Lancaster University

AI SCC361 Week 7

Dr. Hossein Rahmani
Senior Lecturer in Data Science
D17, InfoLab; e-mail: h.rahmani@lancaster.ac.uk

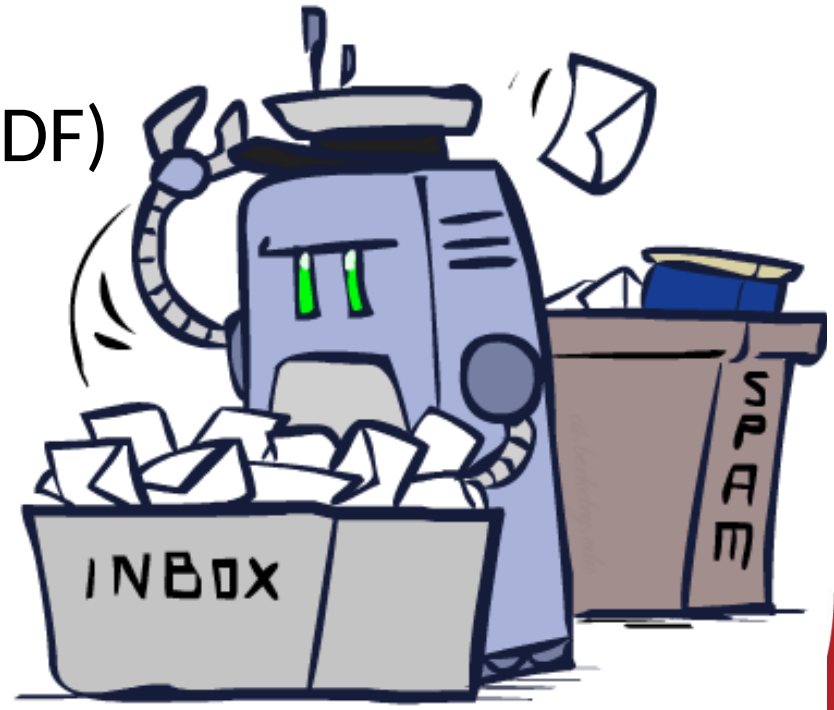
Decision Trees

Random Decision Forests

Outline

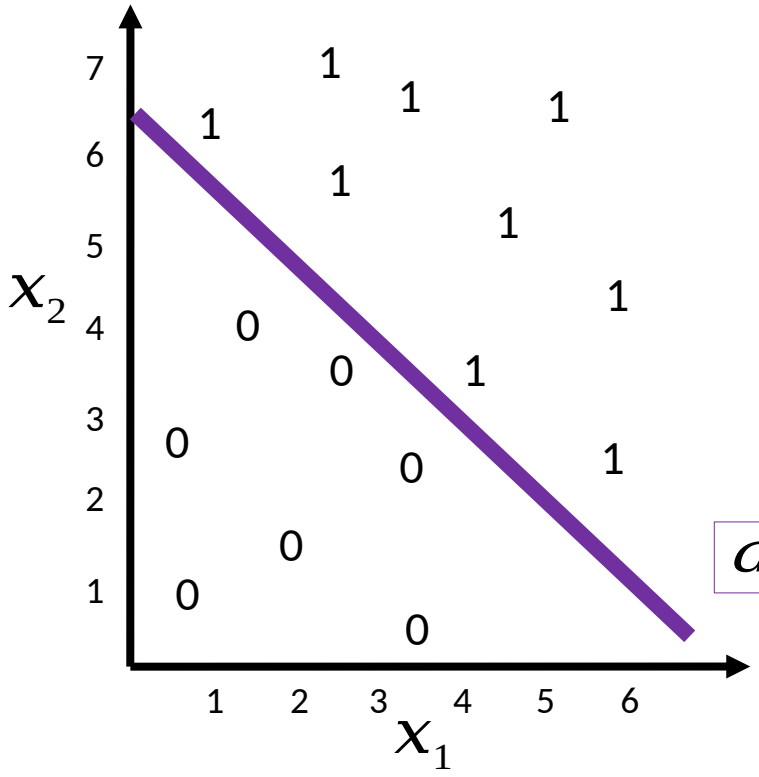
Lancaster University

- Decision Tree (DT)
- Overfitting in DT
- Random Decision Forrest (RDF)



Linearly Separable Data

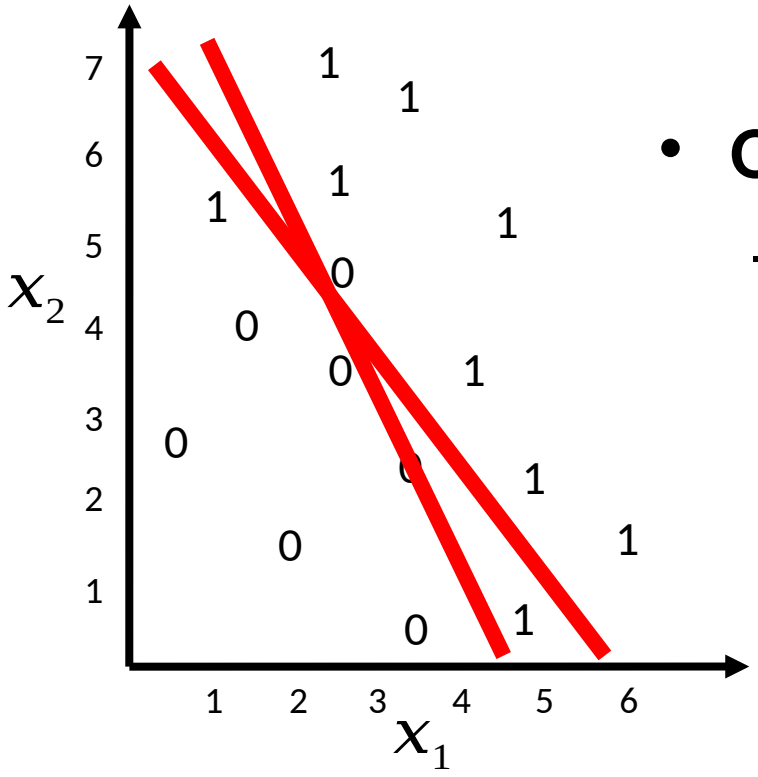
- Continuous-valued features: and
- Binary classification output: **0** or **1**



$$a_1 x_1 + a_2 x_2 + a_3 = 0$$

NOT Linearly Separable Data

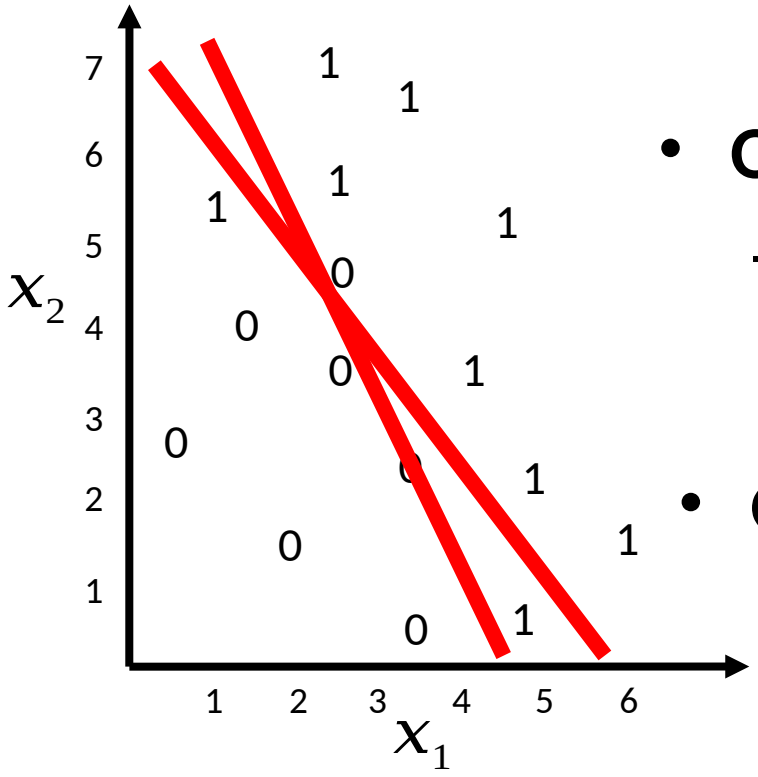
- Continuous-valued features: and
- Binary classification output: **0** or **1**



- **Question:**
  - Is this data **Linearly** separable?

NOT Linearly Separable Data

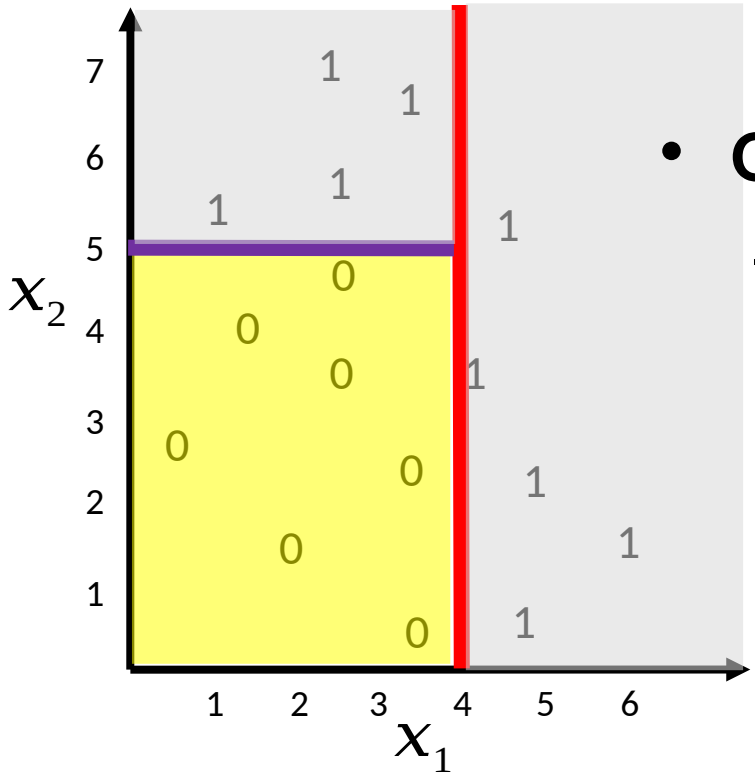
- Continuous-valued features: and
- Binary classification output: **0** or **1**



- **Question:**
  - Is this data **Linearly** separable?
    - NO, as there is no line which can separates samples
- **Question:**
  - How to resolve this issue?

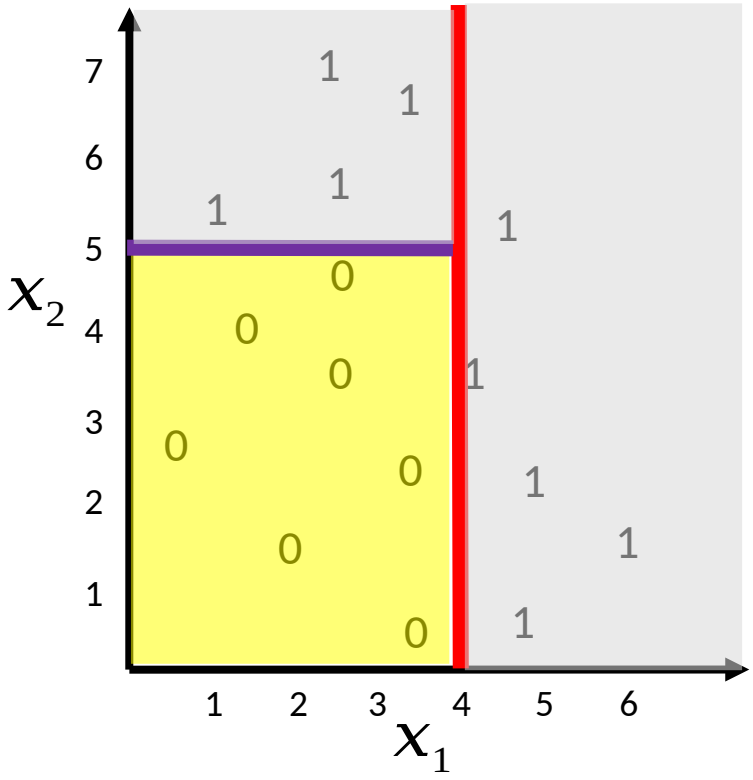
Decision Tree Divides the Feature Space

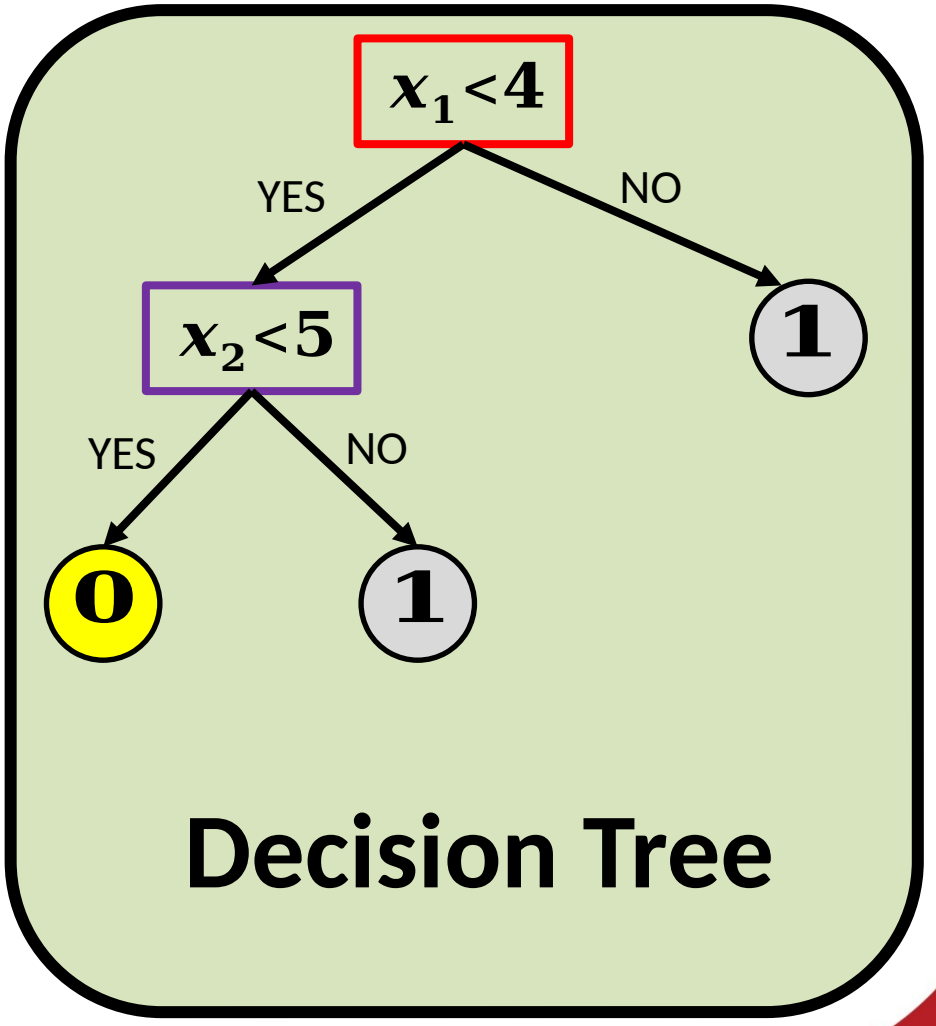
- Continuous-valued features: and
- Binary classification output: **0** or **1**



- **Question:**
  - How do we branch using feature values and to partition the space correctly?

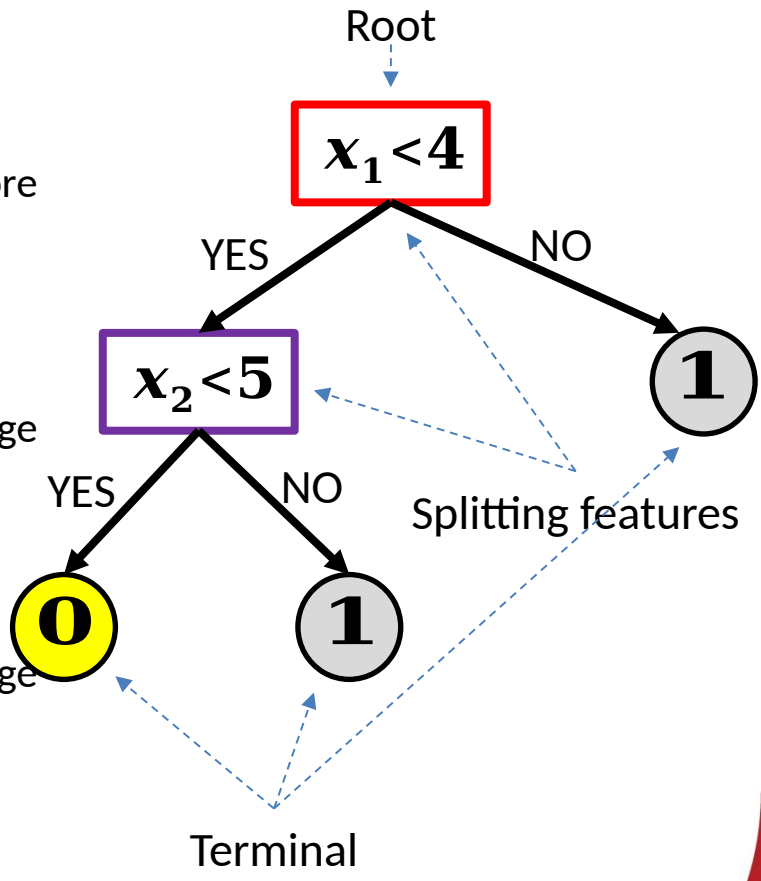
Decision Tree divides the space





Structure of Decision Trees

- A decision tree consists of:
  - A root node:
    - It has no incoming edges and zero or more outgoing edges
    - Test for the value of a certain feature
  - Internal nodes:
    - Each of which has exactly one incoming edge and two or more outgoing edges
    - Test for the value of a certain feature
  - Leaf or Terminal nodes:
    - Each of which has exactly one incoming edge and no outgoing edges
    - Predict the outcome



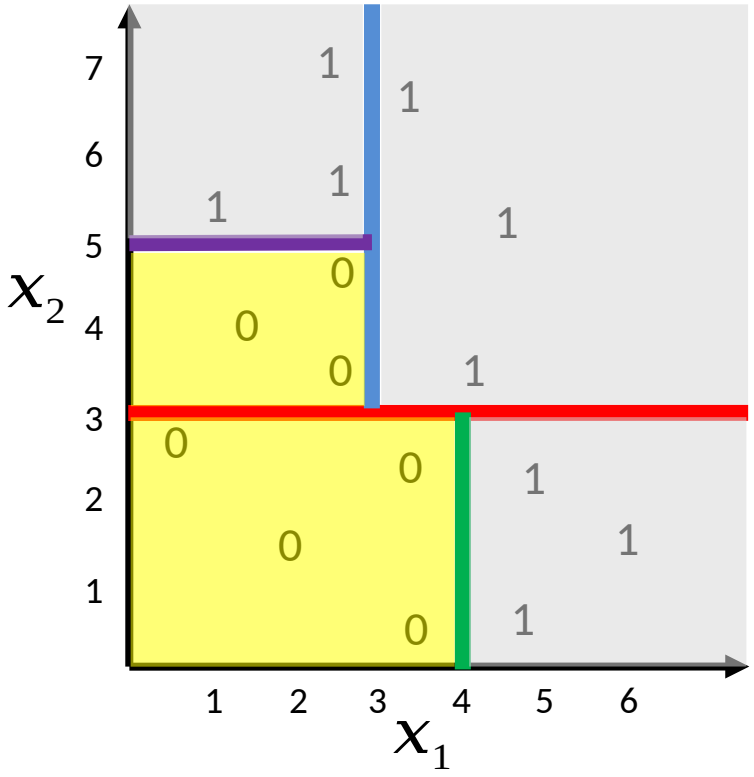
Decision Tree (Definition)

Lancaster University

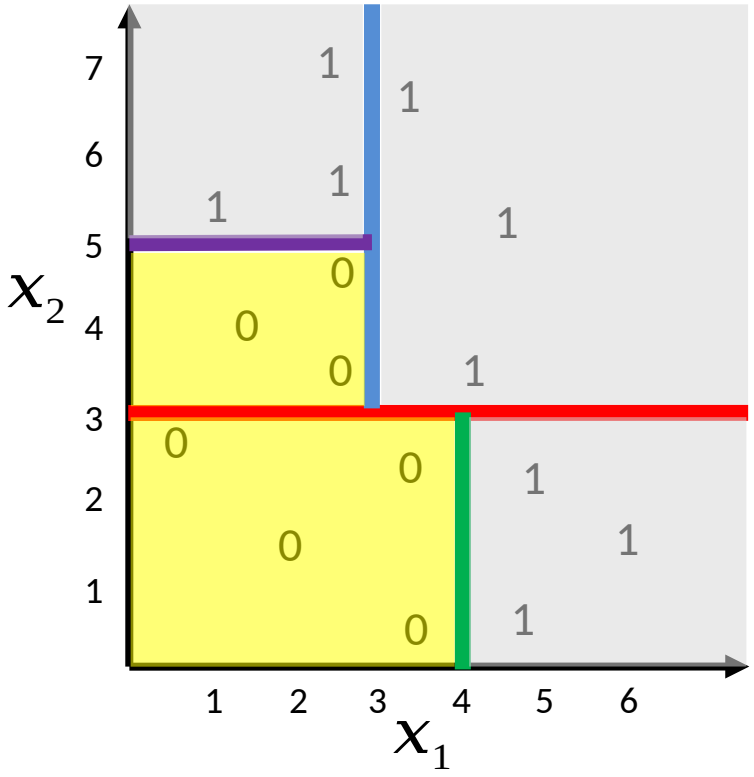
- The decision tree classifiers organized a series of test questions and conditions in *a tree structure*.
- In the decision tree, the root and internal nodes contain *feature test conditions* to separate samples that have different characteristics.
- All the *leaf node* is assigned *a class label*.

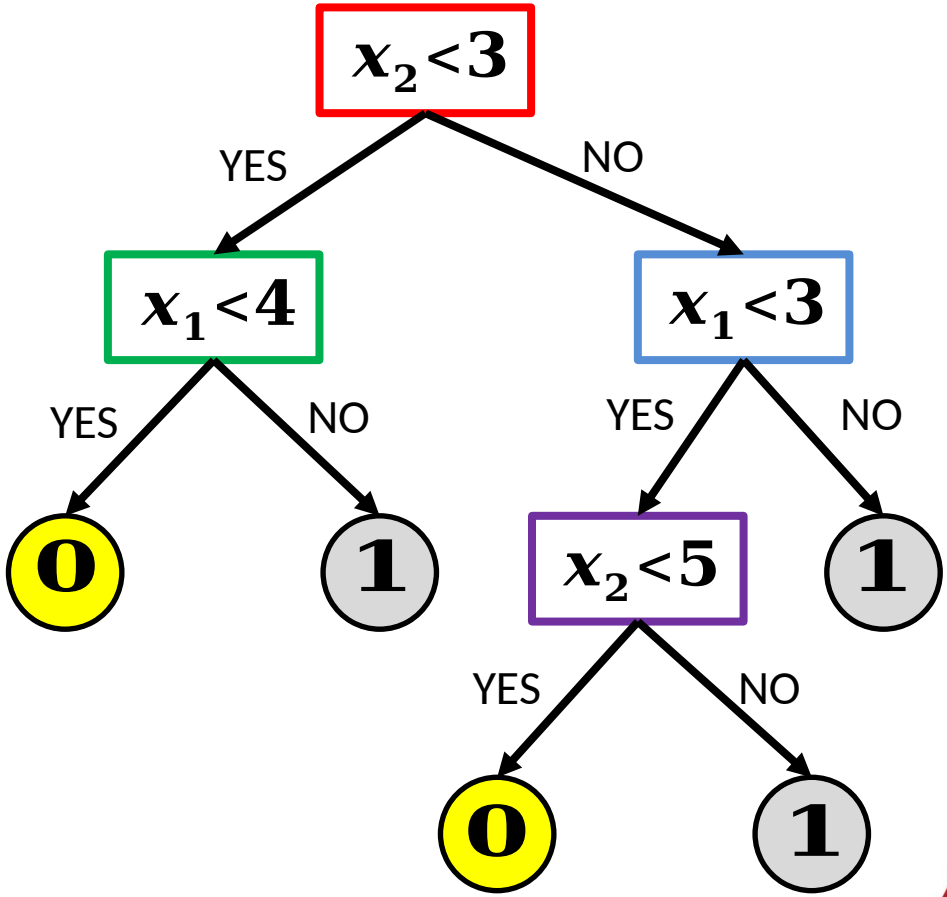
Decision Tree

Another way to divide the space



Decision Tree divides the space

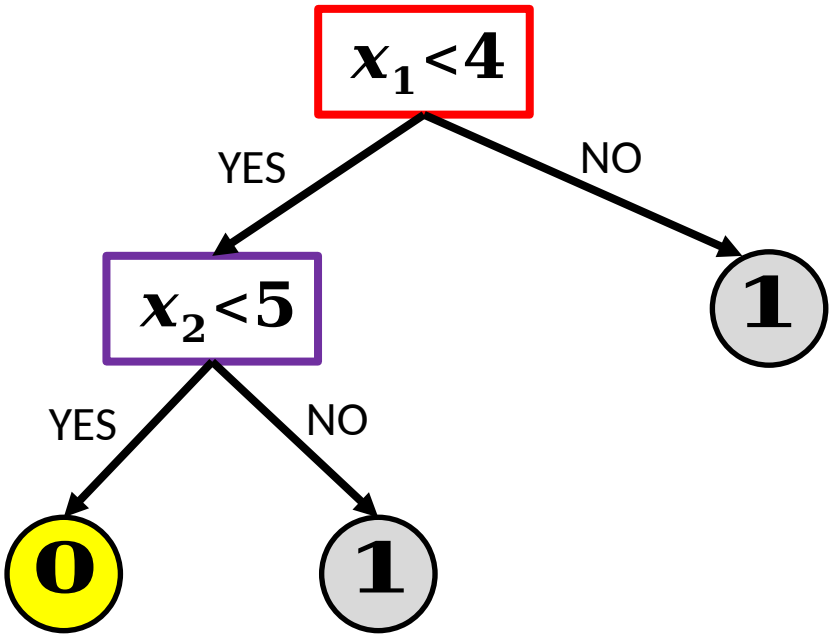




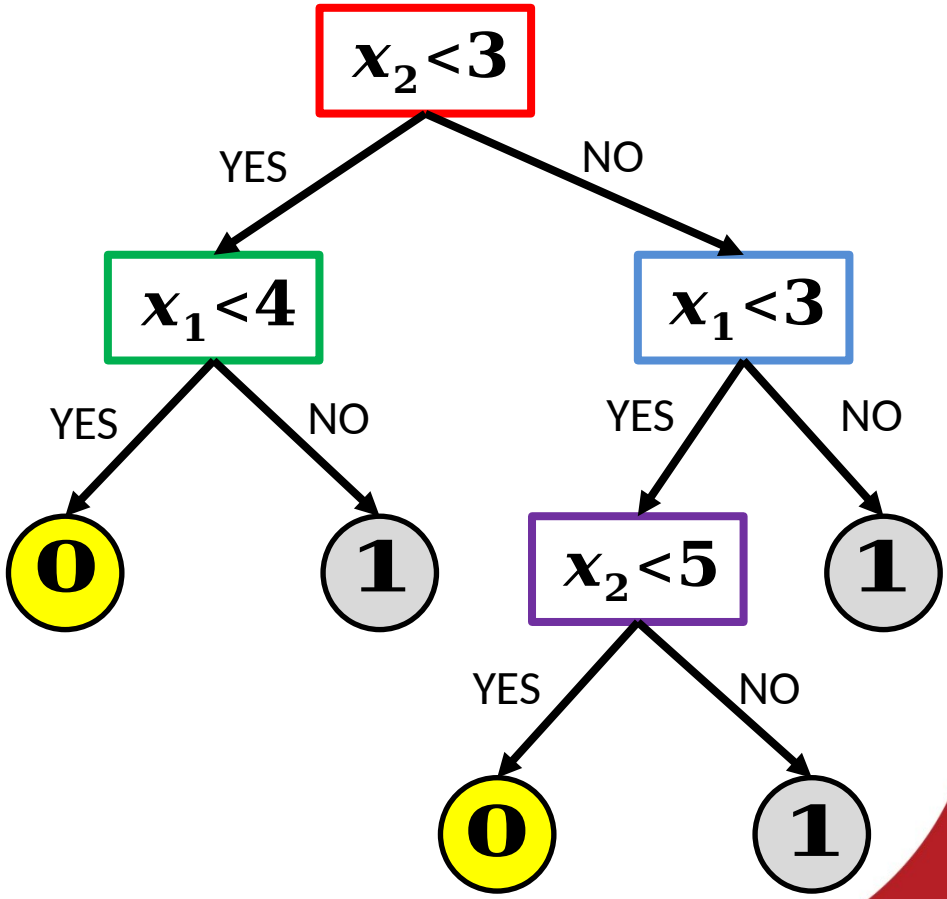
Decision Trees (Many Possibility)

Lancaster University

Decision Tree #1



Decision Tree #2



Decision Tree (many Possibility)

Lancaster University

- There are *exponentially many decision trees* that can be constructed from a given set of features.

- Finding the optimal tree is *computationally infeasible* because the exponential size of the search space.

- Efficient algorithms have been developed to induce a reasonably accurate decision tree in a reasonable amount of time.

What is a good Decision Tree?

Lancaster University

- How to define the goodness of a decision tree?
  - Accuracy
    - We can always build a decision such that each instance has its own leaf node, in which case the decision tree will have 100% accuracy
  - As small as possible
    - Shallow tree, i.e., fewer tests

Lancaster University

Questions?

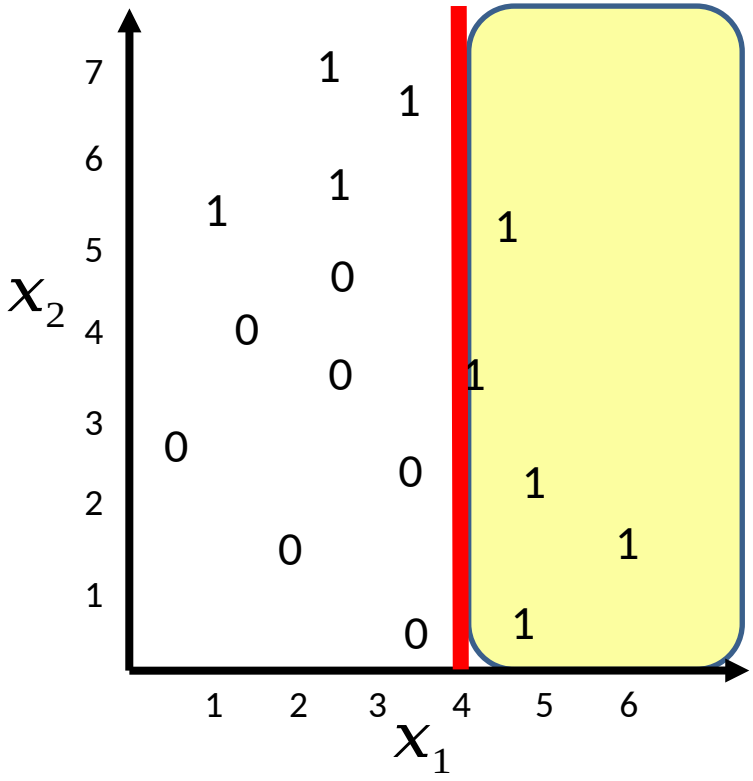
ID3 Algorithm

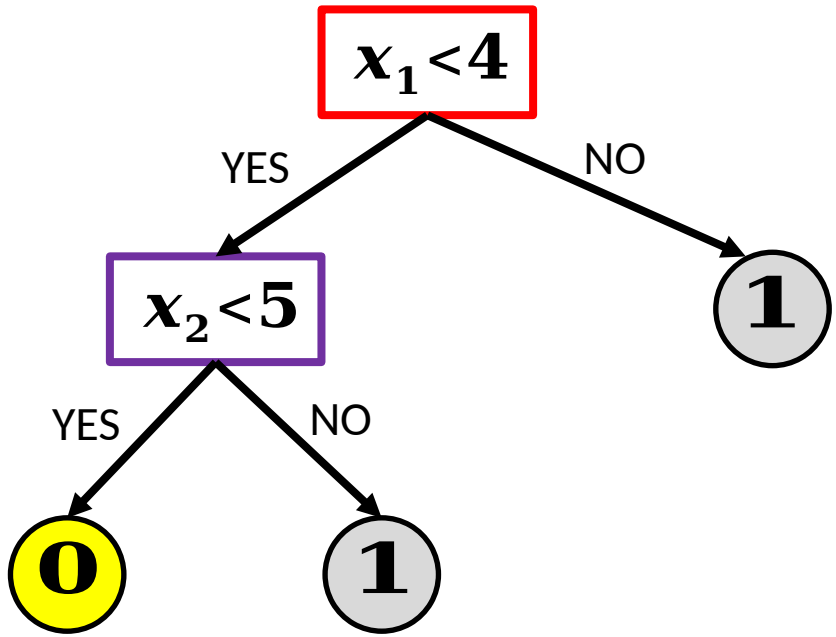
Lancaster University

Guidelines for Finding a Small Decision Tree:

- **Test the most important feature first**
- If you have only one type of example, return a leaf
- Else, choose the next most important feature

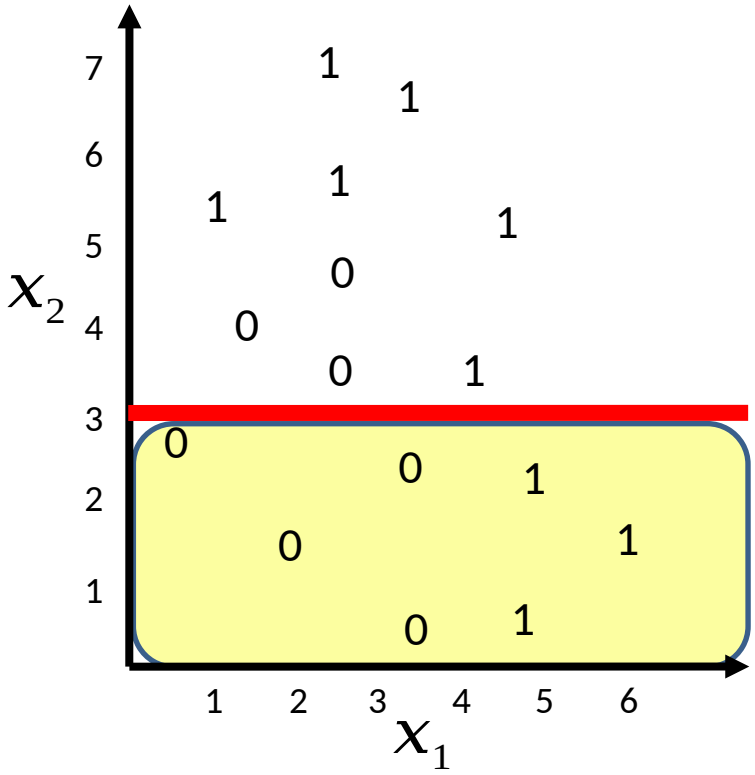
How to determine which feature is better?

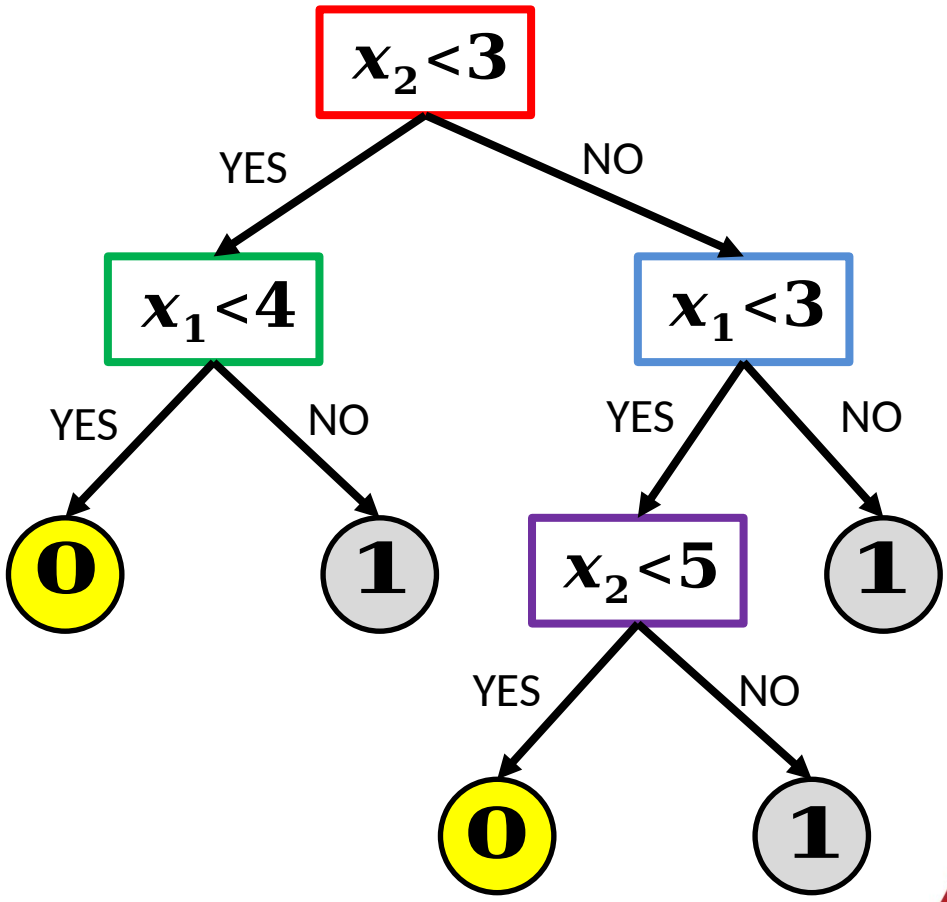




Decision Tree #1
Split based on feature

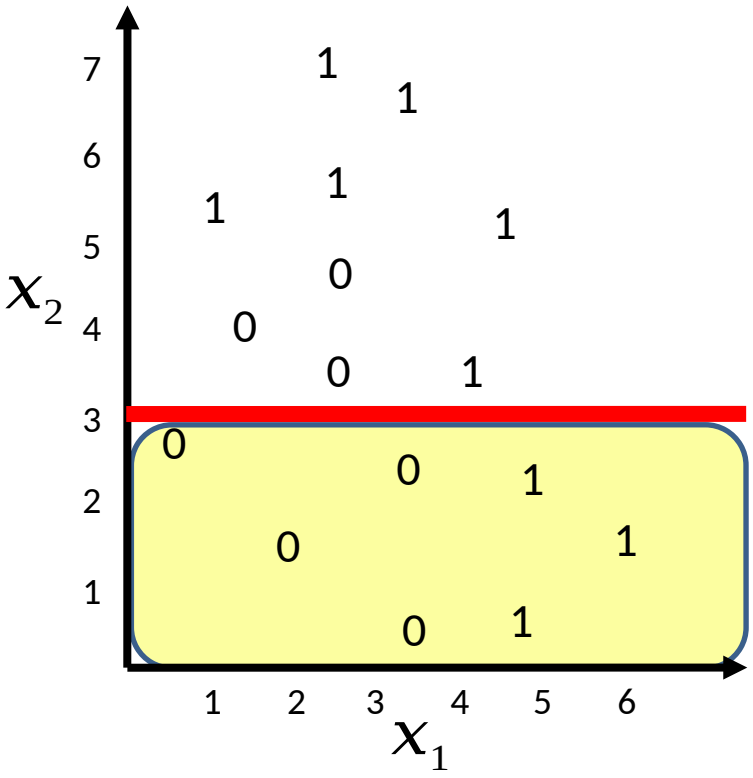
How to determine which feature is better?



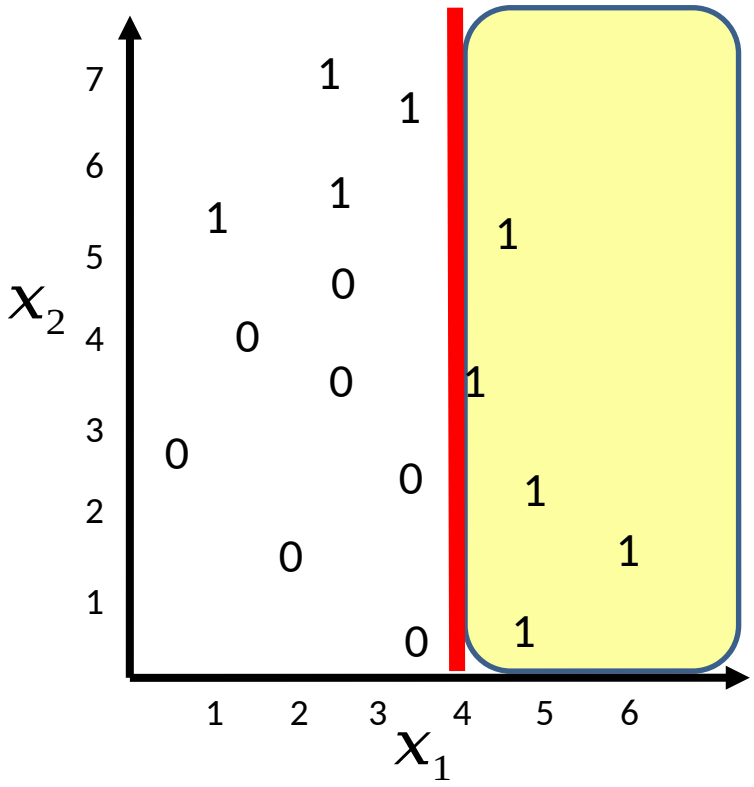


Decision Tree #2
Split based on feature

How to determine which feature is better?



Decision Tree #2
Split based on feature

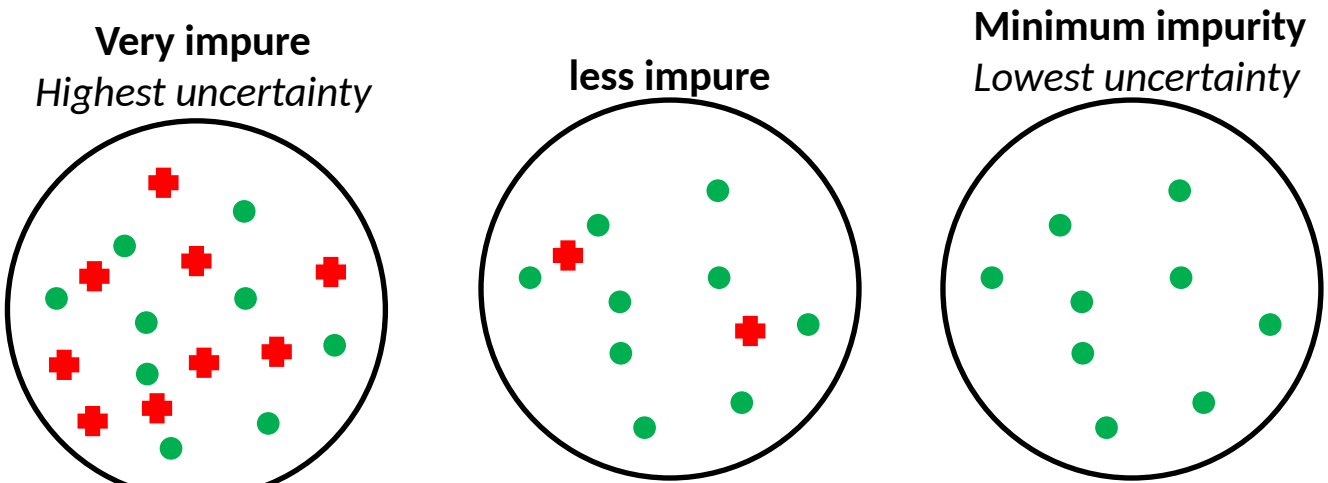


Decision Tree #1
Split based on feature

How to determine which feature is better?

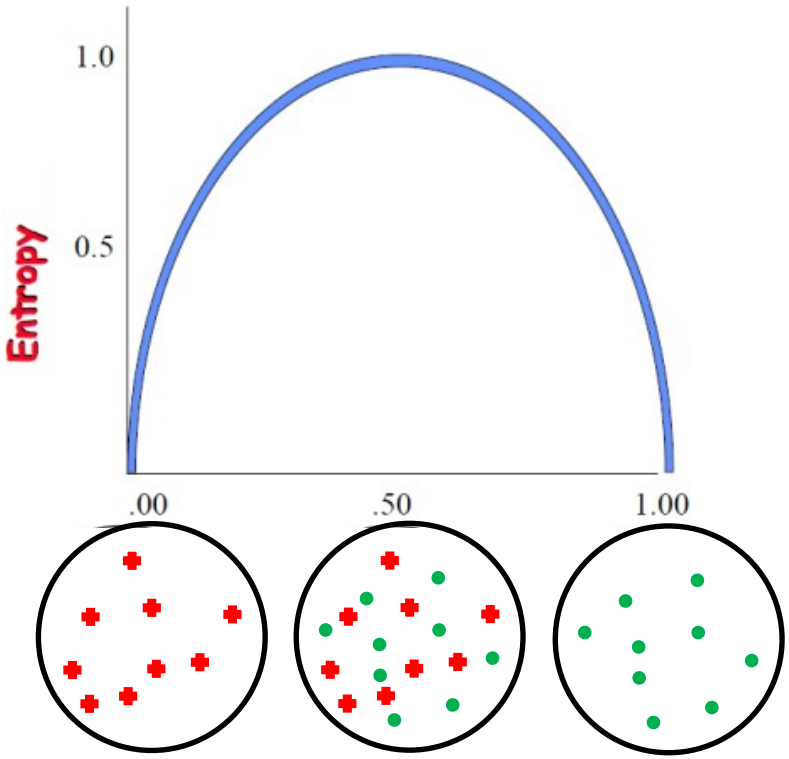
Lancaster University

- A good attribute splits data so that each successor node is as *pure* as possible (i.e. **reduce uncertainty and result in gain in information**)
  - The distribution of samples in each node is such that it mostly contains samples of a single class
- **Entropy**
  - Measure the level of *impurity/uncertainty* in a group of samples



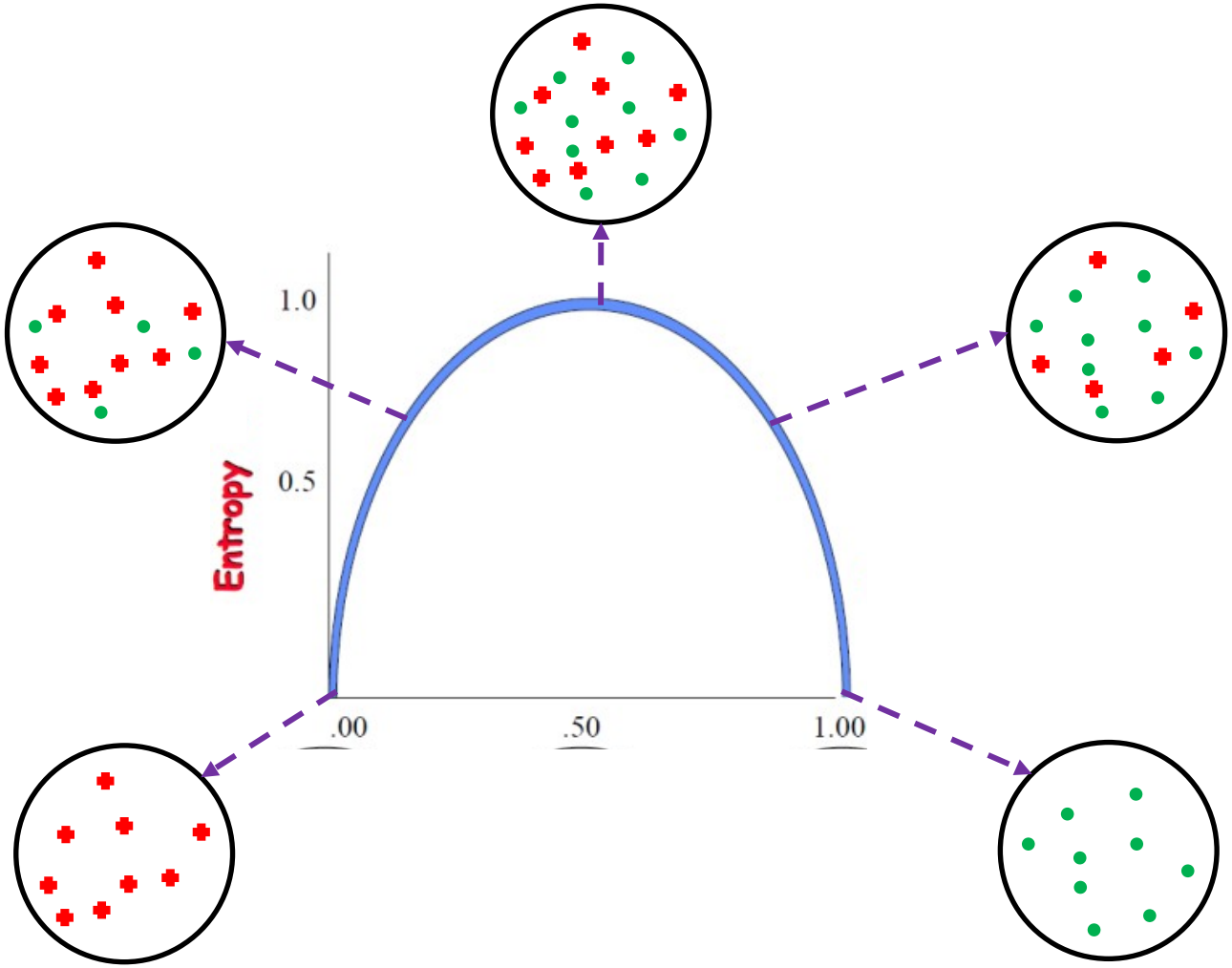
Entropy
(Two Classes)

- is a set of samples
- is the proportion of samples in *class True* (*GREEN*)
- is the proportion of samples in *class False* (*RED*)



Entropy
(Two Classes)

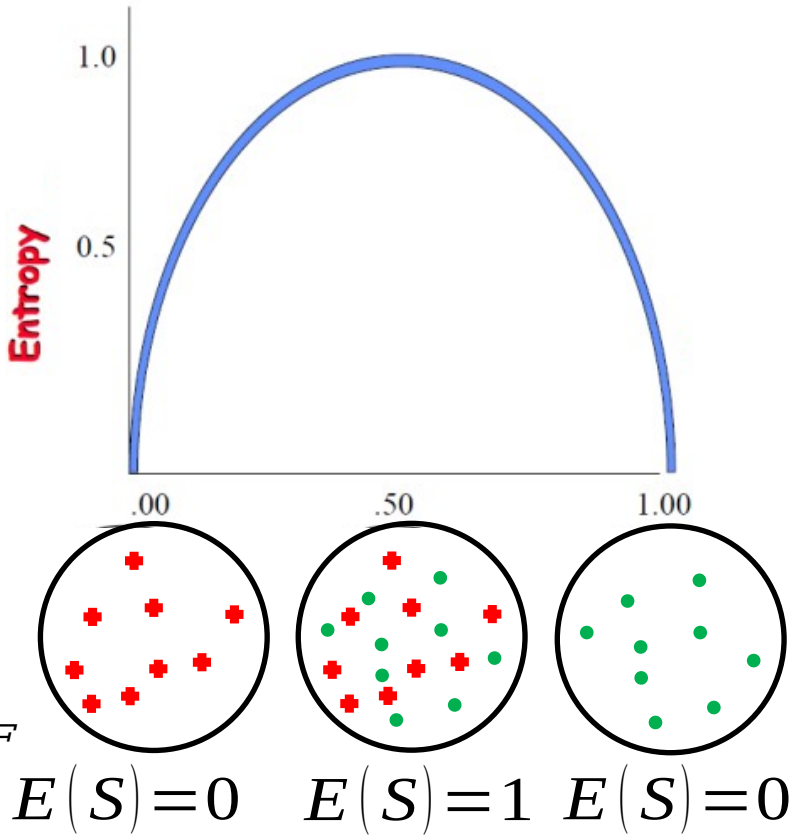
Lancaster University



Entropy (Two Classes)

- is a set of samples
- is the proportion of samples in *class True (GREEN)*
- is the proportion of samples in *class False (RED)*

- **Entropy**
  - Entropy is highest when uncertainty is greatest



$$E(S) = -P_T \log_2 P_T - P_F \log_2 P_F$$

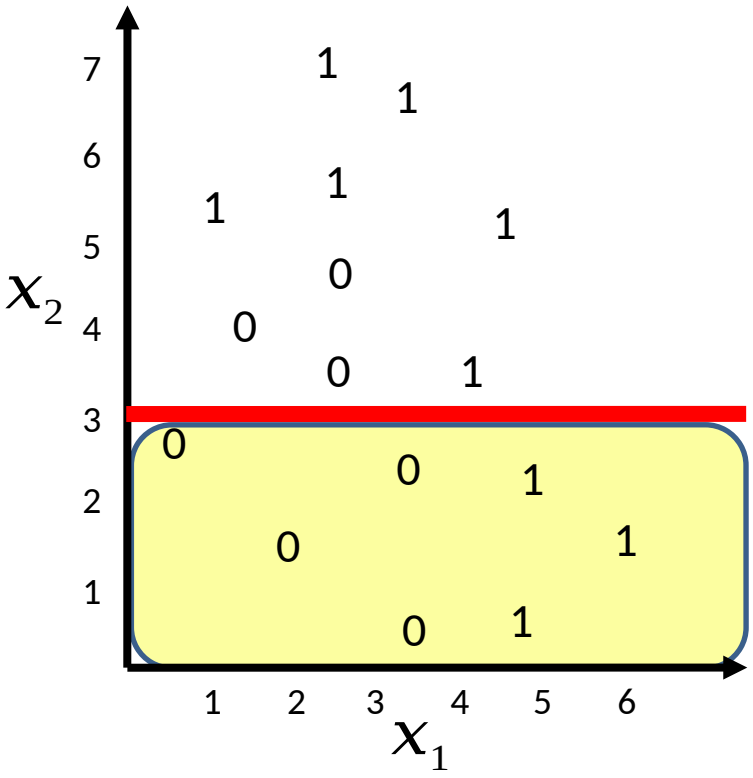
Entropy (More Than 2 Classes)

Lancaster University

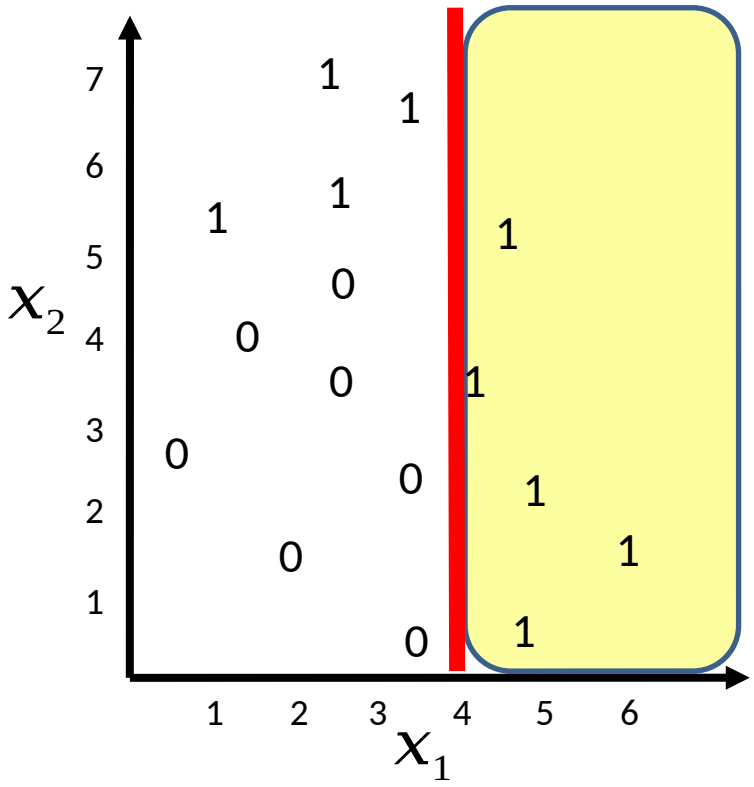
- Entropy can be generalized for classes
- is the proportion of samples in that belong to the *i*-th *class*

$$E(S) = -P_1 \log_2 P_1 - P_2 \log_2 P_2 - \ldots - P_n \log_2 P_n$$

How to determine which feature is better?



Decision Tree #2
Split based on feature

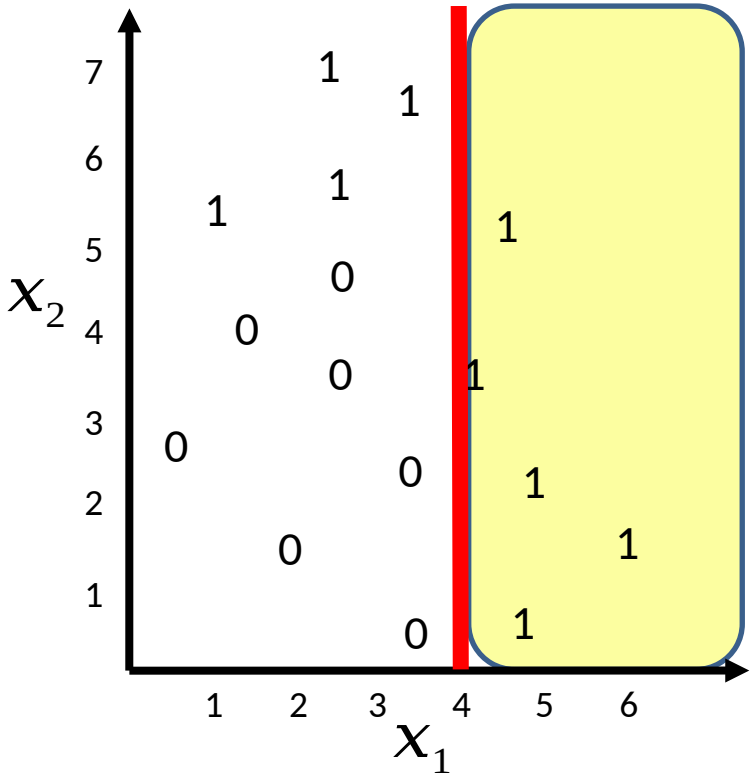


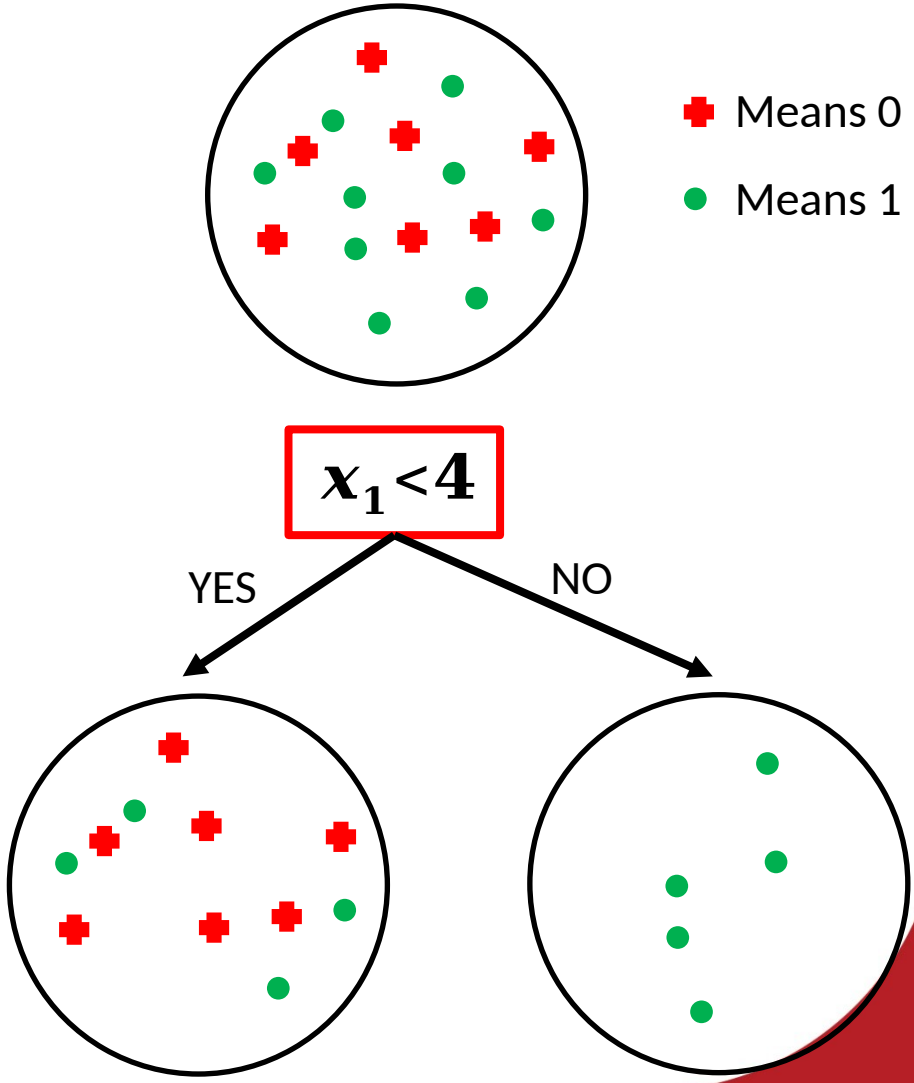
Decision Tree #1
Split based on feature

How to determine which feature is better?

Lancaster University

Decision Tree #1
Split based on feature

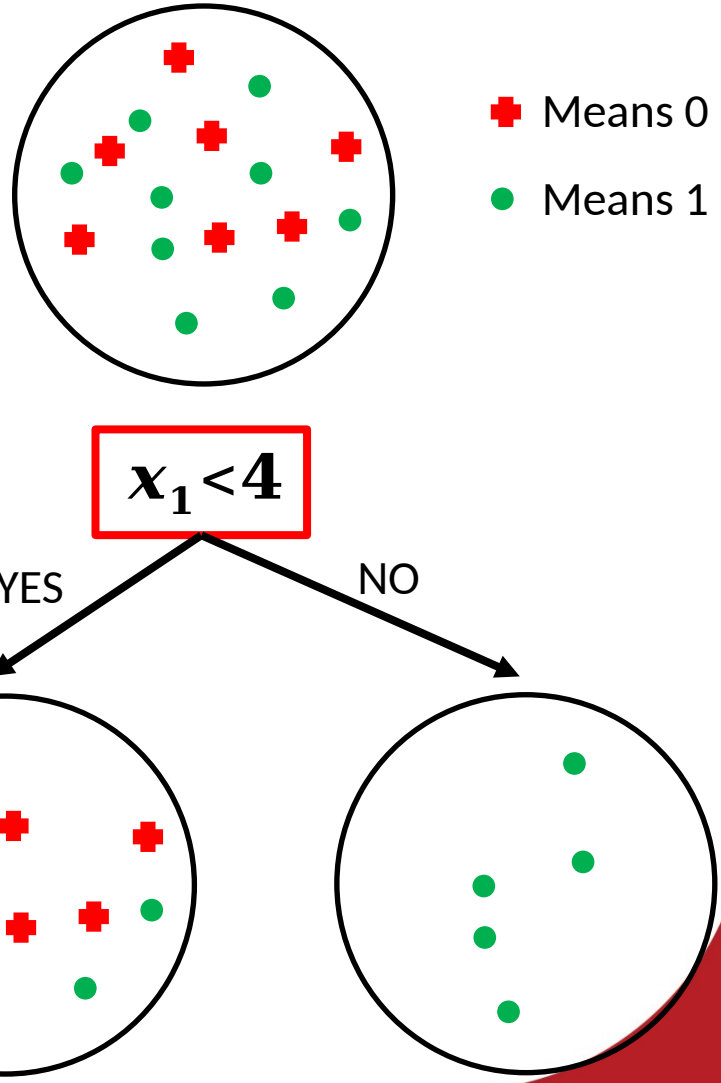




Entropy of Parent Node

Lancaster University

Decision Tree #1
Split based on feature



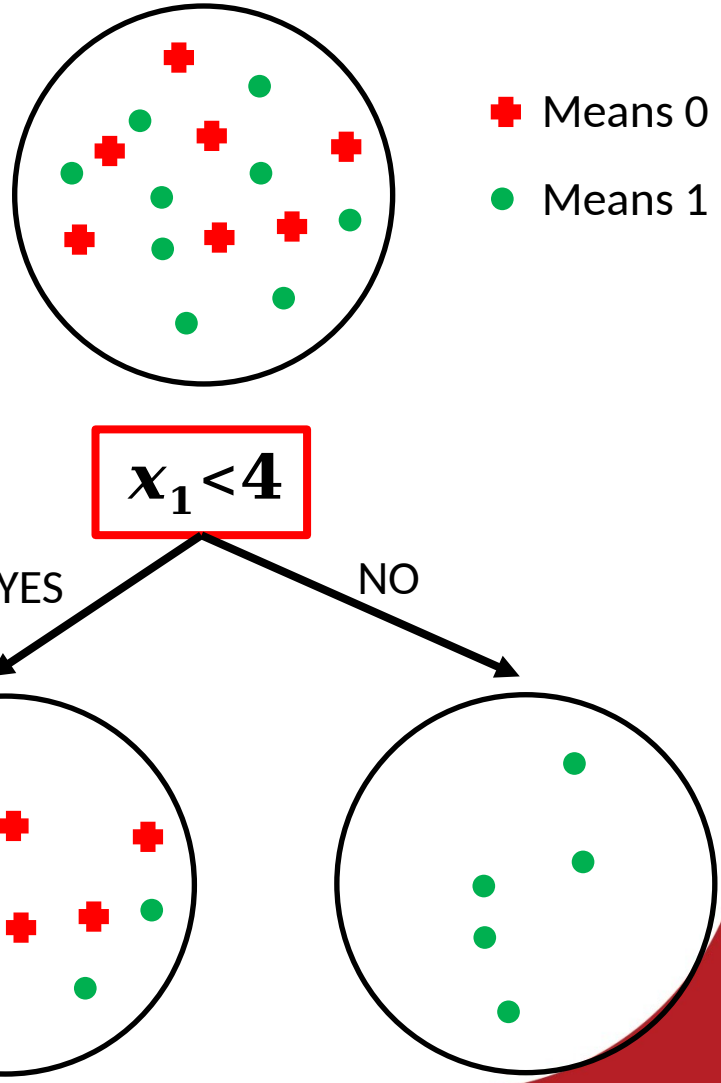
$$E(S) = -P_T \log_2 P_T - P_F \log_2 P_F$$

$$P_T = \frac{9}{16} \qquad P_F = \frac{7}{16}$$

$$E(S) = -\frac{9}{16}\log_2\frac{9}{16} - \frac{7}{16}\log_2\frac{7}{16} = 0.99$$

Entropy of Children Nodes (if is TRUE)

Decision Tree #1
Split based on feature



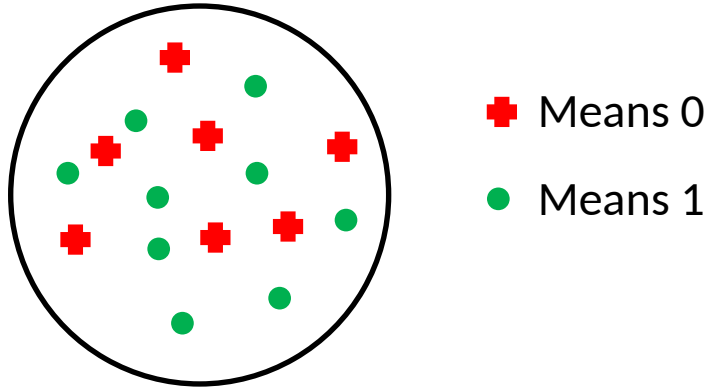
$$E(C_1) = -P_T \log_2 P_T - P_F \log_2 P_F$$

$$P_T = \frac{4}{11} \qquad P_F = \frac{7}{11}$$

$$E(C_1) = -\frac{4}{11}\log_2\frac{4}{11} - \frac{7}{11}\log_2\frac{7}{11} = 0.95$$

Entropy of Children Nodes (if is FALSE)

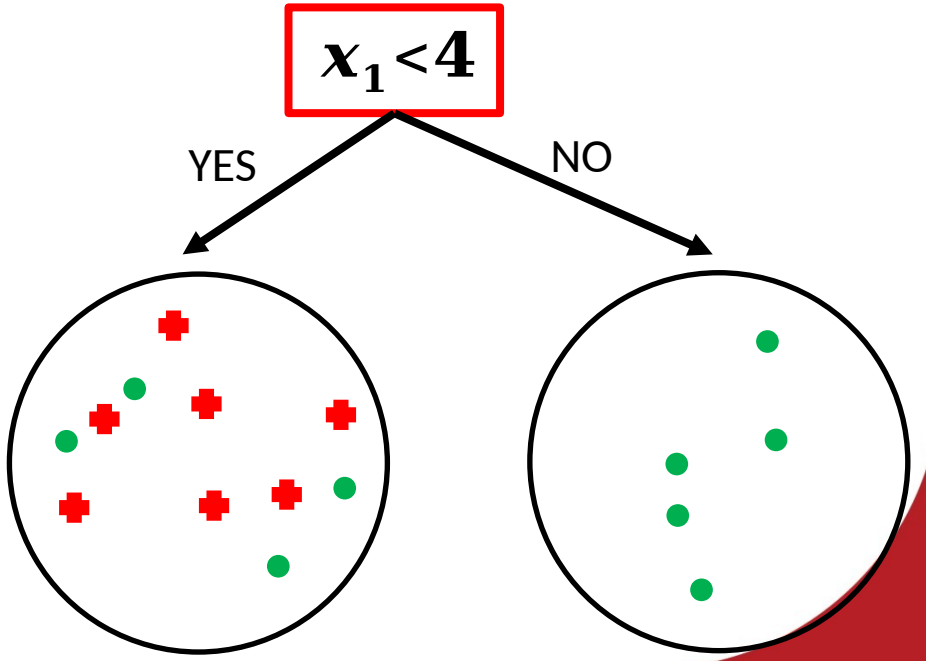
Decision Tree #1
Split based on feature



$$E(C_2) = -P_T \log_2 P_T - P_F \log_2 P_F$$

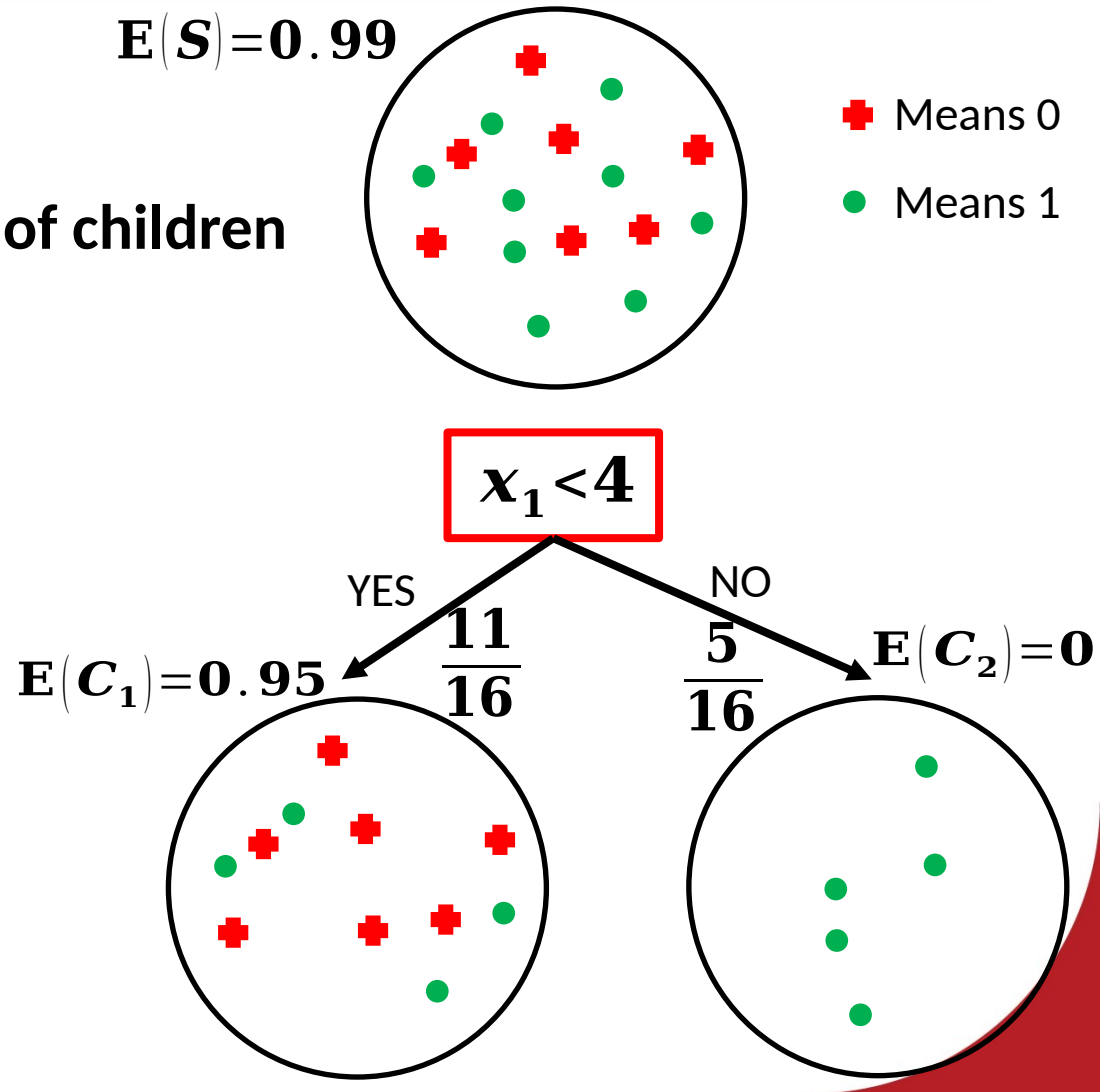
$$P_T = \frac{5}{5} \qquad P_F = \frac{0}{5}$$

$$E(C_2) = -\frac{5}{5}\log_2\frac{5}{5} - \frac{0}{5}\log_2\frac{0}{5} = 0$$



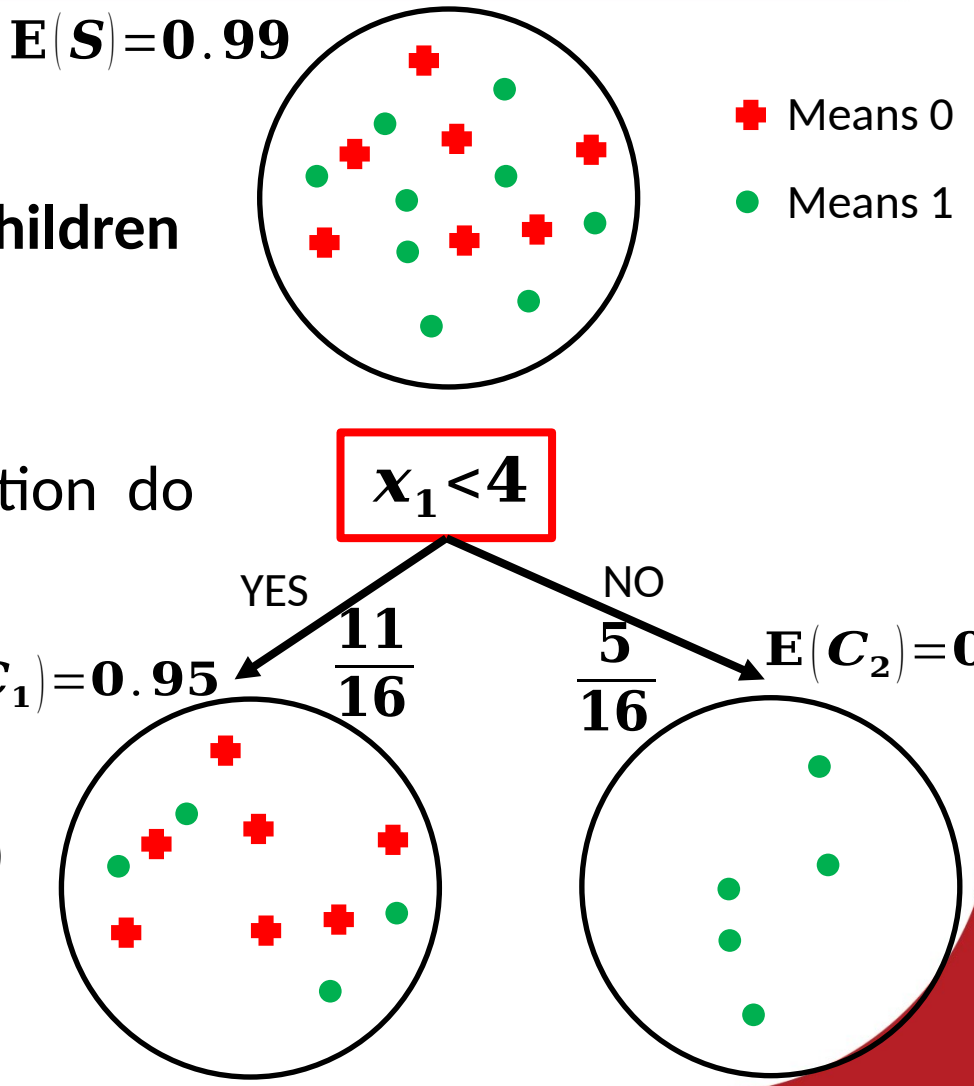
Entropy of children Nodes

Weighted average Entropy of children



Information Gain

Lancaster University



Weighted average Entropy of children

Question: How much information do we gain by using test?

