

EE2211 Introduction to Machine Learning

Lecture 9

Vincent Tan
vtan@nus.edu.sg

Electrical and Computer Engineering Department
National University of Singapore

*Acknowledgement: EE2211 development team
Thomas, Helen, Xinchao, Kar-Ann, Chen Khong, Robby and Haizhou*

Course Contents

- Introduction and Preliminaries (Xinchao)
 - Introduction
 - Data Engineering
 - Introduction to Probability and Statistics
- Fundamental Machine Learning Algorithms I (Vincent)
 - Systems of linear equations
 - Least squares, Linear regression
 - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Vincent)
 - Over-fitting, bias/variance trade-off
 - Optimization, Gradient descent
 - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
 - Performance Issues
 - K-means Clustering
 - Neural Networks

Fundamental ML Algorithms: Decision Trees, Random Forest

Module III Contents

- Overfitting, underfitting and model complexity
- Bias-variance trade-off
- Regularization
- Loss function
- Optimization
- Gradient descent
- Decision trees]
- Random forest]

Review

Classification $y \in \{-1, +1\}$

Regression $y \in \mathbb{R}$

$x \in \mathbb{R}^d$



- Supervised learning: given feature(s) x , we want to predict target y to be some $f(x)$
 - If y is continuous, problem is called “regression”
 - If y is discrete, problem is called “classification”

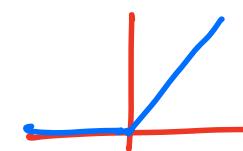
Review

- Supervised learning: given feature(s) x , we want to predict target y to be some $f(x)$
 - If y is continuous, problem is called “regression”
 - If y is discrete, problem is called “classification”
- Previous lectures used linear models for regression (and classification)
 - Nonlinearity added by using polynomial regression or other learning models

Activation
function

$$\sigma(a) = \frac{1}{1 + e^{-a}} \quad \text{sigmoid}$$

$$\sigma(a) = \text{ReLU}(a)$$



$$f(x_i, \underline{w}) = [\underline{x}_i^T]^\top [\underline{w}]$$

affine

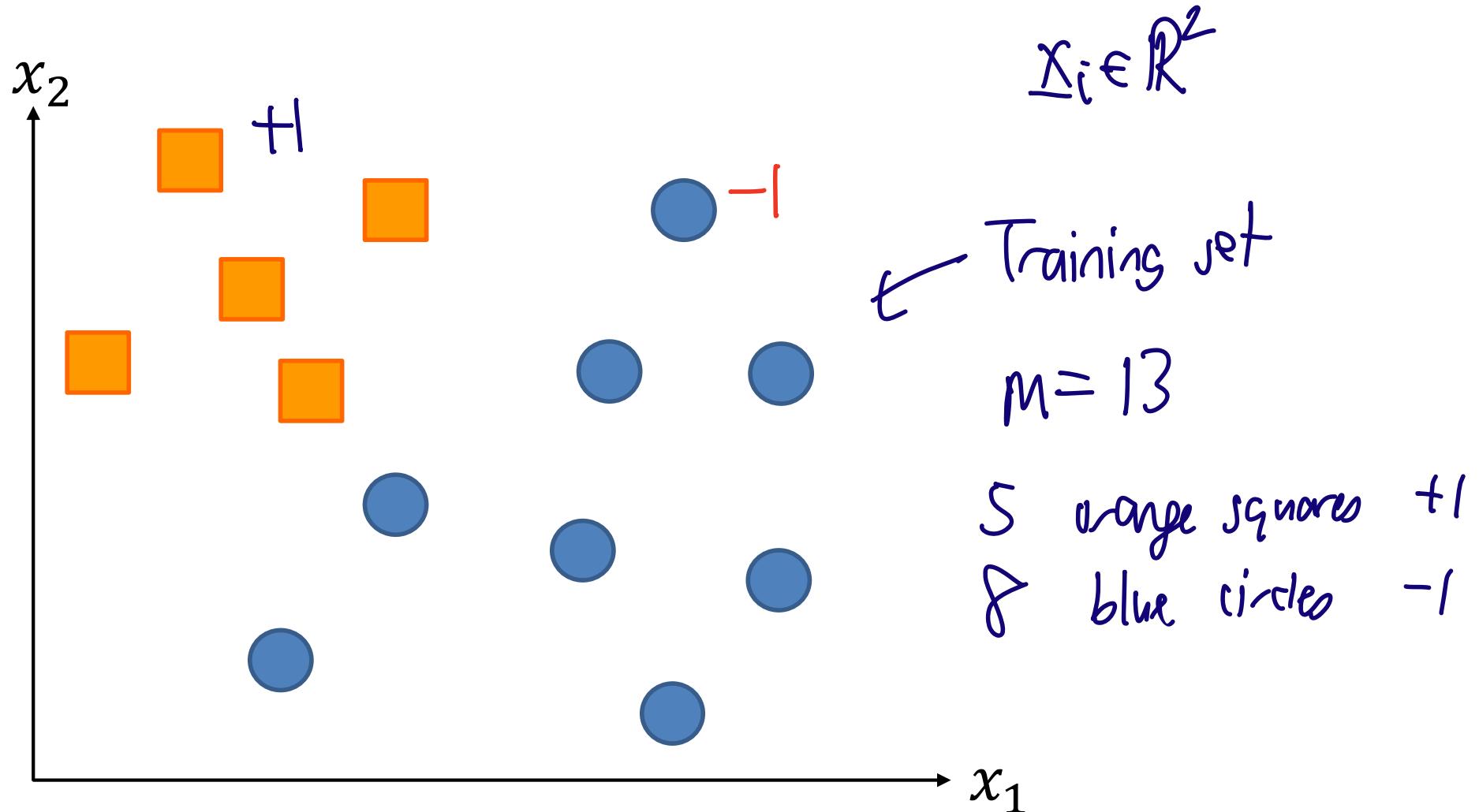
$$= \underline{p}_i^\top \underline{w}$$

Review

- Supervised learning: given feature(s) x , we want to predict target y to be some $f(x)$
 - If y is continuous, problem is called “regression”
 - If y is discrete, problem is called “classification”
- Previous lectures used linear models for regression (and classification)
 - Nonlinearity added by using polynomial regression or other learning models
- New approach today: trees

