n} w_i x_i \right) = 0) \rightarrow 0 \\ otherwise \rightarrow -1 \end{cases}$$

official sign function
3-classes

used in the perceptron

[Russell & Norvig, 1995]

Learning Rate #2

so far

1. Learning rate can be a constant value (as in the previous example)

$$\Delta w = \eta(T - O)$$

learning rate eta =0.05

target output

error

Error = target output – actual output

1-0 = 1

- So:
  - if T=zero and O=1 (i.e. a false positive) -> decrease w by η
  - if T=1 and O=zero (i.e. a false negative) -> increase w by η
  - if T=O (i.e. no error) -> don't change w

delta rule

2. Or, a fraction of the input feature x_i

$$\boxed{\Delta w_i = \eta(\underbrace{T - O}_{\text{error}})\underbrace{x_i}_{0.1}}$$

value of input feature x_i

- So the update is proportional to the value of x
  - if T=zero and O=1 (i.e. a false positive) -> decrease w_i by ηx_i
  - if T=1 and O=zero (i.e. a false negative) -> increase w_i by ηx_i
  - if T=O (i.e. no error) -> don't change w_i
- This is called the delta rule or perceptron learning rule

Perceptron Convergence Theorem

- If a solution with zero error exists
  - i.e. the training data are linearly separable
- The delta rule will find a solution in finite time

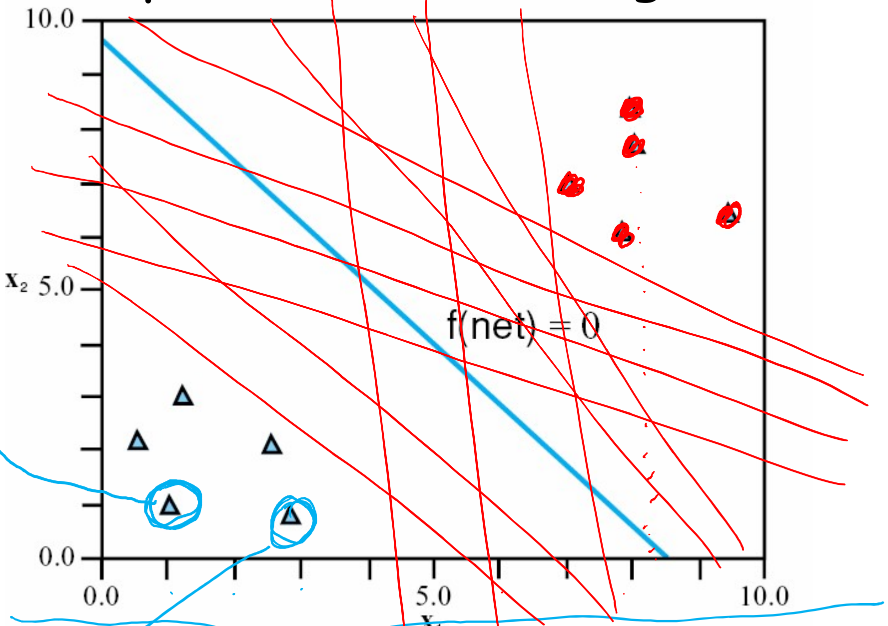
Example of the Delta Rule

- training data: features

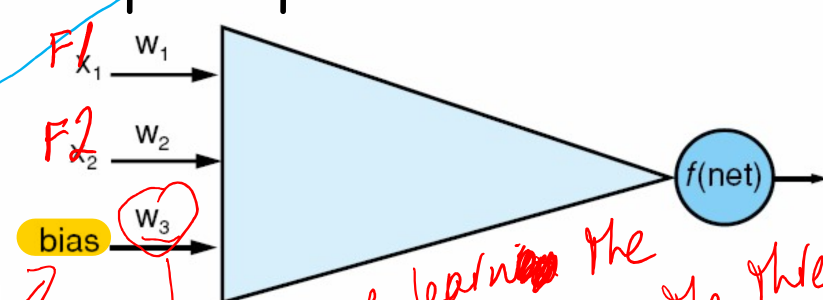
x_1	x_2	x_3	~~Output~~ Target
1.0	1.0	1.0	1
9.4	6.4	1.0	−1
2.5	2.1	1.0	1
8.0	7.7	1.0	−1
0.5	2.2	1.0	1
7.9	8.4	1.0	−1
7.0	7.0	1.0	−1
2.8	0.8	1.0	1
1.2	3.0	1.0	1
7.8	6.1	1.0	−1

