INTRODUCTION TO ARTIFICIAL INTELLIGENCE

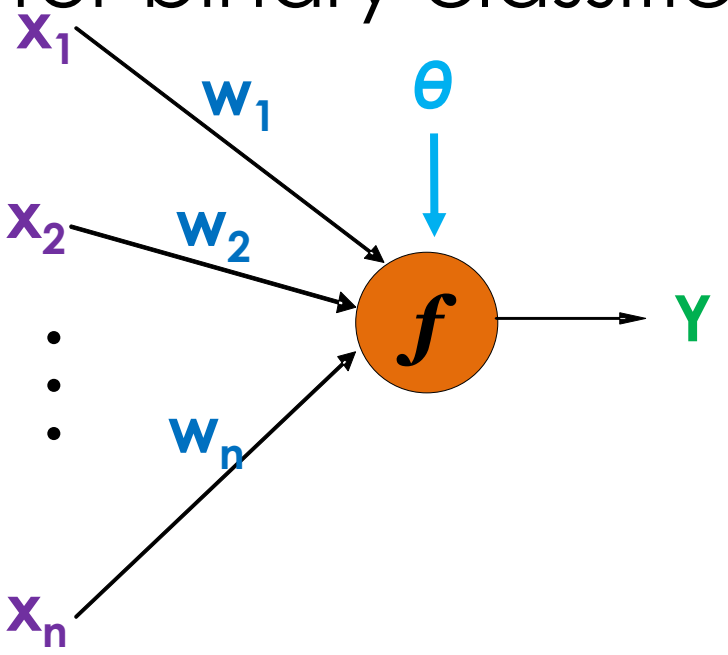
# 6. INTRODUCTION TO **MACHINE LEARNING: EVALUATION**

# The plan for today

- Continue our discussion on Neural Networks
  - Introduce Multi-Layer Networks (not assessed in exam)
- Learn how we can evaluate the performance of algorithms (with a focus on classifiers)
  - Bootstrapping and cross-validation methods
  - Confusion Matrix
    - Visualise performance with a ROC graph
- Discuss overfitting
  - Decision tree pruning
- Work on some sample exam questions.

# Last week: perceptron

- Simplest NN
- Single-layer, feed-forward NN
- Used for binary classification




- $X_i$ is a vector of inputs
- $W_i$ is the weight (real numbers expressing the importance of the respective input to the output)
- **F** is a function, called activation function
- **Y** is the output, which is a single binary value
- $\theta$ is the threshold (can also be considered a weight)

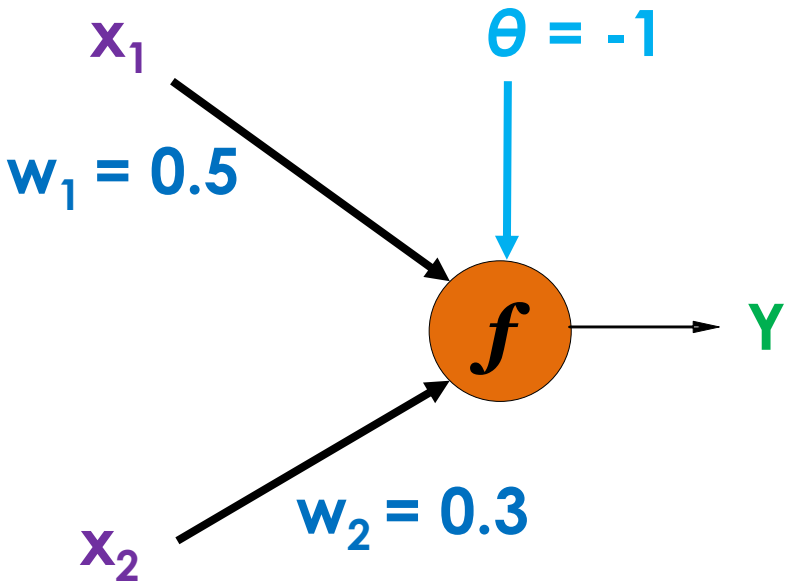
# Calculating the output

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

- $x_1 = 2$ and $x_2 = 1$

$(0.5 * 2) + (0.3 * 1) - 1 = 0.3$

As 0.3 > 0, the output *Y* will be +1



# Learning (updating weights/threshold)

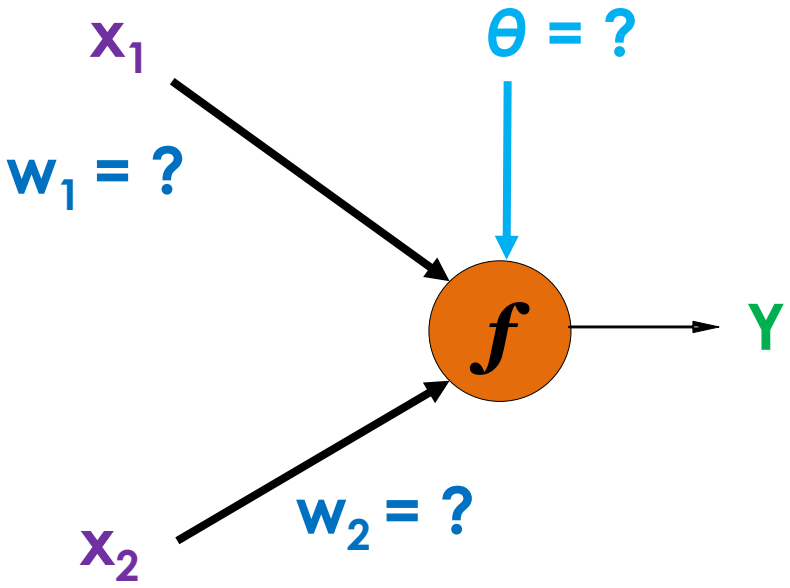
- $x_1 = 2, x_2 = 1$
  $w_1 = 0.5, w_2 = 0.3, \Theta = -1$
- The output was calculated as **Y = 1**
- However, the true output is **$Y^d = 0$**
- How should I update the weights and threshold?

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

$$\theta(p+1) = \theta(p) + \Delta\theta(p)$$

$$\Delta w_i(p) = (Y^d - Y)x_i$$

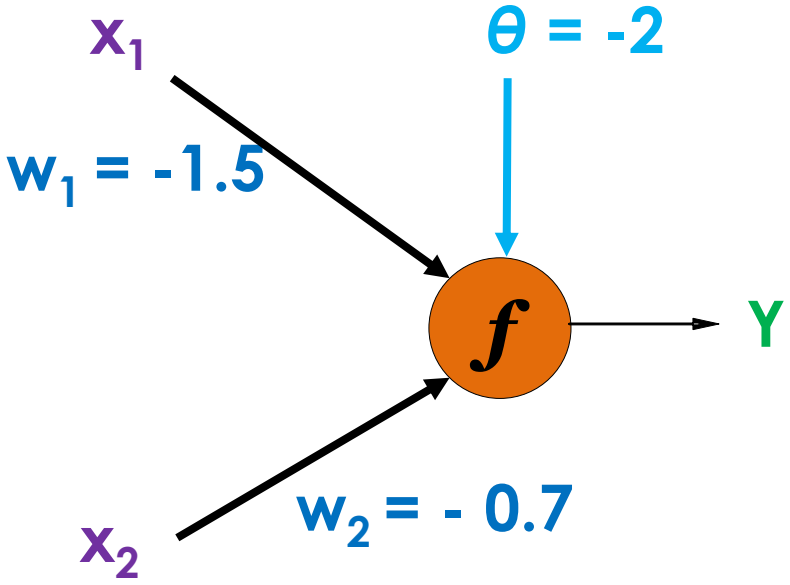
$$\Delta\theta(p) = (Y^d - Y)$$



# Learning

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

- $x_1 = 2$, $x_2 = 1$
  $w_1 = -1.5$, $w_2 = -0.7$, $\Theta = -2$
- The output was calculated as **Y = 1**
- However, the true output is **$Y^d$ = 0**
- How should I update the weights and threshold?




What is the output Y now?

(-1.5 * 2) + (- 0.7 * 1) – 2 =
- 3 – 0.7 – 2 = - 5.7
Y = 0 so we stop

# Perceptron - summary

- Usually the perceptron is initialised with random weights between 0 and 1
- Cases are presented to the perceptron one by one and the weights updated
- If at least one error has been made during this *epoch* then the entire set of cases are presented again
- Repeated until no error is made
- This learning procedure is guaranteed to **converge** to a solution if the problem is **linearly separable**

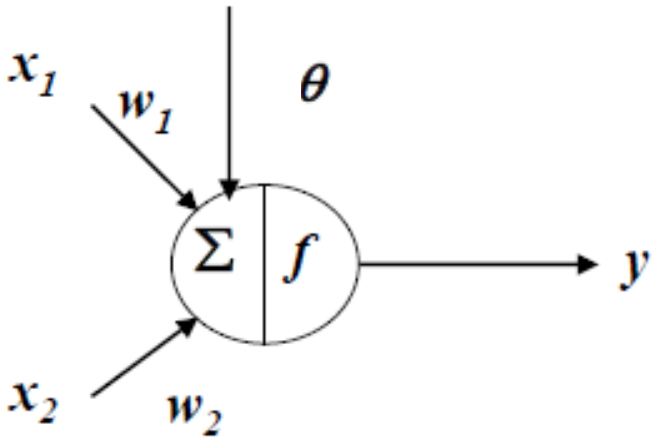
# Perceptron example

- Example: logical AND operator
- Weights and threshold:

$w_1 = 1$, $w_2 = 1$, $\theta = -1.5$

- Activation function: Step

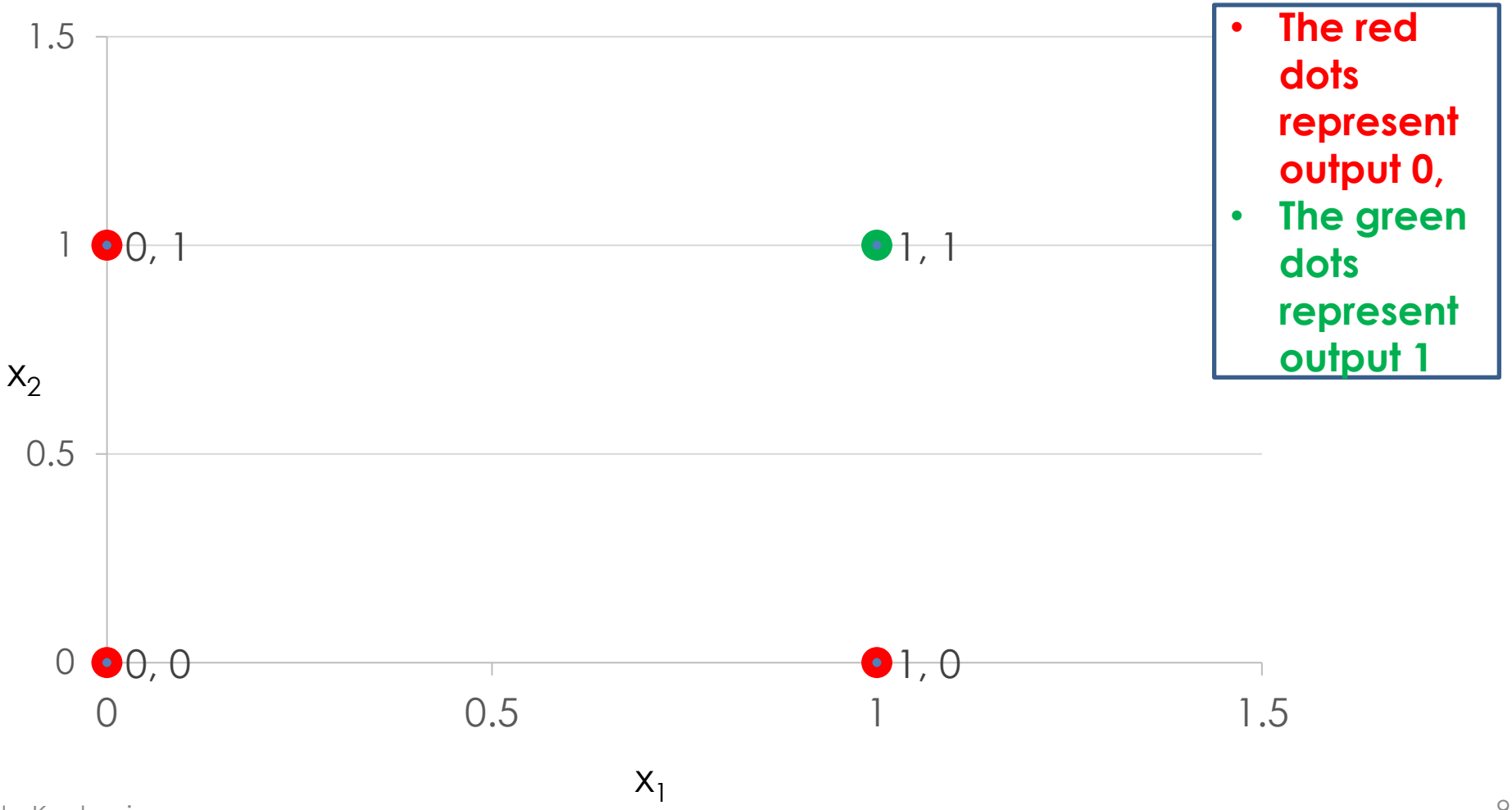
$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$



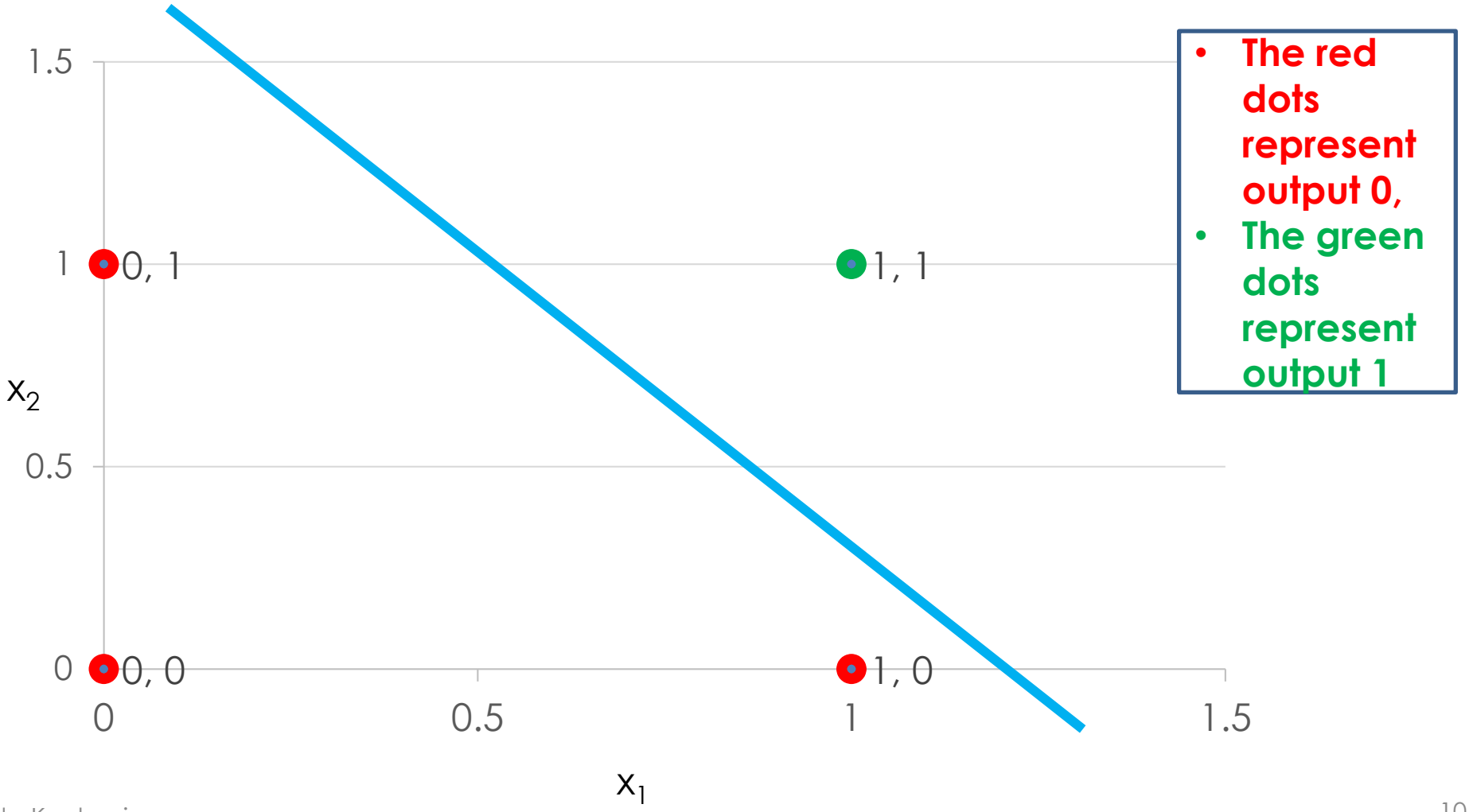

- For inputs 0/1, it correctly **classifies** to 0/1
  - remember the truth-tables