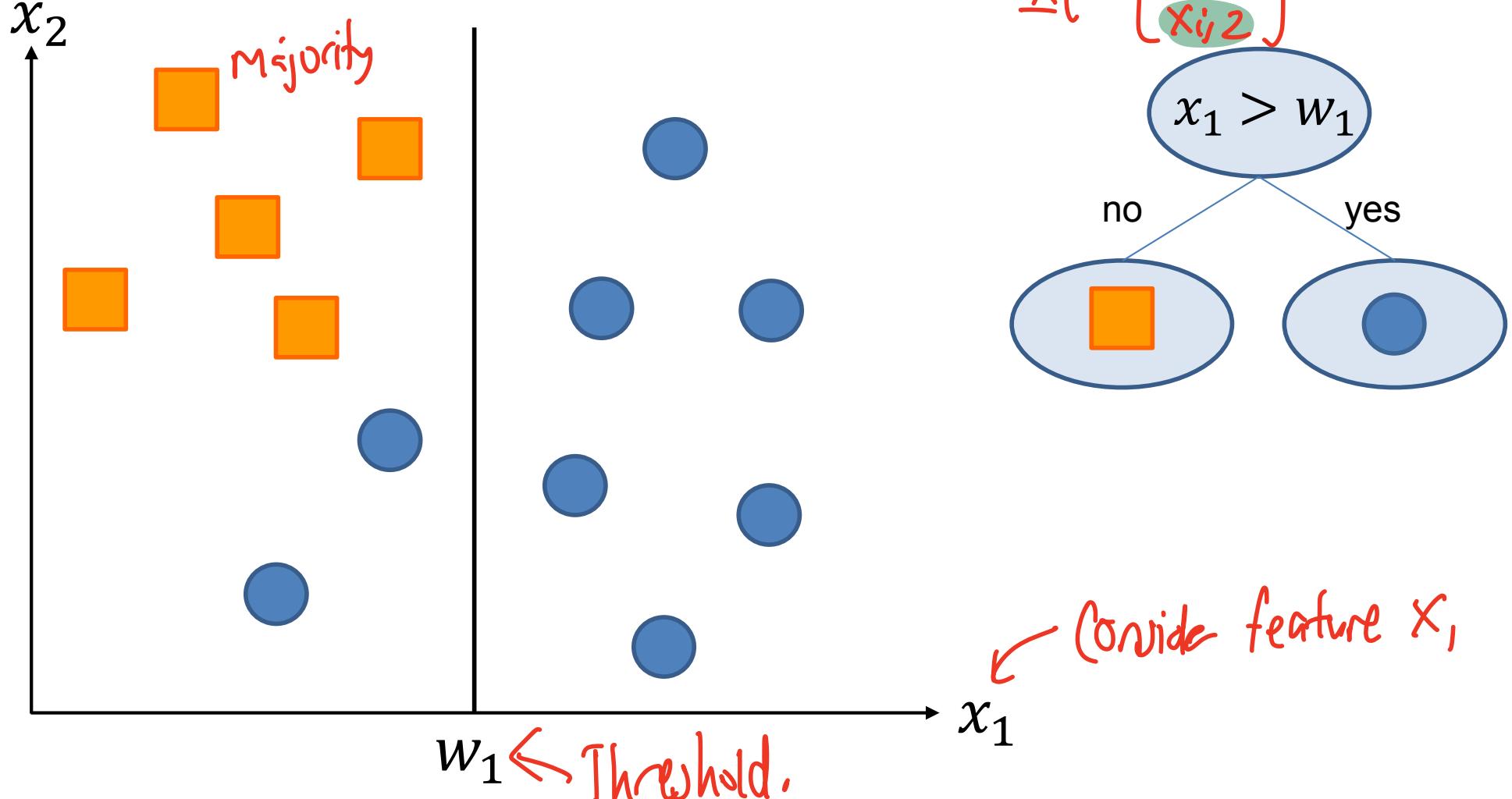
Decision Tree Classification Example

- Goal: predict class labels using two features x_1 & x_2



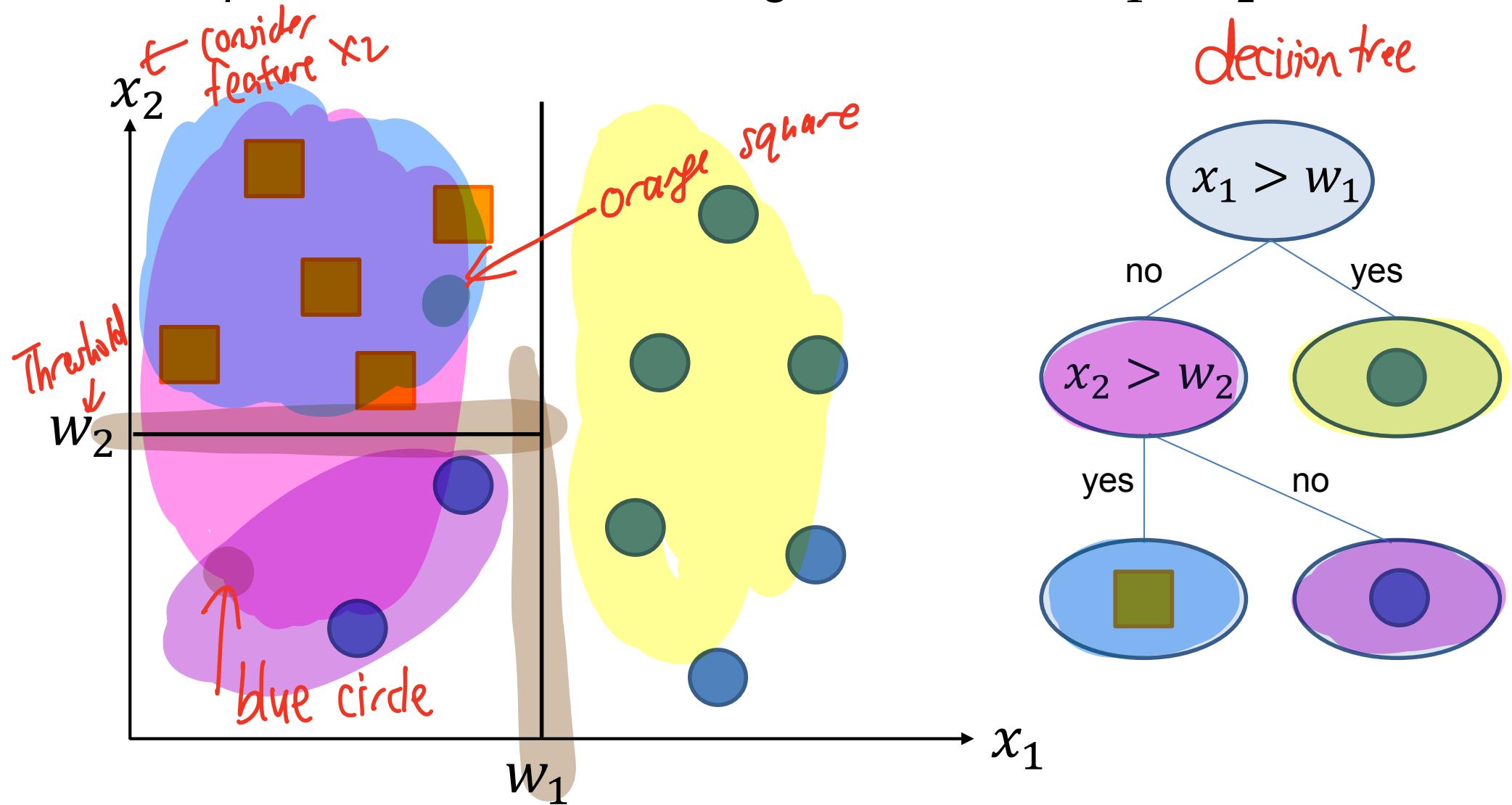
Decision Tree Classification Example

- Goal: predict class labels using two features x_1 & x_2



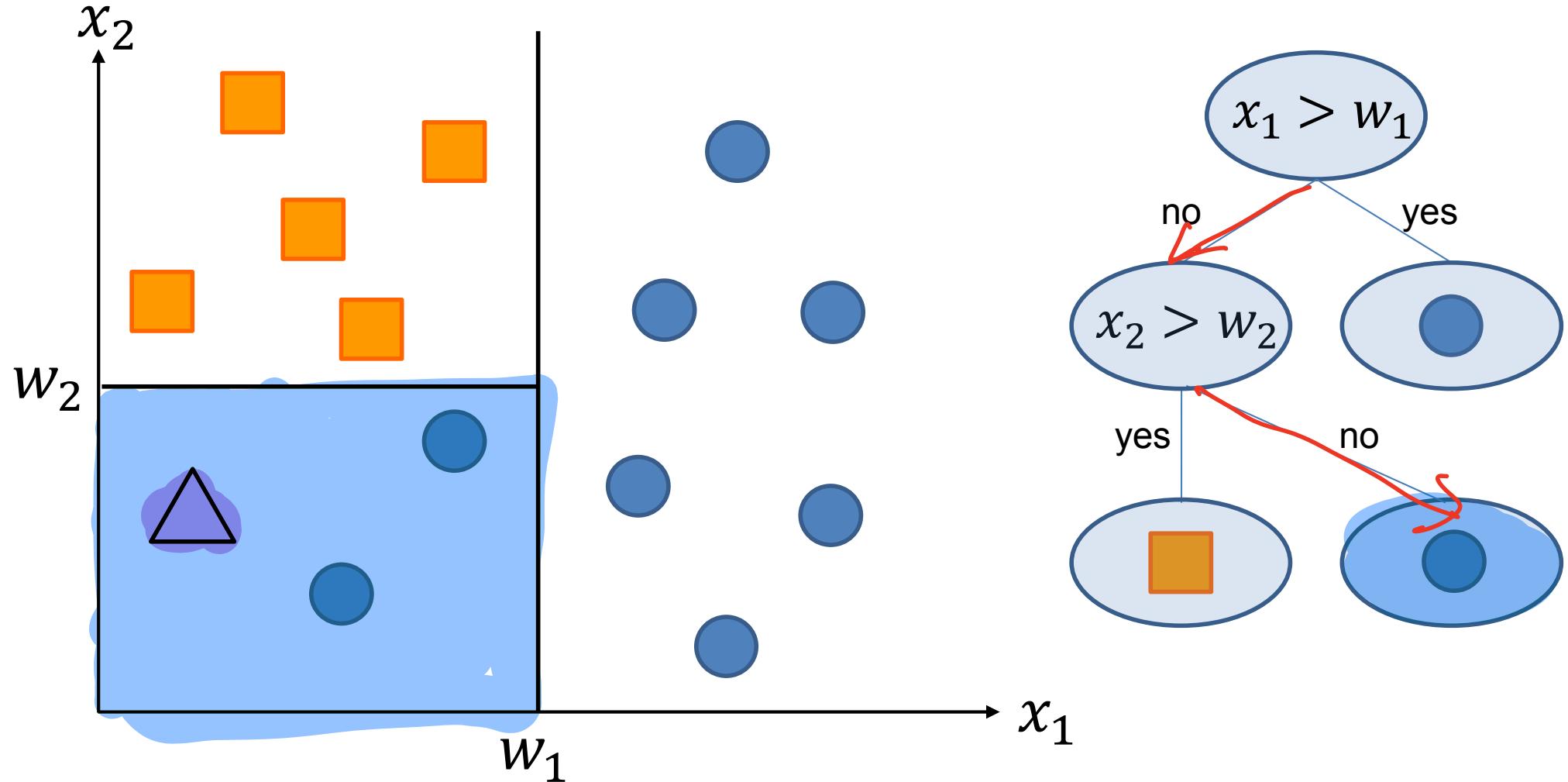