F1 F2 F3 bias

- plot of the training data:



- perceptron

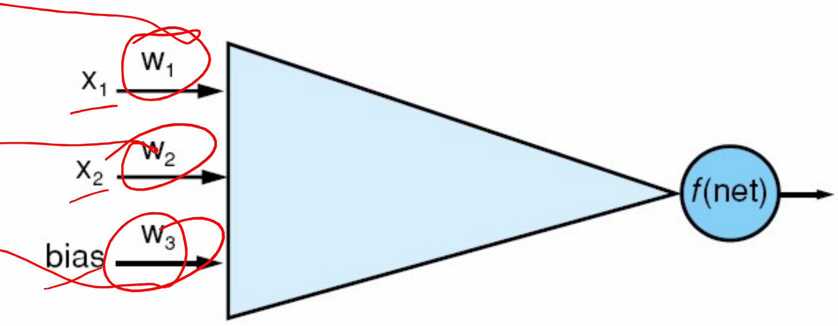


= we learning the bias = the threshold

source: Luger (2005)

Example of the Delta Rule

- assume random initialization
  - w1 = 0.75
  - w2 = 0.5
  - w3 = -0.6



- Assume:
  - sign function (threshold = 0)
  - learning rate η = 0.2 using the delta rule

source: Luger (2005)

Example of the Delta Rule

activation function (sign) → transfer function $\sum_{i=0}^{n} x_i w_i$

- data #1: f(0.75x1 + 0.5x1 - 0.6x1) = f(0.65) -> 1 ✓
- data #2: f(0.75x9.4 + 0.5x6.4 - 0.6x1) = f(9.65) -> 1 ×
  - --> error = (-1 - 1) = -2
  - --> $w_1 = w_1$ -2x0.2x9.4
    - = 0.75 - 3.76 = -3.01 new w_1
  - --> $w_2 = w_2$ -2x0.2x6.4 = -2.06 new w_2
  - --> $w_3 = w_3$ -2x0.2x1 = -1.00 new w_3
- data #3: f(-3.01x2.5 -2.06x2.1 -1x1) = f(-12.84) -> -1 ×
  - --> error = (1 - -1) = 2
  - Delta formula
  - --> w_1 = -3.01 + 2x0.2x2.5 = -2.01
  - --> w_2 = -2.06 + 2x0.2x2.1 = -1.22
  - --> w_3 = -1.00 + 2x0.2x1 = -0.60
- repeat… over 500 iterations, we converge to:
  w_1 = -1.3 w_2 = -1.1 w_3 = 10.9

x_1	x_2	bias	~~Output~~ Target
1.0	1.0	1	1
9.4	6.4	1	−1
2.5	2.1	1	1
8.0	7.7	1	−1
0.5	2.2	1	1

error T−O = 1−1 = 0 ⇒ weights do not change

error: T−O = −1−1 = −2

Output T−O = 1−(−1) = 2

learning rate

source: Luger (2005)