Answer:

Gain(S,) = ***Entropy*** *(parent)* – ***Entropy*** *(children)*

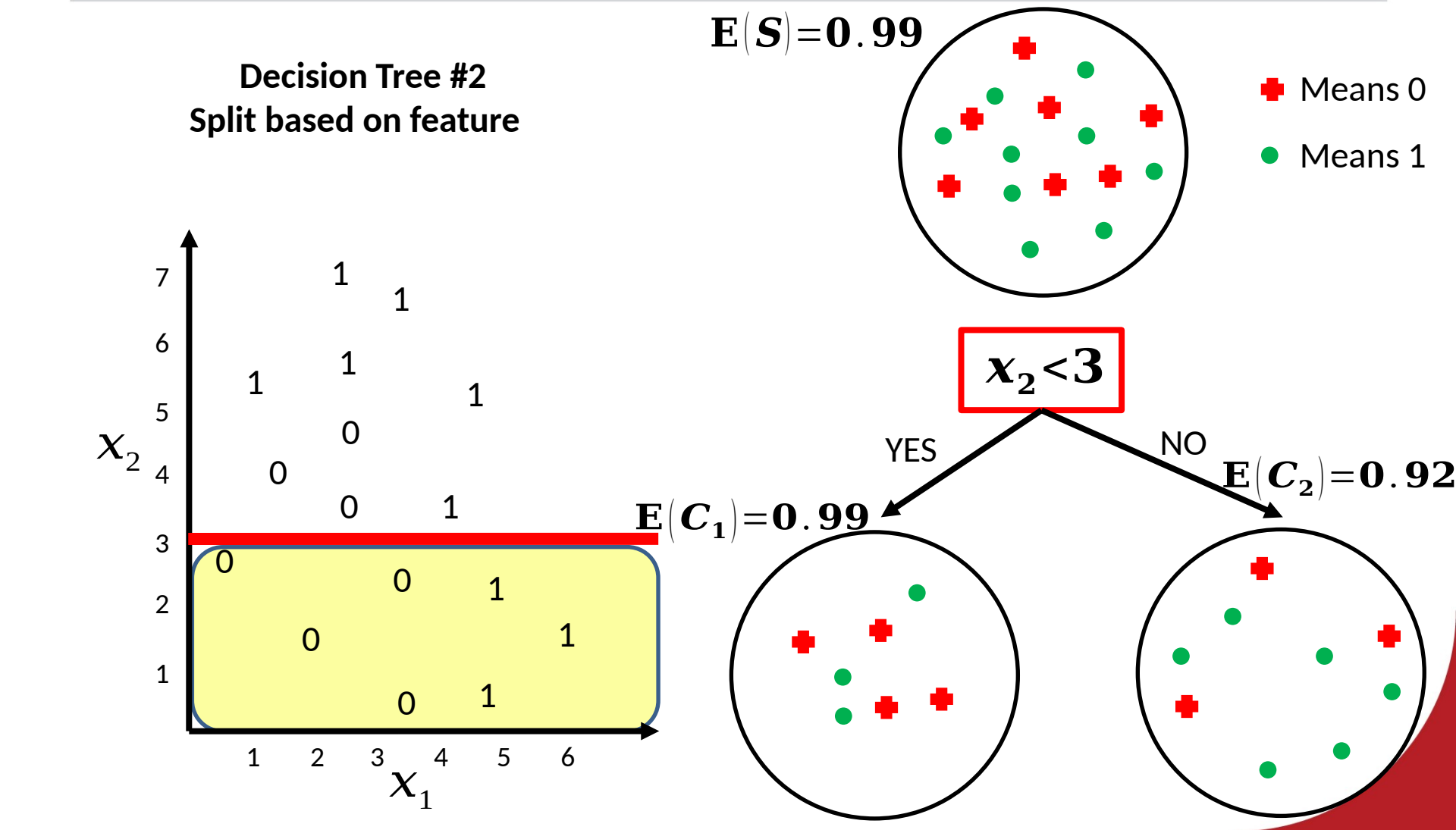
Gain(S,) = -

Information Gain

Lancaster University

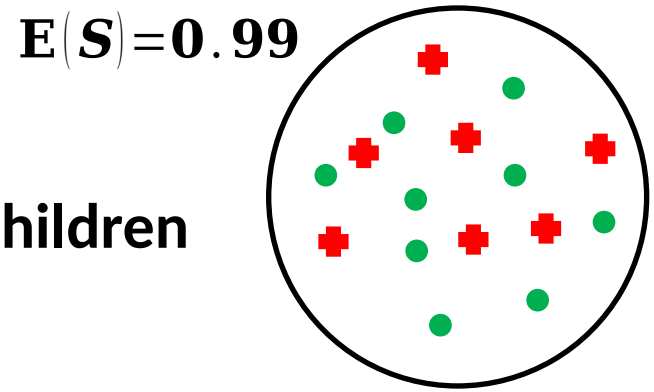
- **Information Gain:** Also called "*the expected reduction in entropy*" is
  - *Gain(S,A)* = ***Entropy*** *(parent)* – *[Weighted Average]* ***Entropy*** *(children)*
  - Decision tree algorithm will **maximize information gain**
    - For every node in a Decision Tree, select a feature which maximize information gain

How to determine which feature is better?



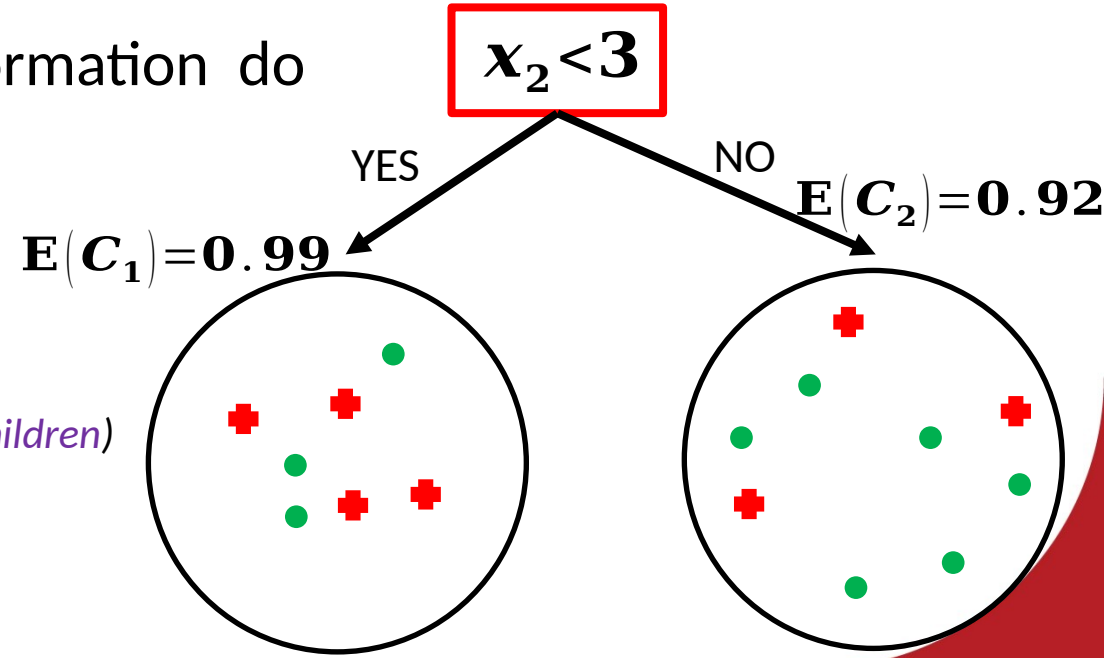
Information Gain

Lancaster University



Weighted average Entropy of children

Question: How much information do we gain by using test?



Answer:

Gain(S,) = ***Entropy*** *(parent)* – ***Entropy*** *(children)*

Gain(S,) = **-**

Decision Tree (Summary)

Lancaster University

- **Advantages**
  - Can be applied to the data from any distribution for example, data does not have to be separable with a linear boundary
  - Simple to understand and interpret
  - Able to handle both numerical and categorical data
  - Extremely fast
- **Disadvantages**
  - Trees can be very non-robust. A small change in the training data can result in a large change in the tree and consequently the final predictions
  - The problem of learning an optimal decision tree is known to be NP-complete.
  - Decision trees are **prone to overfitting**, especially when a tree is particularly **deep**.

Lancaster University

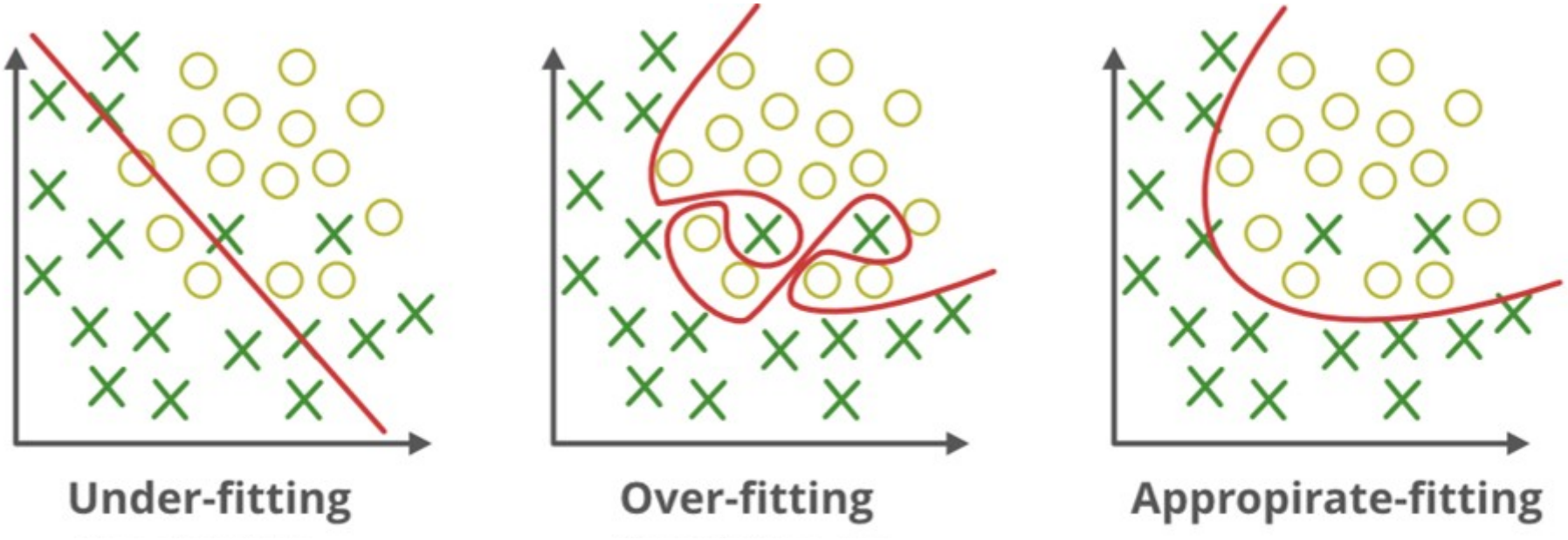
Questions?

What is overfitting?

Lancaster University

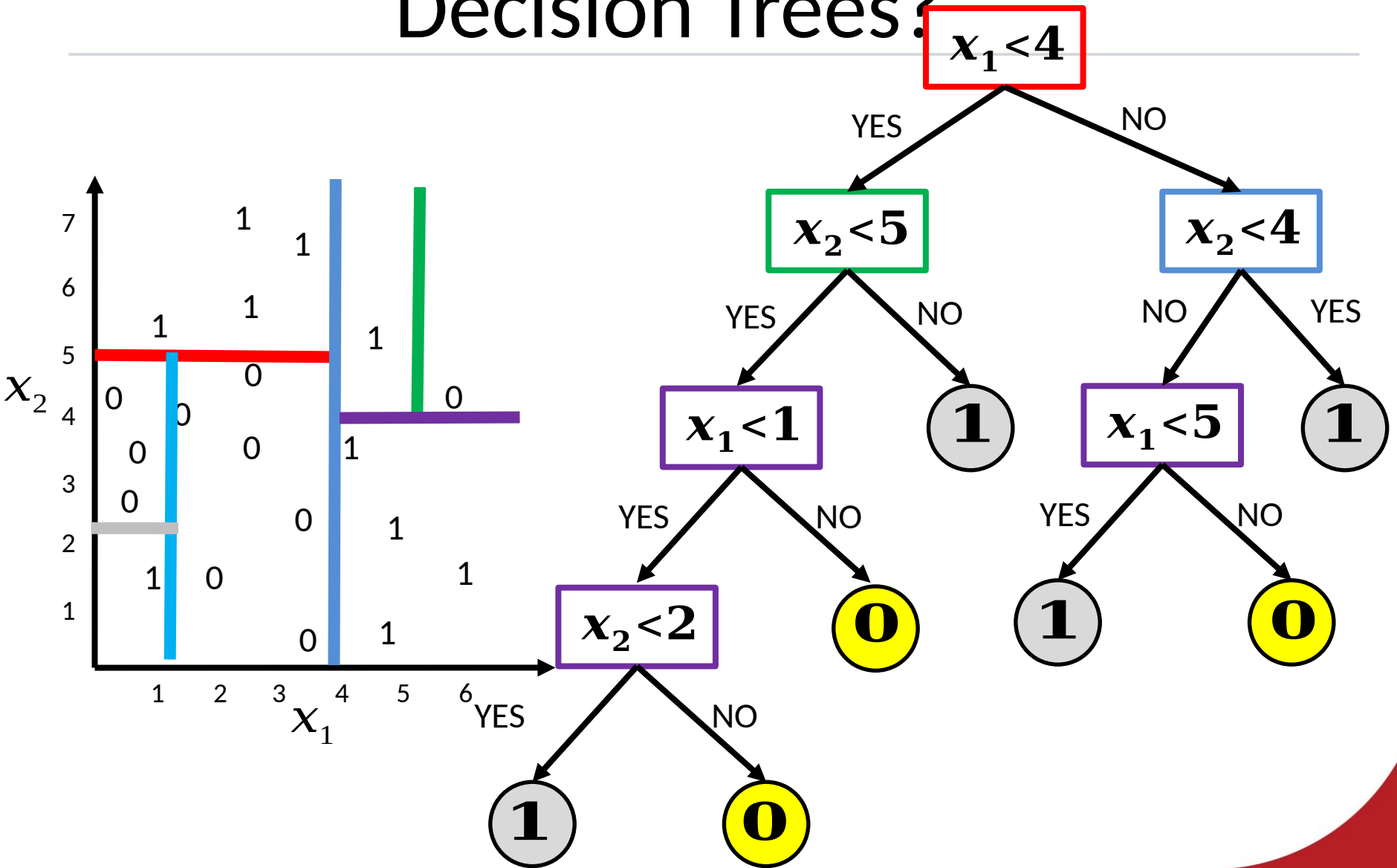
- Overfitting means **memorizing**
- **Memorizing is not learning!**
- Overfitting happens when your model fits too well to the training set.
- It then becomes difficult for the model to generalize to new examples that were not in the training set.
- For example, your model recognizes specific images in your training set instead of general patterns.

Overfitting vs. Underfitting (Illustrative Example)

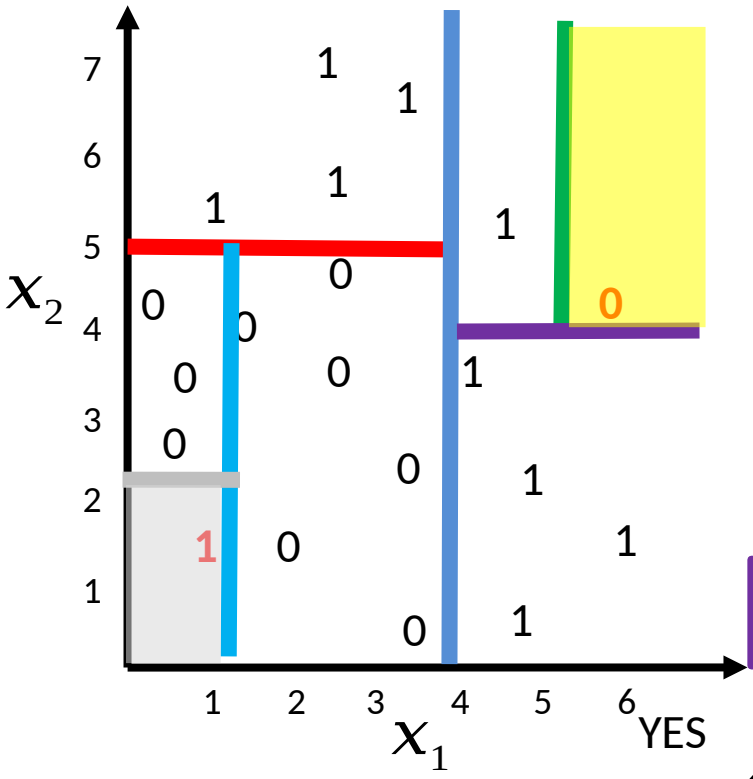


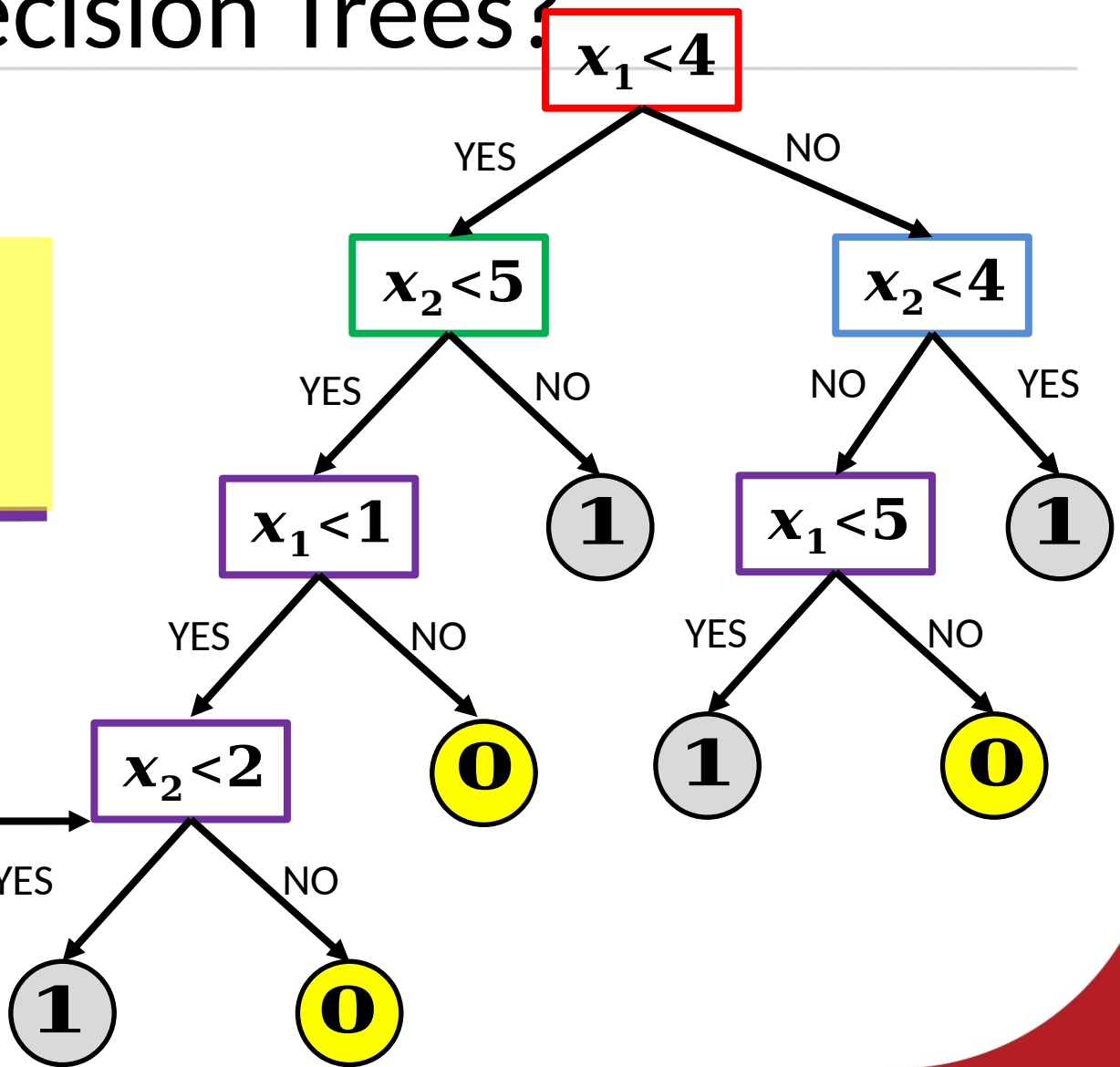
What is overfitting in Decision Trees?

Lancaster University



What is overfitting in Decision Trees?





Decision Tree
How to avoid overfitting?

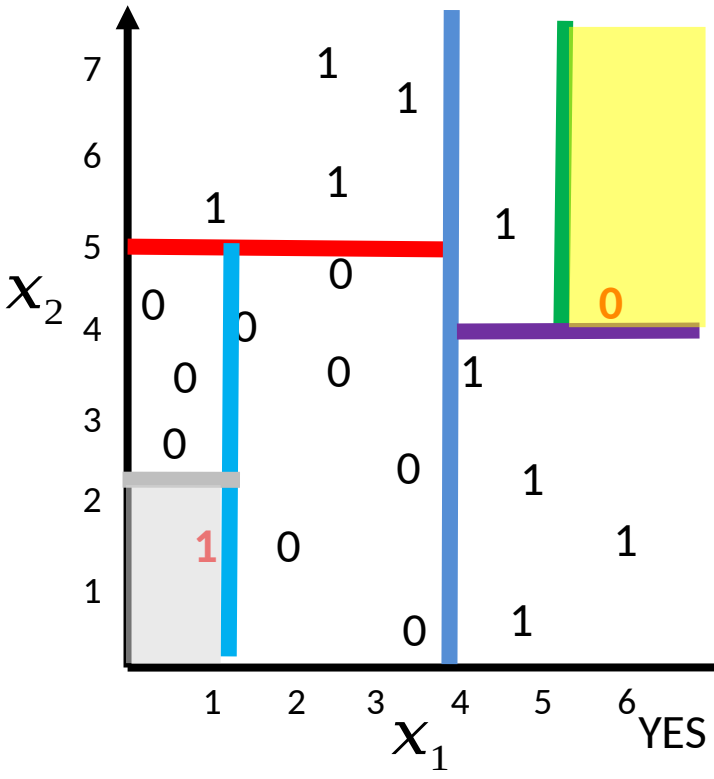
Lancaster University

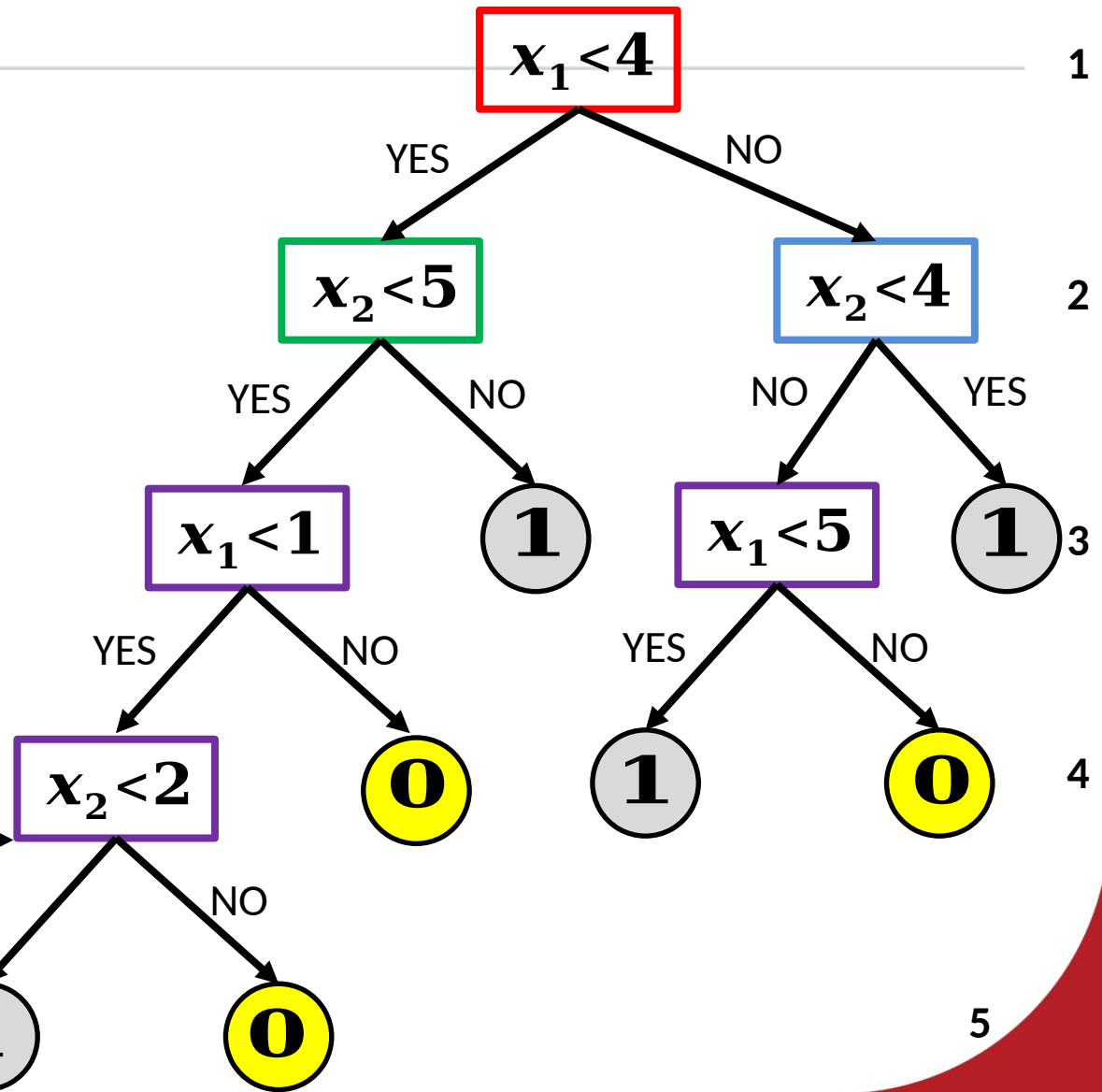
- **Set a max depth of tree**
  - It will increase error

- **Random Decision Forests (RDFs)**
  - A random forest is simply a collection of decision trees whose results are aggregated into one final result.
  - How to train a Random Decision Forest?
    - by training on different samples of the data
    - by using a random subset of features

Set a max depth of tree

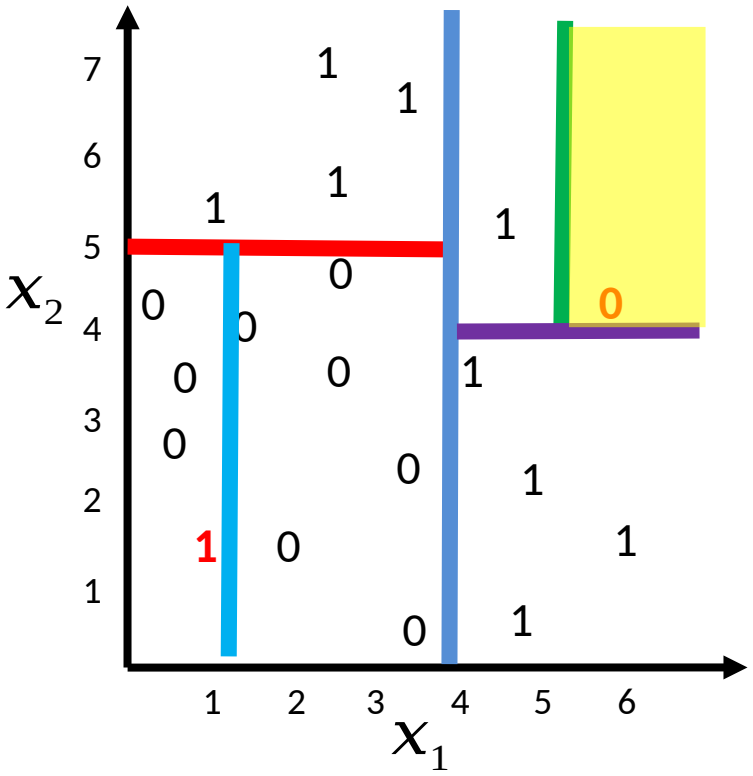
Lancaster University

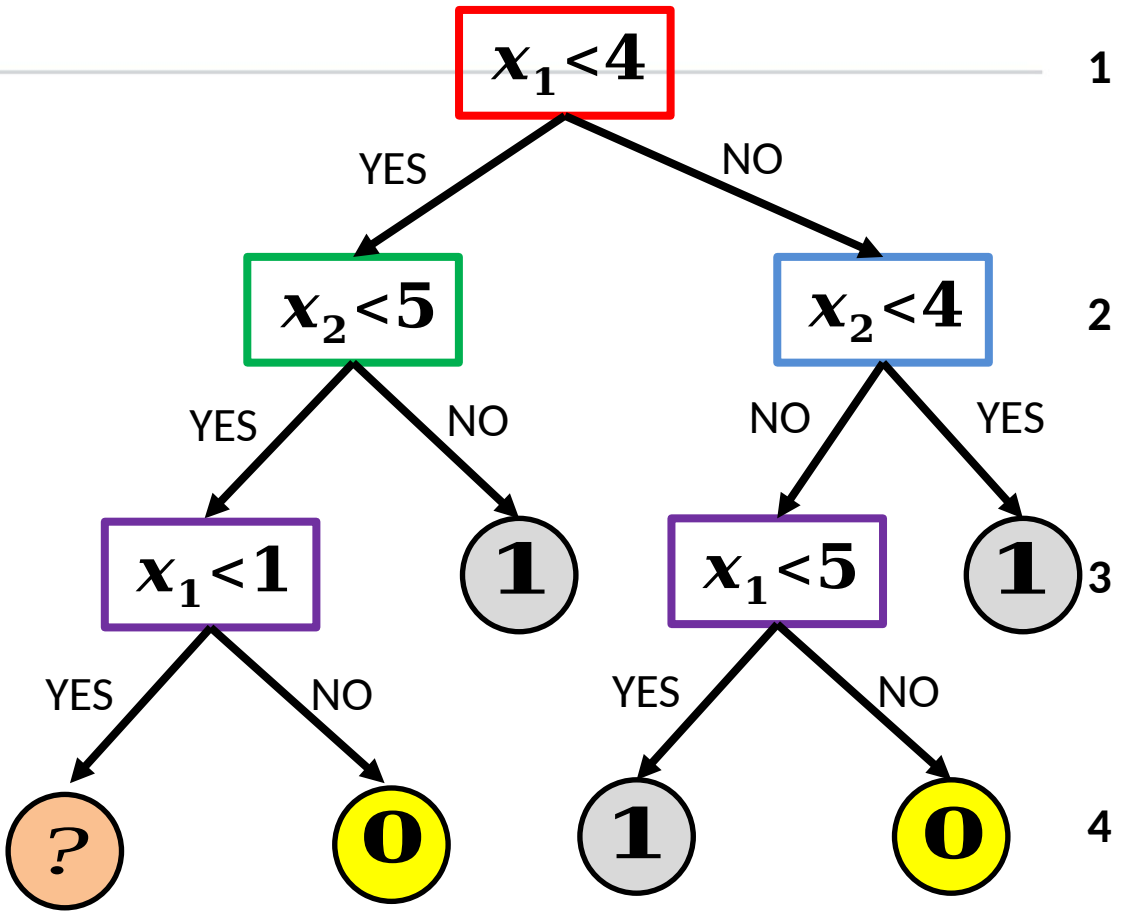




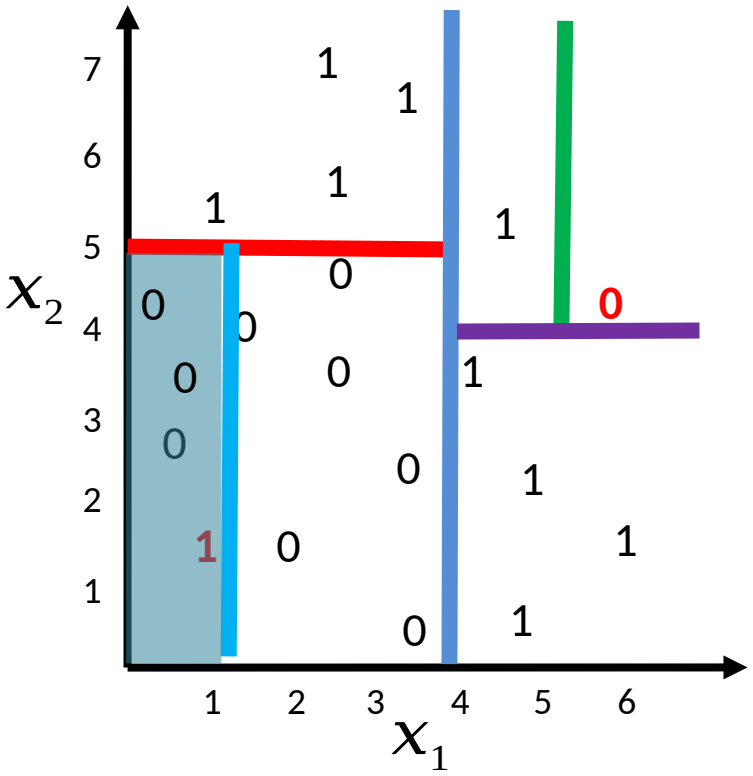
Set a max depth of tree

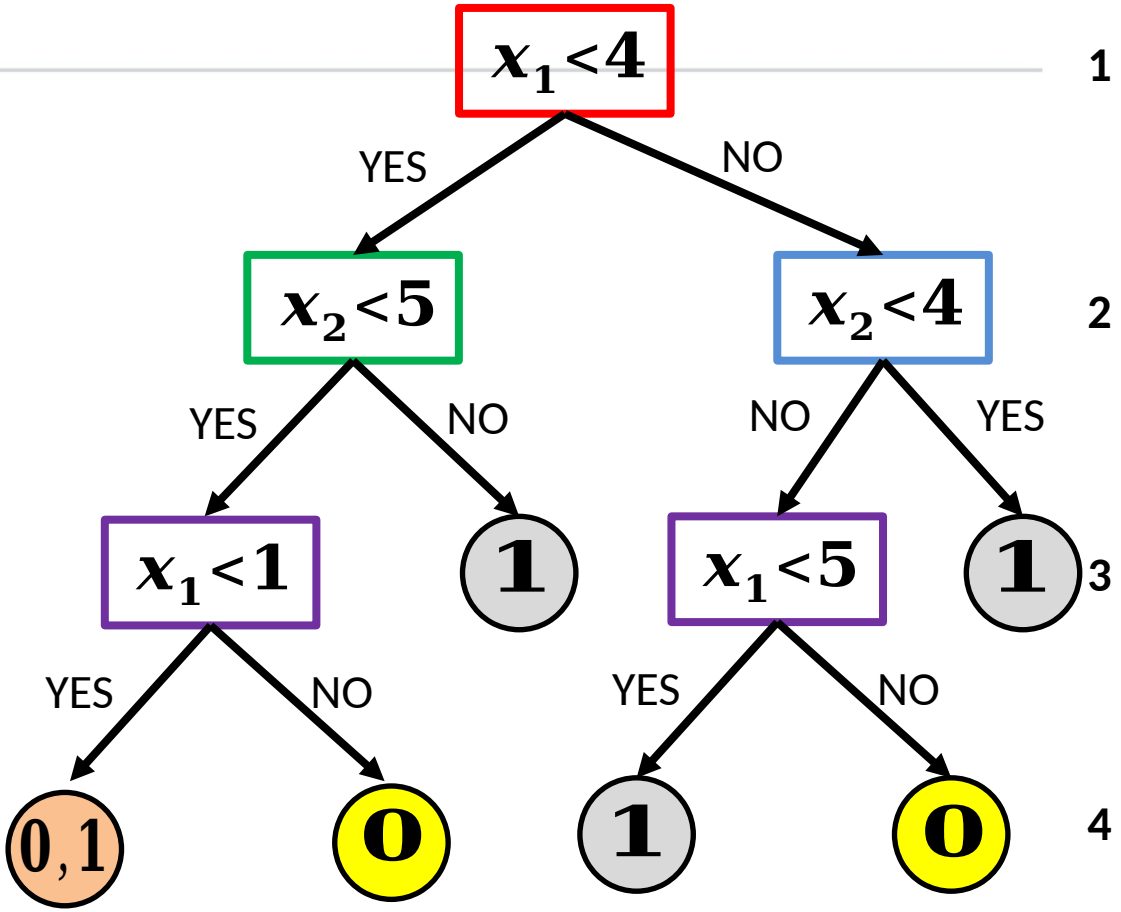
Lancaster University





Set a max depth of tree





Decision Tree
How to avoid overfitting?

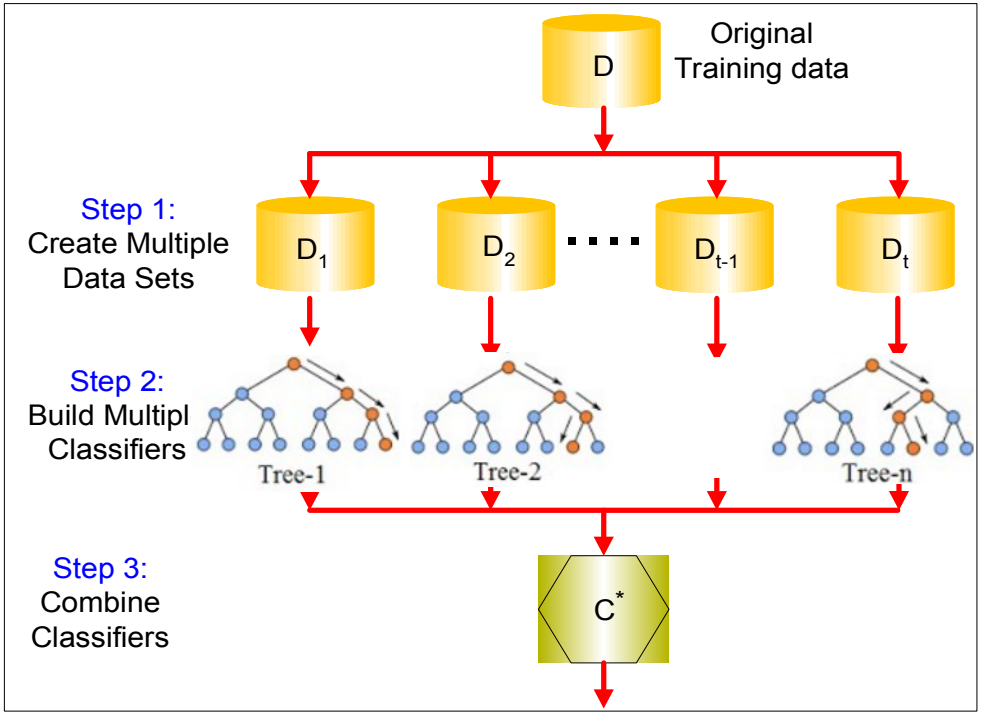
Lancaster University

- **Set a max depth of tree**
  - It will increase error

- **Random Decision Forests (RDFs)**
  - A random forest is simply a collection of decision trees whose results are aggregated into one final result.
  - How to train a Random Decision Forest?
    - by training on different samples of the data
    - by using a random subset of features

RDFs
(by training on different samples of data)

- **Training**
  - Given a set of training samples, a training set is sampled with replacement from
  - A classifier model C is learned for each training set
- **Testing** (classify an unknown sample)
  - Each classifier C returns its class prediction
  - The bagged classifier counts the votes and assigns the class with the most votes to



: Number of trees in RDF

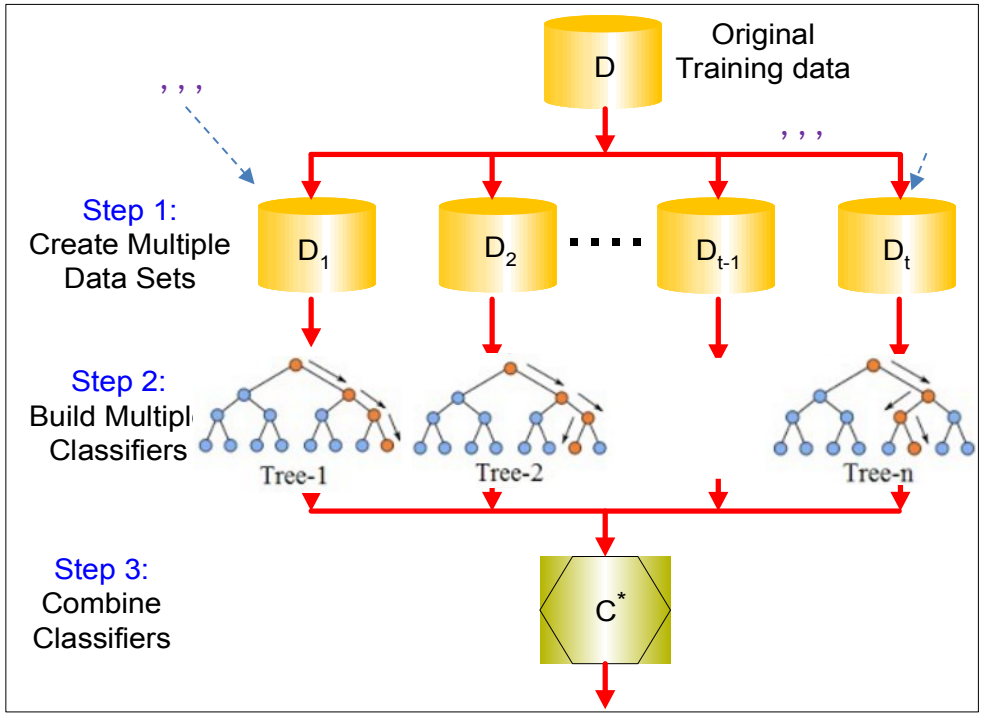
: number of samples

: Number of samples for training th decision tree

RDFs (Random Subset of Features)

- Adding randomness to each decision tree by searching for the best feature among a **random subset of features**
- This results in a wide diversity that generally results in a better classification model

Features: , , ...,



Lancaster University

Questions?

Lancaster University

Example Dataset

Another Example (Restaurant Domain)

Lancaster University

- Model a patron's decision of whether to wait for a table at a restaurant.

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Alternate: alternative restaurant nearby
Bar: bar area to wait
Fri/Sat: true on Fridays and Saturdays
Hungry: whether we are hungry
Patrons: how many people in restaurant

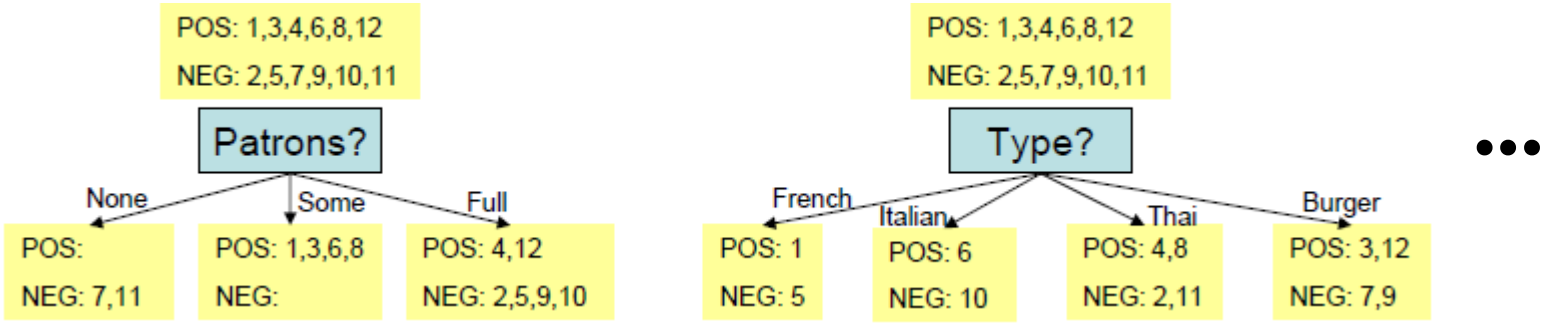
price: price range
raining: raining outside
reservation: whether we made a reservation
type: kind of restaurant
WaitEstimate: estimated wait

Start building a Decision Tree Example (Restaurant Domain)

Lancaster University

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Example – Root Selection Attribute Entropy



- **Which attribute is better as the root of the decision tree?**

Alternate: alternative restaurant nearby
Bar: bar area to wait
Fri/Sat: true on Fridays and Saturdays
Hungry: whether we are hungry
Patrons: how many people in restaurant

price: price range
raining: raining outside
reservation: whether we made a reservation
type: kind of restaurant
WaitEstimate: estimated wait

Example – Root Selection Set Entropy

POS: 1,3,4,6,8,12

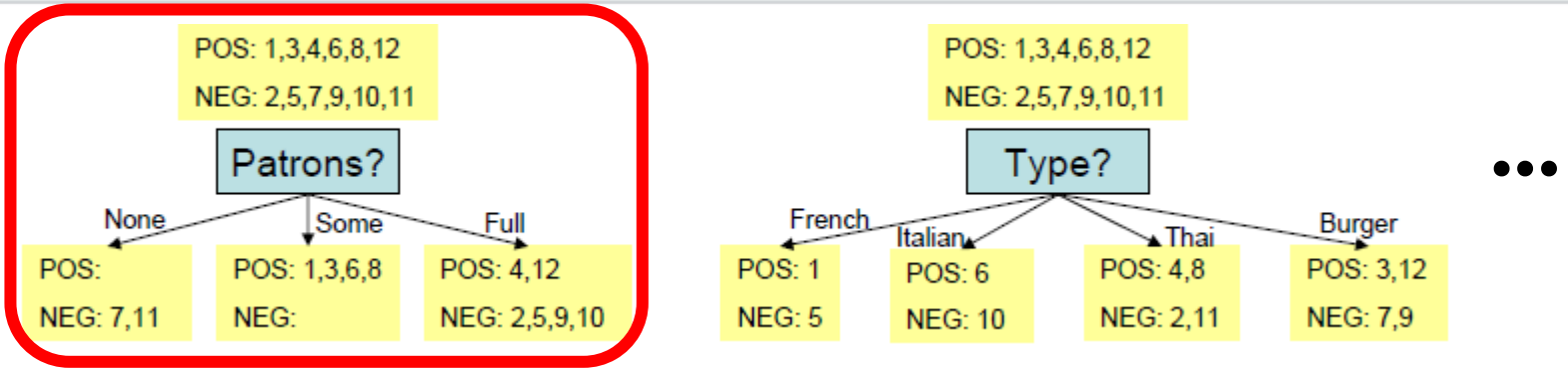
NEG: 2,5,7,9,10,11

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

- For our Restaurant example:

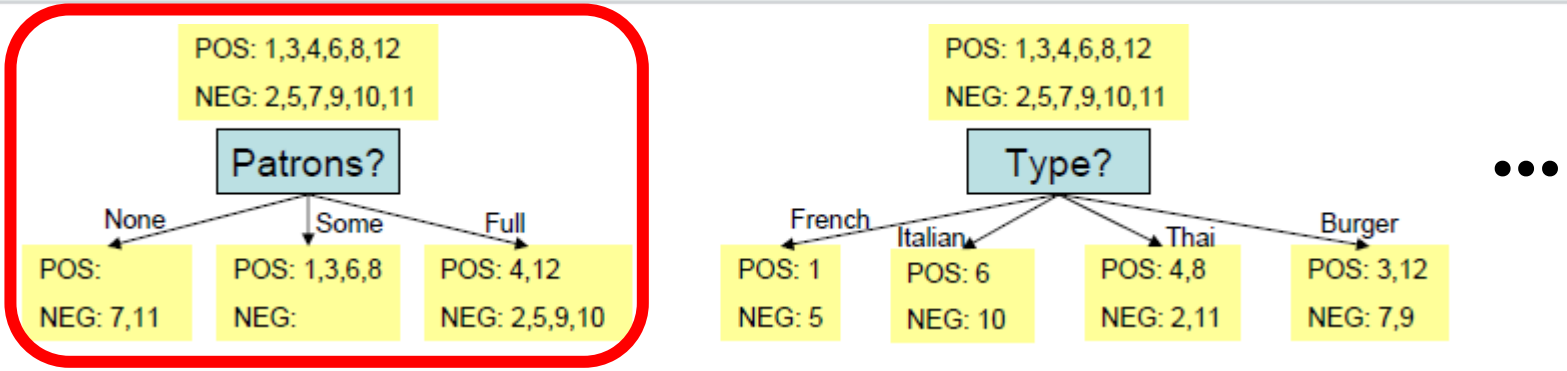
$$E(S) = -P_T \log_2 P_T - P_F \log_2 P_F = -\frac{6}{12}\log_2\frac{6}{12} - \frac{6}{12}\log_2\frac{6}{12} = 1$$

Example - Root Selection Attribute Entropy (*Patrons*)



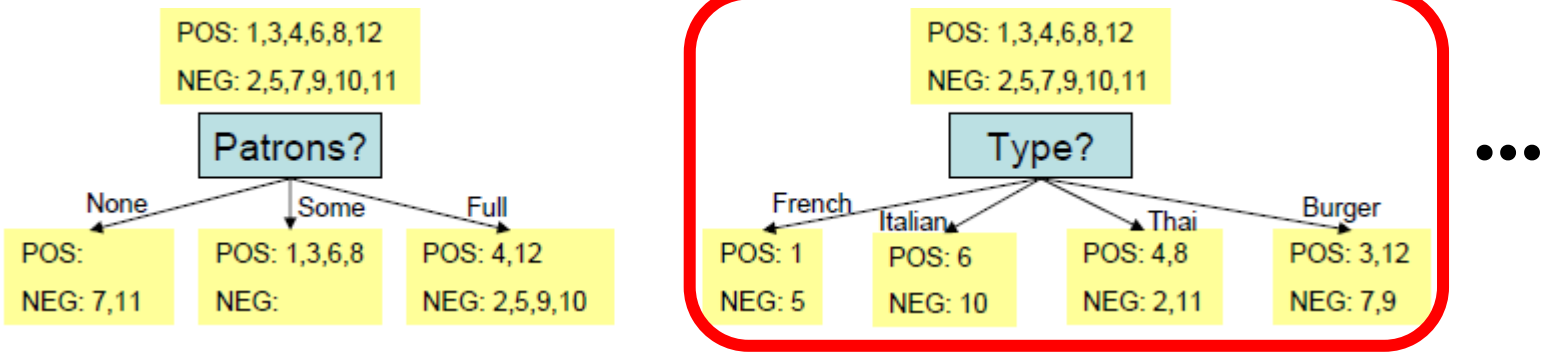
- **Example:** Attribute *Patrons*
  - *Patrons=None: 0 samples YES and 2 samples NO*
  - *Patrons=Some: 4 samples YES and 0 samples NO*
  - *Patrons=Full: 2 samples YES and 4 samples NO*
  - **Average Entropy:** The weighted average over all sets resulting from the split

Example - Root Selection Attribute Entropy (*Patrons*)



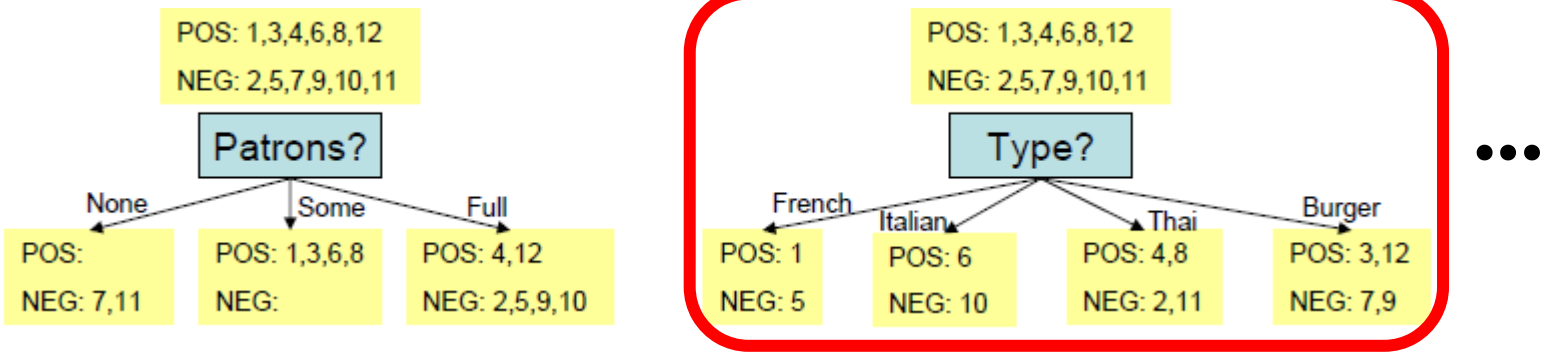
- **Example:** Attribute *Patrons*
  - **Average Entropy:** The weighted average over all sets resulting from the split
  - **Information Gain:** Also called "*the expected reduction in entropy*" is
    - *Gain(S,A) = Entropy before – Average Entropy after choosing A*

Example - Root Selection Attribute Entropy (*Type*)



- **Example:** Attribute *Type*
  - *Type=French: 1 samples YES and 1 samples NO*
  - *Type=Italian: 1 samples YES and 1 samples NO*
  - *Type=Thai: 2 samples YES and 2 samples NO*
  - *Type=Burger: 2 samples YES and 2 samples NO*

Example - Root Selection Attribute Entropy (*Type*)

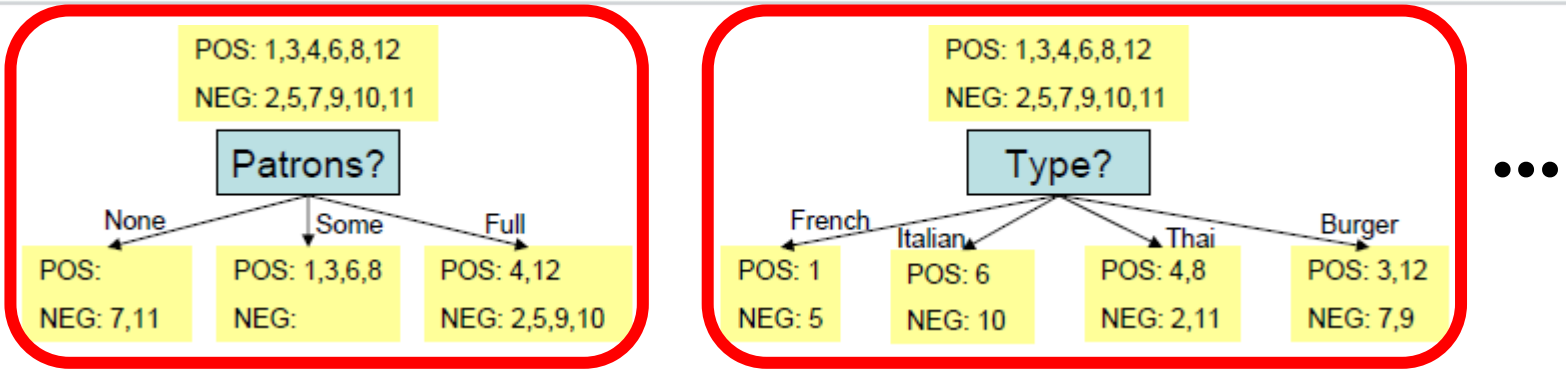


- **Example:** Attribute *Type*
  - **Average Entropy:** The weighted average over all sets resulting from the split
  - **Information Gain:** Also called "*the expected reduction in entropy*"

Example - Root Selection

Lancaster University

Patrons vs. *Type*



...

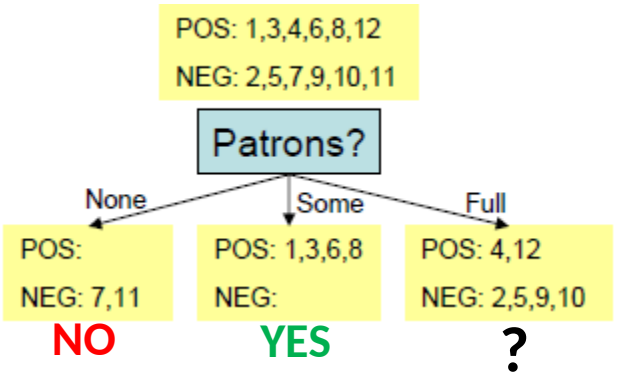
- Attribute *Patrons*

Patrons **has the highest** ***Information Gain*** **of all attributes**

- Attribute *Type*
  - **Information Gain:** Also called "*the expected reduction in entropy*"

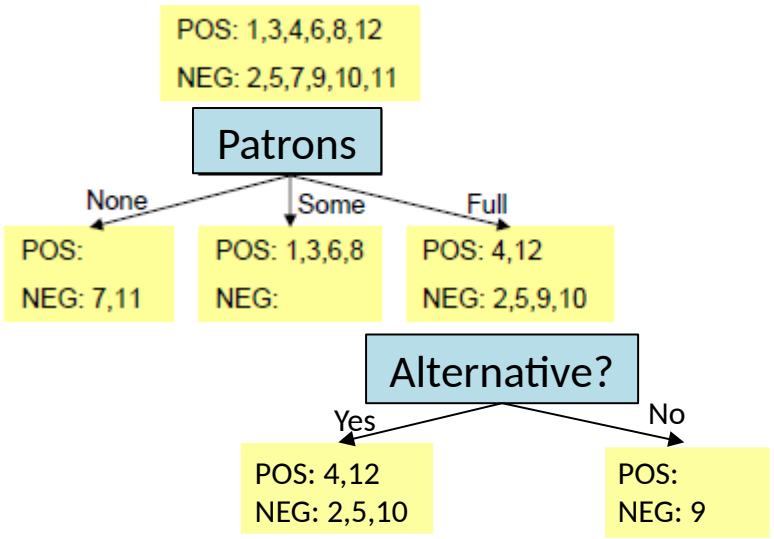
So choosing *Patrons* gains more information!

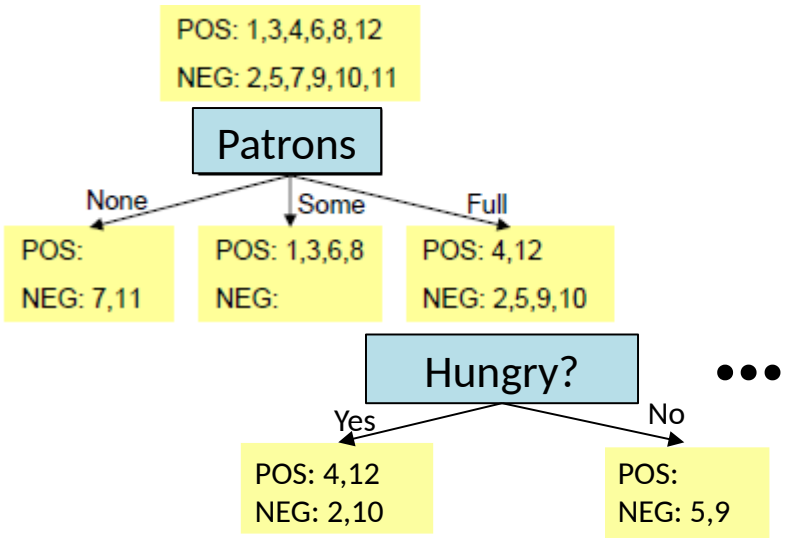
Learned Decision Tree (Root Node selected)



- After root node, we repeat process of "choosing an attribute to split on" for each sub-tree, until we get to leaf nodes that are completely classified to one class

Example – 2nd Level

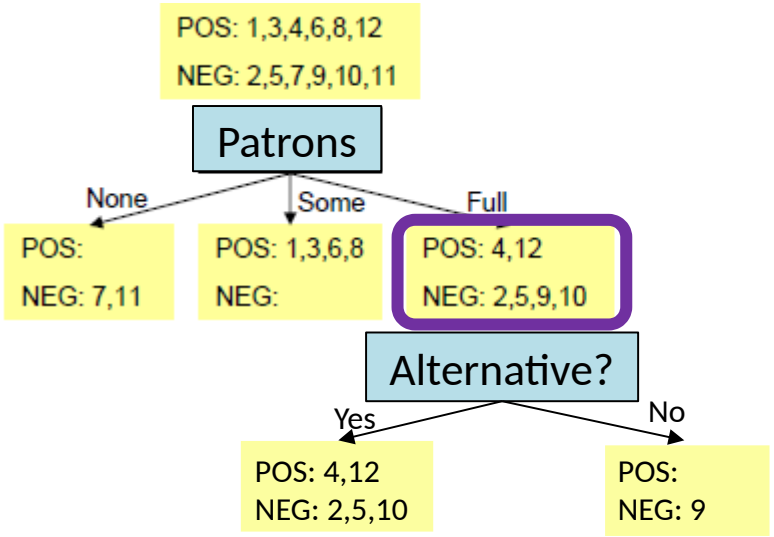


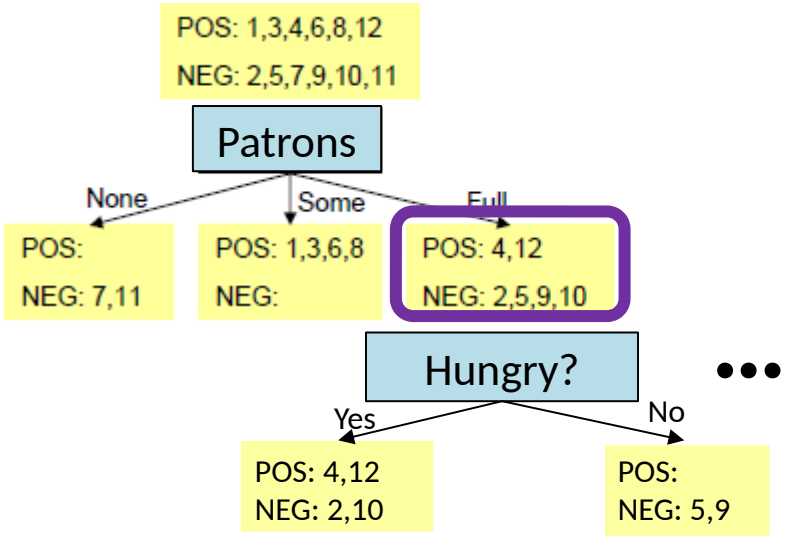


•••

- **Which attribute is better at the 2nd level?**

Example – 2nd Level Set Entropy

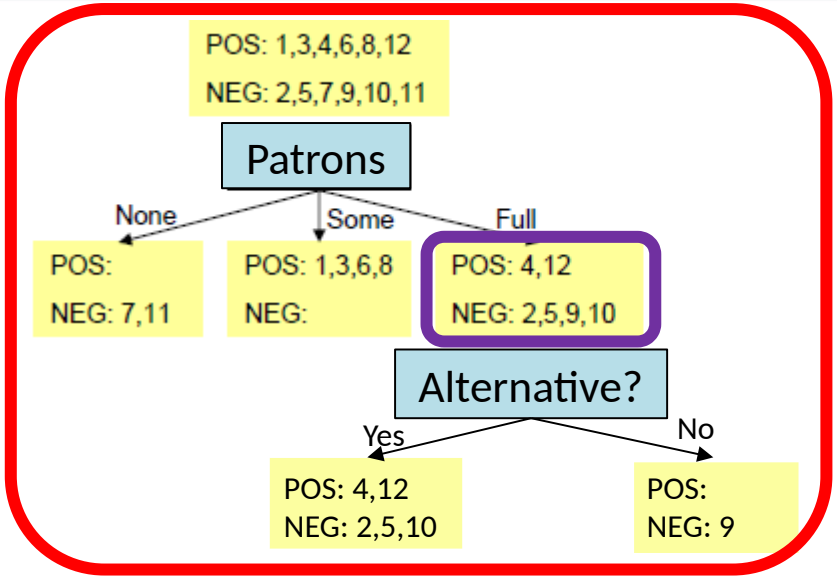


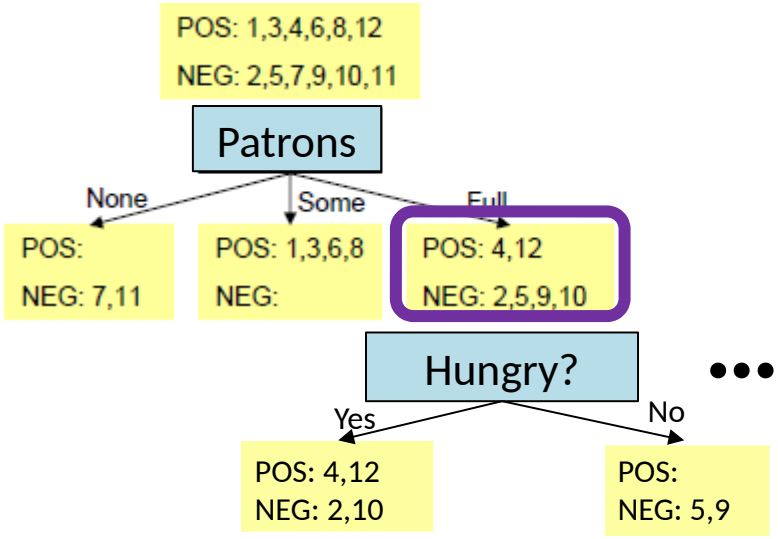


- For our Restaurant example:

$$E(S) = -P_T \log_2 P_T - P_F \log_2 P_F = -\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6} = 0.9183$$

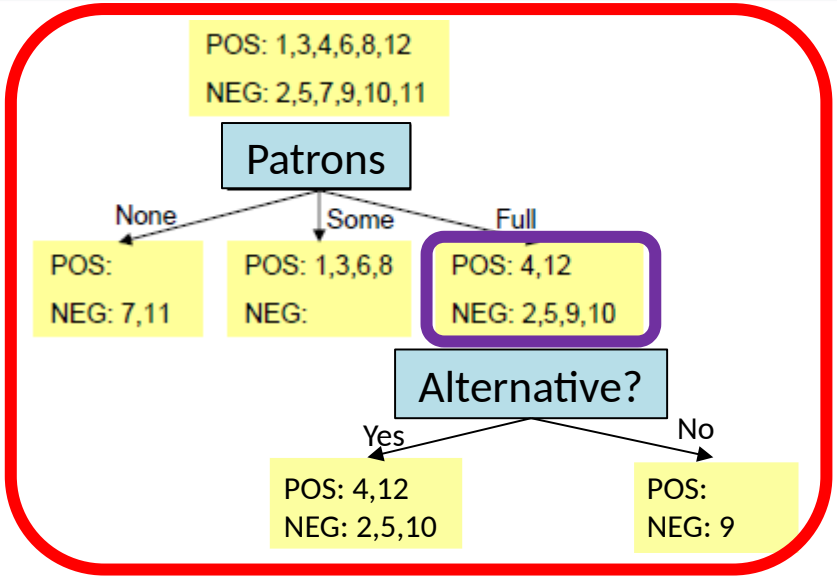
Example – 2nd Level Feature Entropy (*Alternative*)

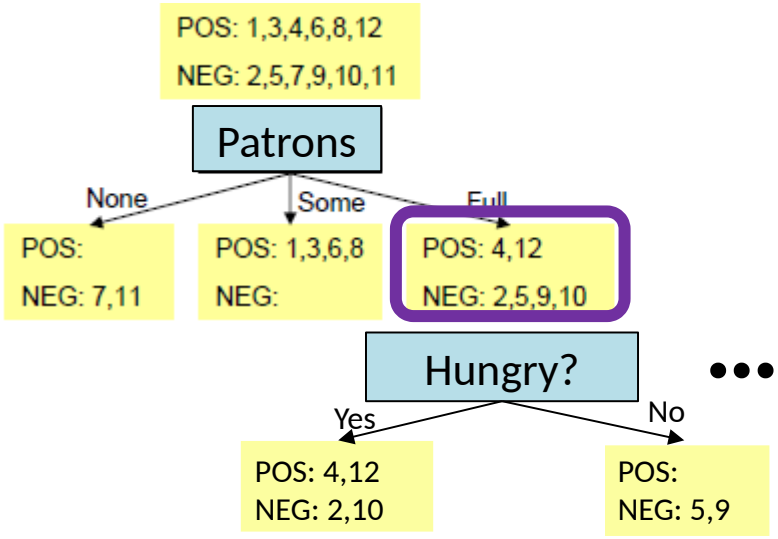




- **Example:** Attribute *Alternative*
  - *Alternative=Yes: 2 samples YES and 3 samples NO*
  - *Alternative=No: 0 samples YES and 1 samples NO*

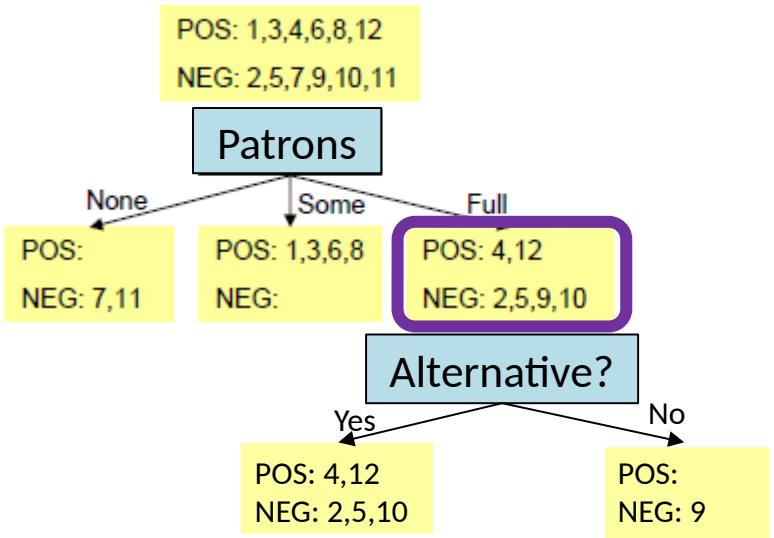
Example – 2nd Level Attribute Entropy (*Alternative*)

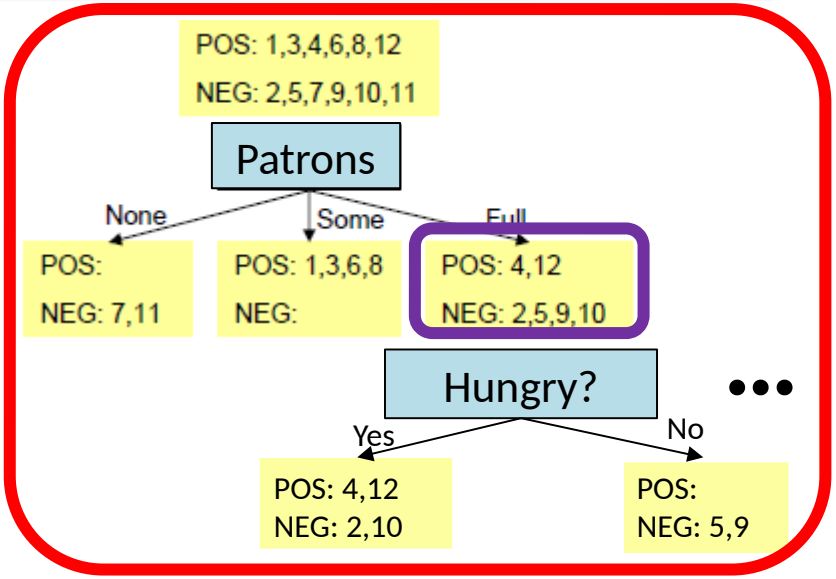




- **Example:** Attribute *Alternative*
  - **Average Entropy:** The weighted average over all sets resulting from the split
  - **Information Gain:** Also called "*the expected reduction in entropy*" is

Example – 2nd Level Attribute Entropy (*Hungry*)





- **Example:** Attribute *Hungry*
  - *Hungry=Yes: 2 samples YES and 2 samples NO*
  - *Hungry=No: 0 samples YES and 1 samples NO*

Example – 2nd Level Attribute Entropy (*Hungry*)

Lancaster University

POS: 1,3,4,6,8,12
NEG: 2,5,7,9,10,11

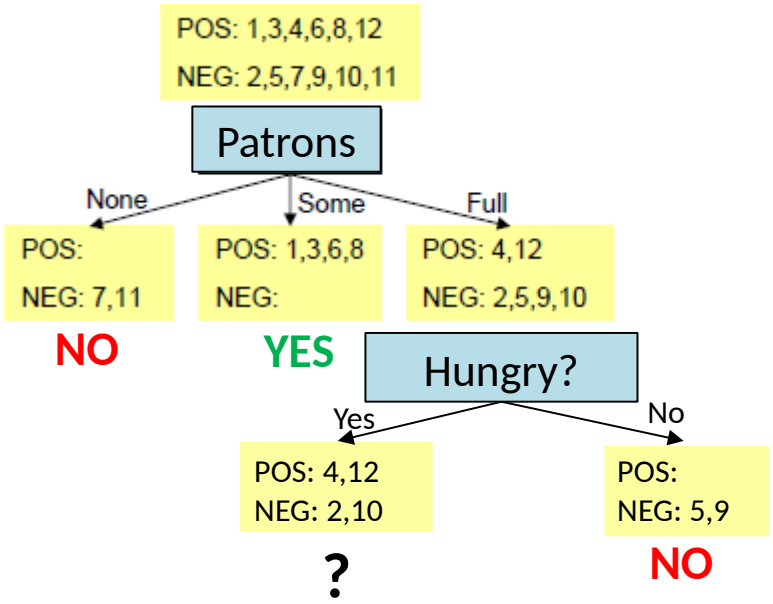
POS: 1,3,4,6,8,12
NEG: 2,5,7,9,10,11

So choosing *Hungry* gains more information!