| X1 | X2 | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

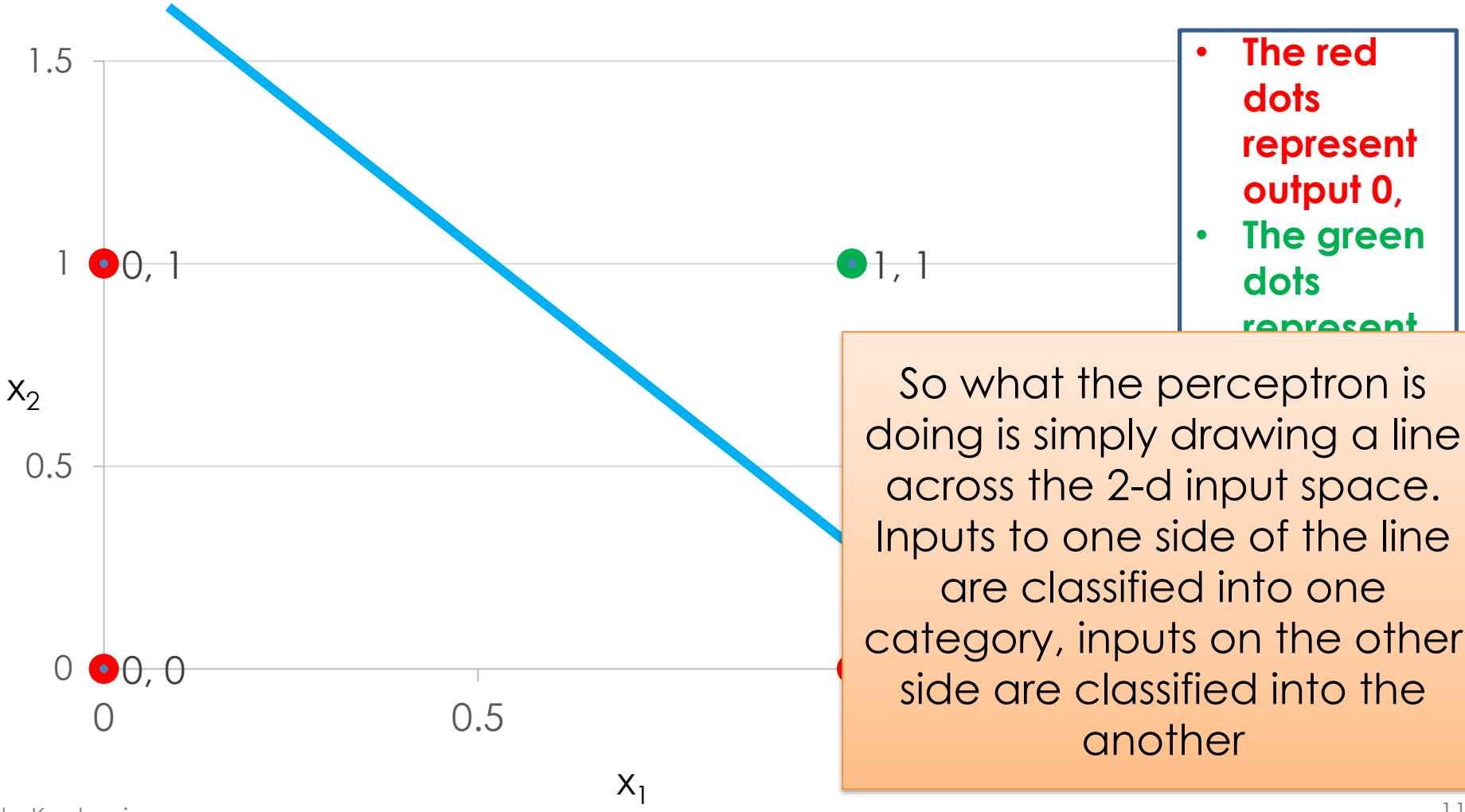
# Let's plot the inputs



# Let's separate the outputs



# This is an example of linearly separable data!




- **The red dots represent output 0,**
- **The green dots represent**

So what the perceptron is doing is simply drawing a line across the 2-d input space. Inputs to one side of the line are classified into one category, inputs on the other side are classified into the another

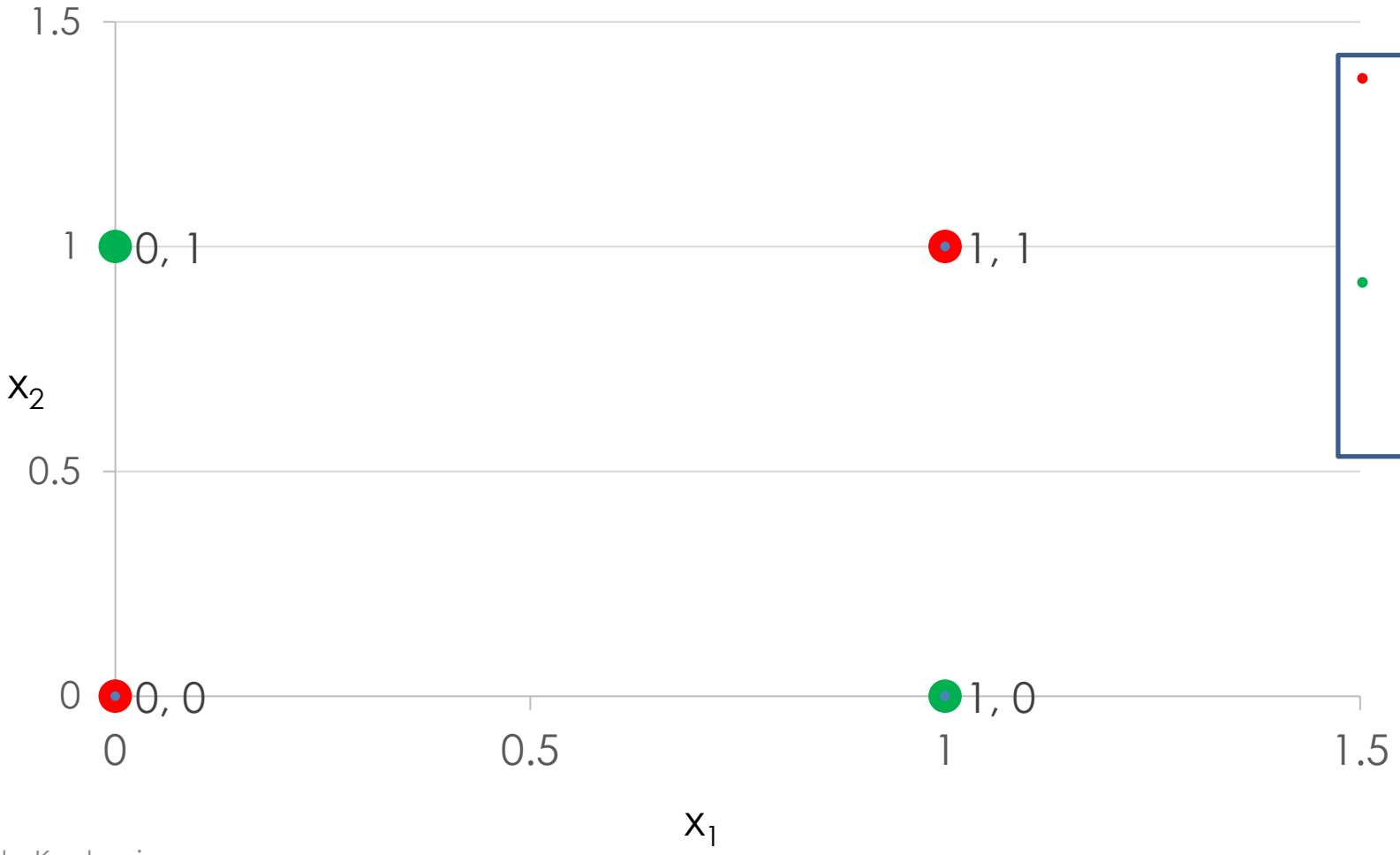
# Example: XOR

- Can the perceptron model XOR operation and correctly classify to 0/1?
- Inputs $x_1$, $x_2$: 0, 1
- Will the algorithm ever converge by updating weights?

| Input variables | | Exclusive-OR |
|---|---|---|
| $x_1$ | $x_2$ | $x_1 \oplus x_2$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Let's plot the inputs

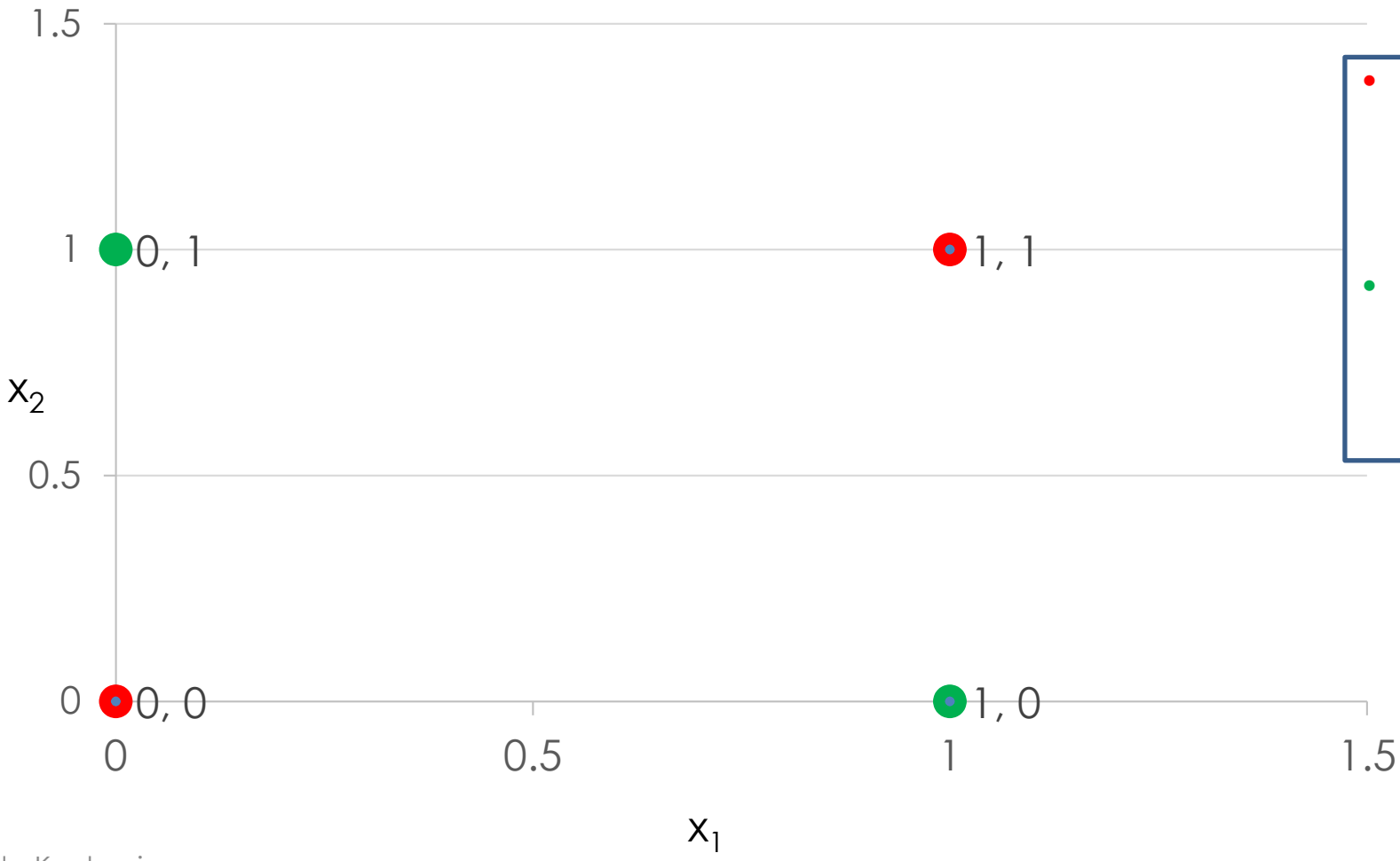
| Input variables $x_1$ | $x_2$ | Exclusive-OR $x_1 \oplus x_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




- The red dots represent output 0,
- The green dots represent output 1

# Let's separate the outputs…?

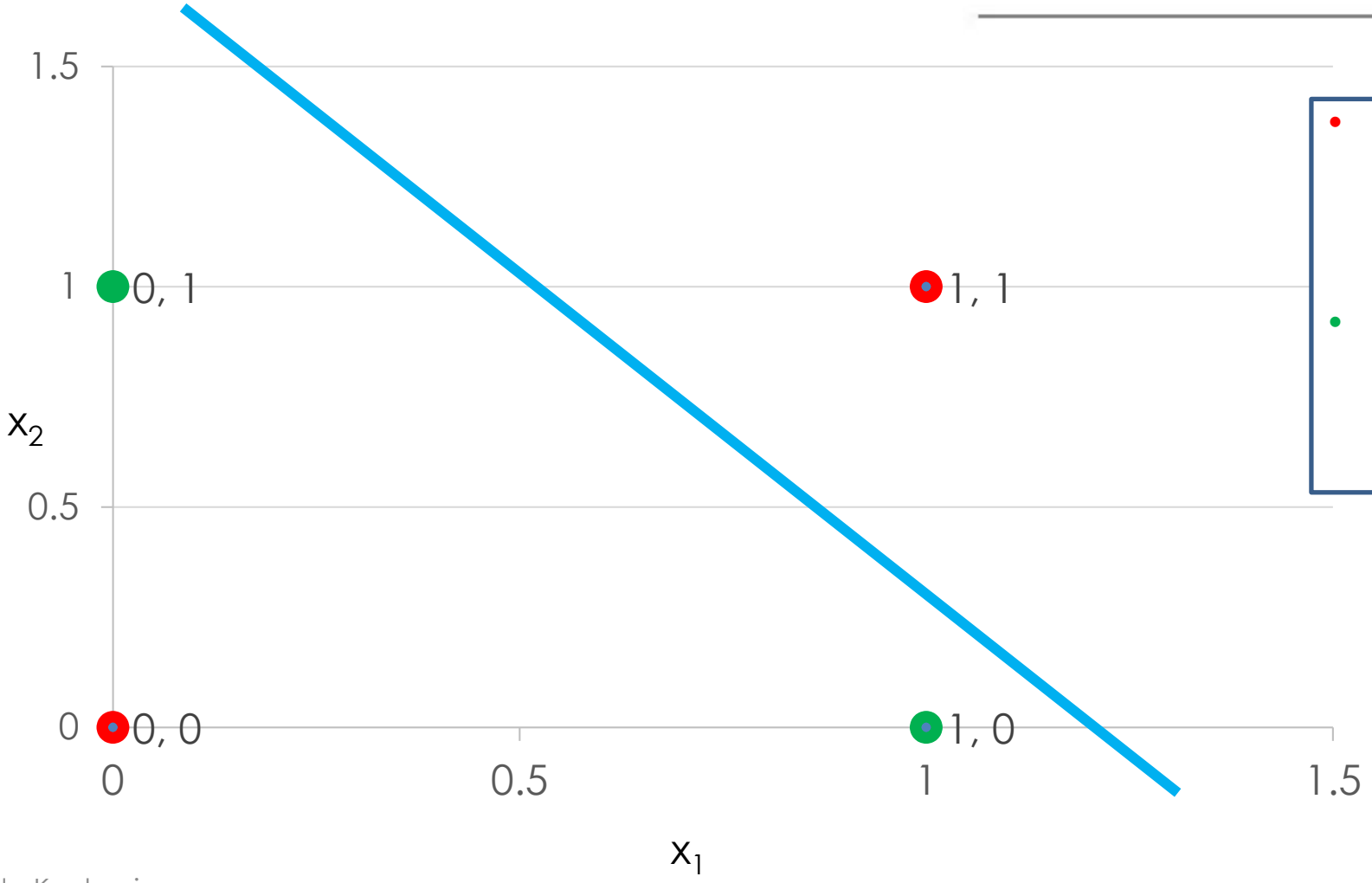
| Input variables | | Exclusive-OR |
|---|---|---|
| $x_1$ | $x_2$ | $x_1 \oplus x_2$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