Decision Tree Classification Example

- Goal: predict class labels using two features x_1 & x_2



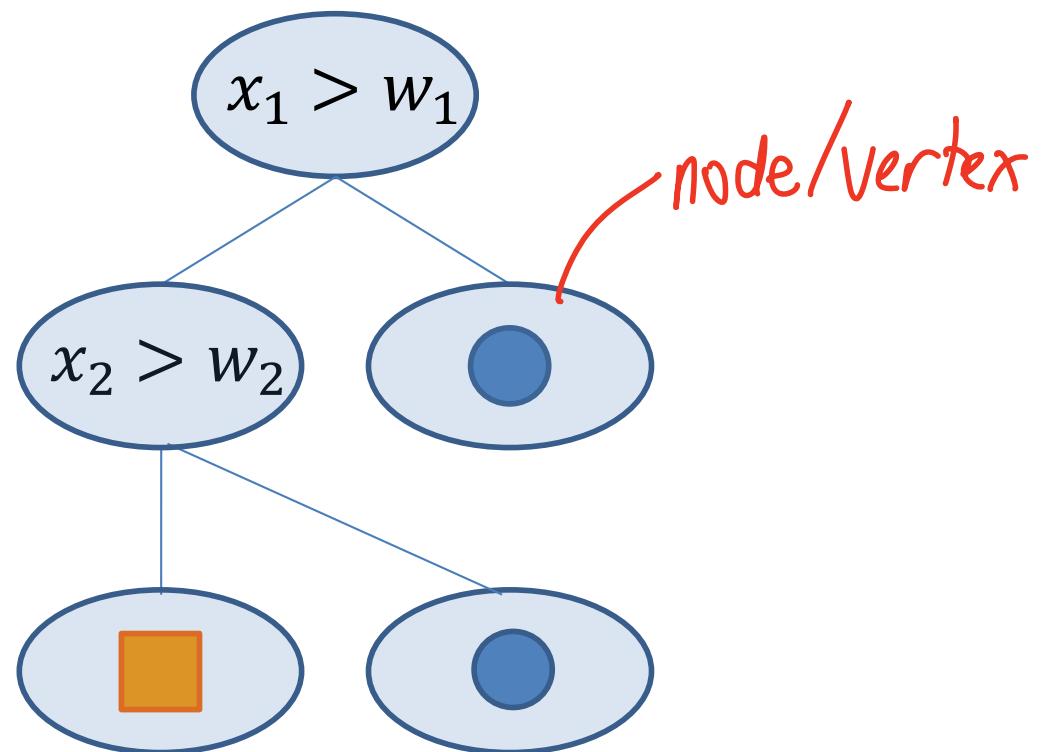
Decision Tree Classification Example

- In our test set, we observe datapoint Δ shown below. How would the decision tree classify this point?

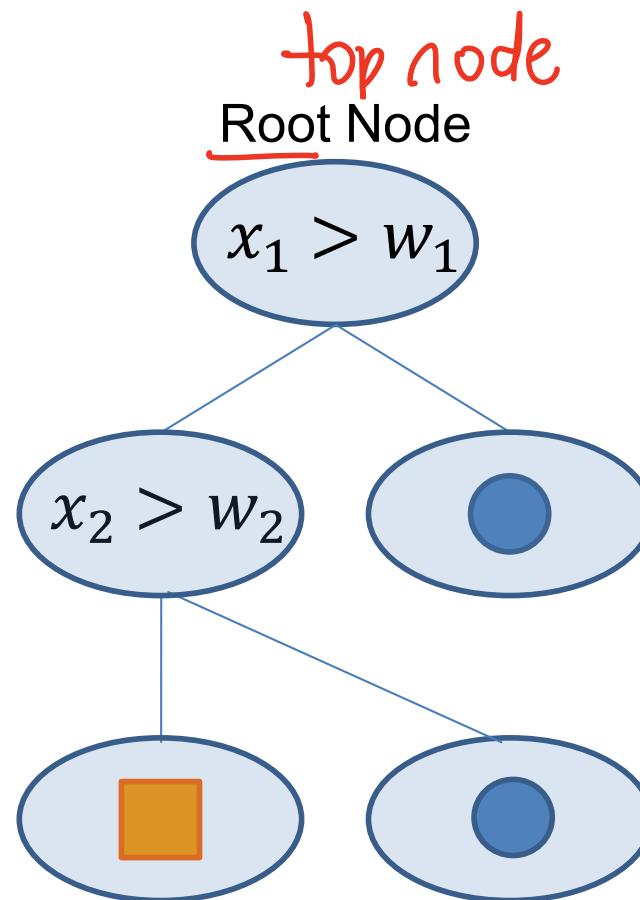


Questions?

