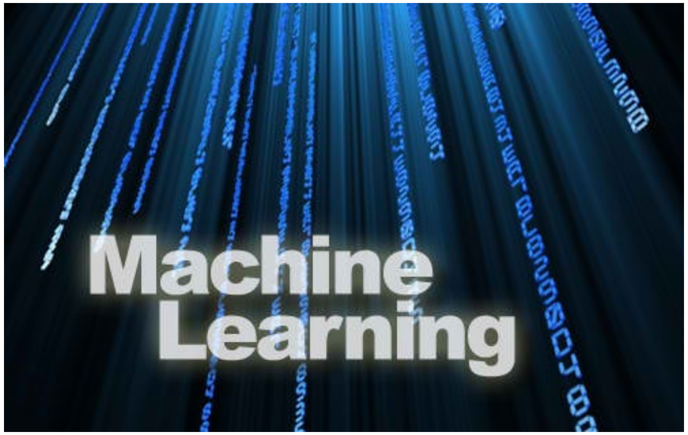
Machine Learning
DV2557

Dr. Prashant Goswami
Assistant Professor, BTH (DIDA)
prashantgos.github.io

prashant.goswami@bth.se



What is ML?

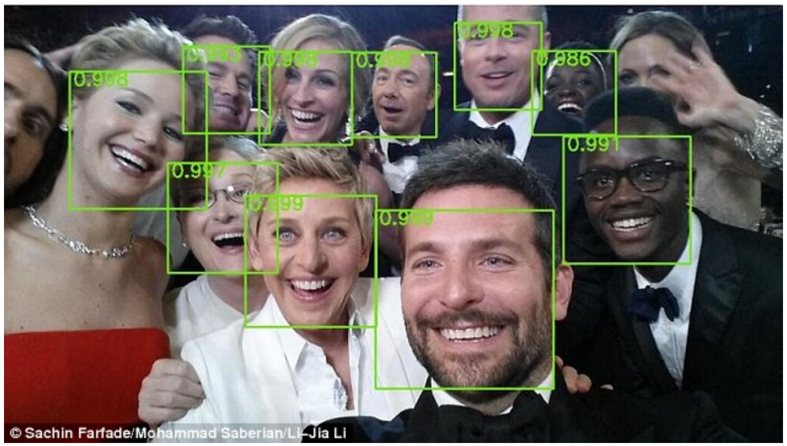
”… the construction and study of systems that can learn from data.”

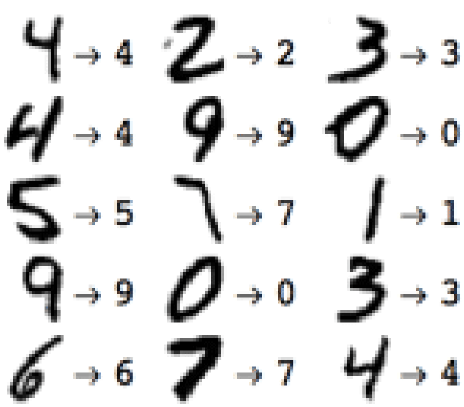
- A system that can:
  - Take known data as input.
  - Learn from the known data. Generalization.
  - Classify or draw conclusions from unseen data.

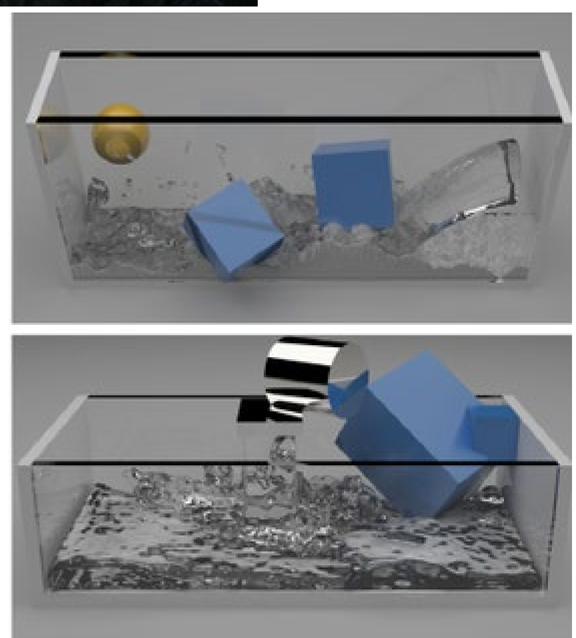
ML vs. Data Mining

- ML focuses on *prediction* based on known properties learned from data.
  - Example: Facial recognition, weather predictions.
- Data Mining focuses on the *discovery* of previously unknown properties on the data.
  - Example: Find patterns in medical history of patients with the same disease.

Machine Learning - Examples







Data Mining - Examples

Service providers

Why did the customer leave?

Crime agencies

Who could be a potential criminal?

https://www.matillion.com/insights/5-real-life-applications-of-data-mining-and-business-intelligence/

DATA REPRESENTATION

Instance

- Useful data consists of a number of instances (examples) with input and a known output.
- Consists of a number of *attributes*, and one or more class attributes (target).
- A set of instances is the input to a classifier.
- The instances are used by the Classifier to form a Concept (the thing to be learned).

Attribute Types

- Nominal
  - Finite set of symbols or numbers.
- Numeric (continuous)
  - Numbers (real or integer values).
- Boolean (special case of nominal)
  - True/false
  - Yes/no
  - 1/0

Preparing the data

1. Decide a suitable format for your data.
   - ARFF is a standardized data format.
2. Gather data and compile it into your format.
3. Input the data to a classifier.
   - Each classifier has pros and cons, and you should select one suitable for your type of data.

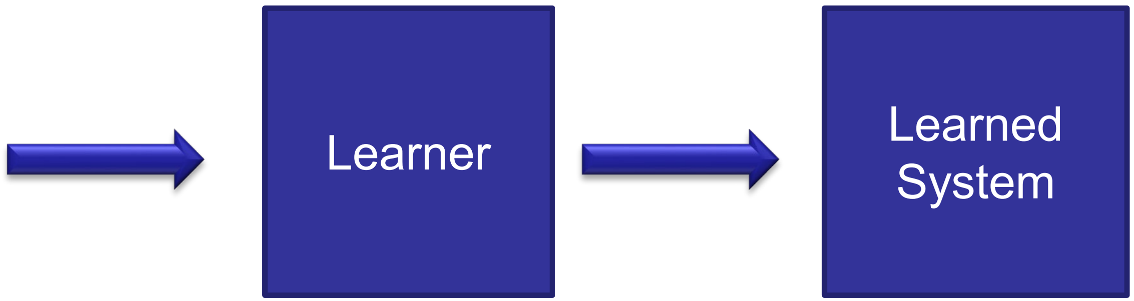
ARFF example

```
% this is a comment
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? {yes, no }

@data
sunny, 85, 85, false, no
sunny, 80, 90, true, no
….
```

When we have the data, we can learn from it

Dataset
(1+1=2)

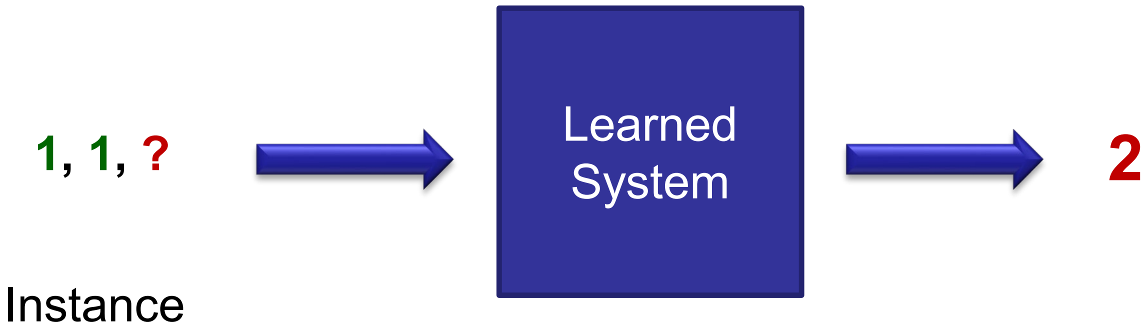
1, 1, 2
1, 1, 2
1, 1, 2
1, 1, 3
1, 1, 2



Attributes

Target

When we have learned something, we can predict stuff



Knowing your data greatly simplifies the experiment design and execution!

TYPES OF CLASSIFIERS

Two main types

- Supervised learning
  - Learning with an instructor.
  - We have some known data.
  - Learn to handle the known data, and hopefully we can classify unknown data.
- Unsupervised learning
  - Learning without an instructor.
  - Try something, and see how it works.
  - Utility function to calculate how well it worked.

Two main types

- Supervised learning
  - Learning with an instructor.
  - We have some known data.
  - Learn to handle the known data, and hopefully we can classify unknown data.
- Unsupervised learning
  - Learning without an instructor.
  - Try something, and see how it works.
  - Utility function to calculate how well it worked.

Decision Trees

- Each node in the tree tests a particular attribute.
- Each leaf represents a class.
- Suitable for nominal attributes.
  - Even though we can discretize numerical attributes…
  - temp < 15 = Cold, 15-24 = Temperate, > 25 = Warm.
- Common algorithms: ID3, C45, J48.

Decision Trees

- Fast at learning and classification.
- Concept readable by humans.
- Low storage requirements.
- Bad at handling inconsistent data.
- Cannot learn some concepts:
    - Parity function: 1 if and only if an even number of inputs are 1.
    - Majority function: 1 if more than half of the inputs are 1.
    - Read more : http://www.cs.utexas.edu/~klivans/f07lec3.pdf
- Discretizing numerical attributes leads to loss of precision.

Example: Weather data

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	NO
sunny	hot	high	true	NO
overcast	hot	high	false	YES
rainy	mild	high	false	YES
rainy	cool	normal	false	YES
rainy	cool	normal	true	NO
overcast	cool	normal	true	YES
sunny	mild	high	false	NO
sunny	cool	normal	false	YES
rainy	mild	normal	false	YES
sunny	mild	normal	true	YES
overcast	mild	high	true	YES
overcast	hot	normal	false	YES
rainy	mild	high	true	NO

Building the tree

- At each node, we need to find the attribute that best divides the data into Yes and No.
- To do this we calculate the information gain for each parameter and value.
- The attribute with the highest information gain is selected at each node.

$$Gain(A) = 1 - \left(\sum_{i=1}^{v} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i} \right) \right)$$

$$I\left(\frac{p}{p+n}, \frac{n}{p+n} \right) = \frac{p+n}{p_{tot} + n_{tot}} \left(- \frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n} \right)$$

Find the root node

Outlook	Sunny	Overcast	Rainy
Yes	2	4	3
No	3	0	2

$$I(sunny) = \frac{5}{14}\left(-\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5}\right) \approx 0.347$$

$$I(overcast) = \frac{4}{14}\left(-\frac{4}{4}log_2\frac{4}{4} - 0\right) = 0$$

$$I(rainy) = \frac{5}{14}\left(-\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}\right) \approx 0.347$$

$$Gain(outlook) \approx 1 - 0.347 - 0 - 0.347 = \boxed{0.306}$$

$$\boxed{I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = \frac{p+n}{p_{tot}+n_{tot}}\left(-\frac{p}{p+n}log_2\frac{p}{p+n} - \frac{n}{p+n}log_2\frac{n}{p+n}\right)}$$

Find the root node

Temperature	Hot	Mild	Cool
Yes	2	4	3
No	2	2	1

$$I(hot) = \frac{4}{14}\left(-\frac{2}{4}log_2\frac{2}{4} - \frac{2}{4}log_2\frac{2}{4}\right) \approx 0.286$$

$$I(mild) = \frac{6}{14}\left(-\frac{4}{6}log_2\frac{4}{6} - \frac{2}{6}log_2\frac{2}{6}\right) \approx 0.394$$

$$I(cool) = \frac{4}{14}\left(-\frac{3}{4}log_2\frac{3}{4} - \frac{1}{4}log_2\frac{1}{4}\right) \approx 0.232$$

$$Gain(temperature) \approx 1 - 0.286 - 0.394 - 0.232 = \boxed{0.088}$$

Find the root node

Humidity	High	Normal
Yes	3	6
No	4	1

$$I(high) = \frac{7}{14}\left(-\frac{3}{7}log_2\frac{3}{7} - \frac{4}{7}log_2\frac{4}{7}\right) \approx 0.493$$

$$I(normal) = \frac{7}{14}\left(-\frac{6}{7}log_2\frac{6}{7} - \frac{1}{7}log_2\frac{1}{7}\right) \approx 0.296$$

$$Gain(humidity) \approx 1 - 0.493 - 0.296 = \boxed{0.211}$$

Find the root node

Windy	True	False
Yes	3	6
No	3	2

$$I(true) = \frac{6}{14}\left(-\frac{3}{6}log_2\frac{3}{6} - \frac{3}{6}log_2\frac{3}{6}\right) \approx 0.429$$

$$I(false) = \frac{8}{14}\left(-\frac{6}{8}log_2\frac{6}{8} - \frac{2}{8}log_2\frac{2}{8}\right) \approx 0.464$$

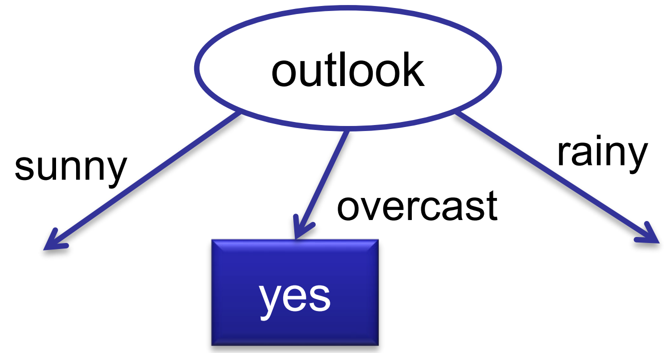
$$Gain(windy) \approx 1 - 0.429 - 0.464 = \boxed{0.107}$$

Find the root node

Attribute	Gain
Outlook	0.306
Temperature	0.088
Humidity	0.211
Windy	0.107

Outlook has the highest gain and is selected as root node.

Find the root node



Overcast has perfect gain = all instances
with overcast belongs to the same class, Yes.

Let's find the sunny node!

All instances with sunny

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	NO
sunny	hot	high	true	NO
sunny	mild	high	false	NO
sunny	cool	normal	false	YES
sunny	mild	normal	true	YES

5 instances left.

