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

- Russell & Norvig: Sections 19.1, 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

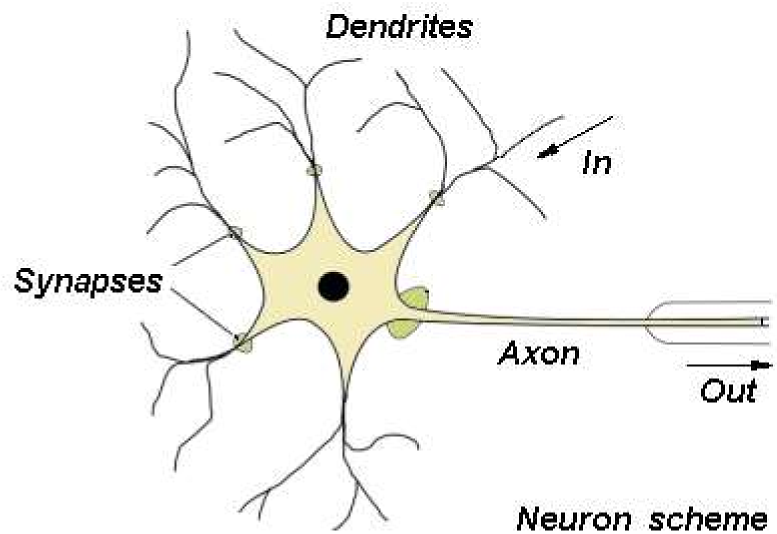
Neural Networks

- Learning approach inspired by biology
  - the neurons in the human brain
    神经元
- Set of many simple processing units (called neurons) connected together 许多简单处理单元在一起，组成一张网
  - the behavior of each neuron is very simple
  - but a network of neurons can have sophisticated behavior and can be used for complex tasks 复杂的
  - neurons are connected to each other to form a network
  - the strength of the connection between neurons are determined by weights connection的强度取决于neurons的重要性
  - the network learns by learning the weights between neurons given training data网络通过学习给定训练数据的神经元之间的权值进行学习
- Different types of network architectures exist
  - feed forward neural networks (FFNN)
  - recurrent neural networks (RNN)
  - convolutional neural networks (CNN)
  - ...

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**
    突触

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



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

Behavior of a Neuron

neuron从nabour收取信息

- A neuron receives inputs from its neighbors
- If enough inputs are received at the same time:

  如果同时收集到足够input

  - the neuron is activated 这个neuron被激活
  - and fires an output to its neighbors fire output给其他neighbor

  突触

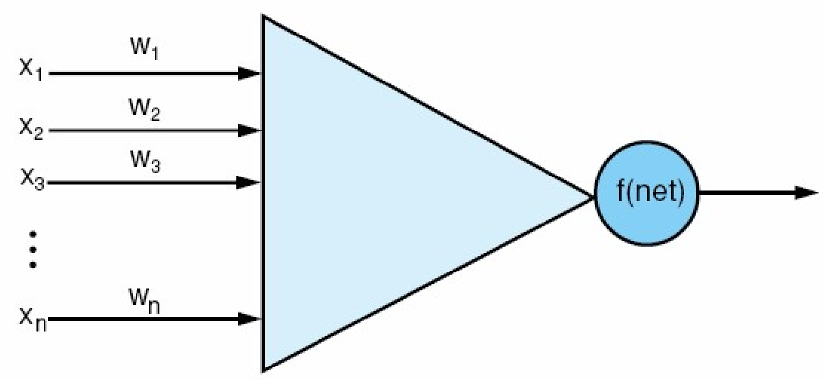
- Repeated firings across a synapse increases its sensitivity and the future likelihood of its firing

  如果总是从同一个突触发出信号，增长他的敏感性和未来firing的可能性

- If a particular stimulus repeatedly causes activity in a group of neurons, they become strongly associated

  刺激

A Perceptron 感知



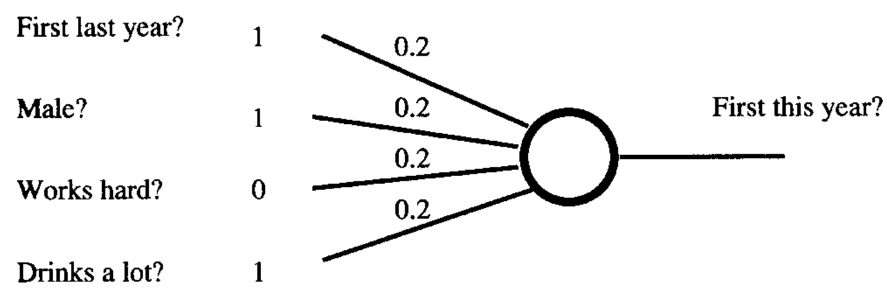
一个简单的计算单元

- A <u>single</u> computational neuron (no network yet...)
- Input:
  - input signals x_i 输入多个信号signal
  - weights w_i for each feature x_i 对每个xi进行称重
    - represents the strength of the connection with the neighboring neurons
- Output:
  如果大于某个限制weight，fire,output=1
  - 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
- Learning : 使用training data来调整perception里的weights
  - use the training data to adjust the weights in the perceptron

source: Luger (2005)