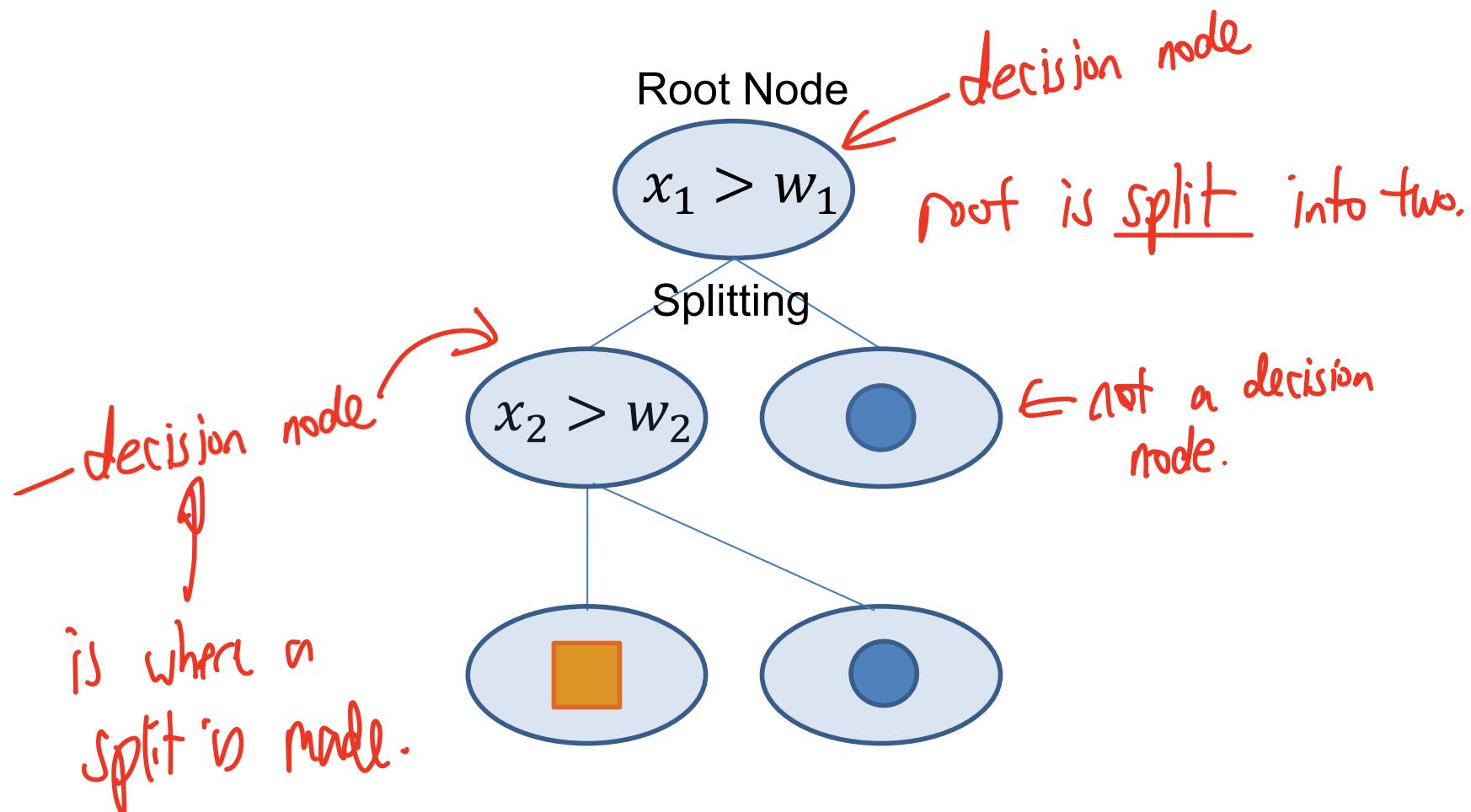
Basic Terminologies



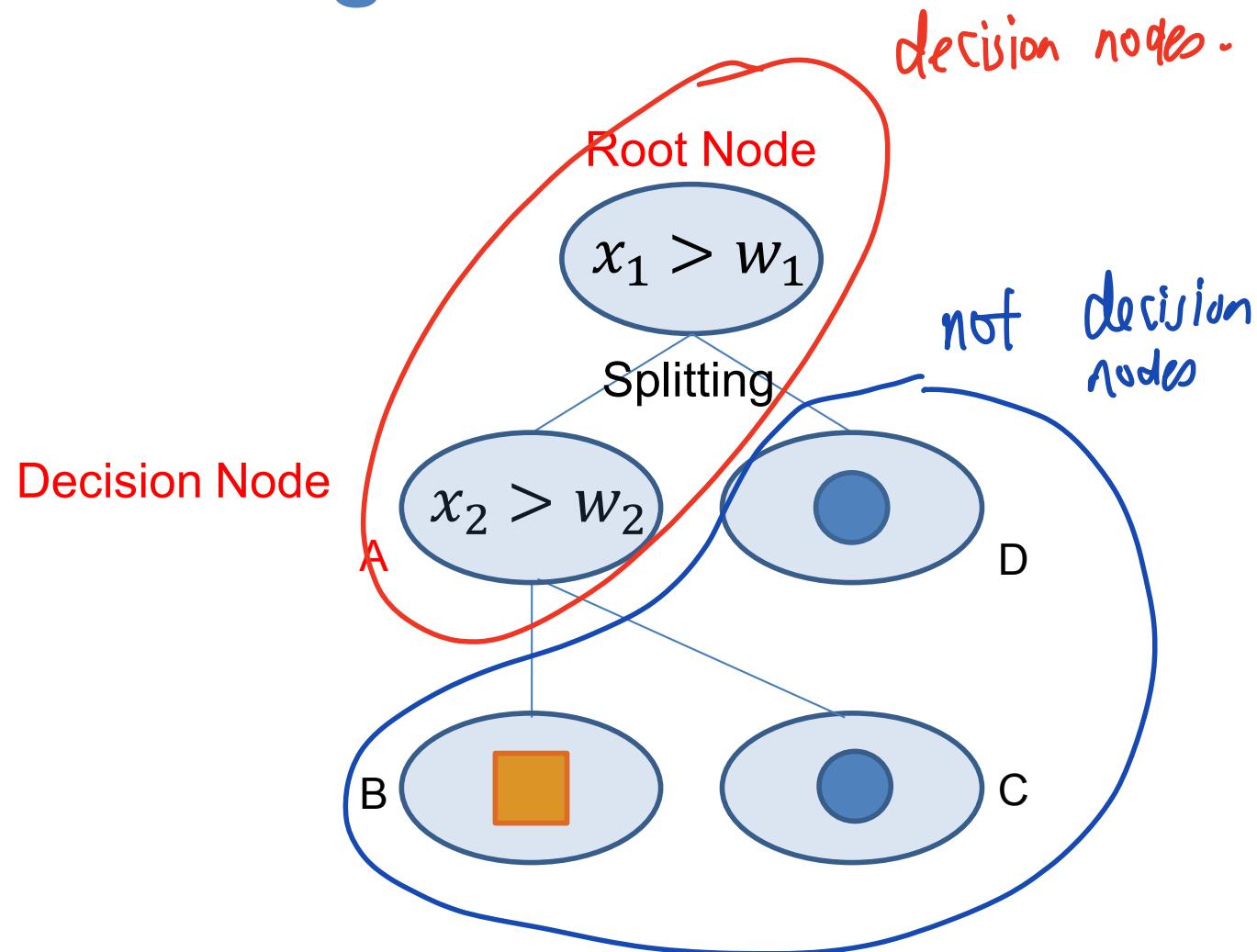
Basic Terminologies



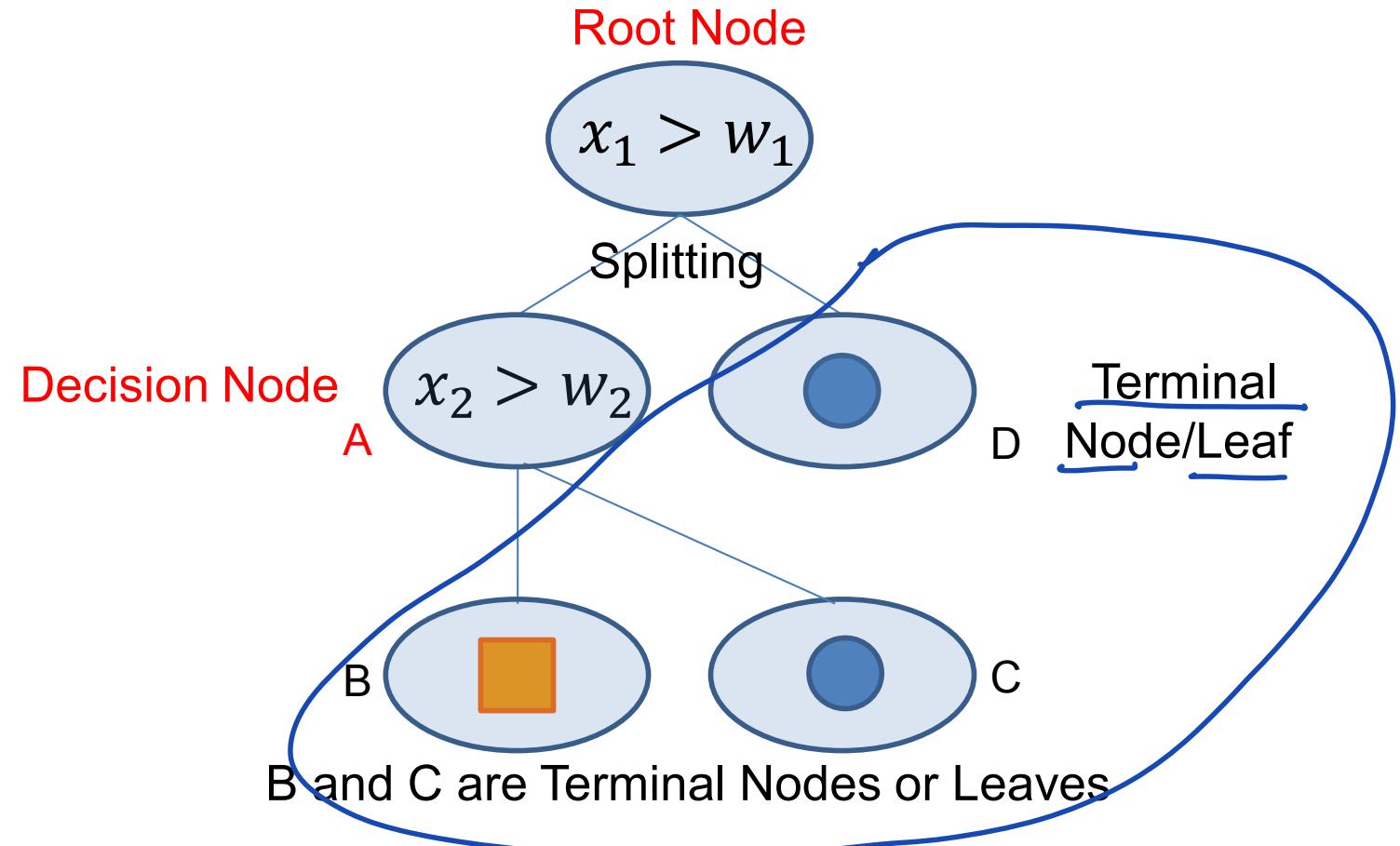
Basic Terminologies



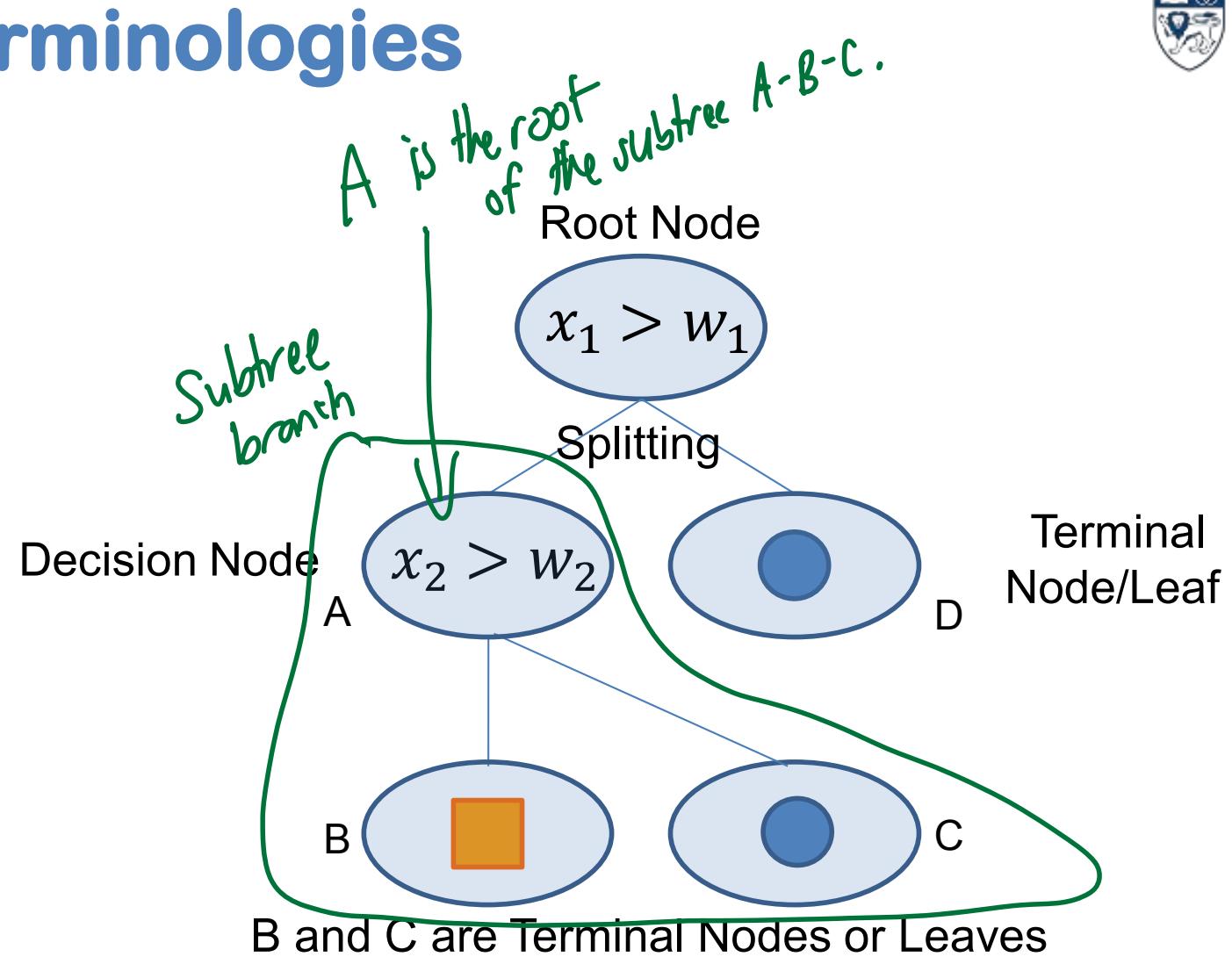
Basic Terminologies



Basic Terminologies