Hungry has the highest *Information Gain* of all attributes

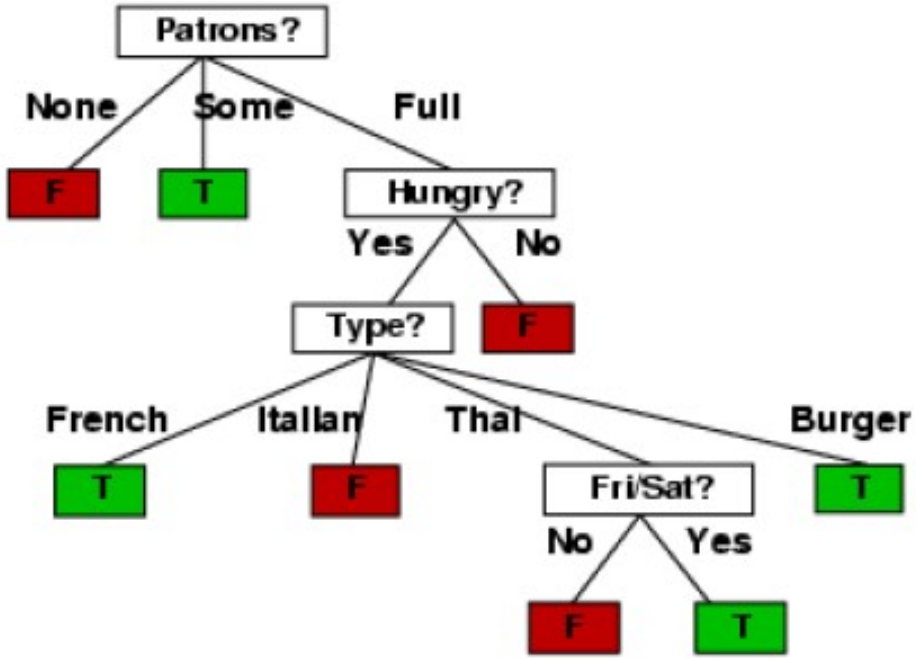
- **Information Gain:** Also called "*the expected reduction in entropy*" is

Learned Decision Tree (Second Node selected)



- We repeat process of "choosing an attribute to split on" for each sub-tree, until we get to leaf nodes that are completely classified to one class

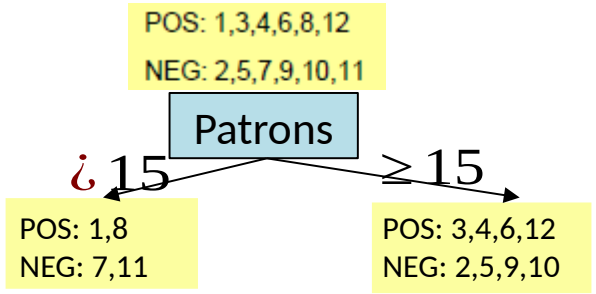
Learned Decision Tree (Final)



Continuous Attributes

- If attributes are continuous, internal nodes can test the value of an attribute against a **Threshold**
- **For example, what should we do if *Patrons* attribute is continuous?**

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	14	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	30	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	16	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	30	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	30	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	17	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	0	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	13	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	30	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	30	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	0	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	30	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$



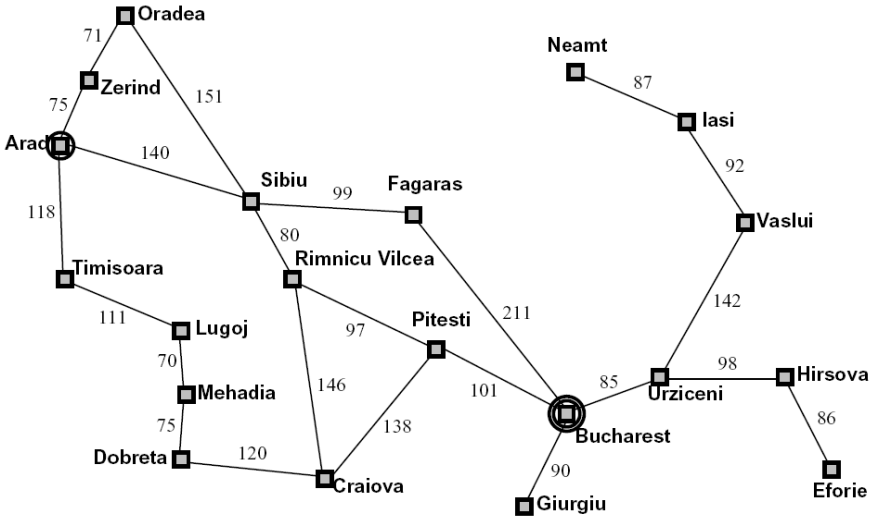
AI SCC361 Lecture 6

Dr. Hossein Rahmani
Senior Lecturer in Data Science
D17, InfoLab; e-mail: h.rahmani@lancaster.ac.uk

Genetic Algorithms

Search Algorithms (Example#1)

- A **genetic algorithm** is a search algorithm

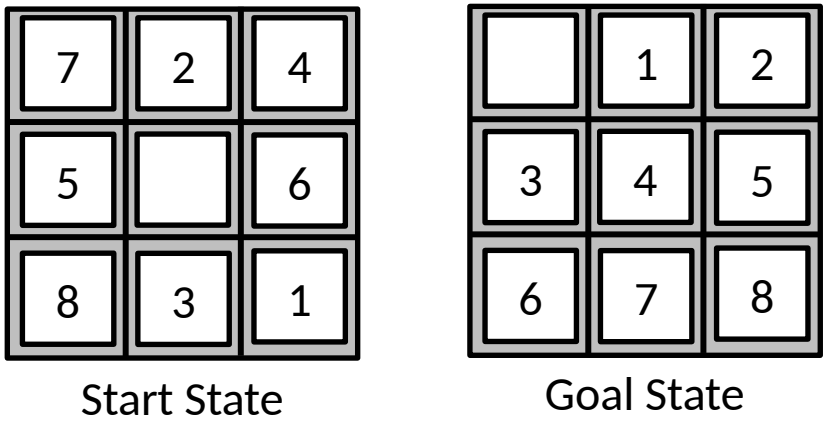


Shortest path problem

Search Algorithms (Example#2)

Lancaster University

- A **genetic algorithm** is a search algorithm

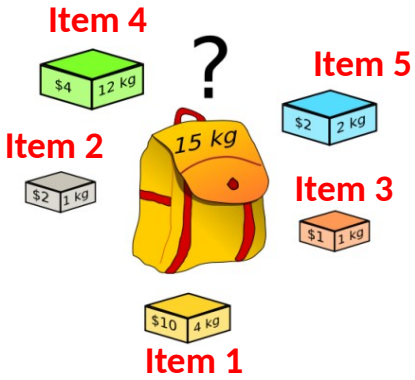


8-puzzle problem

Search Algorithms (Example#3)

- A **genetic algorithm** is a search algorithm

  - Set of items with given value and weight
  - The knapsack has given **capacity**
  - Select items to maximise the value of items in knapsack, but do not exceed total capacity
  - The problem is **NP-Complete**



0-1 Knapsack Problem (Week 6's Lab)

Search Algorithms (Example#4)

Lancaster University

- A **genetic algorithm** is a search algorithm
  - Is a famous touring problem
  - Set of cities with given distances between them
  - Traveling salesman must visit all cities
  - Each city must be visited **exactly once**
  - The aim is to find the **shortest path**, i.e. a sequence of cities to minimise travel distance
  - The problem is **NP-Complete**

Traveling salesman problem (Week 7's Lab)

Search Algorithms (Example#5)

Lancaster University

- A **genetic algorithm** is a search algorithm

$$f_1(x) = x_1 - 0.25428722 - 0.18324757\, x_4 x_3 x_9$$
$$f_2(x) = x_2 - 0.37842197 - 0.16275449\, x_1 x_{10} x_6$$
$$f_3(x) = x_3 - 0.27162577 - 0.16955071\, x_1 x_2 x_{10}$$
$$f_4(x) = x_4 - 0.19807914 - 0.15585316\, x_7 x_1 x_6$$
$$f_5(x) = x_5 - 0.44166728 - 0.19950920\, x_7 x_6 x_3$$
$$f_6(x) = x_6 - 0.14654113 - 0.18922793\, x_8 x_5 x_{10}$$
$$f_7(x) = x_7 - 0.42937161 - 0.21180486\, x_2 x_5 x_8$$
$$f_8(x) = x_8 - 0.07056438 - 0.17081208\, x_1 x_7 x_6$$
$$f_9(x) = x_9 - 0.34504906 - 0.19612740\, x_{10} x_6 x_8$$
$$f_{10}(x) = x_{10} - 0.42651102 - 0.21466544\, x_4 x_8 x_1$$

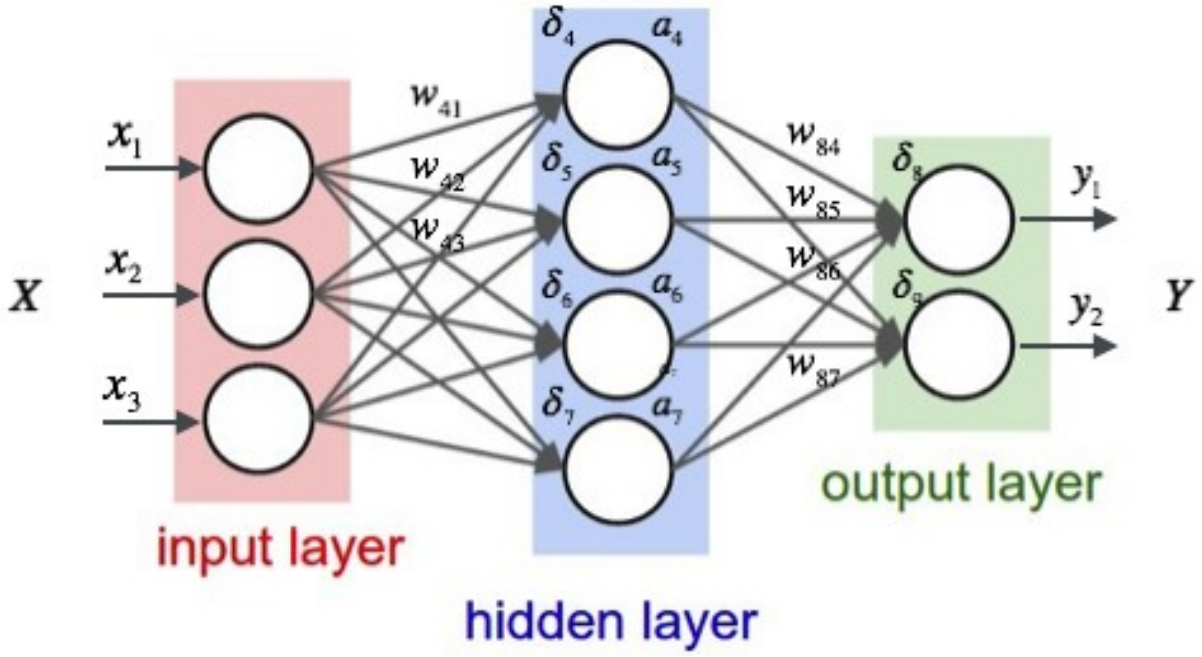
The problem has been considered as

$$\min(f_1(x), f_2(x), \ldots, f_n(x))$$

System of Nonlinear Equations

Search Algorithms (Example#6)

- A **genetic algorithm** is a search algorithm



Finding weights () of Neural Network models

Search Algorithms

Lancaster University

- There are several search algorithms
  - Uninformed Search Methods (operates in brute force-way)
    - Breadth-First Search
    - Depth-First Search
    - Uniform-Cost Search
    - Iterative deepening depth-first search
    - Uniform cost search
    - Bidirectional Search
  - Informed Search Methods (injecting Heuristic)
    - Best-first Search Algorithm (greedy)
    - A*

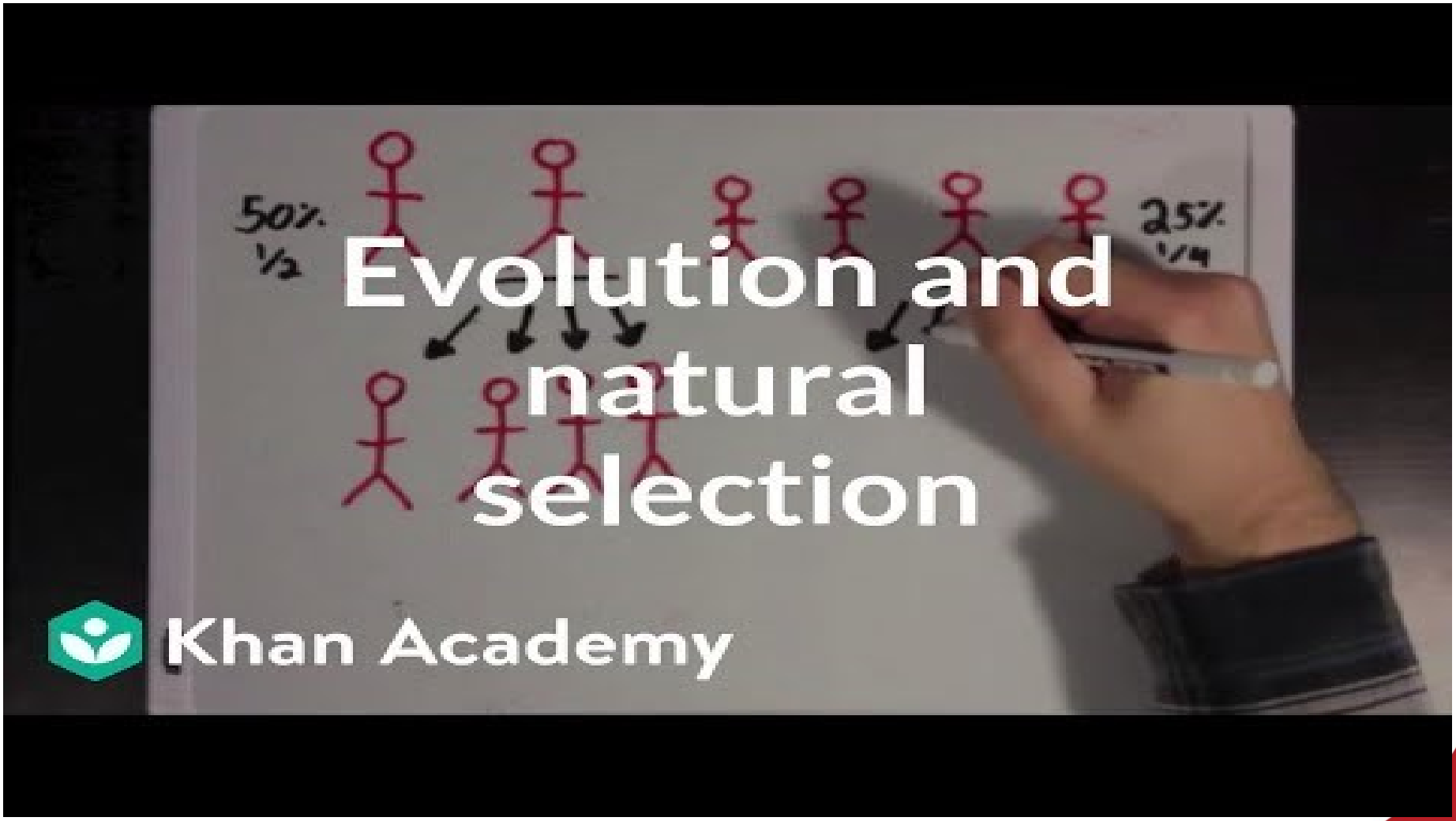
What are Genetic Algorithms?

Lancaster University

- A **genetic algorithm** is a guided random search strategy
- Inspired by Darwin's theory of natural evolution
- The fittest survives, and has a greater chance of breeding and passing forward its genetic information
- Genetic Algorithms are good at taking **large, potentially huge search spaces** and navigating them, looking for optimal solution

Darwin's Theory of Natural Evolution (Video#1)

Darwin's Theory of Natural Evolution (Video#2)



Notion of Natural Selection

Lancaster University

- The process of natural selection starts with the selection of fittest individuals from a population.
- They produce offspring/children which inherit the characteristics of the parents and will be added to the next generation.
- If parents have better fitness, their offspring/children will be better than parents and have a better chance at surviving.
- This process keeps on iterating and at the end, a generation with the fittest individuals will be found.
- This notion can be applied for **a search problem**. We consider a set of solutions for a problem and select a set of best ones out of them.

Outline of the Basic Genetic Algorithm

Lancaster University

1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

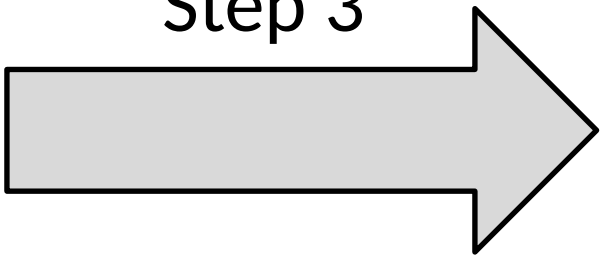
Outline of the Basic Genetic Algorithm (Cont.)

Lancaster University

Current population 8 Chromosomes

Chromosome#1	90
Chromosome#2	20
Chromosome#3	40
Chromosome#4	40
Chromosome#5	10
Chromosome#6	80
Chromosome#7	60
Chromosome#8	50

Iterating through Step 3



New population 8 Chromosomes

Chromosome#9	40
Chromosome#10	50
Chromosome#11	60
Chromosome#12	50
Chromosome#13	20
Chromosome#14	90
Chromosome#15	90
Chromosome#16	90

Outline of the Basic Genetic Algorithm (Cont.)

Lancaster University

Step 4: [Replace] Generate a new population for a further run of algorithm

Current population 8 Chromosomes		**Mixed population** 8 Chromosomes		**New population** 8 Chromosomes	
Chromosome#1	90	Chromosome#6	80	Chromosome#9	40
Chromosome#2	20	Chromosome#7	60	Chromosome#10	50
Chromosome#3	40	Chromosome#1	90	Chromosome#11	60
Chromosome#4	40	Chromosome#10	50	Chromosome#12	50
Chromosome#5	10	Chromosome#11	60	Chromosome#13	20
Chromosome#6	80	Chromosome#14	90	Chromosome#14	90
Chromosome#7	60	Chromosome#15	90	Chromosome#15	90
Chromosome#8	50	Chromosome#16	90	Chromosome#16	90

Outline of the Basic Genetic Algorithm (Cont.)

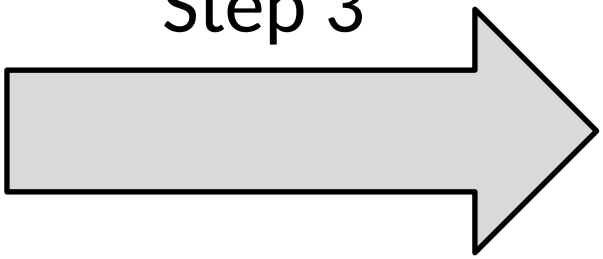
Lancaster University

Step 5 and 6: If the end condition is not satisfied, go to Step 3

Current population
8 Chromosomes

Chromosome#6	80
Chromosome#7	60
Chromosome#1	90
Chromosome#10	50
Chromosome#11	60
Chromosome#14	90
Chromosome#15	90
Chromosome#16	90

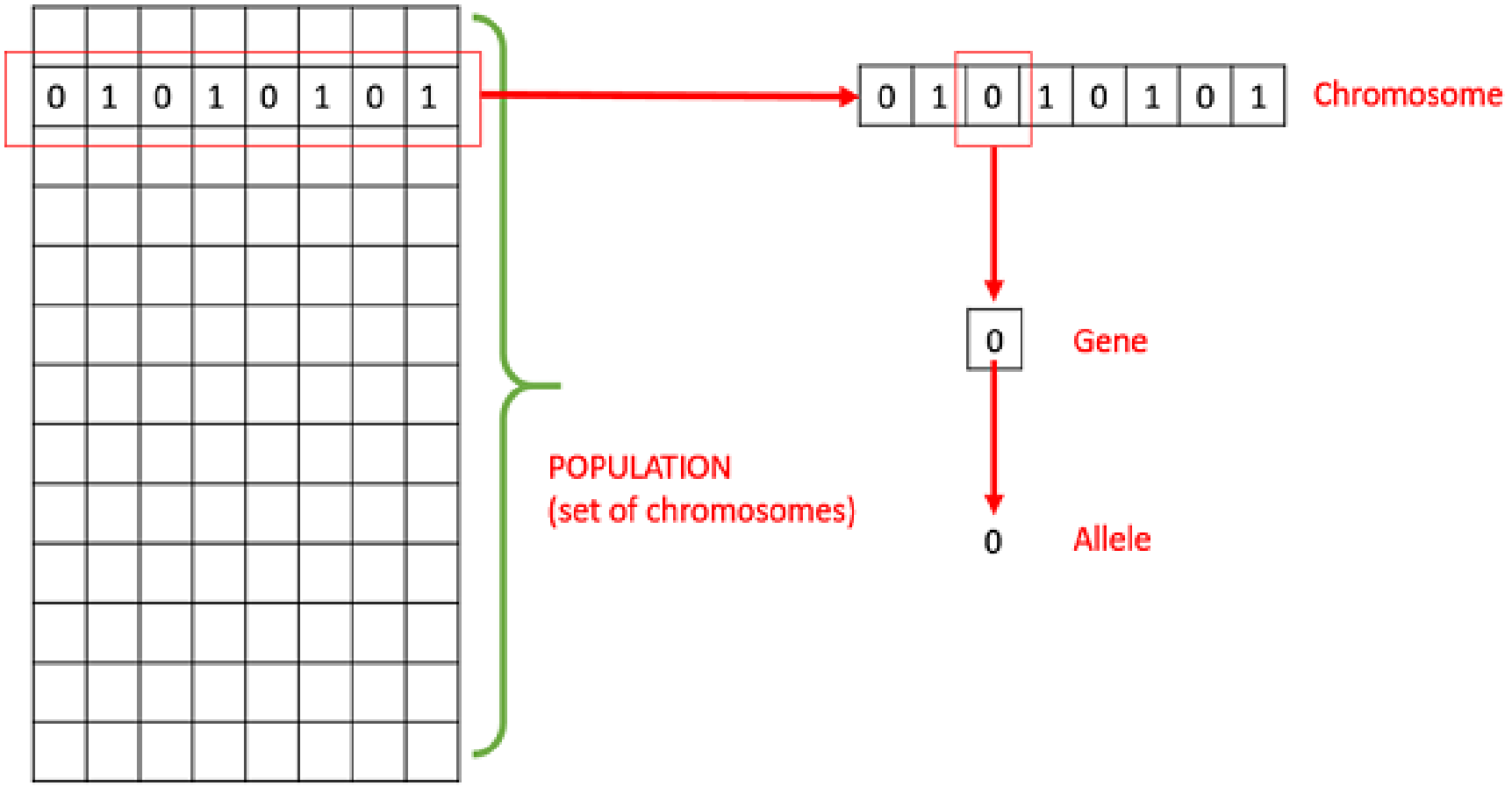
Iterating through Step 3



Chromosome, genes, alleles

- What is a population?
  - A set of chromosomes is referred to as **population**
- What is a chromosome?
  - The string, which is a candidate solution to the search problem, is referred to as ***chromosome/individual***
  - The chromosome should in some way contain information about the solution which it represents.
- What are genes?
  - A chromosome is characterized by a set of parameters (variables) known as ***Genes***.
- What are alleles?
  - The values of genes are called ***alleles***

Chromosome vs genes vs alleles (Example)



Chromosome Encoding

Lancaster University

- Popular encoding methods include:
  - Binary encoding
  - Value encoding
  - Permutation encoding
  - Tree encoding

Binary Encoding

- Binary encoding is the most common to represent chromosomes.
- It was first used because of its relative simplicity.
- Each chromosome is a string of bits: **0** or **1**

Chromosome 1:

1	1	0	0	1	1	0	1	0	1	0	0	0	1

Chromosome 2:

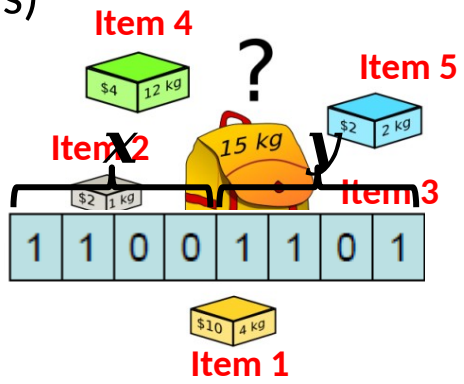
0	1	1	0	1	0	1	0	1	1	1	1	0	1

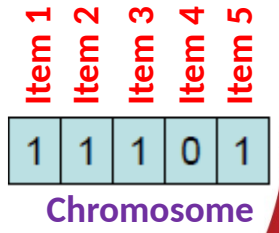
- Meaning of the string can vary
  - Each bit can represent a characteristic of the solution
  - The whole string can represent number(s)

Example: **0-1 Knapsack Problem**

- Set of items with given value and size
- The knapsack has given capacity
- Select items to maximise the value of items in knapsack, do not exceed total capacity
- Each bit says if the corresponding item is in knapsack

$$\min_{x,y} f(x,y) = 3x^5 - 12y^3 + 8xy$$





Value Encoding

- The value encoding can be used in problems where values such as real numbers are used.
- Every chromosome is *a string of some values*
- Values can be anything connected to the problem
  - Numbers

1.31	2.27	9.6	2.21	7.03	6.22

  - Characters

A	G	W	V	J	U	K	R	B

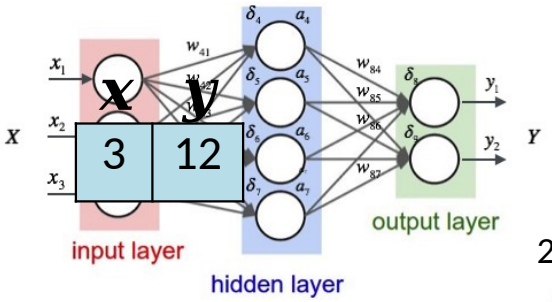
  - Objects

←	→	↓	←	↓	←	↑	←	↑

Example: **Finding weights for a neural network**

$$\min_{x,\,y} f(x, y) = 3x^5 - 12y^3 + 8xy$$

- Given a neural network with a specific architecture
- Find weights to train the network to a desired output
- Real values in chromosomes represent corresponding weights for inputs



Permutation Encoding

Lancaster University

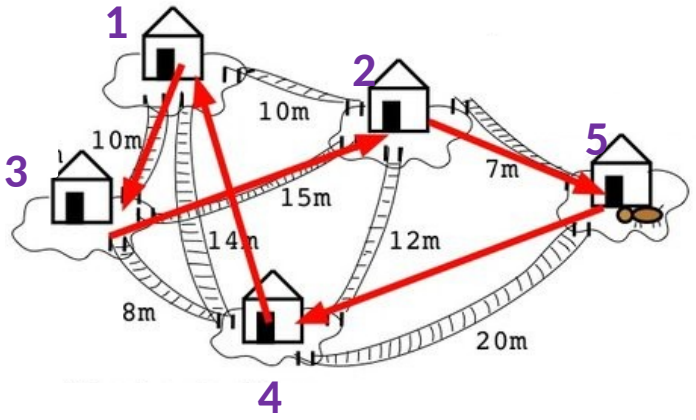
- The permutation encoding can be used in ordering problems such as Travelling Salesman Problem (TSP) or Task Ordering problem.
- Every chromosome is **a string of numbers,** which represent numbers in a sequence
- **The values must not be repeated in a single chromosome**

Example: **Travelling Salesman Problem**

- Is a famous touring problem
- Set of cities with given distances between them
- Traveling salesman must visit all cities
- Chromosome gives order of cities to visit

1	3	2	5	4

Chromosome



- Each city must be visited **exactly once**
- The aim is to find the shortest path, i.e. a sequence of cities to minimise travel distance
- The problem is **NP-Complete**

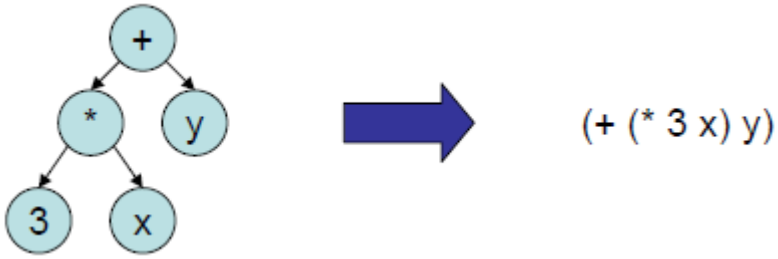
Tree Encoding

Lancaster University

- Tree encoding is mainly used for evolving programs or expressions.
- Every chromosome is **a tree of some objects,** such as functions or commands in programming language.

Example: **Finding a function from given values**

- Some input and output values are given
- Task is to find a function, which will give the best (closest to desired) output to all inputs
- Chromosome are functions represented in a tree



Outline of the Basic Genetic Algorithm

Lancaster University

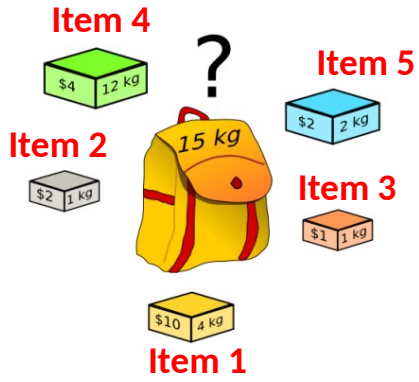
1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

Generating Random Population (Knapsack)

Lancaster University

Example#1: **0-1 Knapsack Problem**

- Set of items with given value and weight
- The knapsack has given capacity
- Select items to maximise the value of items in knapsack, but do not exceed total capacity
- Each bit says if the corresponding item is in knapsack



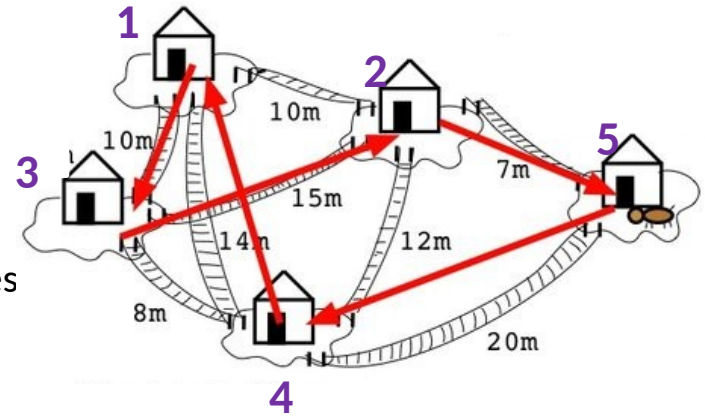
	Item 1	Item 2	Item 3	Item 4	Item 5
Chromosome 1	1	1	1	0	1
Chromosome 2	0	0	1	0	1
Chromosome 3	1	0	1	0	0
⋮		⋮			
Chromosome n	0	0	1	0	0

Number of chromosomes () is a parameter!

Generating Random Population (TSP)

Example#2: **Travelling Salesman Problem**

- Traveling salesman must visit all cities
- Chromosome gives order of cities to visit
- Each city must be visited **exactly once**
- The aim is to find the shortest path, i.e. a sequence of cities to minimise travel distance



	City 1	City 2	City 3	City 4	City 5
Chromosome 1	1	2	3	4	5
Chromosome 2	1	3	5	4	2
Chromosome 3	3	1	2	4	5
⋮	⋮		⋮		
Chromosome n	5	4	2	1	3

Number of chromosomes () is a parameter!

Outline of the Basic Genetic Algorithm

Lancaster University

1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

Fitness Function

Lancaster University

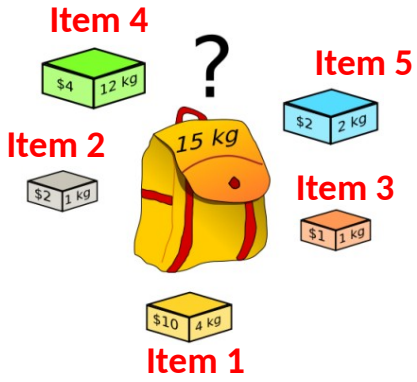
- If the correct answer is known, the fitness is some distance metric toward the correct answer
- If the correct answer is unknown, fitness must be an estimator of the value of the solution
- Combinations of multiple goals into a single numeric function can be difficult
- Evolution can only be as good as the fitness function
- Each problem has its own fitness function
- **Generic Requirements of a Fitness Function**
  - The fitness function should be **clearly defined**.
  - The fitness function should be implemented **efficiently**. (Not to be the bottleneck of the algorithm)
  - The fitness function should generate **intuitive results**. The best/worst candidates should have best/worst score values.

Fitness Function (Example#1)

Lancaster University

Example#1: **0-1 Knapsack Problem**

- Set of items with given value and weight
- The knapsack has given capacity
- Select items to maximise the value of items in knapsack, but do not exceed total capacity
- Each bit says if the corresponding item is in knapsack



Item 1	Item 2	Item 3	Item 4	Item 5
1	1	1	0	1

Chromosome 1

Weight = 4kg+1kg+1kg+0+2kg=8kg

Profit = $10+$2+$1+0+$2=$15

We define fitness as profit value

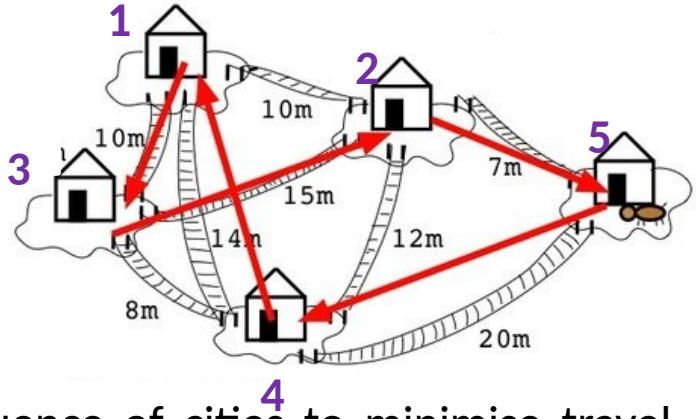
Item 1	Item 2	Item 3	Item 4	Item 5
0	1	1	0	1

Chromosome 2

Weight = 0+1kg+1kg+0+2kg=4kg

Profit = 0+$2+$1+0+$2=$5

Fitness Function (Example#2)

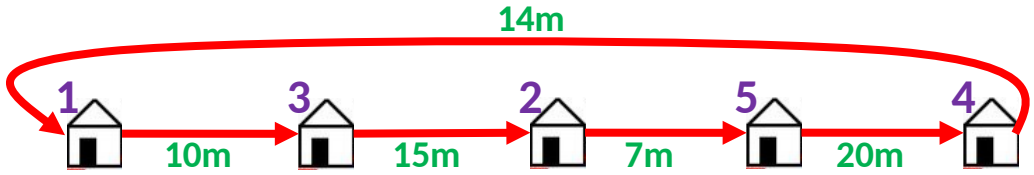


Example: **Travelling Salesman Problem**

- Set of cities with given distances between them
- Traveling salesman must visit all cities
- Chromosome gives order of cities to visit
- Each city must be visited **exactly once**
- The aim is to find the shortest path, i.e. a sequence of cities to minimise travel distance

1	3	2	5	4

Chromosome 1



Distance = 10m+15m+7m+20m+14m=66m Fitness= 1/66 OR $1/(66)^2$

∞

1	2	3	4	5

Chromosome 2

Distance = 10m+15m+8m+20m+= Fitness= 1/ = 0

We define fitness as:

Outline of the Basic Genetic Algorithm

Lancaster University

1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

Selection

Lancaster University

- The first genetic operation in the reproductive phase.
- The objective of selection is to choose the **fitter individuals** in the population that will create individuals for the next generation.

- Selection procedures can be broadly classified into two classes:
  - **Ordinal Selection**
    - Tournament selection (This Lecture)
    - Linear rank selection
    - Truncation selection
  - **Fitness Proportionate Selection**
    - Roulette wheel selection (This Lecture)
    - Stochastic universal selection

Tournament Selection

Lancaster University

- In tournament selection chromosomes are randomly selected from the population.
  - is *tournament size* that refers to number of randomly selected individuals.
  - The most widely used value of is 2.
- Tournament selection selects *the best chromosome* from this group (tournament).
- There are two variations:
  - With replacement
  - Without replacement

❑ Imagine that we are picking individuals from a bag.

- **With replacement:** After picking chromosomes, we ***put them back*** in the bag.
- **Without replacement:** After picking chromosomes, we ***don't put the winner back*** in the bag.

Tournament Selection (Example#1)

Lancaster University

- Consider a population with 5 chromosomes/individuals (), with fitness values as shown in the table below.

Chromosome #	1	2	3	4	5
Fitness ()	28	18	14	9	26

- **With replacement:** After picking chromosomes, we put them back in the bag.

1st run

Chromosome #	2
Fitness ()	18

Chromosome #	5
Fitness ()	26

Which chromosome is better? Chromosome 5

2nd run

Chromosome #	2
Fitness ()	18

Chromosome #	3
Fitness ()	14

Which chromosome is better? Chromosome 2

Get married

Tournament Selection (Example#2)

Lancaster University

- Consider a population with 5 chromosomes/individuals (), with fitness values as shown in the table below.

Chromosome #	1	2	3	4	5
Fitness ()	28	18	14	9	26

- **Without replacement:** After picking chromosomes, we don't put them back in the bag.

1st run

Chromosome #	2
Fitness ()	18

Chromosome #	5
Fitness ()	26

Which chromosome is better?

2nd run

Chromosome #	1
Fitness ()	28

Chromosome #	3
Fitness ()	14

Which chromosome is better?

Tournament Selection (issues)

Lancaster University

- If the tournament size is larger, weak individuals have a smaller chance to be selected. Why?

- **Efficient to code (unlike Fitness Proportionate Selection methods)**

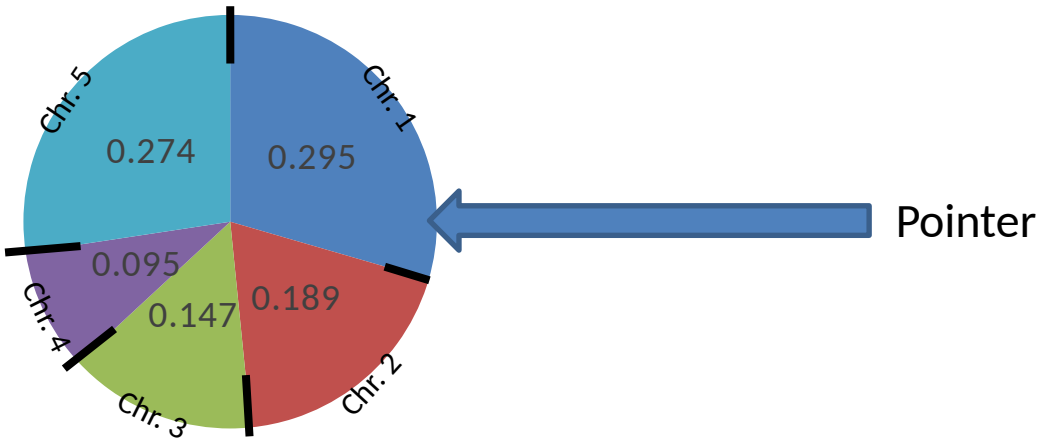
- **Works on parallel architectures**

Lancaster University

Questions?

Roulette Wheel Selection

- Roulette wheel is one of the simplest and traditional stochastic selection approach proposed by Holland.
- It selects the Chromosomes based on a probability **proportional to the fitness**.
- All the chromosomes (individuals) in the population are placed on the roulette wheel according to their fitness value
  - the bigger the value is, the larger the segment is



Roulette Wheel Selection

Lancaster University

- The chromosome corresponding to the segment on which roulette wheel stops, is selected.
- The process is repeated until the desired number of Chromosomes is selected
- Chromosomes with higher fitness have more probability of selection.

Roulette Wheel Selection (Example)

- Consider a population with 5 chromosomes/individuals (), with fitness values as shown in the table below.

Chromosome #	1	2	3	4	5
Fitness ()	28	18	14	9	26

- Compute the *probability*, , of selecting each member of the population:

$$\sum_{j=1}^{n} f_j = 28+18+14+9+26 = 95$$

$$p_1 = \frac{28}{95} = 0.295 \quad p_2 = \frac{18}{95} = 0.189 \quad p_3 = \frac{14}{95} = 0.147 \quad p_4 = \frac{9}{95} = 0.095 \quad p_5 = \frac{26}{95} = 0.274$$

Chromosome #	1	2	3	4	5
Fitness ()	28	18	14	9	26
Probability ()	0.295	0.189	0.147	0.095	0.274

Roulette Wheel Selection (Example, cont.)

Lancaster University

- Consider a population with 5 chromosomes/individuals (), with fitness values as shown in the table below.

Chromosome #	1	2	3	4	5
Fitness ()	28	18	14	9	26
Probability ()	0.295	0.189	0.147	0.095	0.274

- Compute the *cumulative probability*, , of each member of the population:

Chromosome #	1	2	3	4	5
Fitness ()	28	18	14	9	26
Probability ()	0.295	0.189	0.147	0.095	0.274
Cumulative Probability ()	0.295	0.484	0.631	0.726	1.000

Roulette Wheel Selection (Example, cont.)

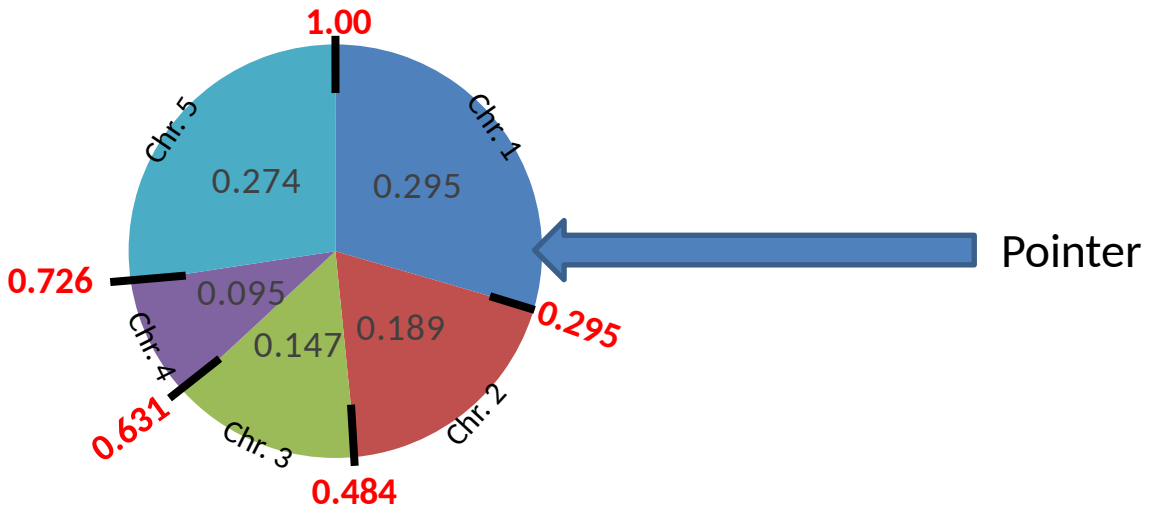
Lancaster University

- Consider a population with 5 chromosomes/individuals (), with fitness values as shown in the table below.

Chromosome #	1	2	3	4	5
Fitness ()	28	18	14	9	26
Probability ()	0.295	0.189	0.147	0.095	0.274
Cumulative Probability ()	0.295	0.484	0.631	0.726	1.000

- **Final step:** *a uniform random number*, , where , is generated, say , then the 3rd chromosome is selected.
- This step is repeated to select parents (usually) selected

Roulette Wheel Selection (Illustration)



- **Implementation:** *a uniform random number*, , where , is generated, say , then the 3rd chromosome is selected.
- This step is repeated to select candidate (usually)

Outline of the Basic Genetic Algorithm

Lancaster University

1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

Crossover

Lancaster University

- Also called **Recombination**
- After selection, chromosomes are recombined (or crossed over) to create new, hopefully better, chromosomes.
- Crossover is performed with a **high probability (e.g. 0.8)**
- Many of the crossover operators used in the literature are *problem-specific*
- But, in this section we will introduce a few generic crossover operators including:
  - *k*-point Crossover (**Today**)
  - Uniform Crossover (**Today**)
  - Uniform Order-Based Crossover
  - Order-Based Crossover
  - Partially Matched Crossover (PMX)
  - Cycle Crossover (CX)

Crossover Probability

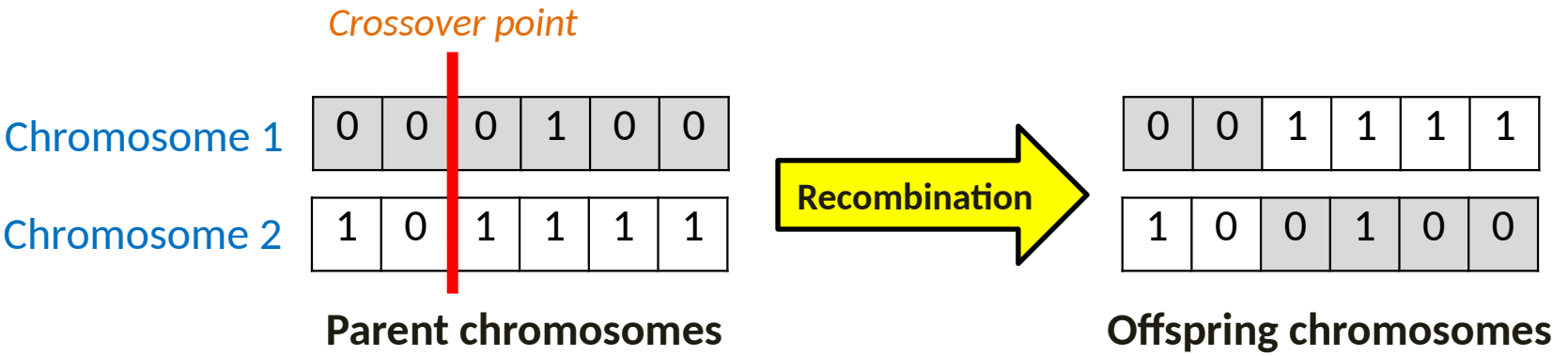
Lancaster University

- In most recombination operators, two individuals are recombined with a probability , called the *crossover probability*.
- The value of can be set *experimentally*
- A uniform random number, , is generated
  - If , the two randomly selected individuals undergo recombination.
  - If If , the two offspring are simply copies of their parents.

k-point Crossover (*k*=1)

- **1-point** and **2-point** crossovers are the simplest and most widely applied crossover methods.

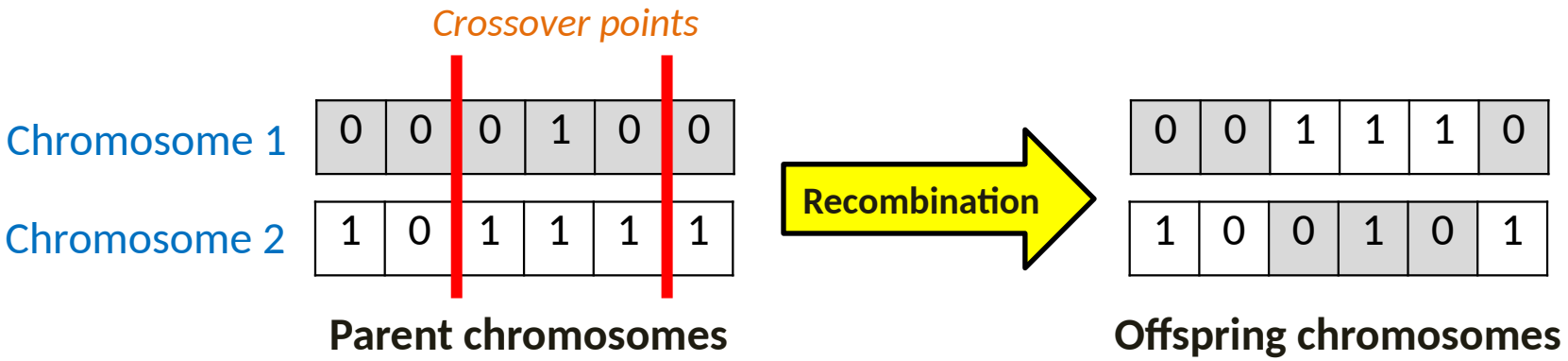
- Example 1: **1-point crossover**
  - Two parents are randomly selected
  - *Crossover point* is generated **randomly**



k-point Crossover (*k*=2)

- **1-point** and **2-point** crossovers are the simplest and most widely applied crossover methods.

- Example 2: **2-point crossover**
  - Two parents are randomly selected
  - *Crossover points* are generated **randomly**

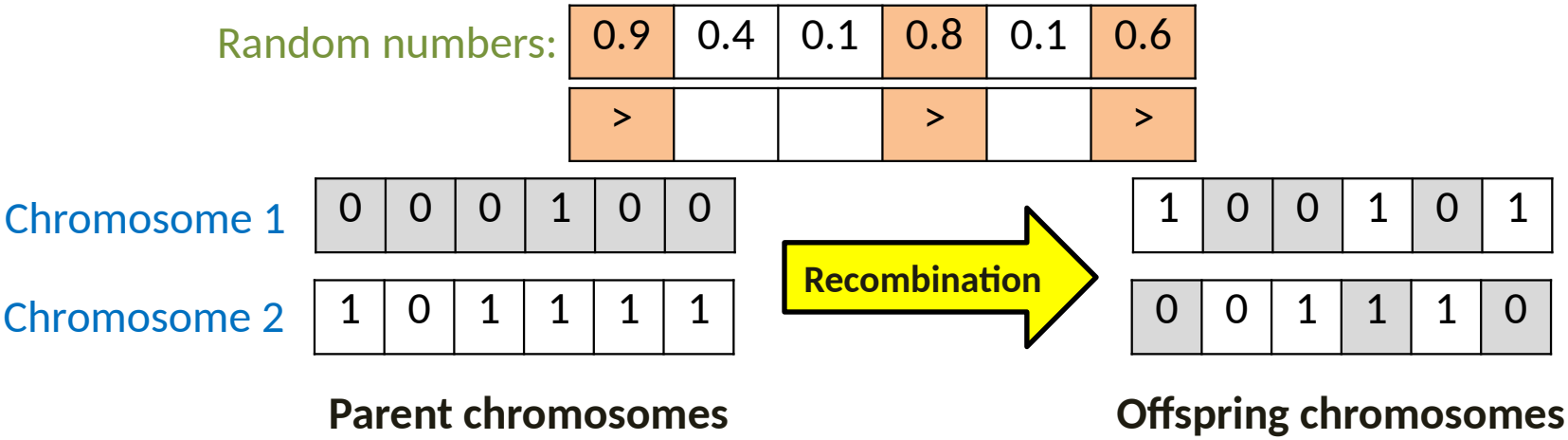


Uniform Crossover

- Another common recombination operator is uniform crossover.
- For each position, exchange alleles with a given probability, known as the *swapping probability* (typically 0.5).

❑ Example: **Uniform crossover**

- *6 numbers (equal to length of chromosome) are* generated **randomly**
- **Values >= swapping probability**: **allele is swapped**
- **Values < swapping probability**: **allele is not swapped**



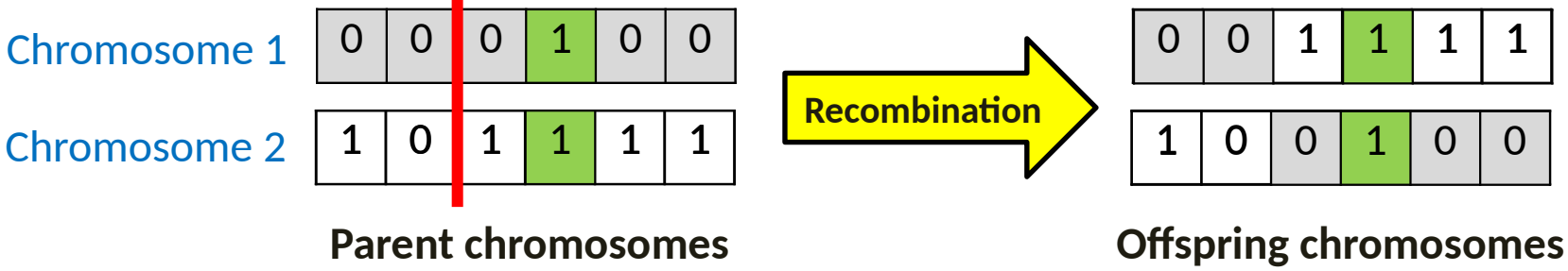
Outline of the Basic Genetic Algorithm

Lancaster University

1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

Mutation

- If we use a crossover operator, such as one-point crossover, we may get better and better chromosomes but the problem is:
  - if the two parents (or worse, the entire population) have the same allele at a given gene then one-point crossover will not change that.
  - **That gene will have the same allele forever**

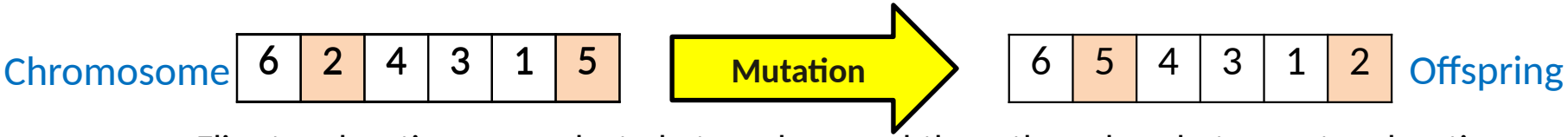


- Mutation is designed to overcome this problem in order to add **diversity to the population** and ensure that it is possible to explore the entire search space.
- Mutation can prevent problems with local minima
- Mutation is performed with a **low probability (e.g. 0.2)**

Mutation (Example#1)

- One of the most common mutations is the **bit-flip** mutation for binary encoding.

- Mutation with Permutation Encoding
  - Swap: two locations are selected at random, and then, their values are exchanged

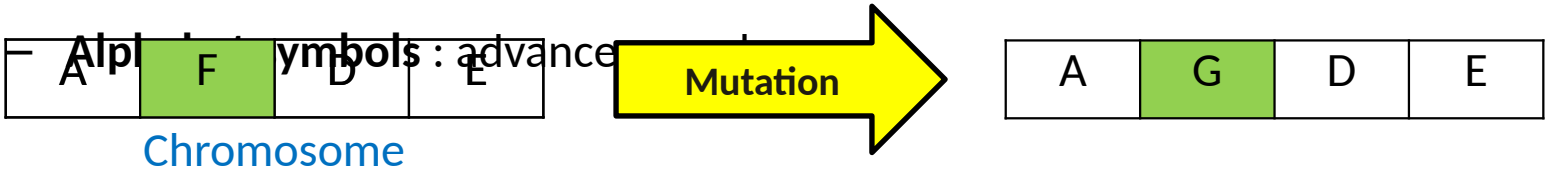


  - Flip: two locations are selected at random, and then, the values between two locations are simply flipped

Mutation (Example#2)

- Mutation with Value Encoding
  - **Real numbers** : add/subtract a small random number

  - **Alphabet symbols** : advance



Outline of the Basic Genetic Algorithm

Lancaster University

1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

Replacement Strategies

Lancaster University

- When creating a new population by crossover and mutation, there is a good chance that the best chromosome will be lost

- Generational
  - All population members are removed on each generation

- Elitism
  - Complete population is replaced except for the best chromosome (or 2 best chromosomes) of each generation which is carried over to next generation without modification.

- Random Replacement
  - In Random Replacement, the children replace two randomly chosen individuals in the population.

Outline of the Basic Genetic Algorithm

Lancaster University

1. [Start] Generate random population of chromosomes
2. [Fitness] Evaluate the fitness of each chromosome
3. [New population] Create a new population by repeating:
   A. [Selection] Select two parent chromosomes based on their fitness
   B. [Crossover] With a crossover probability, cross over the parents to form new offspring (child). If no crossover is performed, offspring is an exact copy of parents.
   C. [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome).
   D. [Accepting] Place new offspring in a new population
   E. [Fitness] Evaluate the fitness of each chromosome
4. [Replace] Generate a new population for a further run of algorithm
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population
6. [Loop] Go to step 3

End Condition

Lancaster University

- The population converges when either 90% of the chromosomes in the **population have the same fitness value**
- Or the **number of generations** is greater than a fixed number
- Or the **average fitness value** of population remains fixed for several iterations.

Parameters of a GA

Lancaster University

- Population size
- Encoding choices
- Crossover probability
- Mutation probability
- Replacement Strategies
- Fitness function
- End condition

Advantages and Disadvantages

Lancaster University

- **Advantages**
  - Parallelism
  - Less likely to get stuck in local extrema than some other optimisation methods
  - Relatively easy to implement
- **Disadvantages**
  - Choosing an encoding and fitness function can be difficult
  - Computational time
  - No guarantee on a solution
  - Strong dependence on parameters

Lancaster University

Questions?

Lancaster University

AI SCC361 Week 8

Dr. Hossein Rahmani
Senior Lecturer in Data Science
D17, InfoLab; e-mail: h.rahmani@lancaster.ac.uk

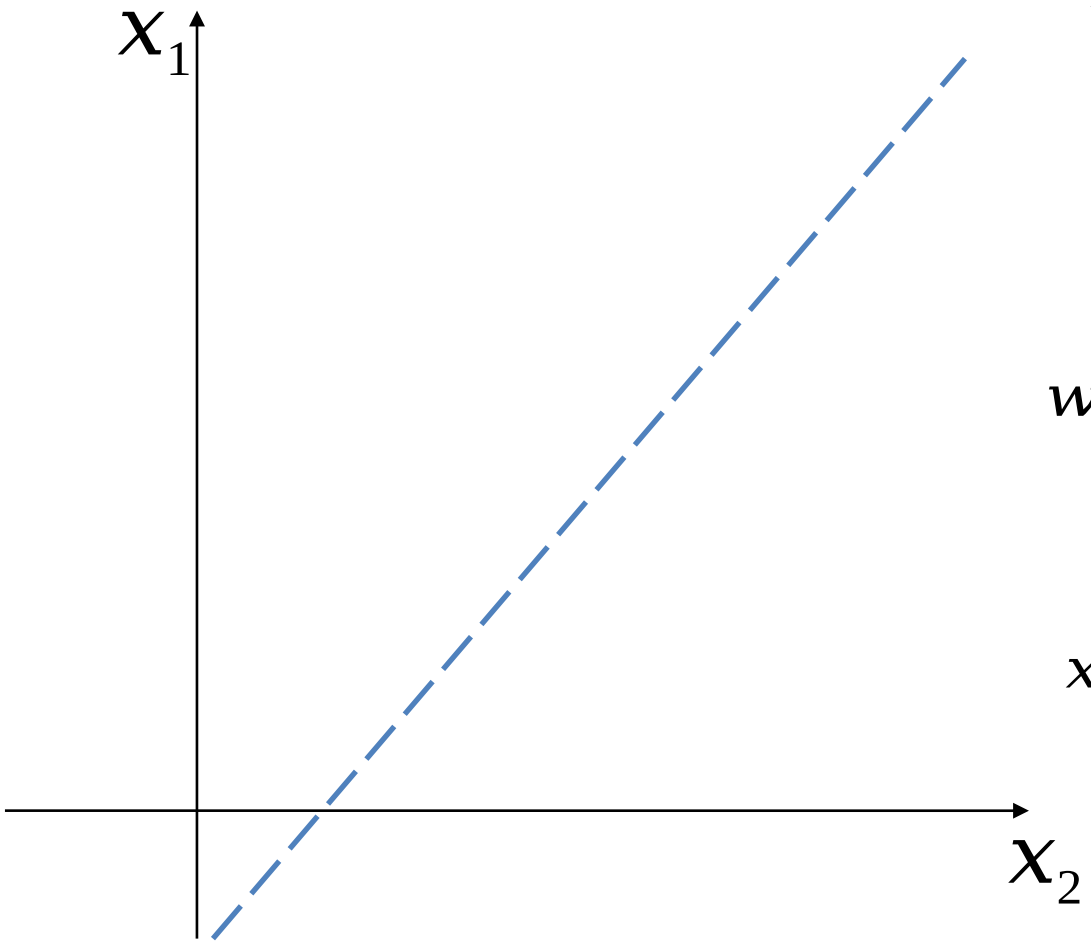
Neural Networks

Outline

Lancaster University

- Decision Surface
- Perceptron
  - Binary classification
- Discrete vs Continuous Perceptron
- Multi-class classification
  - Softmax function
- Error/Loss functions
  - Mean Square Error (MSE)
  - Maximum Likelihood (ML)
  - Cross-Entropy (CE)
- Multi Layer Perceptron (MLP)
- Backpropagation

Line Equation



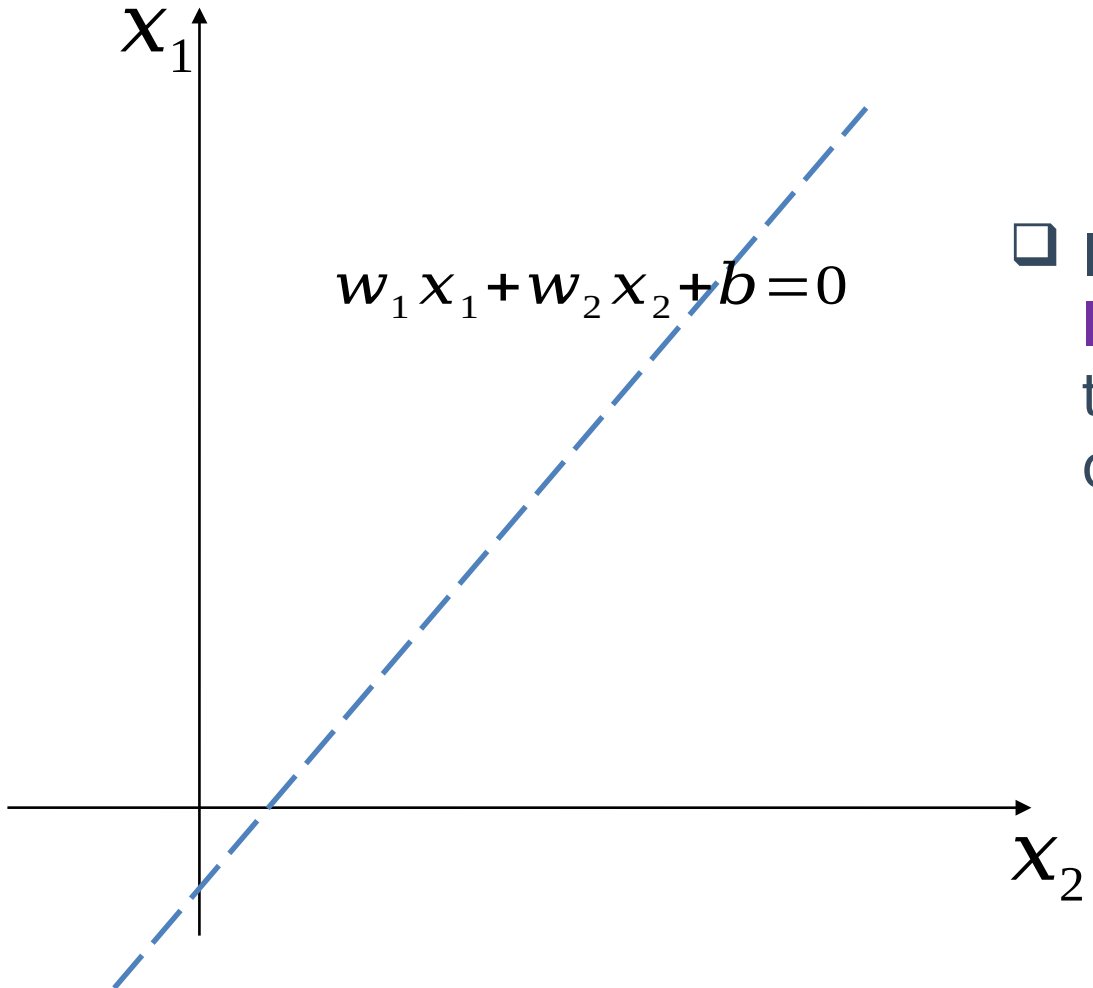
$$a_1 x_1 + a_2 x_2 + a_3 = 0$$

Suppose:

$$w_1 x_1 + w_2 x_2 + b = 0$$

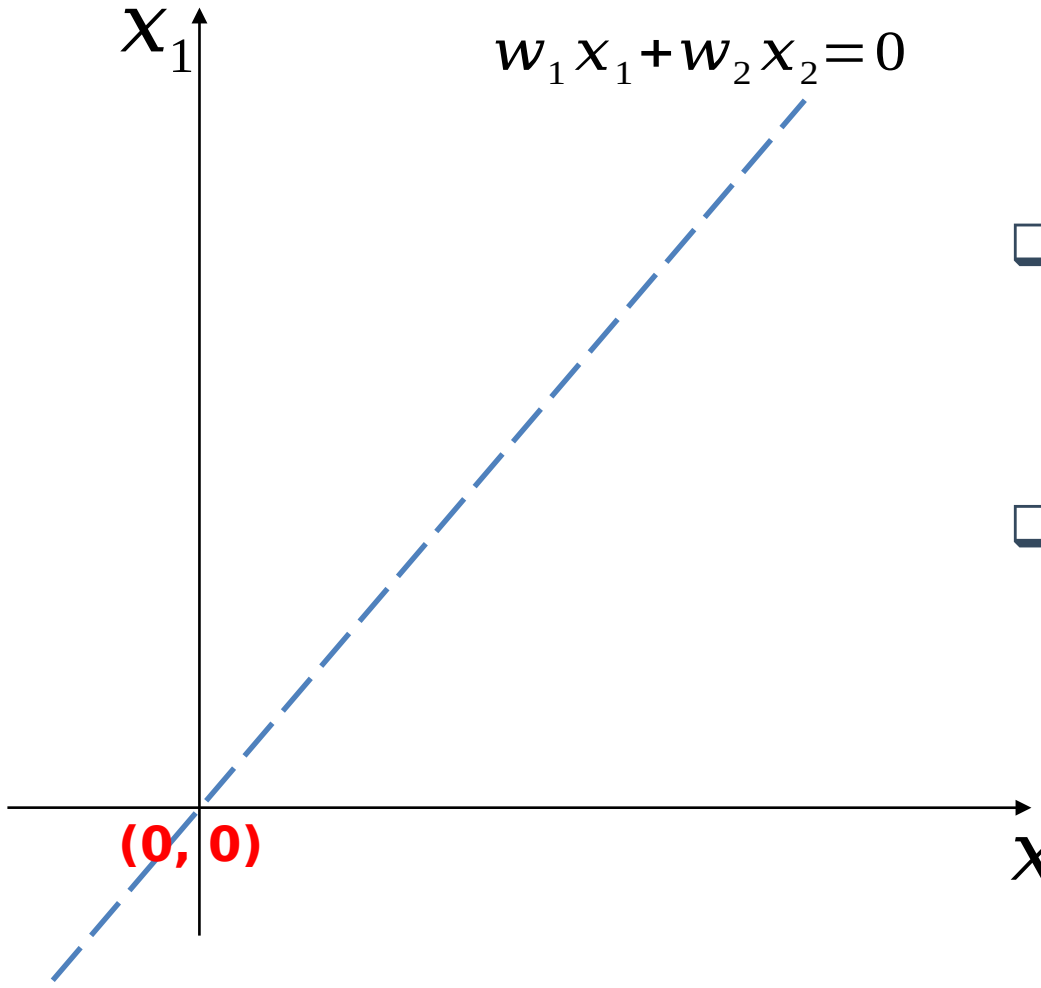
$$x_1 = -\frac{w_2}{w_1} x_2 - \frac{b}{w_1}$$

Line Equation (What is ?)