- The red dots represent output 0,
- The green dots represent output 1

# Let's separate the outputs…?

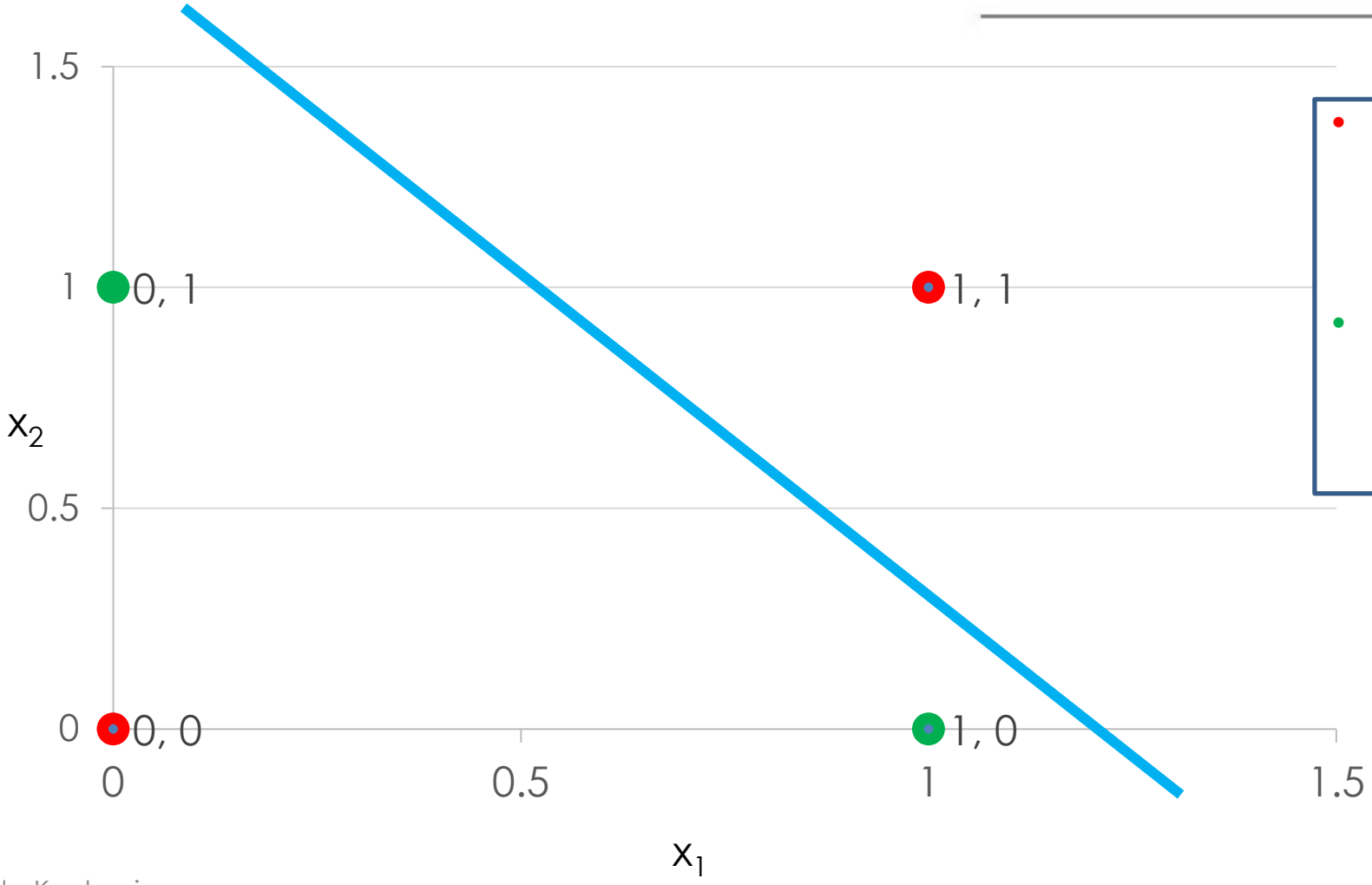
| Input variables $x_1$ | $x_2$ | Exclusive-OR $x_1 \oplus x_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




- The red dots represent output 0,
- The green dots represent output 1

# The outputs are not linearly separable

| Input variables $x_1$ | $x_2$ | Exclusive-OR $x_1 \oplus x_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




- The red dots represent output 0,
- The green dots represent output 1

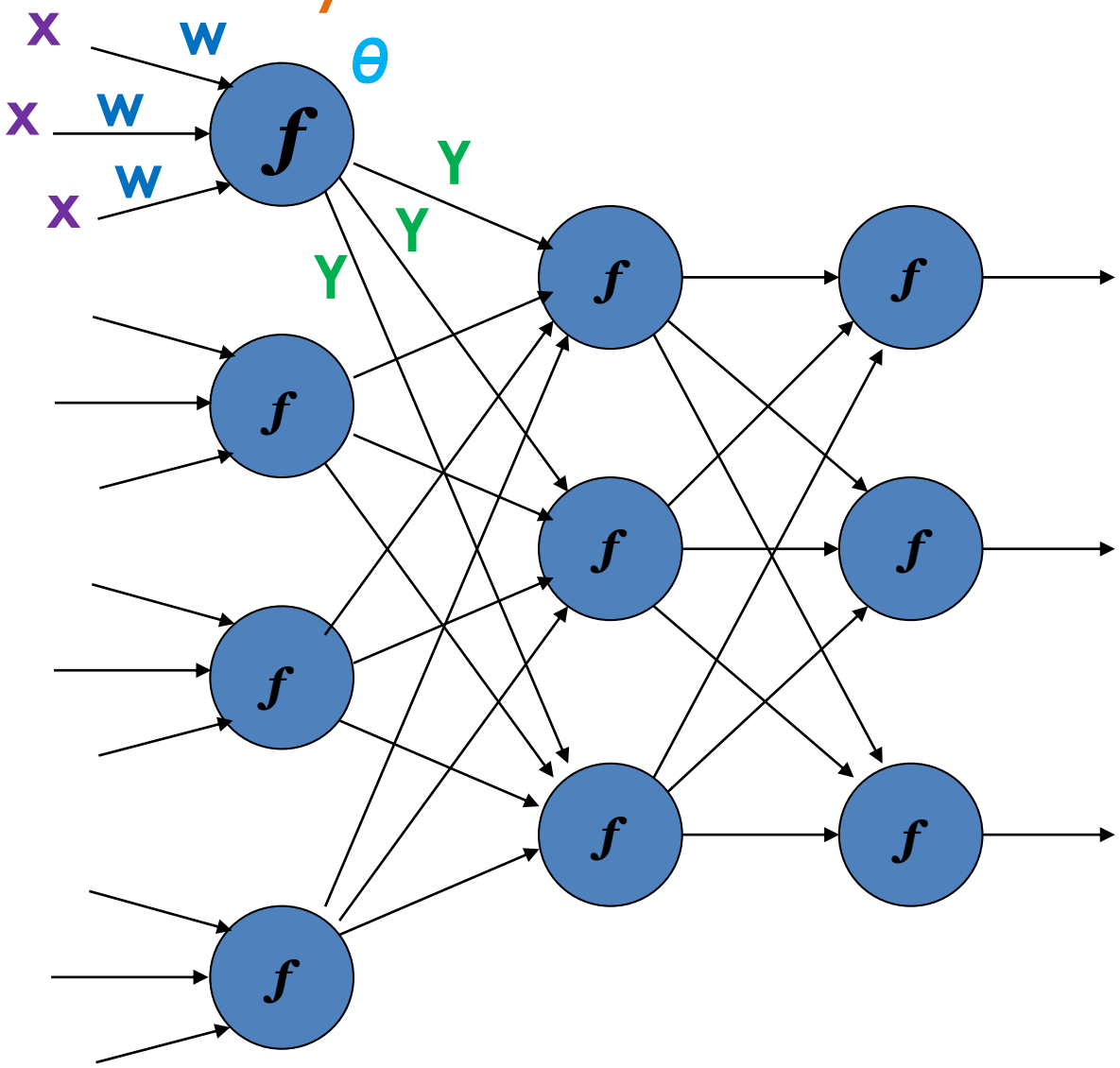
# The XOR Problem

- The ‘single-layer’ perceptron can't model XOR.
- The reason is because the classes in XOR are not linearly separable.
  - You cannot draw a straight line to separate the points (0,0),(1,1) from the points (0,1),(1,0).
- Led to invention of multi-layer networks.

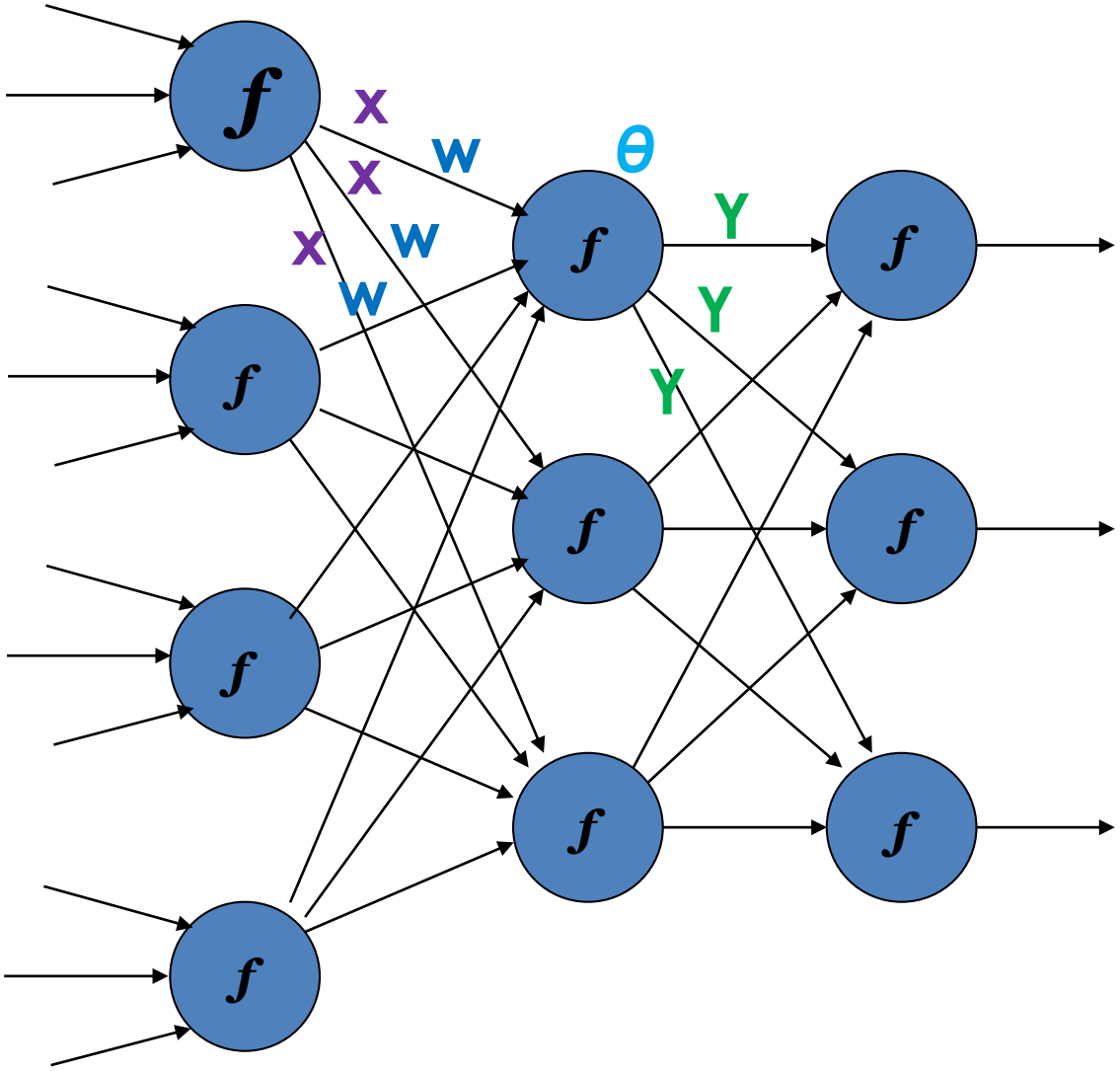
# Multilayer Neural Networks

- Natural extension to the perceptron
- Deals with non-linearly classifiable data
- Essentially involves linking numerous perceptrons together
- Fully connected: All nodes from one layer connected to next
- Can be extended to any number of outputs and hidden layers