Find the sunny node

Temperature	Hot	Mild	Cool
Yes	0	1	1
No	2	1	0

$$I(hot) = 0$$

$$I(mild) = \frac{2}{5} \cdot 1 = 0.4$$

$$I(cool) = 0$$

$$Gain(temperature) = 1 - 0 - 0.4 - 0 = \boxed{0.6}$$

Find the sunny node

Humidity	High	Normal
Yes	0	2
No	3	0

$$I(high) = 0$$

$$I(normal) = 0$$

$$Gain(humidity) = 1 - 0 - 0 = \boxed{1}$$

Find the sunny node

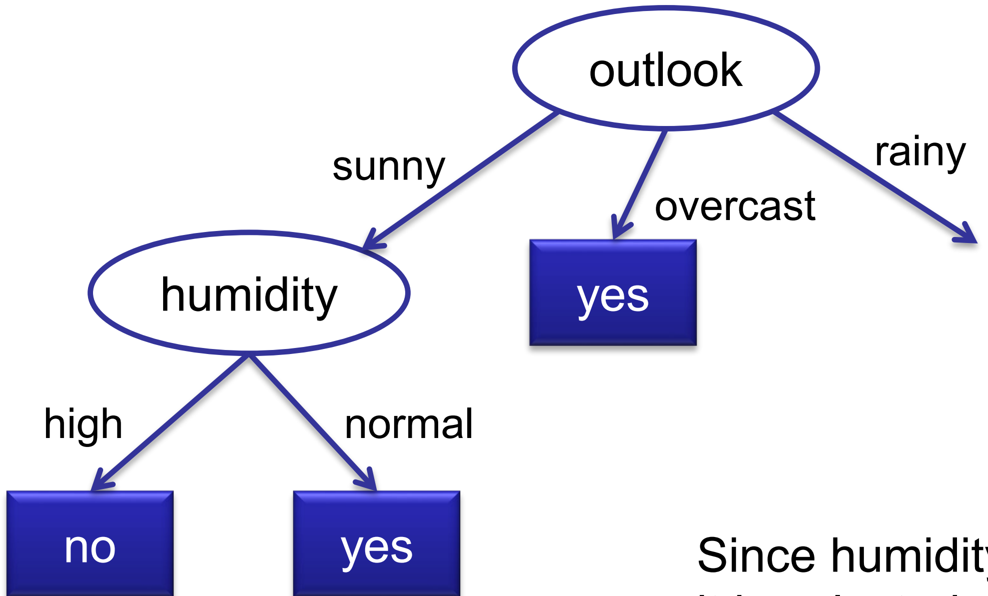
Windy	True	False
Yes	1	1
No	1	2

$$I(true) = \frac{2}{5} \cdot 1 = 0.4$$

$$I(false) = \frac{3}{5}\left(-\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3}\right) \approx 0.551$$

$$Gain(windy) \approx 1 - 0.4 - 0.551 = \boxed{0.049}$$

Find the sunny node



Since humidity has perfect gain it is selected.

Let's find the rainy node!

All instances with rainy

Outlook	Temperature	Humidity	Windy	Play
rainy	mild	high	false	YES
rainy	cool	normal	false	YES
rainy	cool	normal	true	NO
rainy	mild	normal	false	YES
rainy	mild	high	true	NO

5 instances left.

Find the rainy node

Temperature	Hot	Mild	Cool
Yes	0	2	1
No	0	1	1

$$I(hot) = 0$$

$$I(mild) = \frac{3}{5}\left(-\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3}\right) \approx 0.551$$

$$I(cool) = \frac{2}{5} \cdot 1 = 0.4$$

$$Gain(temperature) \approx 1 - 0 - 0.551 - 0.4 = \boxed{0.049}$$

Find the rainy node

Humidity	High	Normal
Yes	1	2
No	1	1

$$I(high) = \frac{2}{5} \cdot 1 = 0.4$$

$$I(normal) = \frac{3}{5}\left(-\frac{2}{3}log_2\frac{2}{3} - \frac{1}{3}log_2\frac{1}{3}\right) \approx 0.551$$

$$Gain(humidity) \approx 1 - 0.4 - 0.551 = \boxed{0.049}$$

Find the rainy node

Windy	True	False
Yes	0	3
No	2	0

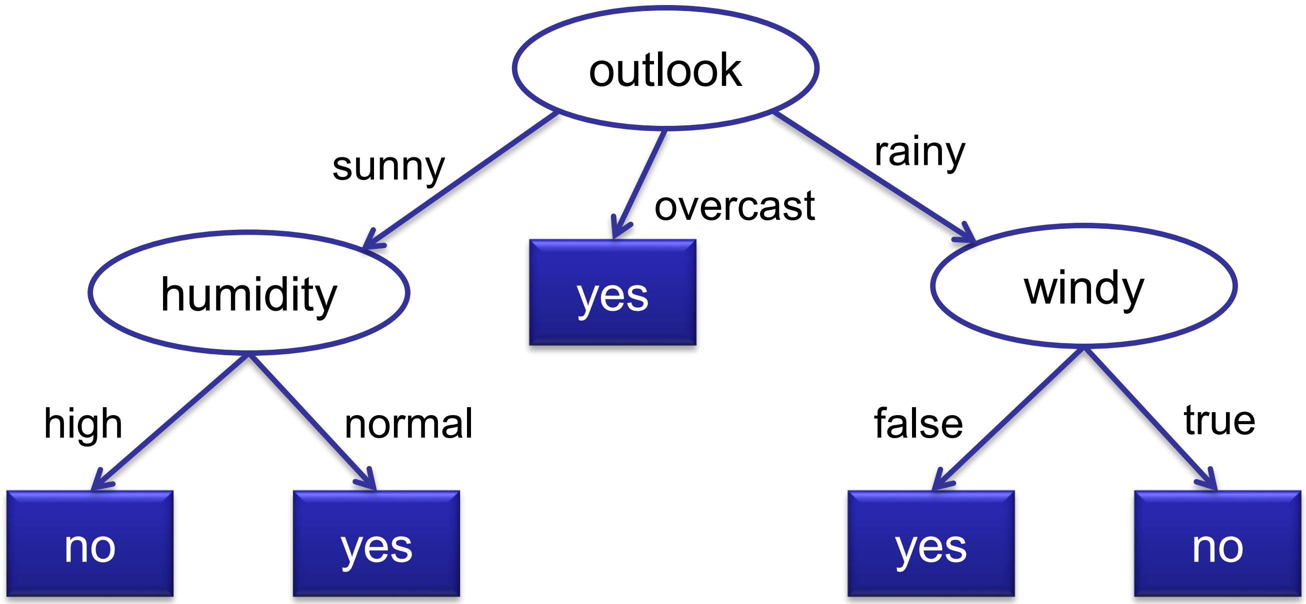
$$I(true) = 0$$

$$I(false) = 0$$

$$Gain(windy) = 1 - 0 - 0 = \boxed{1}$$

Since windy has perfect gain, it is selected.

Final Result

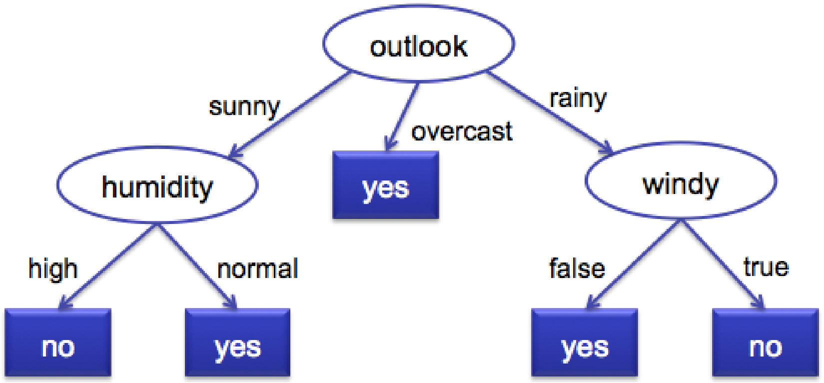


OneR vs. Decision Trees

```
outlook:
         sunny    -> no
         overcast -> yes
         rainy    -> yes
(10/14 instances correct)
```

OneR: Learn one rule that best describes the concept

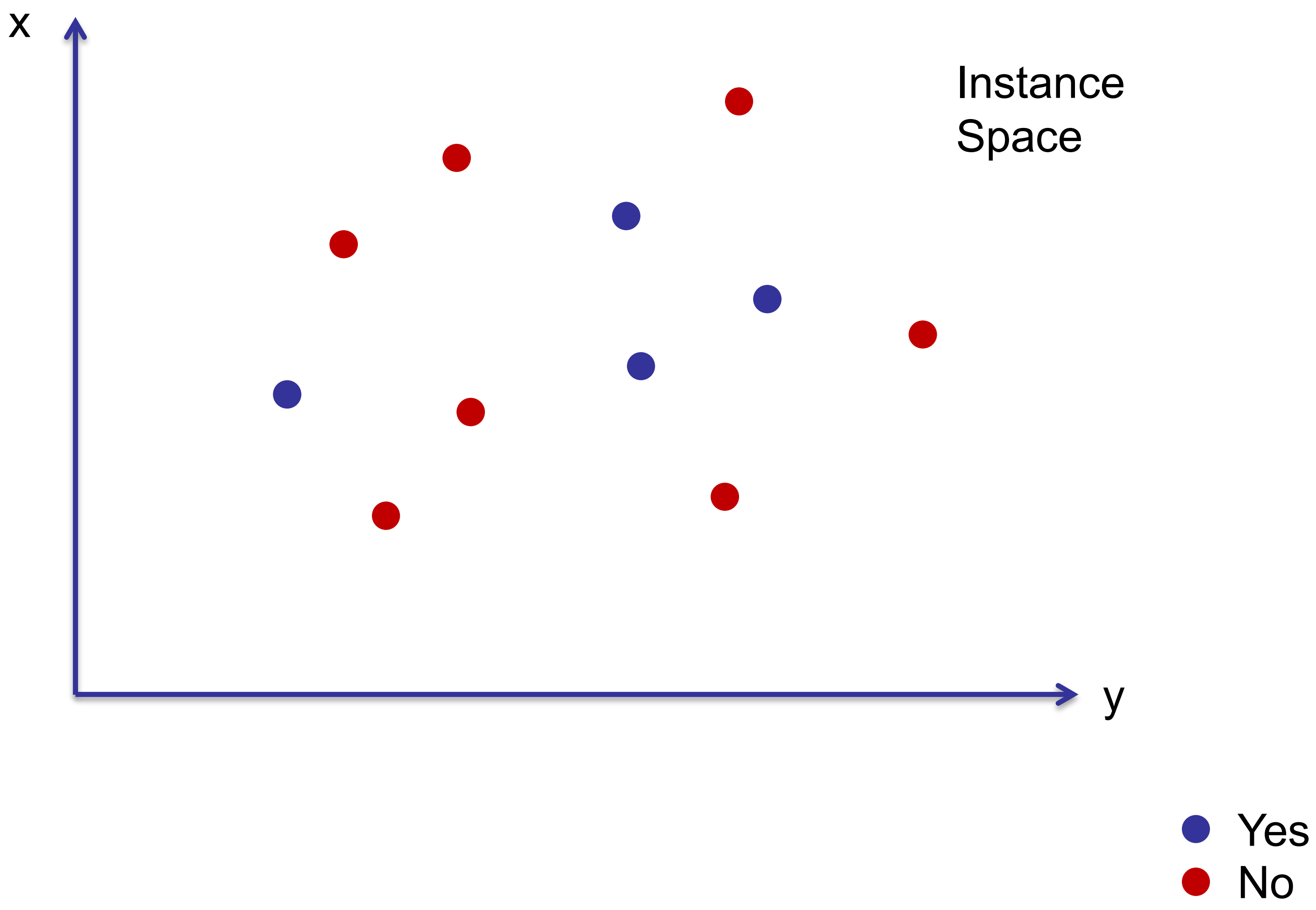
The top node in a DT is the attribute that best divides the data, which is the same as the OneR rule.



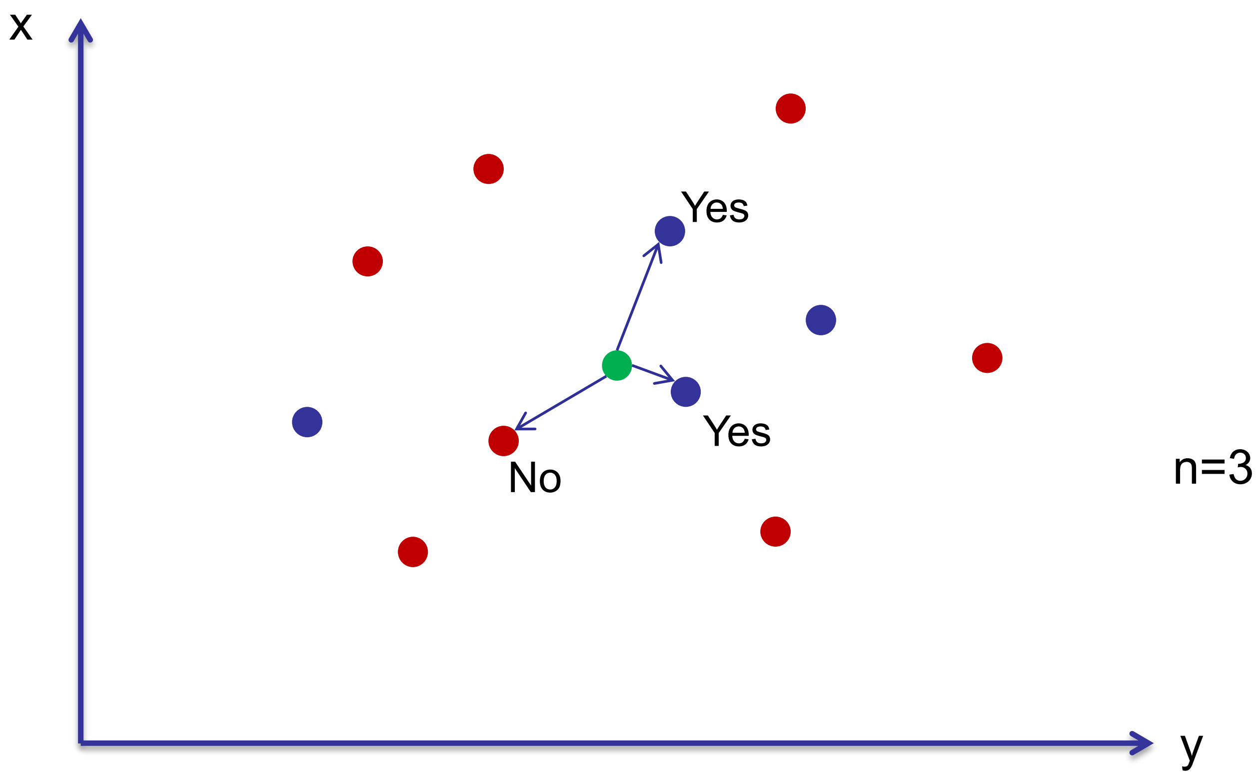
Lazy

- A lazy classifier stores the training data without processing it.
- All the work is done at <u>classification</u> time.
- Calculate the n closest instances using Euclidean distance between attributes.
- K-Nearest-Neighbor.

Lazy



Lazy



The class of the new instance is the same as the most common class of the *n=3* closest training instances = Yes!

Lazy

- High storage requirements.
- No generalization is done.
- Not so good at handling missing attributes.
- Works best with numerical attributes.
  - Is A closer to B than D?
- Is often quite accurate, but the high resource demands at classification time can be a problem.

Naïve Bayes

- Based on Bayes' rule.
- Used as a *Classifier*
- Each attribute is independent!
- Usually not true for real-world problems, but works surprisingly well anyway!

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Bayesian classifiers: Bayes' rule

- Example:
  - Is stiff neck a good sign of the disease meningitis?
  - = we want to find P(m|s): probability for meningitis given a stiff neck.
- Prior probabilities:
  - 50% of meningitis patients have a stiff neck:
    P(s|m) = 0.5
  - One in 50000 suffers from meningitis:
    P(m) = 1/50000
  - One in 20 suffers from stiff neck:
    P(s) = 1/20
- Now we can calculate P(m|s) using Bayes' rule:

$$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.5 \cdot 1/50000}{1/20} = 0.0002$$

Naïve Bayes Classification

1. Calculate frequencies of each class and each attribute value (for all attributes).
2. For each class,
   a. Multiply the conditional probability of each attribute into a product, X
   b. Multiply the class probability with X
3. Classify the instance as belonging to the class with the highest probability.