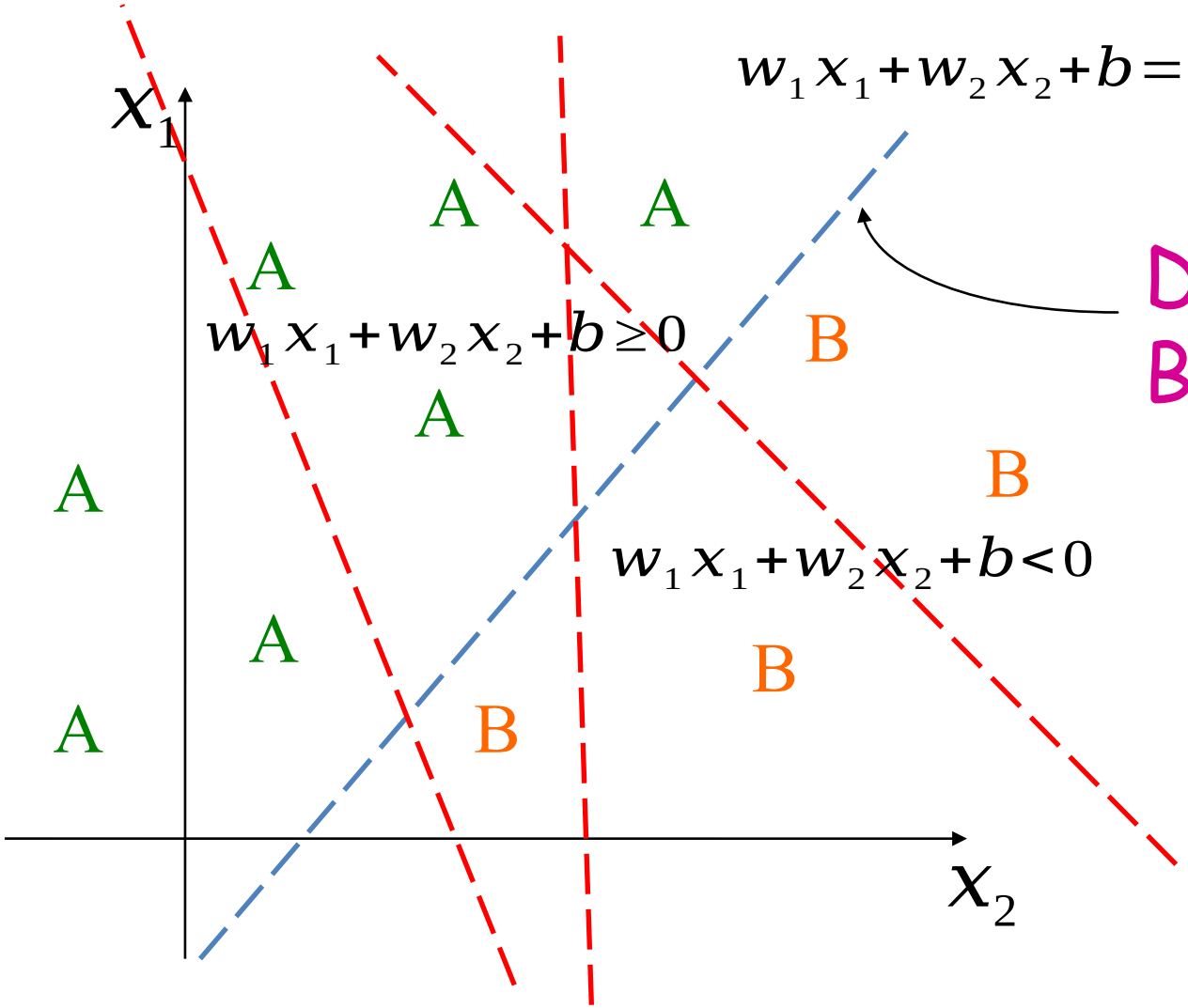
- It allows you to **move the line down and up** fitting the prediction with the data better.

Line Equation
()



- It allows you to **move the line down and up** fitting the prediction with the data better.
- If the constant is absent then the line will **pass through the origin (0, 0)** and you will get a poorer fit.

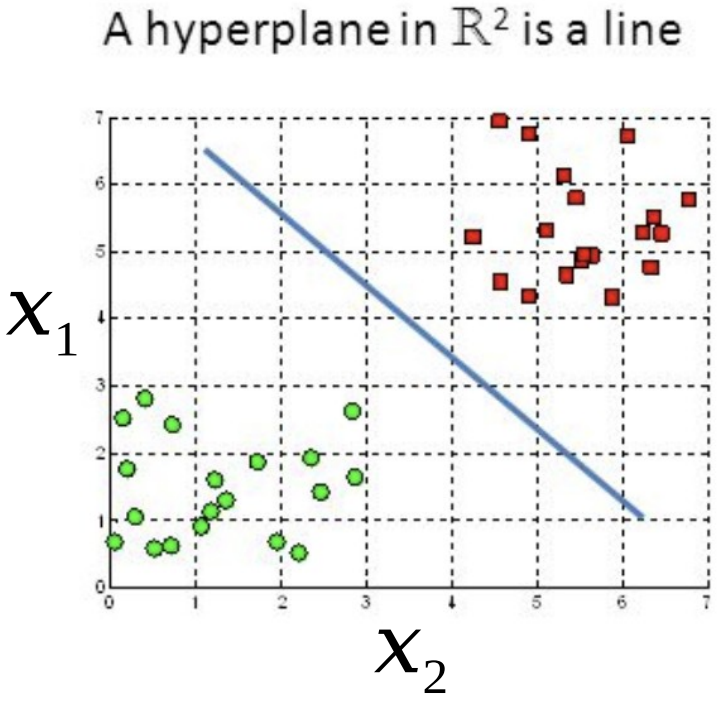
Decision Surface (Linear Decision Boundary)

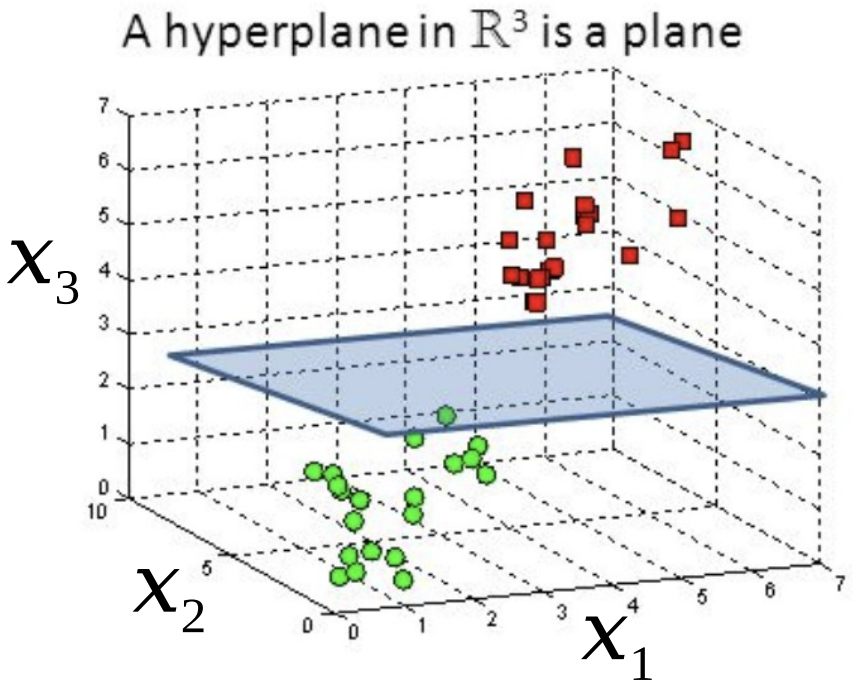


Decision Surface (Higher Dimension, e.g. 3D)

$a_1 x_1 + a_2 x_2 + a_3 = 0$

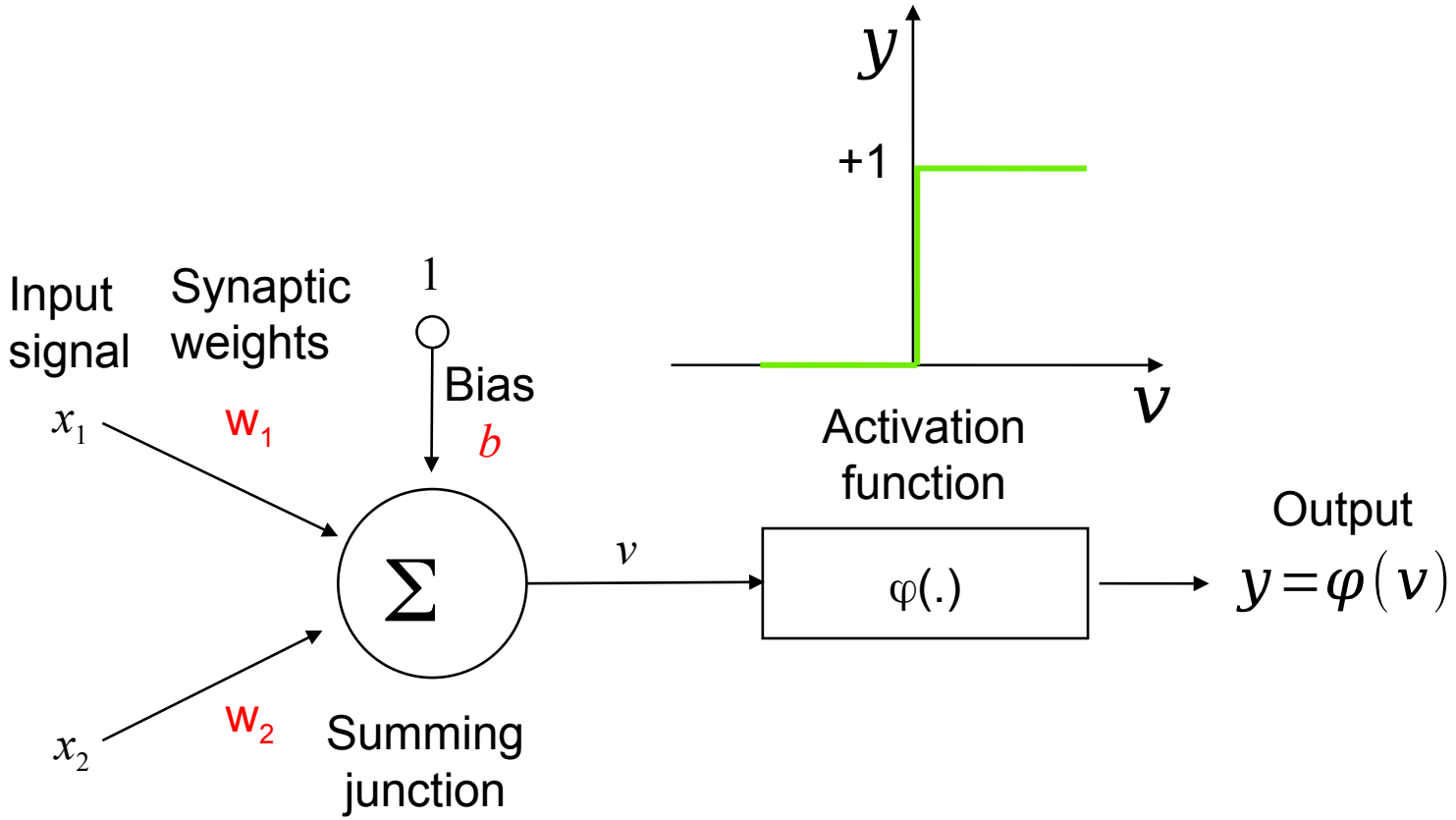
$a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 = 0$





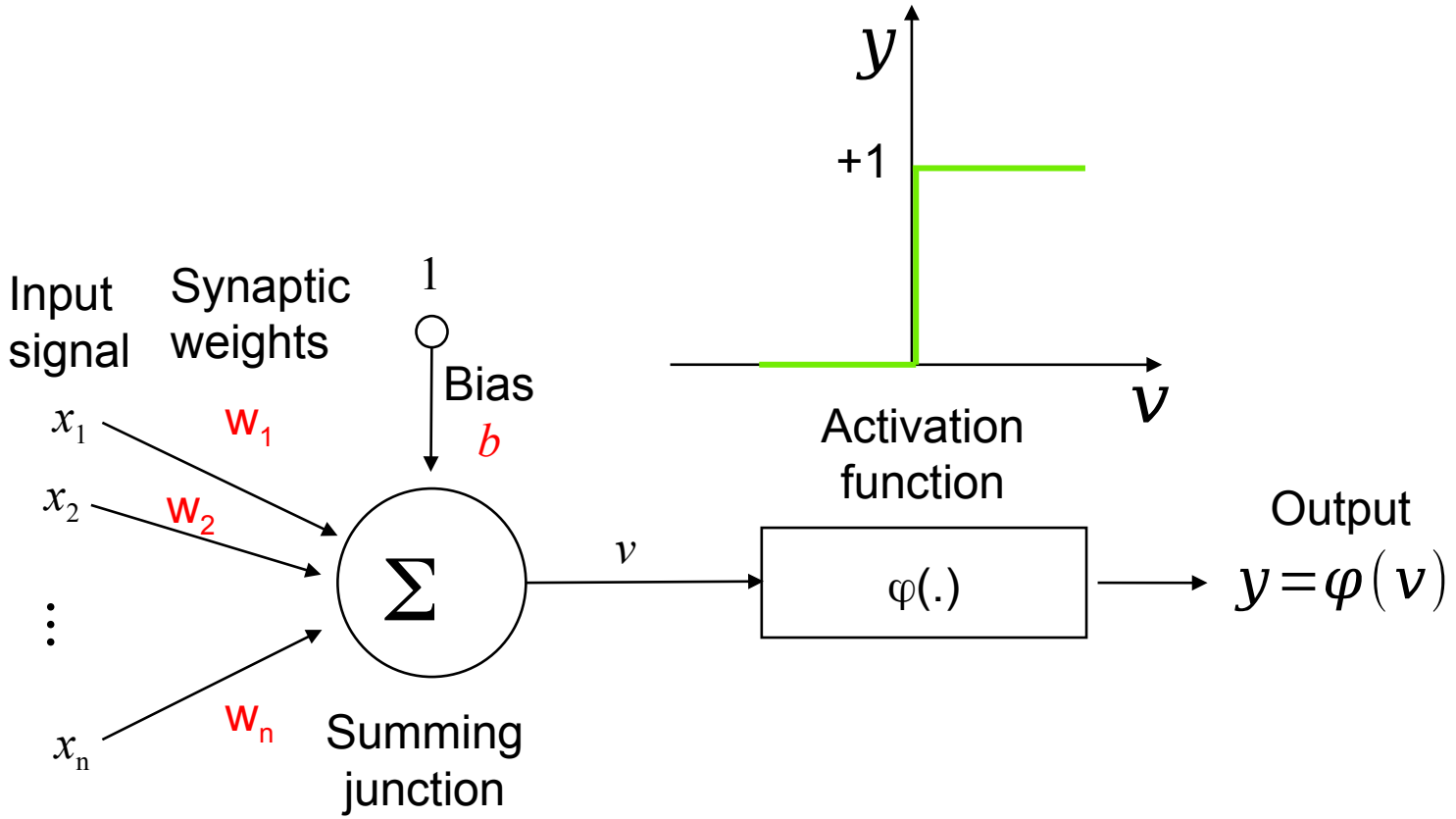
General:

Perceptron (2 Inputs)



$$v=w_1x_1+w_2x_2+b=\sum_{j=1}^{2}w_jx_j+b$$

Perceptron (general)



$$v = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b = \sum_{j=1}^{n} w_j x_j + b$$

Perceptron

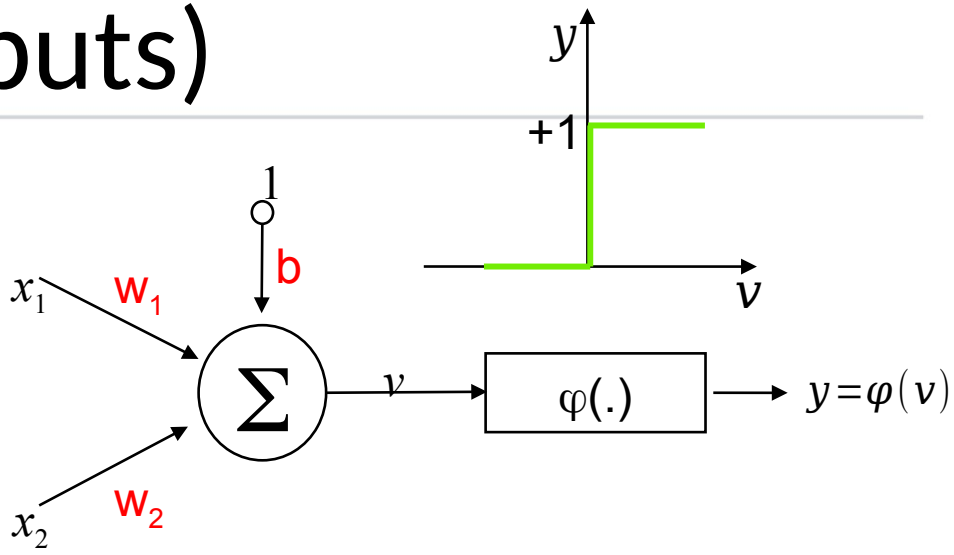
Lancaster University

$$v = w_1 x_1 + w_2 x_2 + b = \sum_{j=1}^{2} w_j x_j + b$$

and

$$v = w_1 x_1 + w_2 x_2 + b = W^T X$$

Perceptron Learning (2 Inputs)



Initialize randomly

While *not converge* **do**

For each sample in the dataset

if then

End

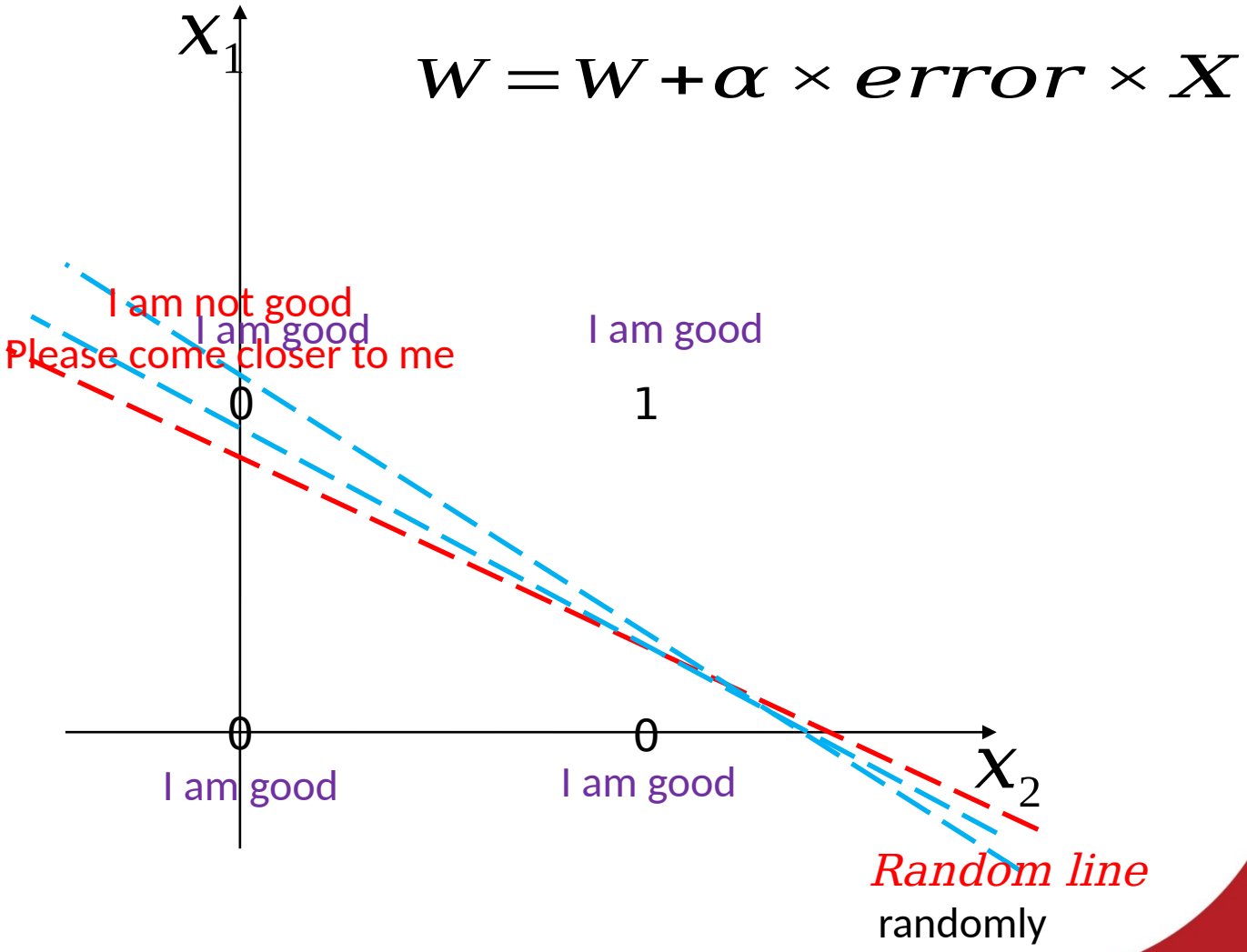
End

End

is learning rate (e.g. 0.1, 0.001)

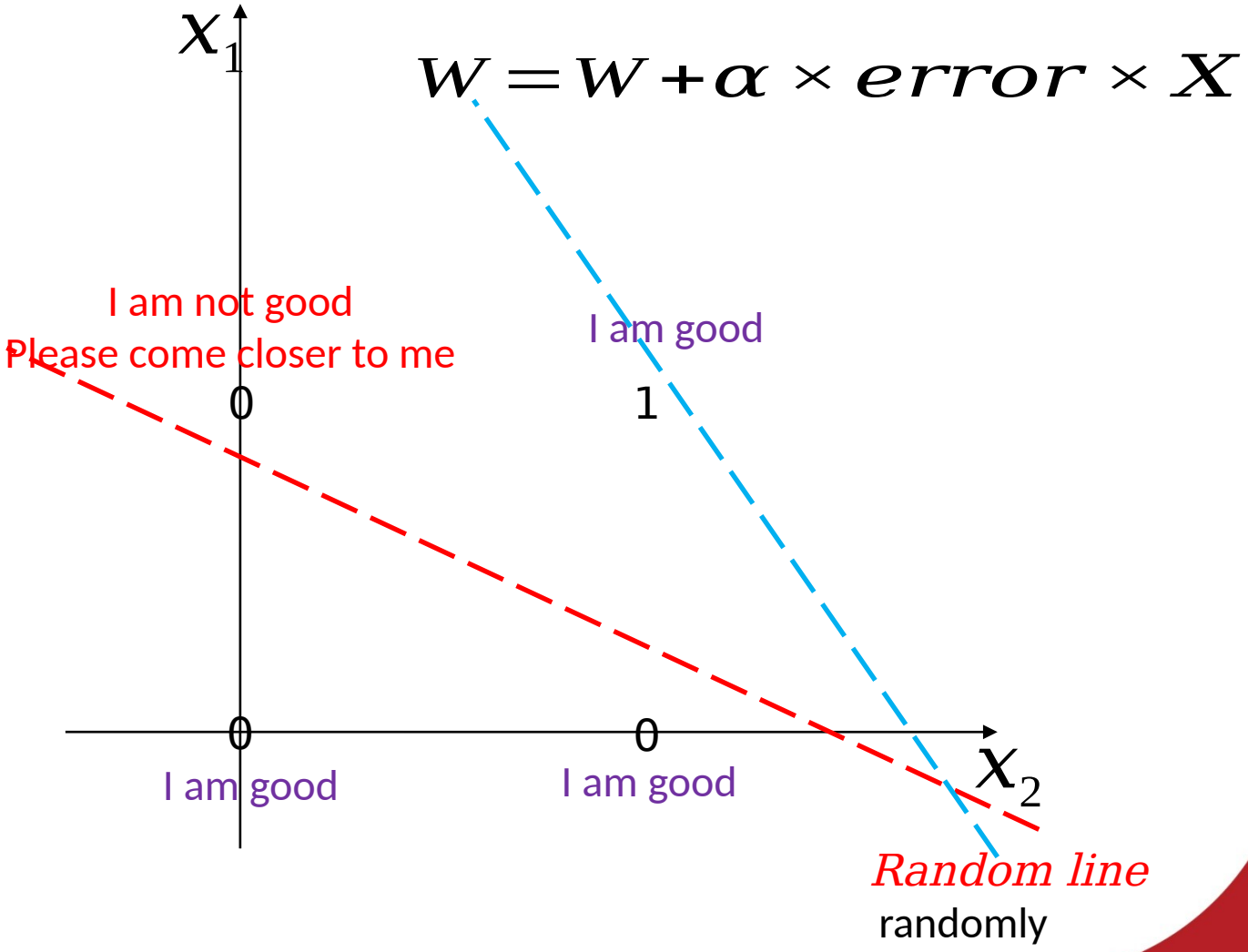
Perceptron Learning (Example - small learning rate)

AND		
		out
0	0	0
0	1	0
1	0	0
1	1	1



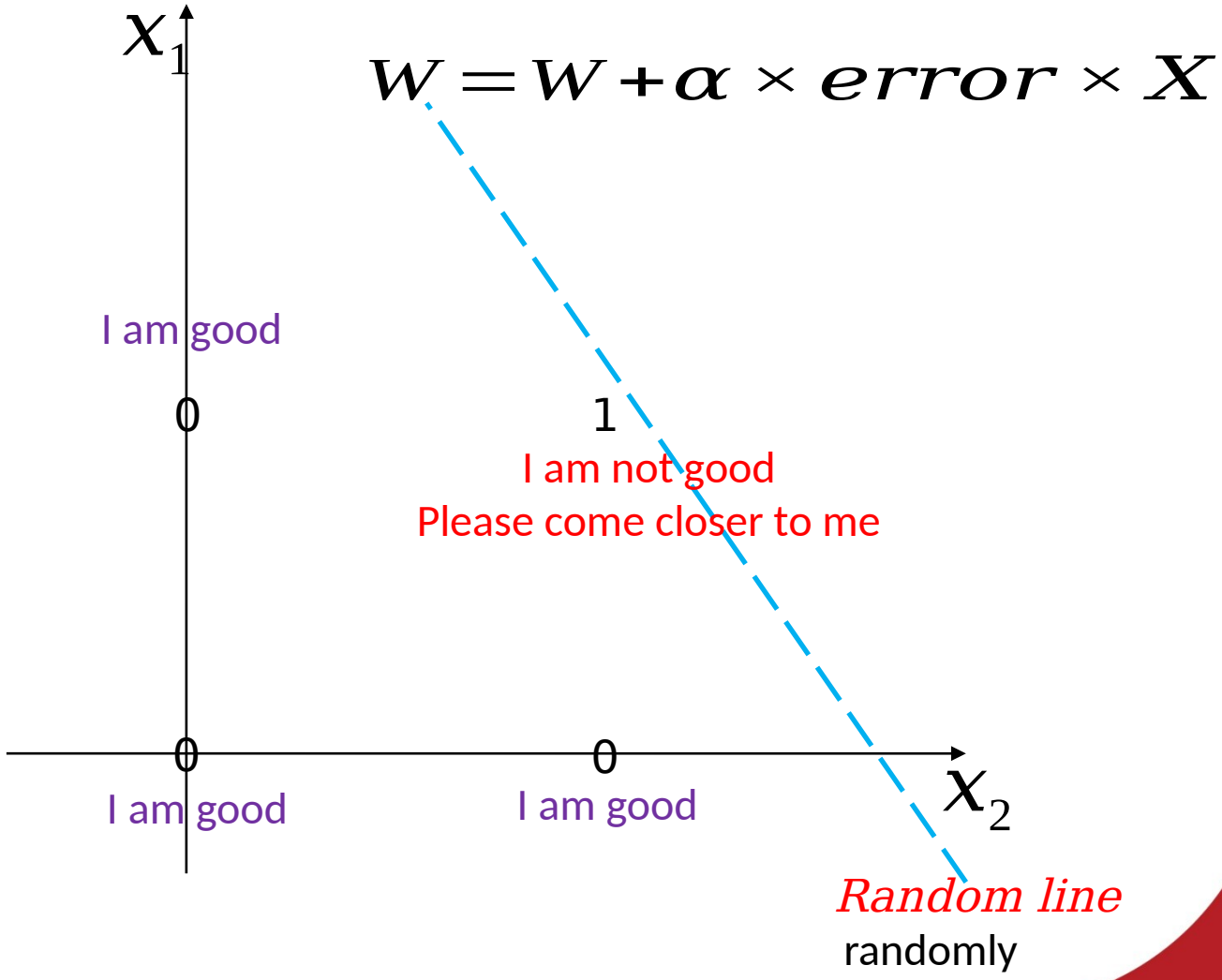
Perceptron Learning (Example - big learning rate)

AND		
		out
0	0	0
0	1	0
1	0	0
1	1	1



Perceptron Learning (Example - big learning rate)

AND		
		out
0	0	0
0	1	0
1	0	0
1	1	1

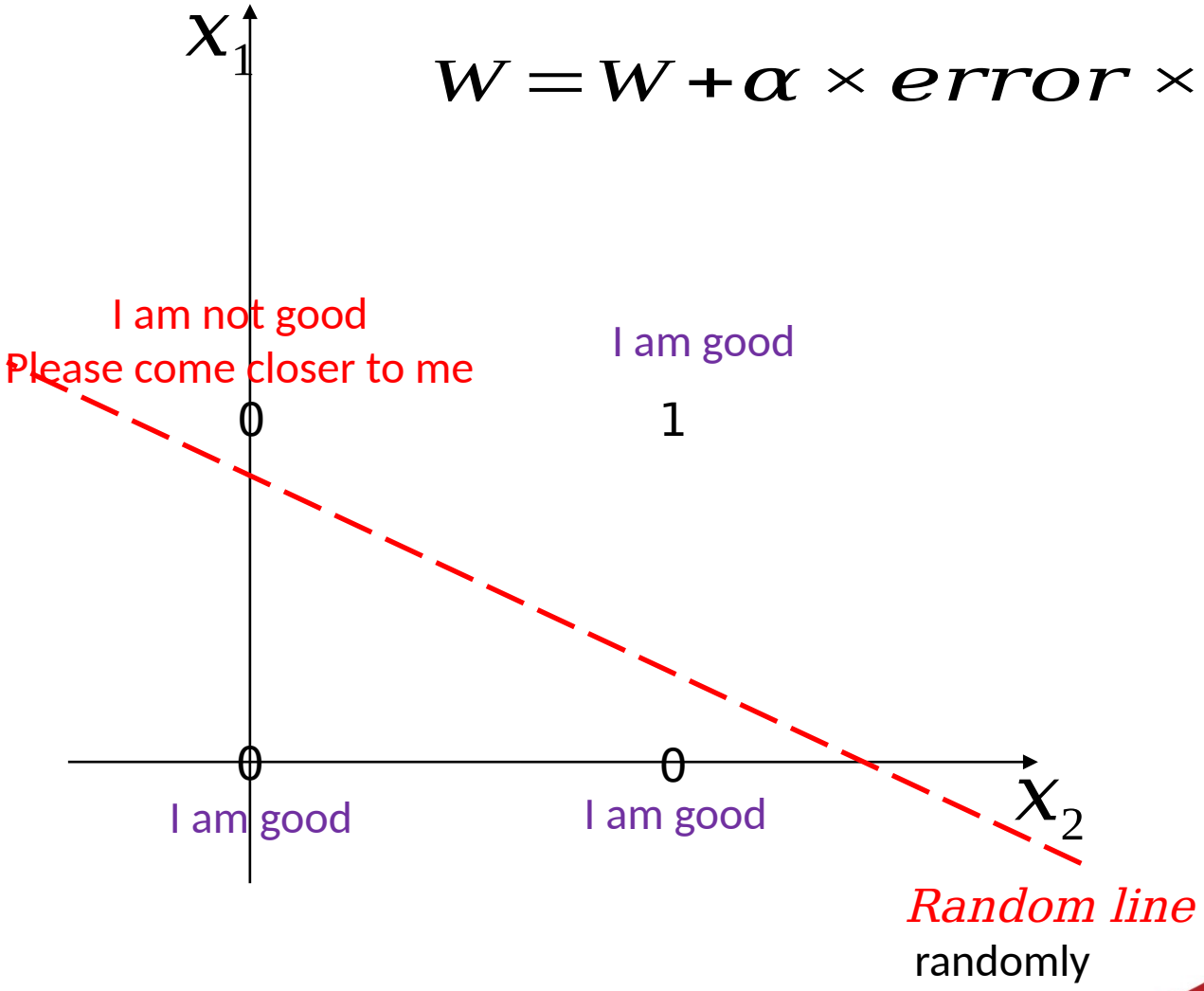


Perceptron Learning (Example - big learning rate)

Lancaster University

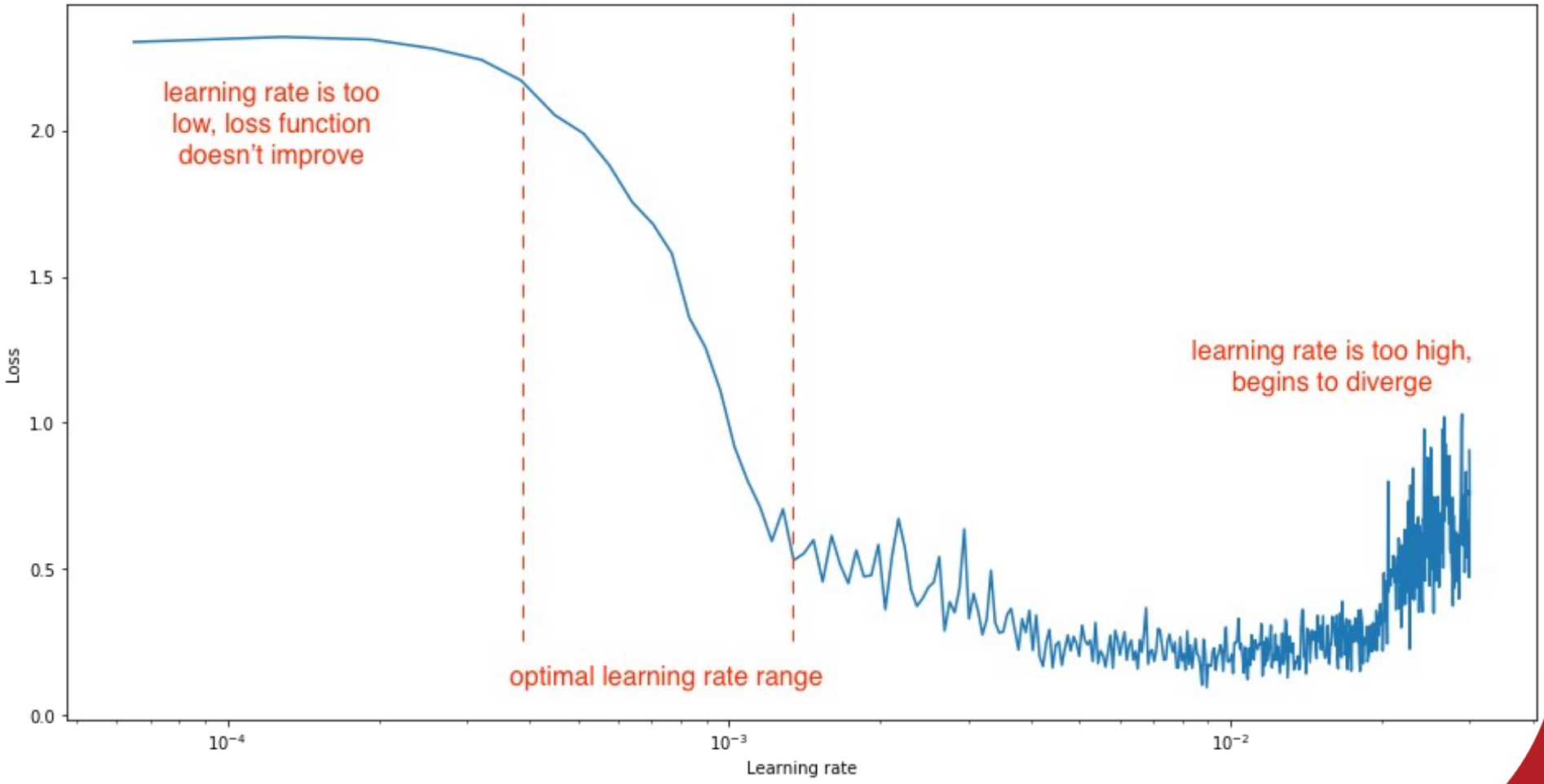
AND		
		out
0	0	0
0	1	0
1	0	0
1	1	1

$$W = W + \alpha \times error \times X$$



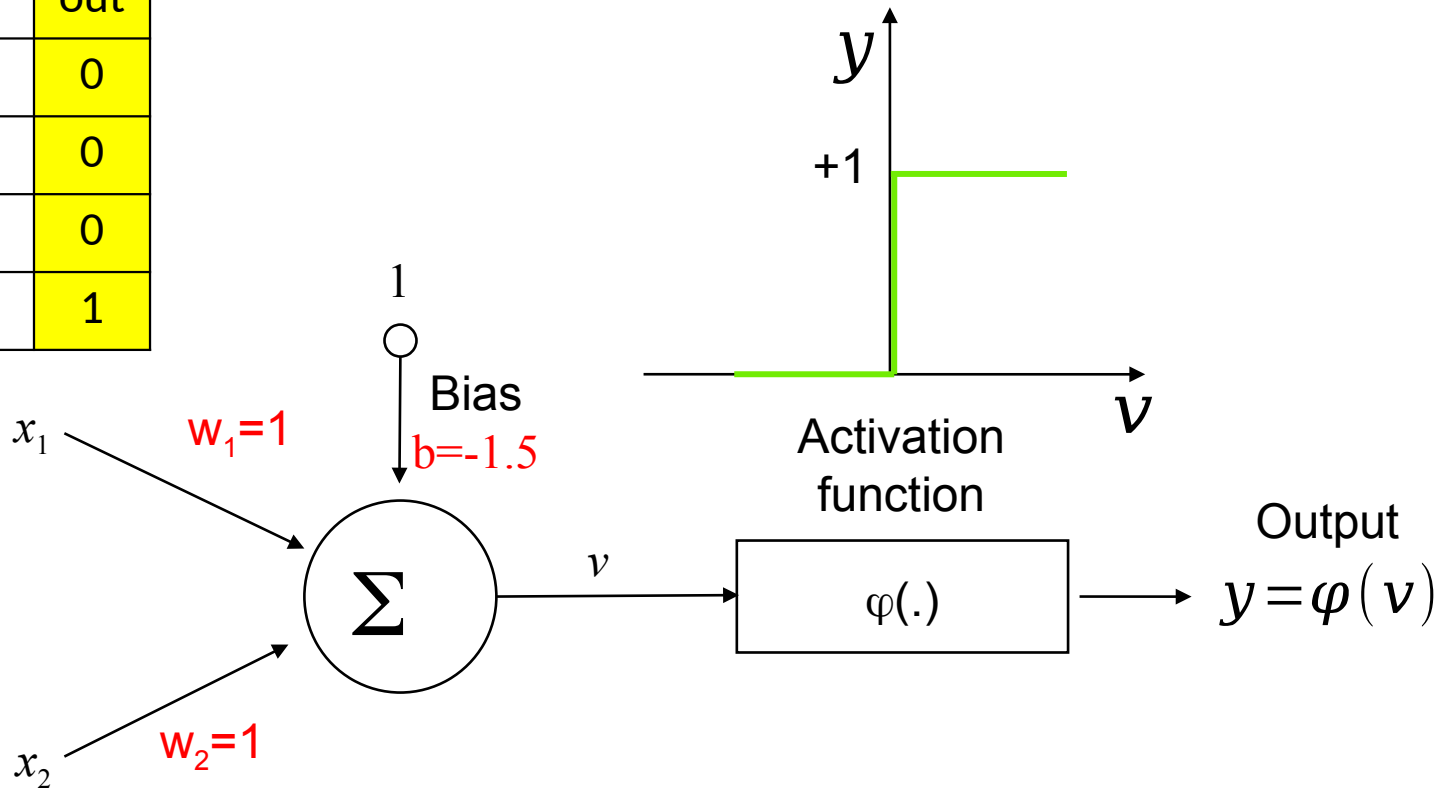
Effects of learning rate

Lancaster University



Trained Network (AND)

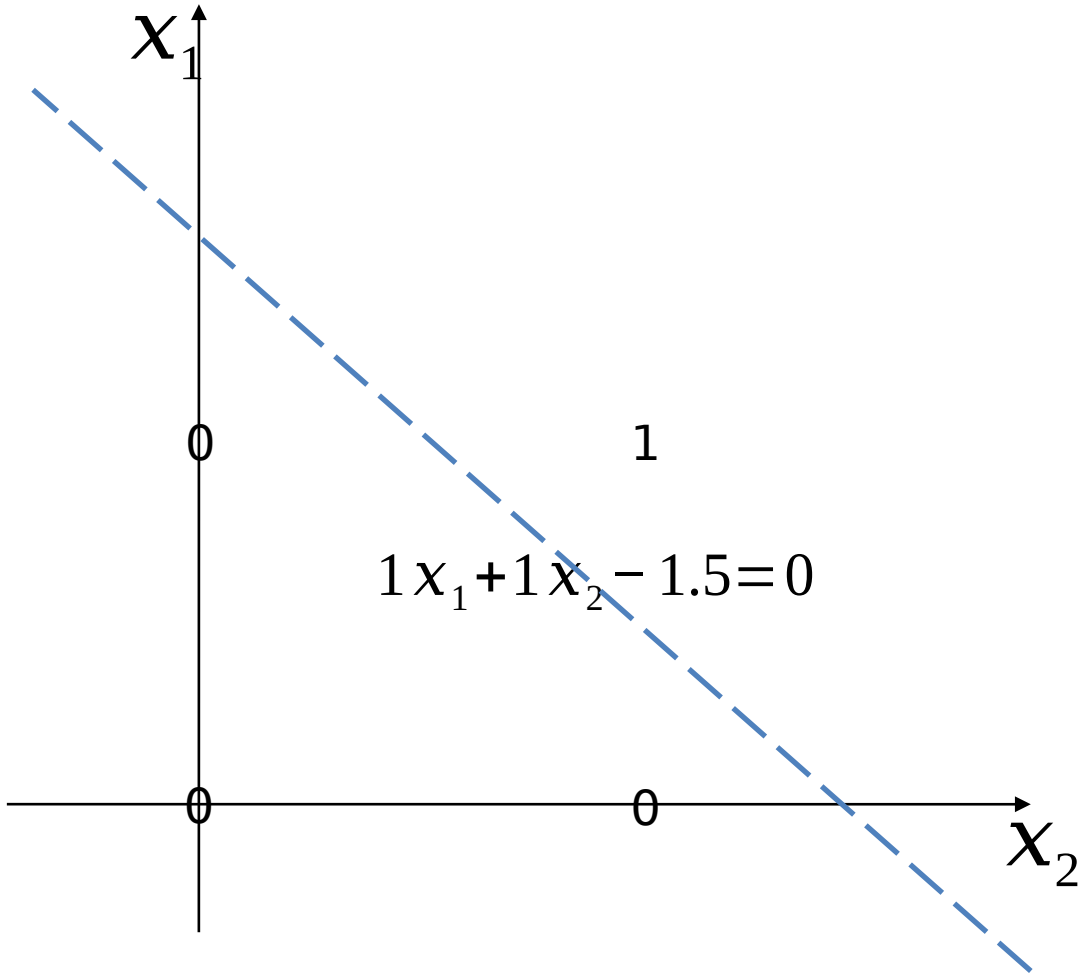
AND		
		out
0	0	0
0	1	0
1	0	0
1	1	1



$$y=\varphi(1\,x¿¿1+1\,x_2-1.5)¿$$

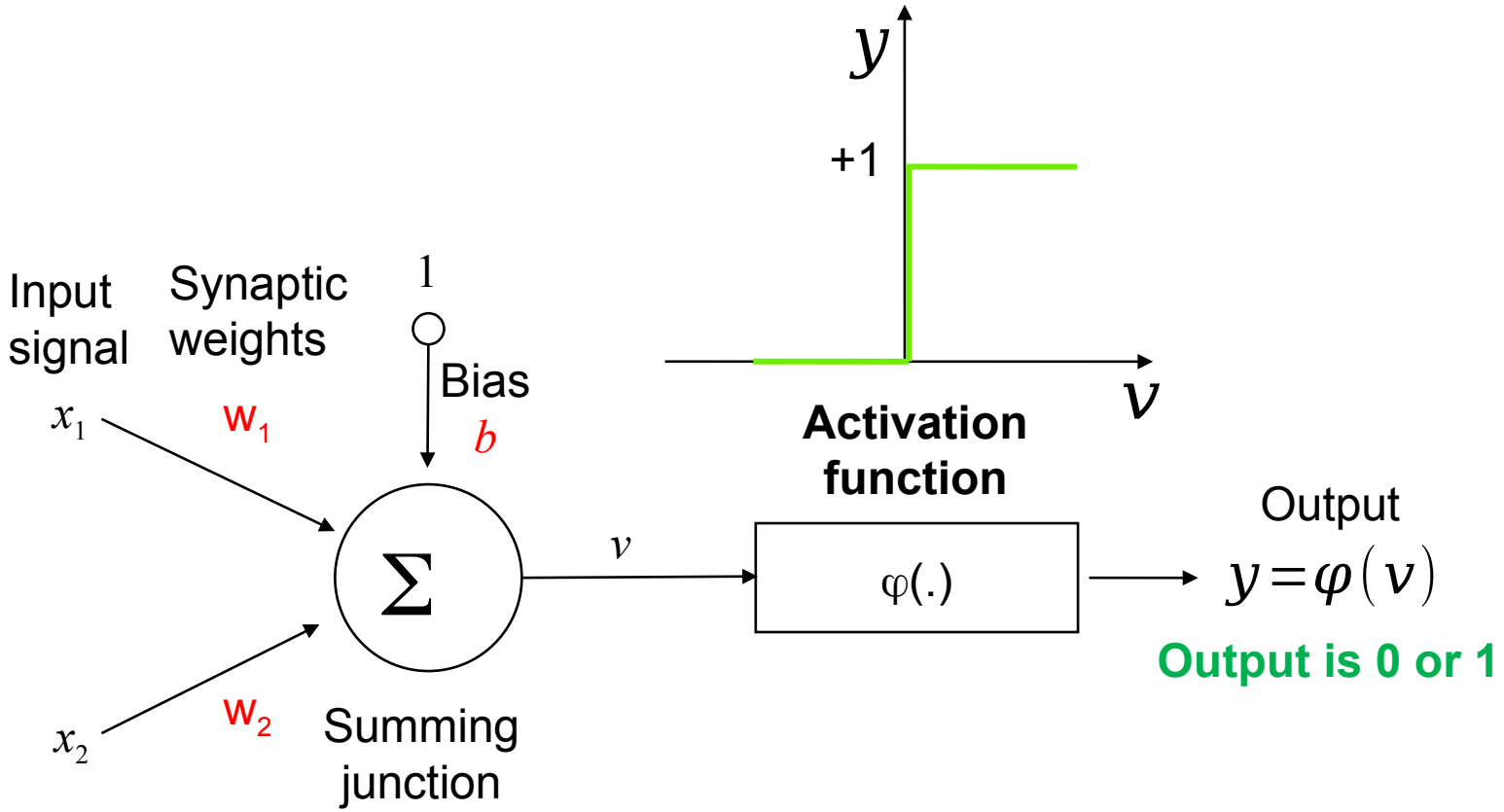
Trained Network (AND)

AND		
		out
0	0	0
0	1	0
1	0	0
1	1	1



Discrete vs. Continuous

Step function (hard threshold)



$$v = w_1 x_1 + w_2 x_2 + b = \sum_{j=1}^{2} w_j x_j + b$$

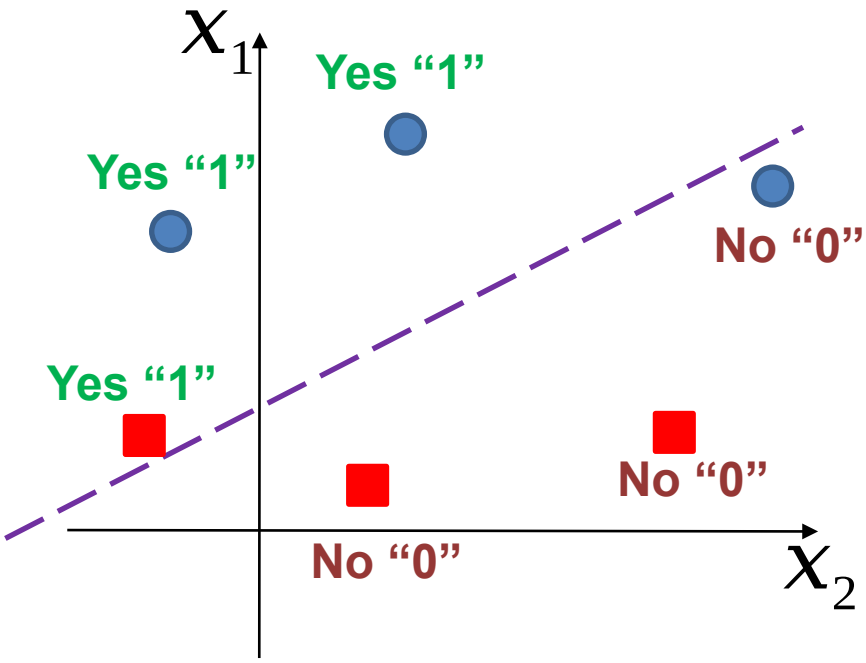
Discrete vs. Continuous

Lancaster University

Step function (hard threshold)

Discrete

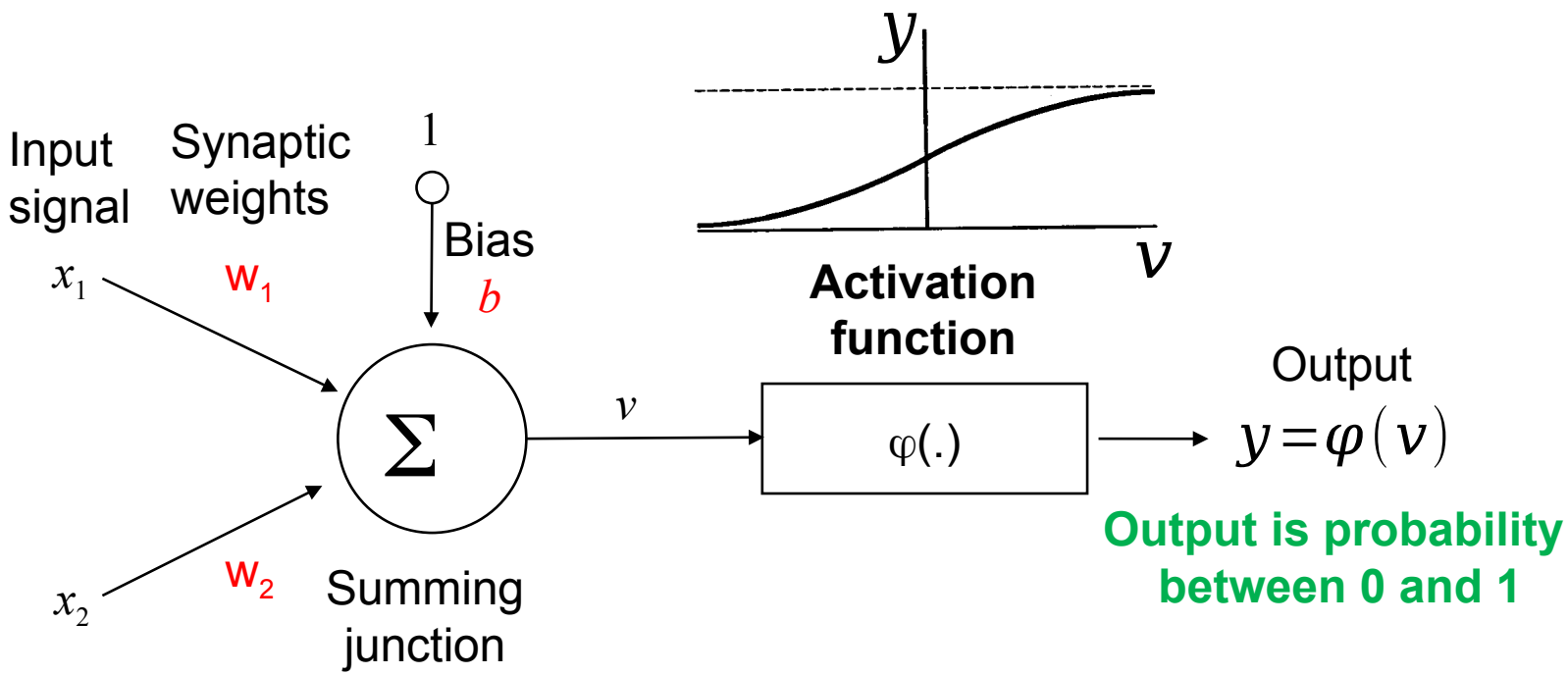
e.g. Step function



Discrete vs. Continuous Sigmoid function

Lancaster University

$$y = \frac{1}{1+e^{-v}}$$

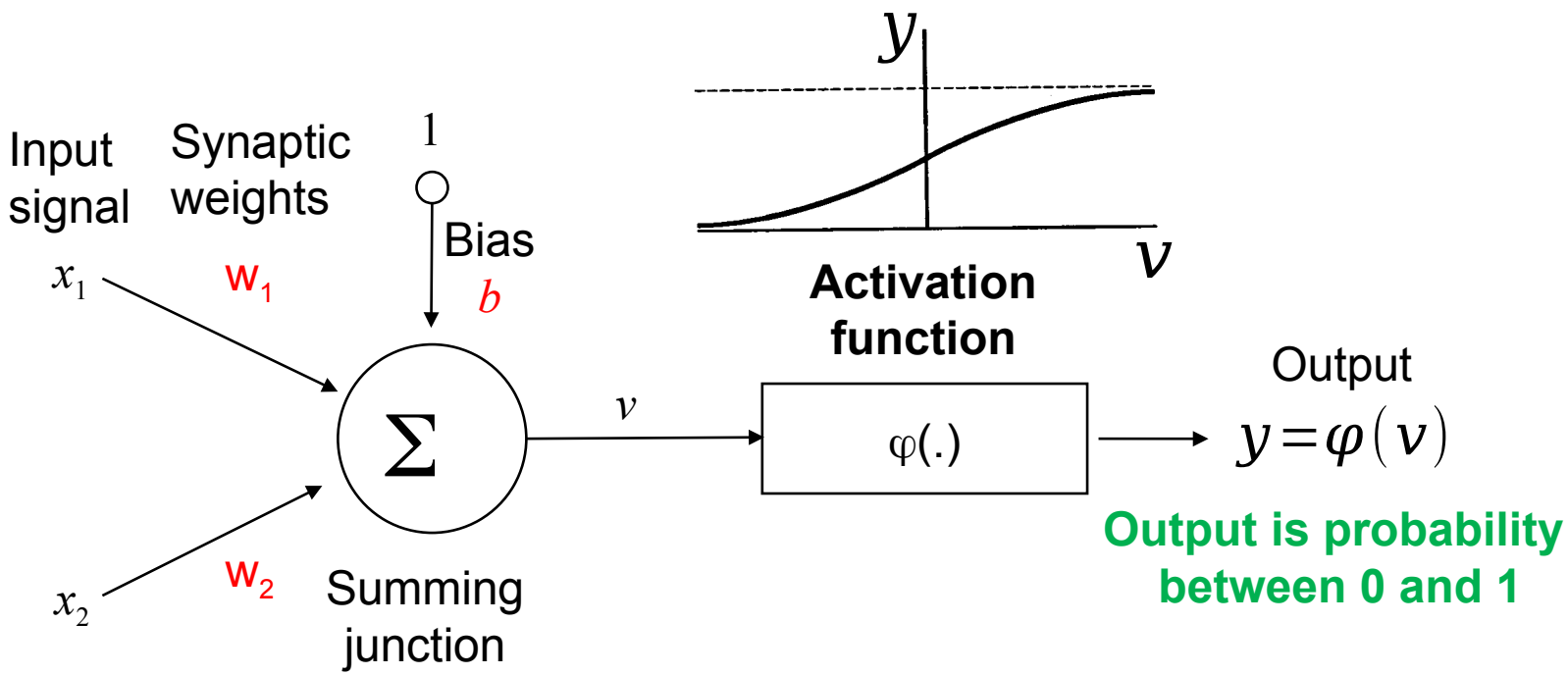


$$v = w_1 x_1 + w_2 x_2 + b = \sum_{j=1}^{2} w_j x_j + b$$

Discrete vs. Continuous Sigmoid function

Lancaster University

Sigmoid $y = \frac{1}{1+e^{-v}}$



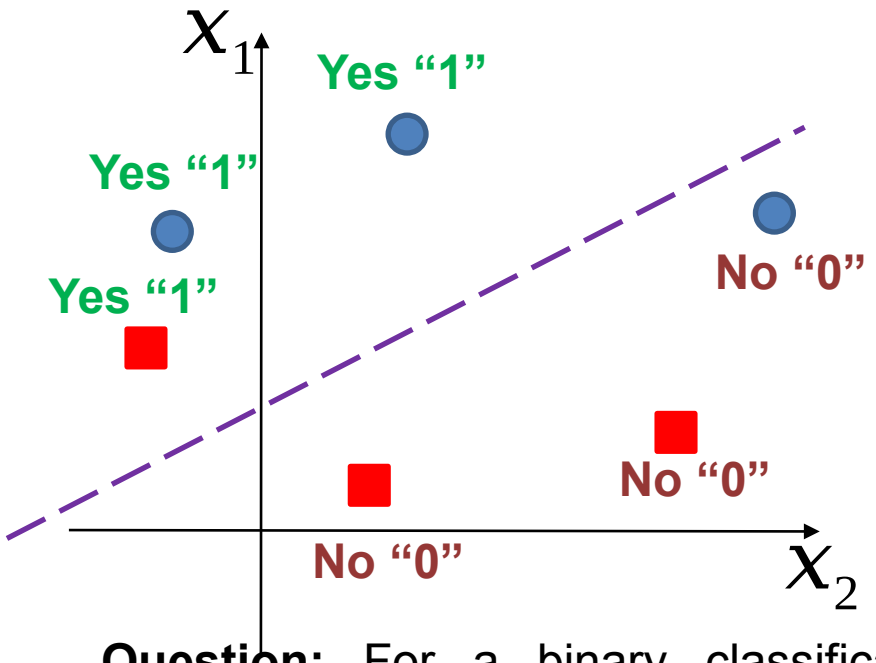
Question: For a binary classification, we need to calculate the probability of class 0 and class 1. How to interpret the output of the above Sigmoid activation function?

Discrete vs. Continuous

Lancaster University

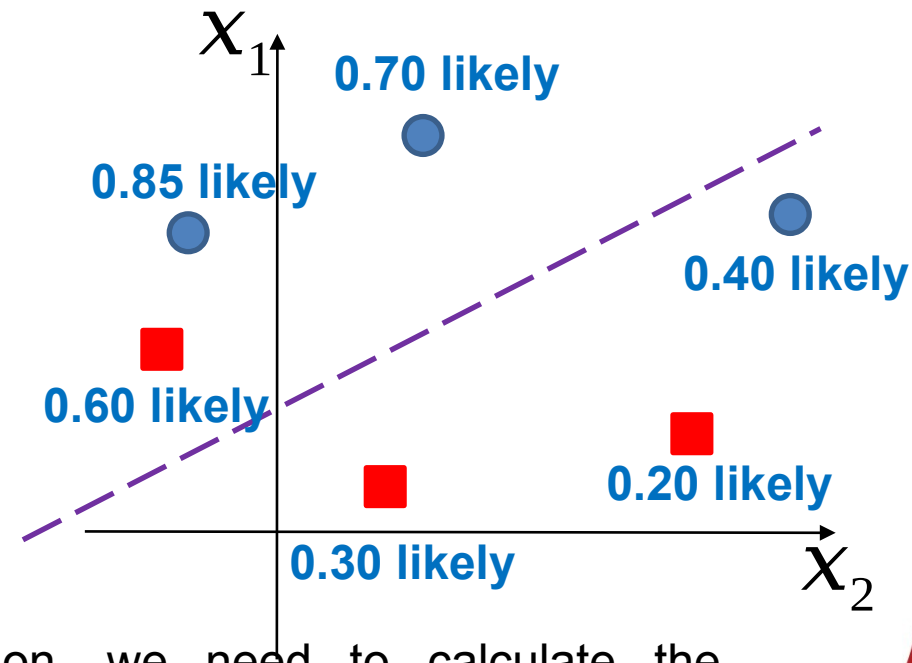
Discrete

e.g. Step function



Continuous

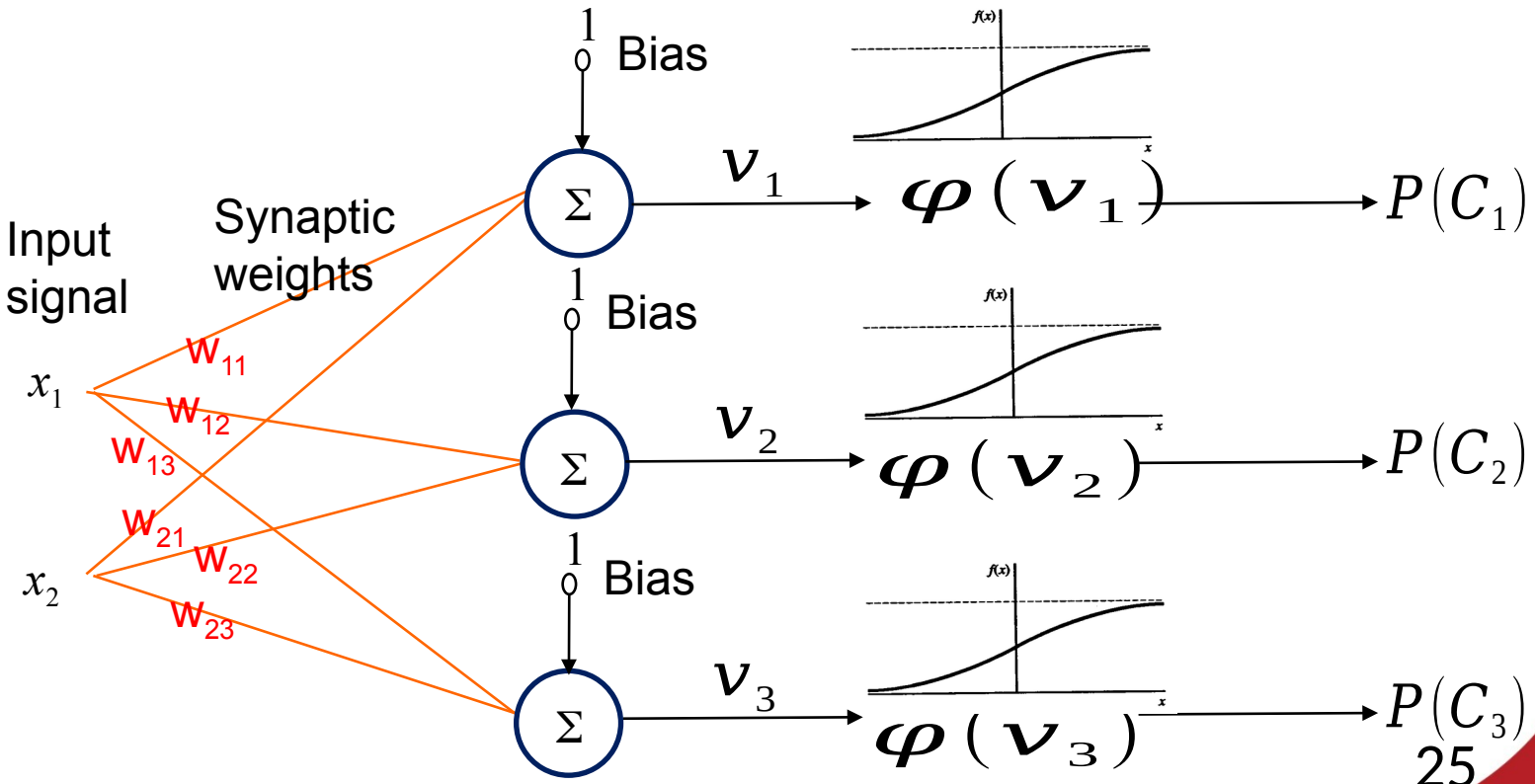
e.g. Sigmoid function



Question: For a binary classification, we need to calculate the probability of class 0 and class 1. How to interpret the output of the above Sigmoid activation function? We define a class of interest (e.g., BLUE)

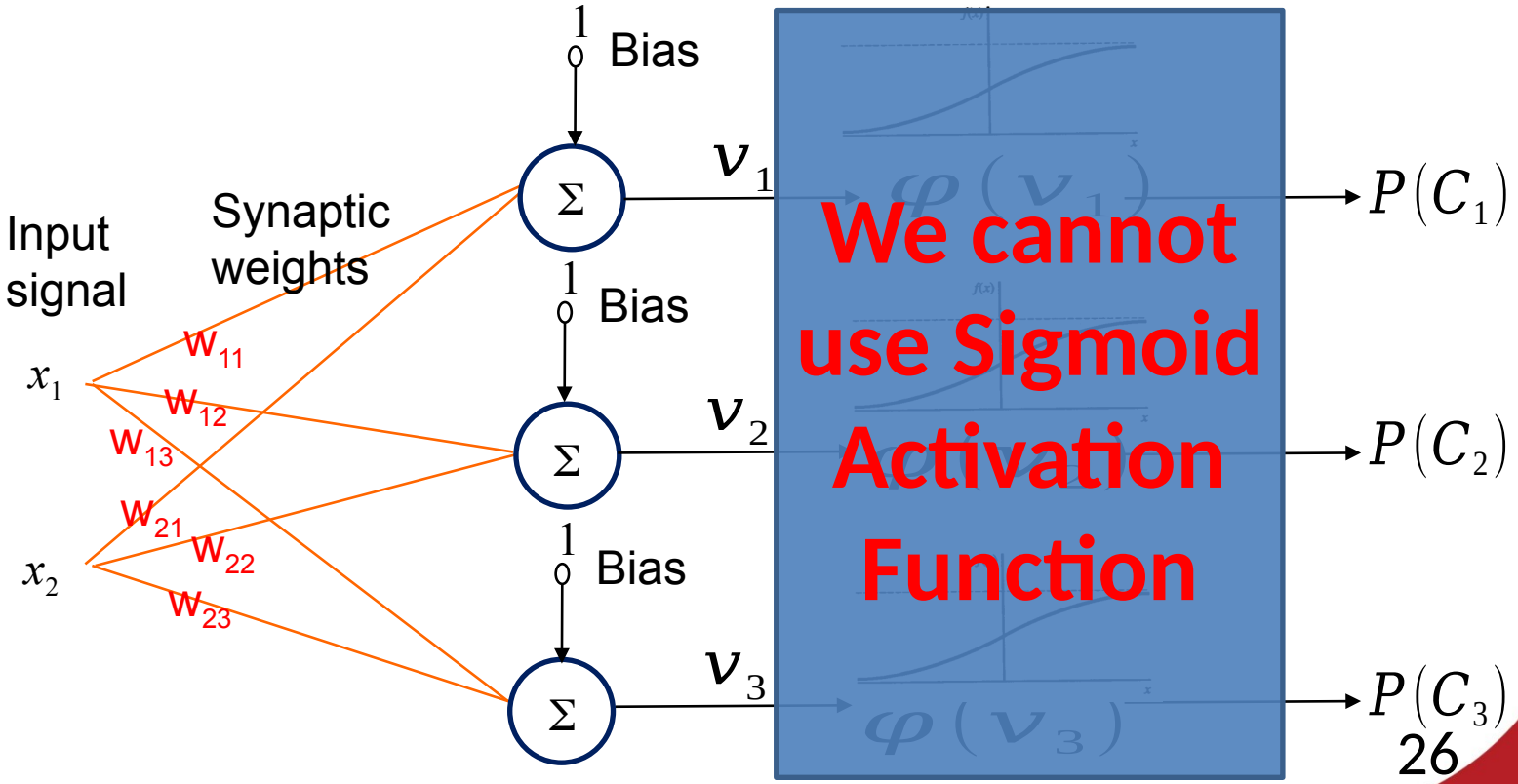
Multi-Class Classification Softmax

- So far, we used Sigmoid activation function to output a probability for a binary (2 class classification) classification
- What about multi-class classification? $P(C_1)+P(C_2)+P(C_3)=$



Multi-Class Classification Softmax

- So far, we used Sigmoid activation function to output a probability for a binary (2 class classification) classification
- What about multi-class classification?



Lancaster University

Questions?

AI SCC361 Week 9

Dr. Hossein Rahmani
Senior Lecturer in Data Science
D17, InfoLab; e-mail: h.rahmani@lancaster.ac.uk

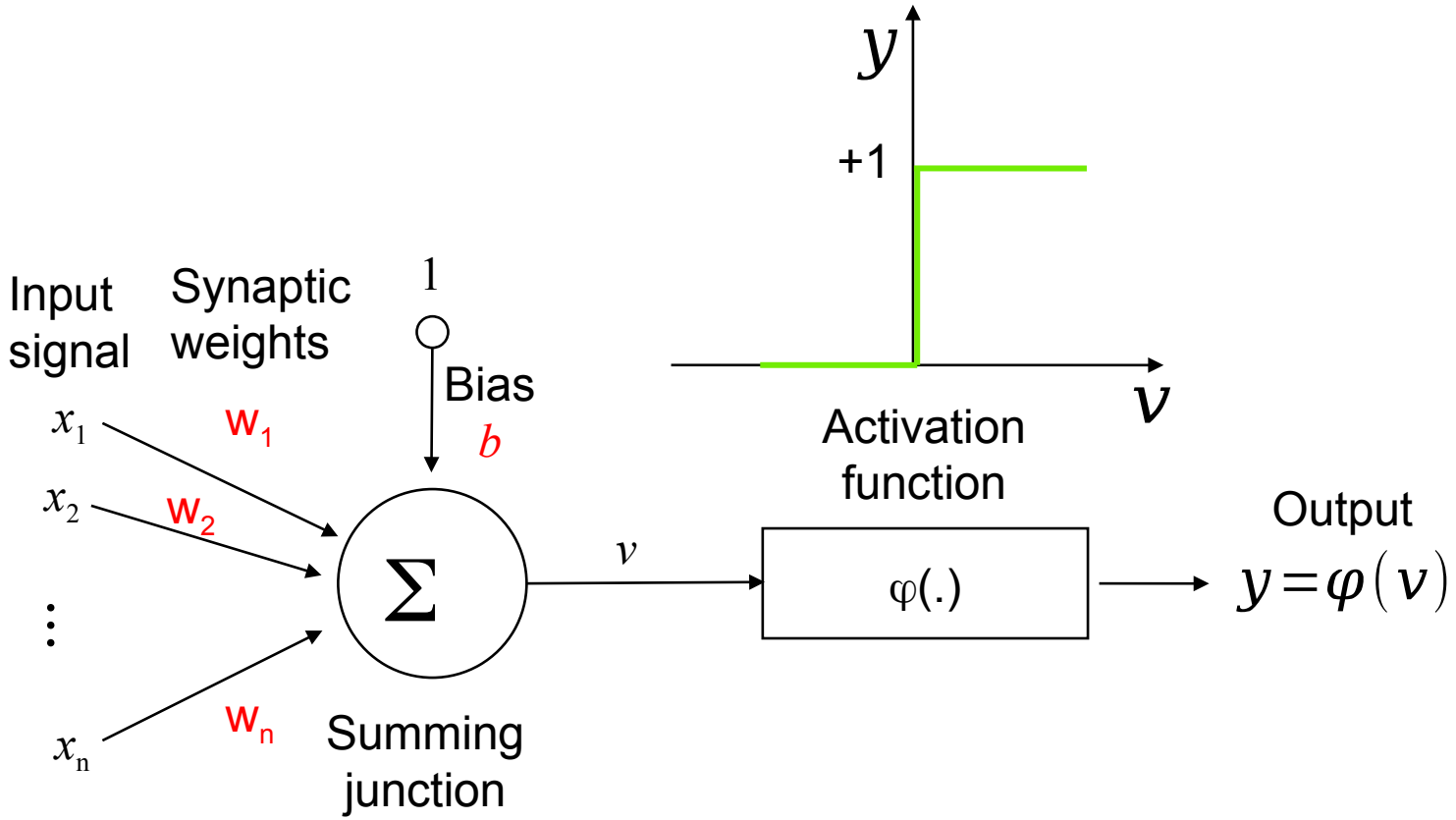
Neural Networks Cont.