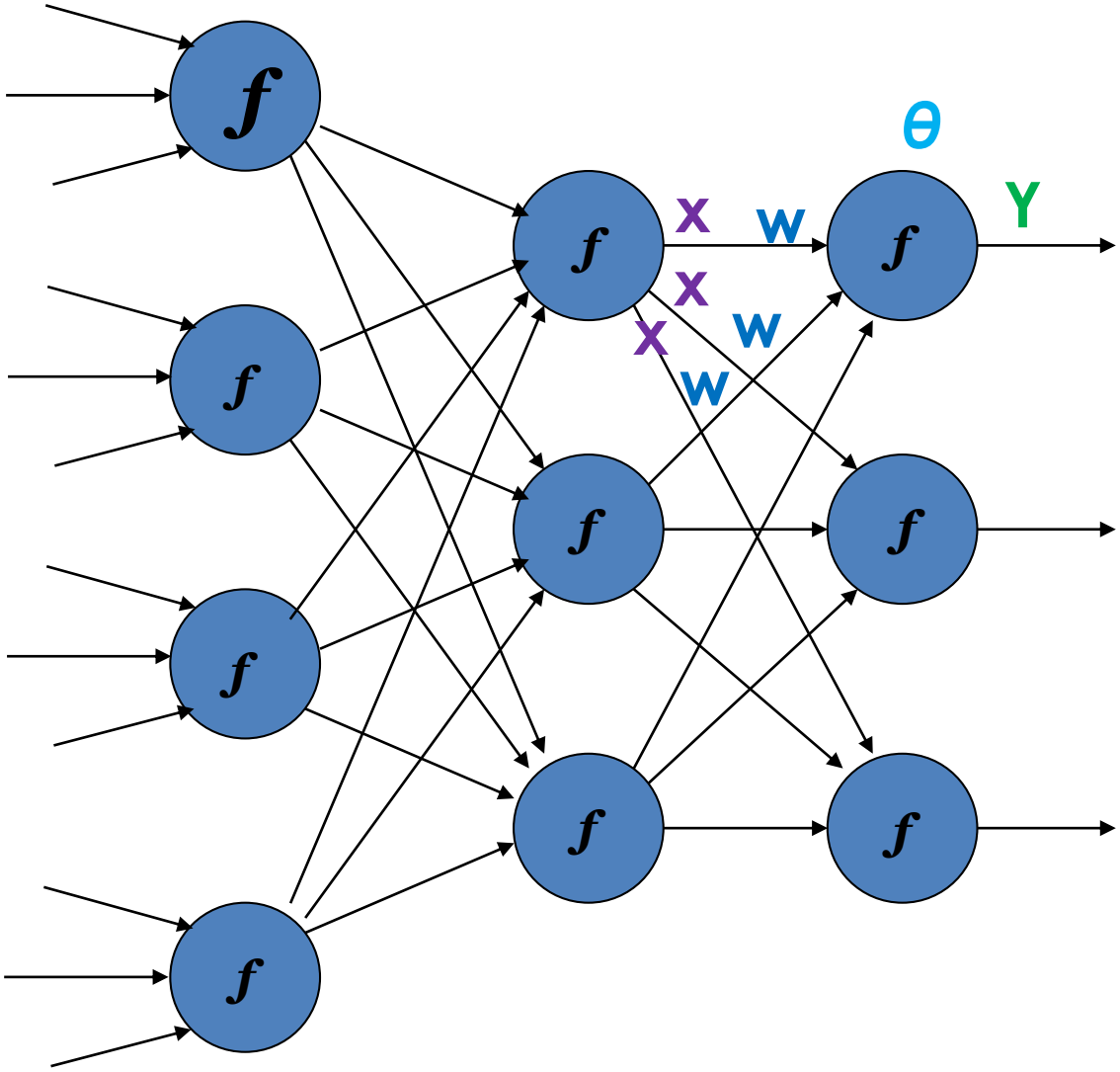
# Multilayer Neural Networks



# Multilayer Neural Networks

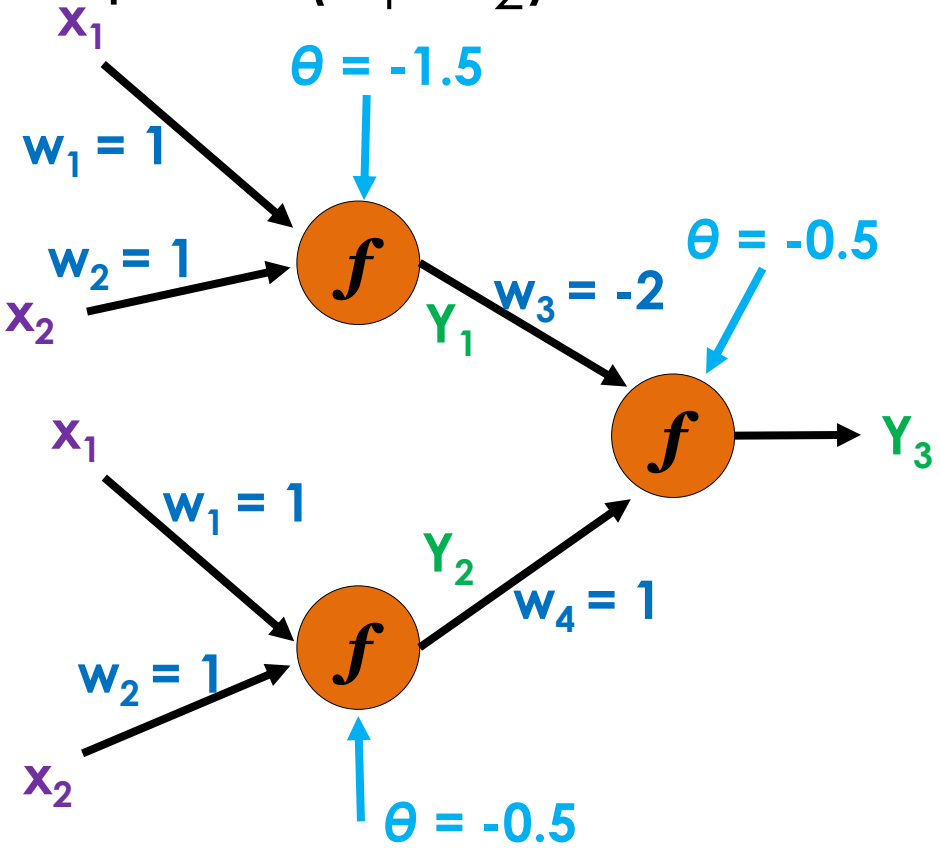


# Multilayer Neural Networks



# Example: Multilayer Neural Network

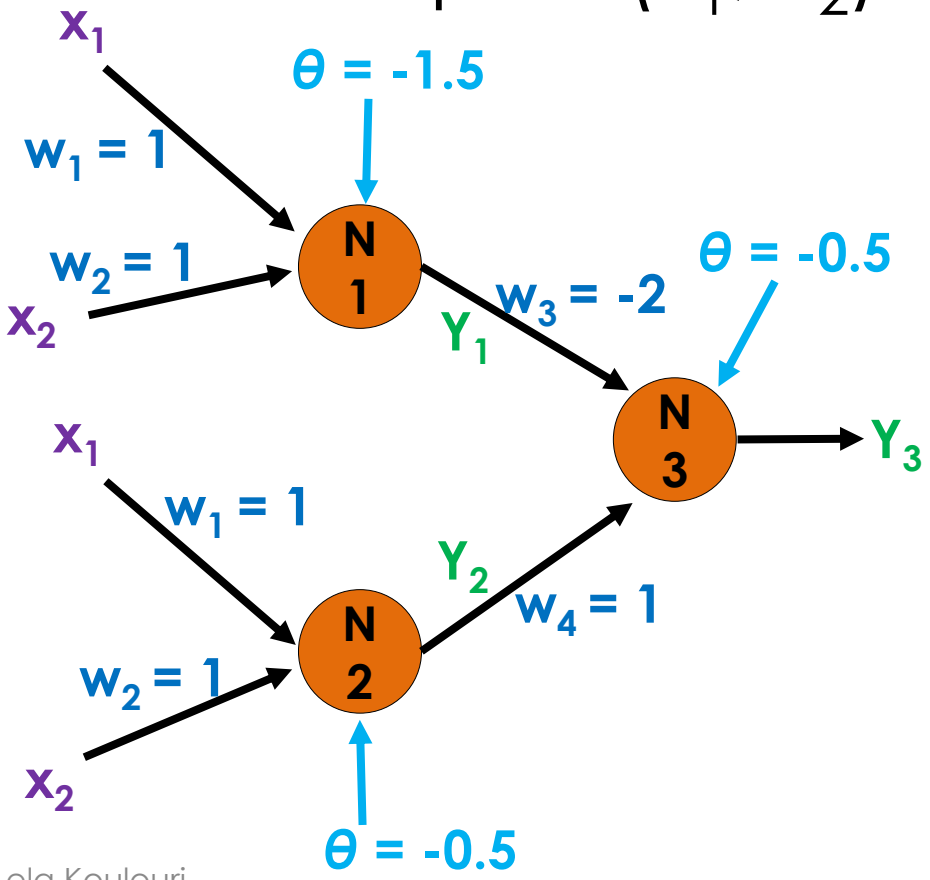
- This MNN can model XOR
- Use inputs ($x_1$, $x_2$) 0 and 1 and check.




| Input variables | | Exclusive-OR |
|---|---|---|
| $x_1$ | $x_2$ | $x_1 \oplus x_2$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Example: Multilayer Neural Network

- This MNN can model XOR
- Use inputs ($x_1$, $x_2$) 0 and 1 and check.



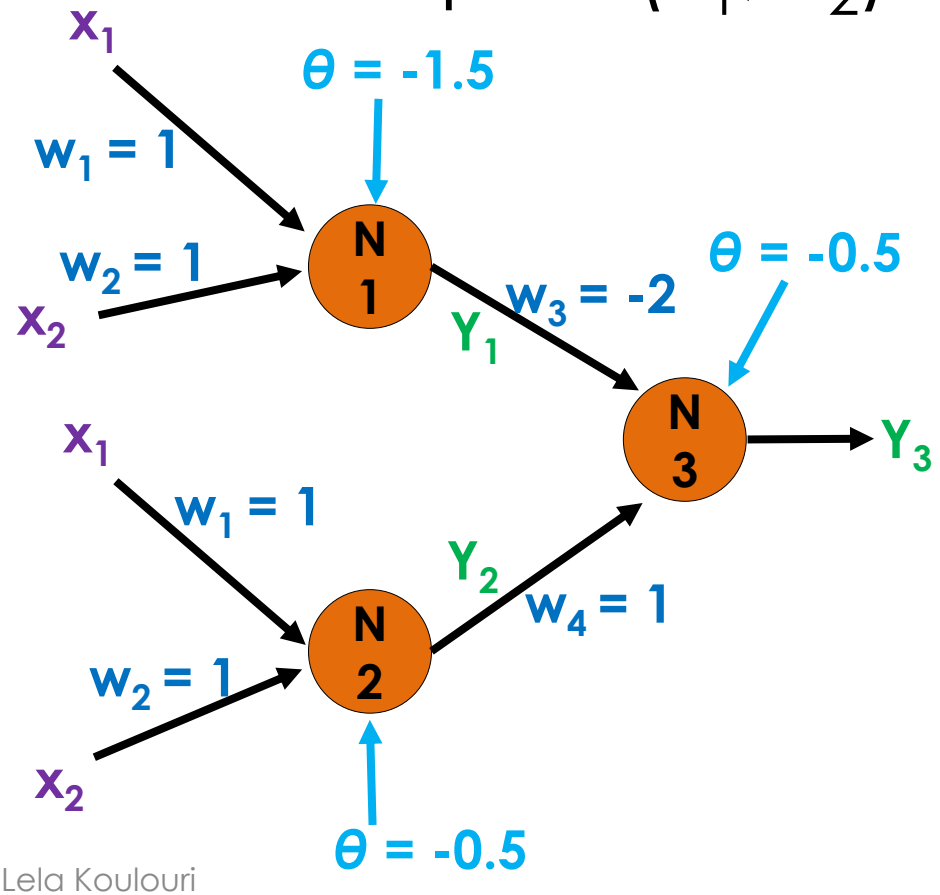

**Input (1,1)**

- Calculate output for Node 1:
- Calculate output for Node 2:
- Calculate output, $Y_3$, for Node 3:

$$Y = \begin{cases} +1, if \sum_i w_i x_i + \theta \geq 0 \\ 0, if \sum_i w_i x_i + \theta < 0 \end{cases}$$

# Example: Multilayer Neural Network

- This MNN can model XOR
- Use inputs ($x_1$, $x_2$) 0 and 1 and check.




**Input (1,1)**

Node 1: 1*1+1*1-1.5 = 0.5

(Step function) $y_1 = 1$

Node 2 : 1*1+1*1- 0.5 = 1.5

(Step function) $y_2 = 1$

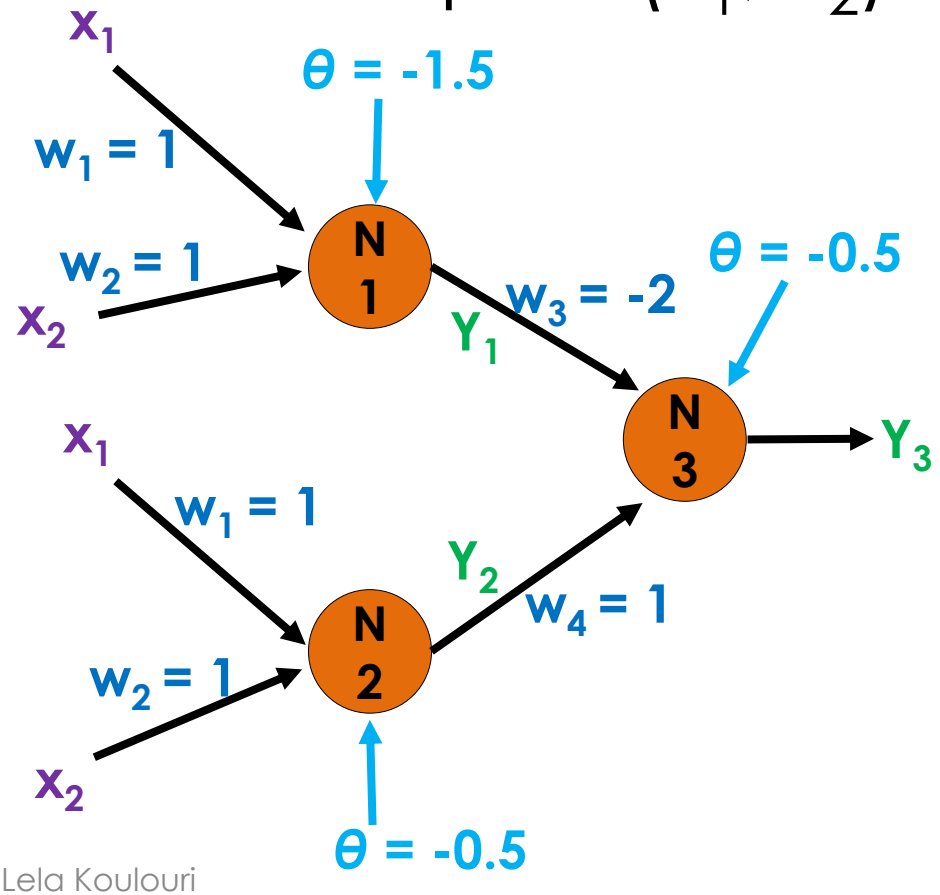
Node 3: $-2 * y_1 + 1 * y_2 - 0.5 =$

$-2 + 1 - 0.5 = -1.5$

(Step function) **$y_3 = 0$**

# Example: Multilayer Neural Network

- This MNN can model XOR
- Use inputs ($x_1$, $x_2$) 0 and 1 and check.




**Input (1,0)**

Node 1: 1*1+1*0-1.5 = -0.5

(Step function) $y_1 = 0$

Node 2 : 1*1+1*0 - 0.5 = 0.5

(Step function) $y_2 = 1$

Node 3: $-2 * y_1 + 1 * y_2 - 0.5 =$

$0 + 1 - 0.5 = 0.5$

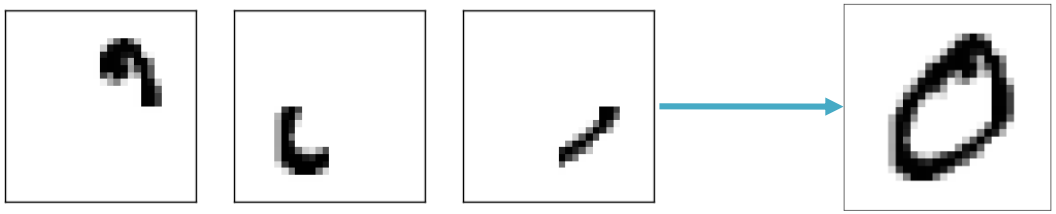
(Step function) **$y_3 = 1$**

# More complex MNNs

- Complex MNNs typically use functions, like the **Sigmoid Function**, rather than Step/Sign

$\sigma(x) = 1 / (1 + \exp(-x))$

  - Output not just 0 or 1, output can now be any continuous value between 0…1
    - E.g., in pattern recognition (recognition of hand-written digits), the output can represent the intensity of each pixel that combined together will indicate the digit.