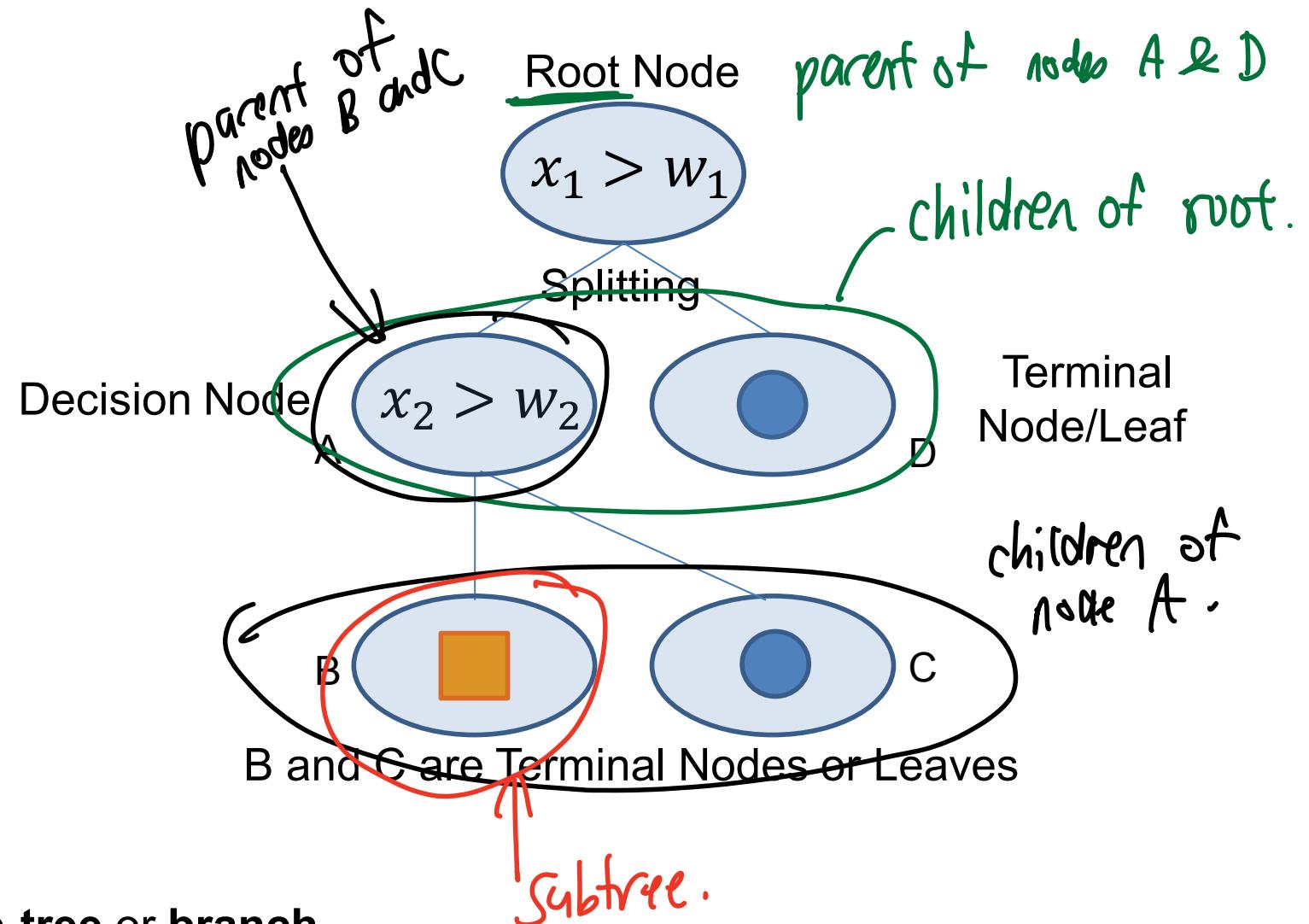


Basic Terminologies

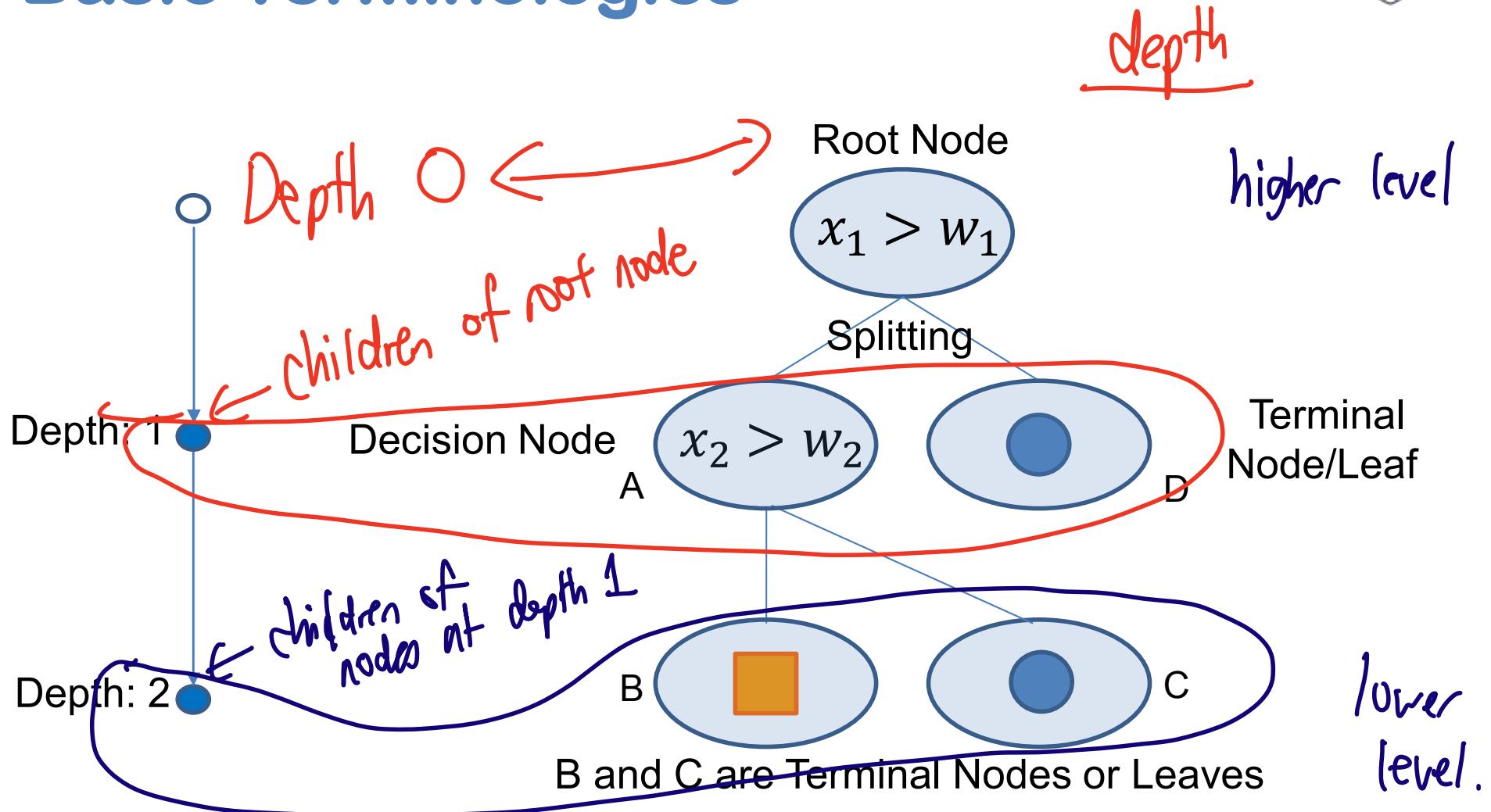


A-B-C forms a **sub-tree** or **branch**.

Basic Terminologies



Basic Terminologies



A-B-C forms a **sub-tree** or **branch**.