The Idea

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



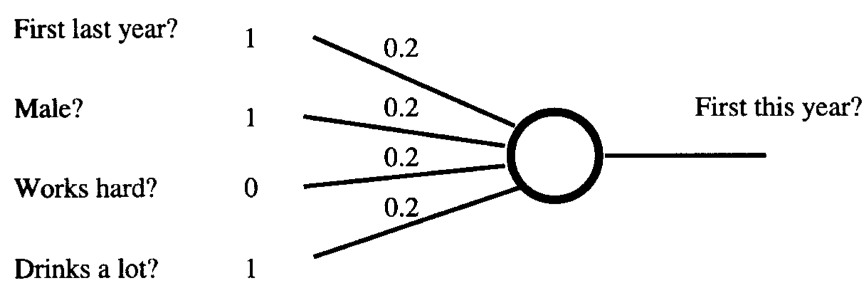
1. Step 1: Set weights to random values 第一步随机同一个weight
2. Step 2: Feed perceptron with an input 给perception一个input
3. Step 3: Compute the network outputs 计算output
4. Step 4: Adjust the weights 调整weight
   1. if output correct → weights stay the same 如果Outpu正确，不变
   2. if output = 0 but it should be 1 → 如果是0，而原来标了1，那么active connection也就是为1的权重增加
      - increase weights on *active* connections (i.e. input x_i =1)
   3. if output = 1 but should be 0 → 如果是1标了0，active的权重减少
      - 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

疯狂重复24，直到network收敛到一个正确结果

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: 假设限制是0.55
  - threshold = 0.55
  - constant learning rate = 0.05

学习率是0.05

source: Cawsey (1998)

A Simple Example (2)

	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**

- Richard:

  大于，所以要reduce，reduce率是0.05

  - $(1 \times 0.2) + (1 \times 0.2) + (0 \times 0.2) + (1 \times 0.2) = 0.6 >= 0.55 \dashrightarrow$ output is 1
  - …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.
  - So we get $w_1 = 0.15$, $w_2 = 0.15$, $w_3 = 0.2$, $w_4 = 0.15$

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**
Alison	0	0	1	0	**0**
Jeff	0	1	0	1	**0**
Gail	1	0	1	1	**1**
Simon	0	1	1	1	**0**

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

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
- epoch 2: Richard, Alan, Alison, Jeff, Gail, Simon
- ...
- epoch n... until all training data points are correctly classified
  - w_1= 0.25 w_2= 0.1 w_3= 0.2 w_4= 0.1

最后这样，直到我们满足所有Output

A Simple Example (5)

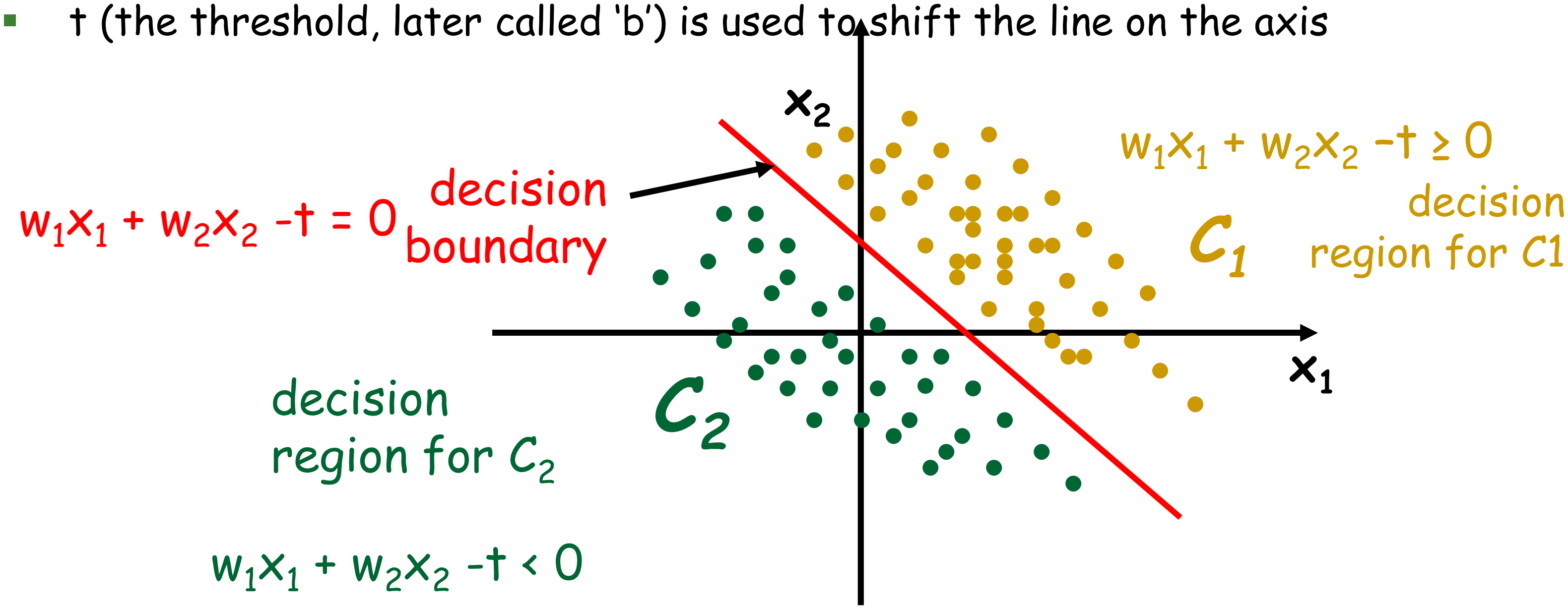
	Features (x_i)				Output
Student	'A' last year?	Male?	Works hard?	Drinks?	**'A' this year?**
Richard	1	1	0	1	**0**
Alan	1	1	1	0	**1**
Alison	0	0	1	0	**0**
Jeff	0	1	0	1	**0**
Gail	1	0	1	1	**1**
Simon	0	1	1	1	**0**