# More complex NNs: Backpropagation

- Two stages in the backpropagation algorithm:
  - Forward propagation of the input pattern form input to output layer
  - Back propagation of the error from output to the input layer
- Learning rate and error derivative used to scale the weight adjustment
- Because of the multiple layers, the input to unit *j* may be outputs of a unit in previous layer $y_i$

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

$$\theta_j(p+1) = \theta_j(p) + \Delta\theta_j(p)$$

$$\Delta w_{ij}(p) = \alpha(errdrv)_j\, y_i$$

$$\Delta\theta_j(p) = \alpha(errdrv)_j$$

# Model Evaluation

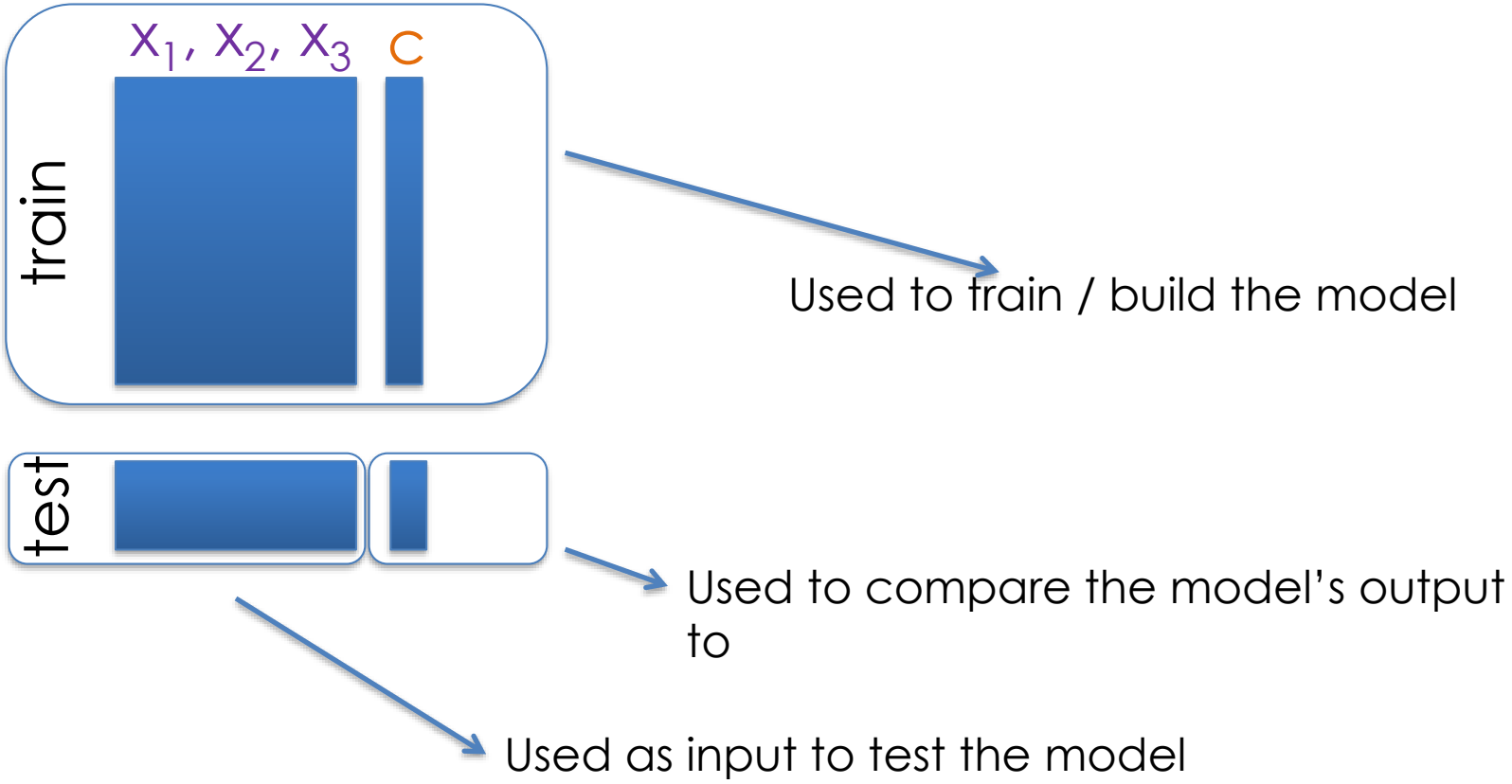
- How do we know that our classification model is 'good'?
- Need a way to score the 'predictive power' of the model
  - That is, how well can it classify new unseen data?

# Training and Test Sets

- Obviously a good idea to split data into a *training* set and a *test* set
  - Known as the *holdout* method
- Use training set to learn model
- Use test set to score accuracy
- Ideally the two sets are independent (generated separately)
- Many methods:
  - K-fold cross validation
  - Bootstrapping

# Training and Test Sets

# Training and Test Sets



# Model Evaluation

- We could use a simple error rate to assess our classifiers:

**Error rate = number of errors/number of test cases**

– For example:

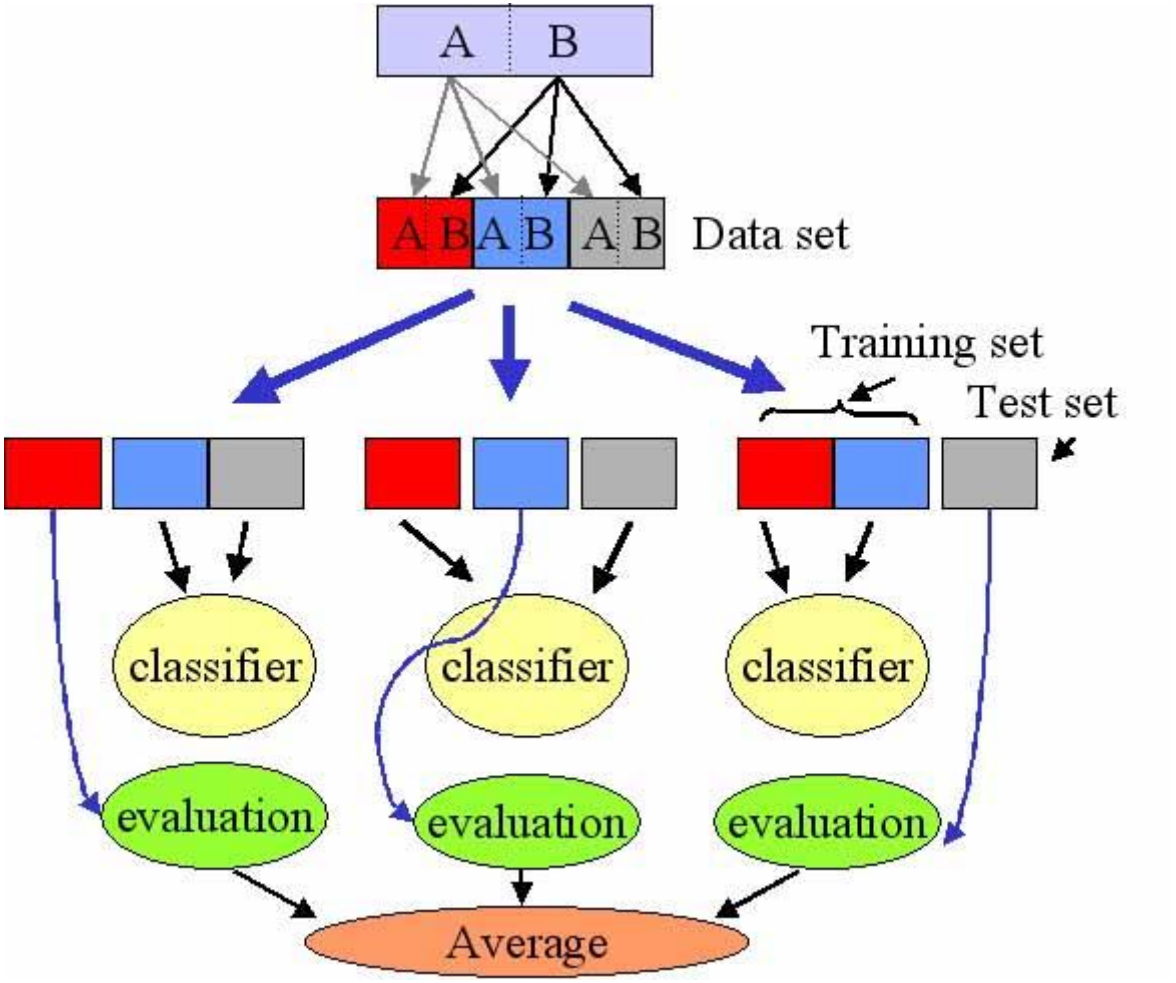
5 incorrect classifications out of 50 test records

5/50 = 0.1 → 10% error rate

# K-fold cross validation method

- *K*-fold cross validation algorithm:
  - Randomly split the data up into ***k*** subsets of equal size
  - Remove **one** of the subsets and train classifier on remaining subsets
  - Test on the removed subset
  - Get error rate
  - Repeat for all subsets (*k* times) so that each subset is used as test subset once.
  - The accuracy estimate for the classifier is the mean of each iteration.
- Procedure has been found to work well when sufficient data is available
- k = 10 has been found to be adequate and accurate

# Cross Validation



# Bootstrapping

- Given a dataset of size ***n***, a sample is randomly selected ***n*** times and used to train the model
  - So, a sample may be selected more than once
- Cases that were not used for training are used for the testing
- Usually 2/3 of data end up being used for training, and 1/3 for testing
- Procedure repeated and average of error rates used to indicate accuracy
- Many iterations needed if small dataset

# Evaluation: Confusion matrix

- But errors are of differing importance
  - E.g. failing to diagnose a disease can be more serious than diagnosing one that is not present
- Such evaluation is based on the count of test records that were correctly/incorrectly classified.
- In table format, it is called the **Confusion Matrix**
  - Actual Class vs Predicted Class
  - Binary classification models only

# Confusion matrix

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | TP | FP |
| | Class=No | FN | TN |

- *IS IT SPAM?*

**True** positive: Email is **spam** and classifier correctly identifies it as **spam**

**False** positive: Email is **not spam**, but classifier incorrectly identified it as **spam**

**True** negative: Email is **not spam** and classifier correctly identified it as **not spam**

**False** negative: Email is **spam**, but classifier incorrectly identified it as **not spam**

# Confusion matrix

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | TP | FP |
| | Class=No | FN | TN |

- **Sensitivity:** TP / (TP + FN)
- **Specificity:** TN / (FP + TN)
- **Accuracy: (TP + TN) / (TP + FN + FP + TN)**

# Sensitivity and Specificity

- For example, a classifier can have high **sensitivity** if it correctly classifies all sick people as having an illness (many TPs)
- ... but low **specificity** if it also classifies lots of healthy people as having an illness (many FPs).

- The trade-off is plotted as a ROC curve

# Classifier 1

- Calculate the sensitivity and specificity of the following classifiers.

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 10 | 3 |
| | Class=No | 15 | 22 |

- Sensitivity: TP / (TP + FN)=
- Specificity: TN / (FP + TN) =

# Classifier 1

- Calculate the sensitivity and specificity of the following classifiers.

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 10 | 3 |
| | Class=No | 15 | 22 |

- Sensitivity: TP / (TP + FN) = 10/25= 0.4
- Specificity: TN / (FP + TN) = 22/25 = 0.88

# Classifier 2

- Calculate the sensitivity and specificity of the following classifiers.

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 3 | 18 |
| | Class=No | 22 | 7 |

- Sensitivity: TP / (TP + FN) =
- Specificity: TN / (FP + TN) =

# Classifier 2

- Calculate the sensitivity and specificity of the following classifiers.

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 3 | 18 |
| | Class=No | 22 | 7 |

- Sensitivity: TP / (TP + FN) = 3/25 = 0.12
- Specificity: TN / (FP + TN) = 7/25 = 0.28

# Classifier 3

- Calculate the sensitivity and specificity of the following classifiers.

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 24 | 12 |
| | Class=No | 1 | 13 |

- Sensitivity: TP / (TP + FN) =
- Specificity: TN / (FP + TN) =

# Classifier 3

- Calculate the sensitivity and specificity of the following classifiers.

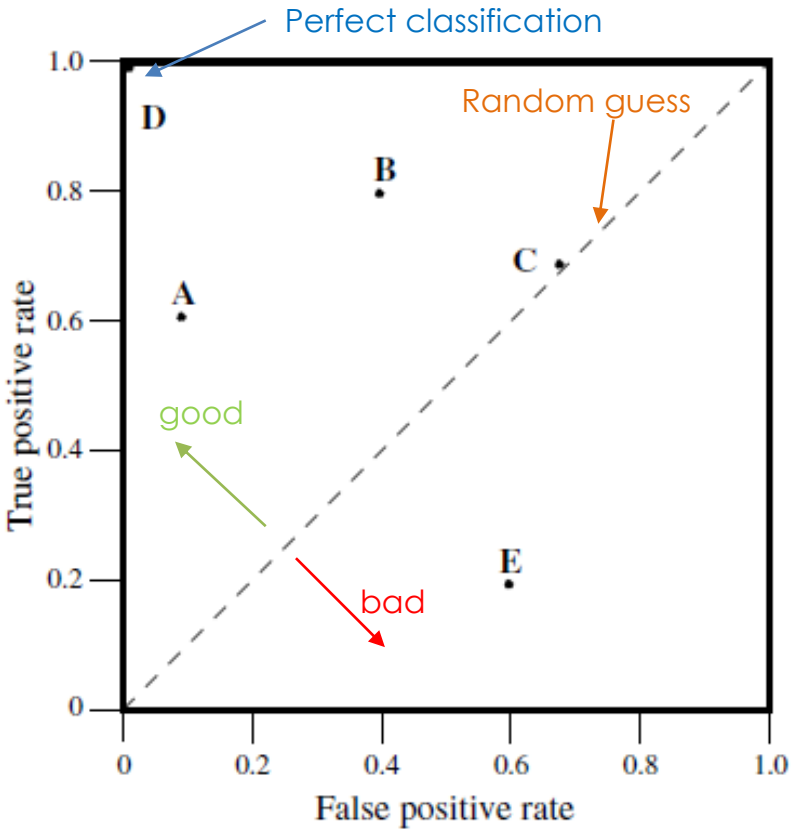
| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 24 | 12 |
| | Class=No | 1 | 13 |

- Sensitivity: TP / (TP + FN) = 24/25 = 0.96
- Specificity: TN / (FP + TN) = 13/25 = 0.52

# Receiver Operating Characteristics (ROC) graphs

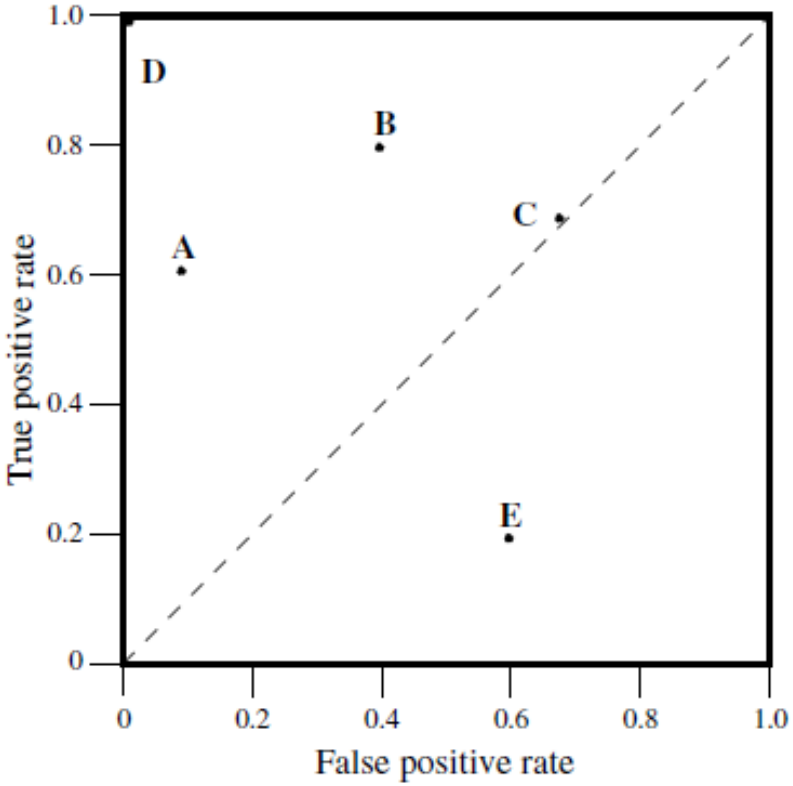
- Used to visualise the performance of classifiers
- Depict the trade-off between **true positives** (**Sensitivity**) and **false positives** (**1 – Specificity**).