Lets look at an example!

Example: A limited Weather set

Outlook	Temperature	Play
sunny	hot	NO
sunny	hot	NO
overcast	hot	YES
rainy	mild	YES
rainy	cool	YES
rainy	cool	NO
overcast	cool	YES
sunny	mild	NO
sunny	cool	YES
rainy	mild	YES
sunny	mild	YES
overcast	mild	YES
overcast	hot	YES
rainy	mild	NO

Example: A limited Weather set

Outlook			Temp			Play	
	Yes	No		Yes	No	YES	NO
Sunny	3	2	Hot	2	2	9	5
Overcast	4	1	Mild	5	0		
Rainy	2	2	Cool	2	3		

Example: A limited Weather set

Outlook			Temp			Play	
	Yes	No		Yes	No	YES	NO
Sunny	3	2	Hot	2	2	9	5
Overcast	4	1	Mild	5	0		
Rainy	2	2	Cool	2	3		
P(Sunny)	3/9	2/5	P(Hot)	2/9	2/5	9/14	5/14
P(Overc)	4/9	1/5	P(Mild)	5/9	0/5		
P(Rainy)	2/9	2/5	P(Cool)	2/9	3/5		

Classifying new instances

- Do we want to go out and play if {outlook=sunny, temp=cool}?
- Now lets try the attributes against the two possible hypotheses H = YES and H = NO:
    - P(YES) * P(sunny|YES) * P(cool|YES) = 9/14 * 3/9 * 2/9 = 0.048
    - P(NO) * P(sunny|NO) * P(cool|NO) = 5/14 * 2/5 * 3/5 = 0.086
- We don’t want to go out and play!

1. Calculate frequencies of each class and each attribute value (for all attributes).
2. For each class,
    a. Multiply the conditional probability of each attribute into a product, X
    b. Multiply the class probability with X
3. Classify the instance as belonging to the class with the highest probability.

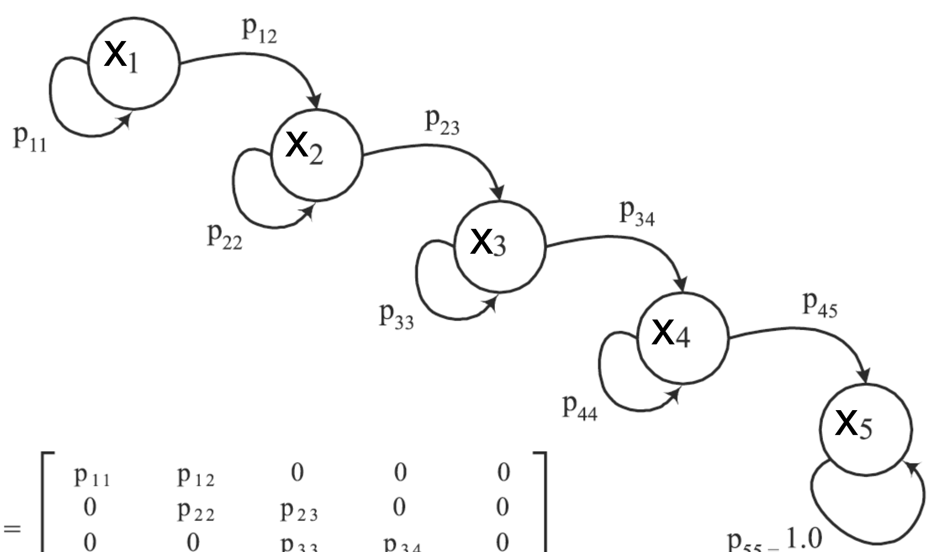
Naïve Bayes

- Very good at analyzing texts.
- Texts are split into word lists:
  - ”Want to buy cheap Rolex watches” → { Want, to, buy, cheap, Rolex, watches}
- Used in many email spam filters.
- Sensitive at learning the wrong thing (for example English vs. Swedish)

Markov Model

- Stochastic, time varying process
- Next state dependent only on current state

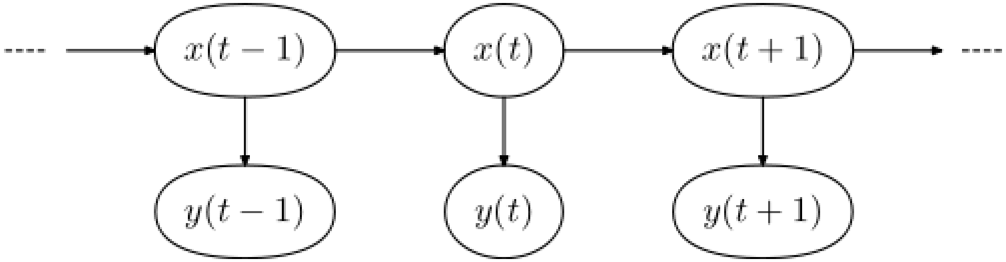
$$\Pr(X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \Pr(X_{n+1} = x \mid X_n = x_n)$$



$$P = \begin{bmatrix} p_{11} & p_{12} & 0 & 0 & 0 \\ 0 & p_{22} & p_{23} & 0 & 0 \\ 0 & 0 & p_{33} & p_{34} & 0 \\ 0 & 0 & 0 & p_{44} & p_{45} \\ 0 & 0 & 0 & 0 & 1.0 \end{bmatrix}$$

Hidden Markov Model

- HMM is a simplified dynamic Bayesian Network classifier.
    - We will not look into more complex Bayesian Network models.
- It contains a set of states and transition probabilities.
- Based on the input and the previous state, there is a probability of moving to a new state, or remain in the same state.



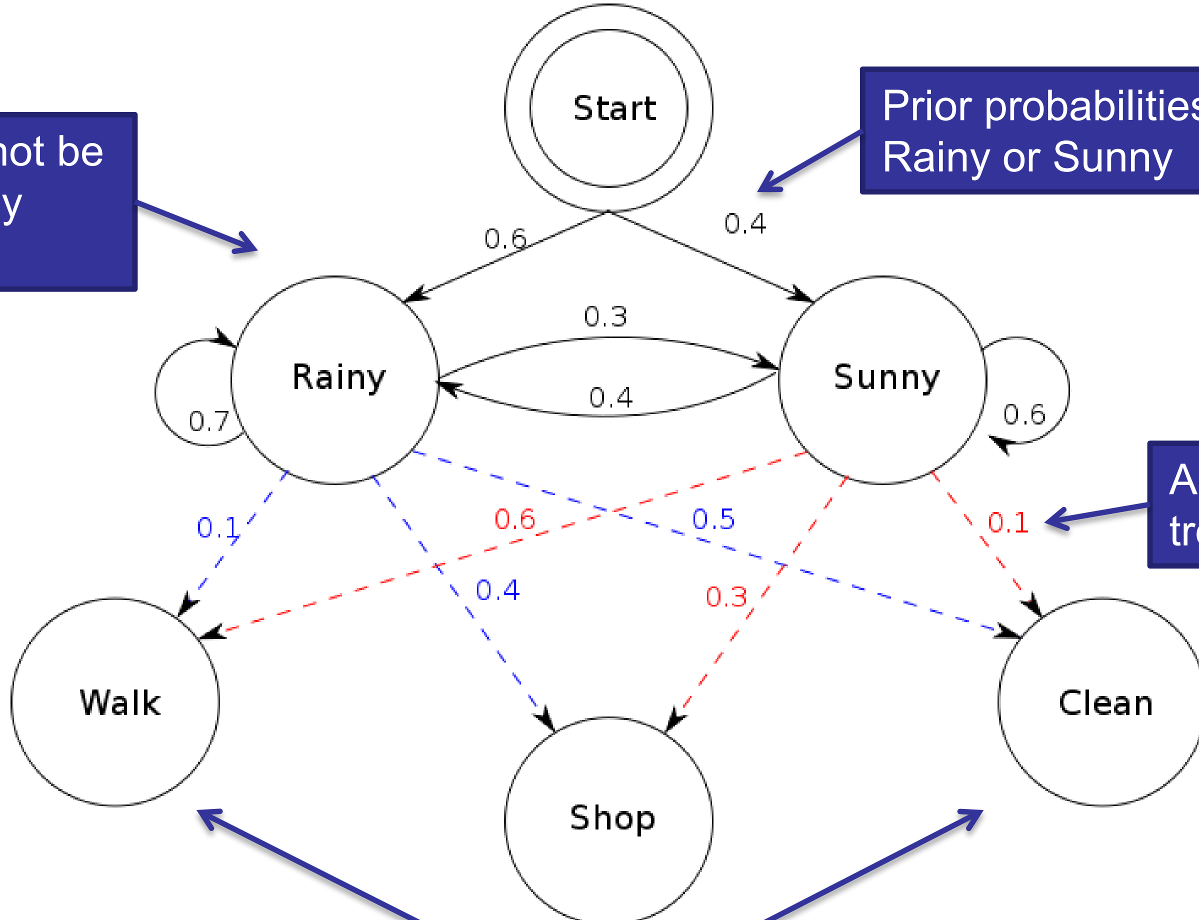
Hidden Markov Model

- Alice calls Bob every day.
- She knows Bob either takes a Walk, goes Shopping or Cleans the apartment.
- What Bob does is depending on the weather (Rainy, Sunny).
- Alice has observed some trends in what Bob tends to do based on the weather.
- Bob tells Alice what he did during the day, and Alice tries to guess the weather.

Hidden Markov Model

States that cannot be observed directly (hidden)

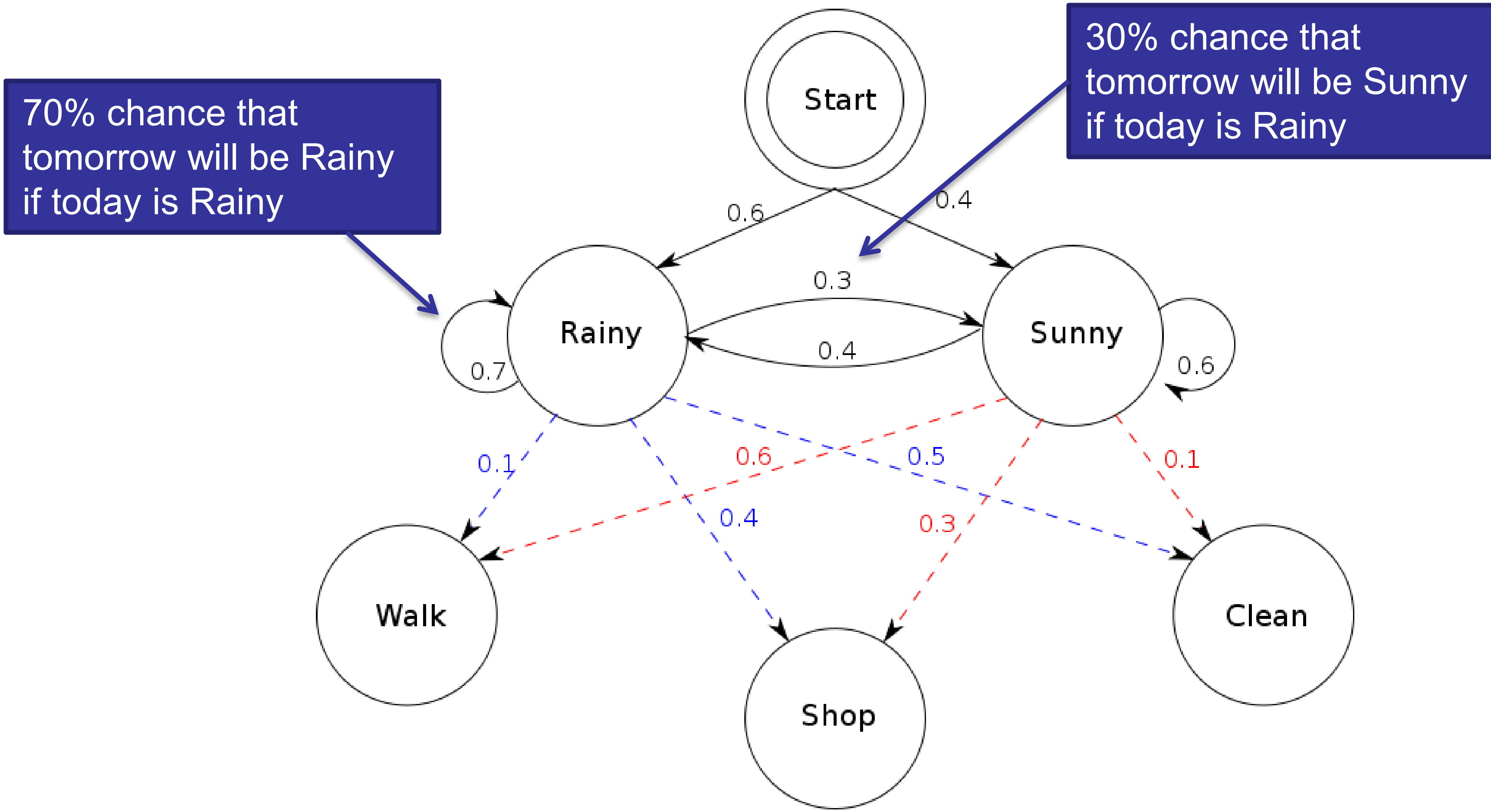
Prior probabilities of Rainy or Sunny



Alice’s observed trends

Known information

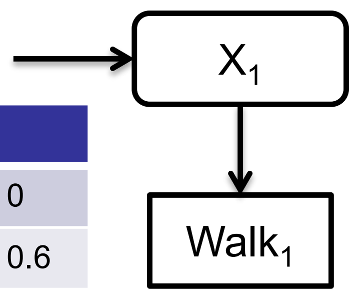
Hidden Markov Model



Transition Probabilities
- based on historical data.