- 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:
  - 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

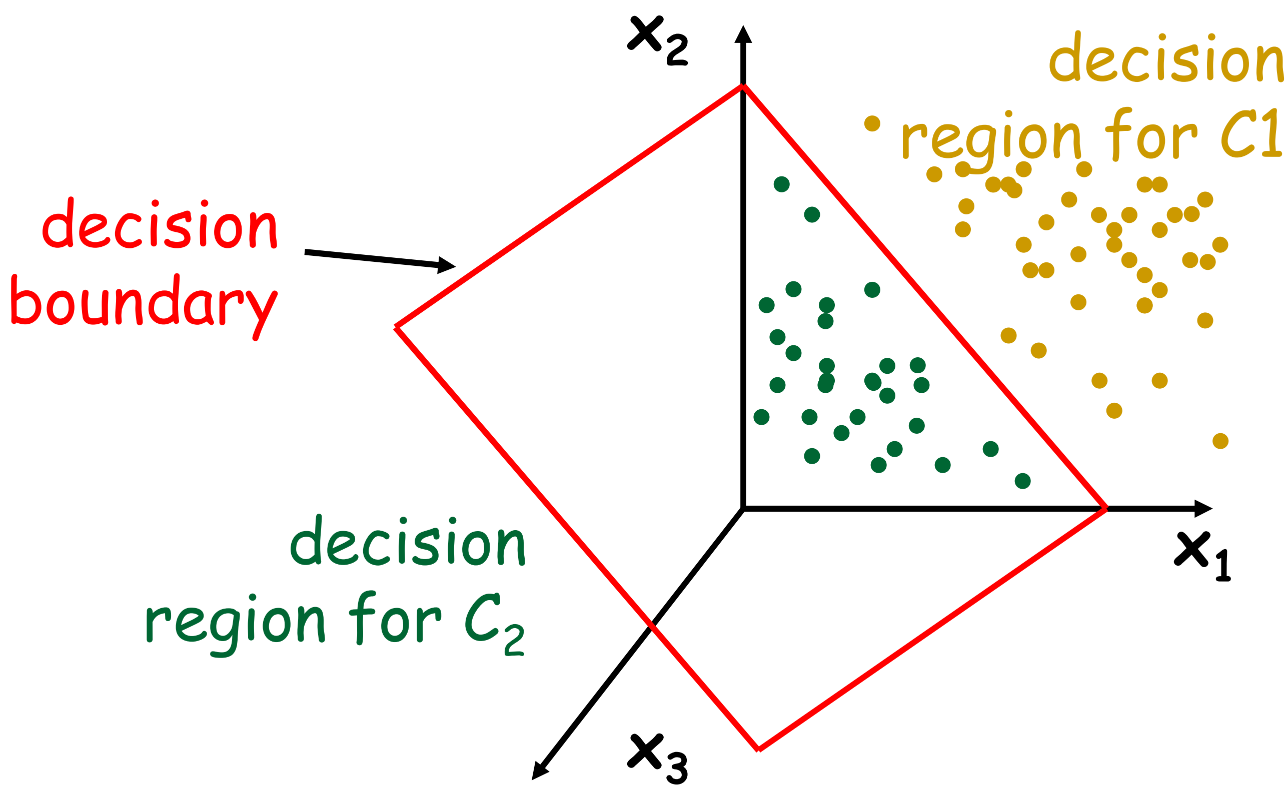
假如只有两个input，我们会发现实际上画了一条直线，而分界点就是t，决定了两个轴的值。



Decision Boundaries of Perceptrons

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

如果增加输入系数，会形成一个hyperplane，一个面



Adding a Bias

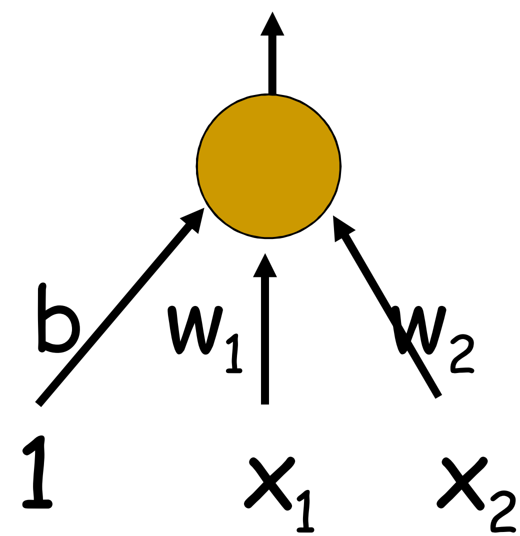
- We can avoid having to figure out the threshold by using a “bias”