- Comparing 5 classifiers (A, B, C, D, E)
- The diagonal line means random classification
- D's performance is perfect
- One classifier is better than another if it is closer to the upper left



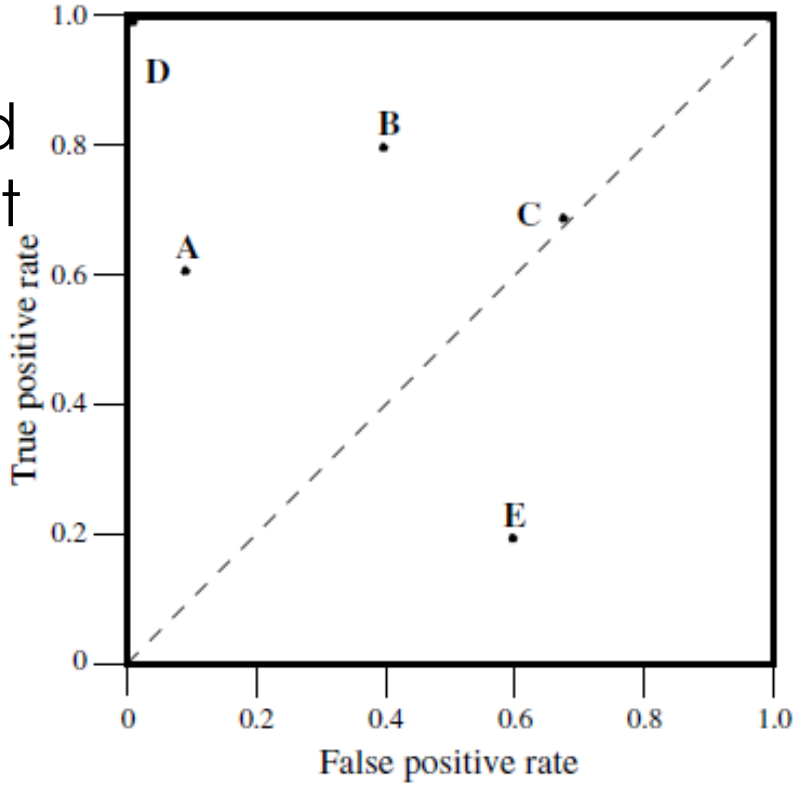
# Receiver Operating Characteristics (ROC) graphs

- A is more conservative than B
  - A has fewer true positives but also very few false positive errors, because it uses strict evidence
  - B makes more positive classifications but more false positive errors, because it uses weaker evidence
    - Most real-world domains have negative instances, so far LHS is more interesting
- Generally:
  - RHS: conservative classifiers
  - LHS: liberal classifiers



# Receiver Operating Characteristics (ROC) graphs

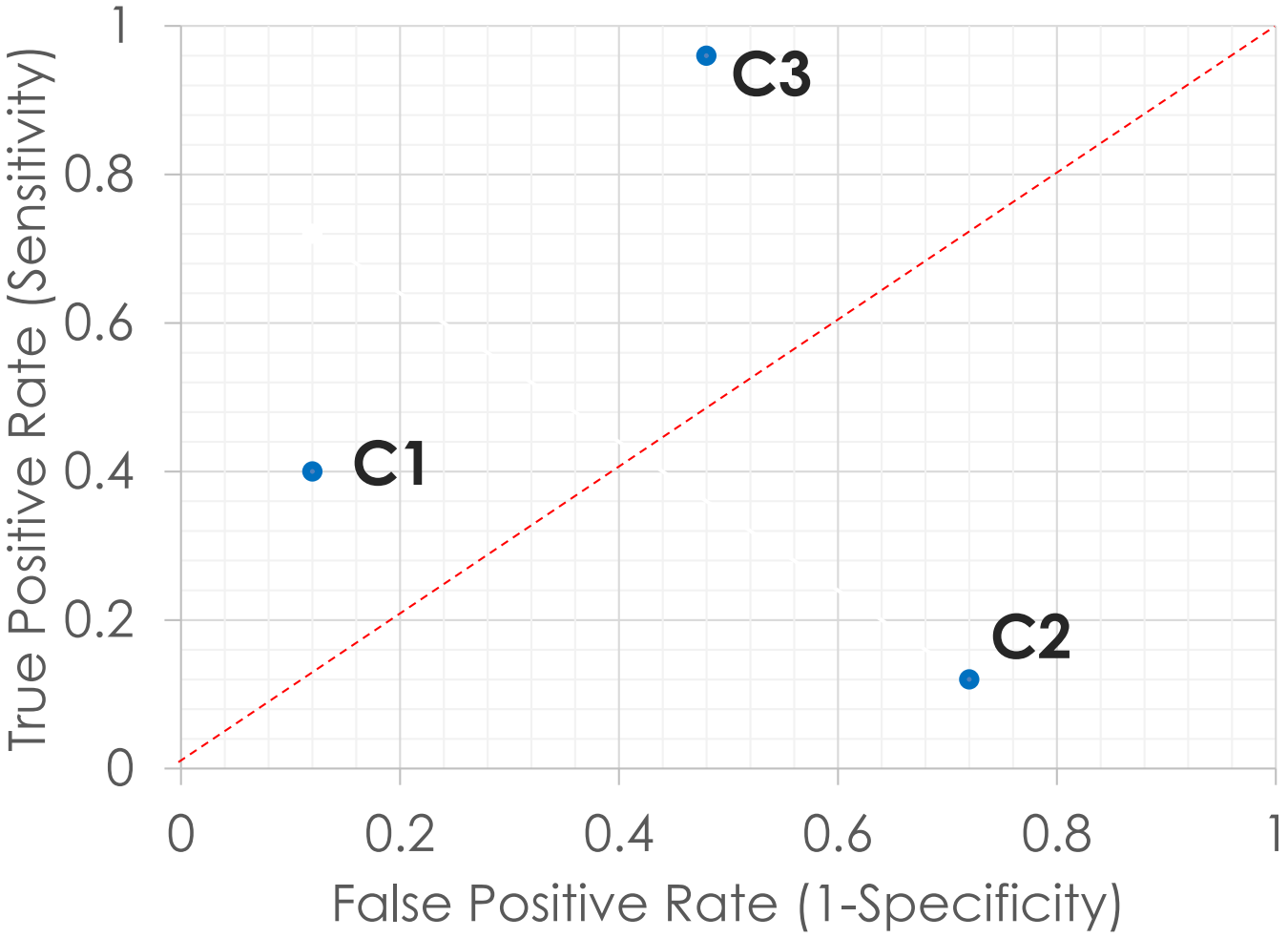
- Diagonal line represents random performance
- C's performance is random
- Any classifier below the diagonal can be negated and it will produce correct classifications (that is, reverse its decisions in every case).
- So ¬E is as good as B



# Exercise

- Show the performance of the 3 classifiers of the previous exercise using a ROC curve?
  - Classifier 1: Sensitivity: 0.4, Specificity: 0.88
  - Classifier 2: Sensitivity: 0.12, Specificity: 0.28
  - Classifier 3: Sensitivity: 0.96, Specificity: 0.52
- Which would be most suitable for detecting patients with high-risk of cancer and refer them for follow up tests?
- Which would be best for selecting patients to undergo high-risk surgery?
- Is Classifier 2 just useless?

# ROC graph


True Positive Rate (Sensitivity)
False Positive Rate (1-Specificity)
0
0.2
0.4
0.6
0.8
1
C1
C2
C3

# Exercise

- Show the performance of the 3 classifiers of the previous exercise using a ROC curve?
  - Classifier 1: Sensitivity: 0.4, Specificity: 0.88
  - Classifier 2: Sensitivity: 0.12, Specificity: 0.28
  - Classifier 3: Sensitivity: 0.96, Specificity: 0.52
- Which would be most suitable for detecting patients with high-risk of cancer and refer them for follow up tests?
  - C3, because high TP rate
- Which would be best for selecting patients to undergo high-risk surgery?
  - C1, because low FP rate (low probability that healthy people are incorrectly being sent to undergo high-risk surgery)
- Is Classifier 2 just useless?

# Classifier 2

Classifier 2's decisions:

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 3 | 18 |
| | Class=No | 22 | 7 |

- Sensitivity: TP / (TP + FN) = 3/25 = 0.12
- Specificity: TN / (FP + TN) = 7/25 = 0.28

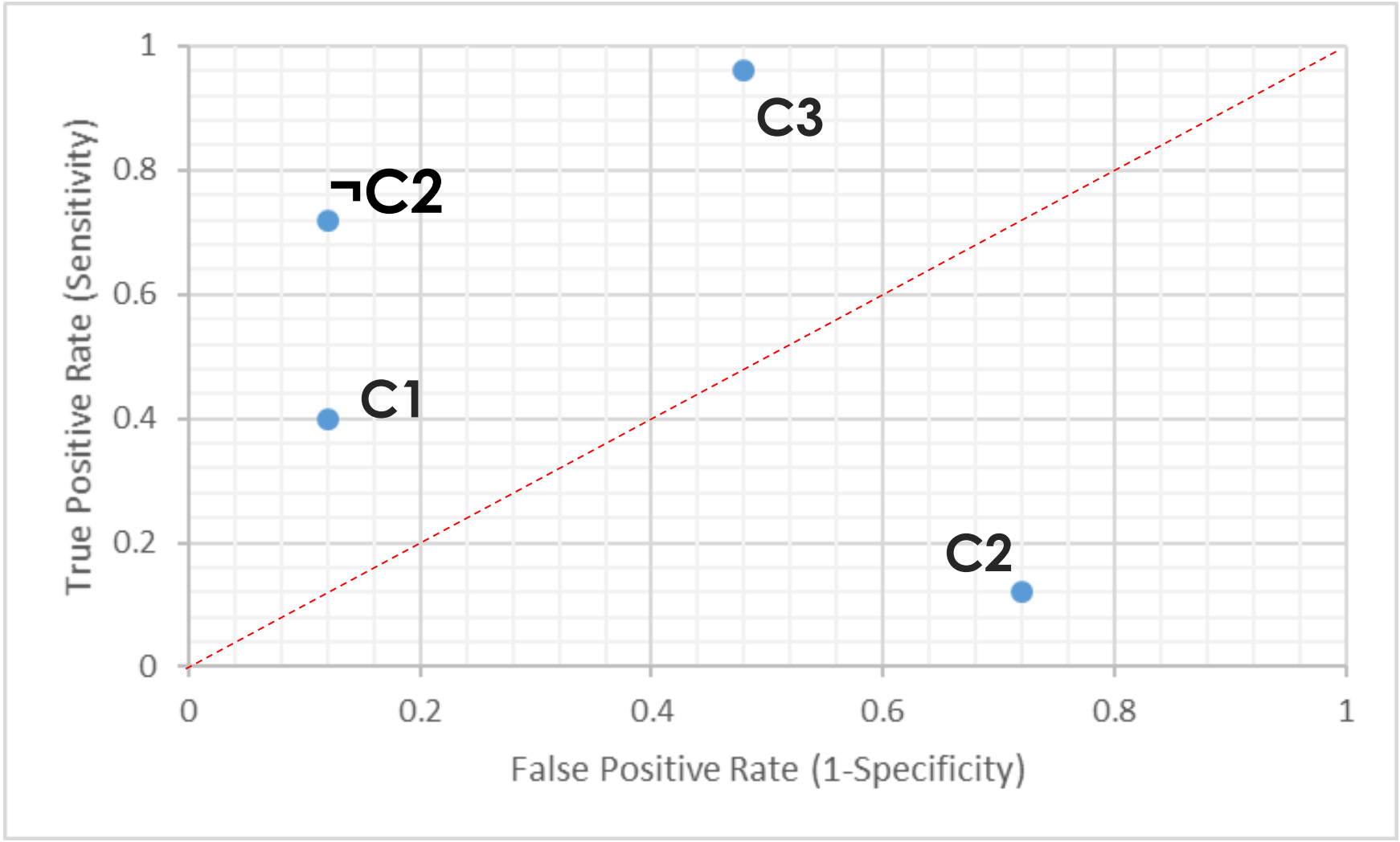
# ¬Classifier 2

- Simply reverse C2's decisions, and we have the best performing classifier!

| | ACTUAL CLASS | | |
|---|---|---|---|
| PREDICTED CLASS | | Class=Yes | Class=No |
| | Class=Yes | 18 | 3 |
| | Class=No | 7 | 22 |

- Sensitivity: TP / (TP + FN) = 18/25 = 0.77
- Specificity: TN / (FP + TN) = 22/25 = 0.8

# ROC Graph

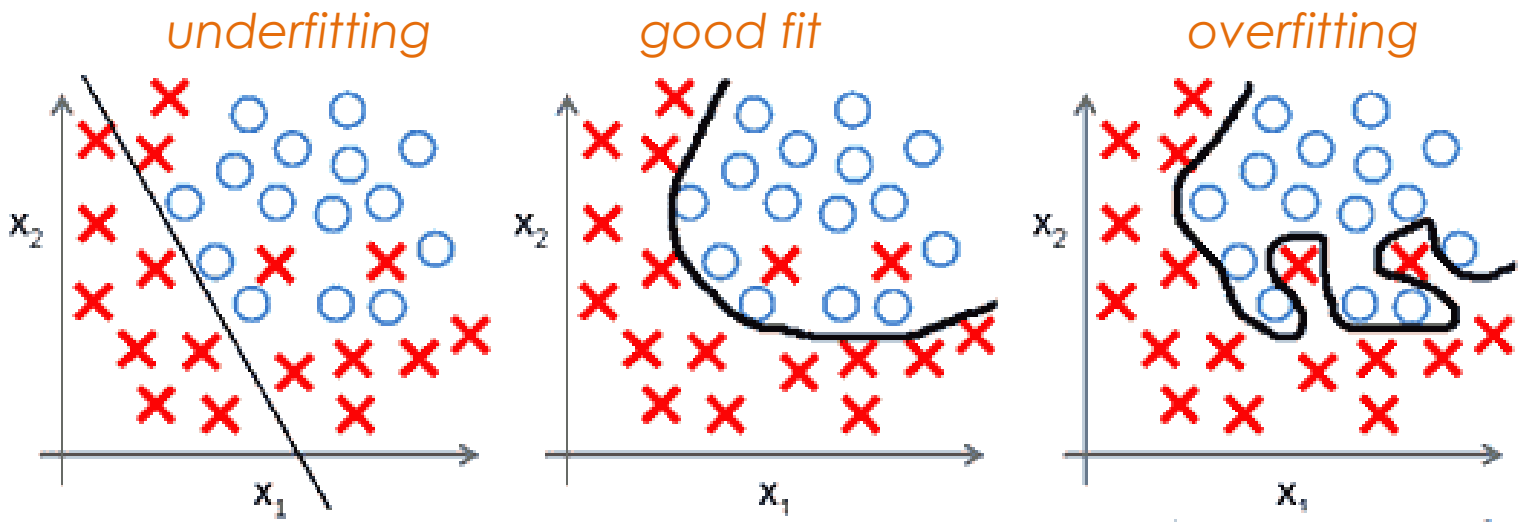

1
0.8
0.6
0.4
0.2
0
True Positive Rate (Sensitivity)
C3
¬C2
C1
C2
0
0.2
0.4
0.6
0.8
1
False Positive Rate (1-Specificity)

# Some other evaluation criteria

- Speed
  - Time or computation cost of constructing the model (e.g. a decision tree)
- Robustness
  - Data typically have errors so method should produce good results in spite of some errors or missing values
- Scalability
  - Algorithms should work for small and large datasets efficiently
- Interpretability
  - The end user should be able to understand and gain insight by the results produced by the method
- Goodness of the model
  - Does the method have a good fit of the data, or does it overfit/underfit?

# Underfitting and overfitting in models

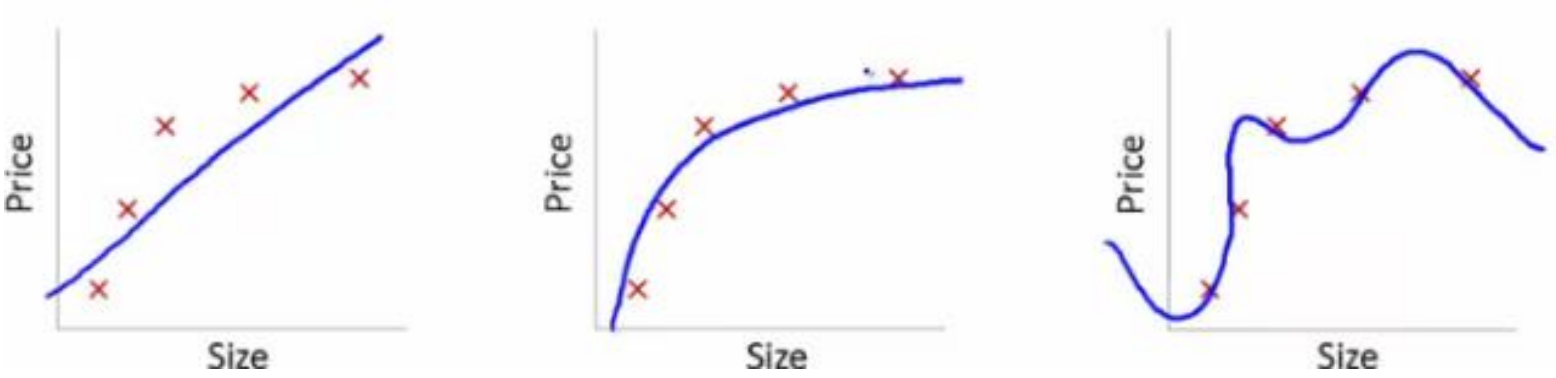
- *Underfitting* refers to a model that cannot model the training data well, and does not perform well on new data either
- *Overfitting* refers to a model that models the training data too well, but then cannot perform well on new data.