The Perceptron in 1958

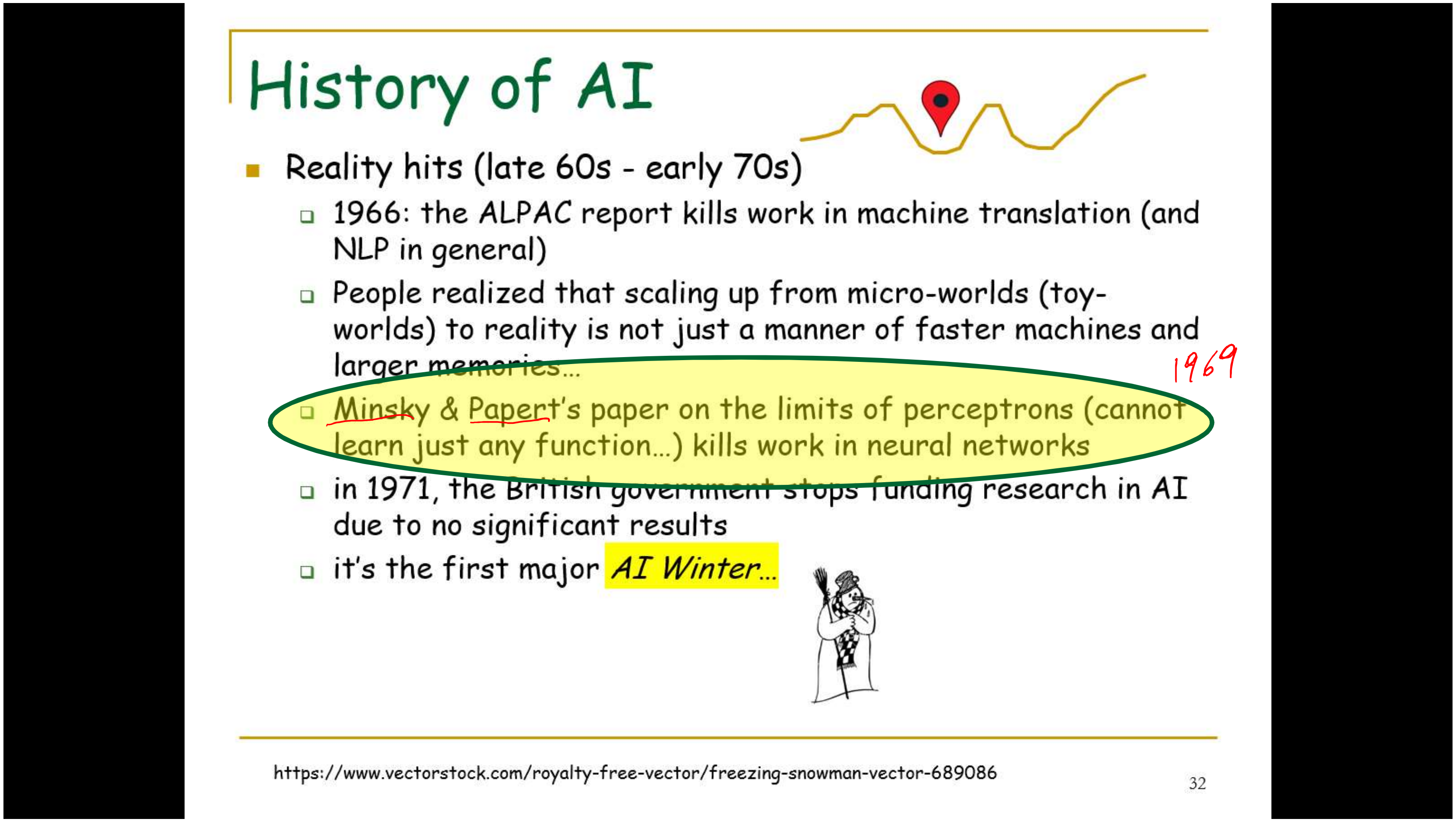
1st perceptron

Frank Rosenblatt

An IBM 704 – a 5-ton computer the size of a room – was fed a series of punch cards. After 50 trials, the computer taught itself to distinguish cards marked on the left from cards marked on the right.

https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon

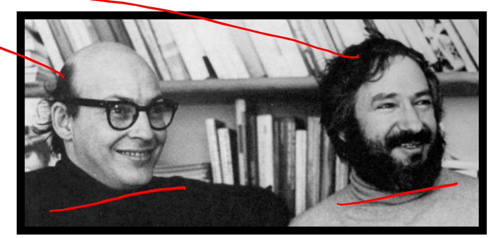
Remember this slide?