$$b + \sum_i x_i w_i$$

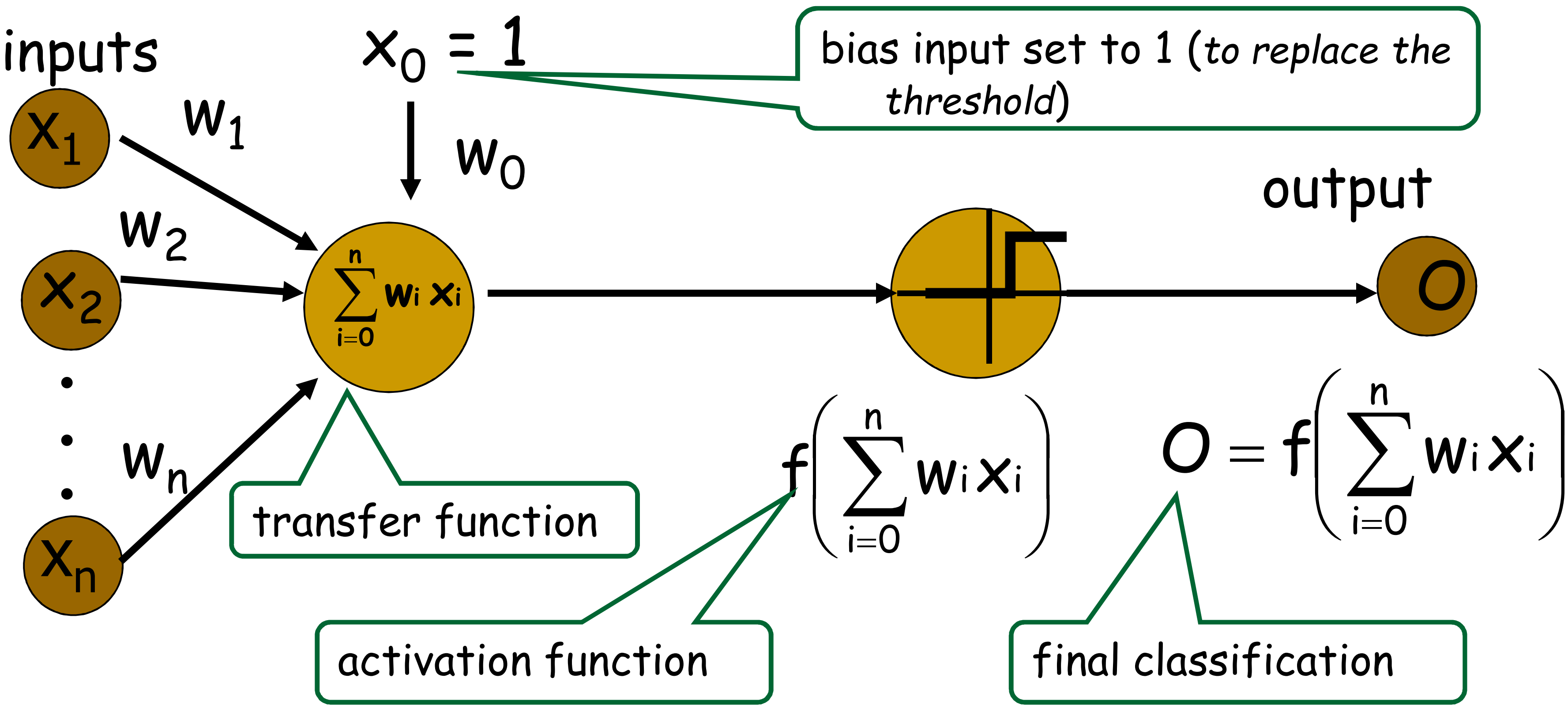
我们可以使用bias来避免找不到threshold分界点

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

bias是一个输入总为1的额外input feature



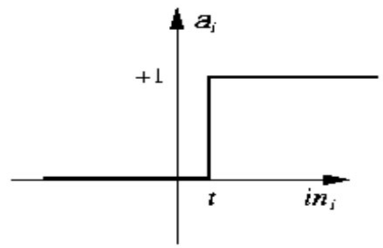
Perceptron - More Generally



Common Activation Functions

- step

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

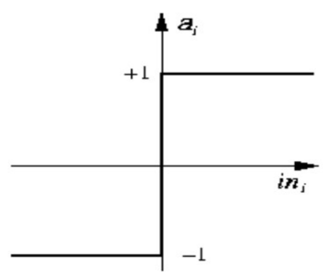


(a) Step function

判断function的方法，step法就是累加超过某个临界点是1
不然就是0

- sign

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



(b) Sign function

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

sign法就是等于等于0就是1
小于0就是-1

[Russell & Norvig, 1995]

Learning Rate

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

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

learning rate

Error = target output – actual output

learning rate可以是一个常量，

- 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

这种就是标准的

2. Or, a fraction of the input feature x_i

$$\Delta w_i = \eta(T - O)\, x_i$$

value of input feature x_i

改变取决于这一次输入Xi本身的值

- 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 就是多乘一个Xi
  - 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

收敛定律

- If a solution with zero error exists 如果存在一个0error解
  - i.e. the training data are linearly separable
- The delta rule will find a solution in finite time