A is **parent node** of B and C; B and C are **children nodes** of A.

Questions?

Building a Classification Decision Tree

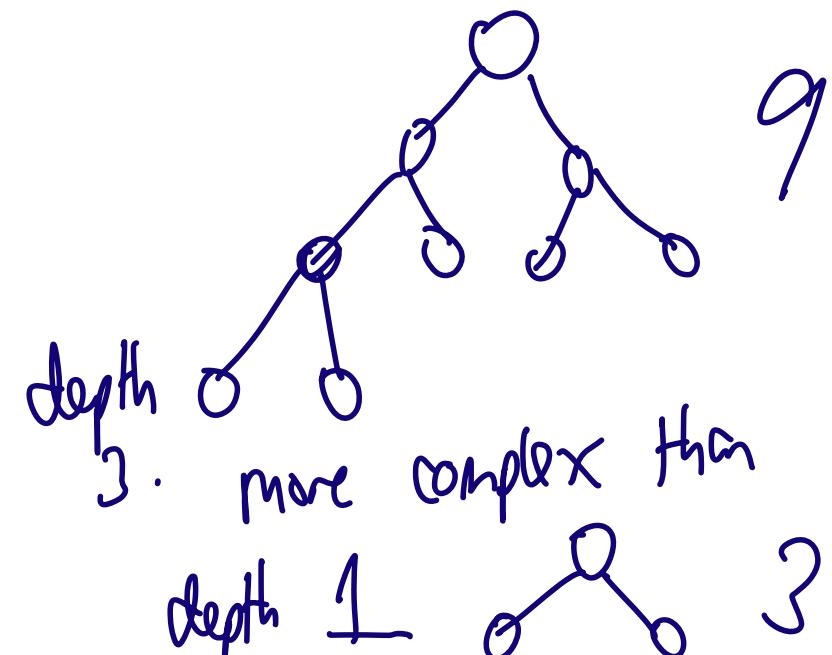
- Classification tree learning (or tree induction or tree growing) is the construction of a classification tree given training data