Outline

Lancaster University

- Decision Surface
- Perceptron
  - Binary classification
- Discrete vs Continuous Perceptron
- Multi-class classification
  - Softmax function
- Error/Loss functions
  - Mean Square Error (MSE)
  - Maximum Likelihood (ML)
  - Cross-Entropy (CE)
- Multi Layer Perceptron (MLP)
- Backpropagation

Perceptron (general)

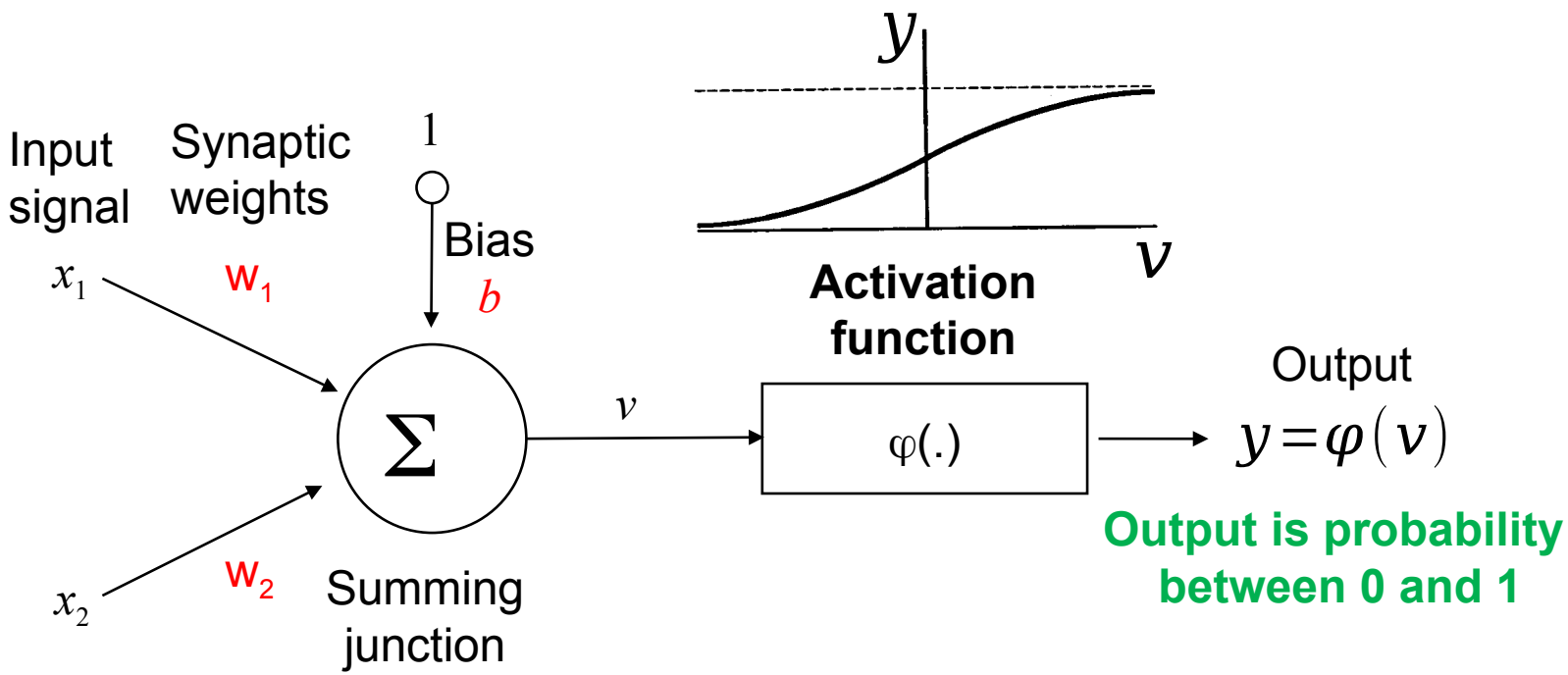


$$v=w_1x_1+w_2x_2+\ldots+w_nx_n+b=\sum_{j=1}^{n} w_jx_j+b$$

Discrete vs. Continuous Sigmoid function

Lancaster University

Sigmoid $y = \frac{1}{1+e^{-v}}$

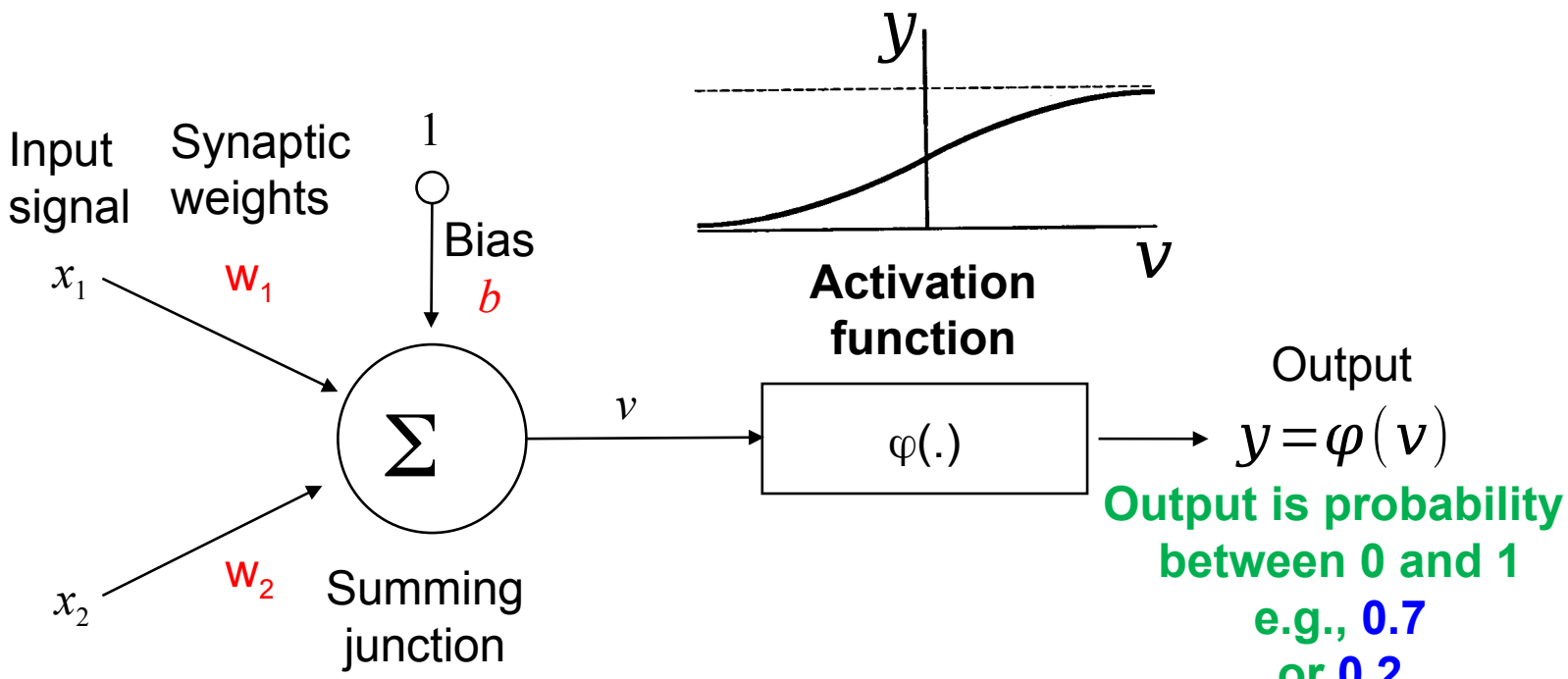


$$v = w_1 x_1 + w_2 x_2 + b = \sum_{j=1}^{2} w_j x_j + b$$

Discrete vs. Continuous

Lancaster University

Sigmoid $y = \frac{1}{1+e^{-v}}$



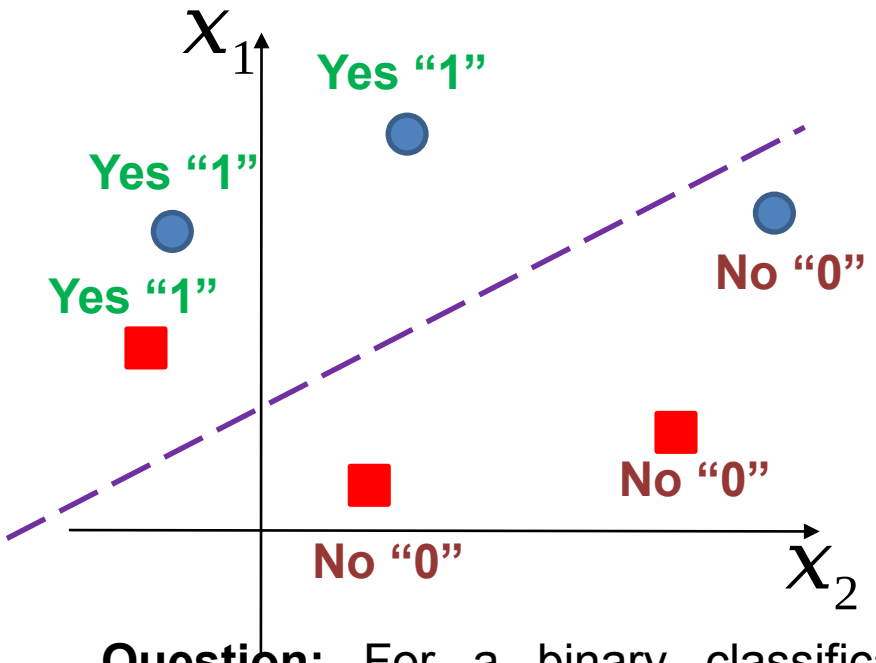
Question: How to interpret the output of the above Sigmoid activation function? We define a class of interest (e.g., BLUE) and thus the output is probability of sample belonging to class BLUE

Discrete vs. Continuous

Lancaster University

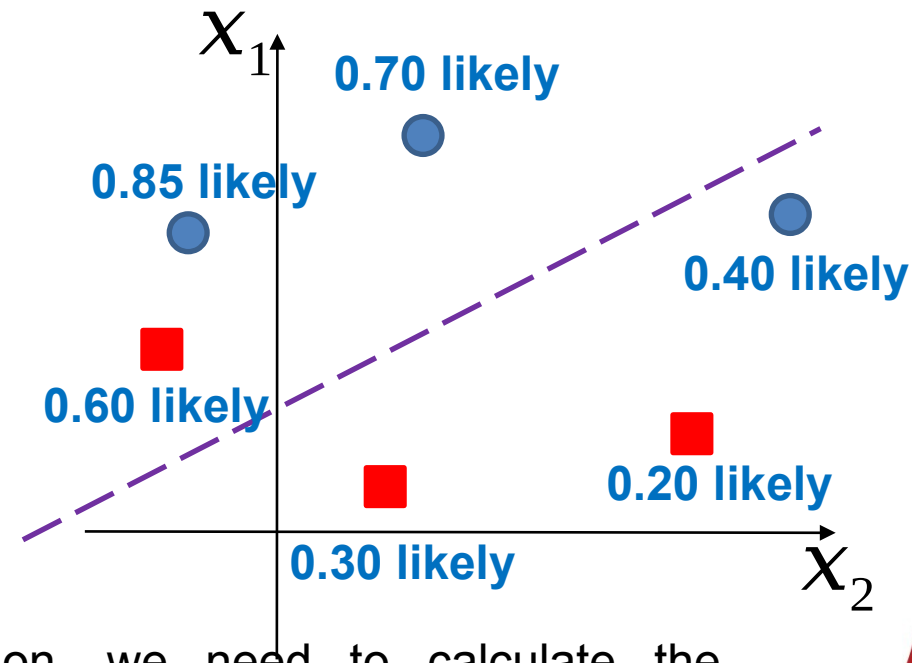
Discrete

e.g. Step function



Continuous

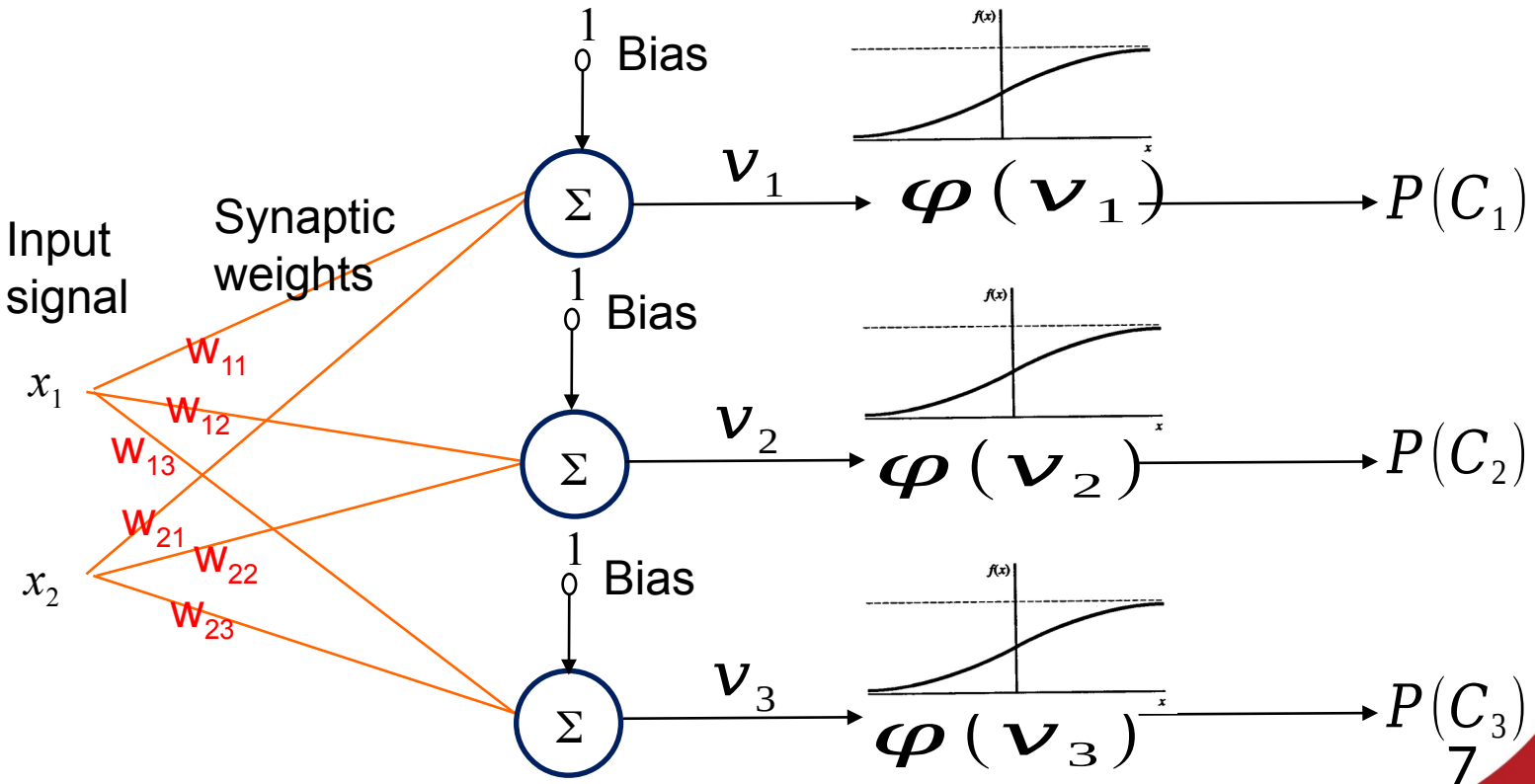
e.g. Sigmoid function



Question: For a binary classification, we need to calculate the probability of class 0 and class 1. How to interpret the output of the above Sigmoid activation function? We define a class of interest (e.g., BLUE)

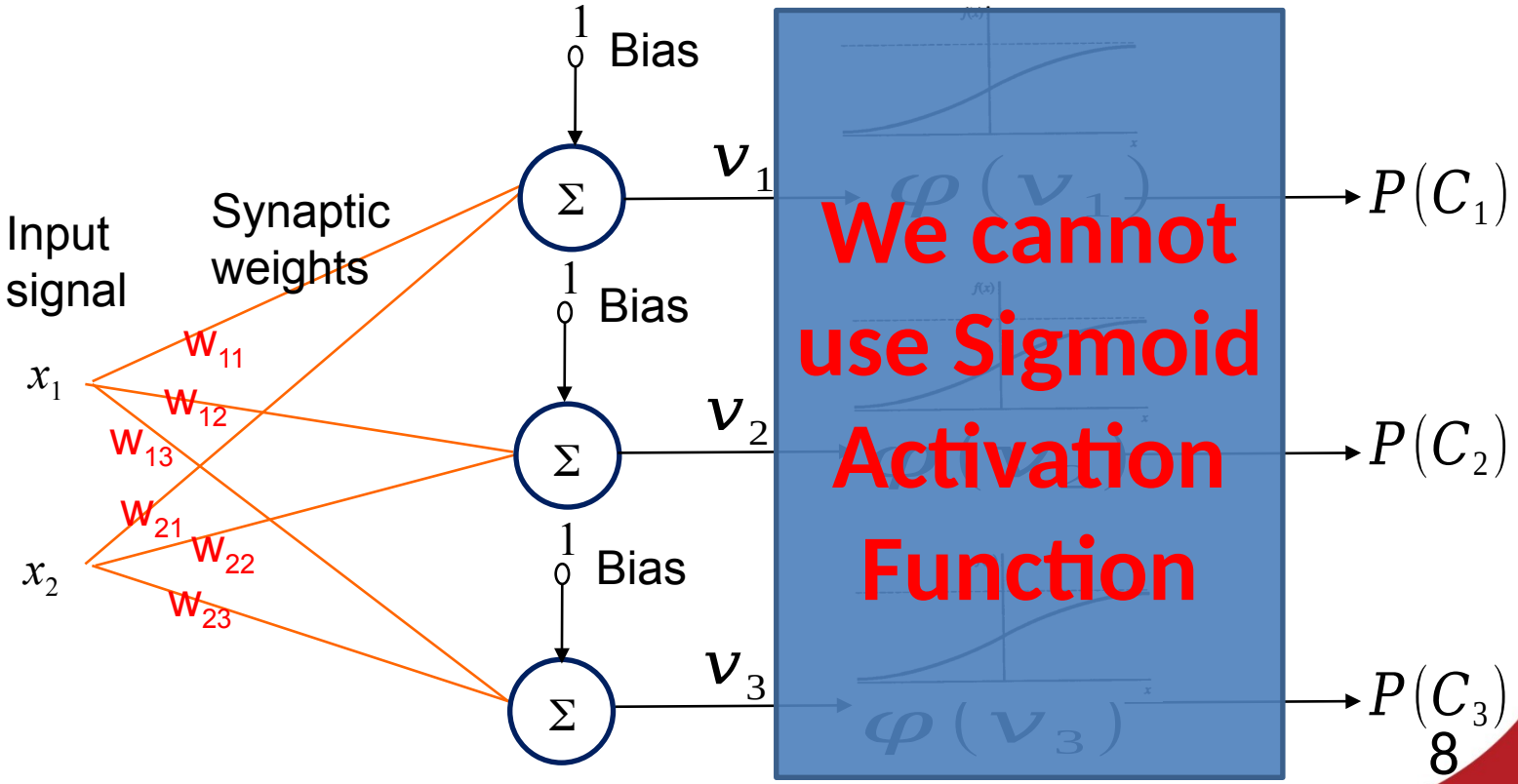
Multi-Class Classification Softmax

- So far, we used Sigmoid activation function to output a probability for a binary (2 class classification) classification
- What about multi-class classification? $P(C_1) + P(C_2) + P(C_3) =$



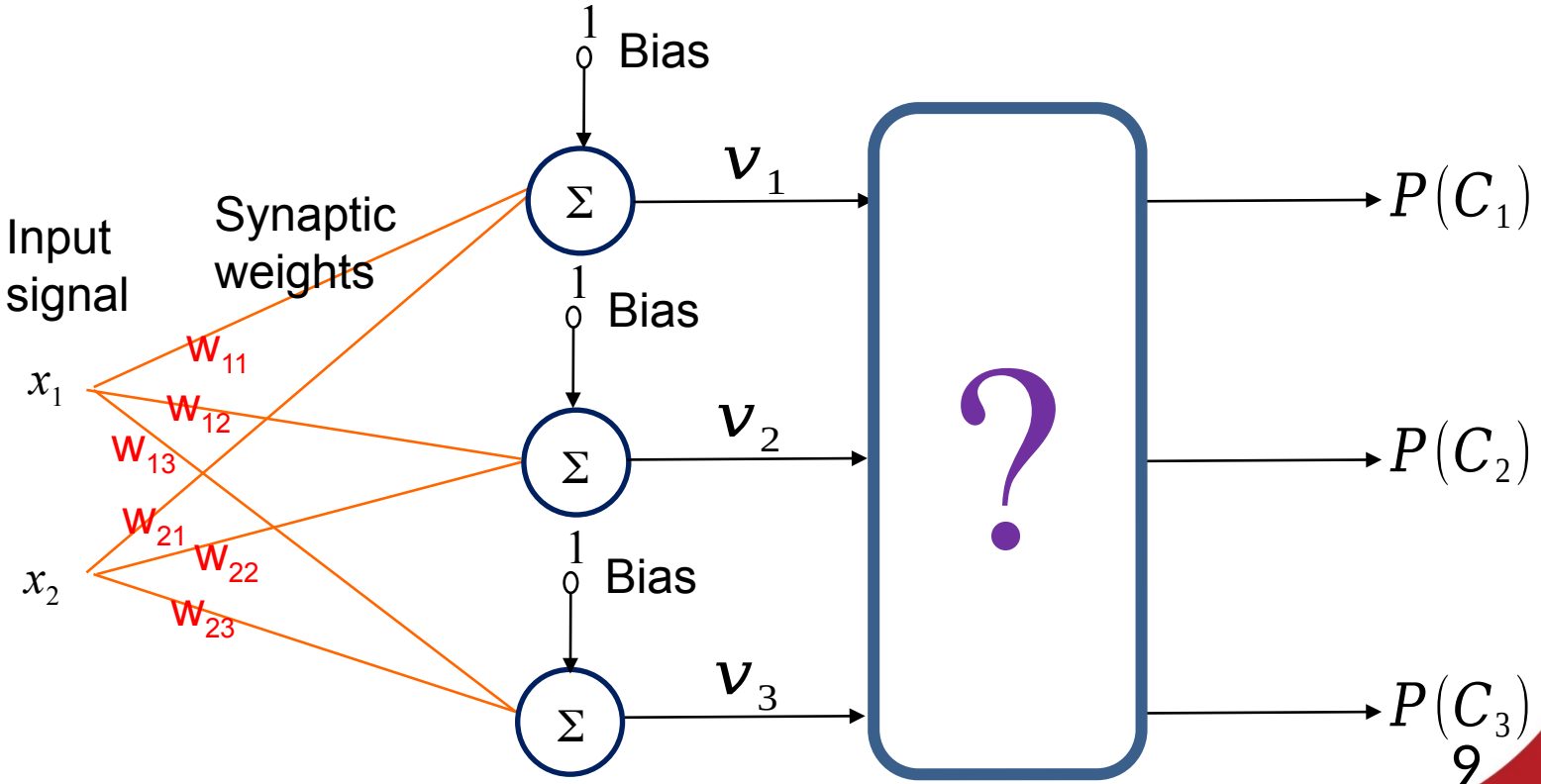
Multi-Class Classification Softmax

- So far, we used Sigmoid activation function to output a probability for a binary (2 class classification) classification
- What about multi-class classification?



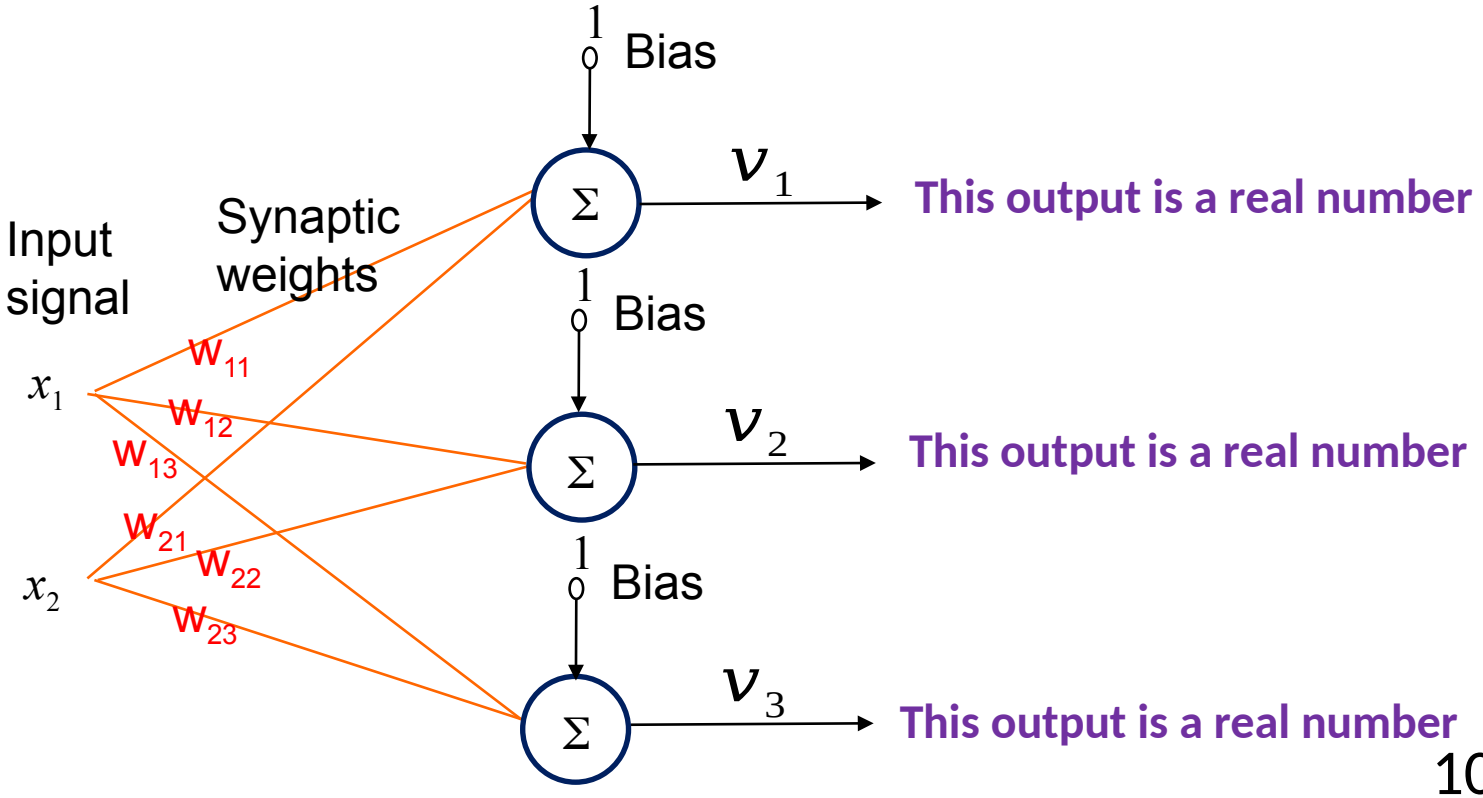
Multi-Class Classification Softmax

- So far, we used Sigmoid activation function to output a probability for a binary (2 class classification) classification
- What about multi-class classification?



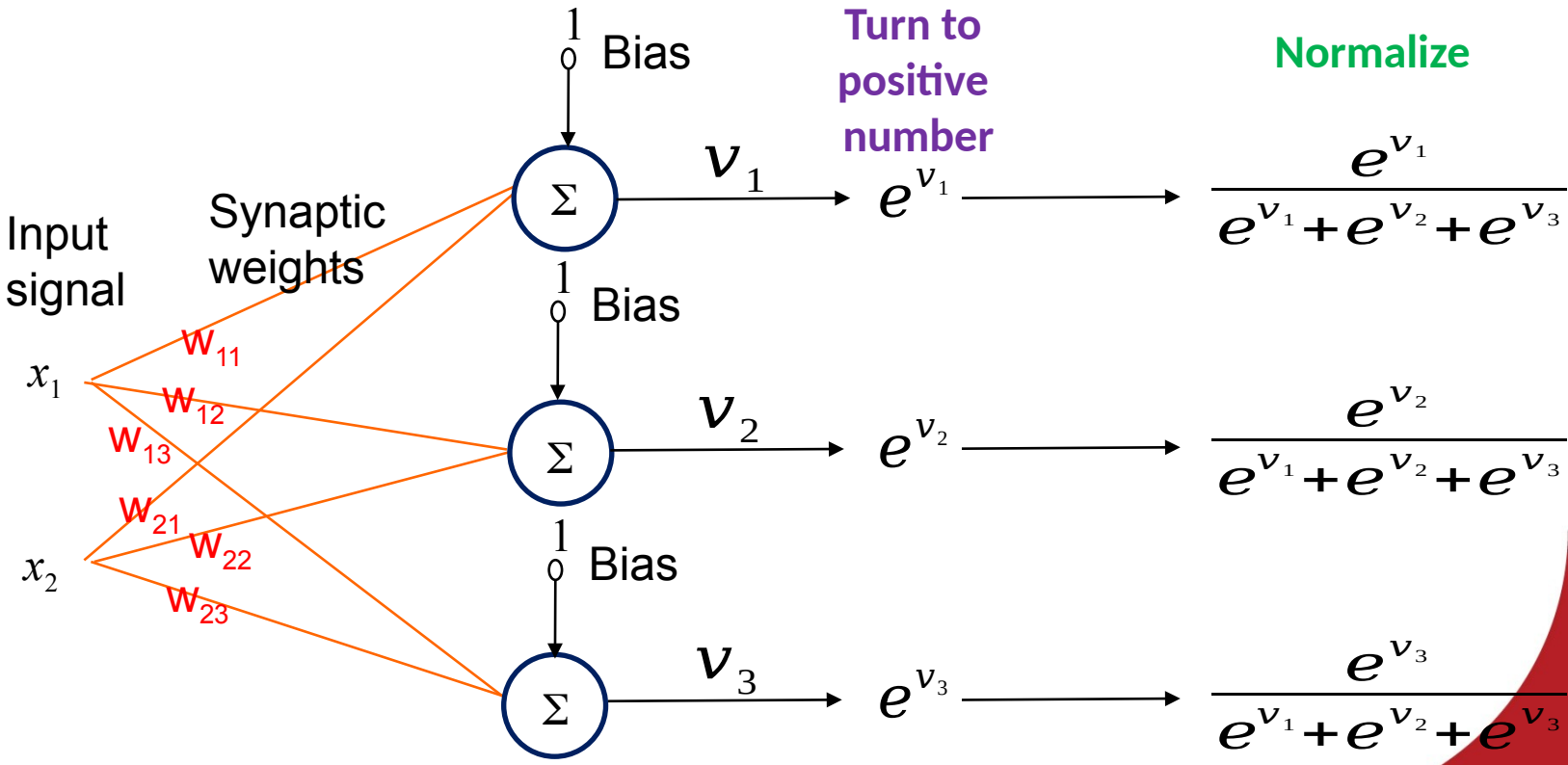
Softmax

- How to turn these 3 real numbers (and) into 3 probability values?
- Note that



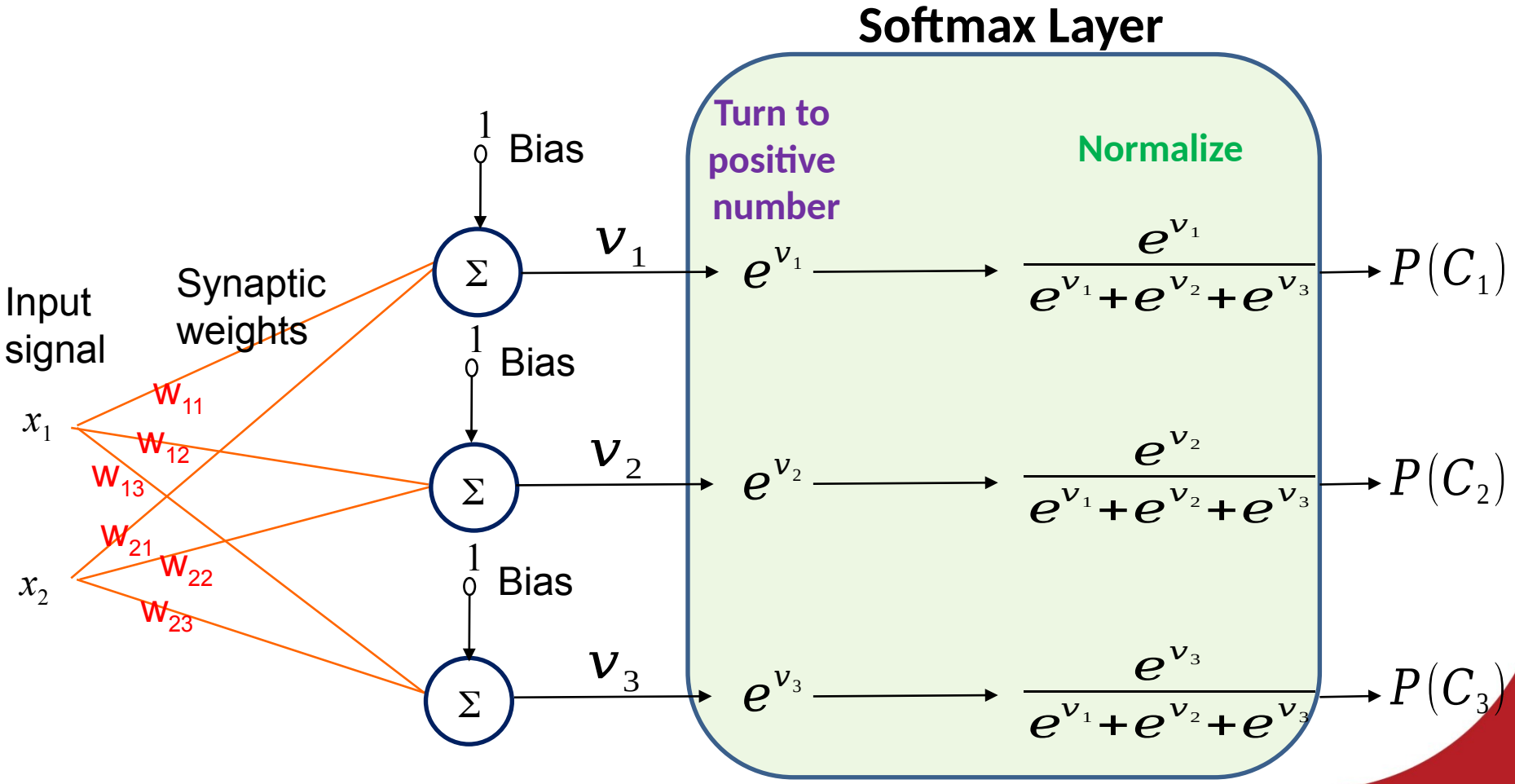
Softmax

- How to turn these 3 real numbers (and) into 3 probability values?
- Note that



Softmax

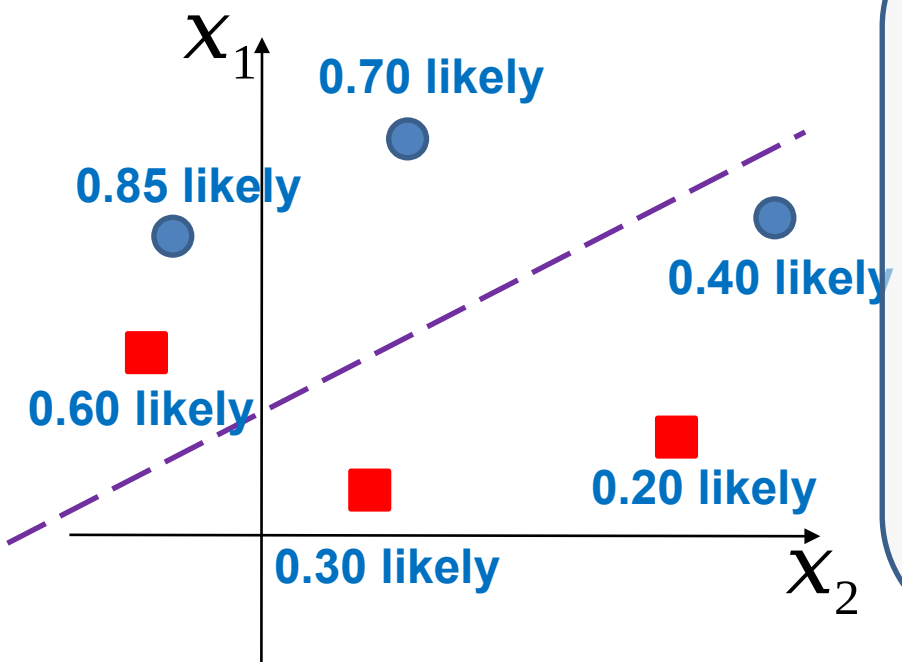
- Note that



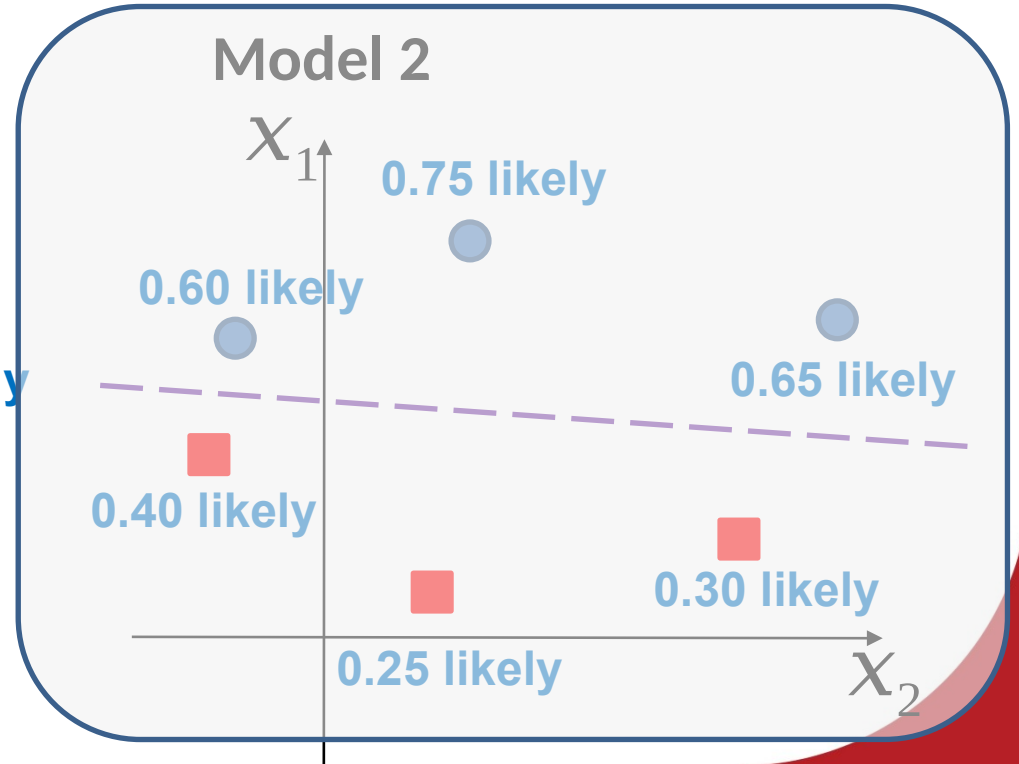
How to evaluate a model? (Error or Loss)

- Let's assume the models below. Which model is better?
- Metrics:
  - Mean Square Error (MSE)
  - Maximum Likelihood
  - Cross Entropy

Model 1



Model 2



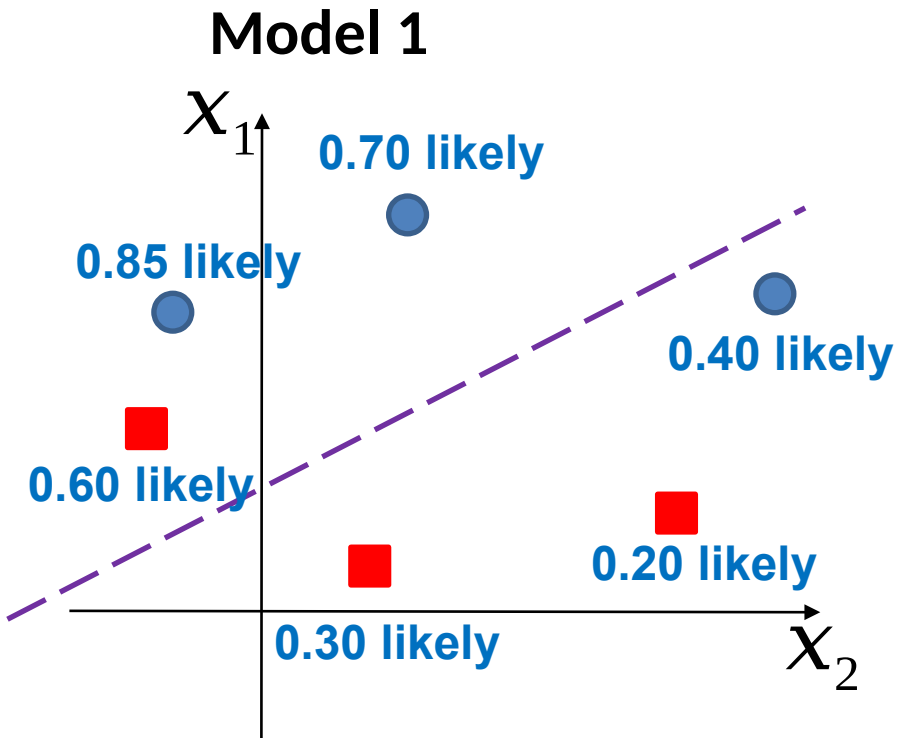
How to evaluate a model? (MSE)

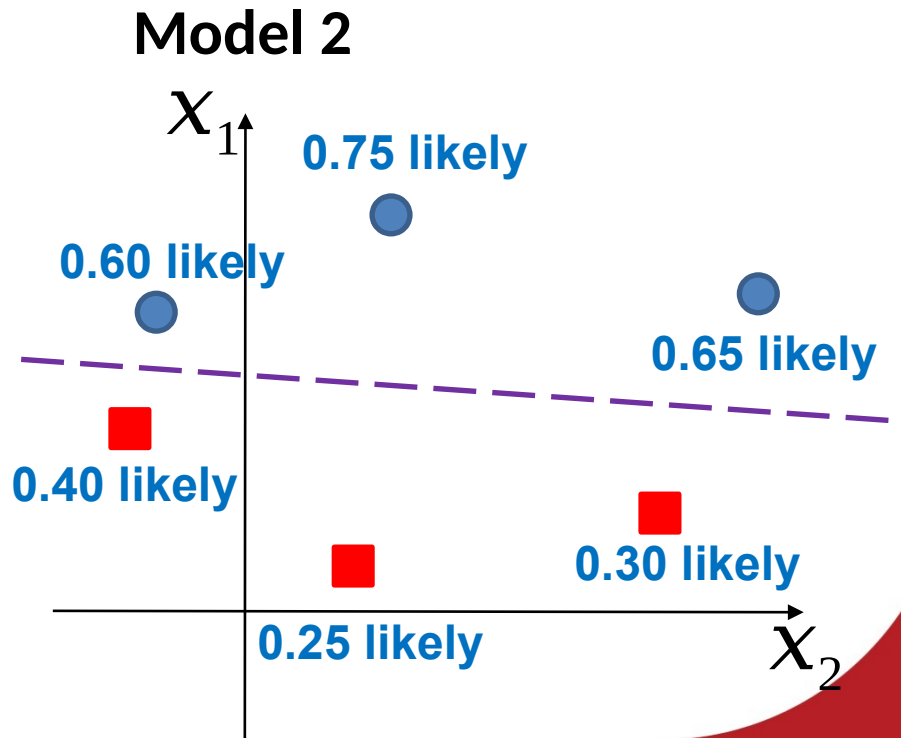
- The best model is a model that gives lower average squared difference between the predicted values and the actual values.

Number of Samples (e.g. 6)

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - P_i)^2$$

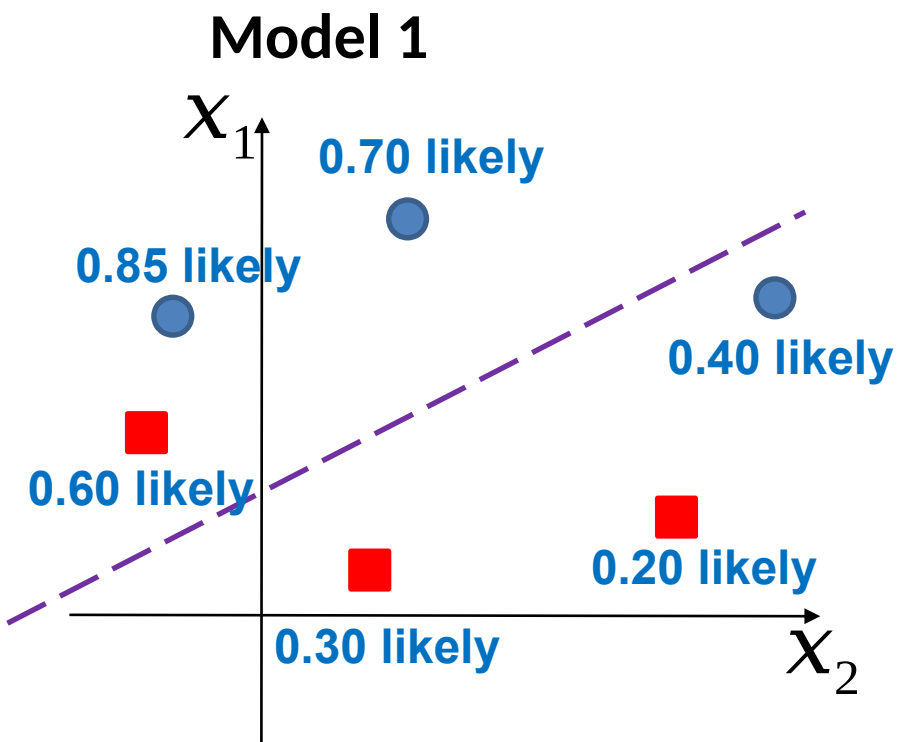
$$y_i = \begin{cases} 1 \\ 0 \end{cases}$$

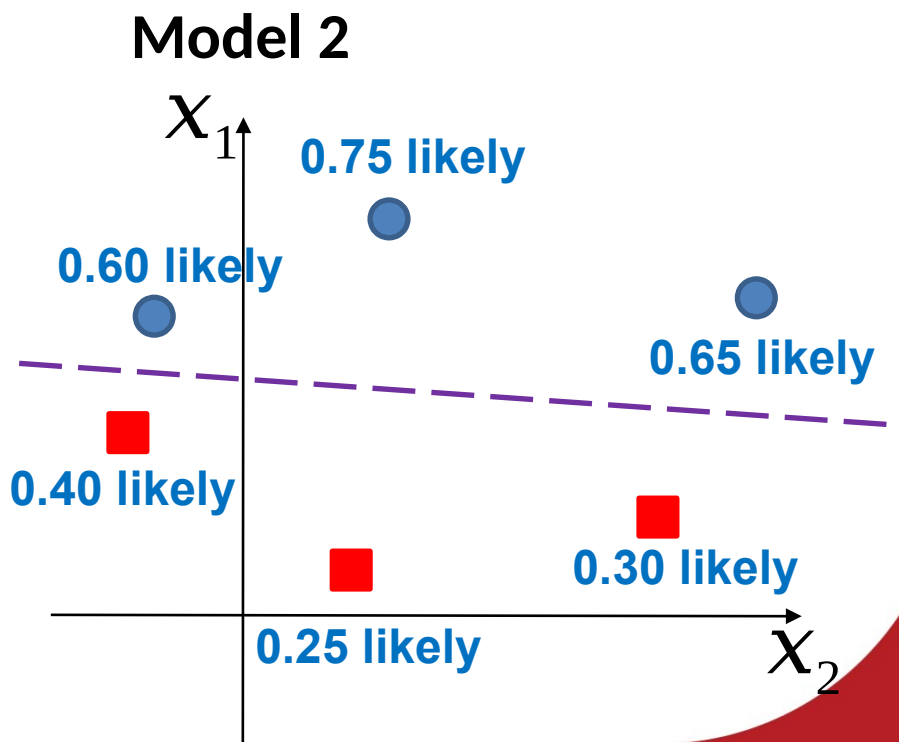




How to evaluate a model? (Maximum Likelihood)

- The best model is a model that gives higher probability to the correct class.
- The method is called **Maximum Likelihood**

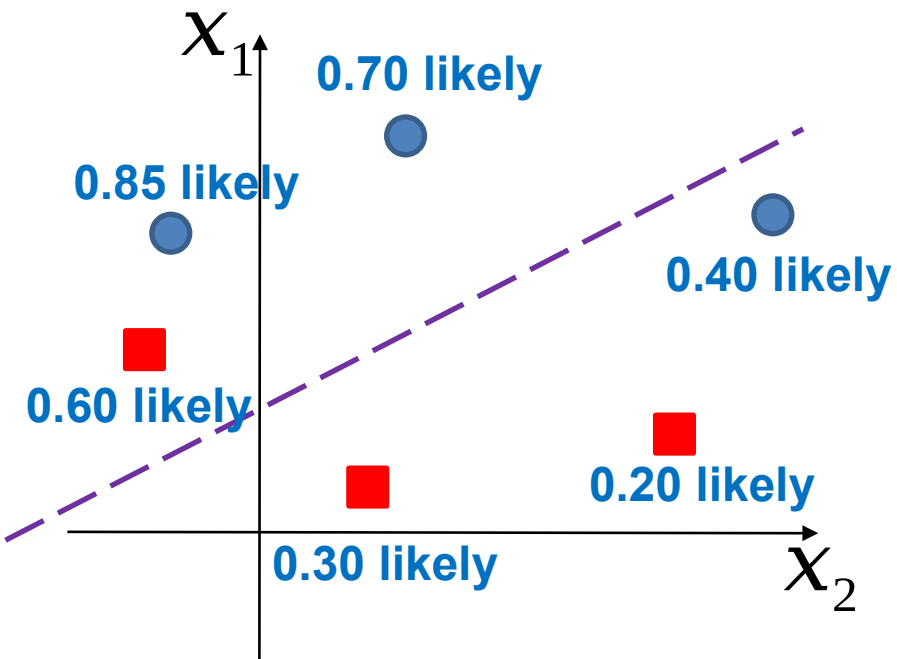




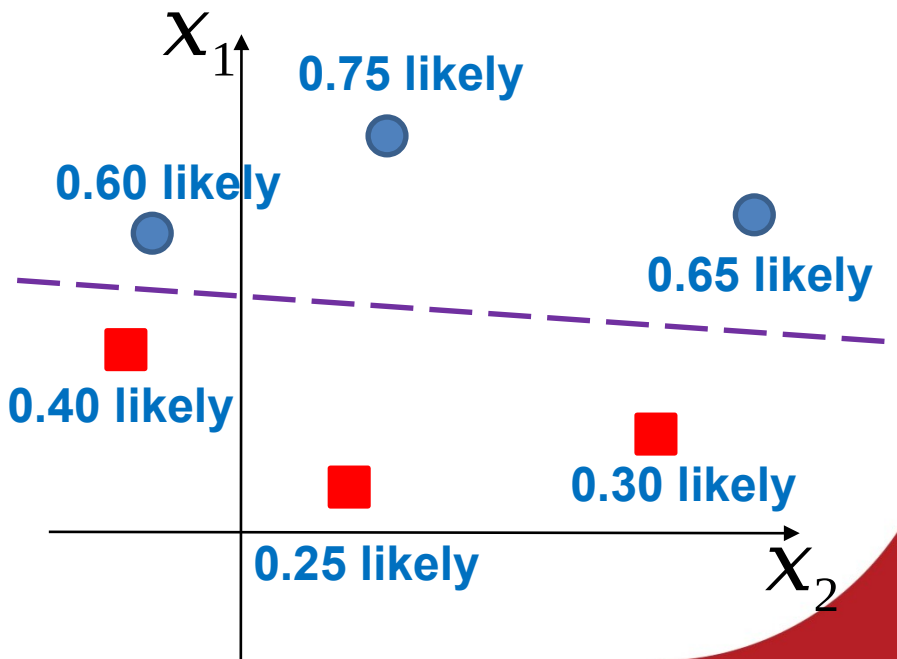
How to evaluate a model? (Maximum Likelihood)

- Why the model 2 is better from the probability perspective (**Maximum Likelihood**)?
- Below shows the probability of a sample be **BLUE** (which is the output of the Sigmoid activation function)

Model 1

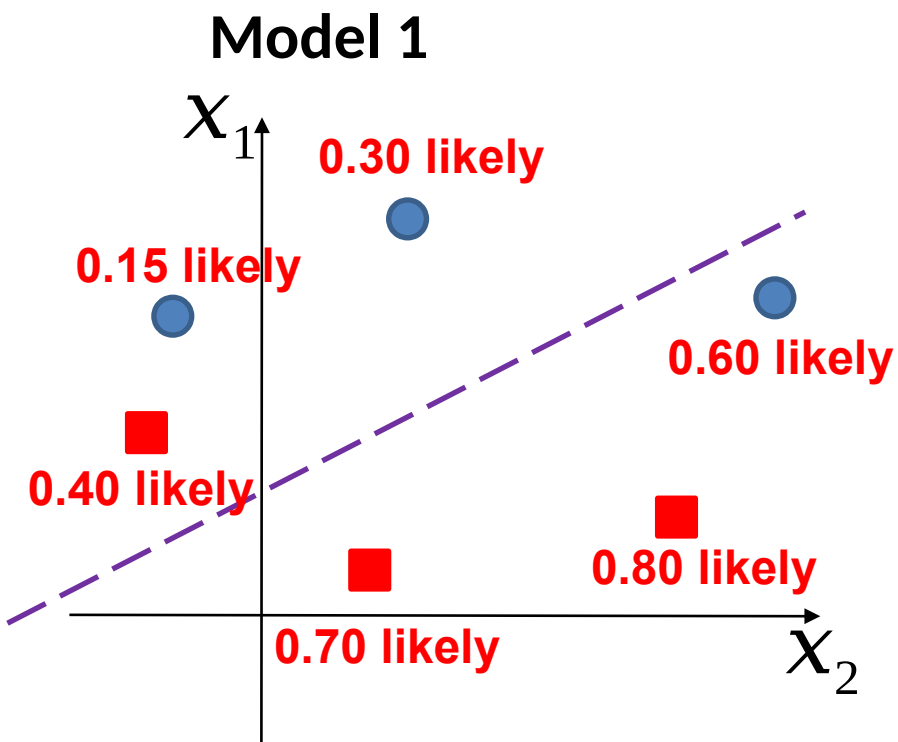


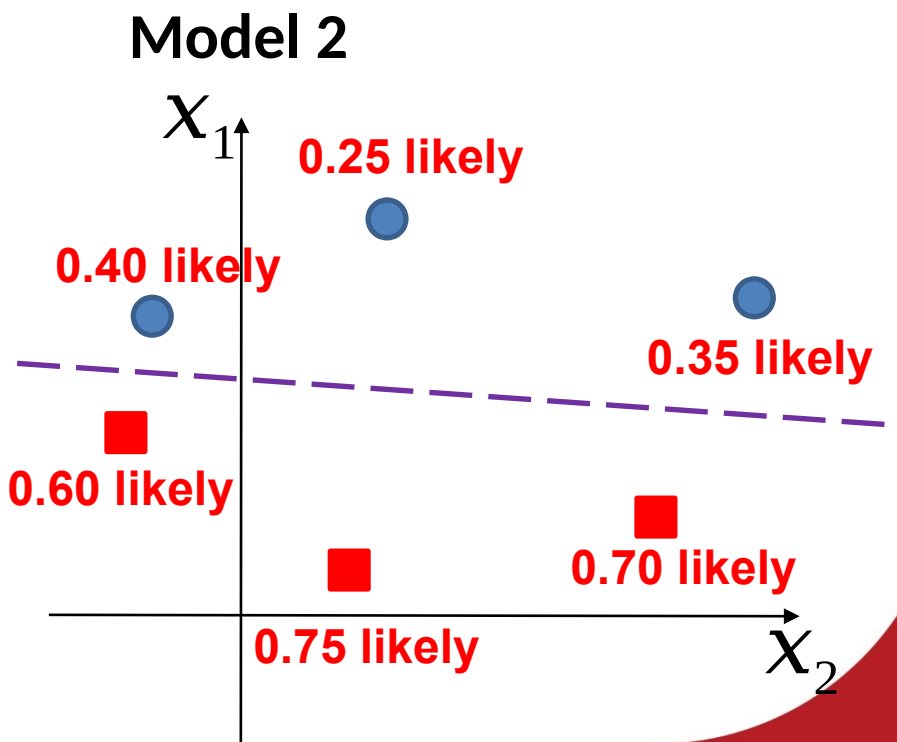
Model 2



How to evaluate a model? (Maximum Likelihood)

- Why the model 2 is better from the probability perspective (**Maximum Likelihood**)?
- Below shows the probability of a sample be **RED**

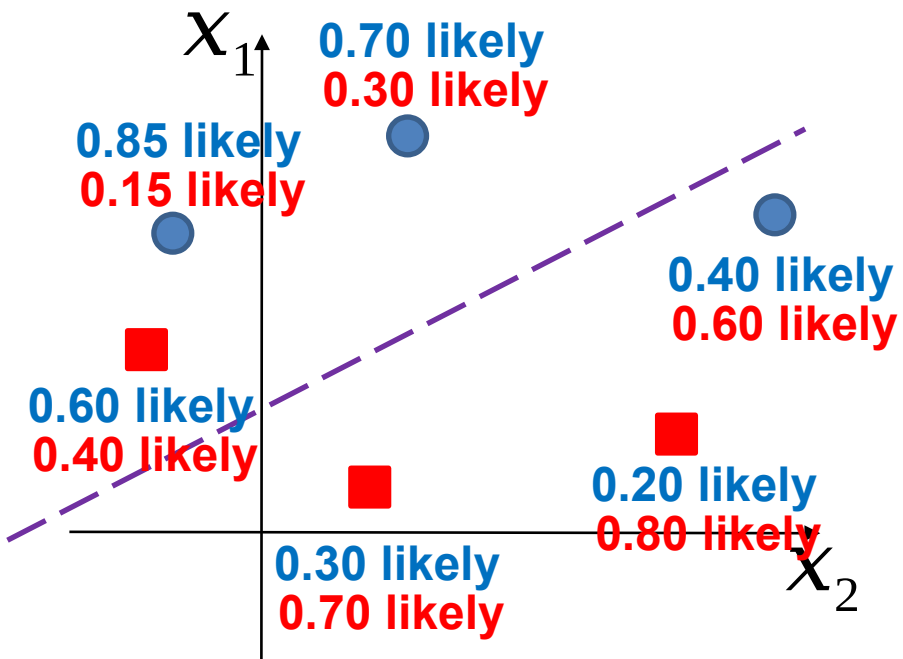




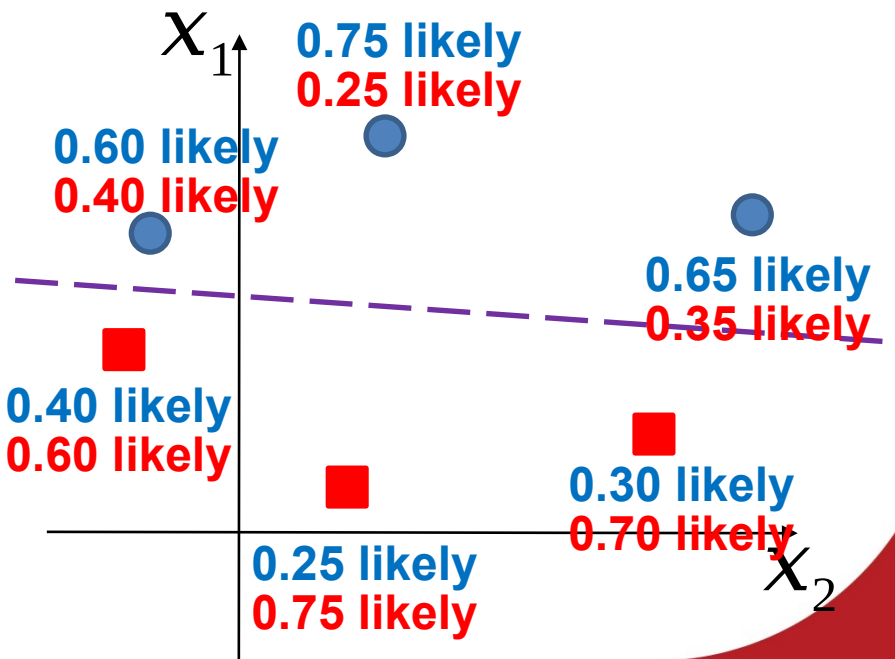
How to evaluate a model? (Maximum Likelihood)

- Why the model 2 is better from the probability perspective (**Maximum Likelihood**)?
- **Step 1:** Both probabilities have been shown for each sample for each model

Model 1



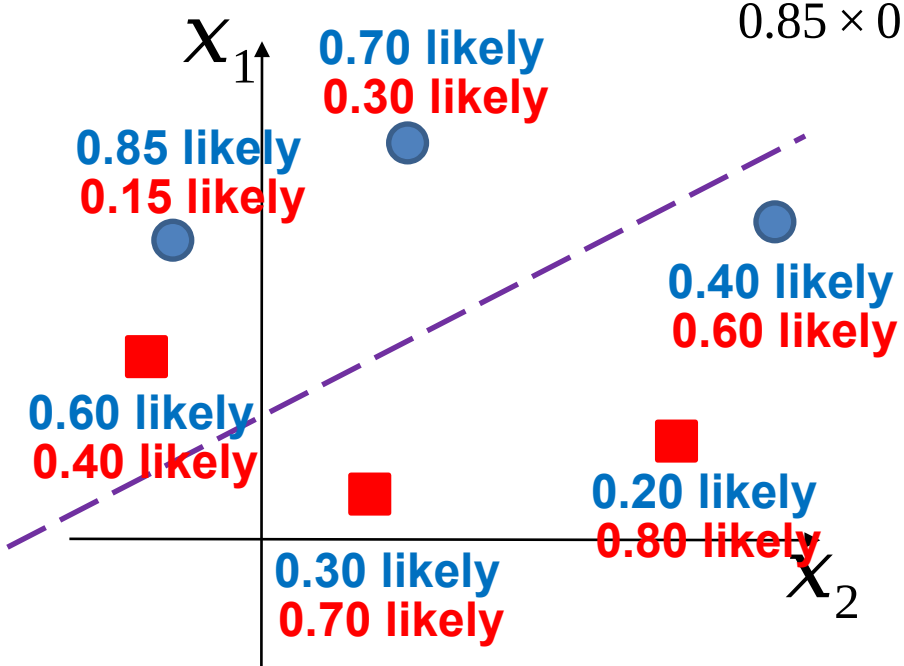
Model 2



How to evaluate a model? (Maximum Likelihood)

- **Step 1:** Both probabilities have been shown for each sample for each model
- **Step 2:** Calculate the probability of all 6 samples are of the classes that they actually are

Model 1



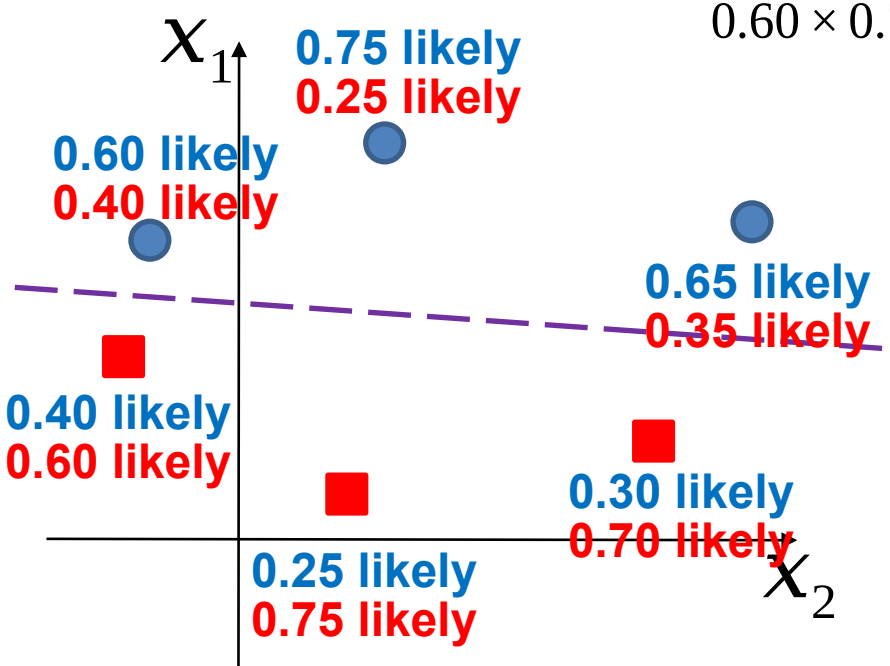
$$0.85 \times 0.70 \times 0.40 \times 0.70 \times 0.40 \times 0.80 = \mathbf{0.053312}$$

Probability for Model 1

How to evaluate a model? (Maximum Likelihood)

- **Step 1:** Both probabilities have been shown for each sample for each model
- **Step 2:** Calculate the probability of all 6 samples are of the classes that they actually are

Model 2



$$0.60 \times 0.75 \times 0.65 \times 0.60 \times 0.75 \times 0.70 = \mathbf{0.0921375}$$

Probability for Model 2

How to evaluate a model? (Maximum Likelihood)

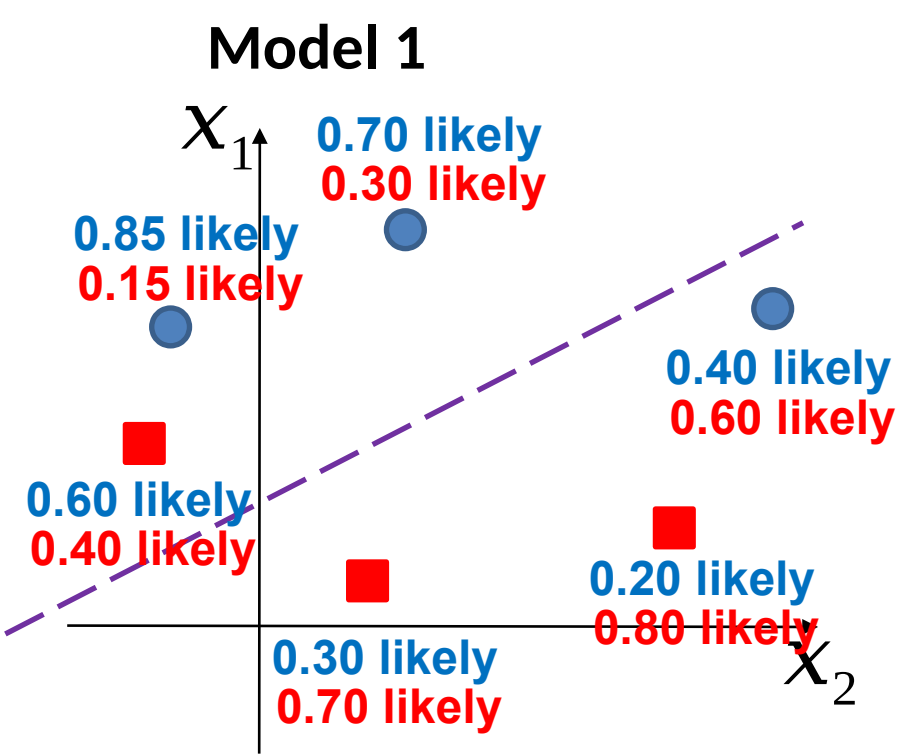
Lancaster University

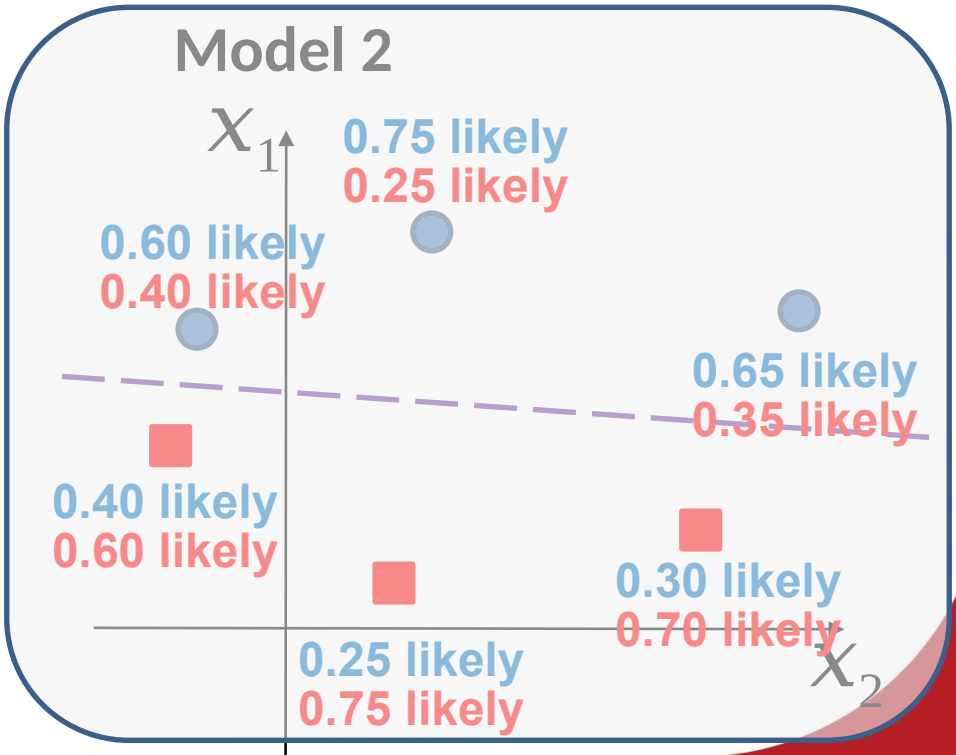
Probability for Model 1

0.053312

Probability for Model 2

0.0921375





How to learn a model? (Maximum Likelihood)

Lancaster University

- So, the aim of learning (for classification) is to maximize this probability of the model
- How to maximize this probability?
  - Product of probabilities (one probability for each sample)
  - It is **hard** to maximize products
  - Moreover, if there are hundreds/thousands of samples, the product of them is very tiny (e.g. 0.000000000003)
  - **Solution:** We need to resolve these issues **by turning the products to sums!!!**

How to learn a model? (Maximum Likelihood => Cross-Entropy)

Lancaster University

$$0.85 \times 0.70 \times 0.40 \times 0.70 \times 0.40 \times 0.80 = \mathbf{0.053312}$$

Likelihood of Model

$$\ln(0.85 \times 0.70 \times 0.40 \times 0.70 \times 0.40 \times 0.80) = \ldots$$

$$\ldots = \ln(0.85) + \ln(0.70) + \ln(0.40) + \ln(0.70) + \ln(0.40) + \ln(0.80) = ¿$$

$$\ldots = (-0.16) + (-0.36) + (-0.92¿ + (-0.36) + (-0.92) + (-0.22) = -2.94$$

Result is always a negative number

$$\ln A \times B = \ln A + \ln B$$

$$\ldots = -\ln(0.85) - \ln(0.70) - \ln(0.40) - \ln(0.70) - \ln(0.40) - \ln(0.80) = ¿$$

$$\ldots = -(-0.16) - (-0.36) - (-0.92¿ - (-0.36) - (-0.92) - (-0.22) = 2.94$$

Result is always a positive number

This negative sum of the log probabilities is called

Cross-Entropy

How to learn a model? (Maximum Likelihood => Cross-Entropy)

Lancaster University

$$0.60 \times 0.75 \times 0.65 \times 0.60 \times 0.75 \times 0.70 = \mathbf{0.0921375}$$

Likelihood of Model

$$-\ln(0.60 \times 0.75 \times 0.65 \times 0.60 \times 0.75 \times 0.70) = \ldots$$

$$\ldots = -\ln(0.60) - \ln(0.75) - \ln(0.65) - \ln(0.60) - \ln(0.75) - \ln(0.70) = \ldots$$

$$\ldots = -(-0.51) - (-0.29) - (-0.43) - (-0.51) - (-0.29) - (-0.36) = 2.39$$

$$\ln A \times B = \ln A + \ln B$$

This negative sum of the log probabilities is called Cross-Entropy

How to learn a model? (Cross-Entropy)

Lancaster University

- The aim of learning (for classification) is to **maximize the Maximum Likelihood** of the model
- Now, our aim is to **minimize the Cross-Entropy** of the model

$$0.85 \times 0.70 \times 0.40 \times 0.70 \times 0.40 \times 0.80 = \mathbf{0.053312}$$

Likelihood of Model 1

$$-\ln(0.85) - \ln(0.70) - \ln(0.40) - \ln(0.70) - \ln(0.40) - \ln(0.80) = \mathbf{2.94}$$

Cross-Entropy of Model 1

$$0.60 \times 0.75 \times 0.65 \times 0.60 \times 0.75 \times 0.70 = \mathbf{0.0921375}$$

