VICTORIA UNIVERSITY OF WELLINGTON
*Te Whare Wananga o te Upoko o te Ika a Maui*

School of Engineering and Computer Science

# COMP 307 — Lecture 07

## Machine Learning 4

## **DT Learning and Perceptron Learning**


Mengjie Zhang (Meng)

*mengjie.zhang@ecs.vuw.ac.nz*




## Outline

- Decision tree learning:
  - Numeric attributes
  - Extension for DT learning
- Perceptron learning:
  - Linear threshold unit
  - Threshold transfer functions
  - Proceptron learning rule/algorithm
  - Property/Problem of perceptrons



## Numeric Attributes

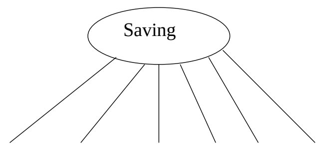
| | **Job** | **Saving** | **Family** | **Class** |
|---|---|---|---|---|
| A | true | $10K | single | Approve |
| B | true | $7K | couple | Approve |
| C | true | $16K | single | Approve |
| D | true | $25K | single | Approve |
| E | false | $12K | couple | Approve |
| 1 | true | $4K | couple | Reject |
| 2 | false | $30K | couple | Reject |
| 3 | true | $15K | children | Reject |
| 4 | false | $6K | single | Reject |
| 5 | false | $8K | children | Reject |

What question goes in the node “Saving”?

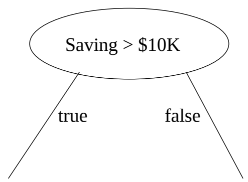


## Numeric Attributes (Continued)

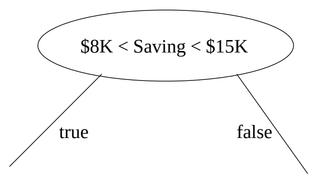
- Could split multiway — one brunch for each value
  - bad idea, no generalisation




- Could split on a simple comparison
  - But what split point?




- Could split on a subrange
  - But which range?



# Numeric Attributes (Continued)

| | Job | Saving | Family | Class |
|---|---|---|---|---|
| 1 | true | $4K | couple | Reject |
| 4 | false | $6K | single | Reject |
| B | true | $7K | couple | Approve |
| 5 | false | $8K | children | Reject |
| A | true | $10K | single | Approve |
| E | false | $12K | couple | Approve |
| 3 | true | $15K | children | Reject |
| C | true | $16K | single | Approve |
| D | true | $25K | single | Approve |
| 2 | false | $30K | couple | Reject |

- Don't need to try each possible split point
  - Order items and only consider *class boundaries*!



# Numeric Attributes to Binary

Consider the class boundaries, choose the best split:

- (Saving $<$ 7): 0.2 imp(0:2) + 0.8 imp(5:3) = 0.188
- (Saving $<$ 8): 0.3 imp(1:2) + 0.7 imp(4:3) = 0.238
- (Saving $<$ 10): 0.4 imp(1:3) + 0.6 imp(4:2) = 0.208
- (Saving $<$ 15): 0.6 imp(3:3) + 0.4 imp(2:2) = 0.250
- (Saving $<$ 16): 0.7 imp(3:4) + 0.3 imp(2:1) = 0.238
- (Saving $<$ 30): 0.9 imp(5:4) + 0.1 imp(0:1) = 0.222

Which one should we choose?



# Extension for DT Learning

- Information theoretical purity measures:
  - impurity = $\sum_{class} [P(class) \log(1/P(class))]$
  - Work for multiple classes
- Pruning:
  - Shrink the learned decision trees
  - Eliminate "overfitting"
- Multi-attribute decisions
  - Non-axis-parallel hyperplanes
- Turning learned decision trees to symbolic rules
- Regression trees
  - Leaves can be regression functions

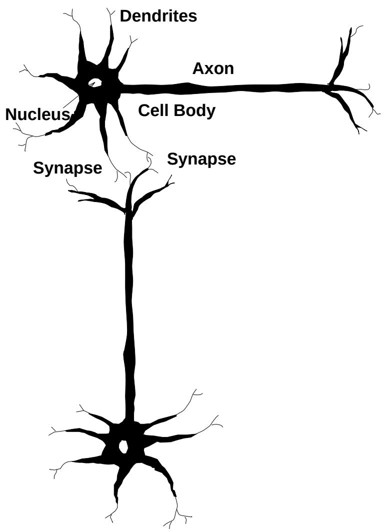


# Why Perceptrons and Neural Networks

- How do human being learn and do classification?
- Use brains (and eyes etc.)
- However, the machine learning methods discussed so far don't seem realistic for implementing/simulating in human brains
- Can we use our knowledge of brain structures to suggest alternative learning mechanisms?
- Using neurons
  - Neurons supports parallel distributed processing
  - Many methods weren't doing so well on hard, no-linear problems
  - Linear threshold units (simple perceptrons)
  - Neural networks (multiple layers, more complex structure)

# Neuron

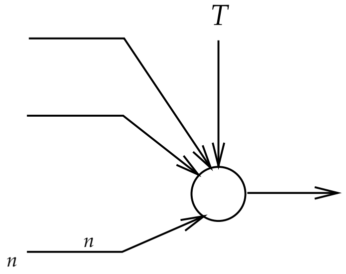
- The neuron receives impulses (signals) from other neurons via dendrites
- The neuron sends impulses to other neurons via the axon
- Synapse: dendrite of one neuron and synapse of another
- Impulses cause neurotransmitters (chemicals) to diffuse across the synapse
- Enhance (excite) or inhibit
- Action adjusted (by learning?)