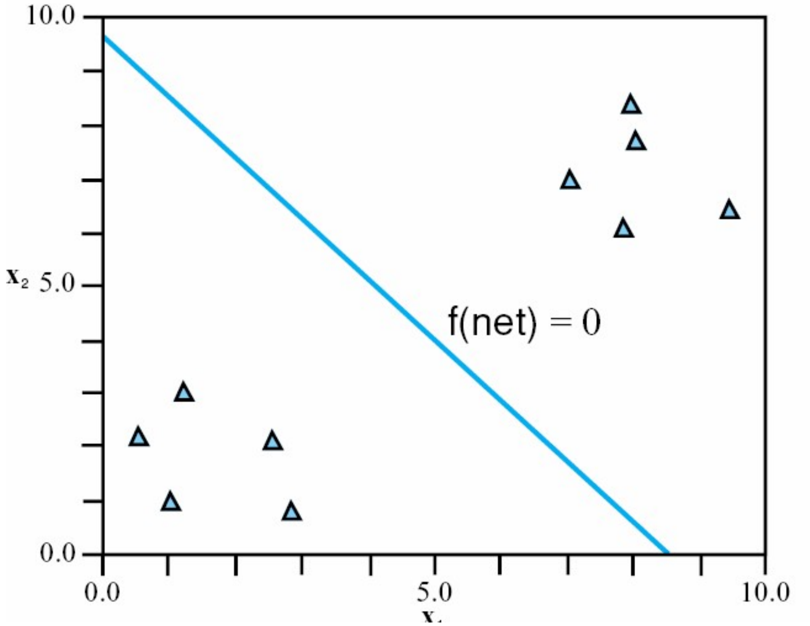
那么delta rule将会在有限时间内找到一个解

Example of the Delta Rule

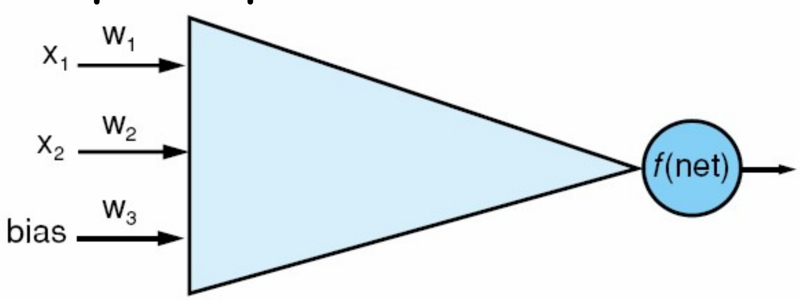
- training data:

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

- plot of the training data:



- perceptron



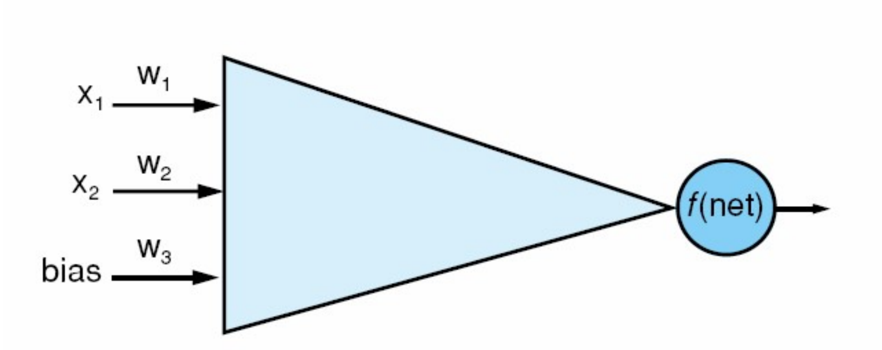
source: Luger (2005)

Example of the Delta Rule

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

随机initialization，
使用sign function

W是比值



- Assume:
  - sign function (threshold = 0)
  - learning rate η = 0.2

source: Luger (2005)

Example of the Delta Rule

- 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

    实际label-输出label

    --> $w_1 = w_1$ -2x0.2x9.4 x的系数

    = 0.75 - 3.76 = -3.01

    --> $w_2 = w_2$ -2x0.2x6.4 = -2.06

    --> $w_3 = w_3$ -2x0.2x1 = -1.00

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

- data #3: f(-3.01x2.5 -2.06x2.1 -1x1) = f(-12.84) -> -1 ×
  - --> error = (1 - -1) = 2

    --> w_1 = -3.01 + 2x0.2x2.5 = -2.01

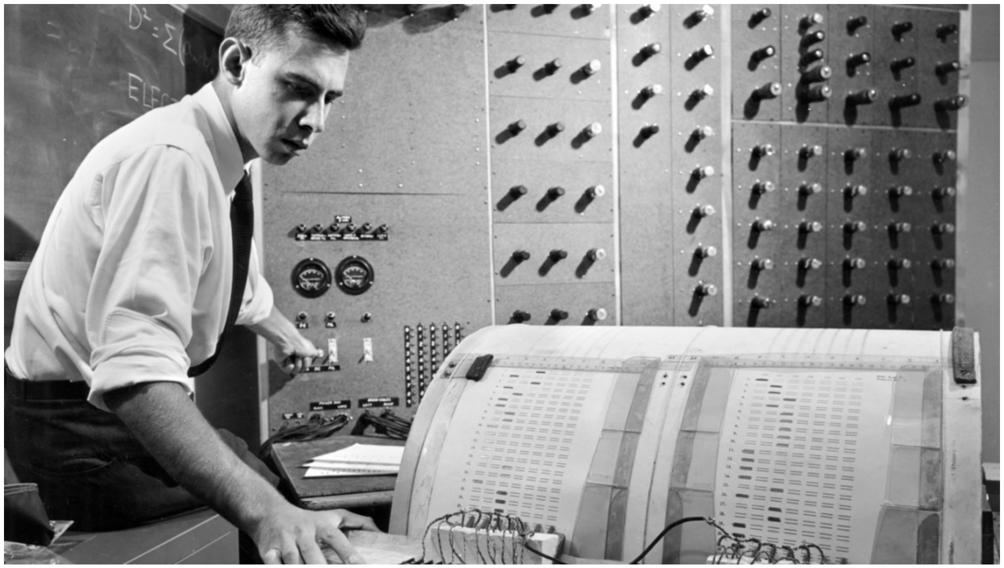
    --> w_2 = -2.06 + 2x0.2x2.1 = -1.22

    --> 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

source: Luger (2005)