Likelihood of Model 2

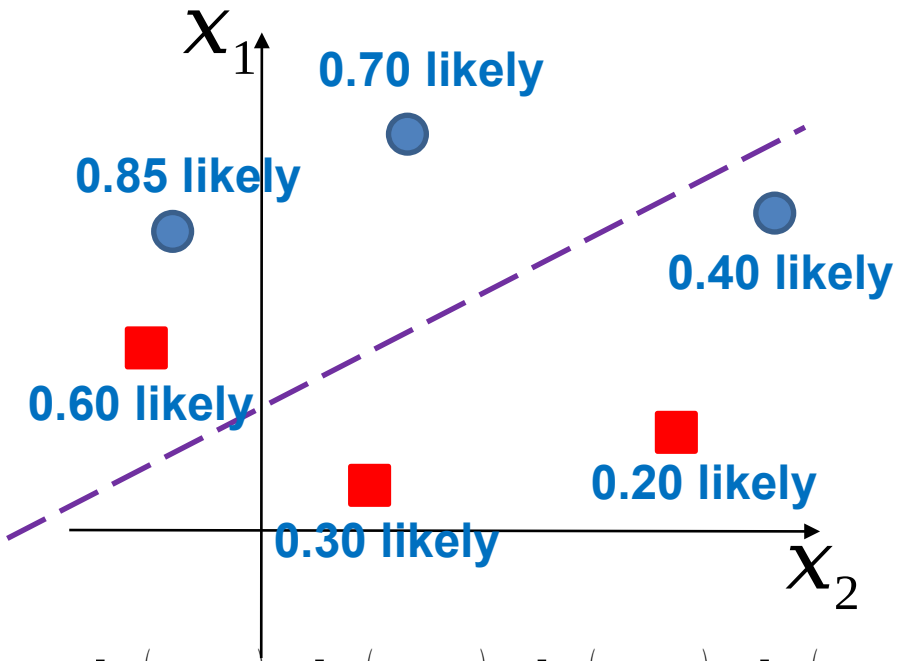
$$-\ln(0.60) - \ln(0.75) - \ln(0.65) - \ln(0.60) - \ln(0.75) - \ln(0.70) = \mathbf{2.39}$$

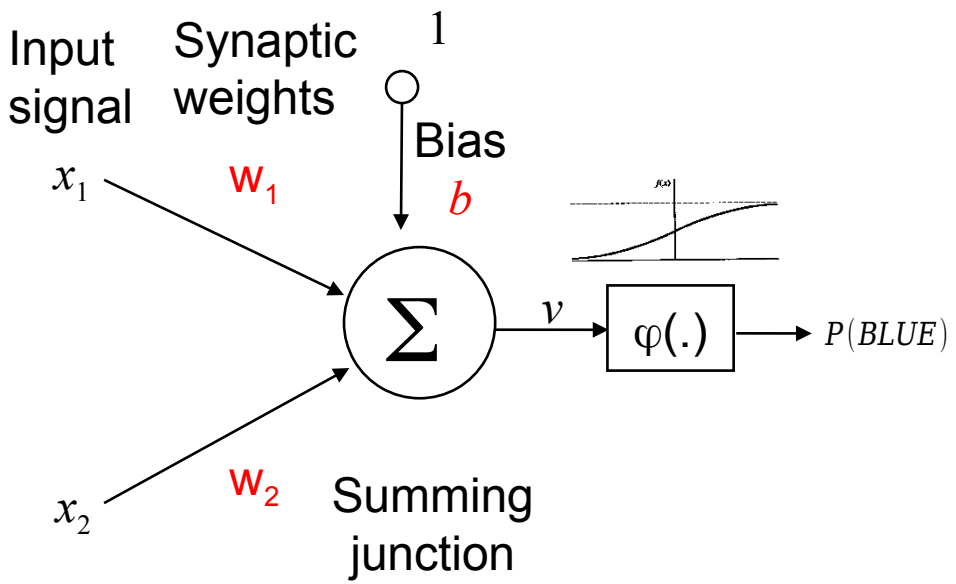
Cross-Entropy of Model 2

How to learn a model? (Cross-Entropy Equation)

Lancaster University

Model 1





$$-\ln(0.85)-\ln(0.70)-\ln(0.40)-\ln(0.70)-\ln(0.40)-\ln(0.80)=\mathbf{2.94}$$

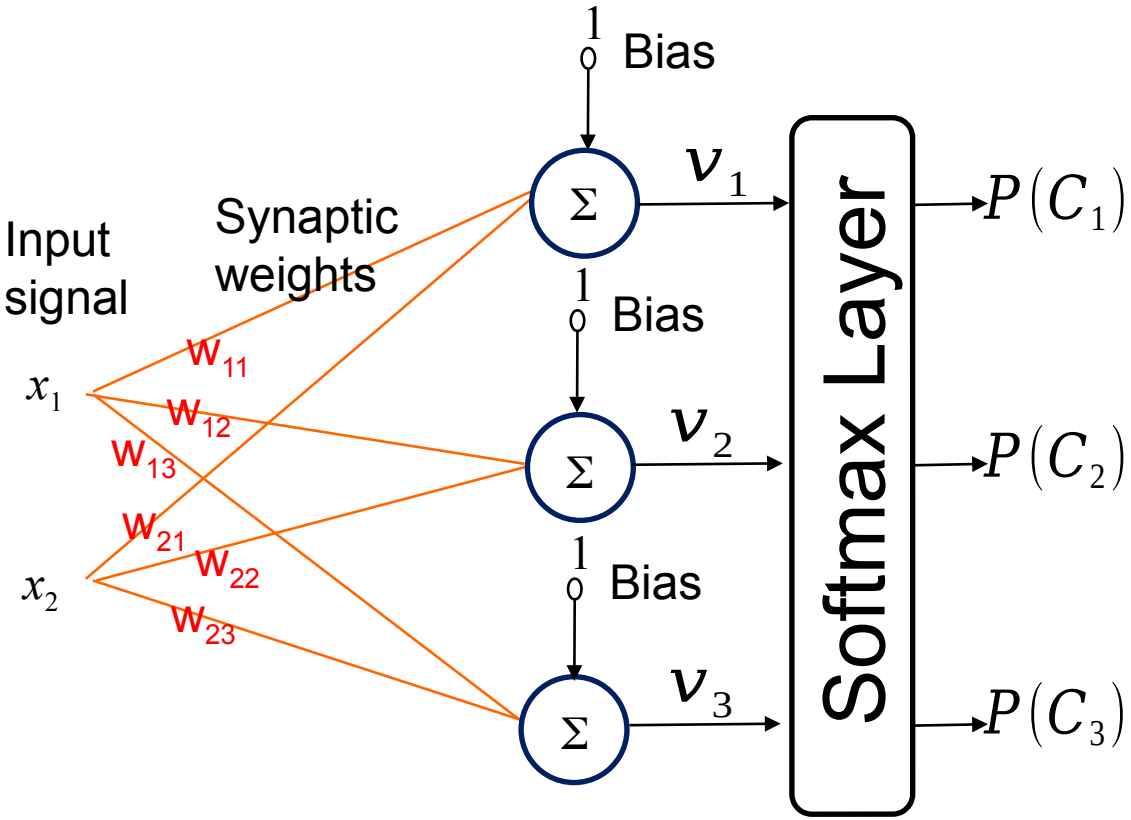
Cross-Entropy of Model 1

Number of Samples (e.g. 6)

$$-\sum_{i=1}^{N} y_i \ln(P_i)+(1-y_i)\ln(1-P_i)$$

$$y_i=\begin{cases}1\\0\end{cases}$$

How to learn a model? (Multi-Class Cross-Entropy)



Number of Samples

Number of Classes

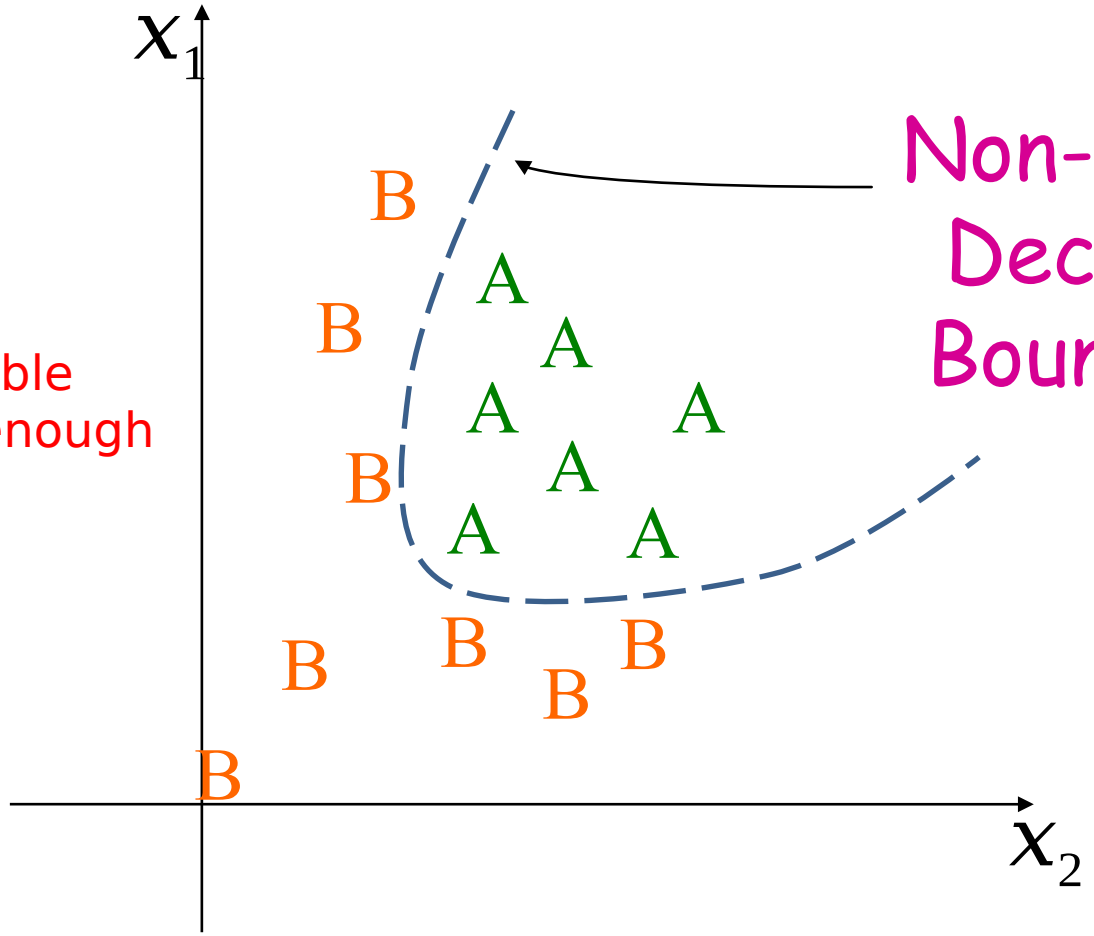
$$-\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j}\ln\left(P_{i,j}\right)$$

If sample belongs to class , then:

Non-linear Decision Boundary

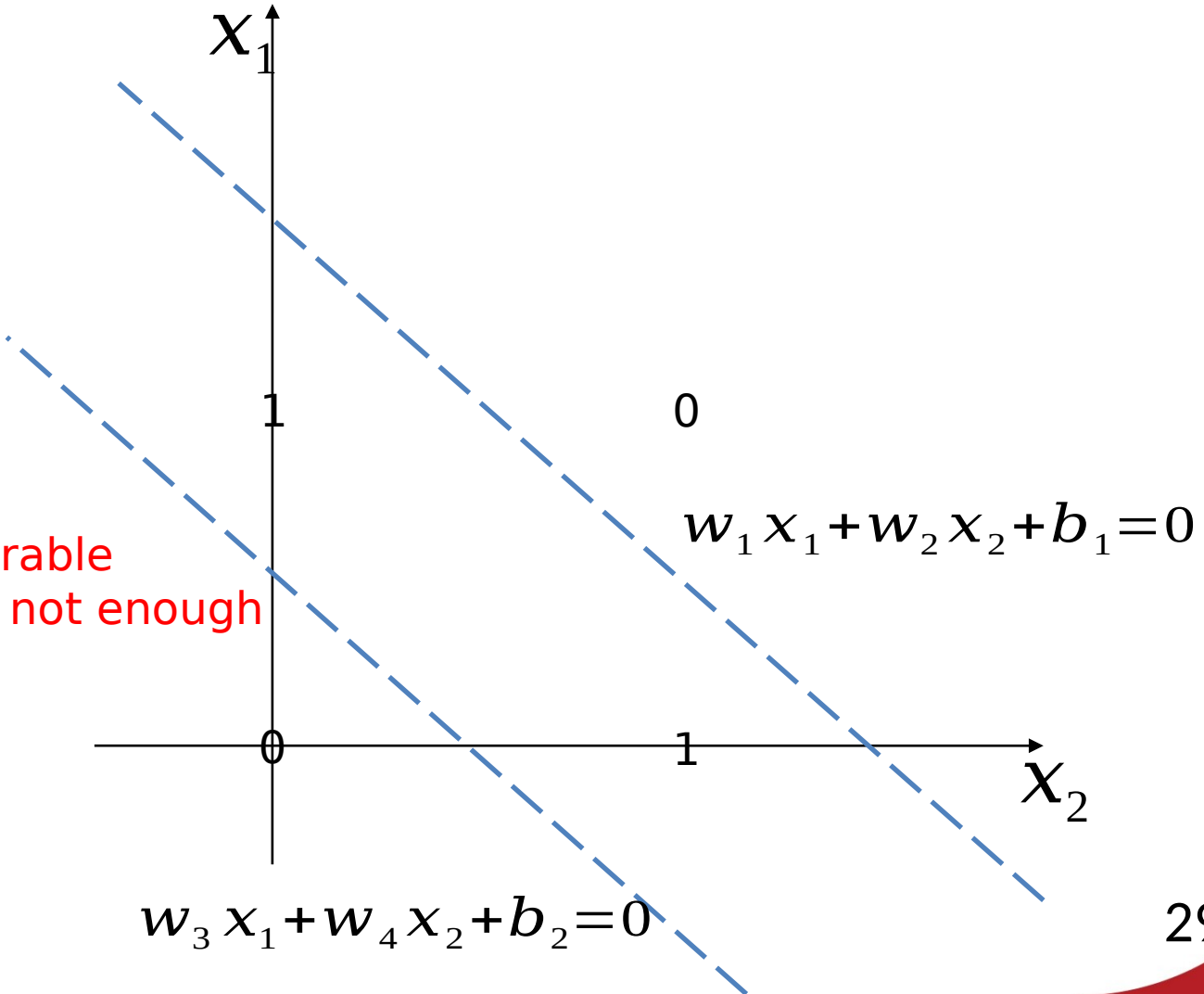
Lancaster University

Non-Linearly separable
One perceptron is not enough



Simple Network (XOR)

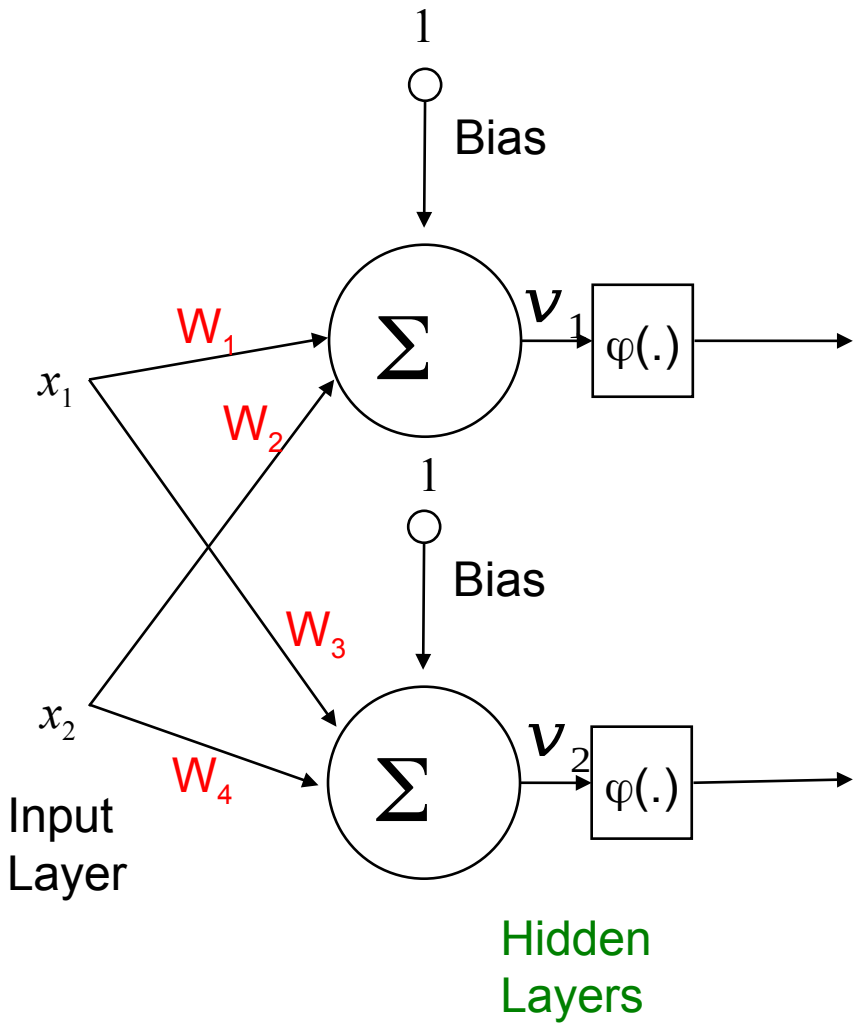
XOR		
		out
0	0	0
0	1	1
1	0	1
1	1	0



Non-Linearly separable
One perceptron is not enough

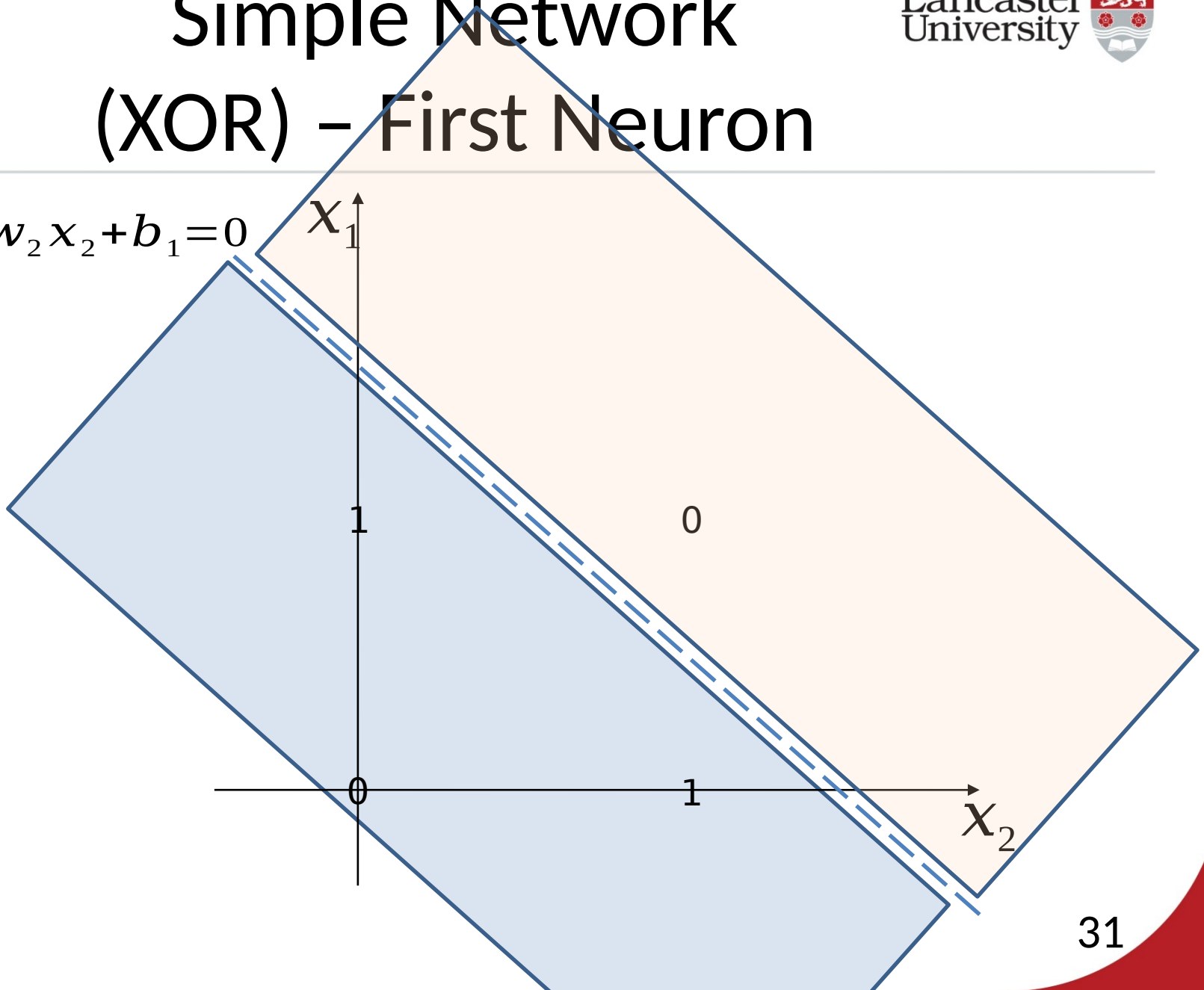
Simple Network (XOR)

XOR		
		out
0	0	0
0	1	1
1	0	1
1	1	0



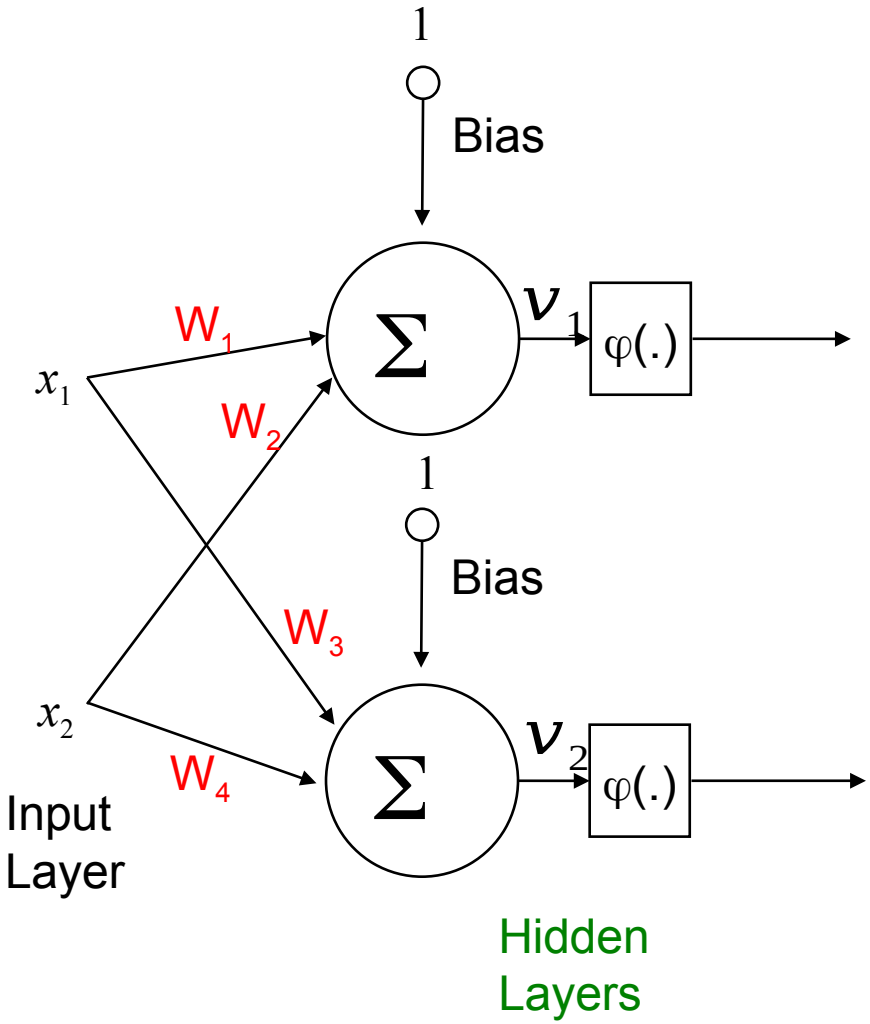
Simple Network (XOR) – First Neuron

$$w_1 x_1 + w_2 x_2 + b_1 = 0$$

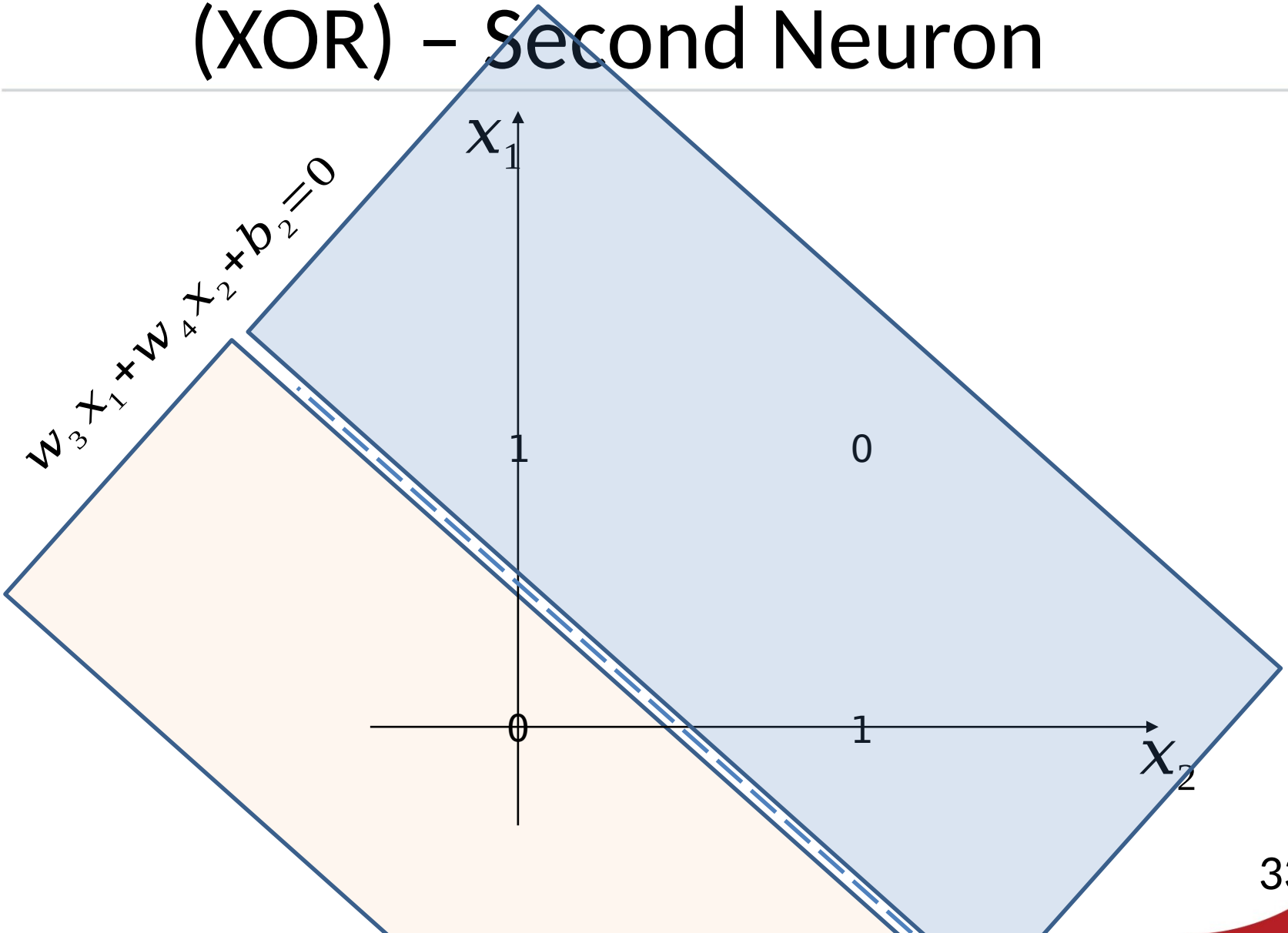


Simple Network (XOR)

XOR		
		out
0	0	0
0	1	1
1	0	1
1	1	0

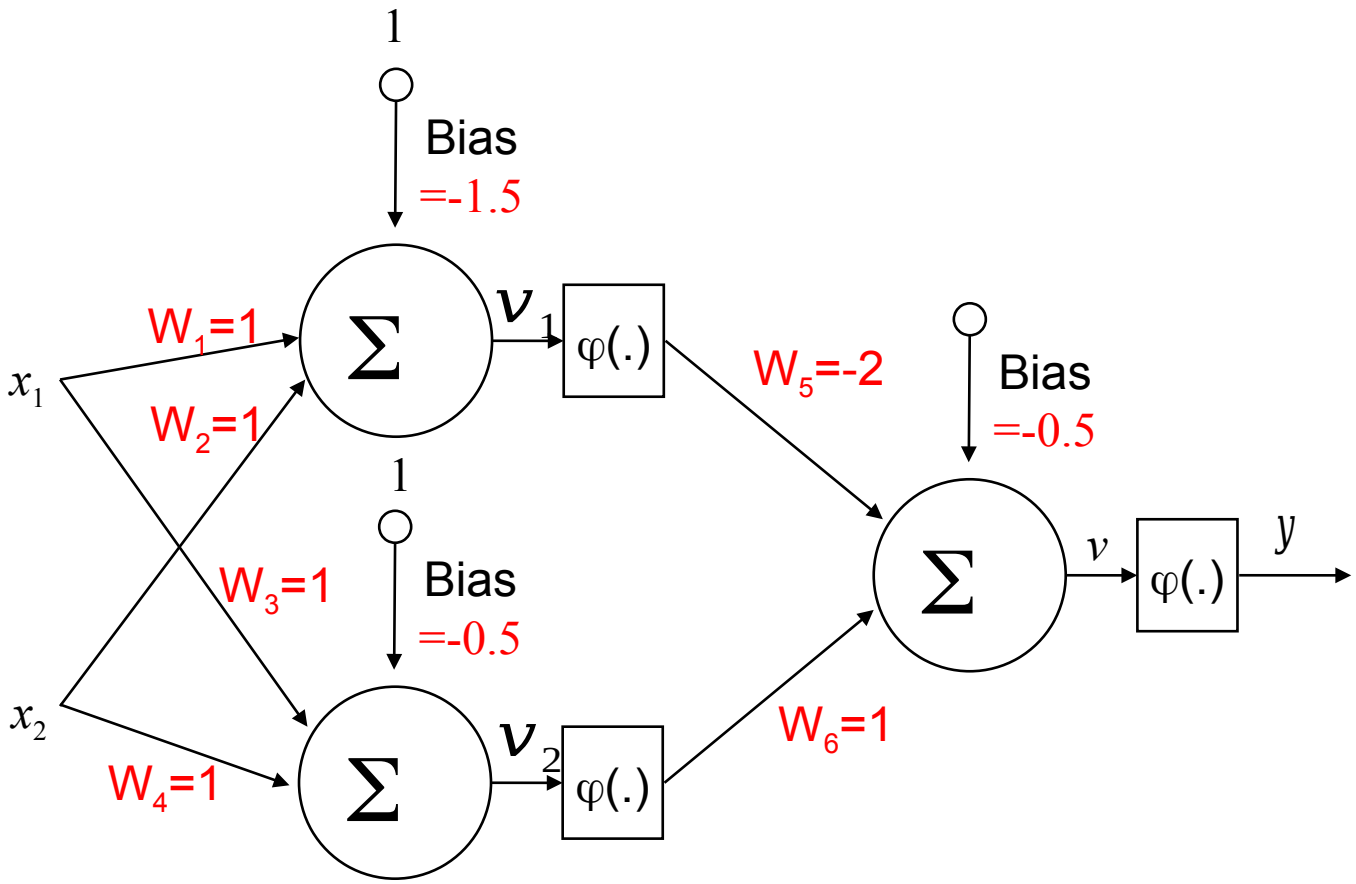


Simple Network (XOR) – Second Neuron

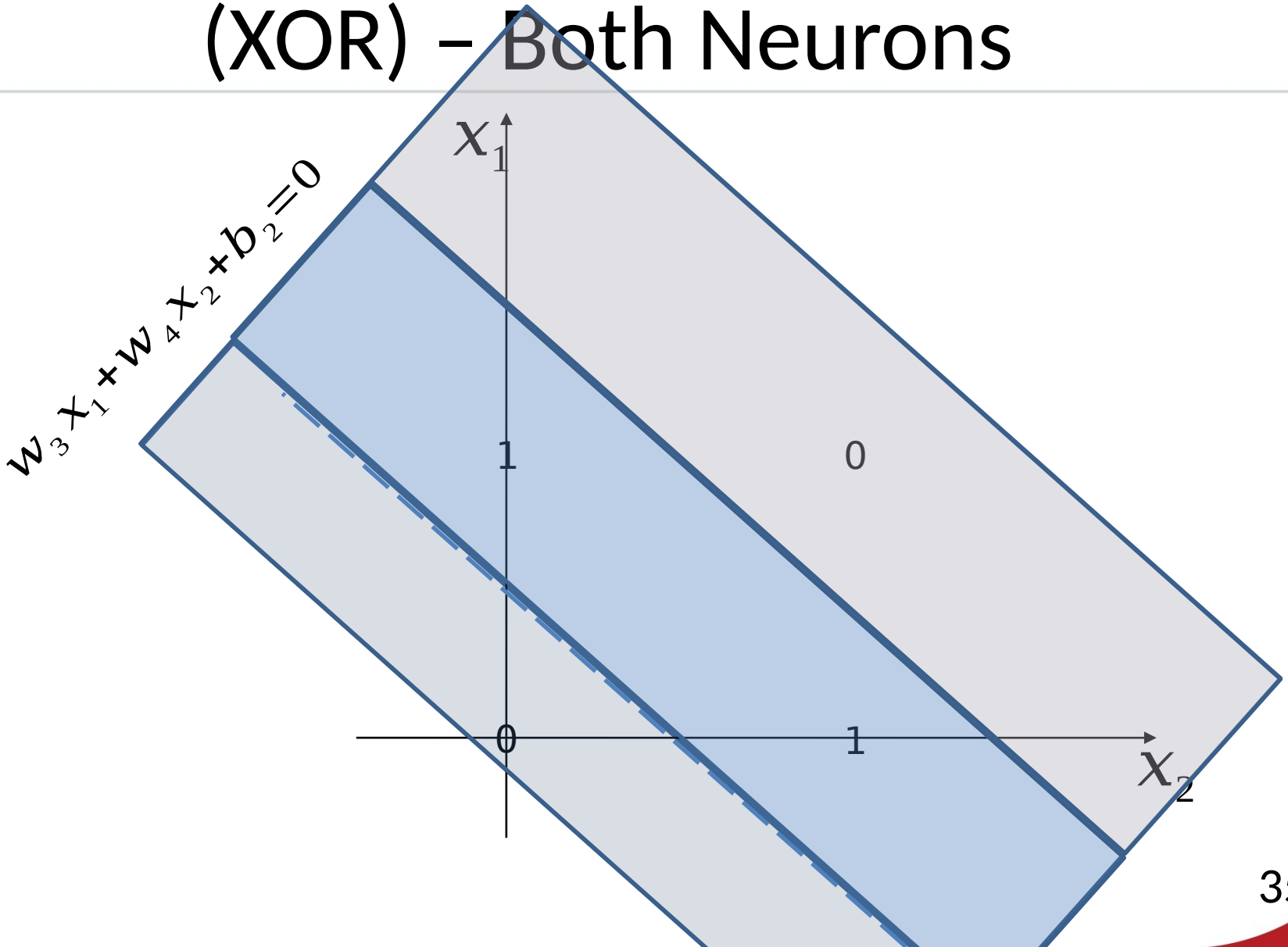


Simple Network (XOR)

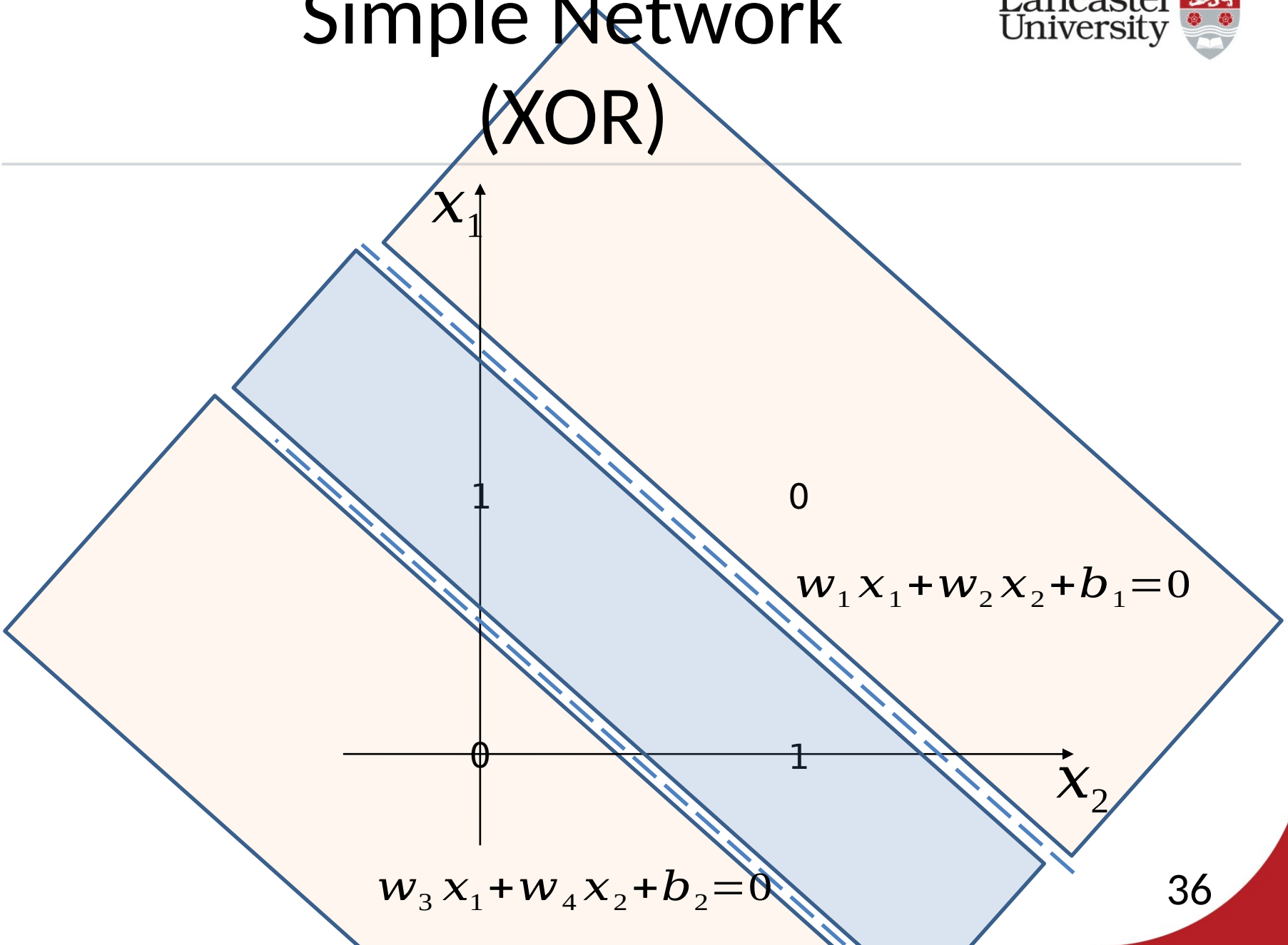
XOR		
		out
0	0	0
0	1	1
1	0	1
1	1	0



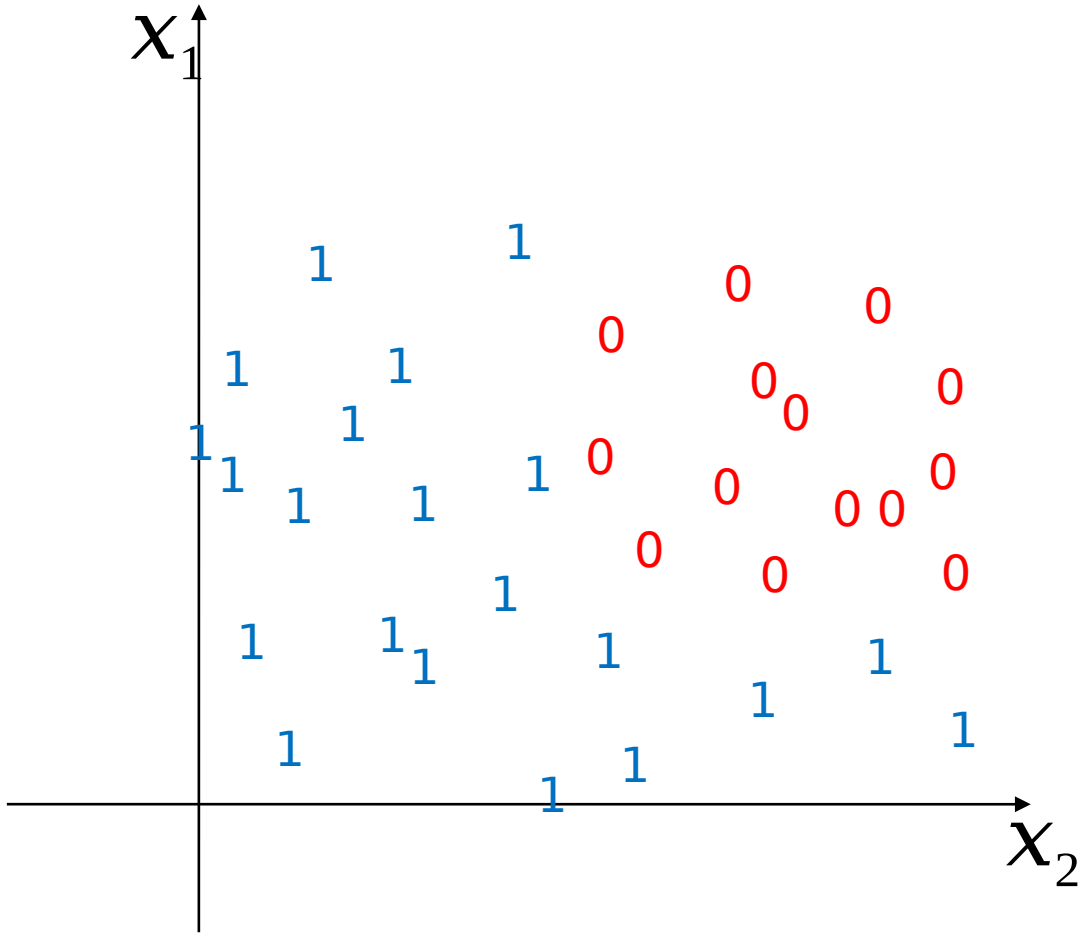
Simple Network (XOR) – Both Neurons



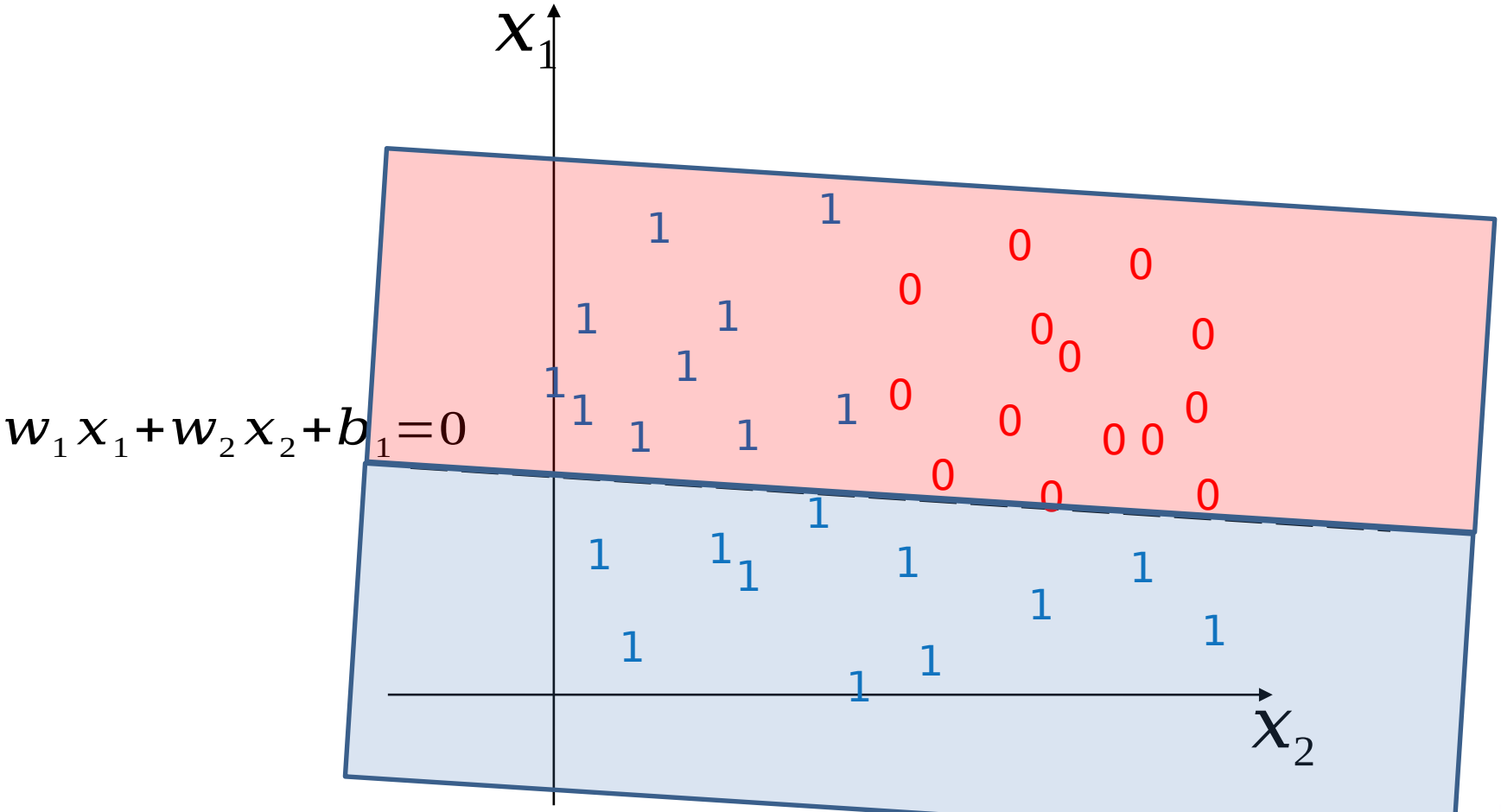
Simple Network (XOR)



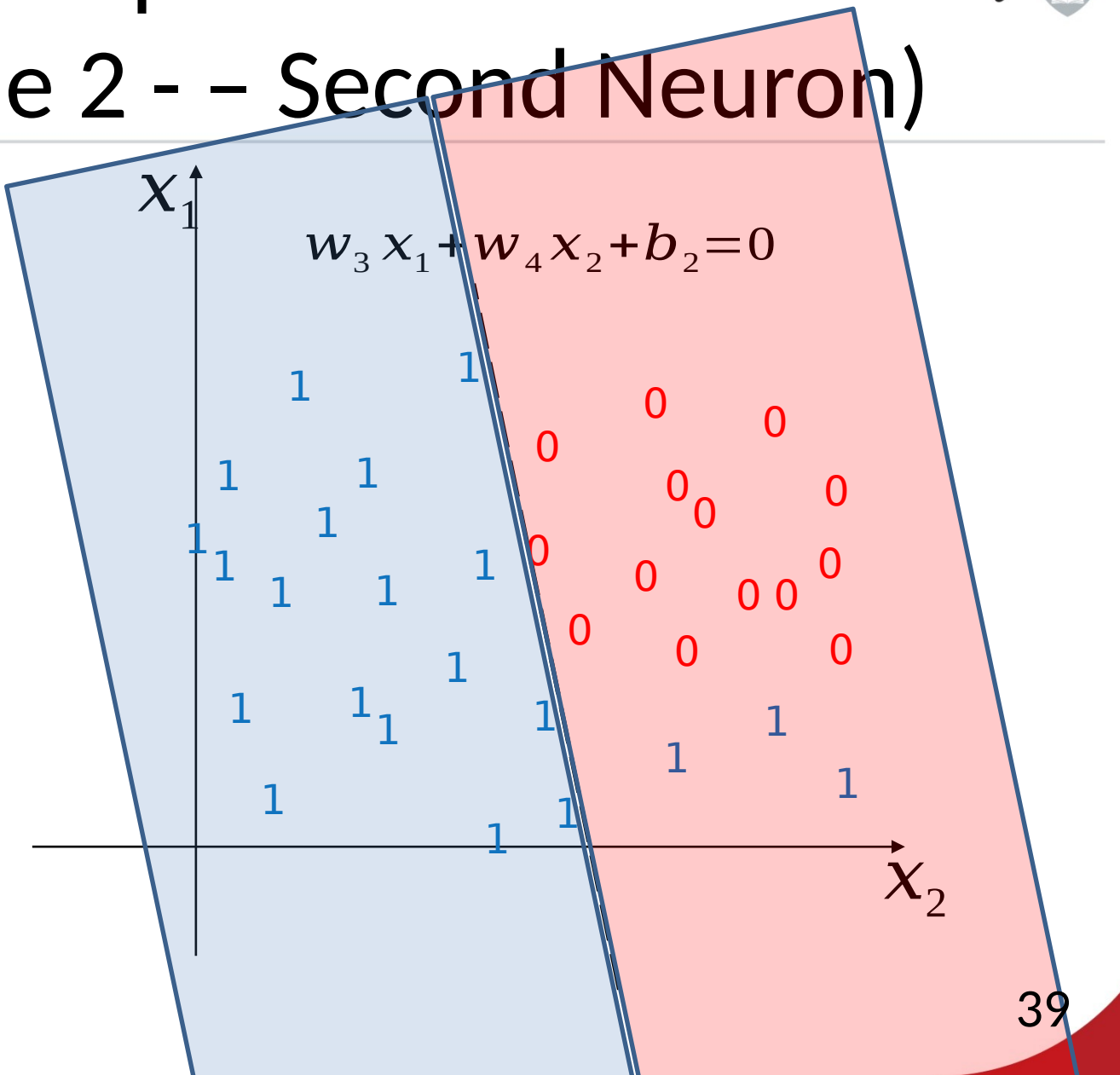
Simple Network (Example 2)



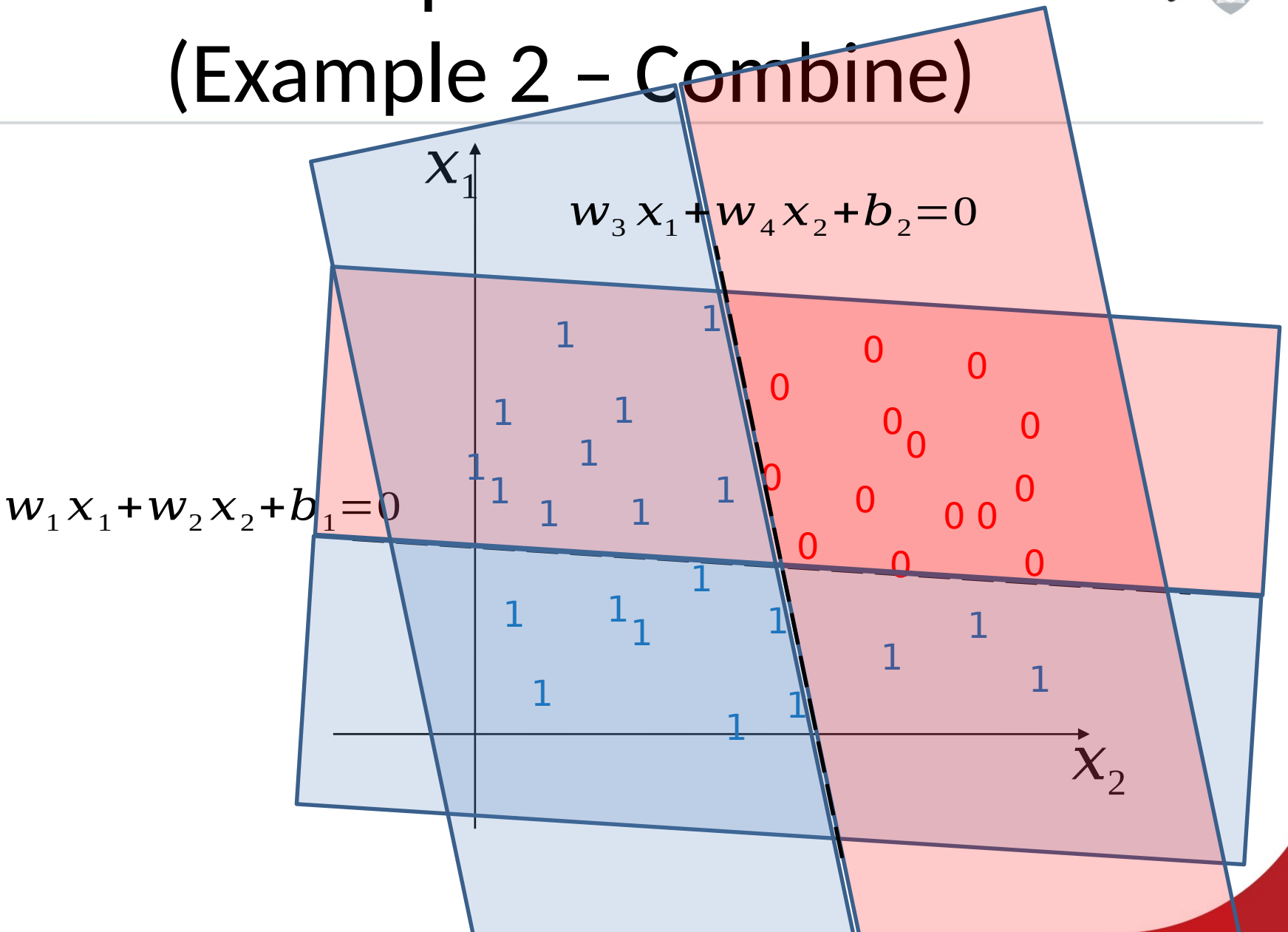
Simple Network (Example 2 – First Neuron)



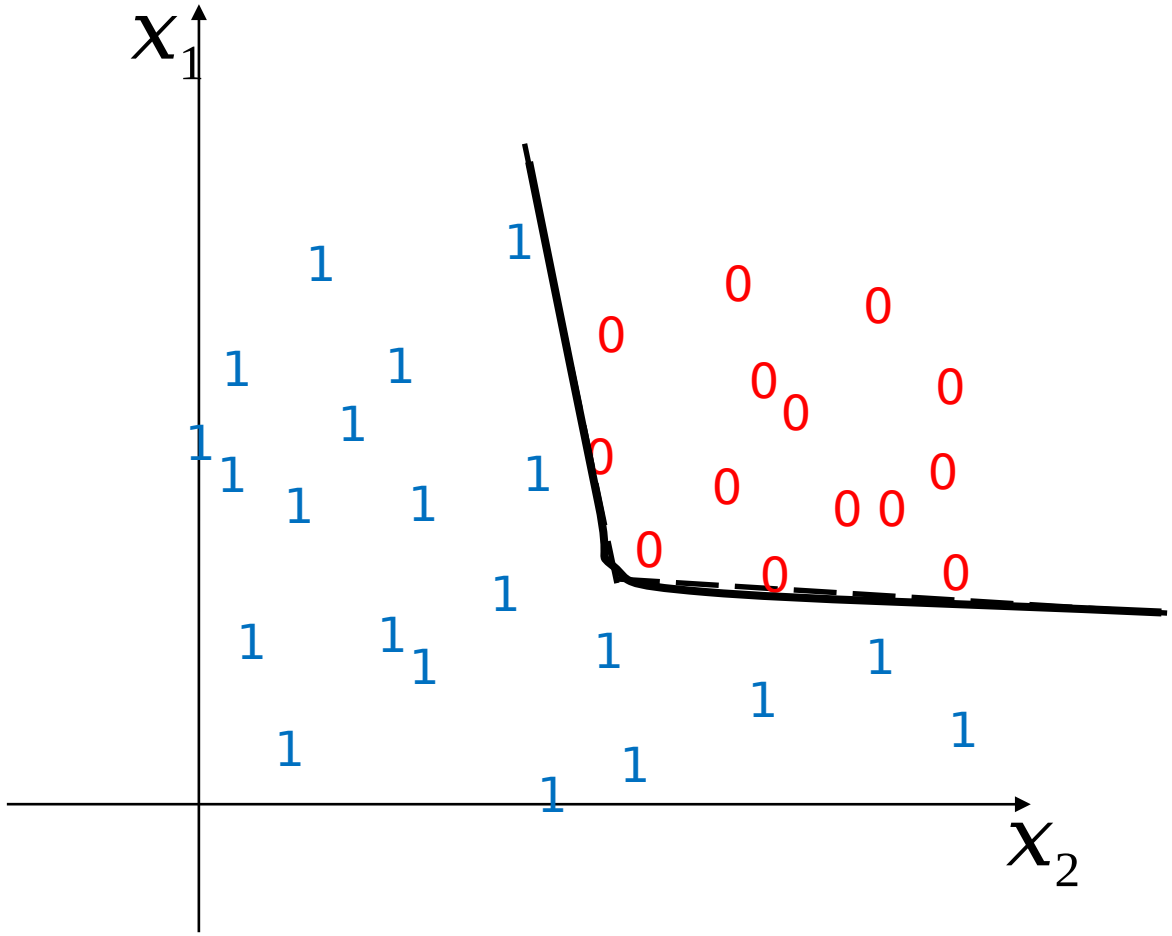
Simple Network (Example 2 - – Second Neuron)



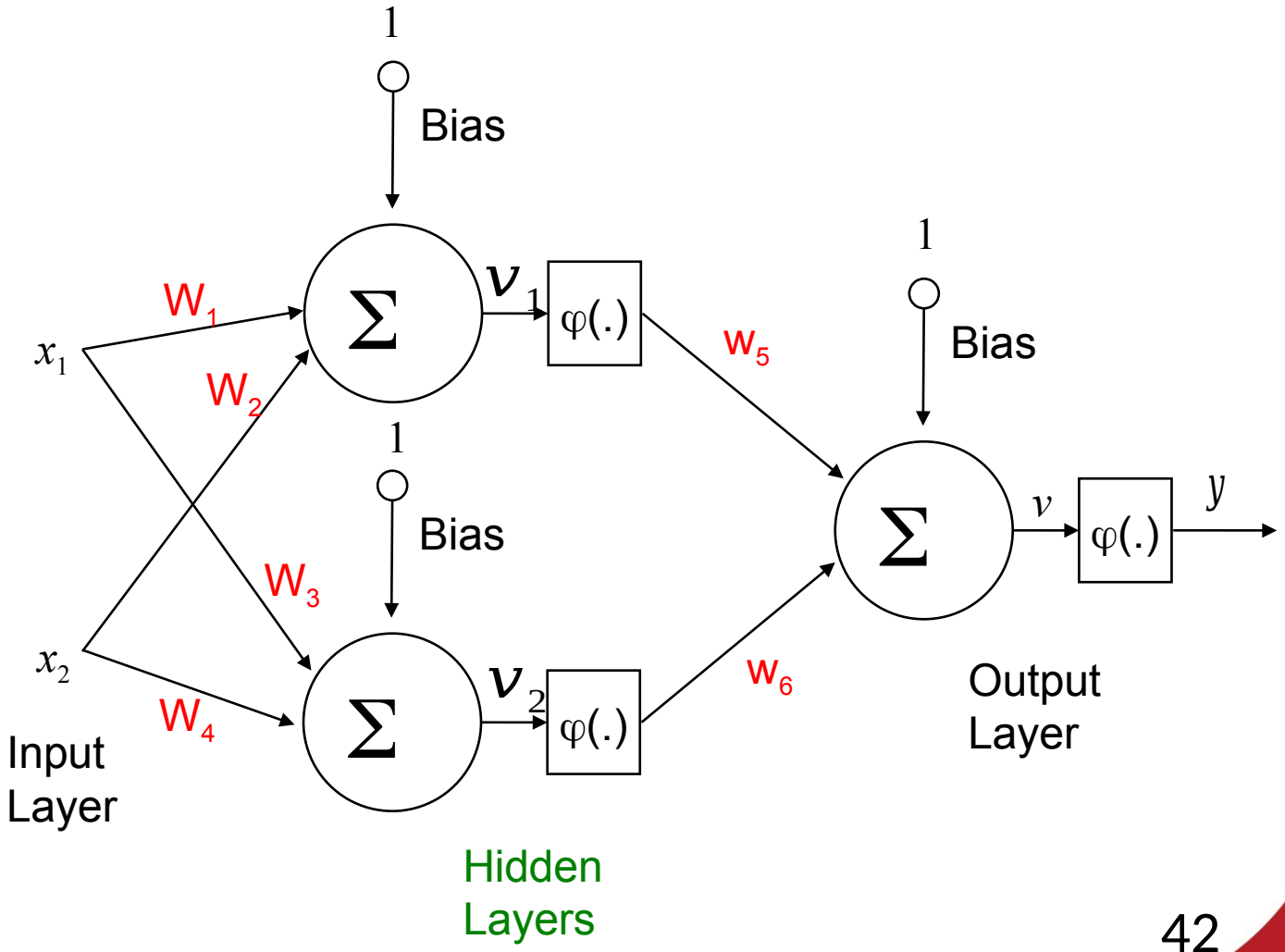
Simple Network (Example 2 – Combine)



Simple Network (Example 2 – Combine)

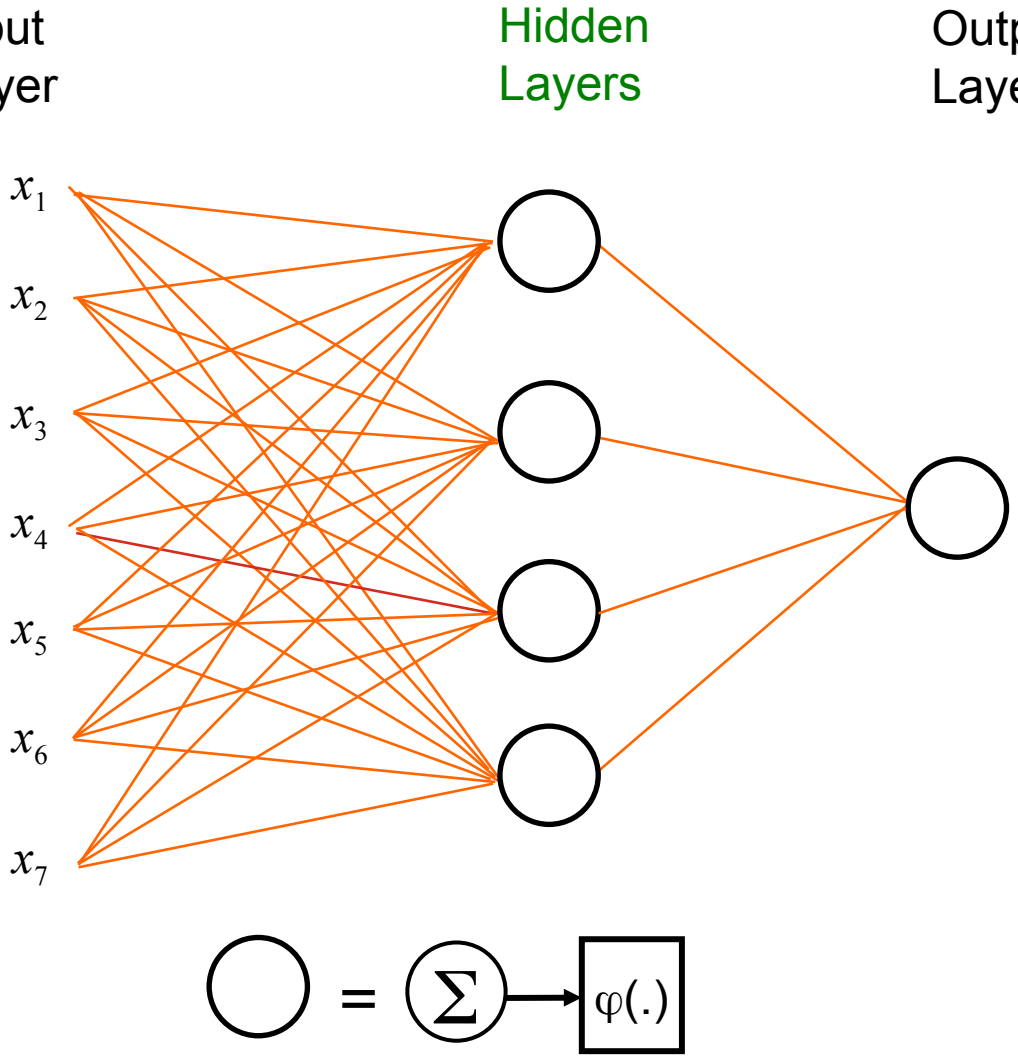


Simple Network (Example 2 – Model)

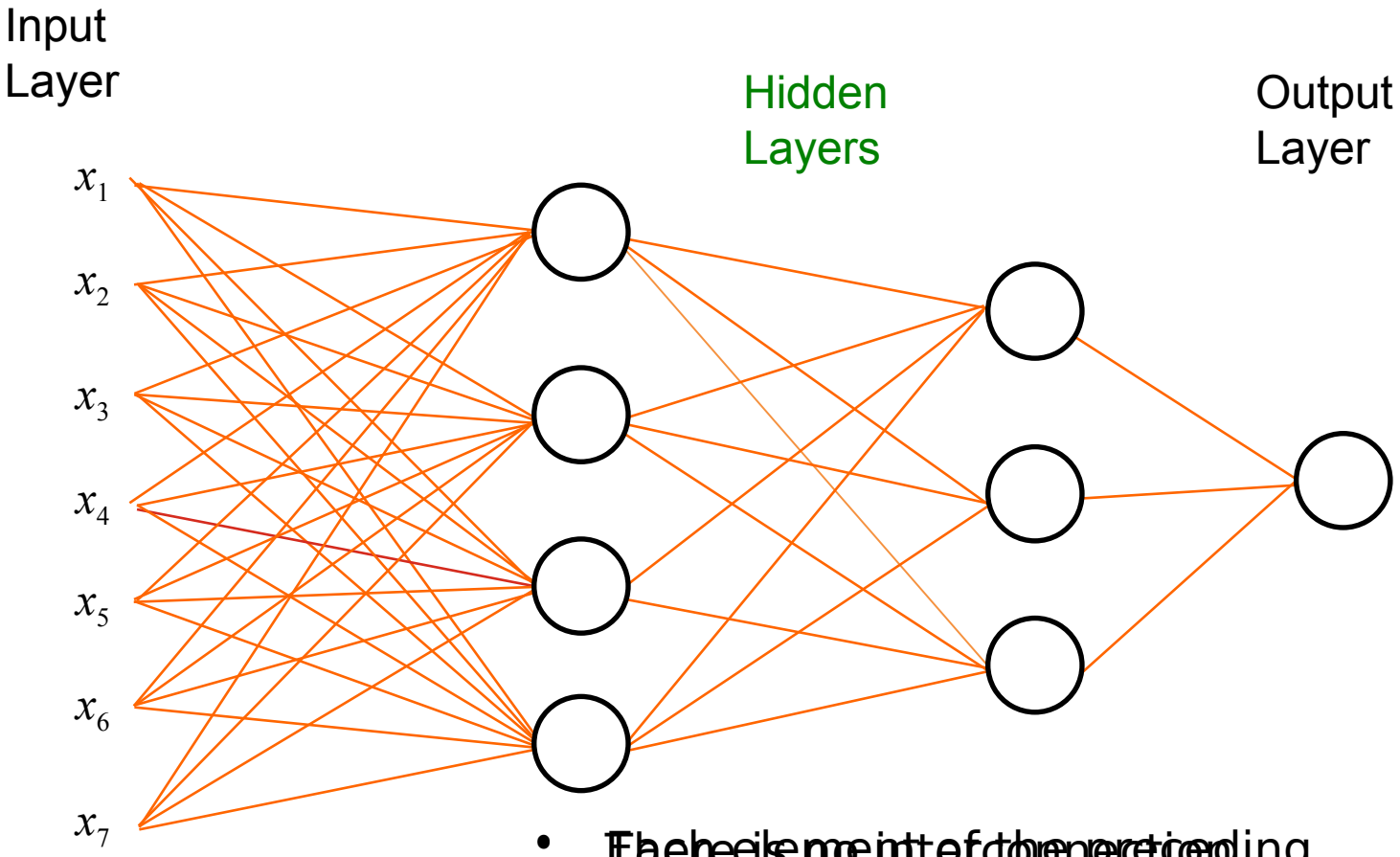


Multi-Layer Perceptron (One hidden layer)

Lancaster University



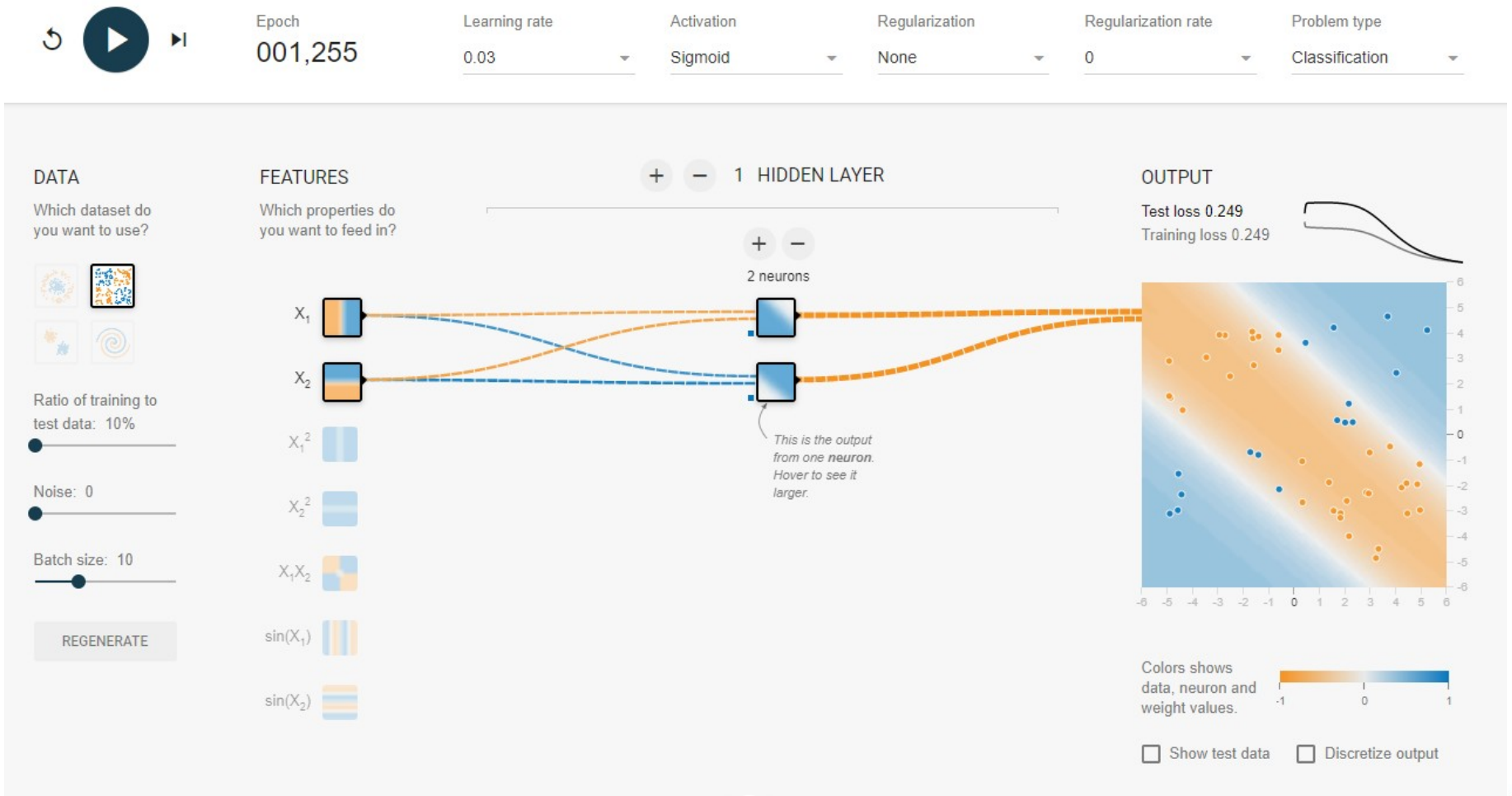
Multi-Layer Perceptron (Two hidden layers)



- Each element of the preceding layer is connected to every element of the next layer (fully connected)

Multi-Layer Perceptron (Visualizing Decision Boundary)

Lancaster University

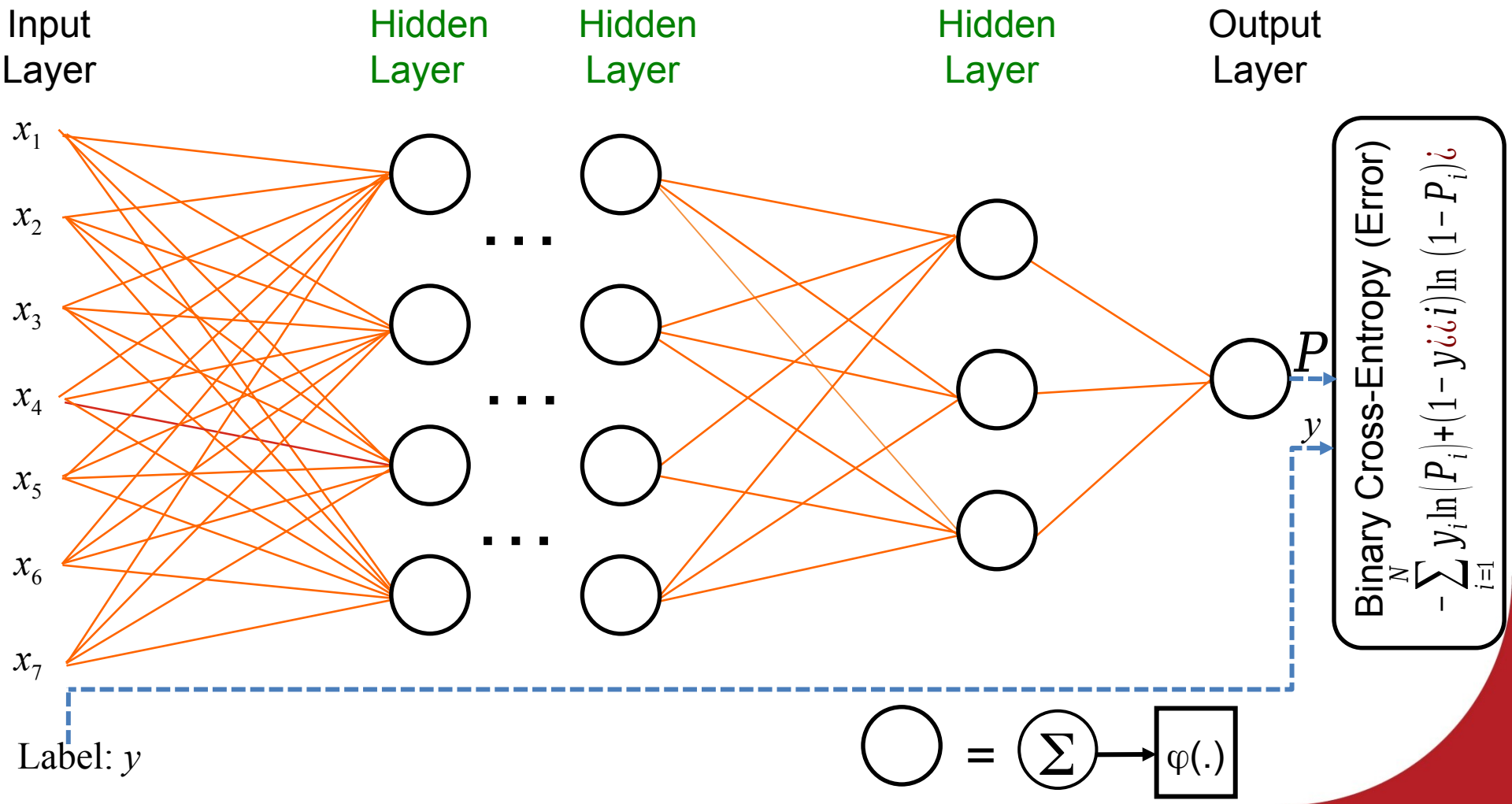


Why the MLP?