Hidden Markov Model

Prior distribution



F_0		
Sunny	0.4	0
Rainy	0	0.6

Sensor Model

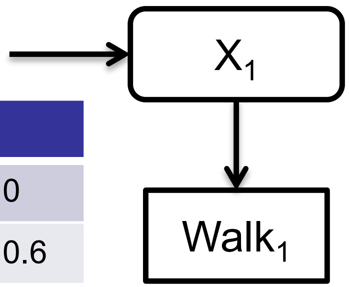
O_1		
Sunny	0.6	0
Rainy	0	0.1

Transition Model

T	S	R
Sunny	0.6	0.4
Rainy	0.3	0.7

Hidden Markov Model

$F_1=O_1*T*F_0$			Π (norm)
Sunny	0.144	0.216	Norm(0.144*0.216) = 0.8
Rainy	0.072	0.108	Norm(0.072*0.108) = 0.2



F_0		
Sunny	0.4	0
Rainy	0	0.6

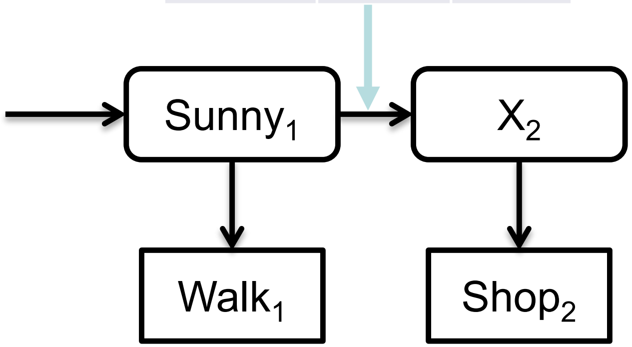
X_1 = Sunny

O_1		
Sunny	0.6	0
Rainy	0	0.1

T	S	R
Sunny	0.6	0.4
Rainy	0.3	0.7

Hidden Markov Model

F_1		
Sunny	0.144	0.216
Rainy	0.072	0.108



$F_2=O_2*T*F_1$			Π (norm)
Sunny	0.037	0.056	0.56
Rainy	0.033	0.050	0.44

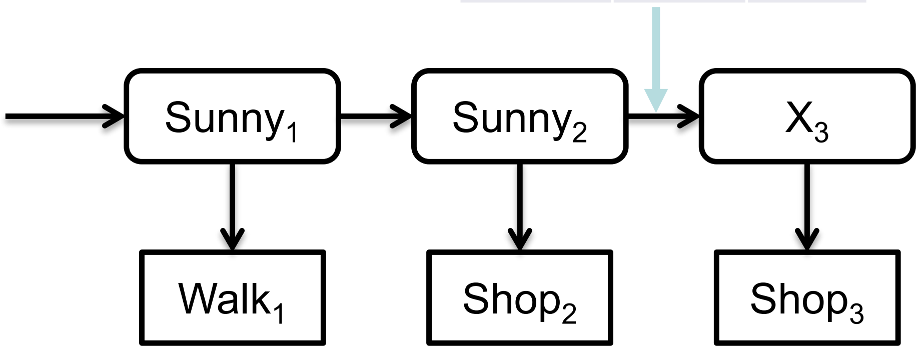
X_2 = Sunny

O_2		
Sunny	0.3	0
Rainy	0	0.4

T	S	R
Sunny	0.6	0.4
Rainy	0.3	0.7

Hidden Markov Model

F_2		
Sunny	0.037	0.056
Rainy	0.033	0.050



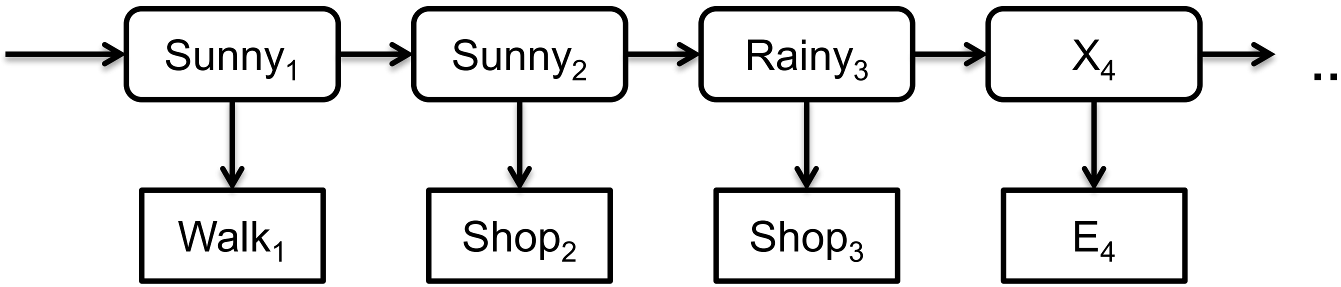
$F_3=O_3*T*F_2$			Π (norm)
Sunny	0.012	0.018	0.48
Rainy	0.013	0.019	0.52

X_3 = Rainy

O_3		
Sunny	0.3	0
Rainy	0	0.4

T	S	R
Sunny	0.6	0.4
Rainy	0.3	0.7

Hidden Markov Model



Hidden Markov Model

- Markov Assumption: The current state depends only on a *finite* history of previous states.
    - First-order HMM = depends on previous state (as in the example)
    - Second-order HMM = depends on last two states.
- Finding the most likely sequence uses the Viterbi Algorithm.
    - Which we showed an example of

Hidden Markov Model

- Suitable for nominal attributes.
- Good at handling sequences of variable length, for example:
  - Biological data like protein sequences:

…MMNEHSIDTDNRKANNALYLFIIIGLEHSMNEMALY…

Length can vary from around 150 to 10000+

Linear Regression

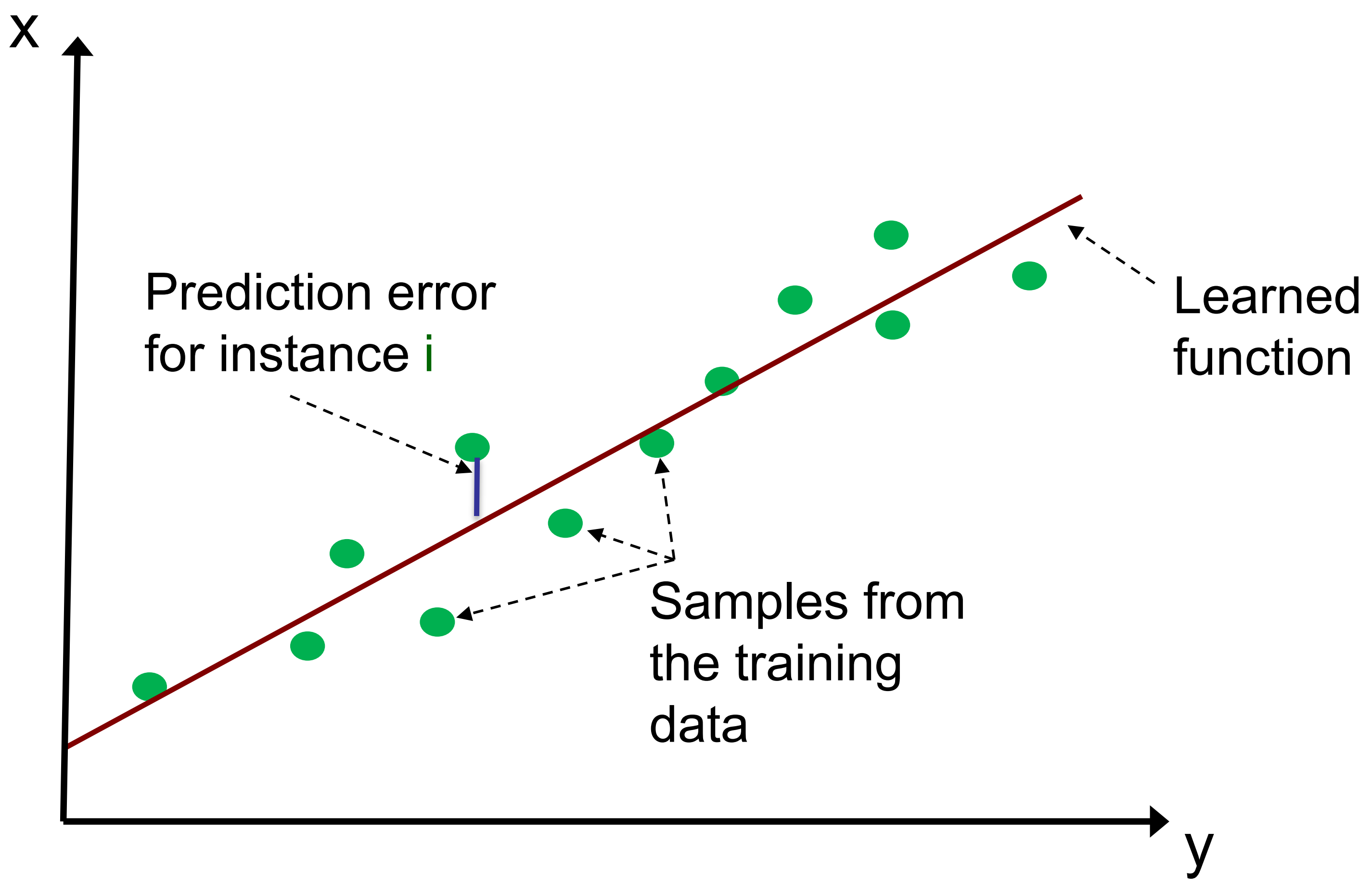
- Basic idea: Express the class as a linear combination of its attributes:
    - $x = w_0 + w_1a_1 + w_2a_2 + \ldots w_ka_k = \sum w_ja_j$
    - x = class
    - a_k = attribute values
    - w_k = weights for each attribute
- The weights are calculated (learned) from the training data.

Linear Regression

- When learning we need to minimize the prediction error over the training data.
- Let
    - $x^{(i)}$ = actual class value for instance i
    - $\sum w_j a_j^{(i)}$ = predicted class value for instance i
    - $(x^{(i)} - \sum w_j a_j^{(i)})^2$ = squared error for instance i

- We aim to minimize the sum of the squared error over all instances in the training data.

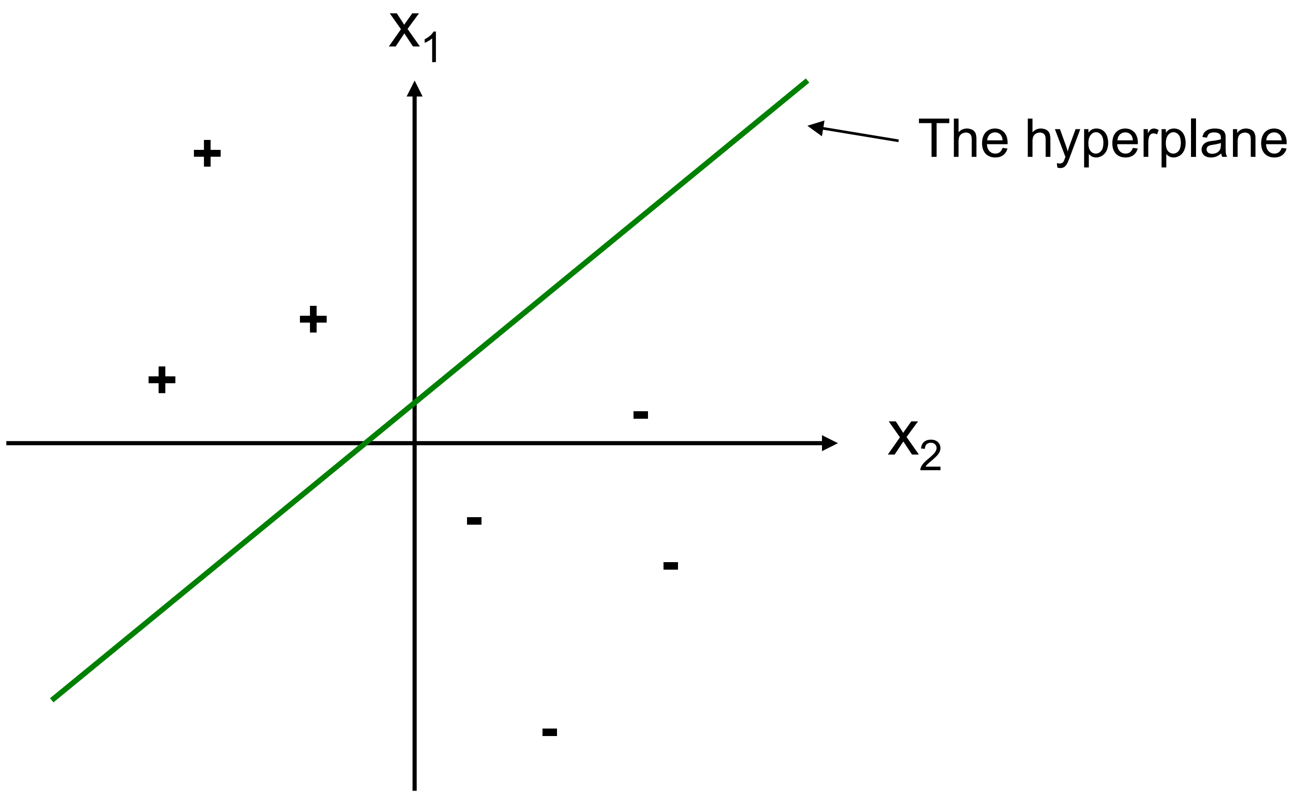
Linear Regression



The Hyperplane

- If the data can be clearly separated into two groups, we can use the perceptron training rule.
- We learn a plane, the *hyperplane*, that separate the groups.
- Equation:
    - $w_0 + w_1a_1 + w_2a_2 + \ldots w_ka_k = 0$

The Hyperplane



x_1 and x_2 are attributes.
+ and – are the classes.

The Perceptron

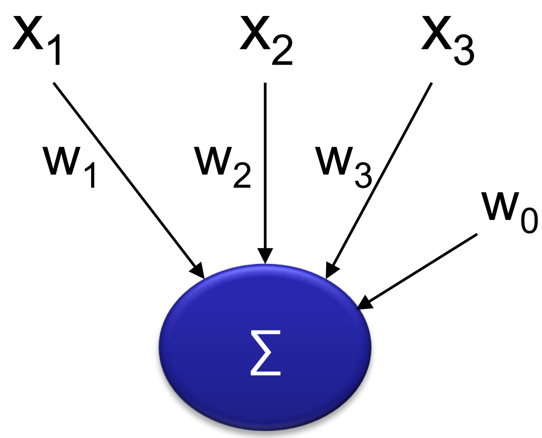
x_1 x_2 x_3

w_1 w_2 w_3

w_0

Start with a set of attributes and weights, and a constant.

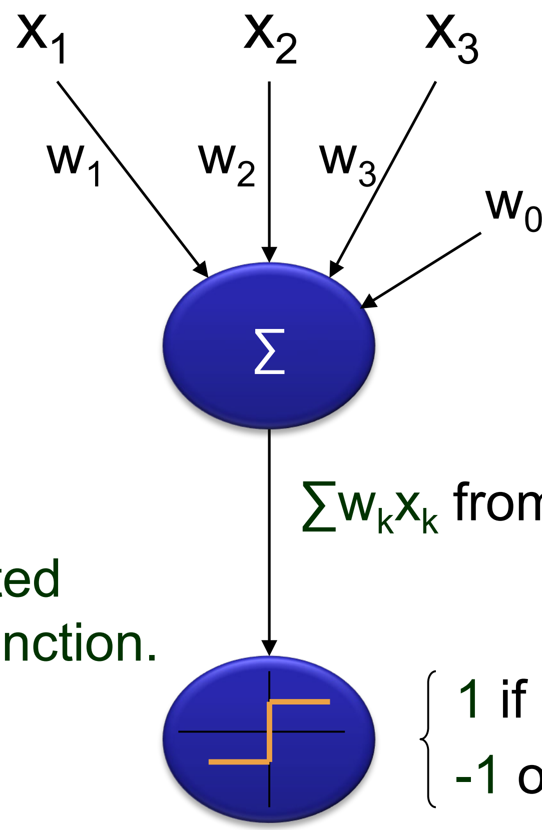
The Perceptron



Feed them to a unit that calculates the weighted sum.