# Underfitting and overfitting in models

- *Underfitting* refers to a model that cannot model the training data well, and does not perform well on new data either
- *Overfitting* refers to a model that models the training data too well, and then cannot perform well on new data.

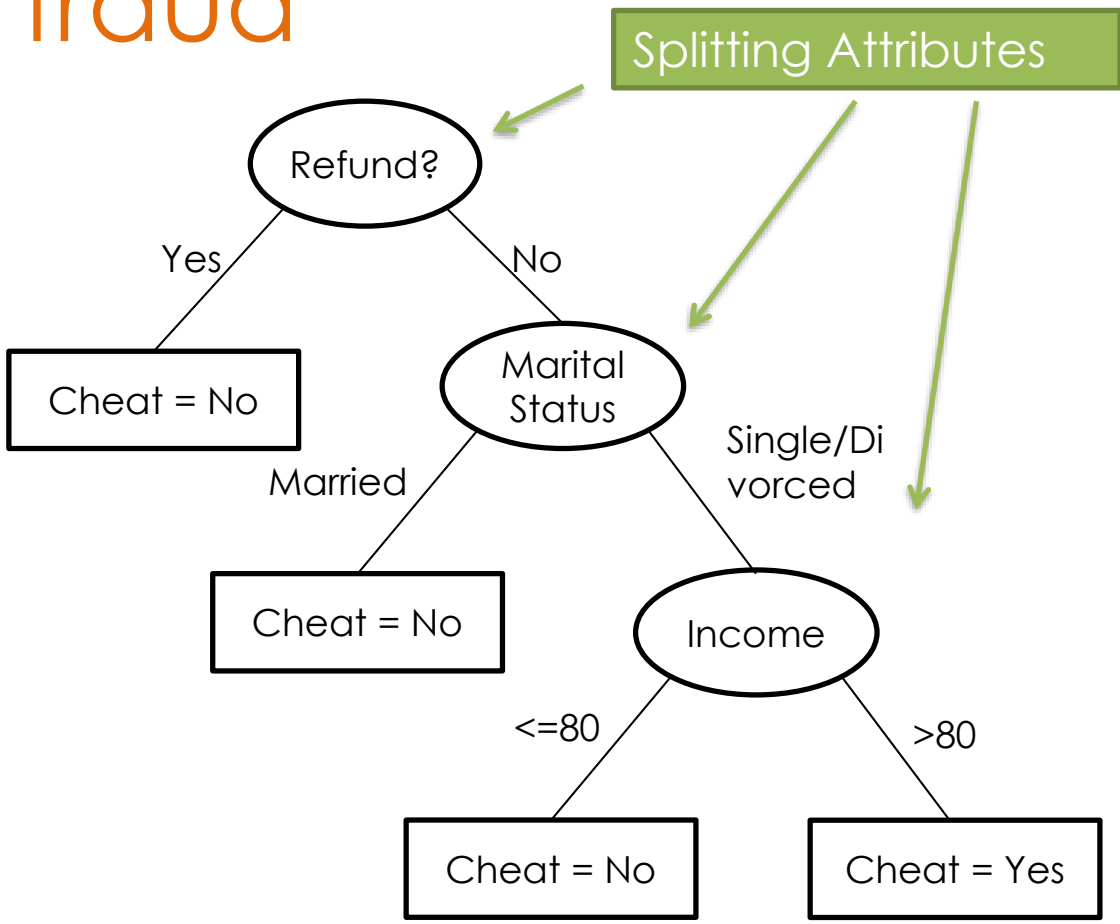
*underfitting* *good fit* *overfitting*



# Decision tree: example from tax fraud

categorical categorical continuous class

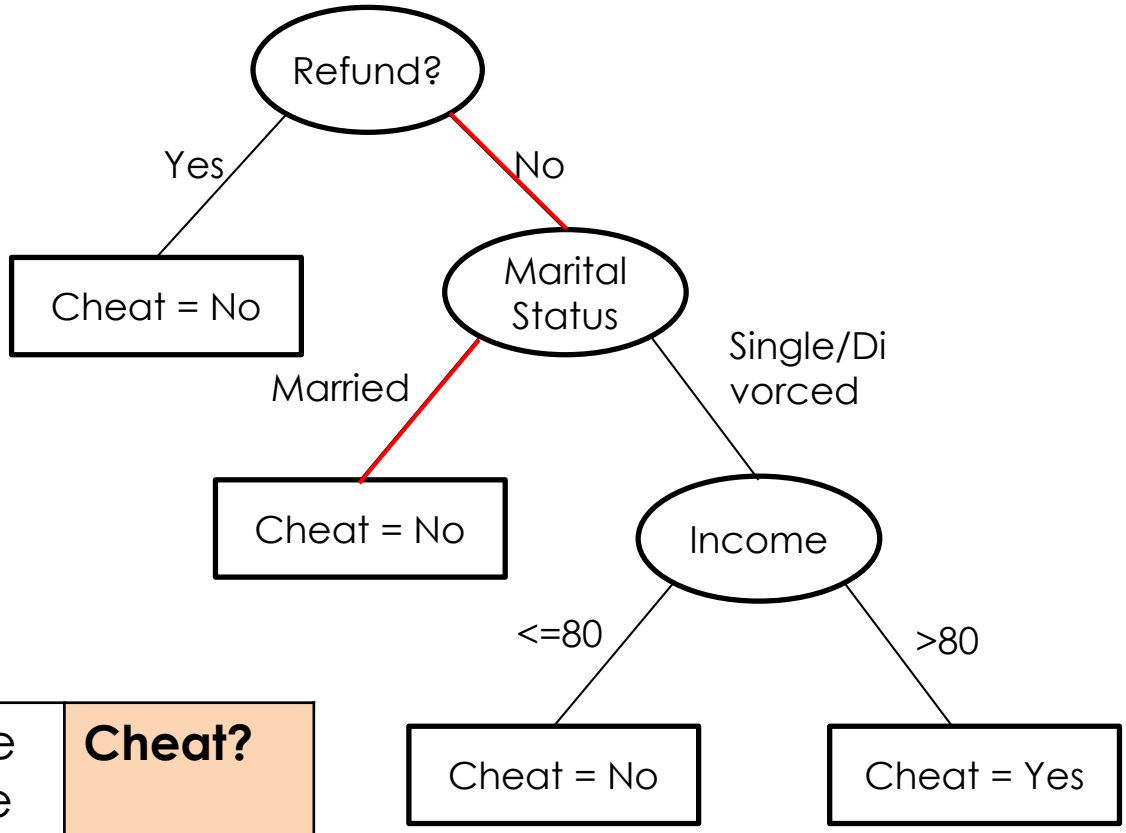
| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |




Training Data → Model: Decision Tree

# Decision tree

Model: Decision Tree




Test Data

| Refund | Marital Status | Taxable Income | **Cheat?** |
|---|---|---|---|
| No | Married | 80K | **?** |

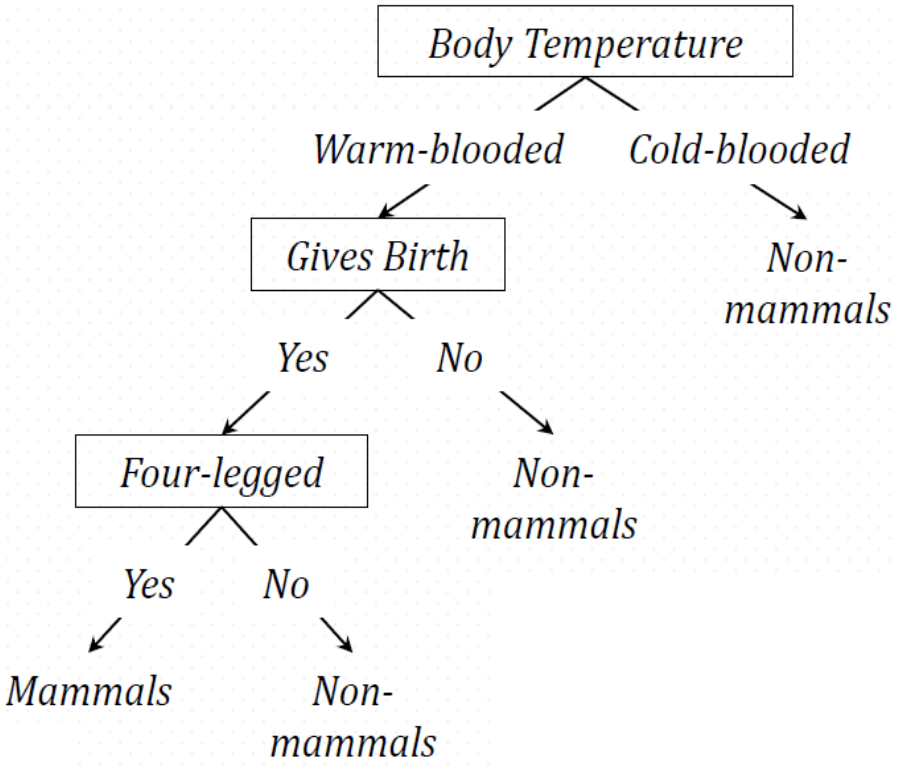
# Overfitting in decision trees

- The decision tree algorithm continues until either all nodes are terminal nodes or no more attributes are available.
- But when training objects have a large number of attributes, the tree will keep growing to fit the training data, but will not be able to generalise to new data

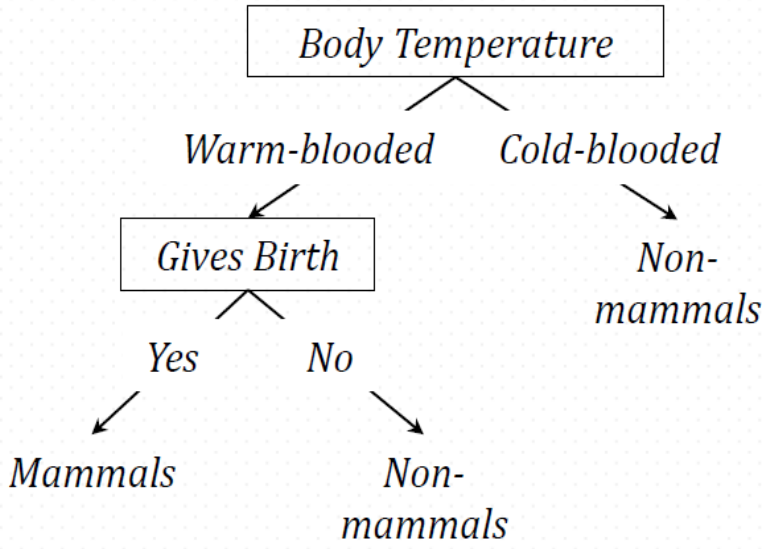
# Overfitting in decision trees

| Name | Body Temperature | Skin Cover | Gives Birth | Aquatic Creature | Aerial Creature | Has Legs | Hiber-nates | Class Label |
|---|---|---|---|---|---|---|---|---|
| human | warm-blooded | hair | yes | no | no | yes | no | mammal |
| python | cold-blooded | scales | no | no | no | no | yes | reptile |
| salmon | cold-blooded | scales | no | yes | no | no | no | fish |
| whale | warm-blooded | hair | yes | yes | no | no | no | mammal |
| frog | cold-blooded | none | no | semi | no | yes | yes | amphibian |
| komodo dragon | cold-blooded | scales | no | no | no | yes | no | reptile |
| bat | warm-blooded | hair | yes | no | yes | yes | yes | mammal |
| pigeon | warm-blooded | feathers | no | no | yes | yes | no | bird |
| cat | warm-blooded | fur | yes | no | no | yes | no | mammal |
| leopard shark | cold-blooded | scales | yes | yes | no | no | no | fish |
| turtle | cold-blooded | scales | no | semi | no | yes | no | reptile |
| penguin | warm-blooded | feathers | no | semi | no | yes | no | bird |
| porcupine | warm-blooded | quills | yes | no | no | yes | yes | mammal |
| eel | cold-blooded | scales | no | yes | no | no | no | fish |
| salamander | cold-blooded | none | no | semi | no | yes | yes | amphibian |

**Model 1**




**Model 2**




When to stop?

Is model 1 overfit?

Is model 2 too simple?

# Model selection criteria

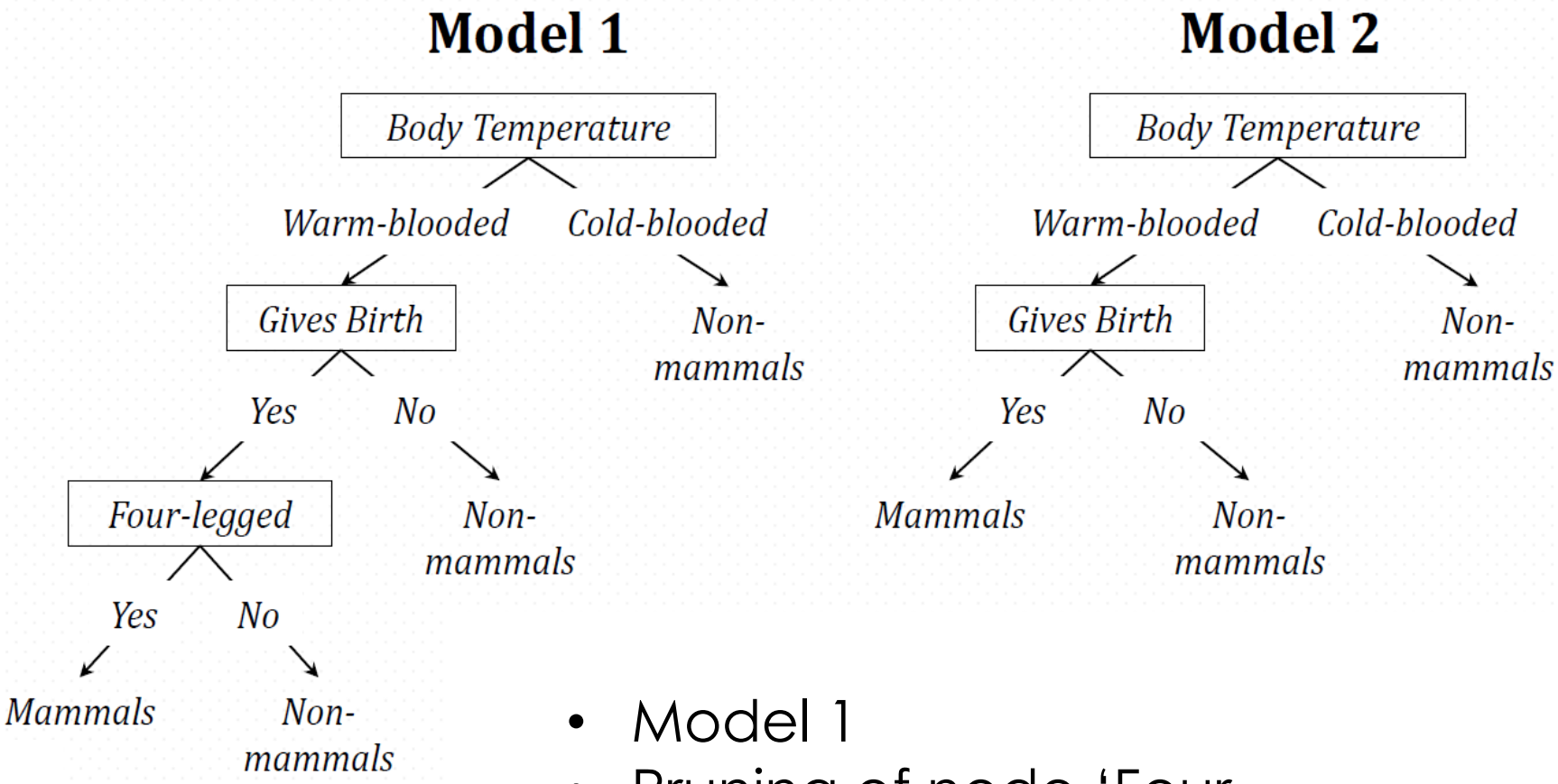
- Model selection criteria attempt to find a good compromise between:
  a)The model's complexity
  b)The model's prediction accuracy on unseen data
- Reasoning: a good model is a simple model that achieves high accuracy on the given data
- Also known as Occam's Razor: the best theory is the smallest one that describes all the facts

# Pruning to avoid overfitting

- We perform pruning (removing nodes) to avoiding overfitting in building decision trees. Two approaches:

- **Pre-pruning** that stop growing the tree earlier, before it perfectly classifies the training set.
- **Post-pruning** that allows the tree to perfectly classify the training set, and then post prune the tree.
  - Post-pruning is more successful because it is not easy to precisely estimate when to stop growing the tree.
- **Post-pruning** procedure:
  - The available data are separated into two sets of examples: a training set, which is used to build the decision tree, and a validation set, which is used to evaluate the impact of pruning the tree.