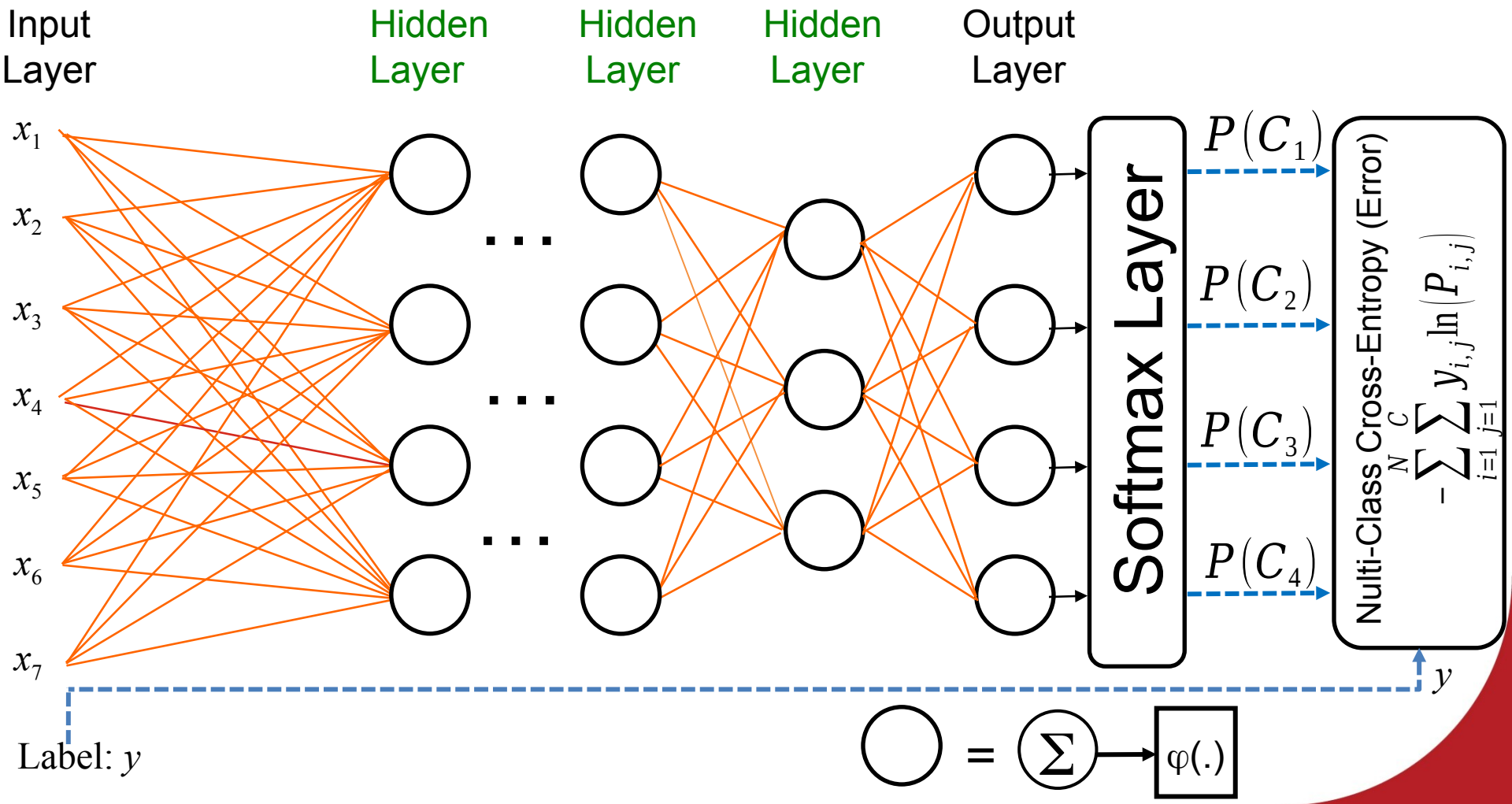
Lancaster University

- The single-layer perceptron classifiers discussed previously can only deal with linearly separable sets of patterns.
- The multilayer networks are the most widespread neural network architecture
  - Made useful until the 1980s, because of lack of efficient training algorithms
  - The introduction of the backpropagation training algorithm.

General Architecture (Binary Classifier)



General Architecture (Multi-Class Classifier)



Training MLP

Lancaster University

- This process tunes the parameters such that the input space is correctly mapped to the output space.
- The current estimate of the output variables is matched with the desired output.
- This matching function (e.g. **Cross Entropy**) serves as an objective function during the MLP training and it is usually called the **loss function** or the **error function**.
- CNN training process involves the optimization of the model parameters such that the loss function is minimized.
- The optimization of such non-linear models is a **hard task**.
- Search for **the locally optimal** solution at each step
- The **gradient based methods** come as a natural choice

Training MLP (Backpropagation Algorithm)

Lancaster University

- The **backpropagation** algorithm looks for the minimum of the error function in weight space using the method of **gradient descent**.
- The combination of weights which minimizes the error function (**e.g. cross-entropy**) is considered to be a solution of the learning problem.
- Since this method requires computation of the gradient of the error function at each iteration step, we must guarantee the continuity and differentiability of the error function.

Backpropagation

Lancaster University

- The back propagation algorithm is a generalization of the perceptron learning algorithm
- This algorithm updates the weights of the network by means of successive iterations, that minimize the cost function of the error:

$$E = -\sum_{i=1}^{N} y_i \ln\left(P_i\right) + \left(1 - y¿¿i\right) \ln\left(1 - P_i\right)¿$$

- The minimization of the error is obtained using the gradient of the cost function: $\frac{\partial}{\partial w_{i,j}}(E)$
- On the basis of this gradient the weights will be updated with the following mechanism:

$$w_{i,j} = w_{i,j} - \alpha \frac{\partial}{\partial w_{i,j}}(E)$$

Variants of Gradient Descent

Lancaster University

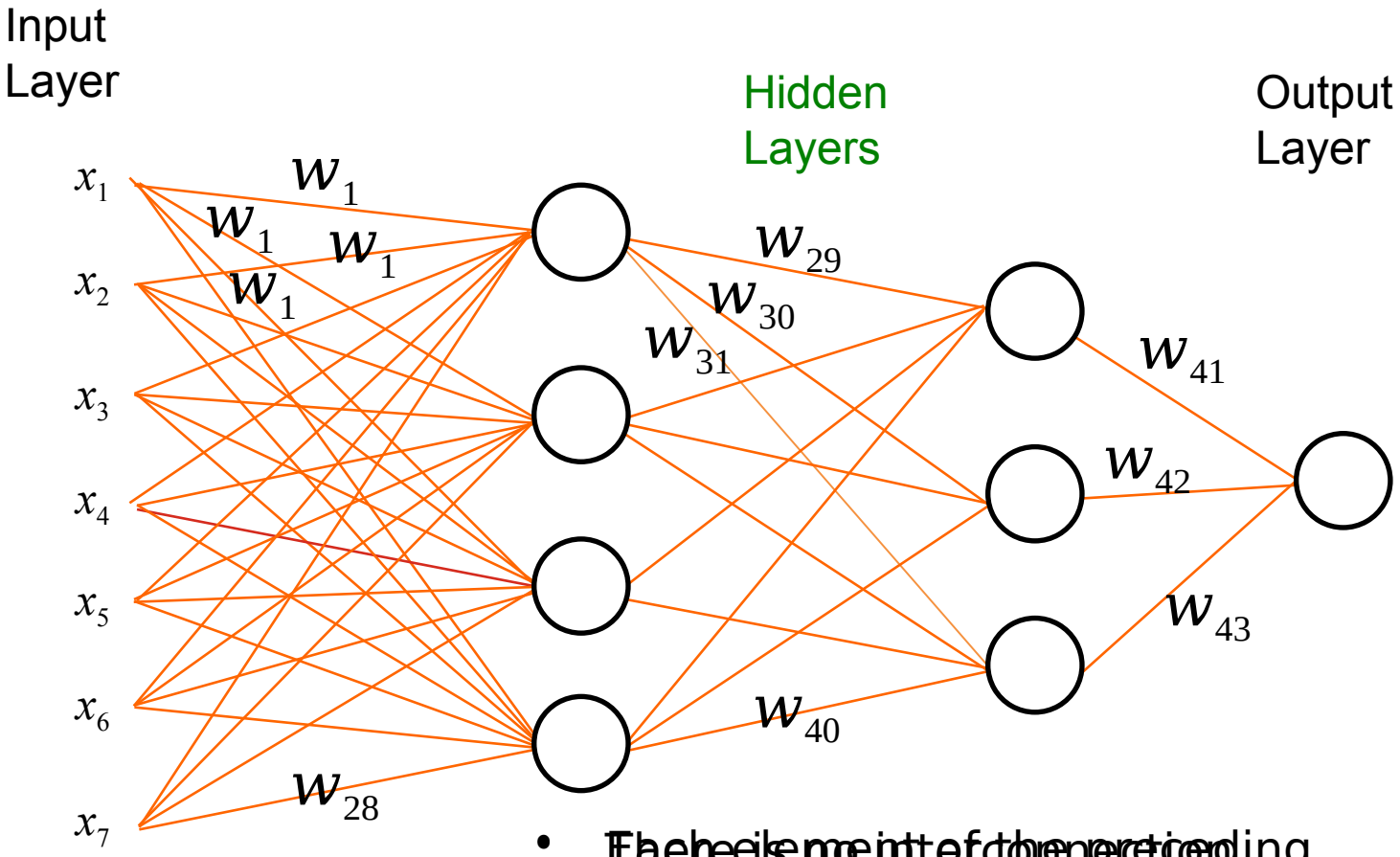
- **STOCHASTIC GRADIENT DESCENT**
  - Performs parameter update for each set of input and output present in the training set
  - converges much faster compared to the batch gradient descent
  - Convergence behaviour is usually instable
- **BATCH GRADIENT DESCENT**
  - Computes gradient on the entire training set
- **MINI-BATCH GRADIENT DESCENT**
  - dividing the training set into a number of mini-batches
  - an improved form of stochastic gradient descent
  - provides a decent trade-off between convergence efficiency and convergence stability

Outline

Lancaster University

- **Multi Layer Perceptron (MLP)**
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - Convolution Layer
  - Activation Function
  - Pooling Layer
  - Transposed Convolution (Deconv)
  - Un-pooling layer

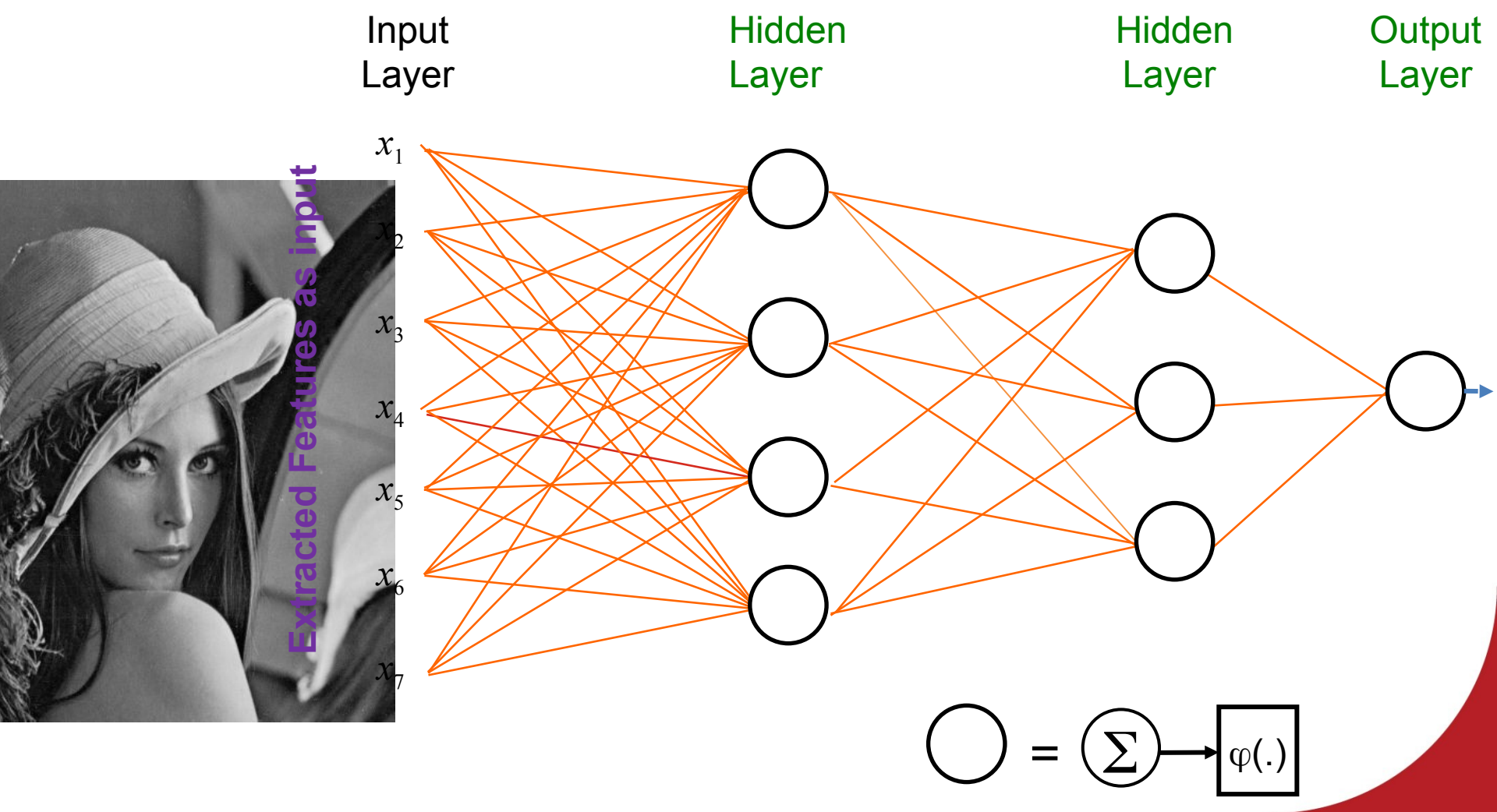
Multi-Layer Perceptron (Two hidden layers)



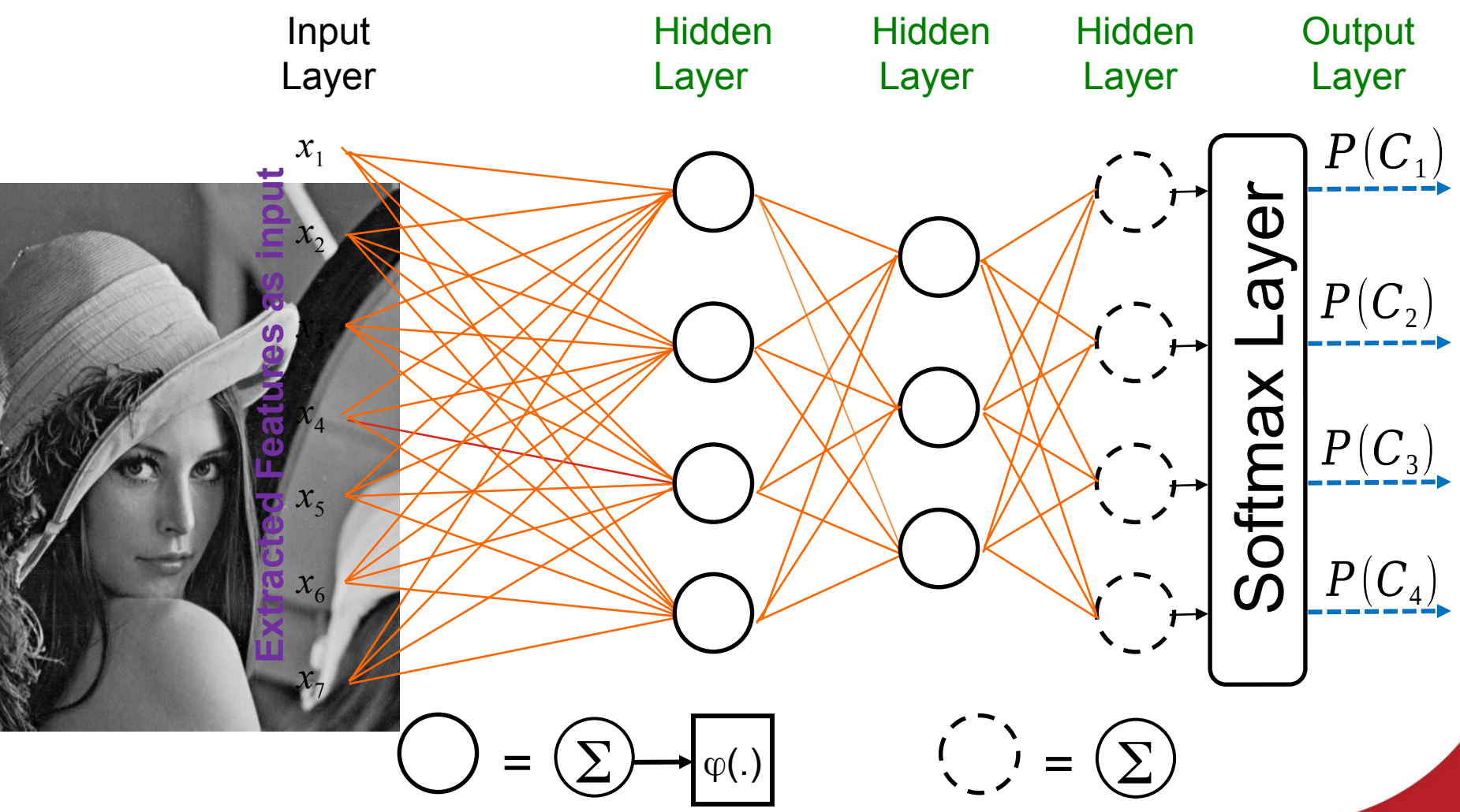
- Each element of the preceding layer is connected with each element of the next layer (fully connected)

Image Classification using Features and MLP

Lancaster University



Multiple Outputs (more than 2 Class - Classification)



Feature Representations

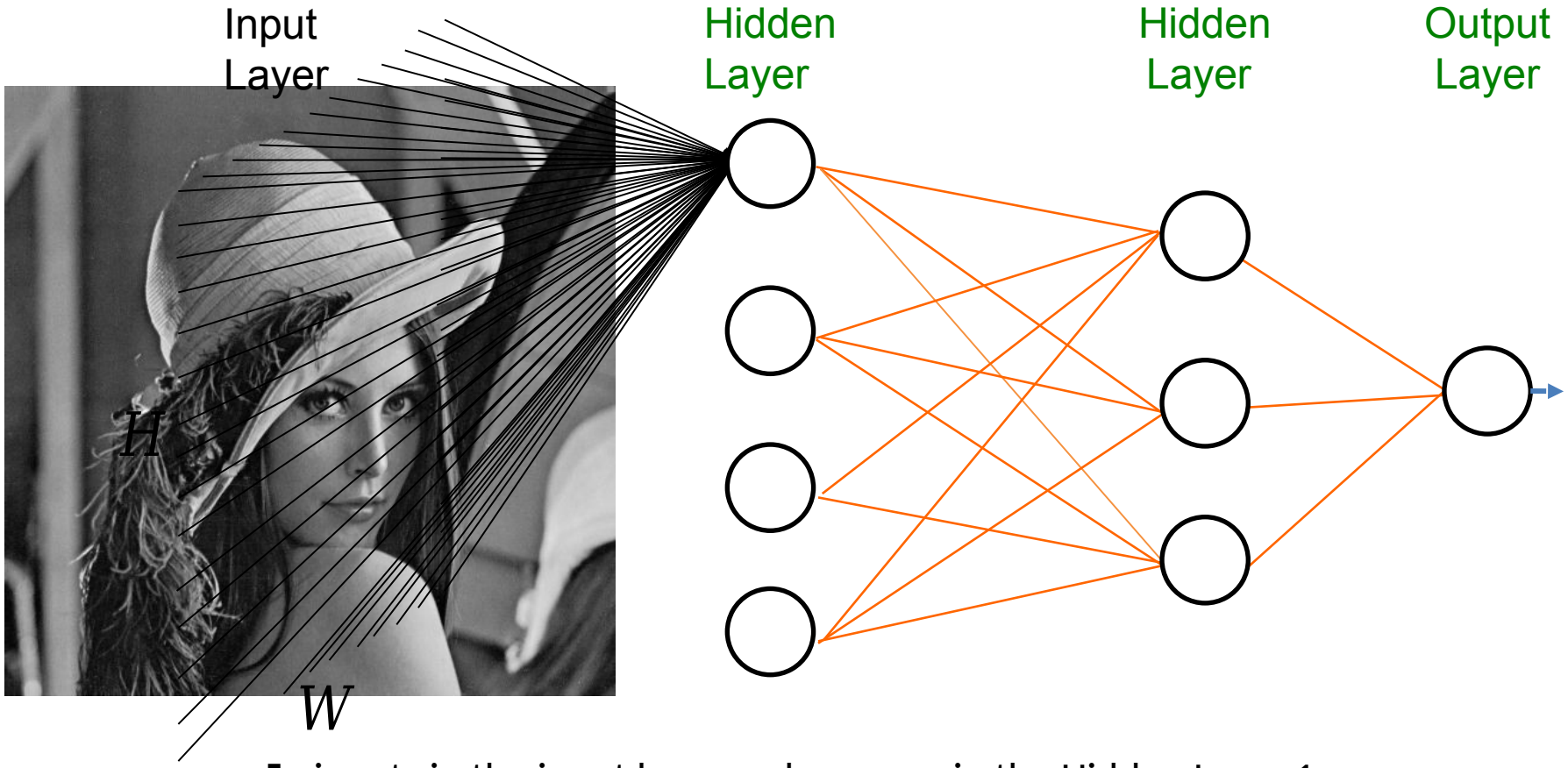
Input

Pre-processing → Feature extraction → Feature selection → Class Prediction

Feature learner

Feature learning: Instead of design features, let's design feature learners

Image Classification using MLP



- inputs in the input layer and neurons in the Hidden Layer 1

Example: image

hidden units: weight parameters !!!

Lancaster University

Questions?

Lancaster University

AI SCC361 Week 10

Dr. Hossein Rahmani
Senior Lecturer in Data Science
D17, InfoLab; e-mail: h.rahmani@lancaster.ac.uk

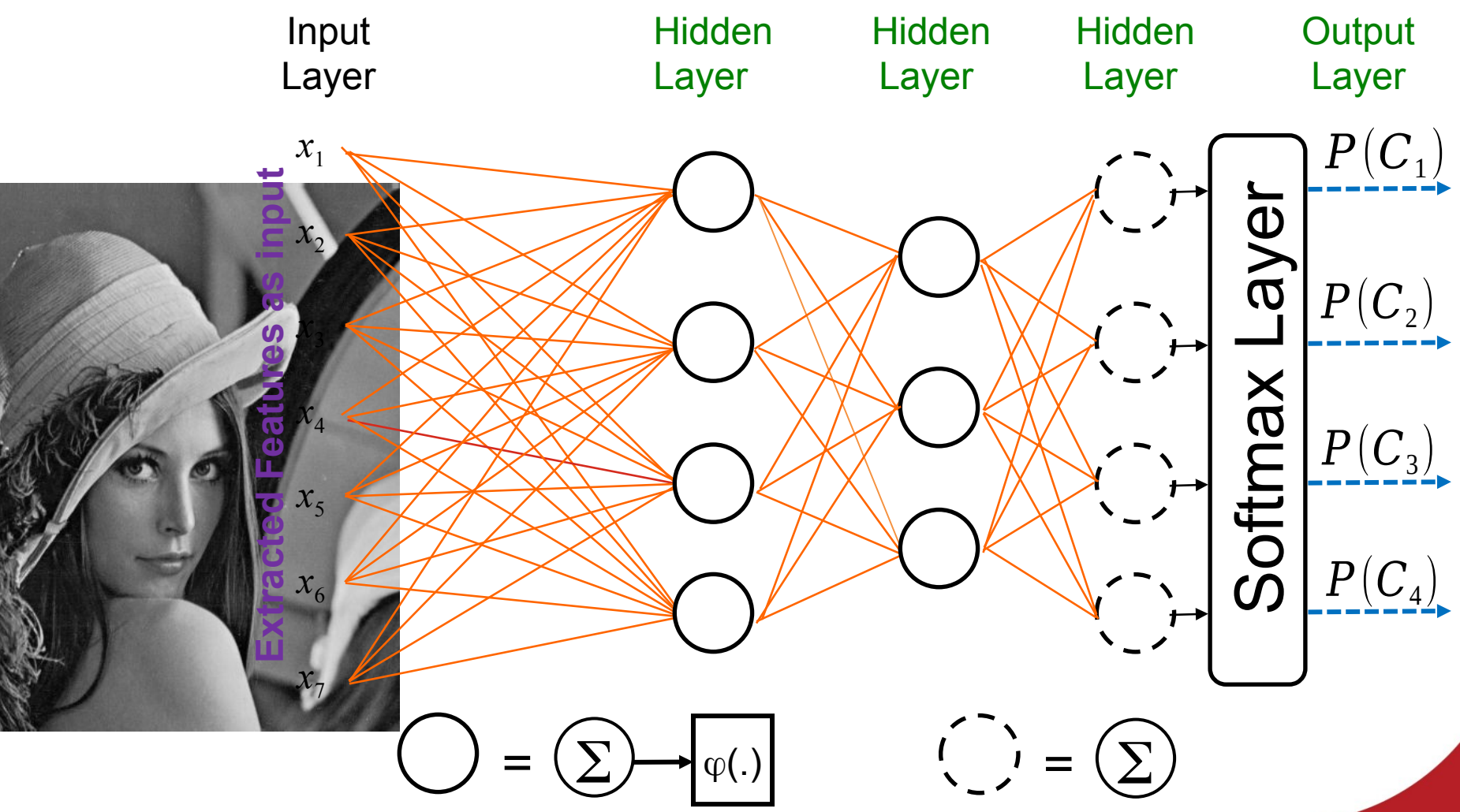
Convolutional Neural Networks

Outline

Lancaster University

- **Multi Layer Perceptron (MLP)**
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - Convolution Layer
  - Activation Function
  - Pooling Layer
  - Transposed Convolution (Deconv)
  - Un-pooling layer

Multiple Outputs (more than 2 Class - Classification)



Feature Representations

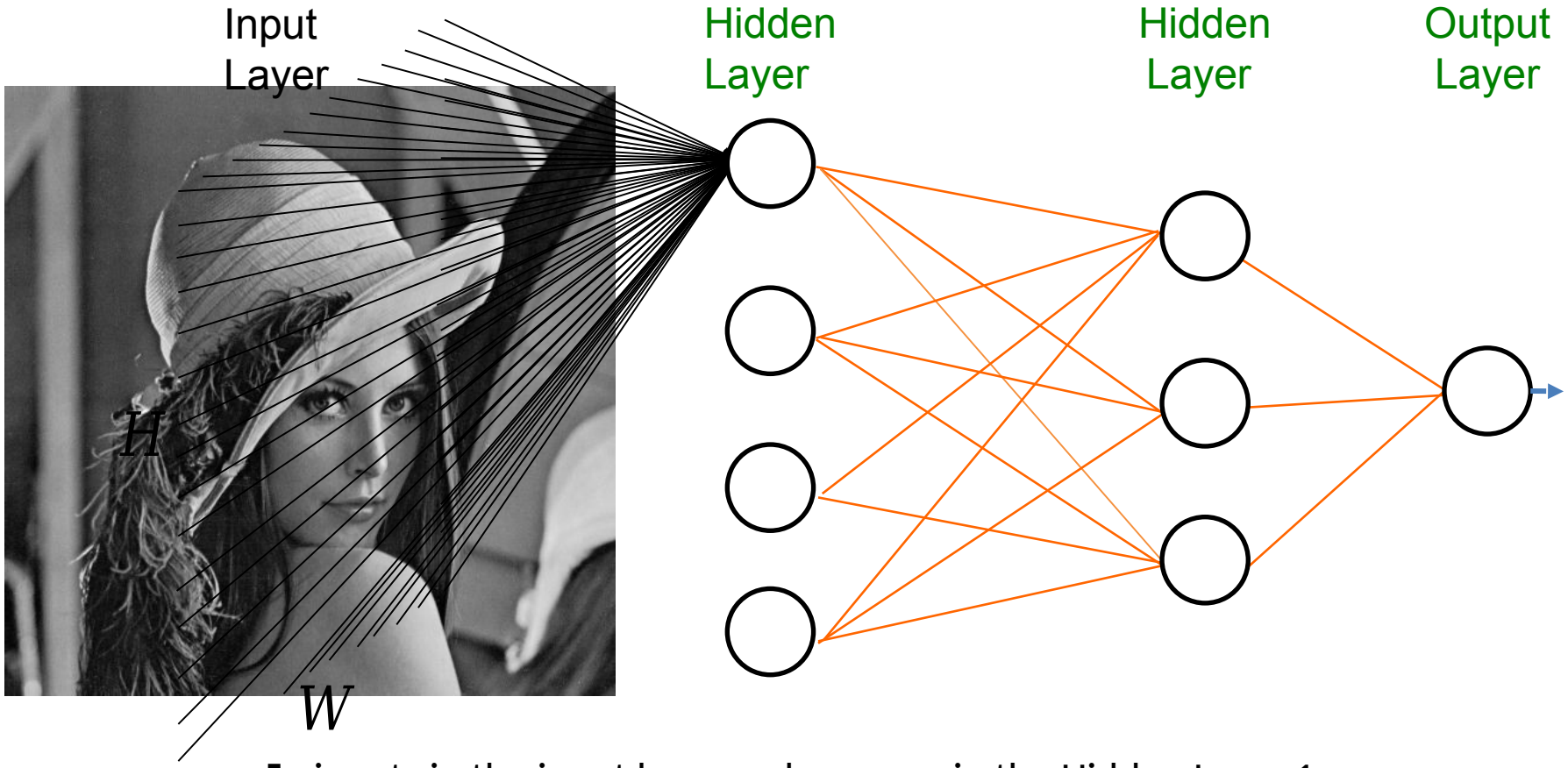
Input

Pre-processing → Feature extraction → Feature selection → Class Prediction

Feature learner

Feature learning: Instead of design features, let's design feature learners

Image Classification using MLP

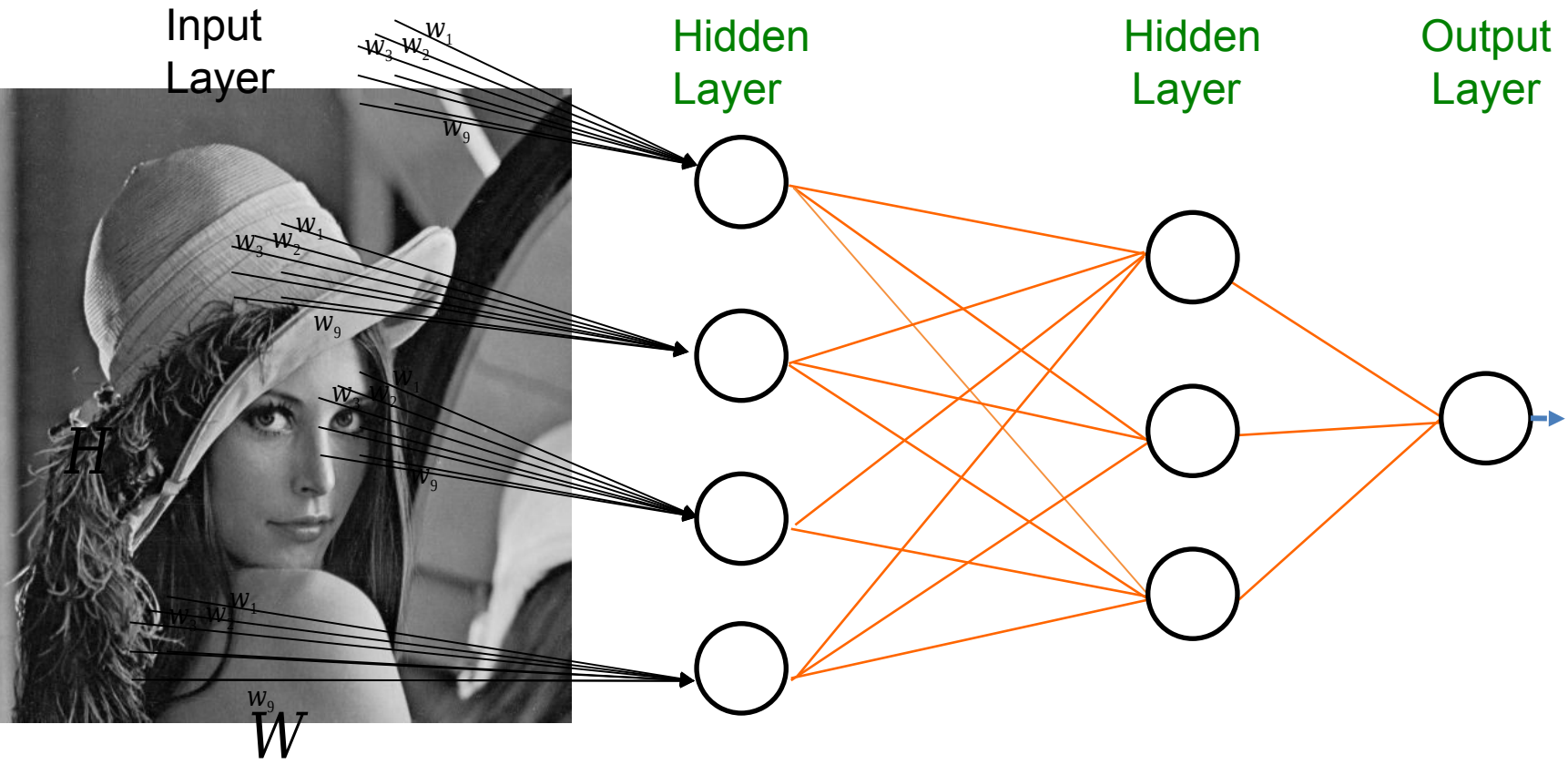


- inputs in the input layer and neurons in the Hidden Layer 1

Example: image

hidden units: weight parameters !!!

Sparse Interactions



Example: image

Filter size: (): parameters!!!

Outline

Lancaster University

- Multi Layer Perceptron (MLP)
  - Why it is not good for images
- **Convolutional Neural Networks (CNN)**
  - Convolution Layer
  - Activation Function
  - Pooling Layer
  - Transposed Convolution (Deconv)
  - Un-pooling layer

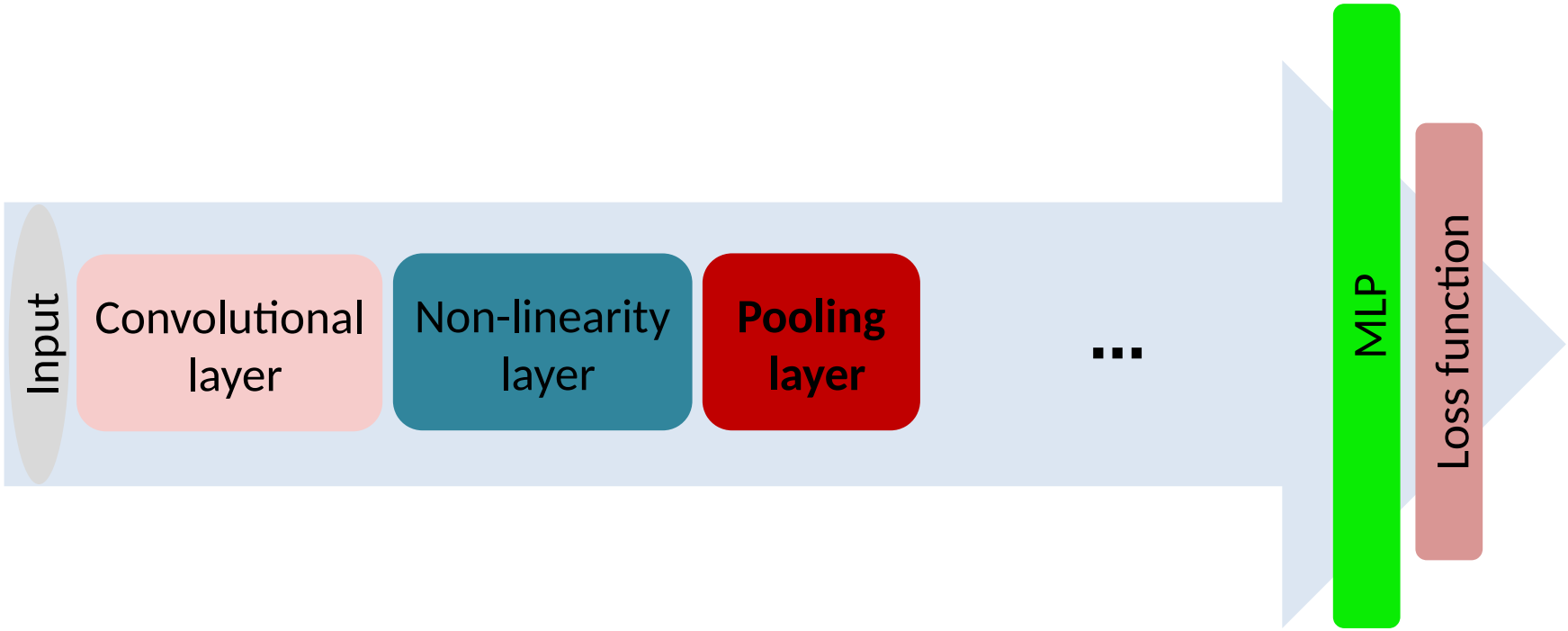
Convolutional Neural Networks

Lancaster University

- Convolutional neural networks are also known as Convolutional networks, CNNs, or ConvNets
- Specialized for data having grid-like topology
  - 1D grid – time series data
  - 2D grid –Most successful on 2D image topology

Typical CNN Layers

Lancaster University



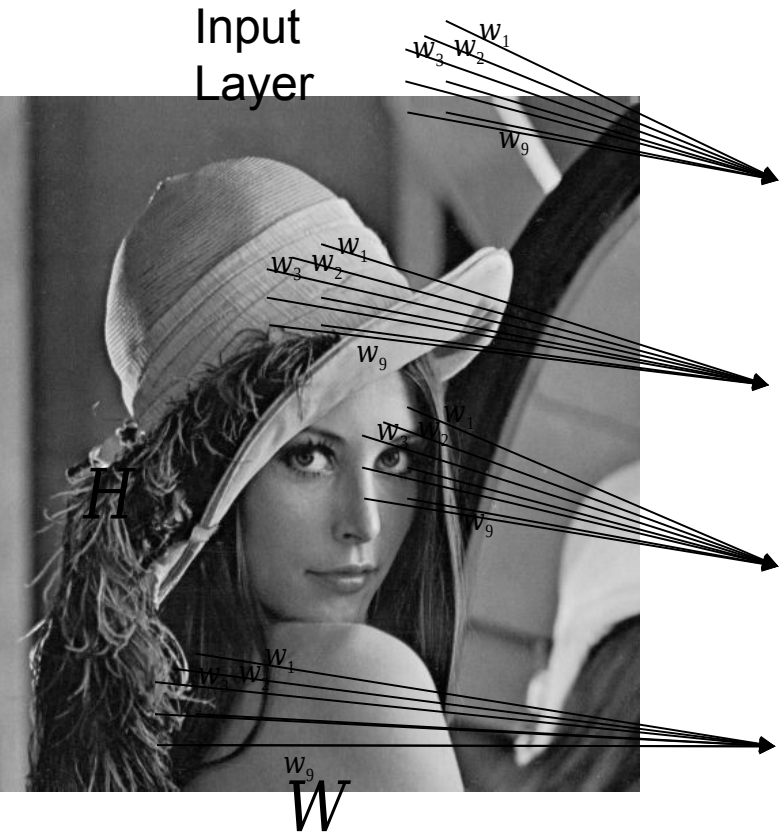
Outline

Lancaster University

- Multi Layer Perceptron (MLP)
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - **Convolution Layer**
  - Activation Function
  - Pooling Layer
  - Transposed Convolution (Deconv)
  - Un-pooling layer

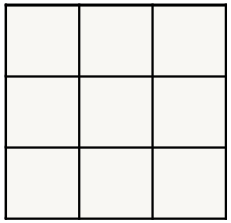
Convolution Layer (Gray images)

- Given a gray-scale input image and a filter/kernel ,

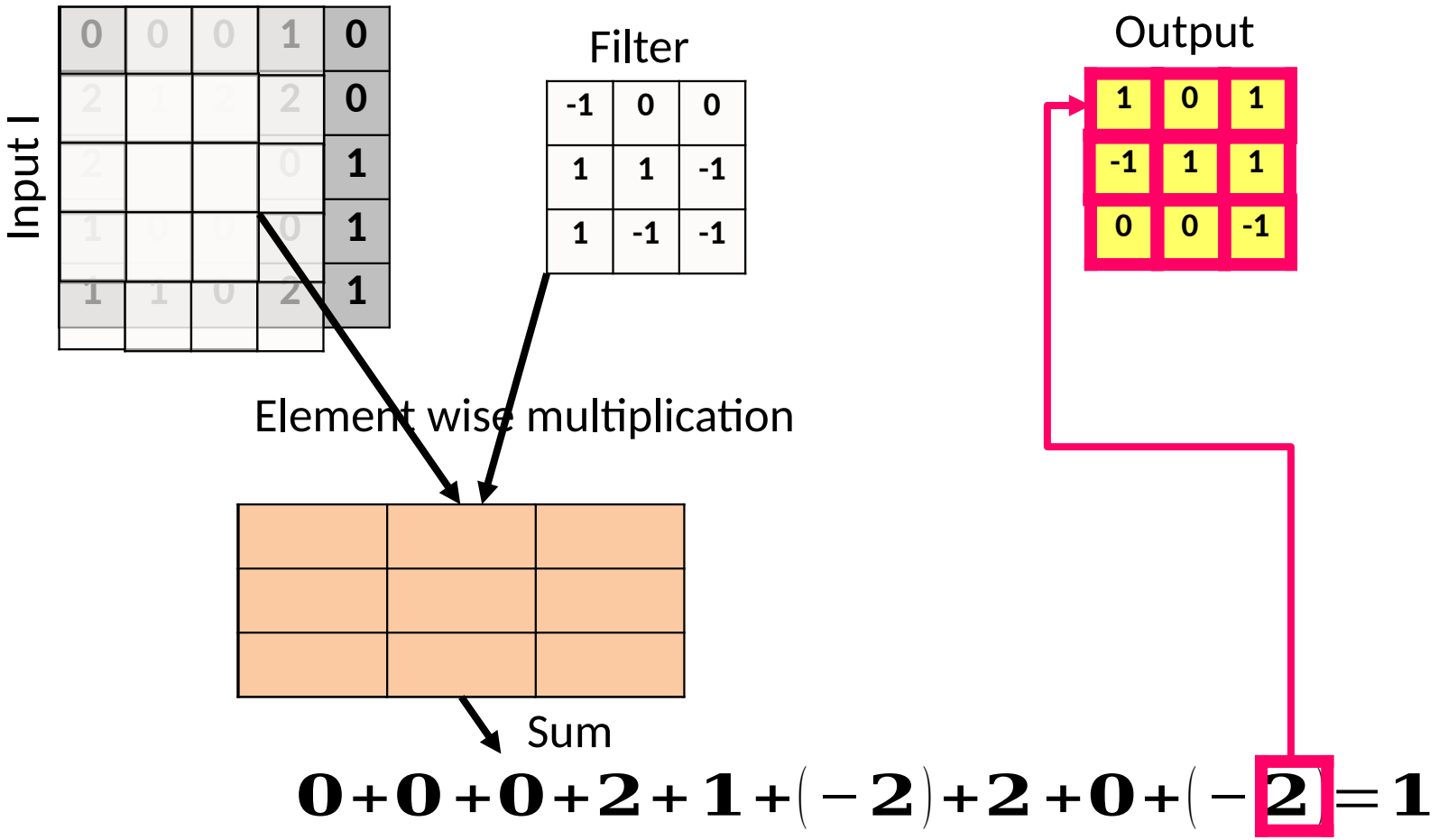


Input image I

Filter



Convolution Layer (Gray images)



Convolution Layer (RGB images)

- Given an RGB input image and a filter/kernel ,

Input image

RGB

Filter

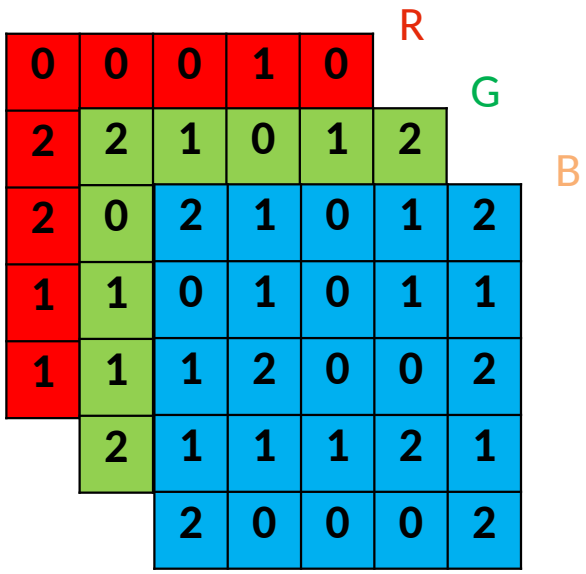
-1	0	0		
1	1	0	1	
1	-1	-1	-1	-1
	0	1	0	-1
		-1	-1	1

Convolution Layer (RGB images)

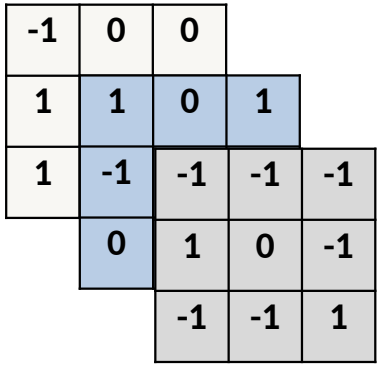
Lancaster University

- Given an RGB input image and a filter/kernel ,

Input image

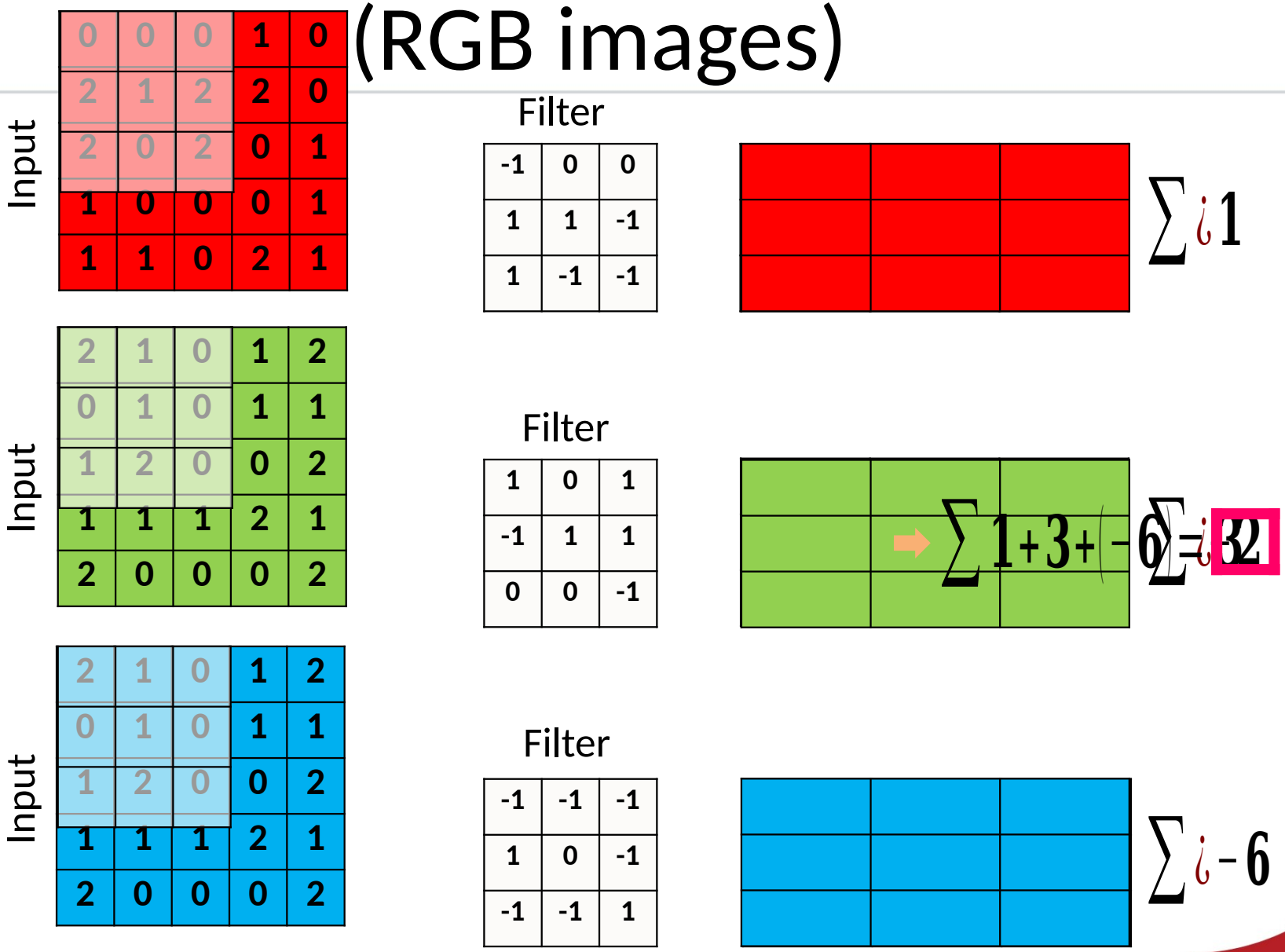


Filter



Convolution Layer (RGB images)

Lancaster University



Convolution Layer (RGB images)

Lancaster University

Input

0	0	0	1	0
2			2	0
			0	1
1			0	1
1	1	0	2	1

Filter

-1	0	0
1	1	-1
1	-1	-1

Input

2	1	0	1	2
0			1	1
1			0	2
1			2	1
2	0	0	0	2

Filter

1	0	1
-1	1	1
0	0	-1

Input

2	1	0	1	2
0			1	1
1			0	2
1			2	1
2	0	0	0	2

Filter

-1	-1	-1
1	0	-1
-1	-1	1

Output $o_0(:,:,1)$

-2	0	1
-1	1	1
0	0	-1

Variants of the Basic Convolution Function

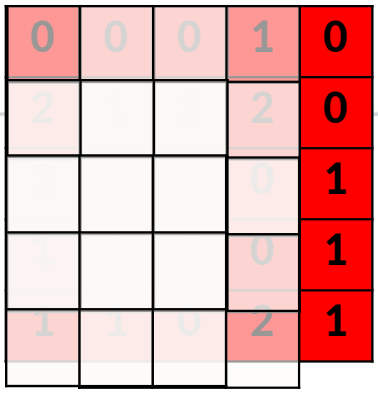
Lancaster University

- Stride
  - is the amount of down-sampling
- Zero padding
  - avoids layer-to-layer shrinking
- Unshared convolution
  - Like convolution but without sharing
- Tiled convolution
  - Cycle between shared parameter groups

Convolution with Stride 1

Lancaster University

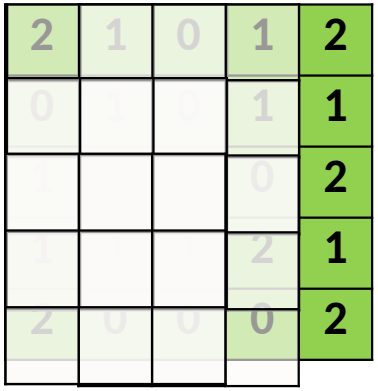
Input



Filter

-1	0	0
1	1	-1
1	-1	-1

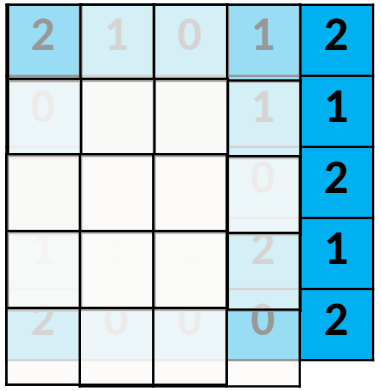
Input



Filter

1	0	1
-1	1	1
0	0	-1

Input



Filter

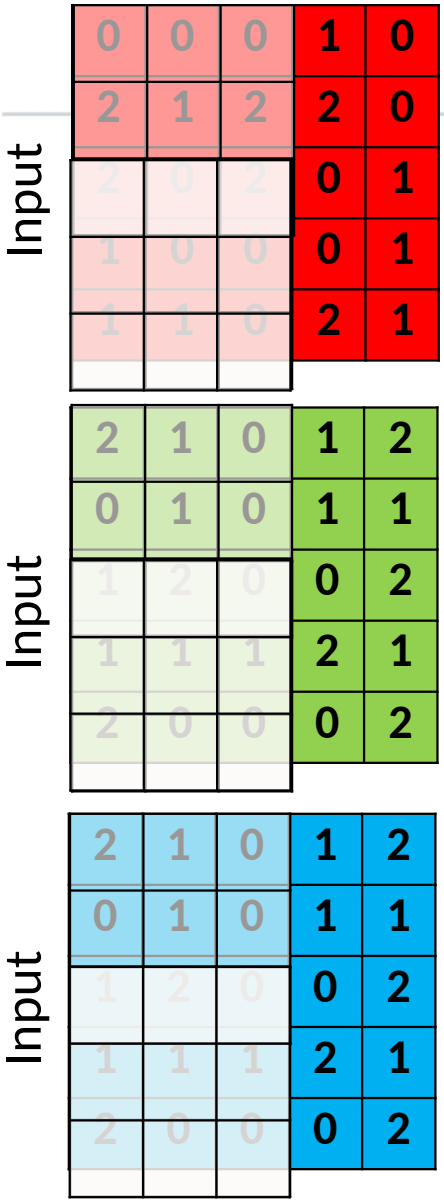
-1	-1	-1
1	0	-1
-1	-1	1

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Convolution with Stride 2

Lancaster University



Filter

-1	0	0
1	1	-1
1	-1	-1

Filter

1	0	1
-1	1	1
0	0	-1

Filter

-1	-1	-1
1	0	-1
-1	-1	1

Output $o_0(:,:,1)$

1	0
-1	1

shrinking

Variants of the Basic Convolution Function

Lancaster University

- Stride
  - is the amount of down-sampling
- Zero padding
  - avoids layer-to-layer shrinking
- Unshared convolution
  - Like convolution but without sharing
- Tiled convolution
  - Cycle between shared parameter groups

Convolution with Zero Padding

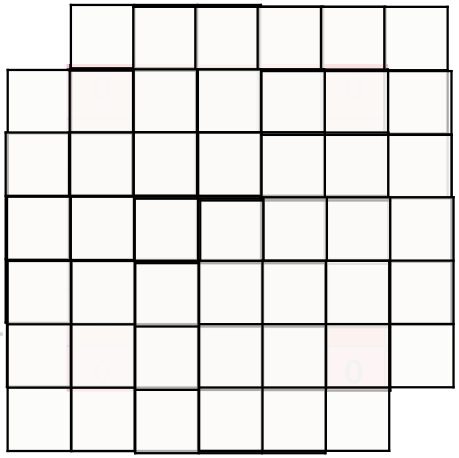
Lancaster University

- Convolution without Zero Padding results in shrinking

- **Why use padding?**
  - Improves performance
  - Without Zero Padding the inputs,
    - the size of the volumes would reduce by a small amount after each CONV, and
    - the information at the borders would be “washed away” too quickly.

Lancaster University

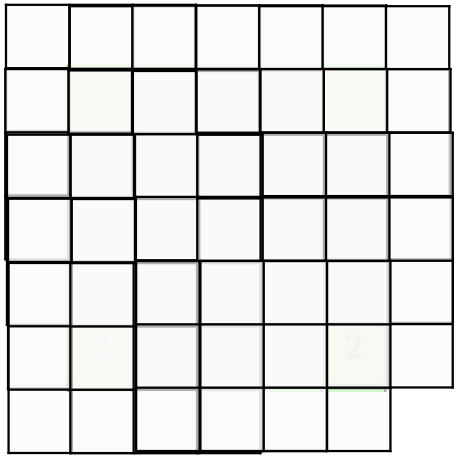
Input



Filter

-1	0	0
1	1	-1
1	-1	-1

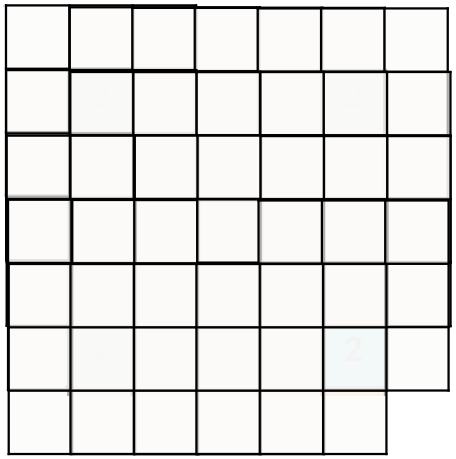
Input



Filter

1	0	1
-1	1	1
0	0	-1

Input I



Filter

-1	-1	-1
1	0	-1
-1	-1	1

Output

2	1	0	1	2
0	1	0	1	1
1	2	0	0	2
1	1	1	2	1
2	0	0	0	2

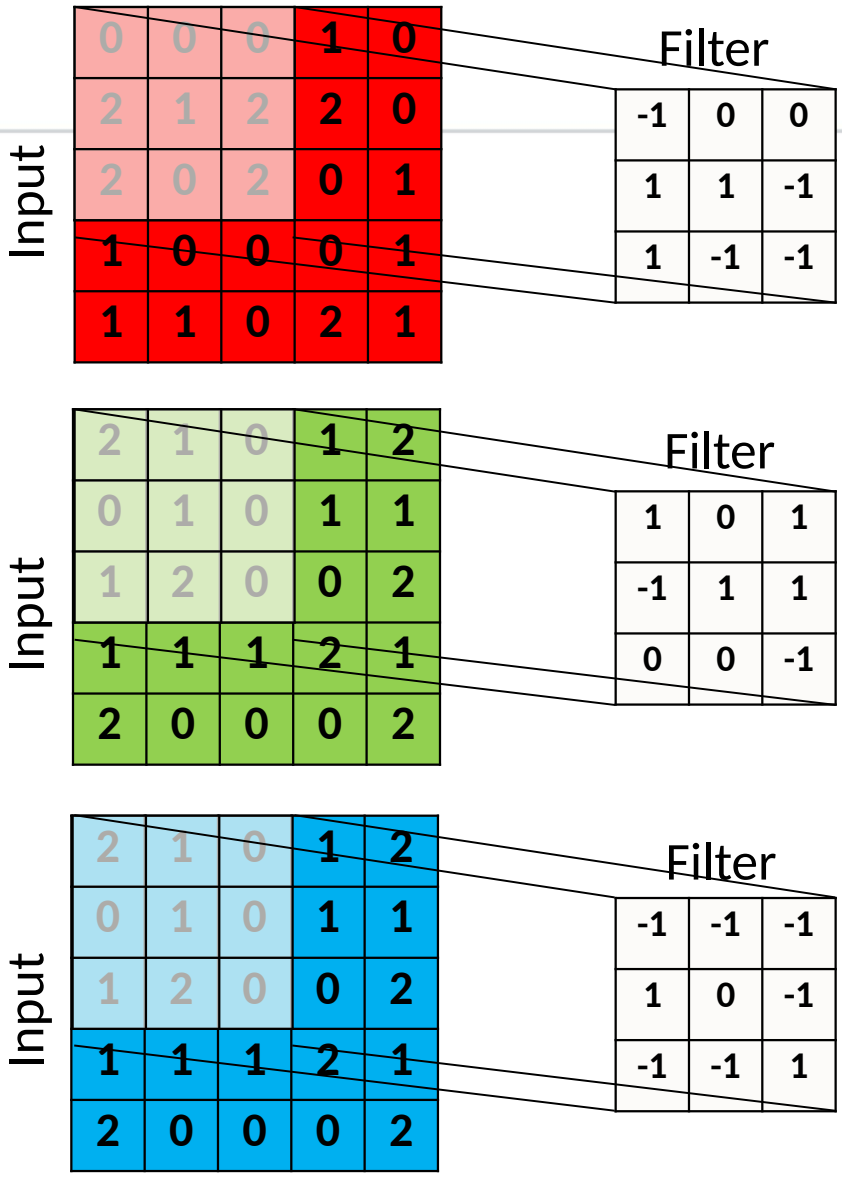
Convolution with Zero Padding
(output is)

Variants of the Basic Convolution Function

Lancaster University

- Stride
  - is the amount of down-sampling
- Zero padding
  - avoids layer-to-layer shrinking
- Unshared convolution
  - Like convolution but without sharing
- Tiled convolution
  - Cycle between shared parameter groups