$\sum w_k x_k$ from 0 to k

The Perceptron

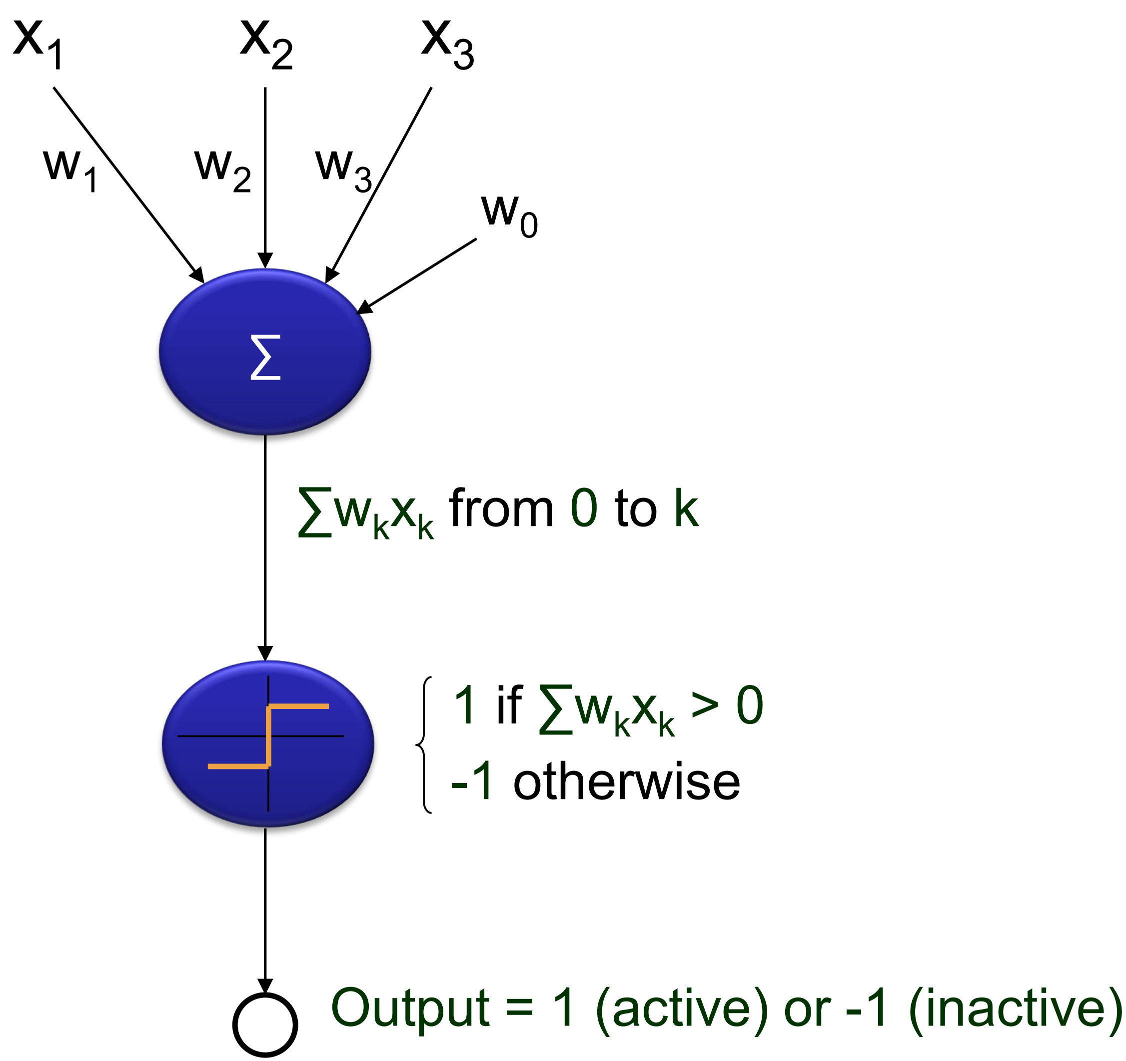


$\sum w_k x_k$ from 0 to k

Feed the weighted sum to a step function.

$$\begin{cases} 1 \text{ if } \sum w_k x_k > 0 \\ -1 \text{ otherwise} \end{cases}$$

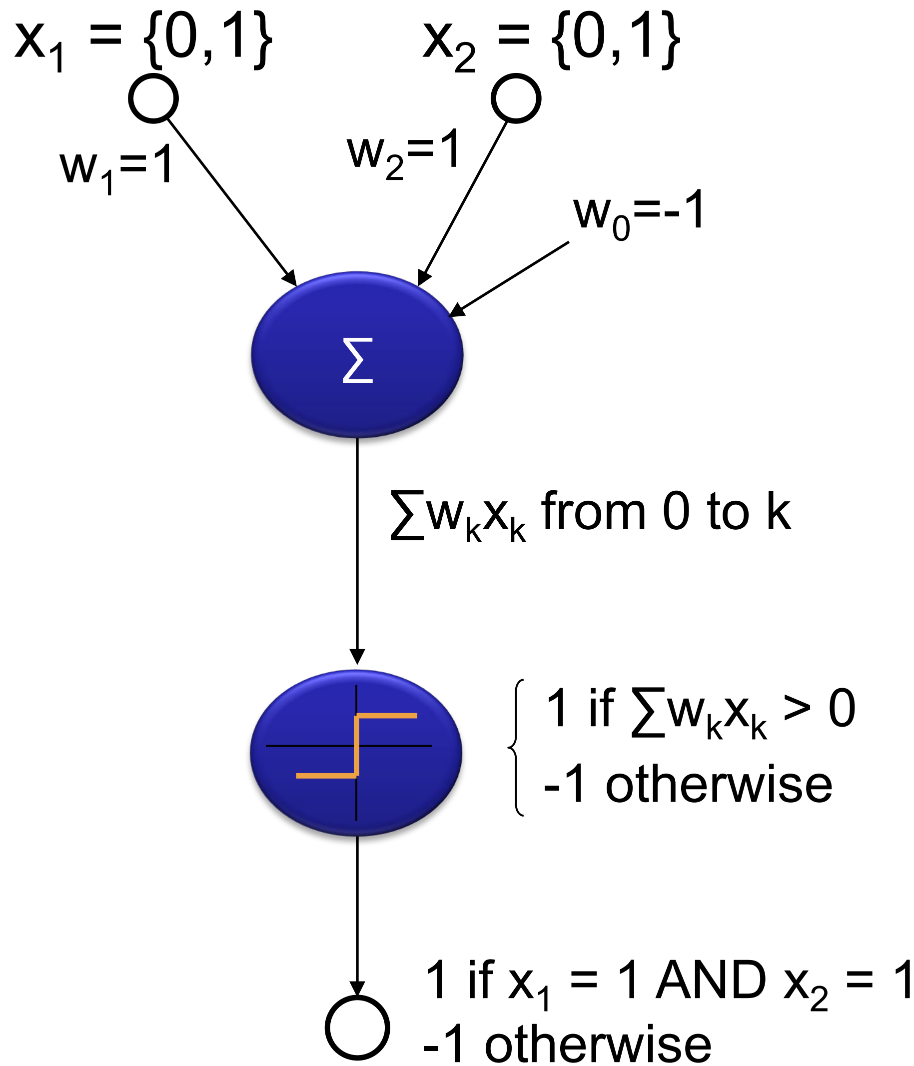
The Perceptron



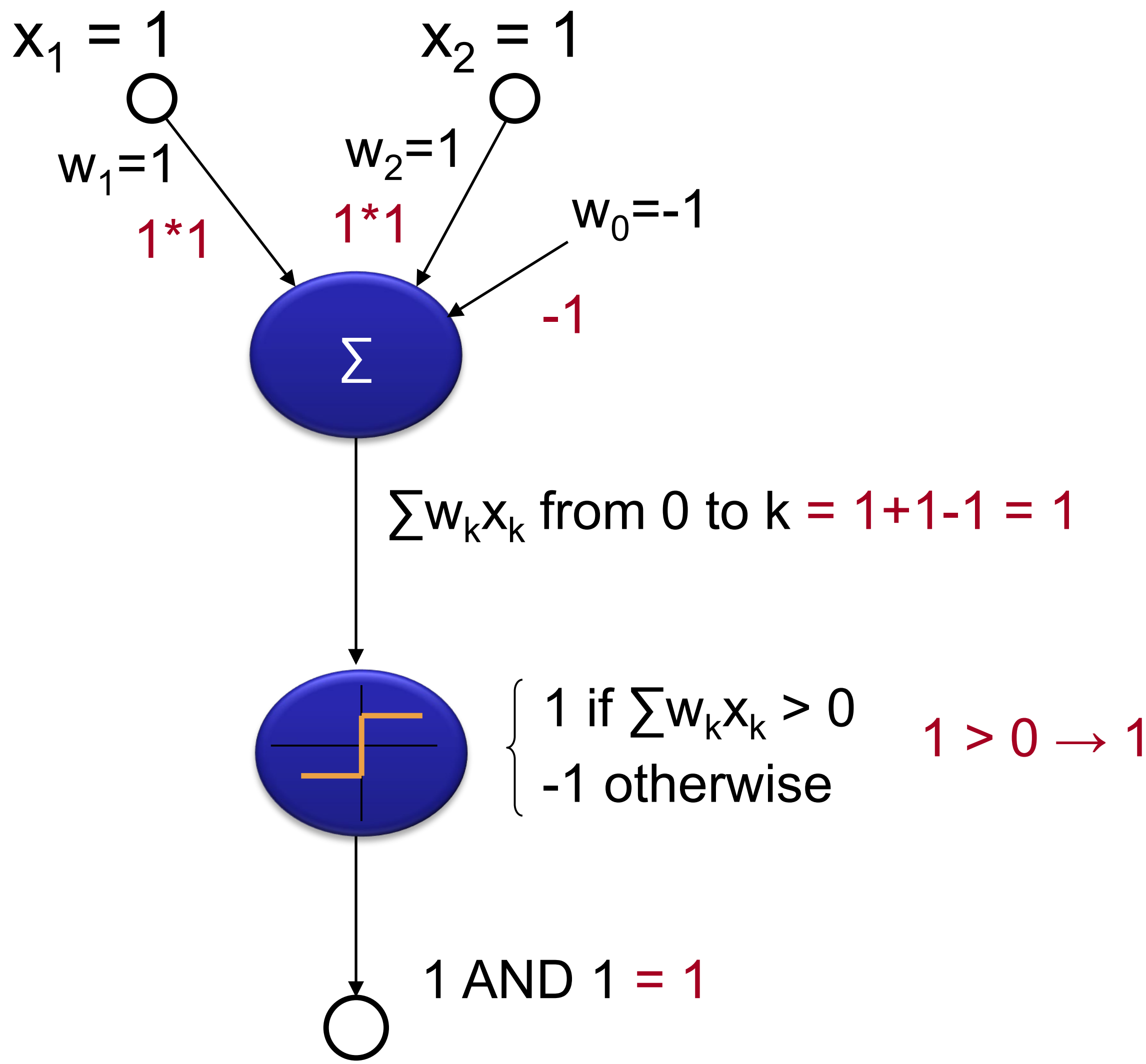
Perceptron Training

- If a misclassified instance is encountered:
    - Change parameters of the hyperplane so the instance moves closer to it.
- Each iteration goes through all training instances.
- Iteration continues until a solution is found.
- Only works if the classes can be linearly separable.

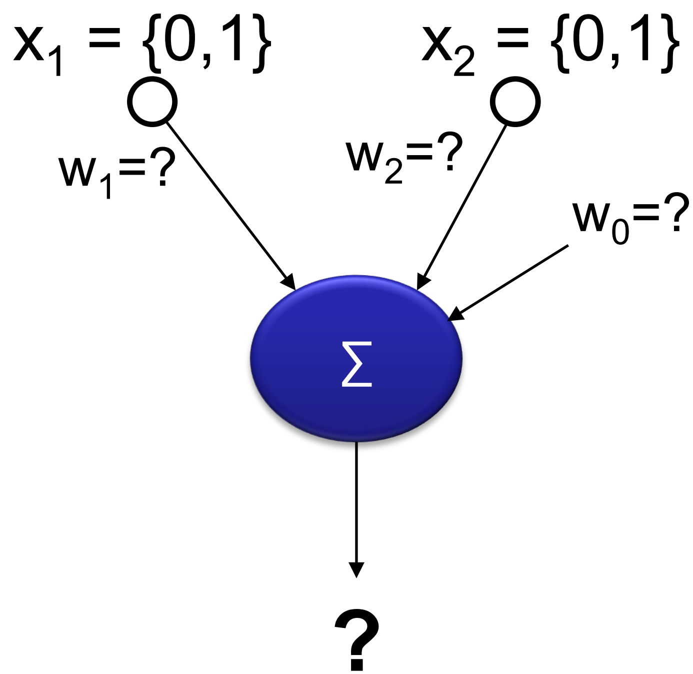
Boolean functions - AND



Boolean Functions - AND

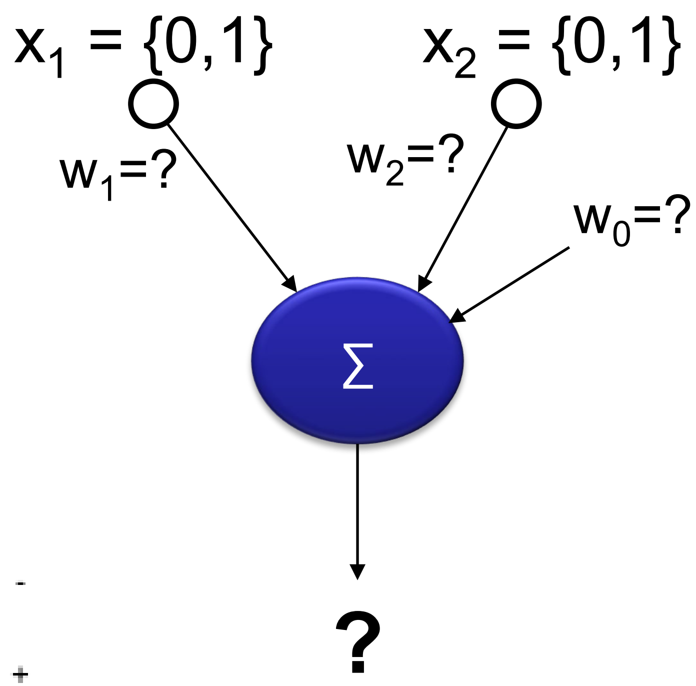


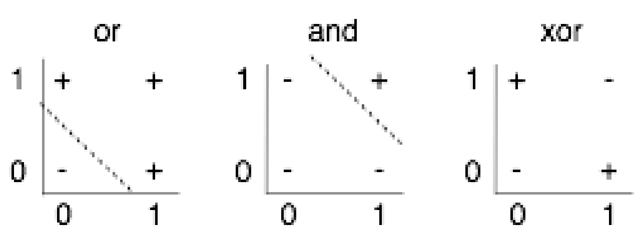
Works similarly with OR, but what about XOR?