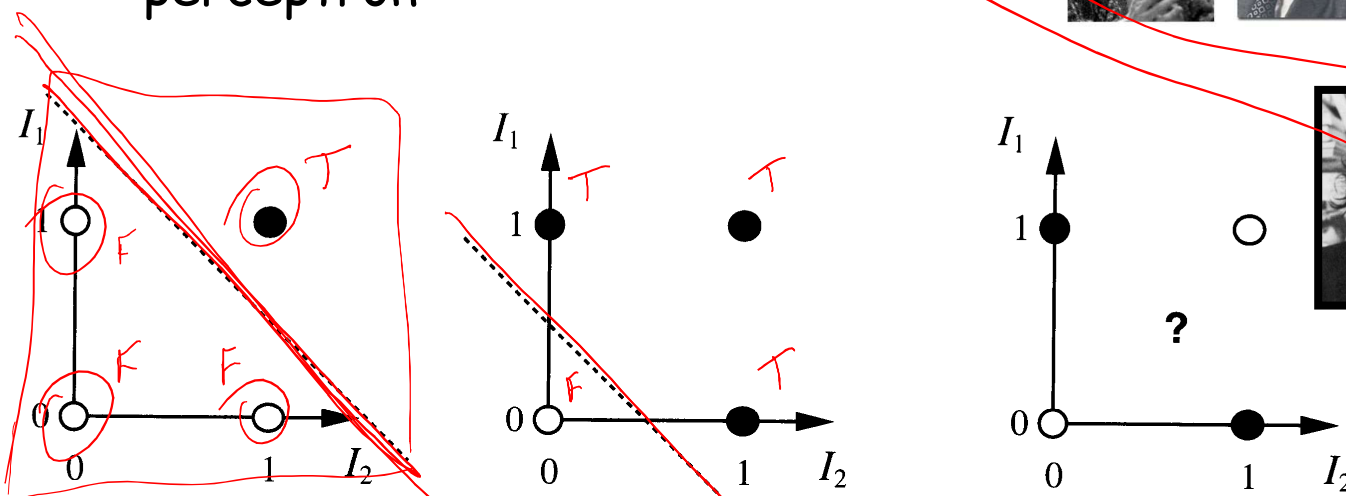


Limits of the Perceptron

pro: symbolic computation eg. Prolog, Lisp, GOFAI

- In 1969, Minsky and Papert showed formally what functions could and could not be represented by perceptrons
- Only linearly separable functions can be represented by a perceptron



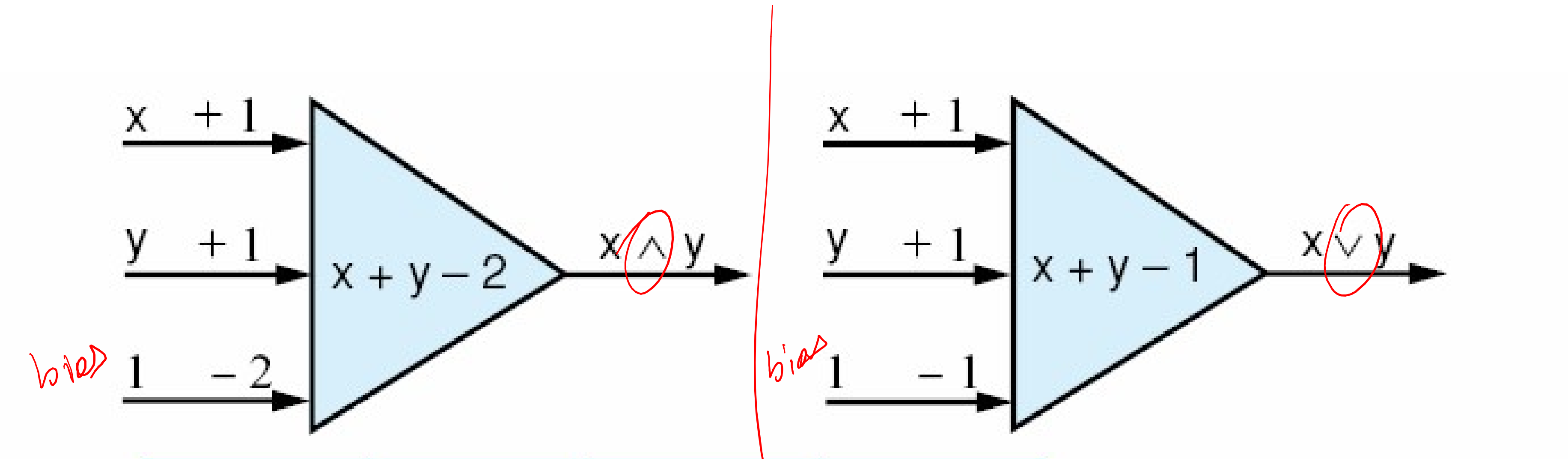


(a) I_1 and I_2 (b) I_1 or I_2 (c) I_1 xor I_2

F1 F2

source: Luger (2005)