The Perceptron in 1958

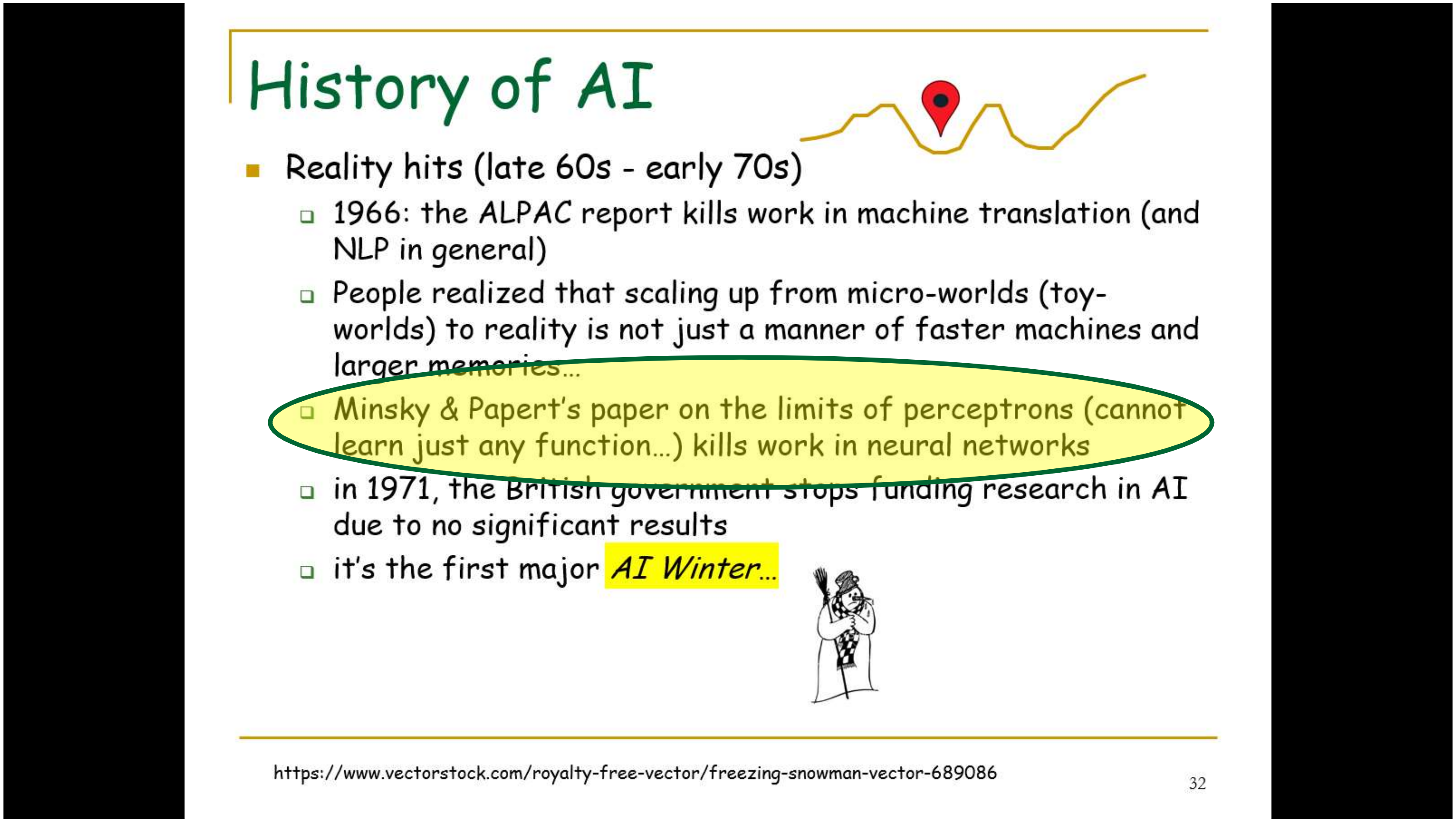


Frank Rosenblatt

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

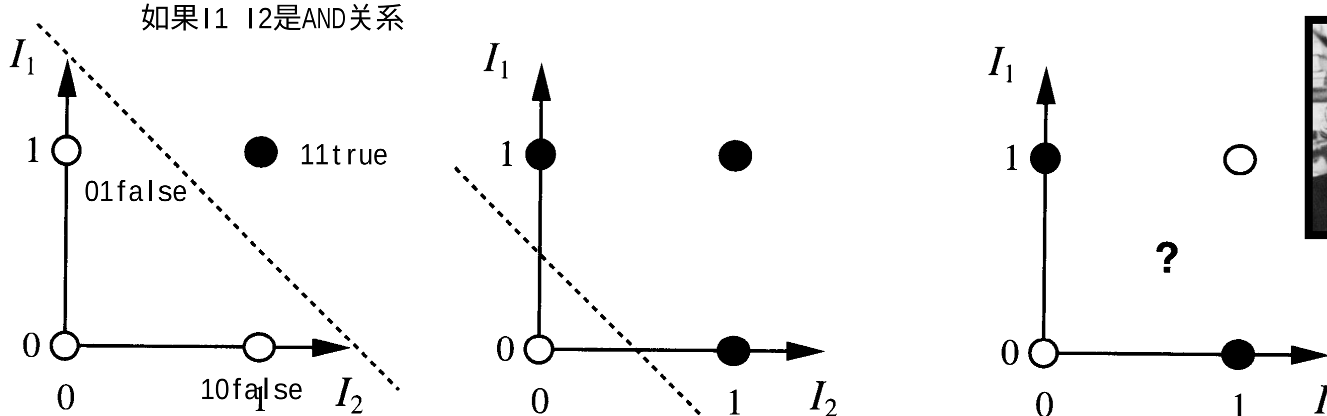
70年代转向symbolic，结果symbolic也dead了

- In 1969, Minsky and Papert showed formally what functions could and could not be represented by perceptrons

只有linearly separable function可以用perception表示