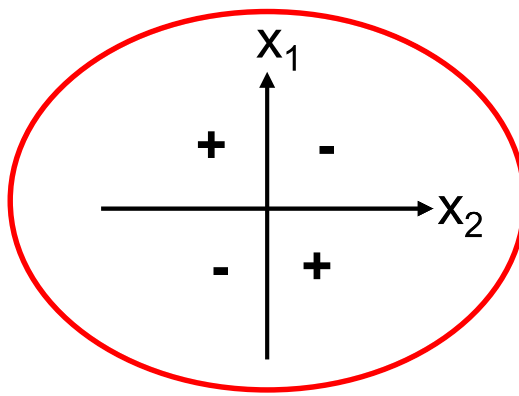


XOR:
1 if $x_1 \neq x_2$
-1 otherwise

Boolean functions - XOR

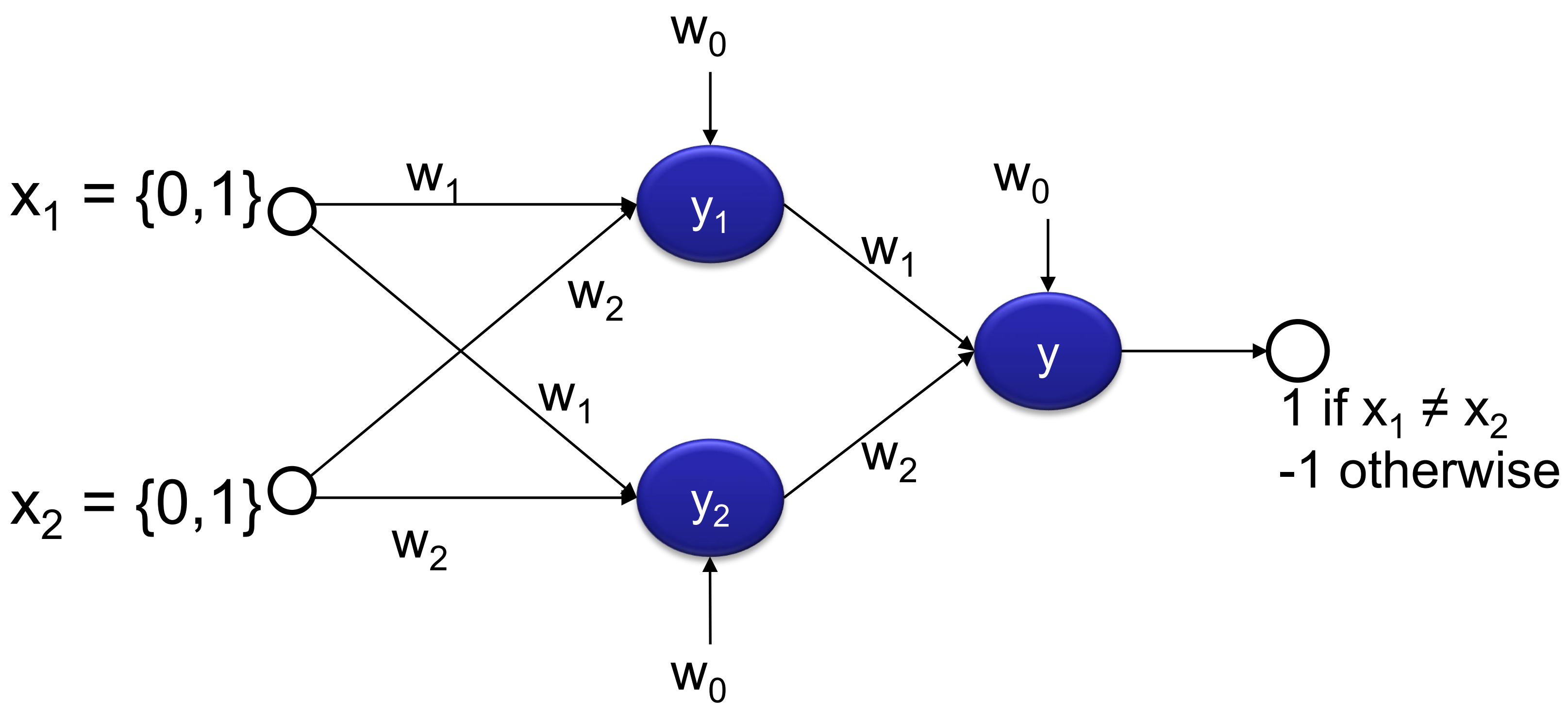






Not linearly separable!

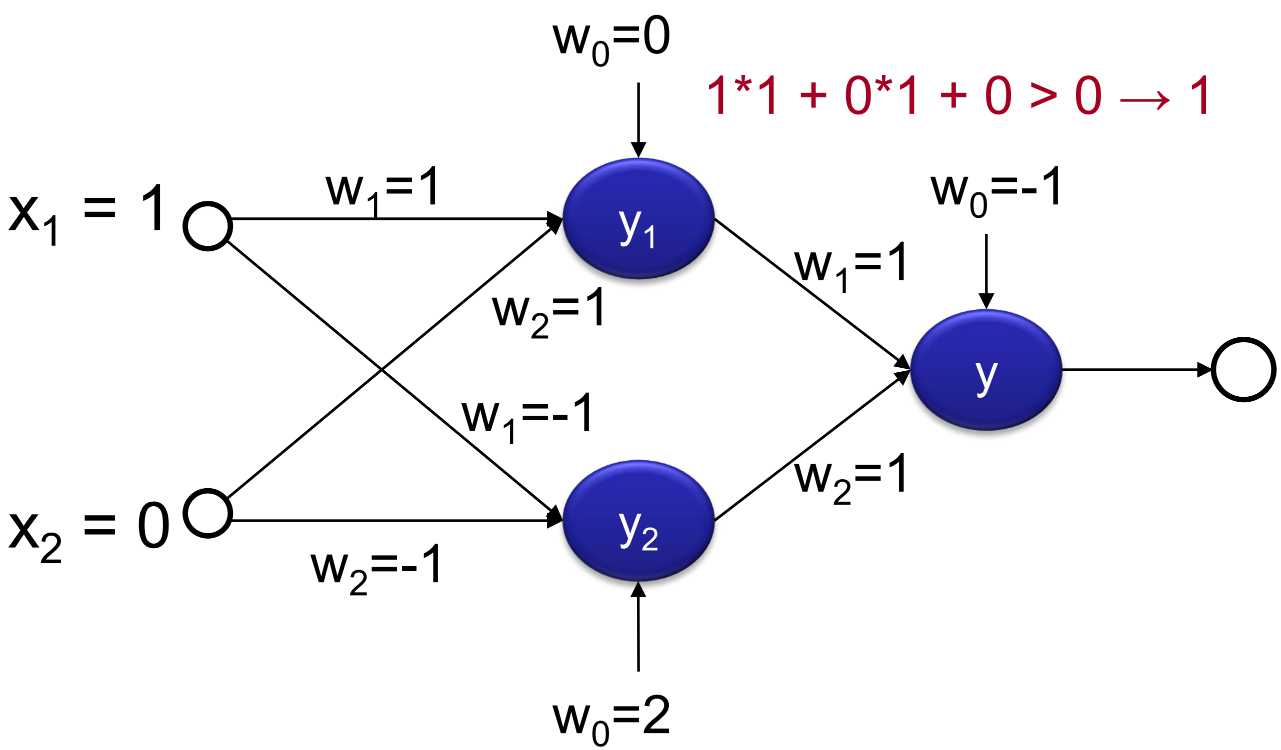
XOR – The Solution



Multi-layered network! Boolean logic tells us that XOR can be expressed as:

$$\left.\begin{array}{l} y_1 = x_1 \text{ OR } x_2 \\ y_2 = \text{NOT}(x_1 \text{ AND } x_2) \\ y = y_1 \text{ AND } y_2 \end{array}\right\} \quad y = x_1 \text{ XOR } x_2$$

XOR – True

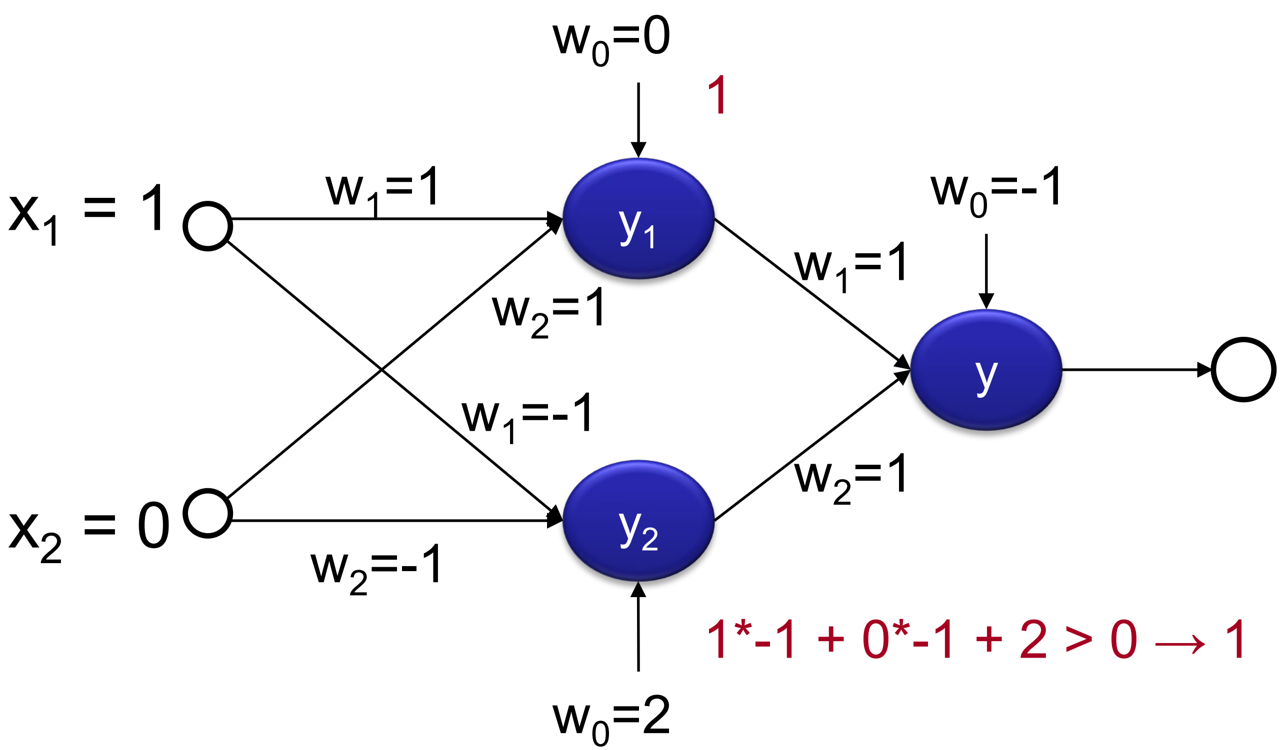


$y_1 = x_1$ OR x_2

$y_2 =$ NOT(x_1 AND x_2)

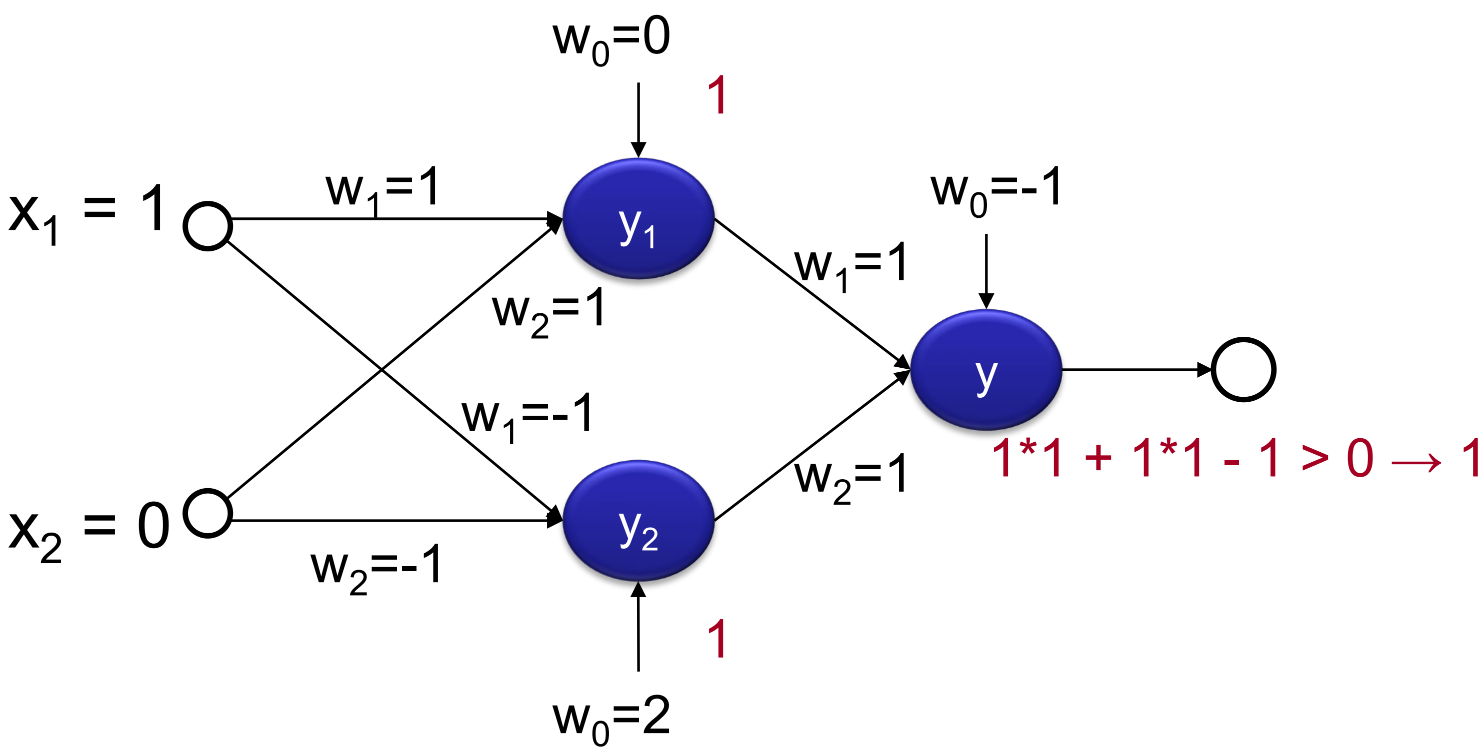
$y = y_1$ AND y_2

XOR – True



$y_1 = x_1$ OR x_2
$y_2 =$ NOT(x_1 AND x_2)
$y = y_1$ AND y_2

XOR – True

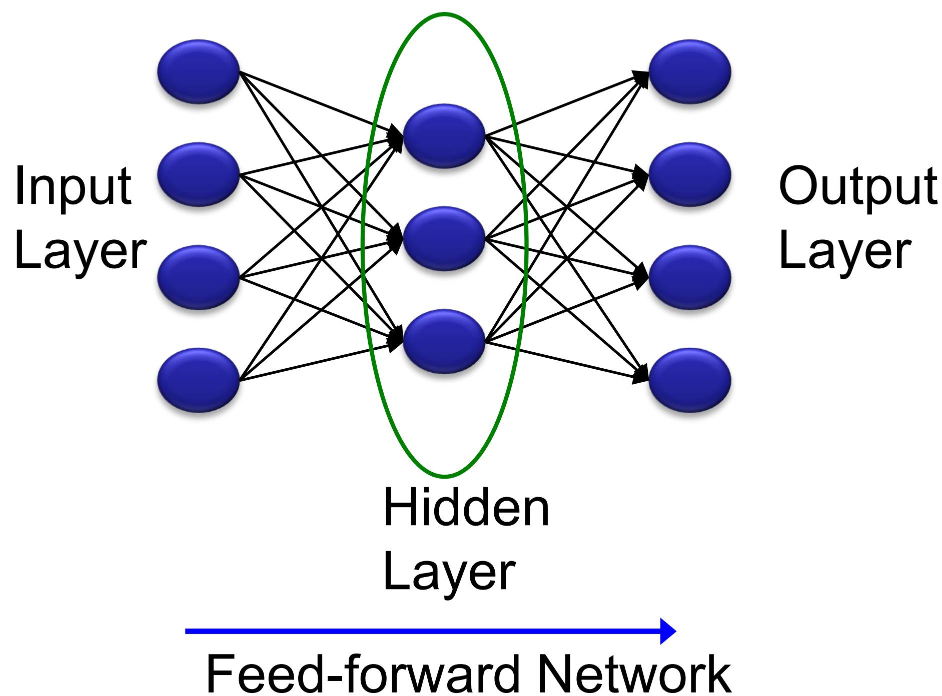


$y_1 = x_1$ OR x_2
$y_2 =$ NOT(x_1 AND x_2)
$y = y_1$ AND y_2

1 XOR 0 = 1

A *linearly inseparable* function, like XOR, can be expressed in a multi-layered perceptron network!