Unshared Convolution

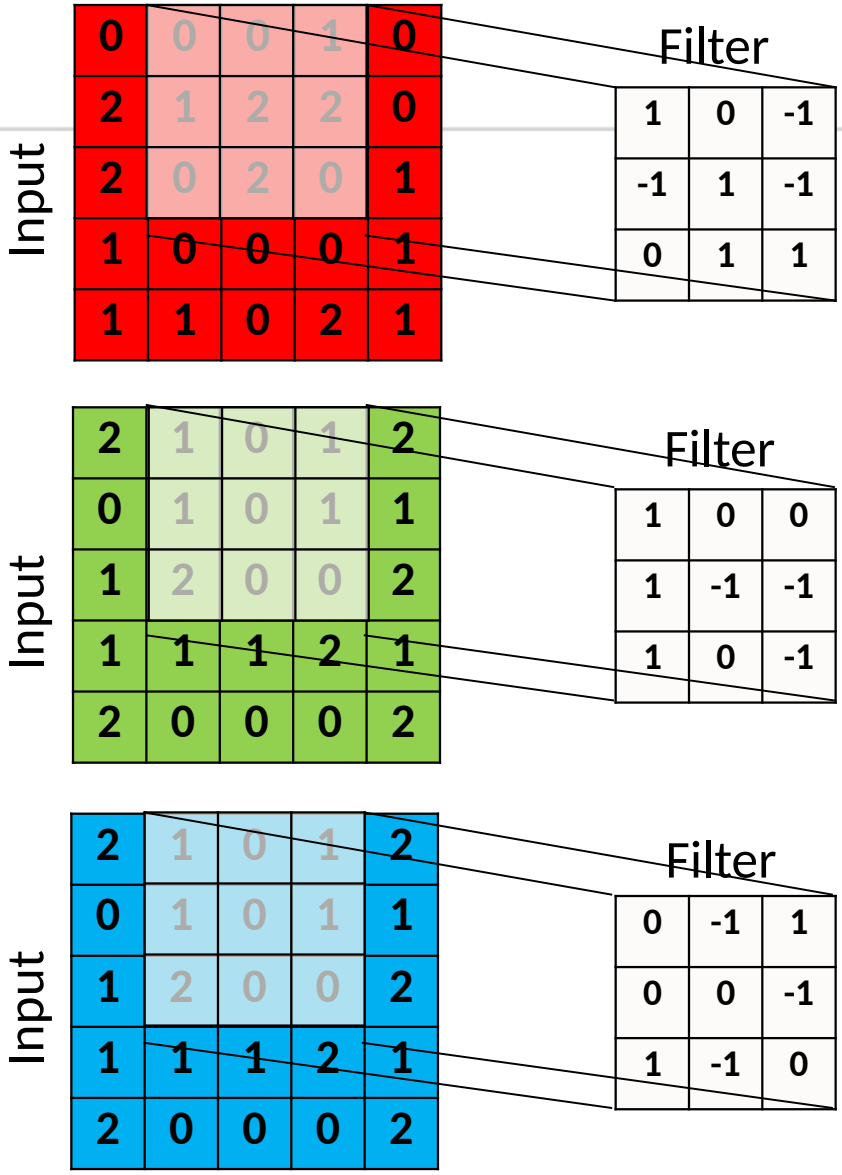


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

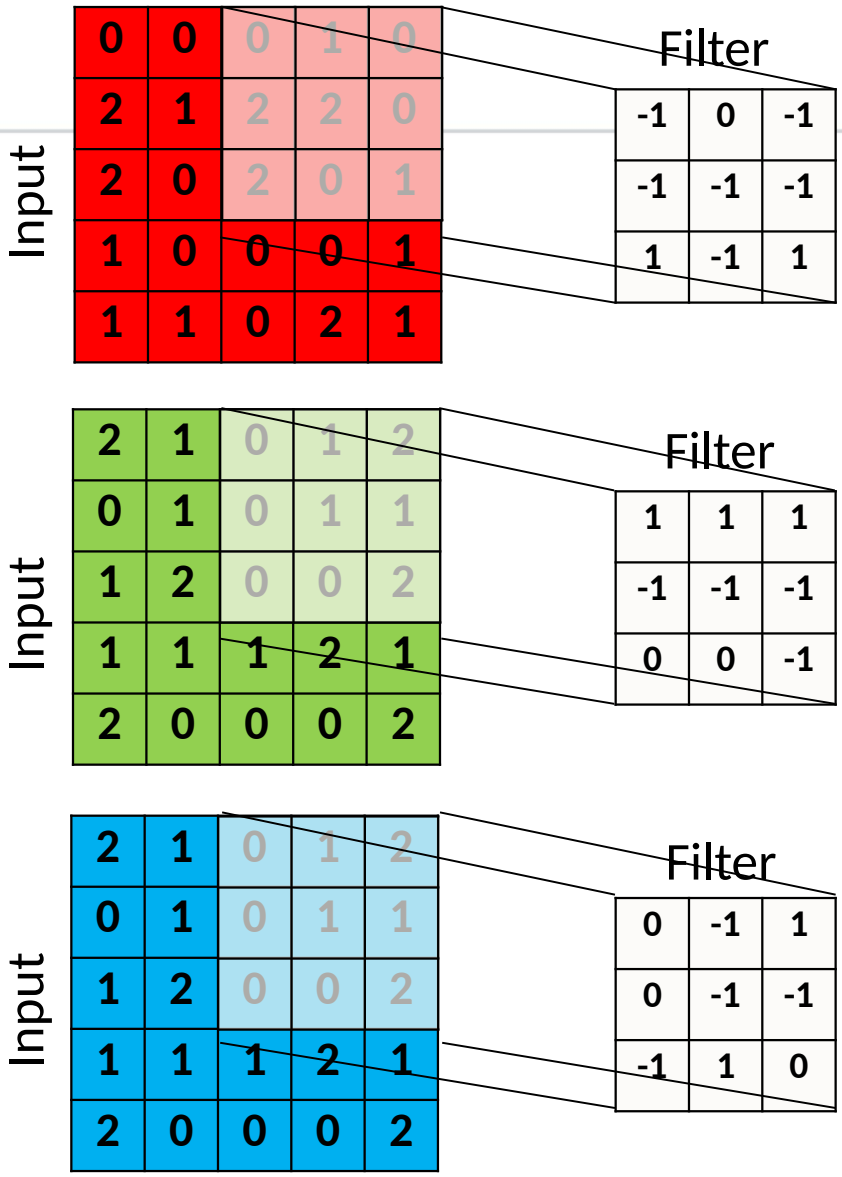


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

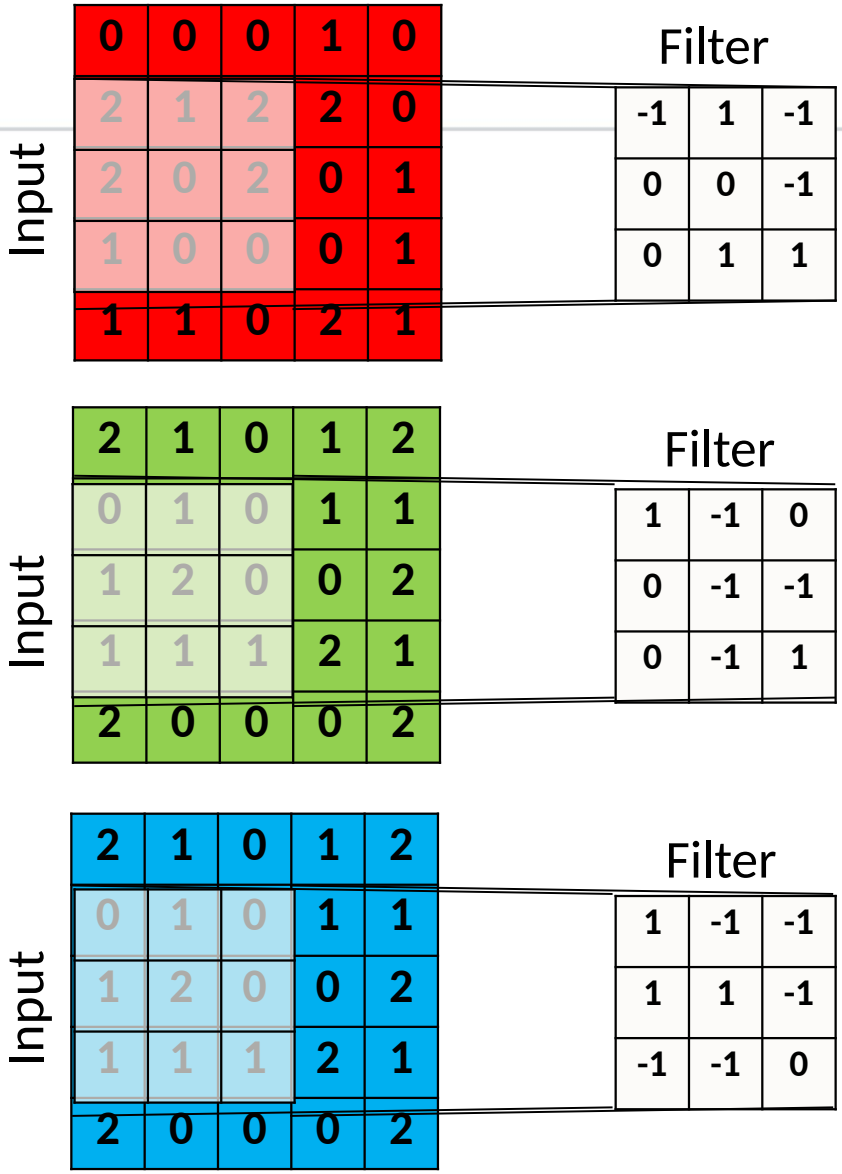


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

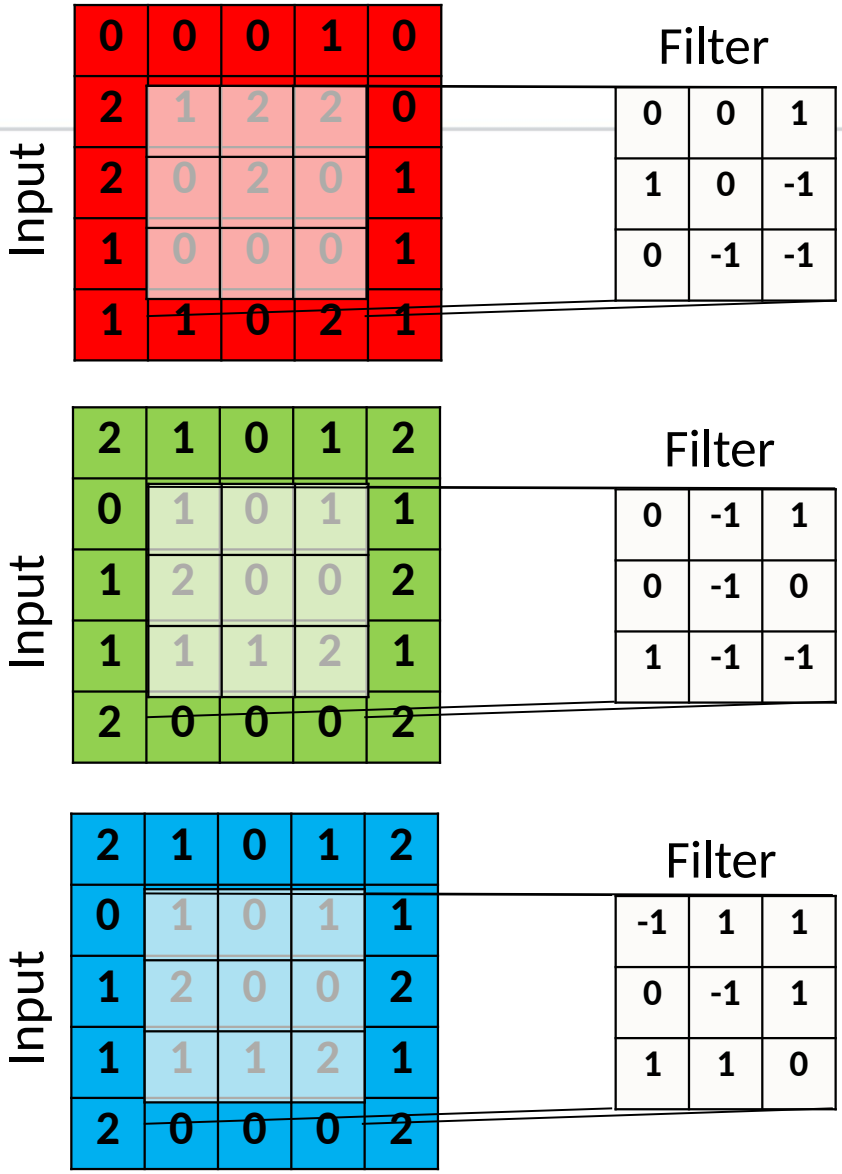


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

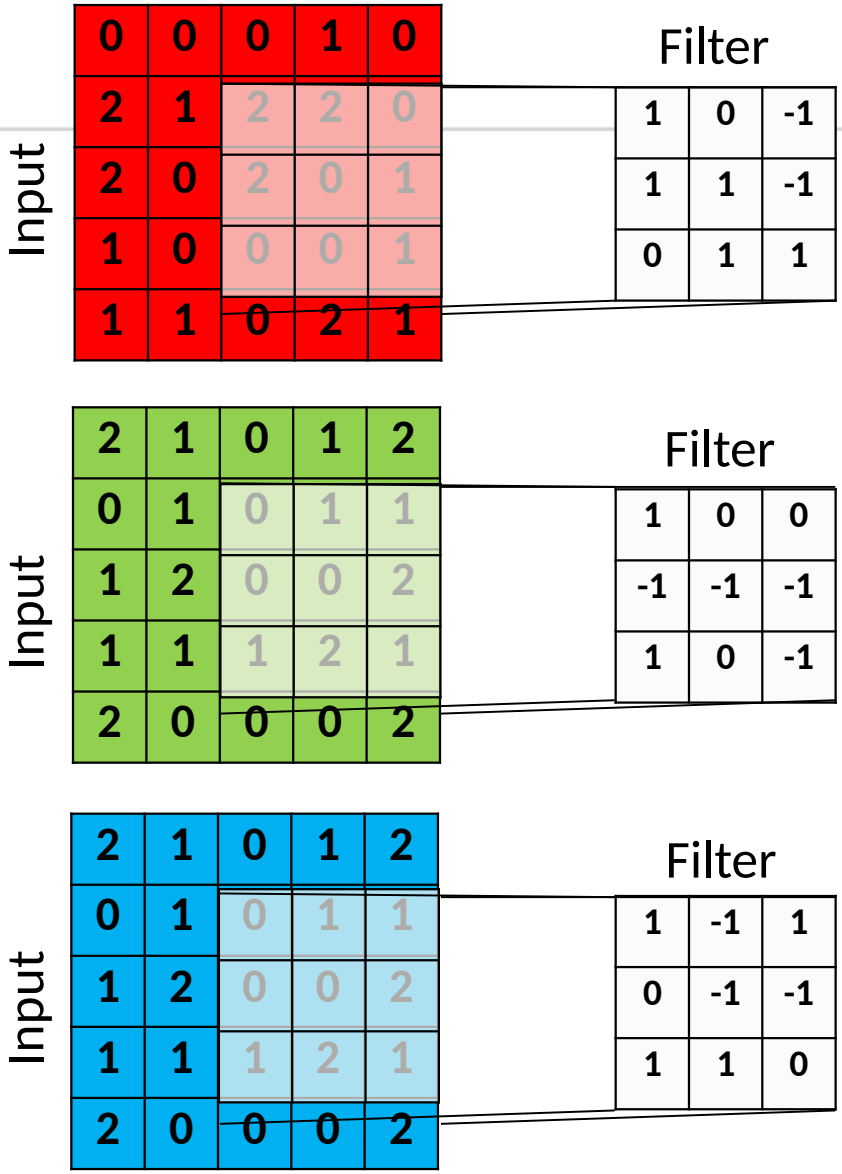


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

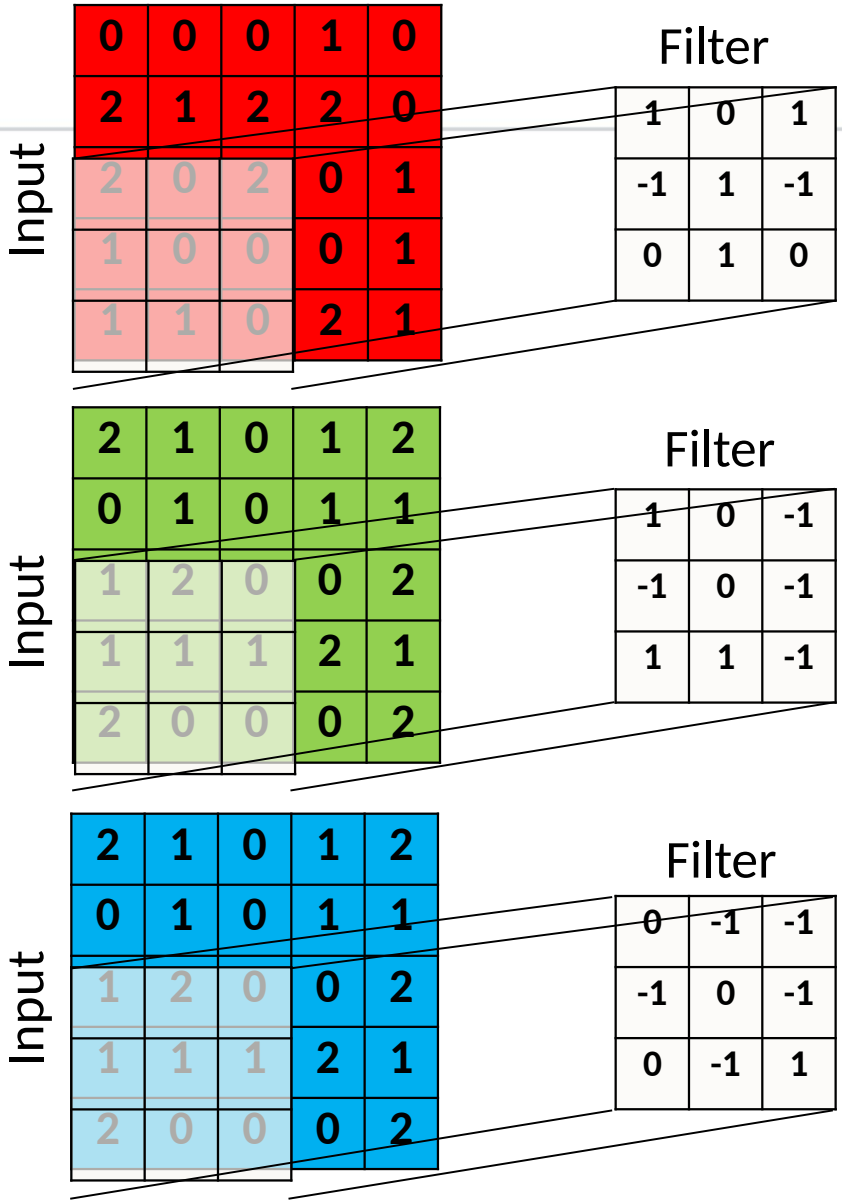


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

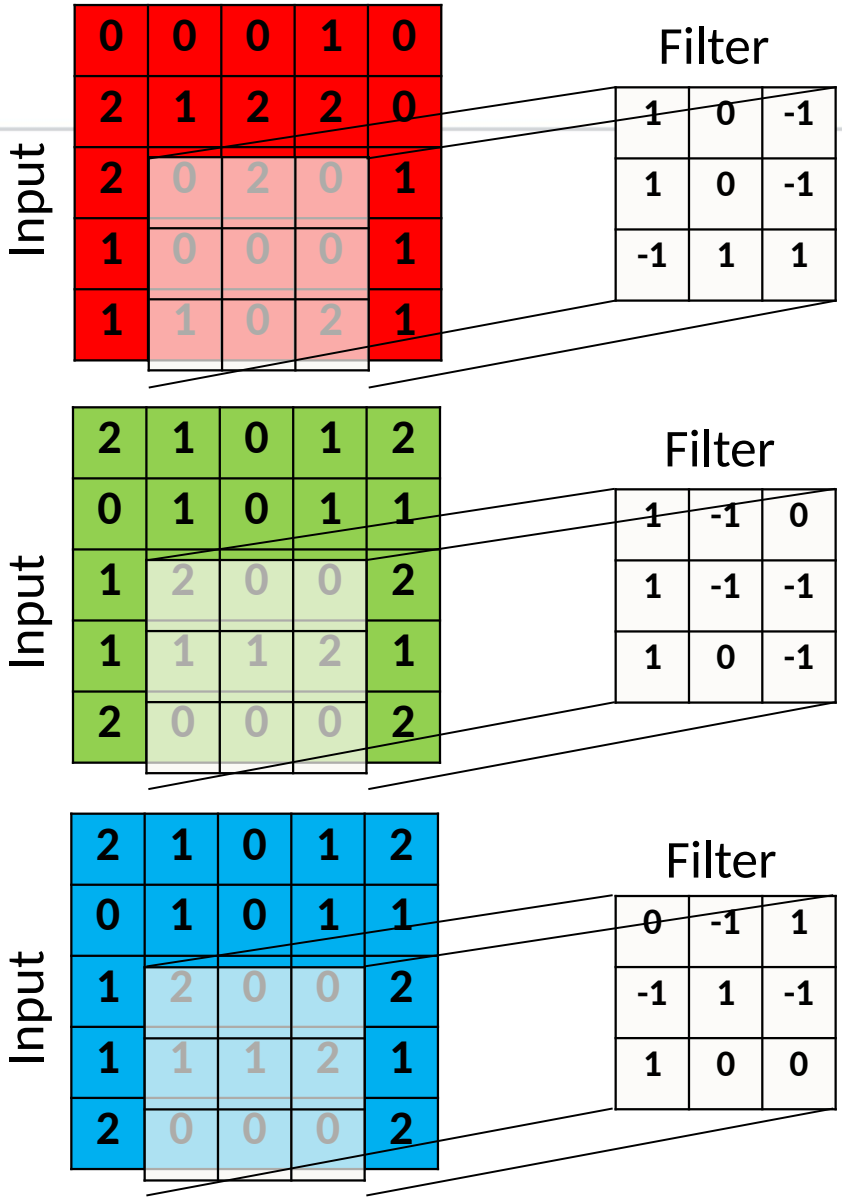


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

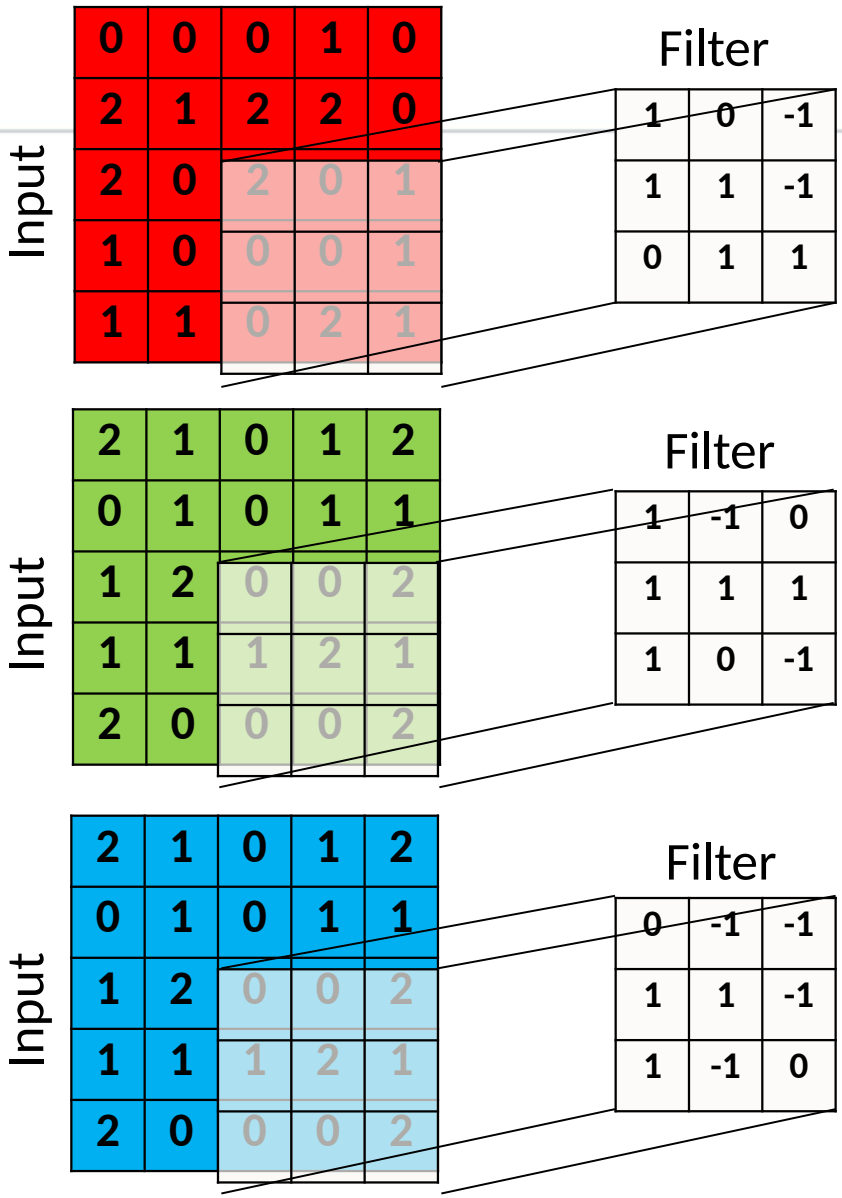


- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution



- Local connection: like convolution, but no sharing

Output $o_0(:,:,1)$

1	0	1
-1	1	1
0	0	-1

Unshared Convolution

Lancaster University

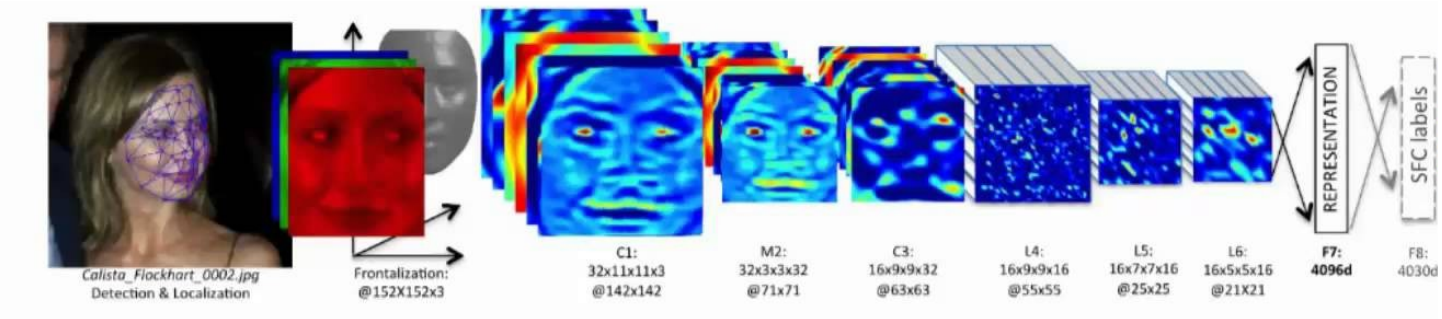
- Face recognition problem is one of the application of unshared convolution.
- Different regions of an aligned image have different local statistics.

DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Yaniv Taigman Ming Yang Marc'Aurelio Ranzato Lior Wolf

Facebook AI Research
Menlo Park, CA, USA
{yaniv, mingyang, ranzato}@fb.com

Tel Aviv University
Tel Aviv, Israel
wolf@cs.tau.ac.il



Variants of the Basic Convolution Function

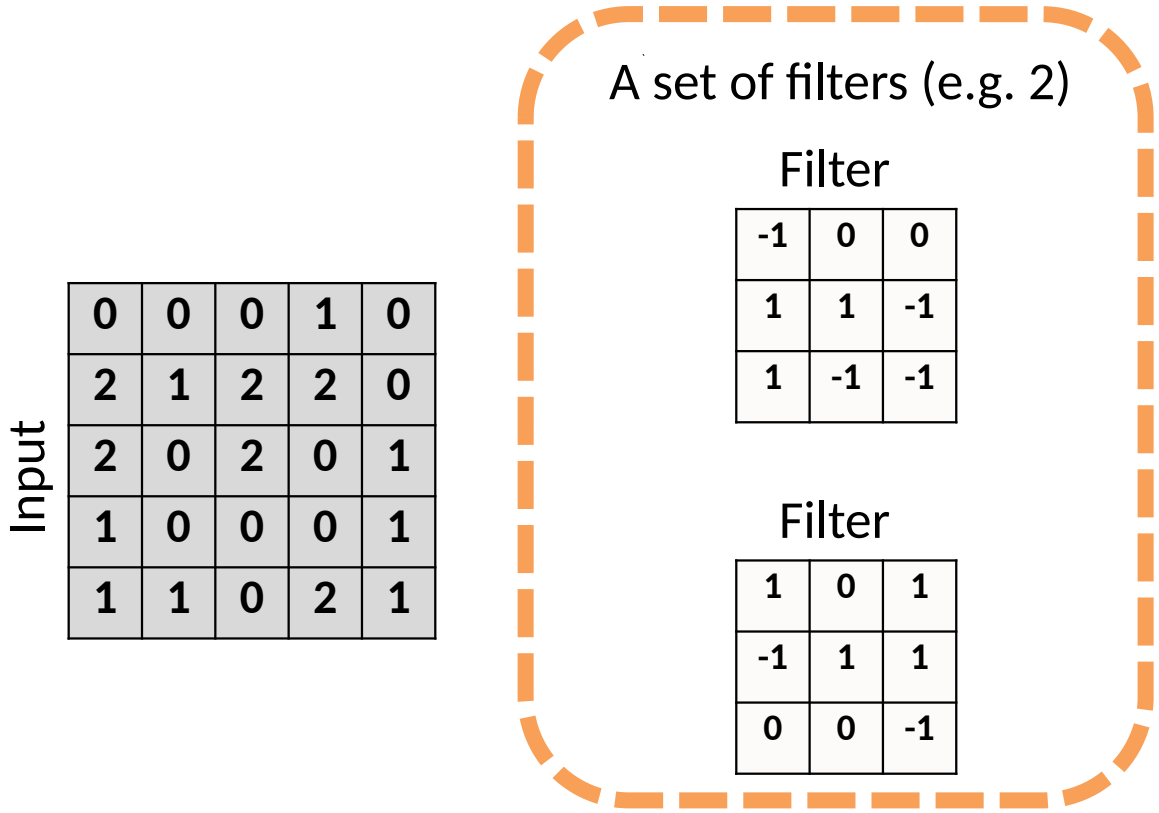
Lancaster University

- Stride
  - is the amount of down-sampling
- Zero padding
  - avoids layer-to-layer shrinking
- Unshared convolution
  - Like convolution but without sharing
- Tiled convolution
  - Cycle between shared parameter groups

Tiled Convolution

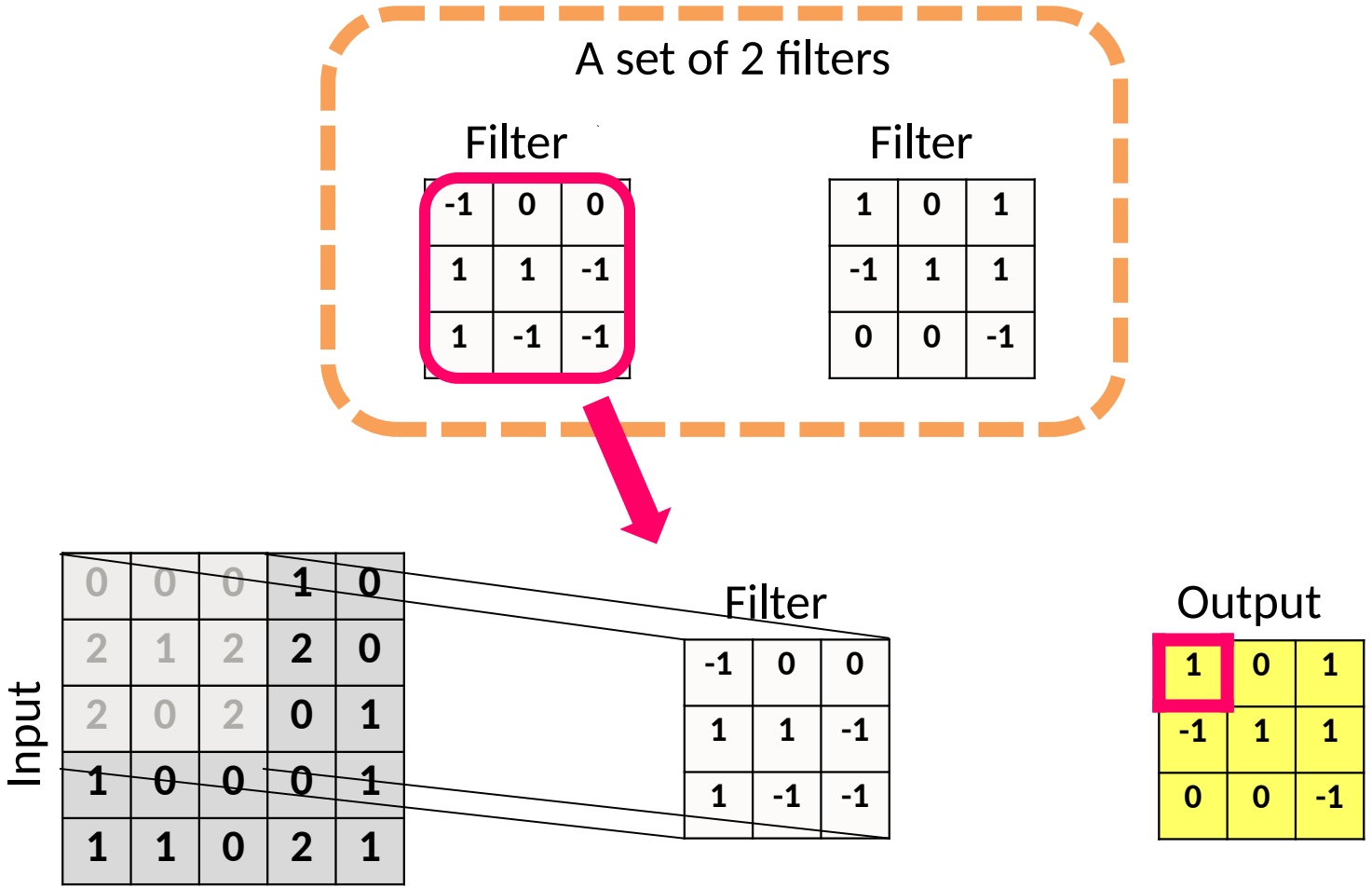
Lancaster University

- A compromise between convolutional layer and unshared convolution
- Cycle between shared parameter groups



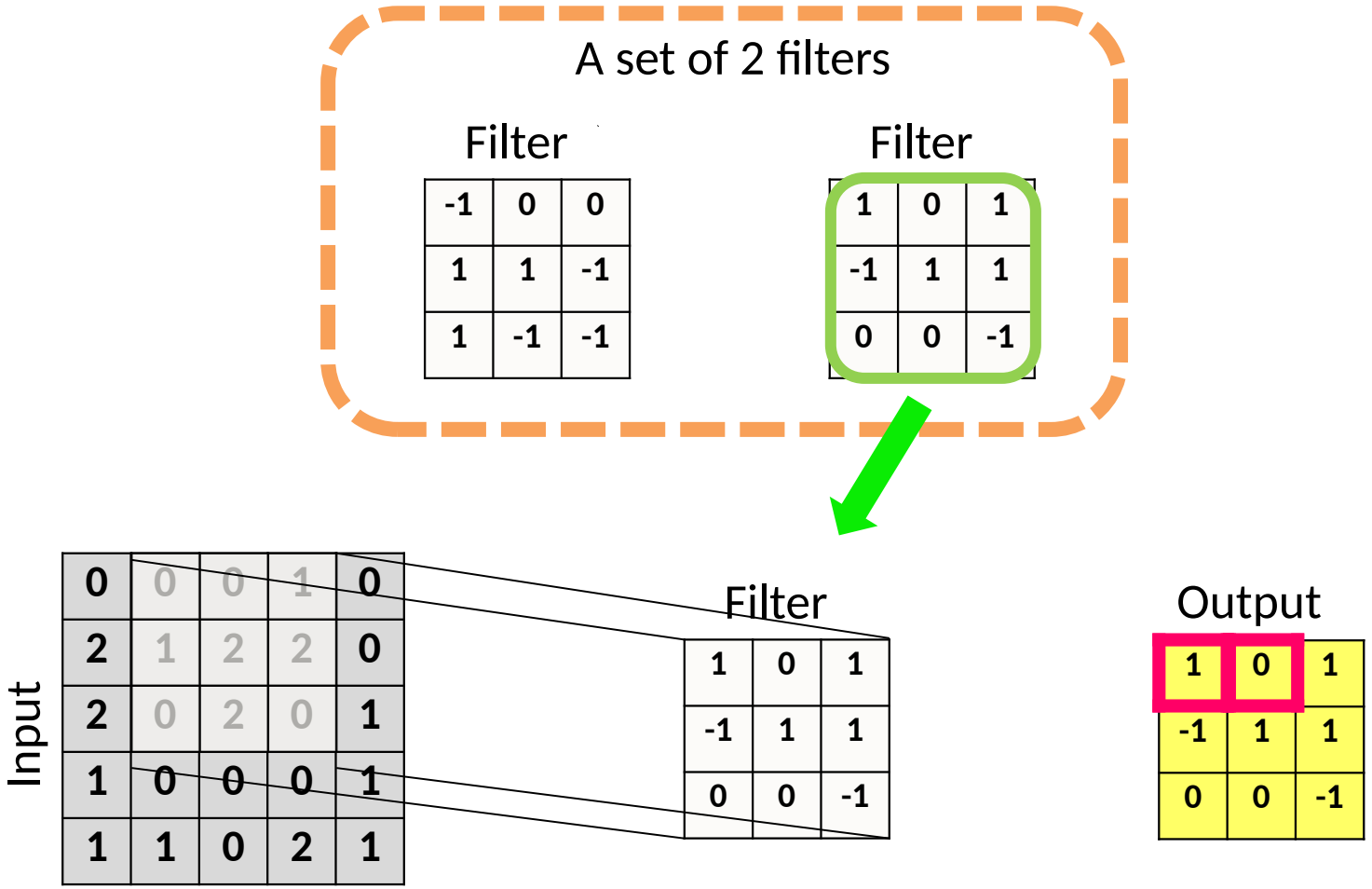
Tiled Convolution

Lancaster University

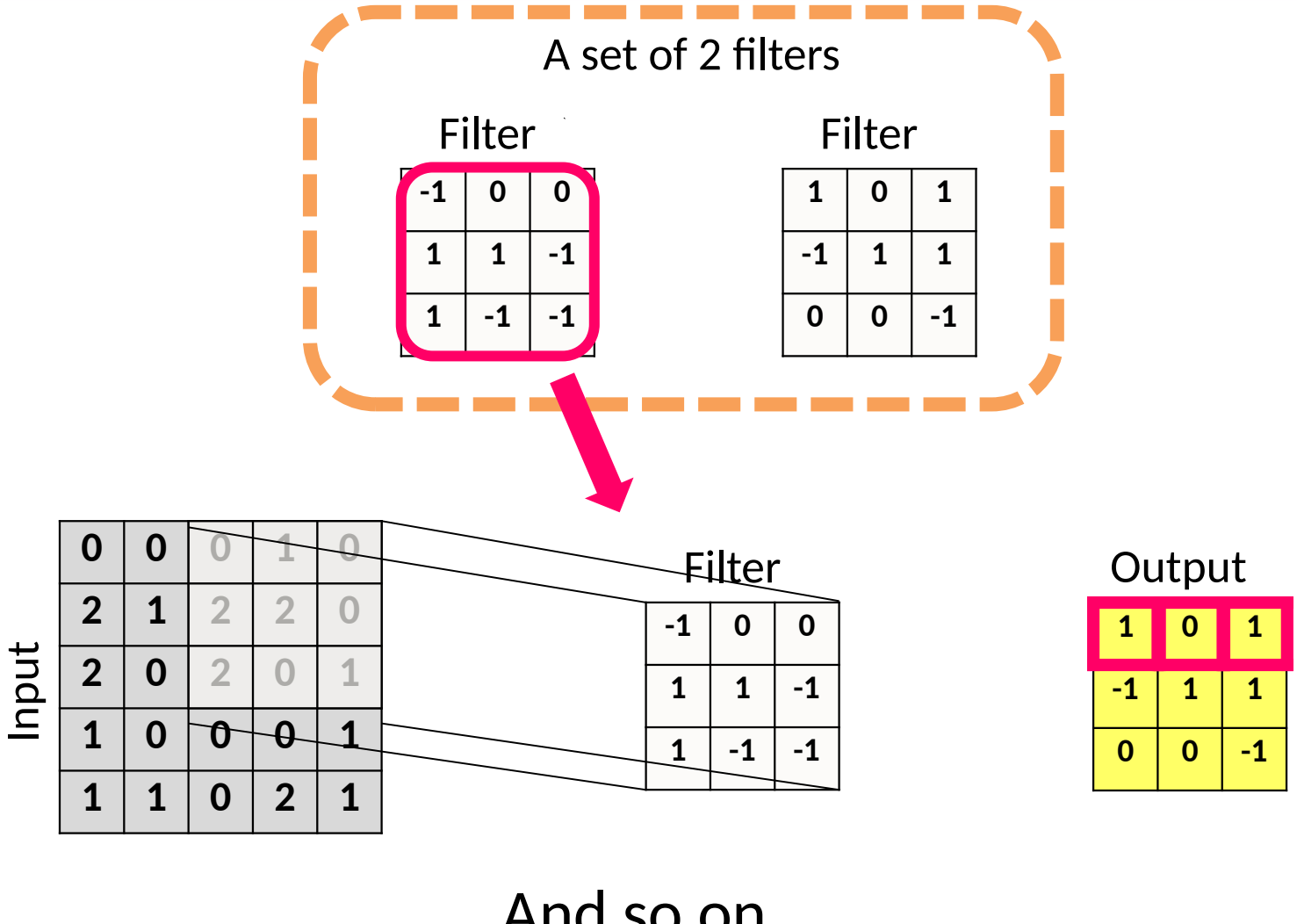


Tiled Convolution

Lancaster University



Tiled Convolution



And so on …

Tiled Convolution

Lancaster University

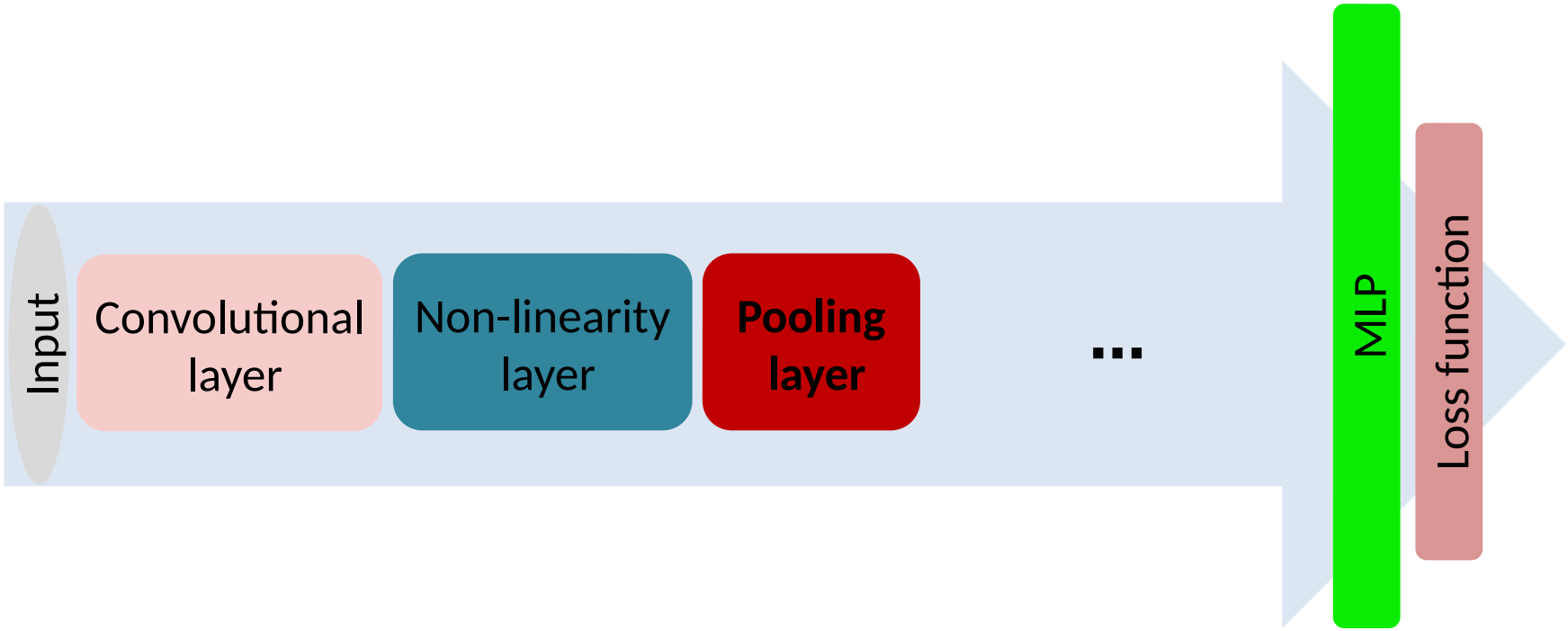
Tiled convolutional neural networks

Quoc V. Le, Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang Wei Koh, Andrew Y. Ng
Computer Science Department, Stanford University
{quocle,jngiam,zhenghao,danchia,pangwei,ang}@cs.stanford.edu

- It has shown that this model learn a more complex range of invariances, such as scale and rotational invariance beyond translational invariance.

Typical CNN Layers

Lancaster University



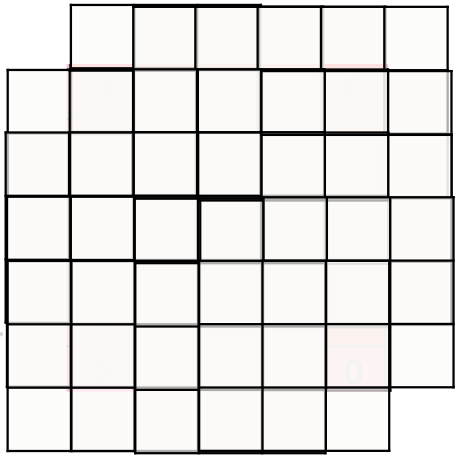
Outline

Lancaster University

- Multi Layer Perceptron (MLP)
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - Convolution Layer
  - **Activation Function**
  - Pooling Layer
  - Transposed Convolution (Deconv)
  - Un-pooling layer

Lancaster University

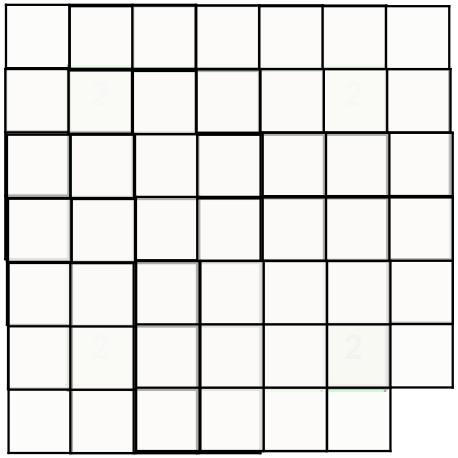
Input



Filter

-1	0	0
1	1	-1
1	-1	-1

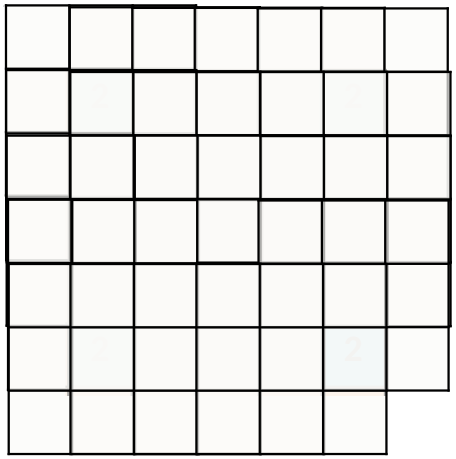
Input



Filter

1	0	1
-1	1	1
0	0	-1

Input I



Filter

-1	-1	-1
1	0	-1
-1	-1	1

Output

2	1	0	1	-2
0	-1	0	1	1
1	2	0	0	2
1	1	-1	-2	1
2	0	0	0	2

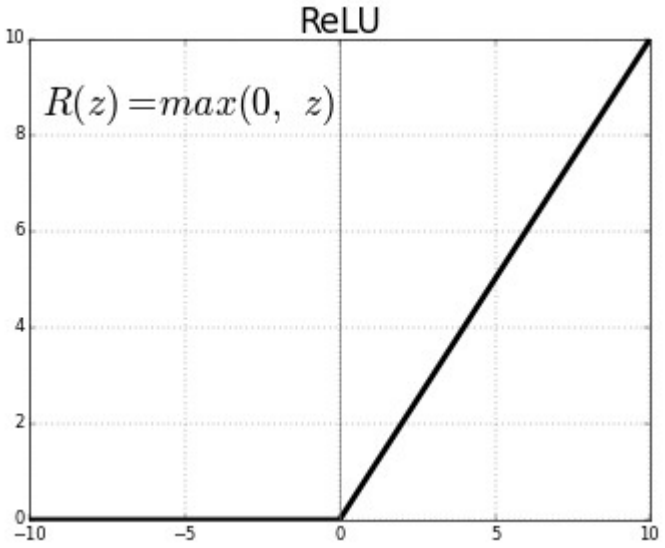
Convolution with Zero Padding
(output is)

Activation Functions

Lancaster University

- **Rectifier Linear Unit (ReLU)**
  - **ReLU** is a simple activation function
  - Quick computation
  - Most popular activation function for deep networks
  - Avoids saturation issues and makes learning faster

$$f_{relu}(x) = \max(0, x)$$



Variants:

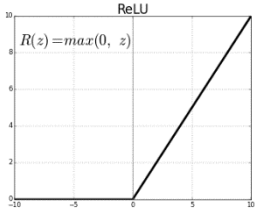
- Noisy ReLU
- Leaky ReLU
- Parametric Linear Units
- Randomized Leaky Rectier Linear Unit
- Exponential Linear Units

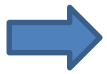
Activation Functions

Lancaster University

Output

2	1	0	1	-2
0	-1	0	1	1
1	2	0	0	2
1	1	-1	-2	1
2	0	0	0	2

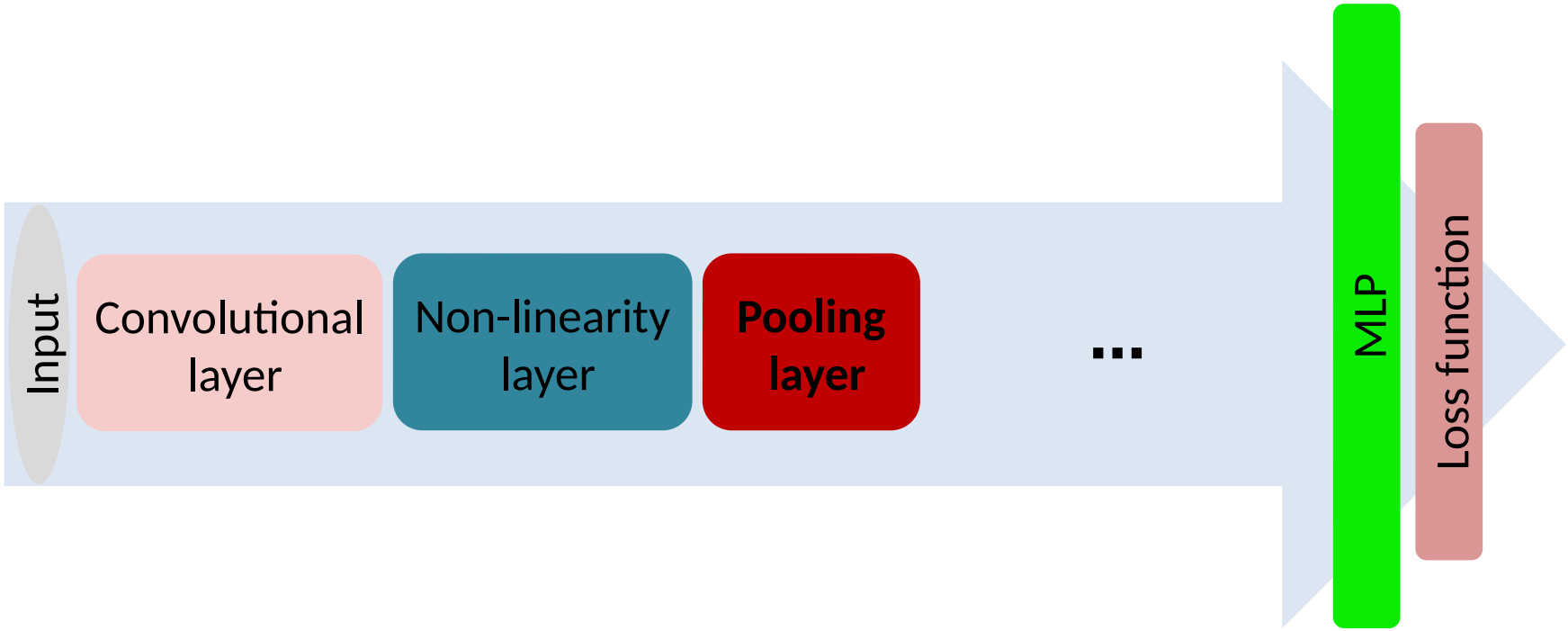




2	1	0	1	0
0	0	0	1	1
1	2	0	0	2
1	1	0	0	1
2	0	0	0	2

Typical CNN Layers

Lancaster University



Outline

Lancaster University

- Multi Layer Perceptron (MLP)
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - Convolution Layer
  - Activation Function
  - **Pooling Layer**
  - Transposed Convolution (Deconv)
  - Un-pooling layer

Pooling Layer

Lancaster University

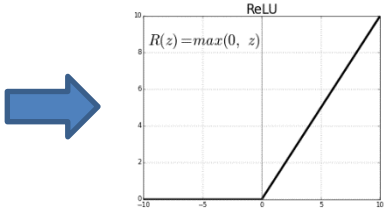
- Replaces the output of the network at a certain location with a summary statistic of the nearby outputs.
- Reduces computation for upper layers
- Non-linear down-sampling
- Provides translation invariance
  - Useful property, if we care more about whether some feature is present than exactly where it is, thus adds robustness to position
- Variants:
  - Max pooling (popular): reports the maximum output within a rectangular neighborhood
  - Average pooling: reports the average output
  - L2 norm of neighbourhood

Polling Layer

Lancaster University

Output

2	1	0	1	-2
0	-1	0	1	1
1	2	0	0	2
1	1	-1	-2	1
2	0	0	0	2

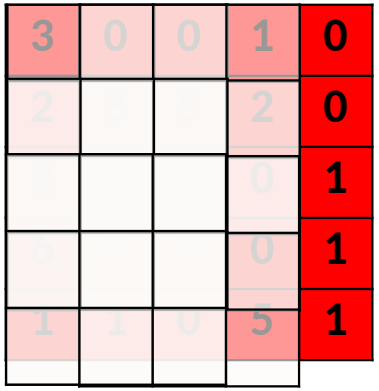


2	1	0	1	0
0	0	0	1	1
1	2	0	0	2
1	1	0	0	1
2	0	0	0	2

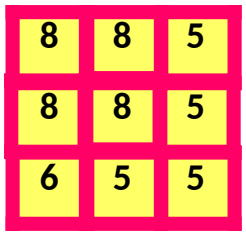
Pooling Layer

- The maximum activation is chosen from the selected block of values
- Similar to the convolution layer, we need to specify the **size of pooled region** and the **stride**.
- If the size of pooled region is given by , with a stride , the size of the output feature map can be given by:

Stride 1, size of pooled region



Input features map



Output after Max-Pooling

Pooling Layer

Lancaster University

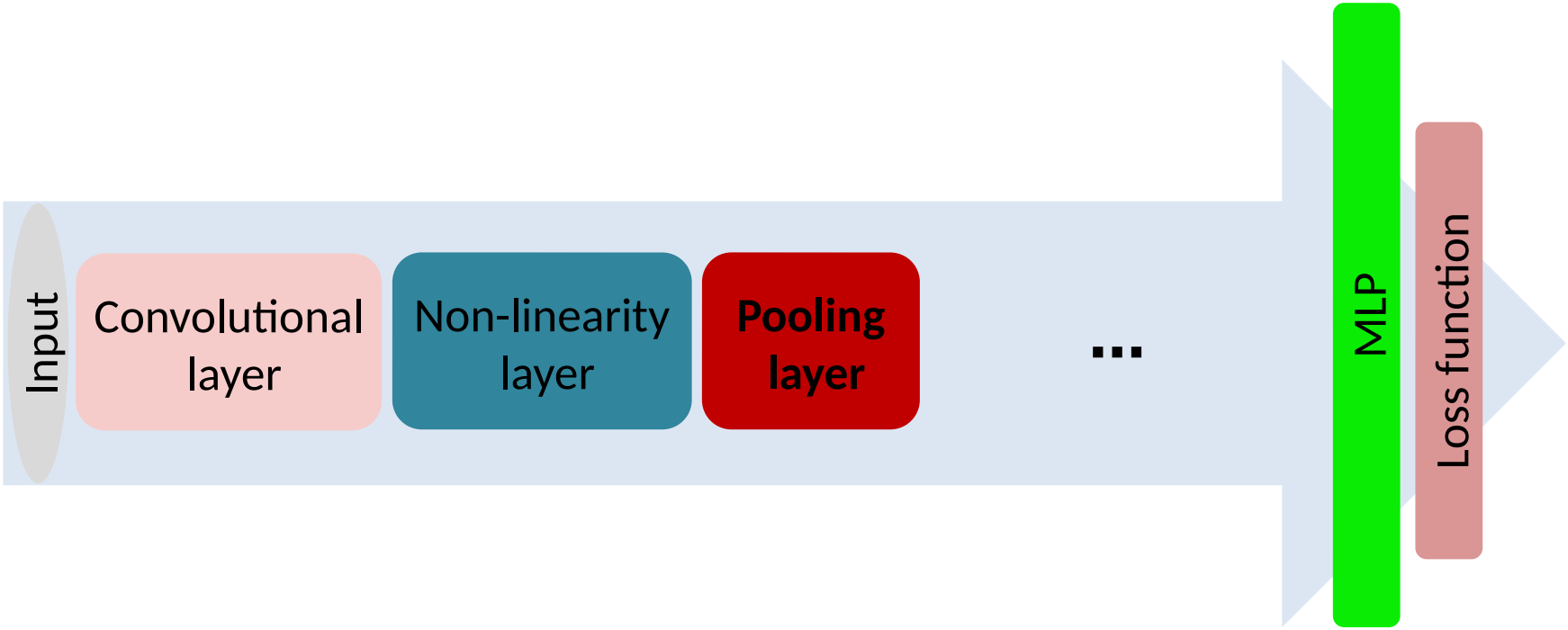
- **Common setting**:
  - receptive fields (i.e.),
  - a stride of 2 (i.e. s).
  - This discards exactly 75% of the activations in an input volume.

- Another slightly less common setting is to use receptive fields with a stride of 2.

- It is very uncommon to see receptive field sizes for max pooling that are larger than 3 because,
  - too lossy and aggressive
  - worse performance

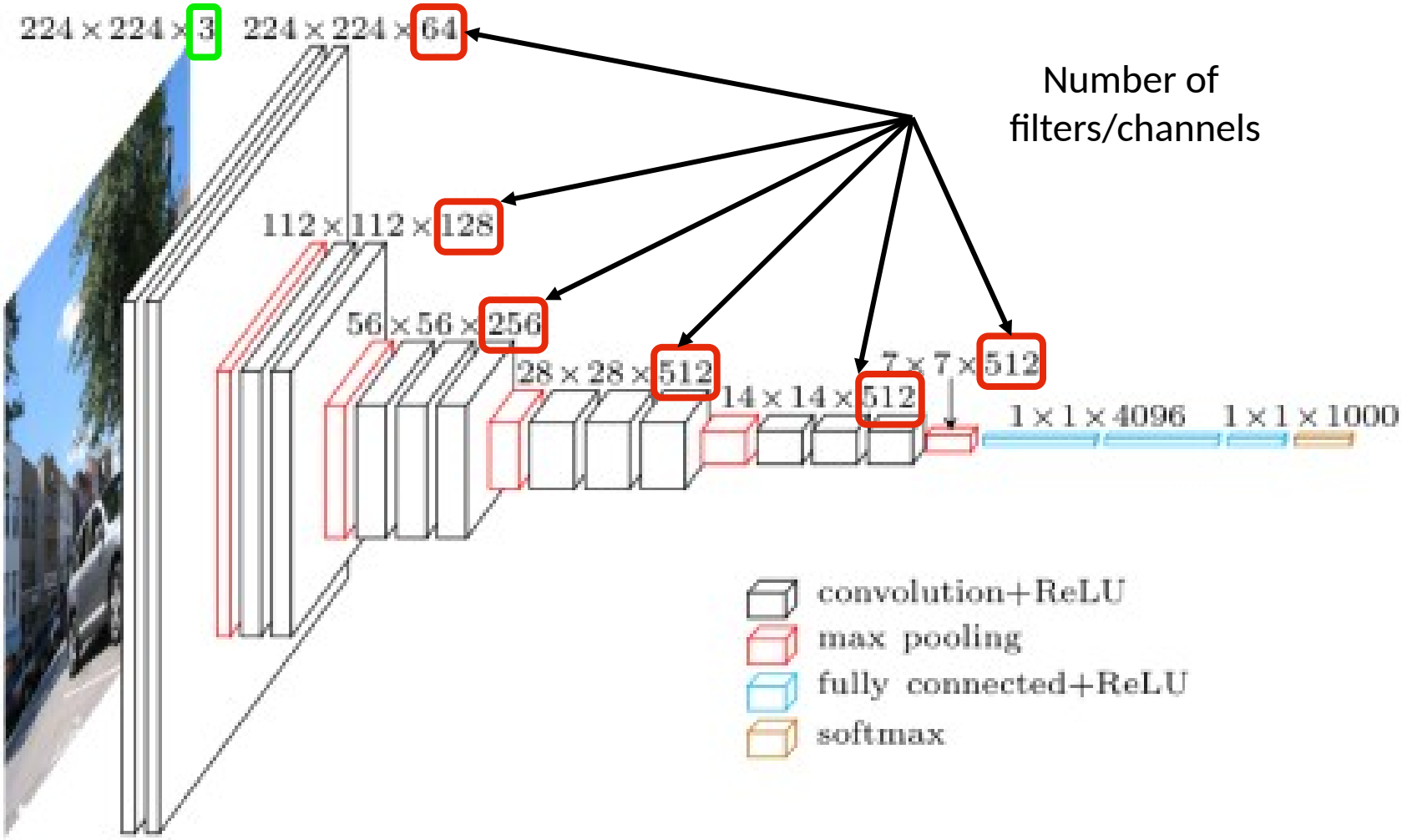
Typical CNN Layers
Putting It All Together!

Lancaster University



VGGNET (Example)

Lancaster University



Multiple Filters-Multiple Outputs

Lancaster University

Input

0	0	0	1	0
2	1	2	2	0
2	0	2	0	1
1	0	0	0	1
1	1	0	2	1

Filter

-1	0	0
1	1	-1
1	-1	-1

Output

1	0	1
-1	1	1
0	0	-1

Input

2	1	0	1	2
0	1	0	1	1
1	2	0	0	2
1	1	1	2	1
2	0	0	0	2

Filter

1	0	1
-1	1	1
0	0	-1

Input

2	1	0	1	2
0	1	0	1	1
1	2	0	0	2
1	1	1	2	1
2	0	0	0	2

Filter

-1	-1	-1
1	0	-1
-1	-1	1

Multiple Filters-Multiple Outputs

Lancaster University

Input

0	0	0	1	0
2	1	2	2	0
2	0	2	0	1
1	0	0	0	1
1	1	0	2	1

Filter

1	0	0
-1	-1	-1
1	0	1

Output

1	0	1
-1	1	1
0	0	-1

Input

2	1	0	1	2
0	1	0	1	1
1	2	0	0	2
1	1	1	2	1
2	0	0	0	2

Filter

1	-1	-1
1	1	0
1	-1	-1

Output

-1	0	1
1	-1	1
1	0	1

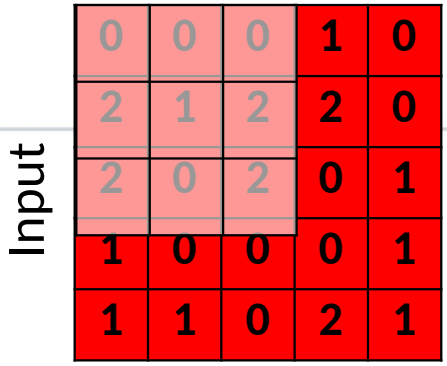
Input

2	1	0	1	2
0	1	0	1	1
1	2	0	0	2
1	1	1	2	1
2	0	0	0	2

Filter

-1	1	-1
-1	-1	-1
-1	1	-1

Multiple Filters-Multiple Outputs

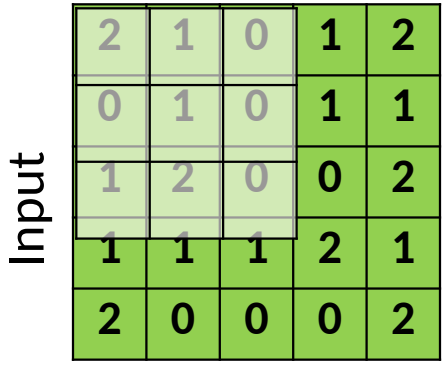


Filter

0	-1	0
-1	0	0
-1	0	1

Output

1	0	1
-1	1	1
0	0	-1

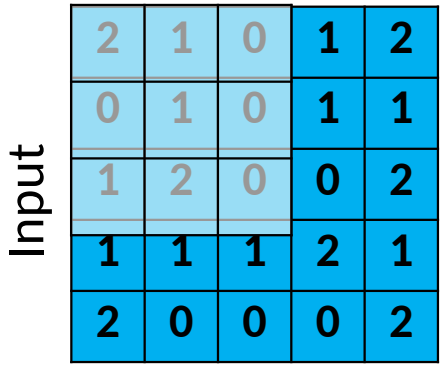


Filter

1	-1	-1
0	-1	0
-1	0	-1

Output

-1	0	1
1	-1	1
1	0	1



Filter

0	1	-1
1	0	-1
0	1	-1

Output

1	0	1
-1	1	1
0	0	-1

E.g. 3 filters ⇒ 3 outputs

Lancaster University

Questions?

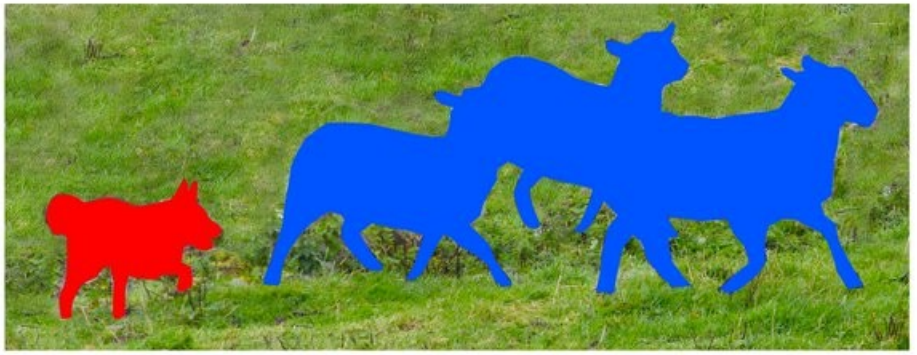
Outline

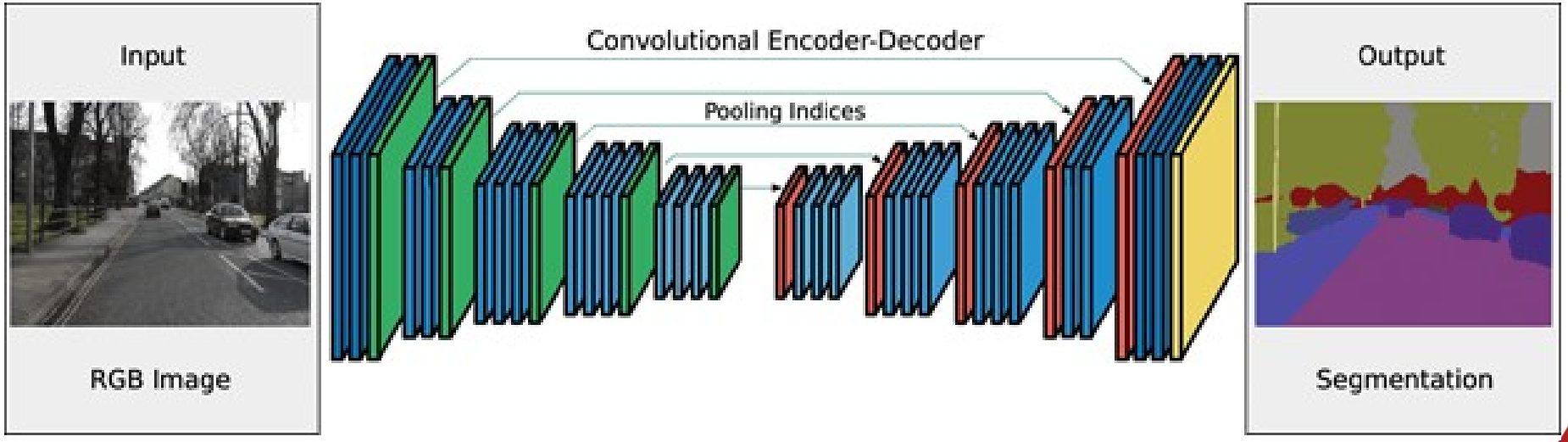
Lancaster University

- Multi Layer Perceptron (MLP)
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - Convolution Layer
  - Activation Function
  - Pooling Layer
  - **Transposed Convolution (Deconv)**
  - **Un-pooling layer**

Semantic Segmentation

Lancaster University





Outline

Lancaster University

- Multi Layer Perceptron (MLP)
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - Convolution Layer
  - Activation Function
  - Pooling Layer
  - **Transposed Convolution (Deconv)**
  - Un-pooling layer

Transposed Convolution

Lancaster University

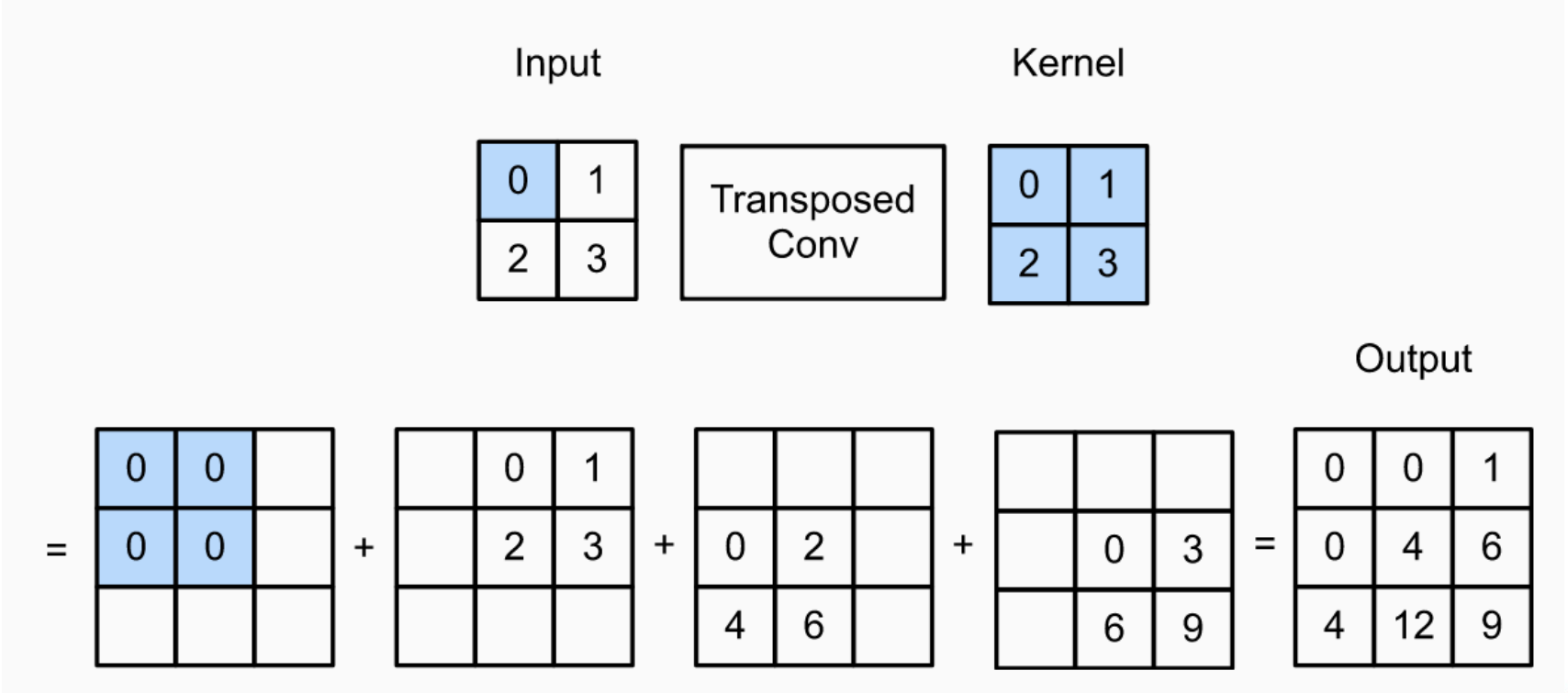
- The CNN layers we have seen so far, such as convolutional layers and pooling layers, typically reduce (downsample) the spatial dimensions (height and width) of the input or keep them unchanged.

- In semantic segmentation that classifies at pixel-level, it will be convenient if the spatial dimensions of the input and output are the same.

Transposed Convolution (Example)

Lancaster University

- transposed convolution with a 2×2 kernel is computed for a 2×2 input.



Outline

Lancaster University

- Multi Layer Perceptron (MLP)
  - Why it is not good for images
- Convolutional Neural Networks (CNN)
  - Convolution Layer
  - Activation Function
  - Pooling Layer
  - Transposed Convolution (Deconv)
  - **Un-pooling layer**

Un-Pooling Layer

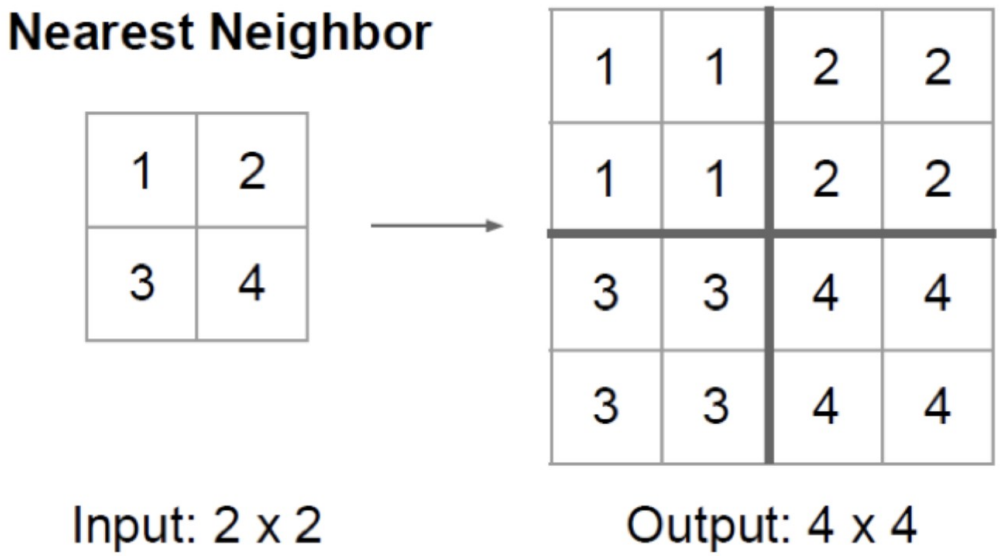
Lancaster University

- Max-unpooling is an upsampling procedure
- Max pooling is an non-invertable operation, instead we obtain an approximation
- Approaches include:
  - Nearest-Neighbor
  - Bed of Nails
  - Max Unpooling

Un-Pooling Layer (Nearest-Neighbor)

Lancaster University

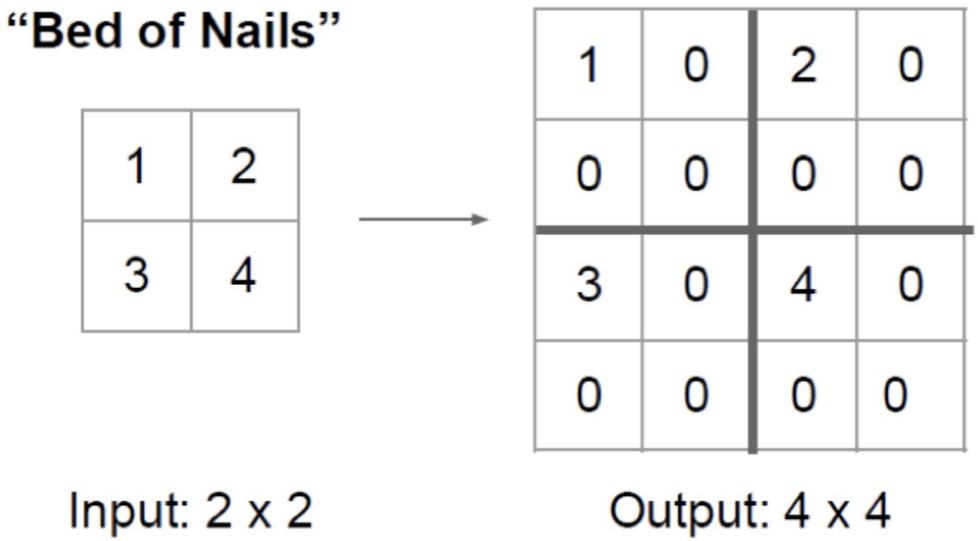
- Although its simplicity, the problem of this approach is that the **output structure becomes blocky** as all pixels in each subregion have the same value.



Un-Pooling Layer (Bed of Nails)

Lancaster University

- By doing so, it achieves the fine-grained output structure. However, **the upsampled elements always have a fixed location**, which is the upper-left corner



Un-Pooling Layer (Max Unpooling)

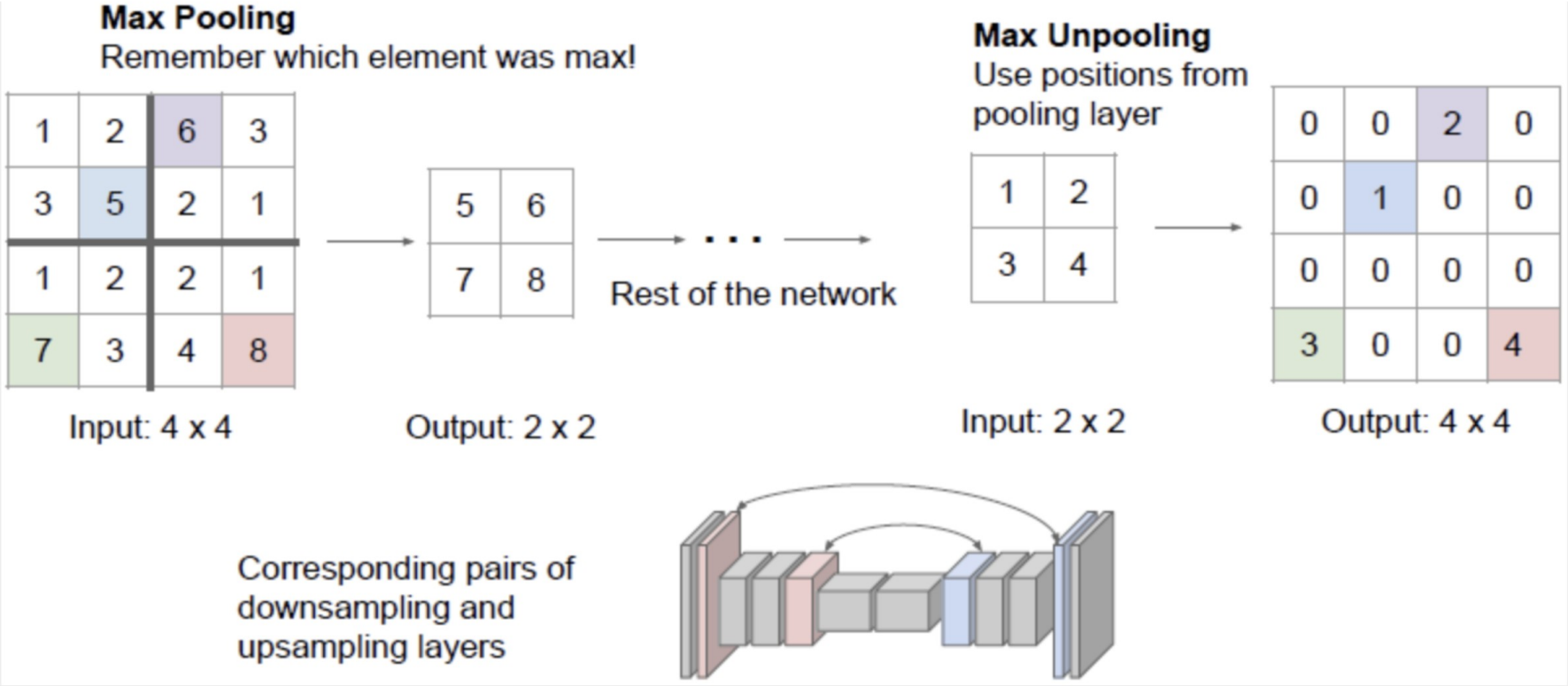


Figure 5. Illustration of Max Unpooling, from [11]

Lancaster University

Questions?