- Only linearly separable functions can be represented by a perceptron

如果I1 I2是AND关系



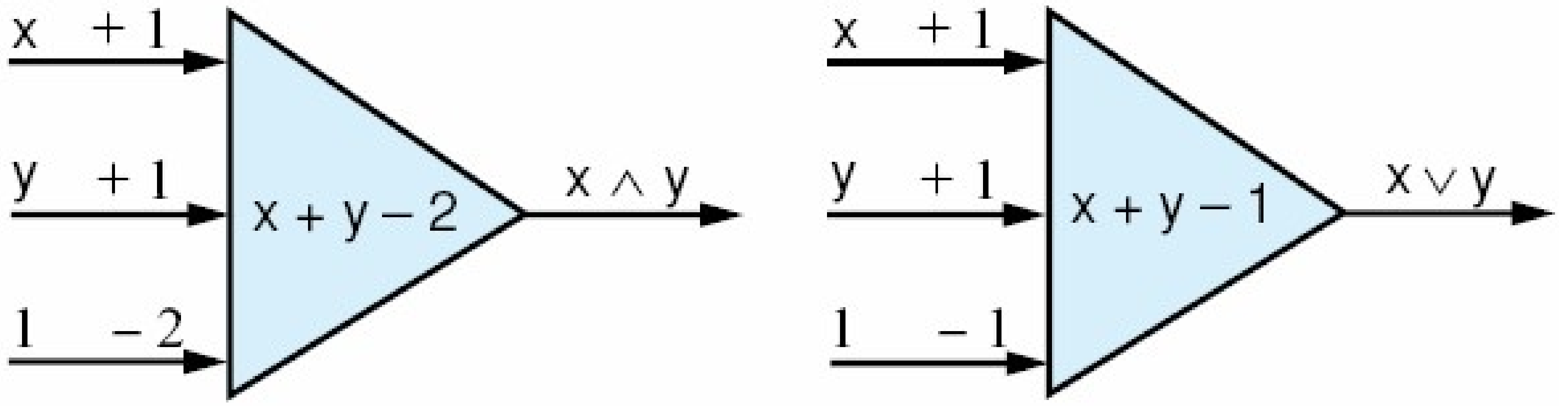
但如果是XOR关系，我们找不到这条线

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

source: Luger (2005)

限制了神经网络的范围

AND and OR Perceptrons

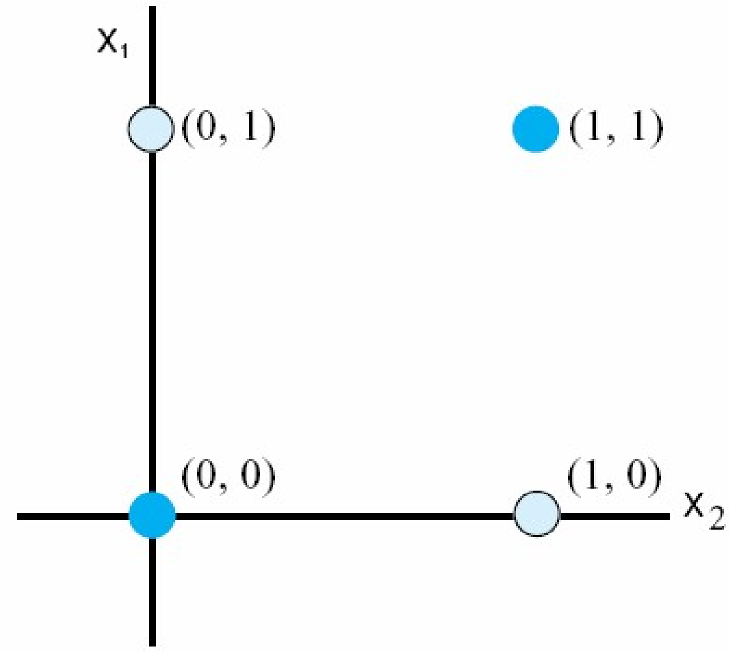


x	y	$x + y - 2$	Output
1	1	0	1
1	0	−1	−1
0	1	−1	−1
0	0	−2	−1

source: Luger (2005)