Also known as an Artificial Neural Network (ANN).

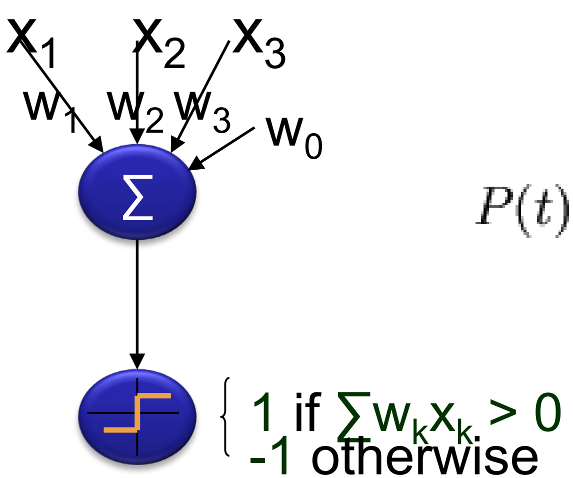


How do we learn the weights?

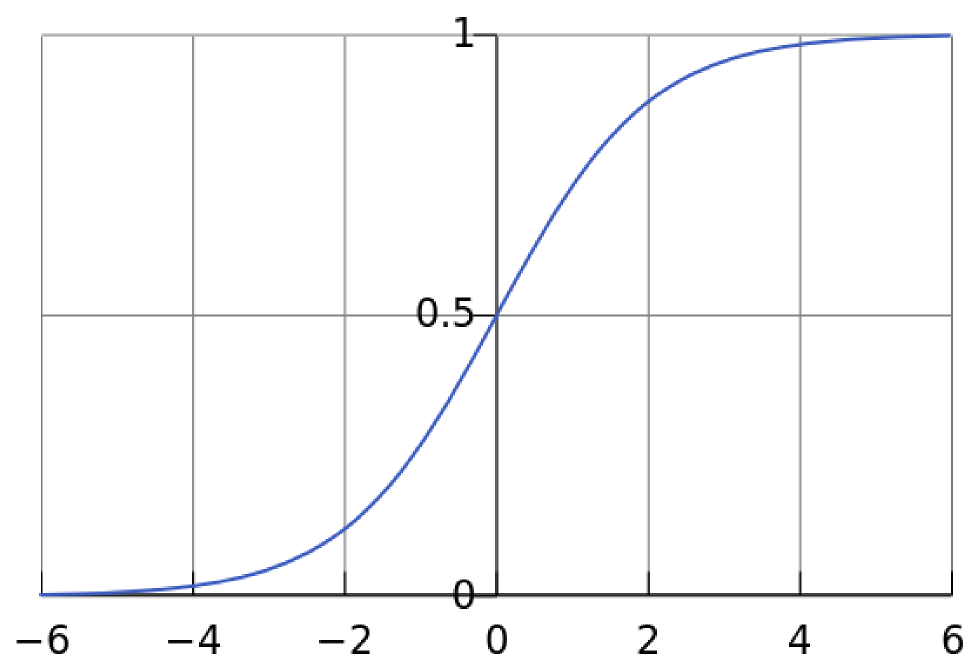
- Feed an instance through the network.
- If not correctly classified, calculate the squared error.
- Propagate the error backwards through the network and modify the weights.
- Continue for all training instances and until the total error is below a threshold.
- *Backpropagation* is a common algorithm for training ANNs.

Backpropagation

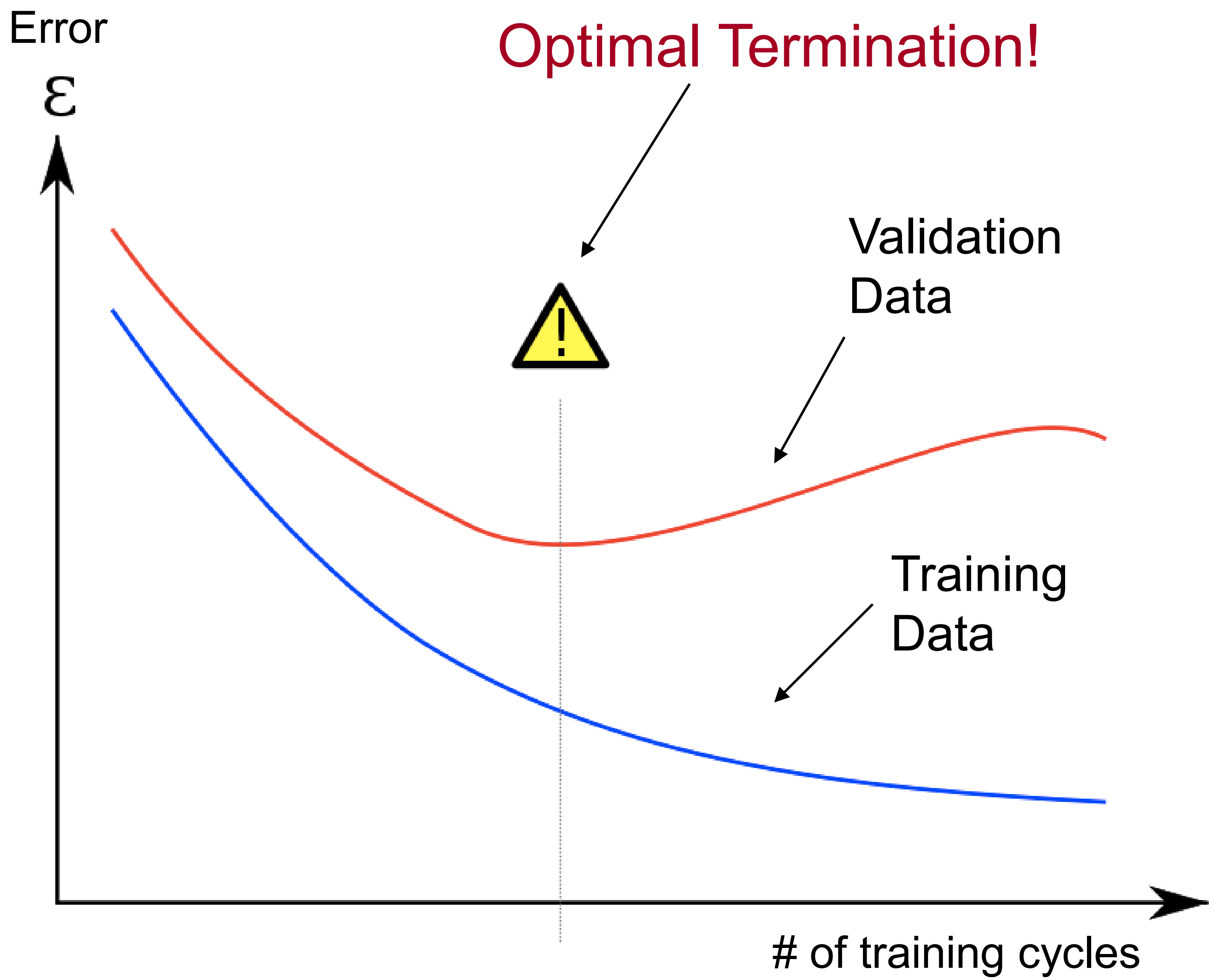
- BP does however require that the activation functions is differentiable for all *x*.
- A step function (-1 if below threshold, +1 if equal or above threshold) is not differentiable.
- Instead we use a sigmoid function.



$$P(t) = \frac{1}{1 + e^{-t}}$$



Overfitting

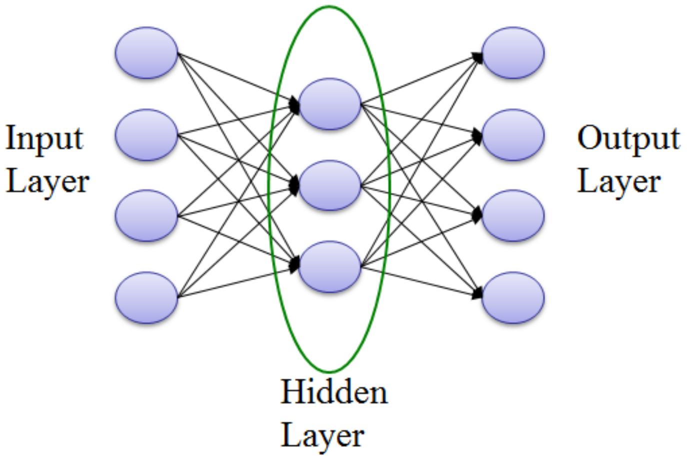


Overfitting occurs when a model begins to “memorize” training data rather than “learning” to generalize from trend

Properties of ANNs

- Good at handling inconsistent, noisy data.
  - Good at handling missing or unimportant attributes.
- Learning is usually <u>very</u> slow.
- Risk of overfitting.
- Not understandable by humans.
- Requires lots of diverse data.
- Image classification is a problem very well suited for ANNs.

ANN design



- No general rules for designing the layers.
- Start with one input node for each attribute, and enough output nodes to represent all classes.
- Try with a small number (3-5) of hidden nodes and see how it works. Increase if needed.
- Larger networks are slower to train, but does not have to be better at classifying your data!

Support Vector Machines

- Linear Models can only represent Linearly Separable classes.
- For many applications this is too simple.
  - The XOR problem

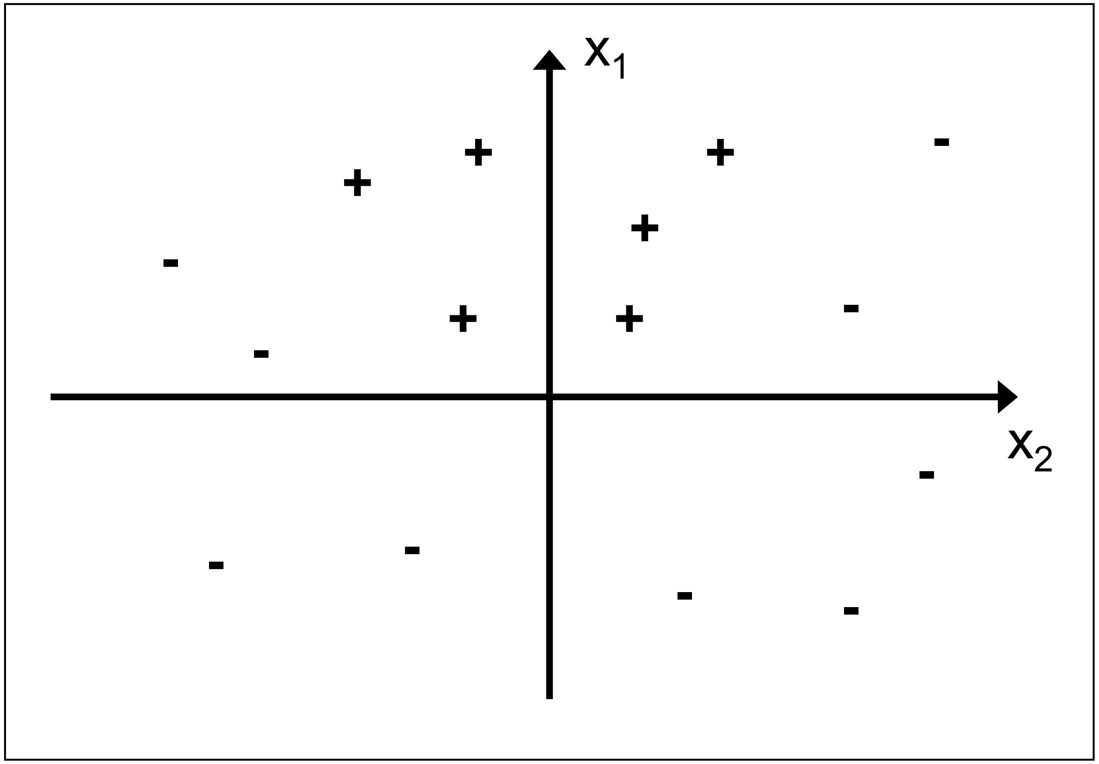
- What if we somehow can transform a non-linear problem to a linear problem?

Non-linear Mapping

- A non-straight line in one space, can in fact be straight in another space.
- So by transforming the class boundary to another space, the problem can be linearly separable.

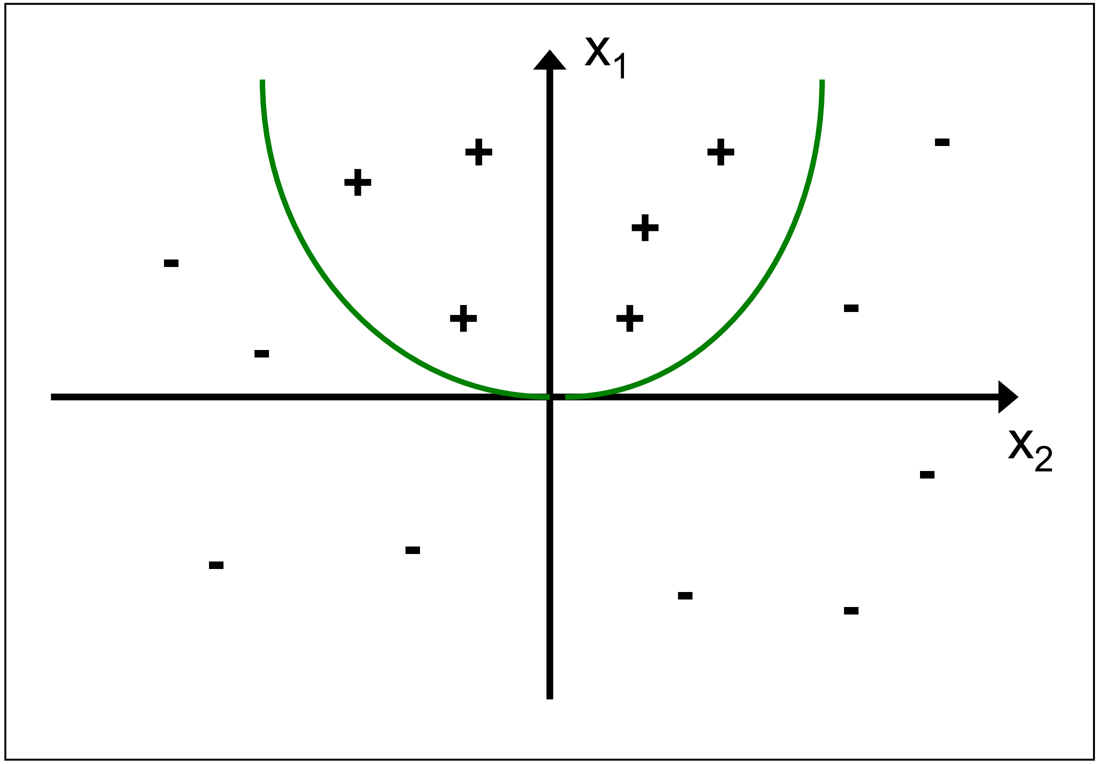
- Better understood with an example!