# Model selection



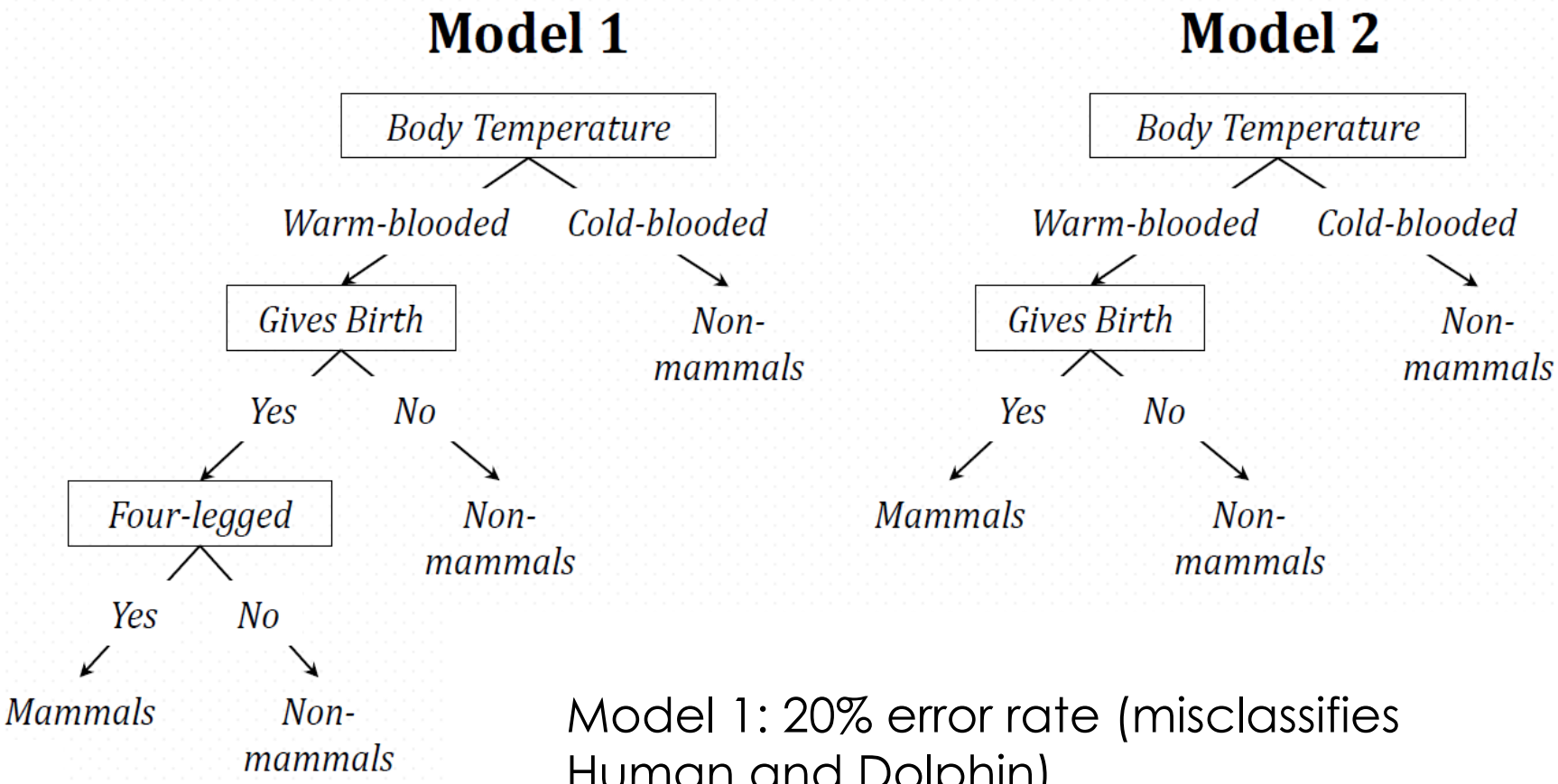

- Model 1
- Pruning of node 'Four-Legged' results in Model 2

# Evaluate the 2 models

- Test data:

| Name | Body Temperature | Gives Birth | Four-legged | Hibernates | Class Label |
|---|---|---|---|---|---|
| Human | Warm-blooded | Yes | No | No | ***Yes*** |
| Pigeon | Warm-blooded | No | No | No | ***No*** |
| Elephant | Warm-blooded | Yes | Yes | No | ***Yes*** |
| Leopard shark | Cold-blooded | Yes | No | No | ***No*** |
| Turtle | Cold-blooded | No | Yes | No | ***No*** |
| Penguin | Cold-blooded | No | No | No | ***No*** |
| Eel | Cold-blooded | No | No | No | ***No*** |
| Dolphin | Warm-blooded | Yes | No | No | ***Yes*** |
| Spiny anteater | Warm-blooded | No | Yes | Yes | ***Yes*** |
| Gila monster | Cold-blooded | No | Yes | Yes | ***No*** |

# Evaluate the 2 models



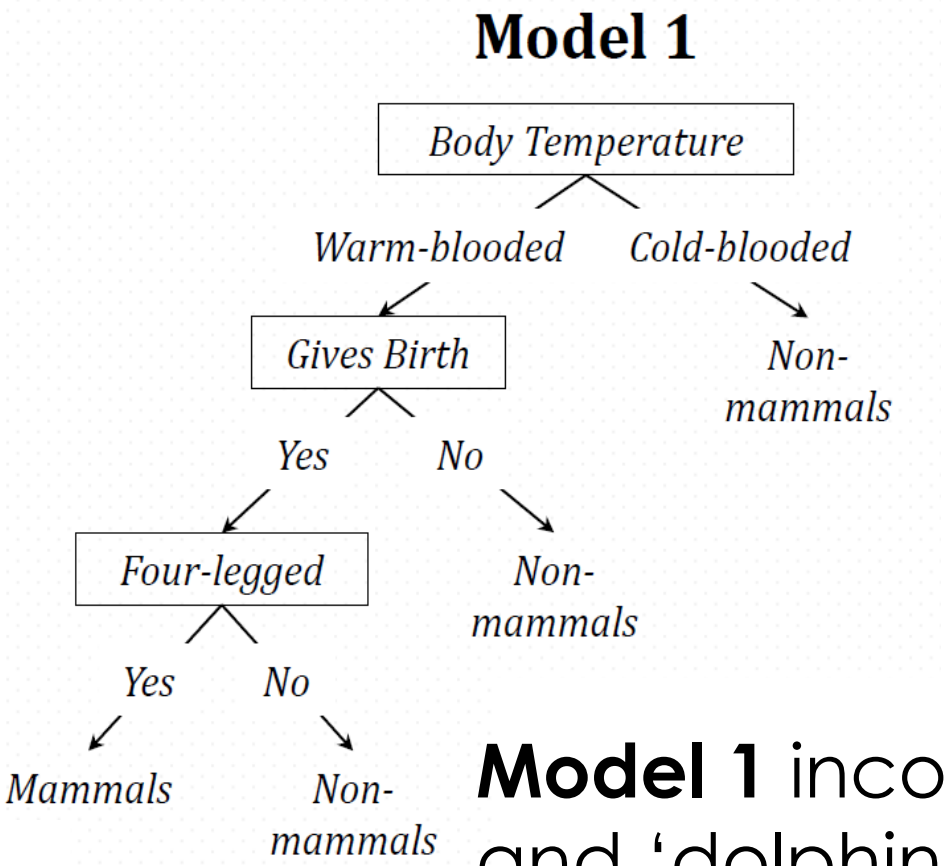

Model 1: 20% error rate (misclassifies Human and Dolphin)
Model 2: 10% error rate (misclassifies spiny anteater)
**So Model 2 is better**

# Some causes of overfitting

- Overfitting due to noise in the data
  - Example training data set for classifying mammals and non-mammals. The set contains mislabellings (denoted with a *)

| Name | Body Temperature | Gives Birth | Four-legged | Hibernates | Class Label |
|---|---|---|---|---|---|
| Porcupine | Warm-blooded | Yes | Yes | Yes | ***Yes*** |
| Cat | Warm-blooded | Yes | Yes | No | ***Yes*** |
| Bat | Warm-blooded | Yes | No | Yes | ***No**** |
| Whale | Warm-blooded | Yes | No | No | ***No**** |
| Salamander | Cold-blooded | No | Yes | Yes | ***No*** |
| Komodo dragon | Cold-blooded | No | Yes | No | ***No*** |
| Python | Cold-blooded | No | No | Yes | ***No*** |
| Salmon | Cold-blooded | No | No | No | ***No*** |
| Eagle | Warm-blooded | No | No | No | ***No*** |
| Guppy | Cold-blooded | Yes | No | No | ***No*** |

# Produces Model 1



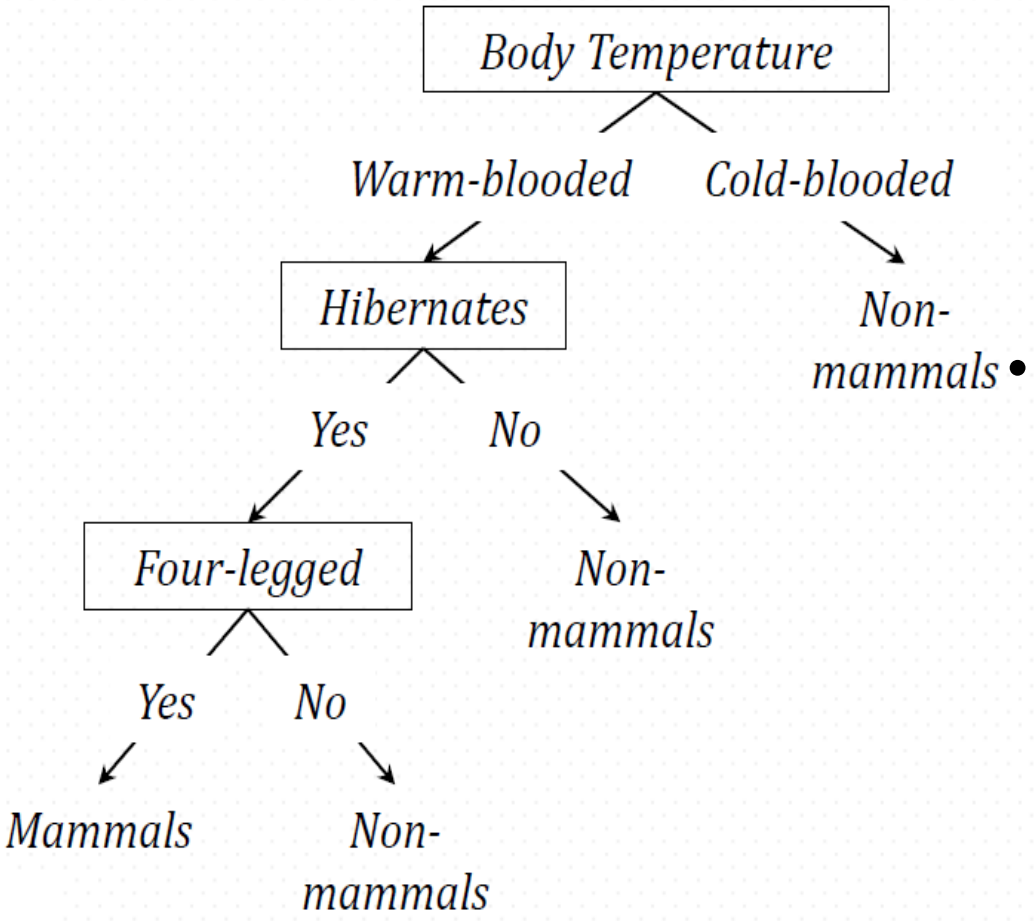

**Model 1** incorrectly classifies 'human' and 'dolphin'. **20%** error rate (as previously seen)

# Some causes of overfitting

- Overfitting due to small training data set
  - Data set with only 5 training examples

| Name | Body Temperature | Gives Birth | Four-legged | Hibernates | Class Label |
|---|---|---|---|---|---|
| Salamander | Cold-blooded | No | Yes | Yes | ***No*** |
| Guppy | Cold-blooded | Yes | No | No | ***No*** |
| Eagle | Warm-blooded | No | No | No | ***No*** |
| Poorwill | Warm-blooded | No | No | Yes | ***No*** |
| Platypus | Warm-blooded | No | Yes | Yes | ***Yes*** |

# Produces the following model




- **This model** incorrectly classifies 'human', 'elephant' and 'dolphin'.
  … 30% error rate.
- The tree arrives at this classification because there was only one training record (eagle) that is warm-blooded and does not hibernate, classified as non-mammal.

# Evaluation of **clustering** algorithms

- Calculate number of correct 'classifications' of objects to clusters
  - Can use confusion matrix
  - 'Is object X classified to cluster A?'
- Calculate ratio of intra-cluster and inter-cluster distances
  - The tighter and more well-separated the clusters are, the better the performance
    - Calculate average sum of squared distance of each object in a cluster from cluster centroid (I)
    - Calculate average sum of squared pairwise distances between the clusters (E)
    - E/I

# Software implementations of ML algorithms

- R *
- Weka *
- ScipY and scikit-learn python libraries *
- Orange *
- MATLAB
- SPSS
- SAS
- STATA
- SQL Server Analysis Services

* Free software/OS licence

# Exam 2016

- Which of the following best describes reinforcement learning?
  a) Learning without any pre-defined 'correct' answers, e.g. Clustering.
  b) A non-parametric approach to learning from previous examples, e.g. K-Nearest Neighbour.
  c) Learning through the use of repeated punishment and reward 'signals'
  d) Learning to accurately predict the class label of previously unseen records.

# Exam 2016

- The K-Means algorithm terminates when
  a) a user-defined minimum value for the summation of squared error differences between instances and their corresponding cluster centre is seen.
  b) the cluster centres for the current iteration are identical to the cluster centres for the previous iteration.
  c) the number of instances in each cluster for the current iteration is identical to the number of instances in each cluster of the previous iteration.
  d) the number of clusters formed for the current iteration is identical to the number of clusters formed in the previous iteration.

# Exam 2016

- Which one of the following is not true about clustering?
  a) The clusters are suggested by the data, and are not defined beforehand.
  b) The aim of clustering is often exploratory.
  c) Objects in each cluster tend to be similar to each other and dissimilar to objects in the other clusters.
  d) Training data must be available.

# Sample question

- What are hidden nodes?
  a) Nodes that do not do any computation and so do not assist in producing the output
  b) Nodes that have '0' as an input value
  c) Nodes that have no direct connection to the input or the output
  d) Nodes that have no direct connections to any other nodes

# Sample question

- The values input into a feed-forward neural network
  a) may be categorical or numeric.
  b) must be either all categorical or all numeric but not both.
  c) must be numeric.
  d) must be categorical.

# Sample question

- Which of the following activities is more likely to be a regression problem?
  a) Dividing the customers of a company according to their profitability
  b) Predicting the future stock price of a company using historical records.
  c) Discriminating high-risk and low-risk patients
  d) Recognising hand-written alphanumeric characters.

# Sample question

- Which algorithm starts with as many clusters as there are records, with each cluster having only one record?

a) Hierarchical agglomerative clustering.
b) KNN
c) K-means
d) Decision tree

# Sample question

- Which of the following two-input logic operation is linearly inseparable?

(i) AND (ii) OR (iii) XOR (iv) NAND (v) NOT XOR

a) (ii) and (iii)

b) (iii) only

c) (iii) and (v)

d) (i), (ii), and (iv)

# Additional resources

**Further Reading:**

- Model Evaluation & Overfitting:
  - Tan et al. 'Introduction to Data Mining'
  - http://www-users.cs.umn.edu/~kumar/dmbook/index.php
  - Chapter 4
- Multi-layer Networks:
  - Russell and Norvig textbook, section 19.4

**Watching:**

- Andrew Ng's Stanford Machine Learning lectures.

Brilliant module – it covers everything in ML!