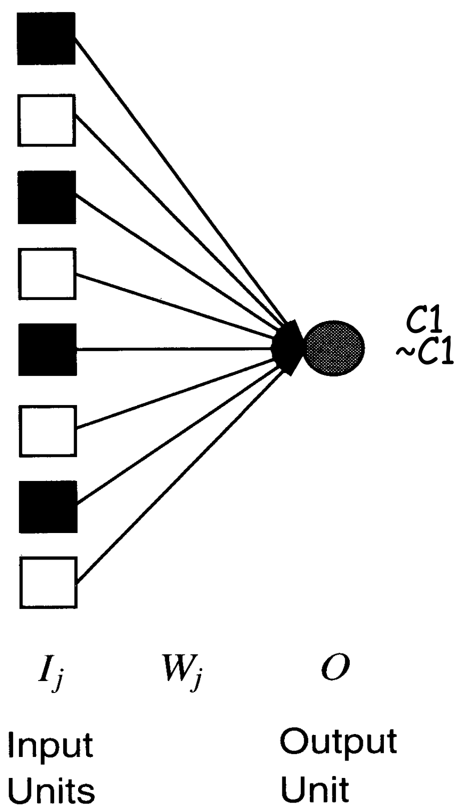
The XOR Function - Visually

- 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)

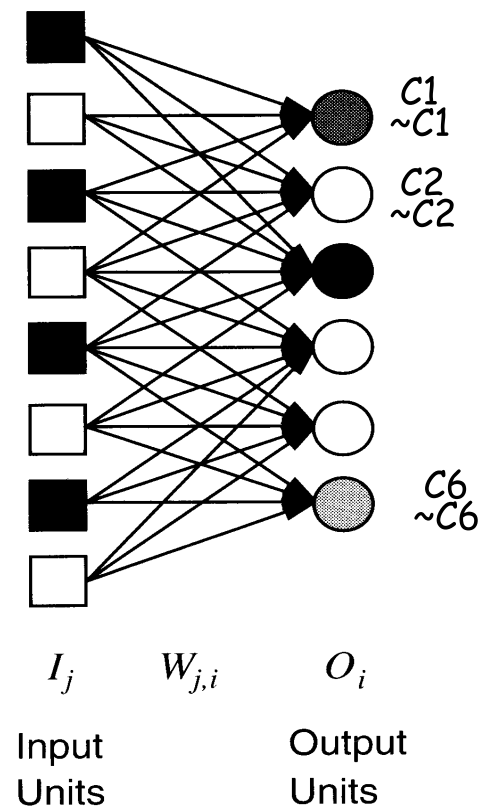
A Perceptron Network



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

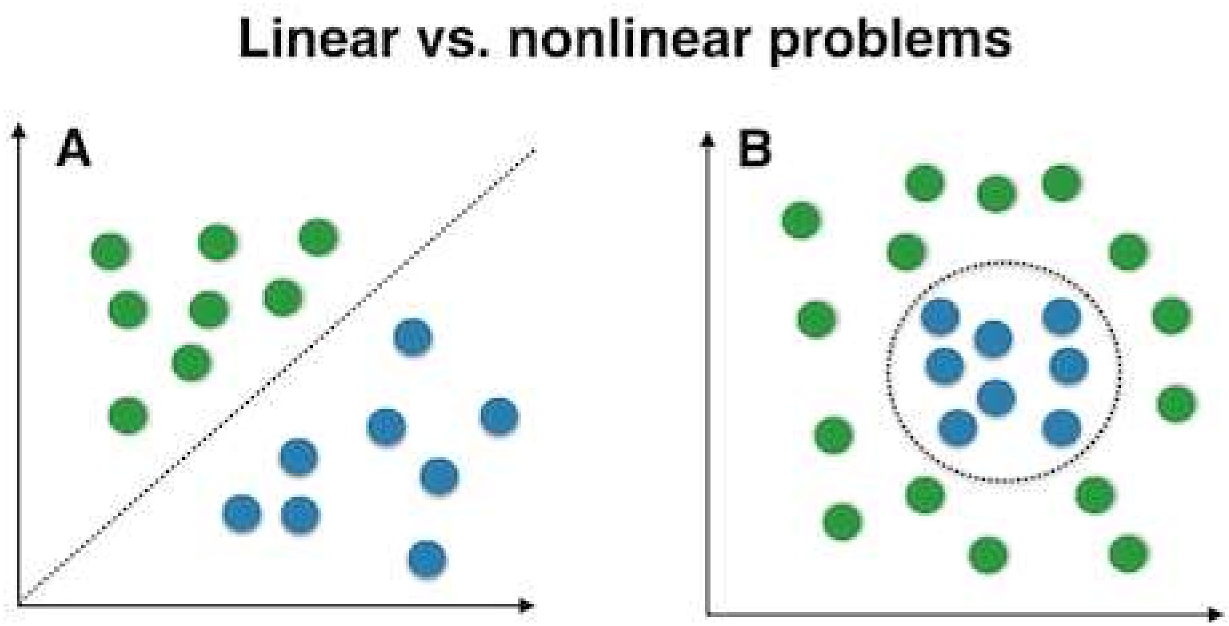
- Eg: learning to recognize digit --> 10 possible outputs --> need a perceptron network



Perceptron Network

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

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

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