Non-linear Mapping - Example



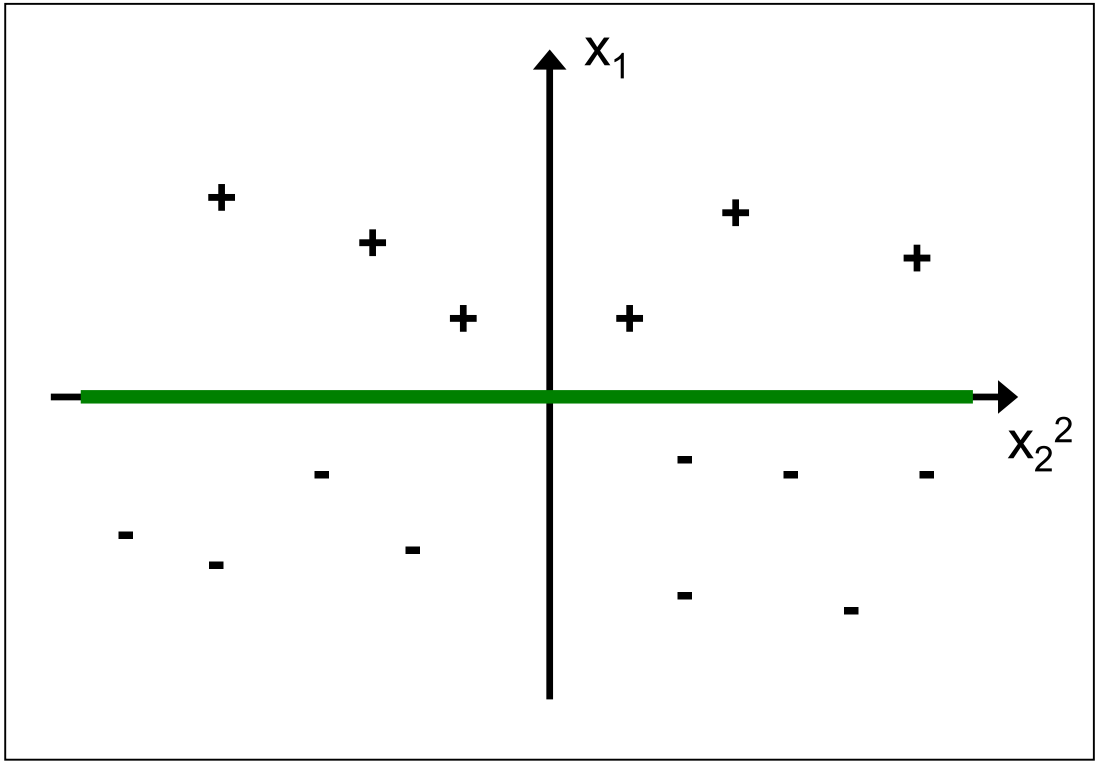
Not linearly separable in the (x_1,x_2) plane

Non-linear Mapping - Example



Separable, but not a straight line ($x_1=x_2^2$)

Non-linear Mapping - Example



Separable in the (x_1,x_2^2) plane!

Non-linear Mapping

- In fact, polynomials of sufficiently high degree can represent any function accurately:
  - $x=w_1x_1^3+w_2x_1^2a_2+w_3x_1x_2^2+\ldots$
- In a linear model, weights are learned to find a plane that separate the classes.
- In a non-linear model, weights and coefficients are learned to find a plane that separate the classes in some space.

EVALUATION

Evaluation

- Evaluation is a key element in successful machine learning experiments.
- We need to know how different methods work and compare them with each other.
    - Is ANN better than KNN on my data?
- Accuracy = how many instances correctly classified.
    - 9/14 = 64.3%

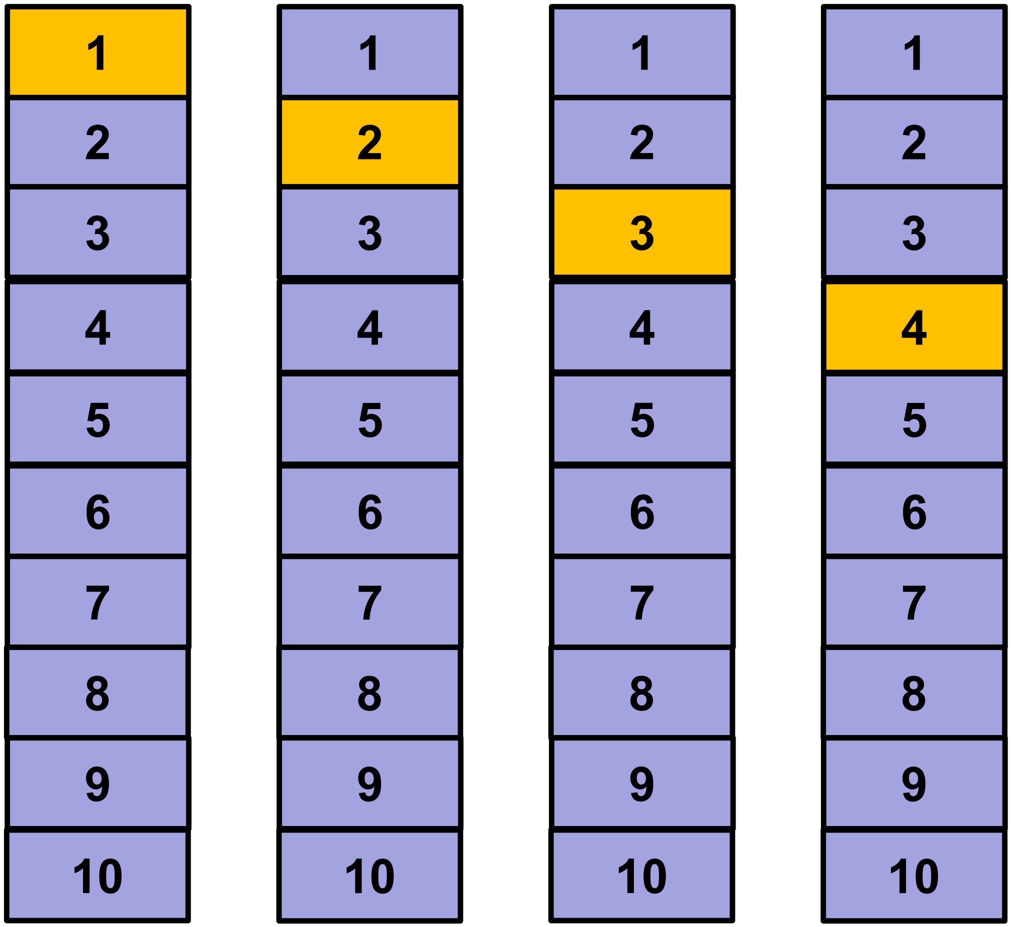
Data usage

- All known data can be used for training and validation.
- Not a very good idea, since we don’t test the generalization capabilities of the classifier.
- = does not prove that our system works on new data!

Data usage

- The data can be divided into a training set and a validation set.
  - Use the training set to train the classifier.
  - Use the validation set to see how well it worked.
- Better, but very sensitive to how we divide the data!

Data usage – Cross Validation

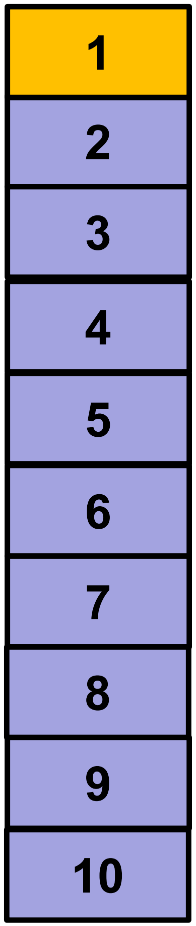


10-fold Cross Validation = divide data into 10 parts.

9 parts are used for training, 1 part for validation.

Iterate until all parts have been used for validation.

Data usage – Cross Validation



- More accurate than using one training set and one validation set.
- Drawback is increased training time.
- Still sensitive to how we divide the data, especially for small datasets.

Is Accuracy a good metric?

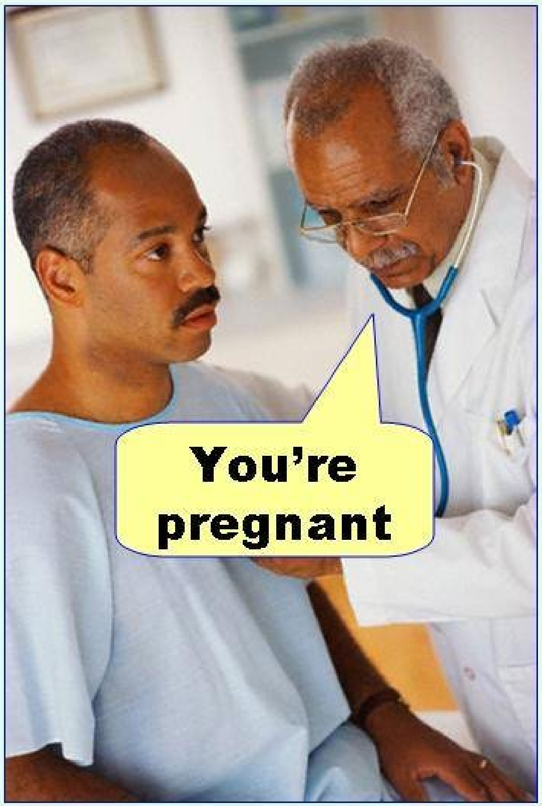
- Simple to calculate, but is often overly optimistic.

- TP = true positives
- FP = false positives
- TN = true negatives
- FN = false negatives

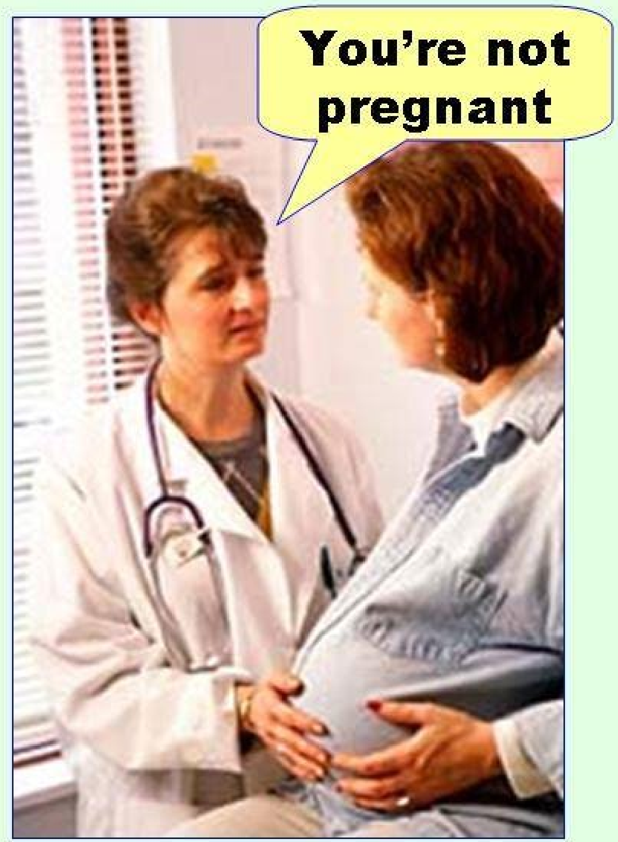
A more complete description of the learner.

A good way to put it…

Type I error
(false positive)

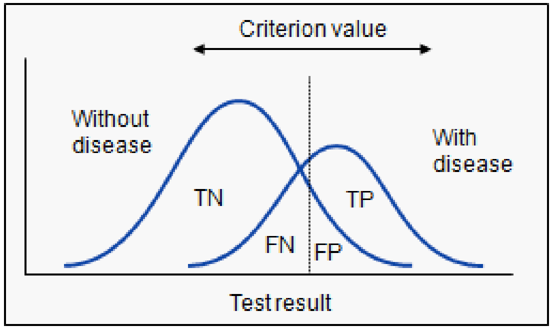


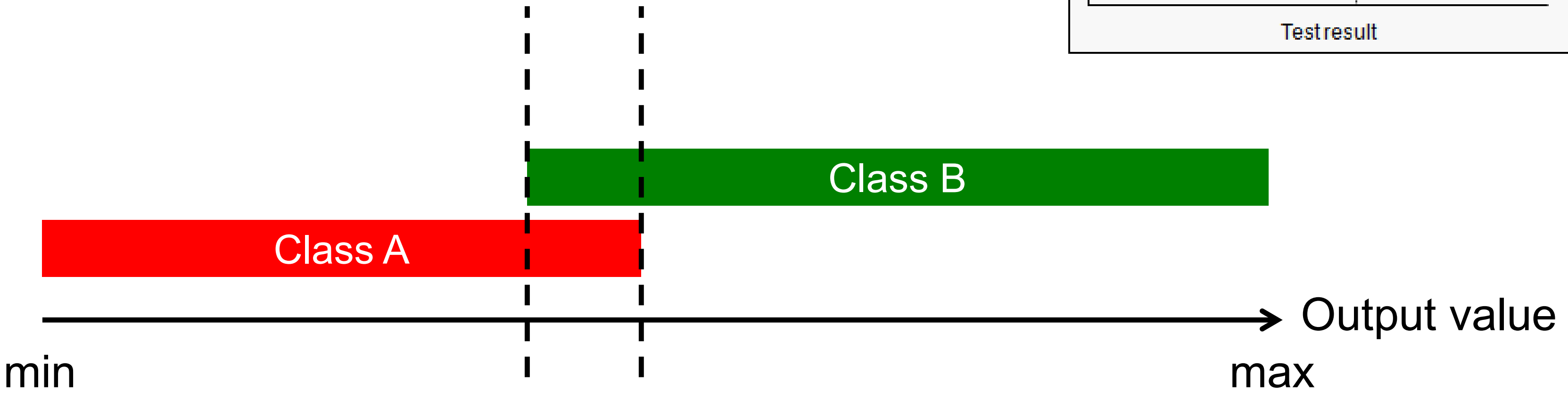
Type II error
(false negative)



Discrimination Threshold

Depending on where we put the discrimination threshold, the TPR and FPR will vary.





TPR = TP / (TP + FN)
FPR = FP / (FP + TN)

- Weka is a free project containing a large number of machine learning algorithms.
- It can be used with a GUI or through a Java API.
- It supports data in csv or arff format.
- Website:
    - http://www.cs.waikato.ac.nz/ml/weka/

That was all for this lecture

http://etc.ch/45Cv

Acknowledgements

Dr. Johan Hagelbäck

Linnæus University

johan.hagelback@lnu.se

http://aiguy.org