AND and OR Perceptrons

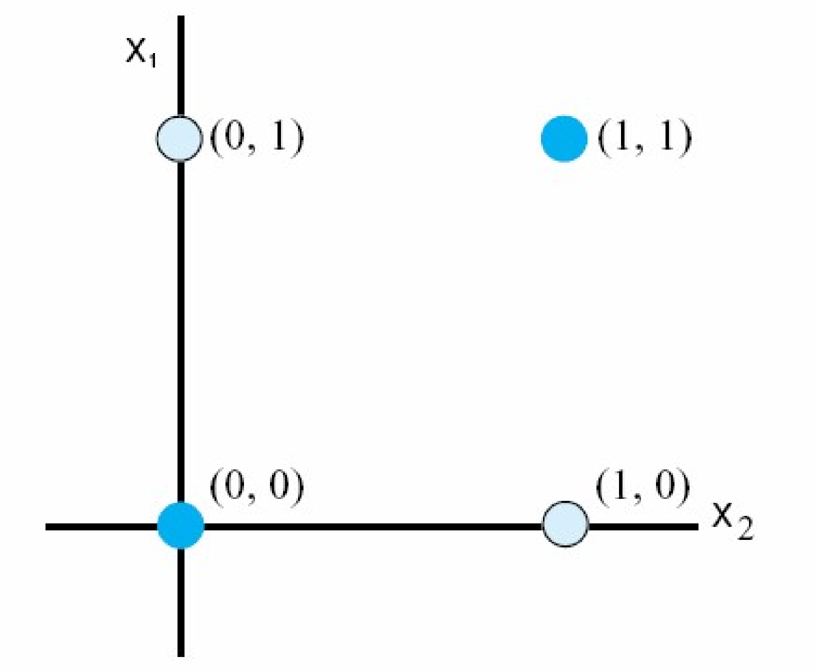


x	y	x + y − 2	Output
1	1	0	1
1	0	−1	−1
0	1	−1	−1
0	0	−2	−1

source: Luger (2005)

The XOR Function - Visually

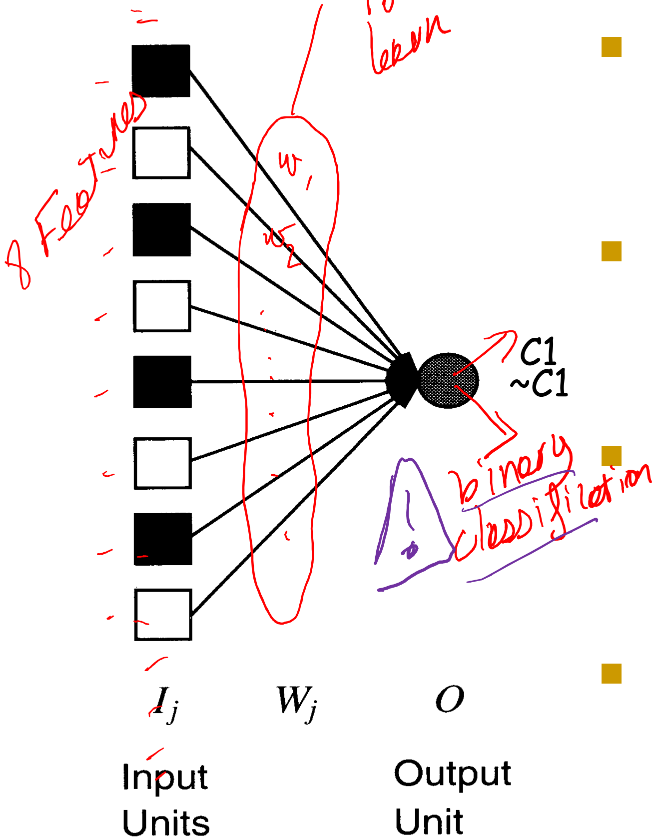
- In a 2-dimentional space (2 features for the X)
- No straight line in two-dimensions can separate
  - (0, 1) and (1, 0) from
  - (0, 0) and (1, 1).



source: Luger (2005)