Building a Classification Decision Tree

- Classification tree learning (or tree induction or tree growing) is the construction of a classification tree given training data
- For a given training set, there can be many trees with 0 training error, so we prefer less “complex” trees

Building a Classification Decision Tree

- Classification tree learning (or tree induction or tree growing) is the construction of a classification tree given training data
- For a given training set, there can be many trees with 0 training error, so we prefer less “complex” trees
- Complexity** can be defined as **number of nodes** in the tree



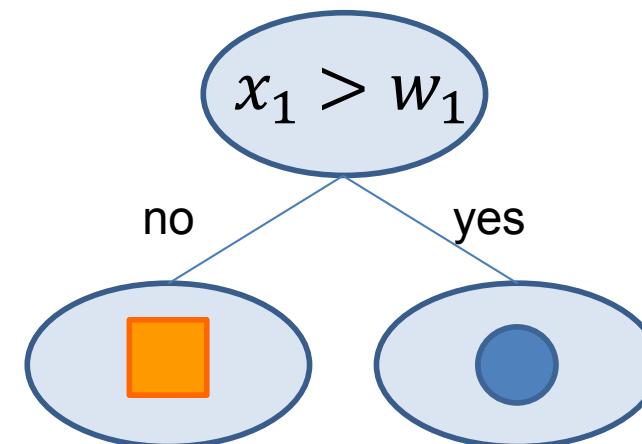
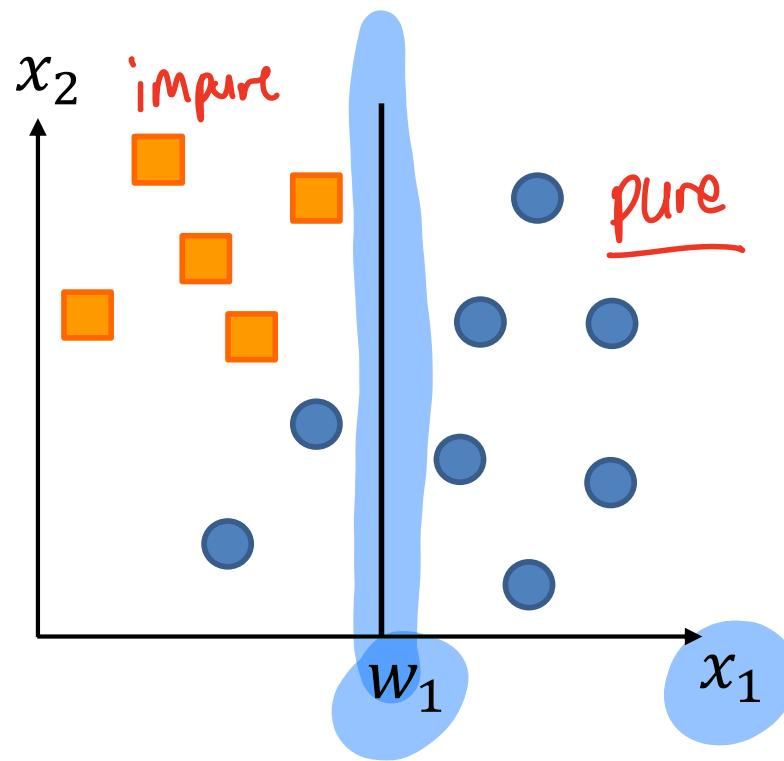
Building a Classification Decision Tree

- Classification tree learning (or tree induction or tree growing) is the construction of a classification tree given training data
- For a given training set, there can be many trees with 0 training error, so we prefer less “complex” trees
- Complexity can be defined as number of nodes in the tree
- Finding smallest tree is computationally hard, so we typically use some **greedy algorithm** not guaranteed to find the best tree

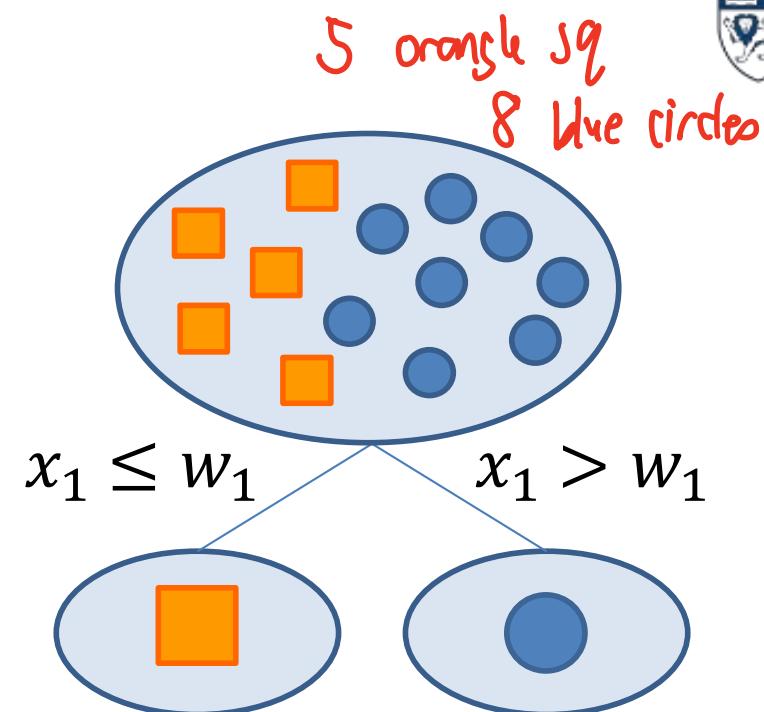
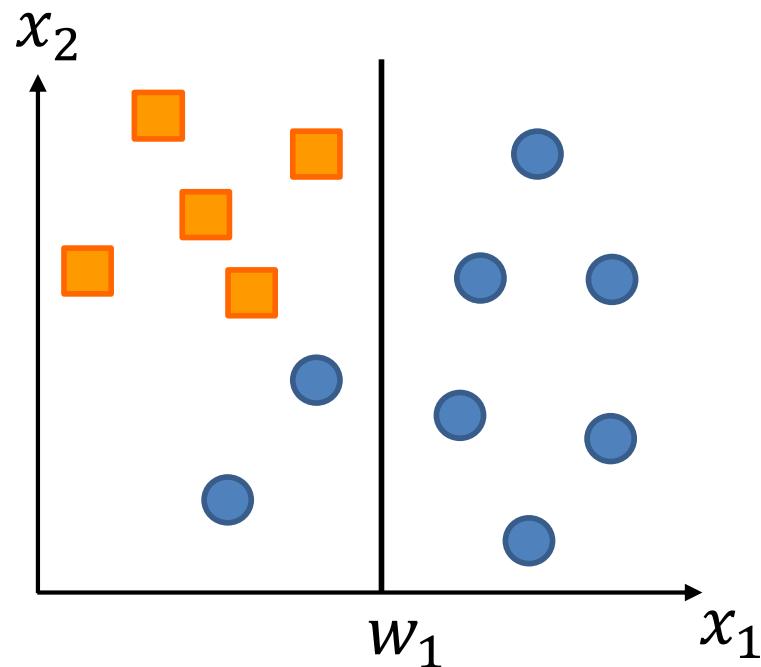
Building a Classification Decision Tree

- Classification tree learning (or tree induction or tree growing) is the construction of a classification tree given training data
- For a given training set, there can be many trees with 0 training error, so we prefer less “complex” trees
- Complexity can be defined as number of nodes in the tree
- Finding smallest tree is computationally hard, so we typically use some greedy algorithm not guaranteed to find the best tree
- We need to first define the concept of node impurity