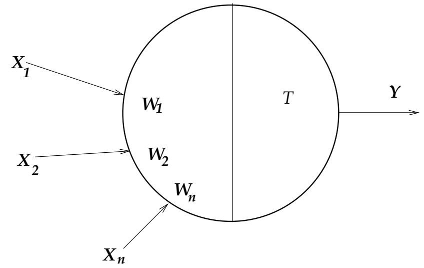





# Perceptron

- A kind of artificial neuron
- Simplest kind of neural networks
- *Linear threshold unit*
  - $Netinput = \sum_{i=1}^{n} x_i w_i$
  - $if\ Netinput \geq T\ then\ y = 1\ else\ y = 0$

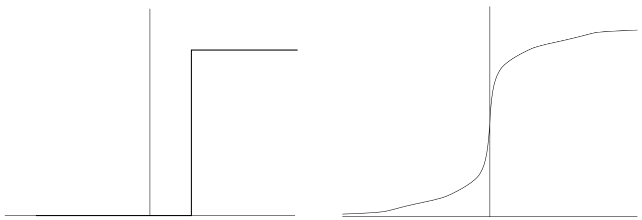





# Perceptron (Continued)

- Transfer functions



Threshold function    Sigmoid/Logistic function

- Input values (or "features") may *binary (0/1)* (usually) or *numeric*
- Output: binary (1 or 0)
- Problems:
  - How do you learn the weights?
  - How do you choose the input features?



# Simplify the Formula

- Replacing the threshold with a "dummy" feature:
  - Set $x_0 = 1$
  - Let $w_0 = -T$
- Change the rule

$$\text{if } \sum_{i=1}^{n} w_i x_i > T \text{ then } y = 1 \text{ else } y = 0$$

to

$$\text{if } \sum_{i=0}^{n} w_i x_i > 0 \text{ then } y = 1 \text{ else } y = 0$$

- The weights in the perceptron can be learned an algorithm

## Perceptron Learning Algorithm

- Initialize weights to (small) random numbers
  - Present an example (+ve/1, or -ve/0)
  - If perceptron is correct, do nothing
  - If -ve example and wrong
    - (weights on active features are too big/threshold is too low)
    - Subtract feature vector from weight vector
  - If +ve example and wrong
    - (weights on active features are too small/threshold is too high)
    - Add feature vector from weight vector
- Repeat the above steps for every input-output pair until $y = d$ for every pattern pair



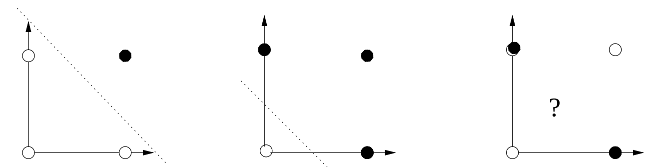
## Perceptron Classifier

- Consider a perceptron with two inputs, $n = 2$ ($\vec{x} = (x_1, x_2)$)
- Output is true (1) if $w_0 + w_1x_1 + w_2x_2 \geq 0$
  Output is false (0) if $w_0 + w_1x_1 + w_2x_2 < 0$
- Thus the line given by $w_0 + w_1x_1 + w_2x_2 = 0$
  i.e. $x_2 = -(w_1/w_2)x_1 - w_0/w_2$
  is the separating line between the two regions
- If $n = 3$ there will be a separating plane
  If $n > 3$ there will be a separating hyperplane
- A training set is *linearly separable* if the data points corresponding to the classes can be separated by a line.
- *Perceptron convergence theorem:* The perceptron training algorithm will converge **if and only if** the training set is linearly separable.



## Perceptron Learning(Continued)

- What can the perceptron learn?
- It can learn to discriminate linearly separable categories such as AND and OR.



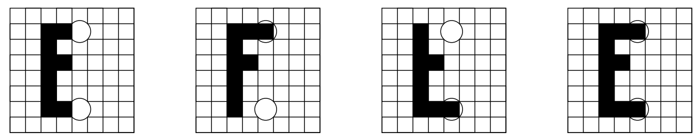

(a) and (2) or (3) xor

- XOR is not linearly separable. There is no way to draw a line to correctly classify all points.
- Perceptron cannot learn the XOR function!!! (Proved in 1969 by Minsky and Papert, with controversial result.)



## Perceptron Learning(Continued)

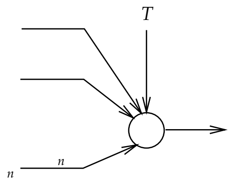
- Another example: “E” vs “not E”



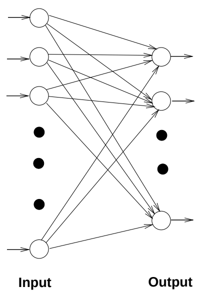
- If we use the pixel values marked in the figure as input features, can the perceptron method successfully solve this classification problem?
- Perhaps perceptron is not really useful if it can’t compute something as simple as XOR?
- Need better features!
- Need better network architecture!
- Need better learning algorithm!

# Perceptron Networks

- Perceptron (Linear Threshold Unit):
  $n$ inputs, 1 output




- Perceptron, perceptron network:
  $n$ inputs, $m$ outputs
- Input units just pass their input activation unchanged to all output arrows
- *Two* layers of nodes, *one* layer of weights.
- Still need improvement
  — **Multilayer perceptron** or
  — **Feed forward neural networks**!!






# Summary

- Numeric attributes in decision tree learning
- IG purity measure in decision trees
- Perceptron structure
- Perceptron learning algorithm
- Limitation of perceptron learning

- Next lecture: Neural networks
- Reading: Text book section 20.5 (2nd edition) or section 18.7 (3rd edition) or Web materials