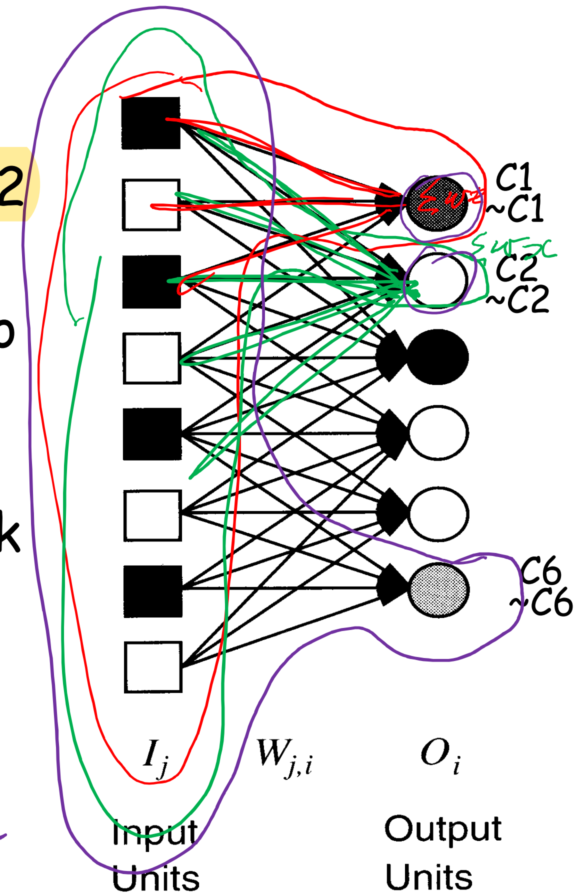
A Perceptron Network

weights To learn i.e The parameters



Single Perceptron

- A perceptron is a binary classifier (i.e. 2 classes)
- if the output needs to learn more than a binary decision
- we can have a network of perceptrons
- Eg: learning to recognize digit --> 10 possible outputs --> need a perceptron network

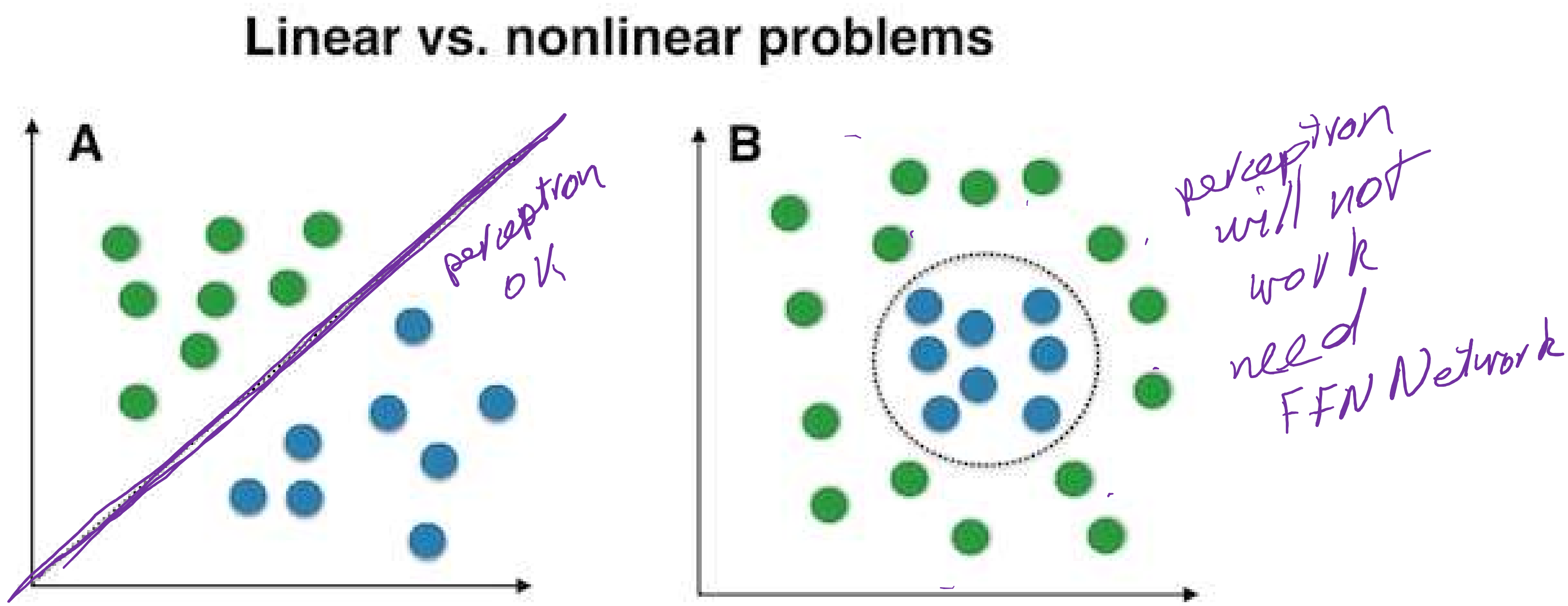


Perceptron Network

n

Non-Linearly Separable Functions

- Real-world problems cannot always be represented by linearly-separable functions…



- This caused a decrease in interest in neural networks in the 1970's 1980's

http://sebastianraschka.com/Articles/2014_kernel_pca.html

Today

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks video #8.

Up Next

1. Introduction to ML
2. Naïve Bayes Classification
   a. Application to Spam Filtering
3. Decision Trees
4. (Evaluation
5. Unsupervised Learning)
6. Neural Networks
   a. Perceptrons
   b. Multi Layered Neural Networks