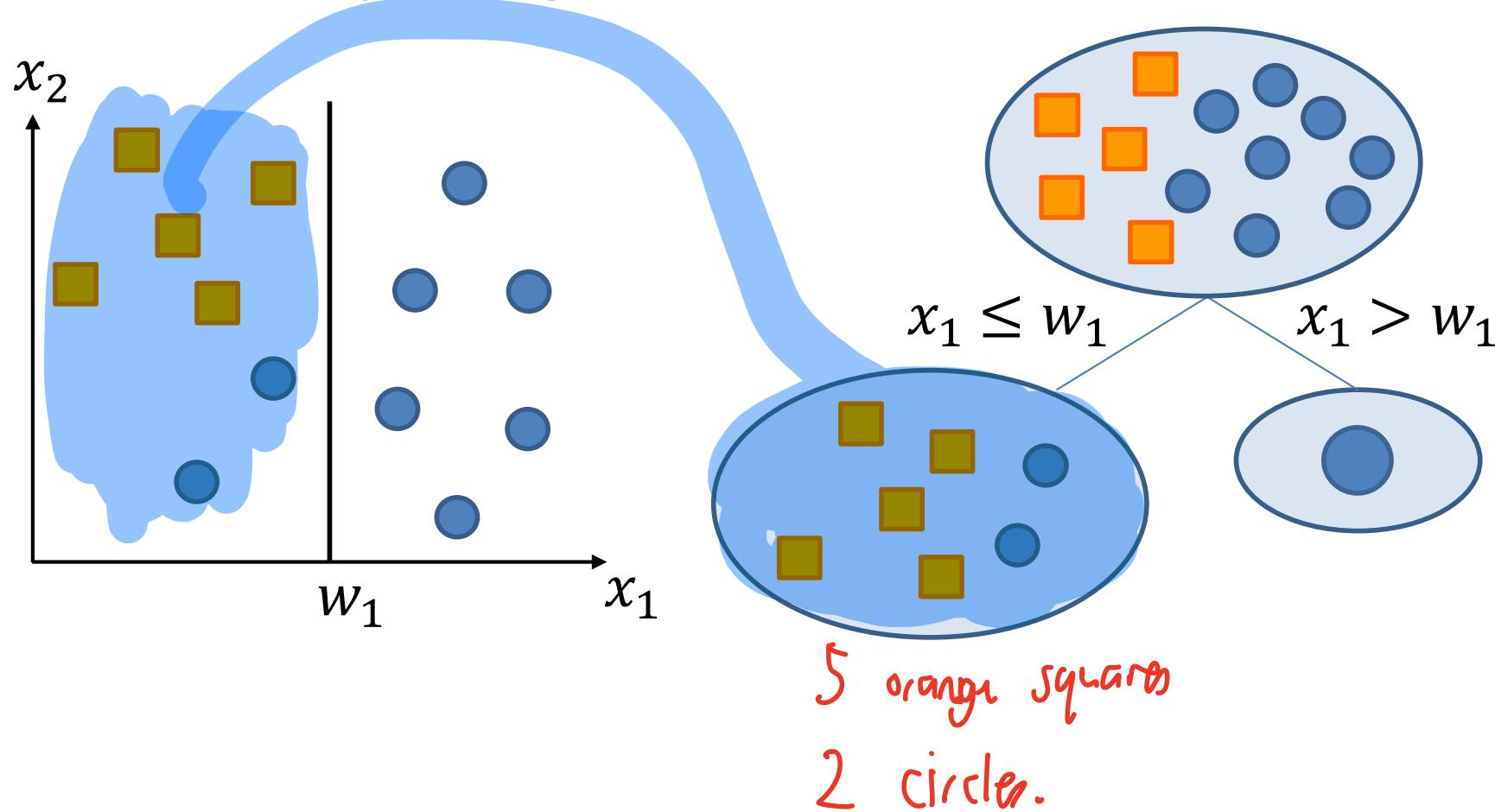
Node Impurity



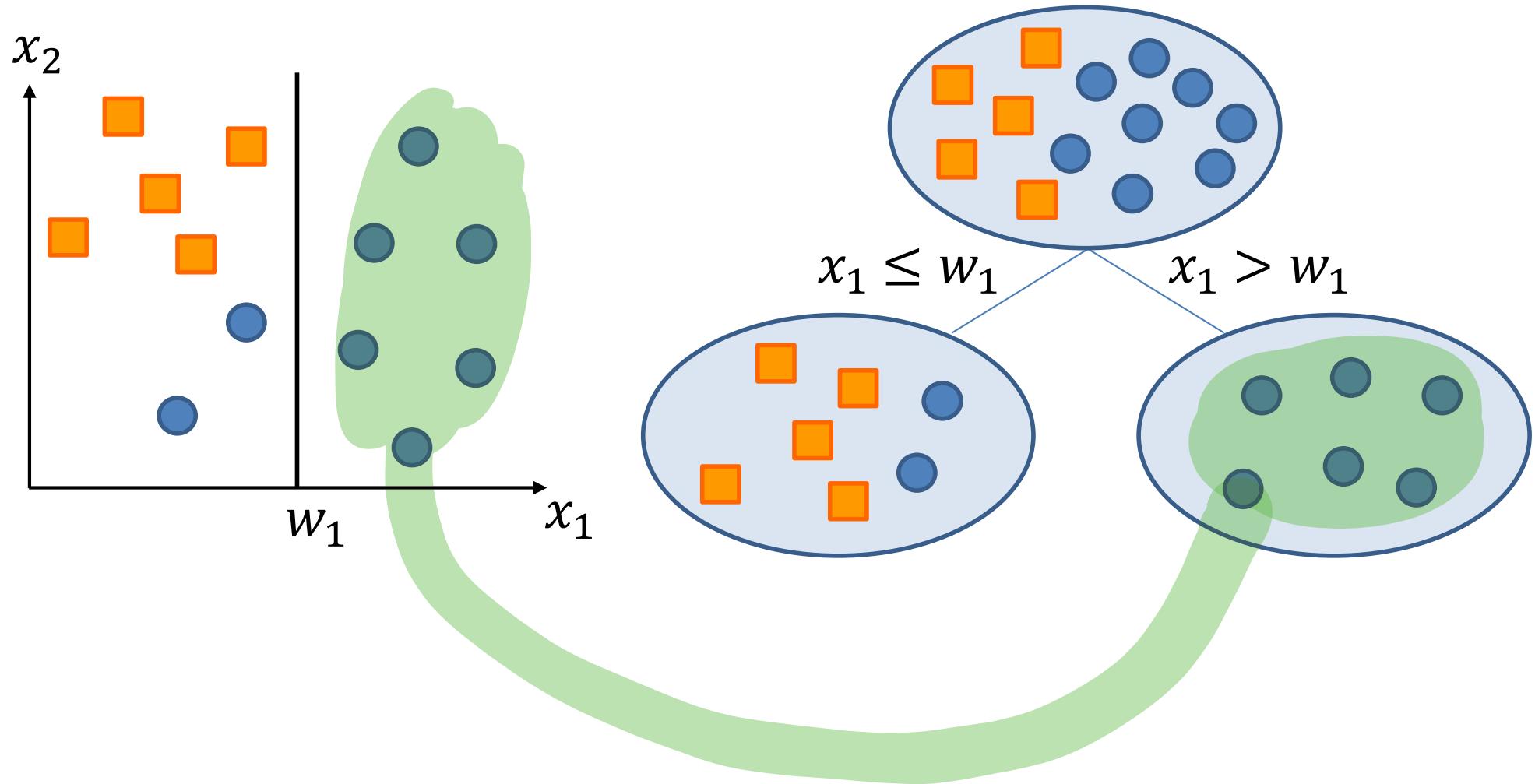
Node Impurity



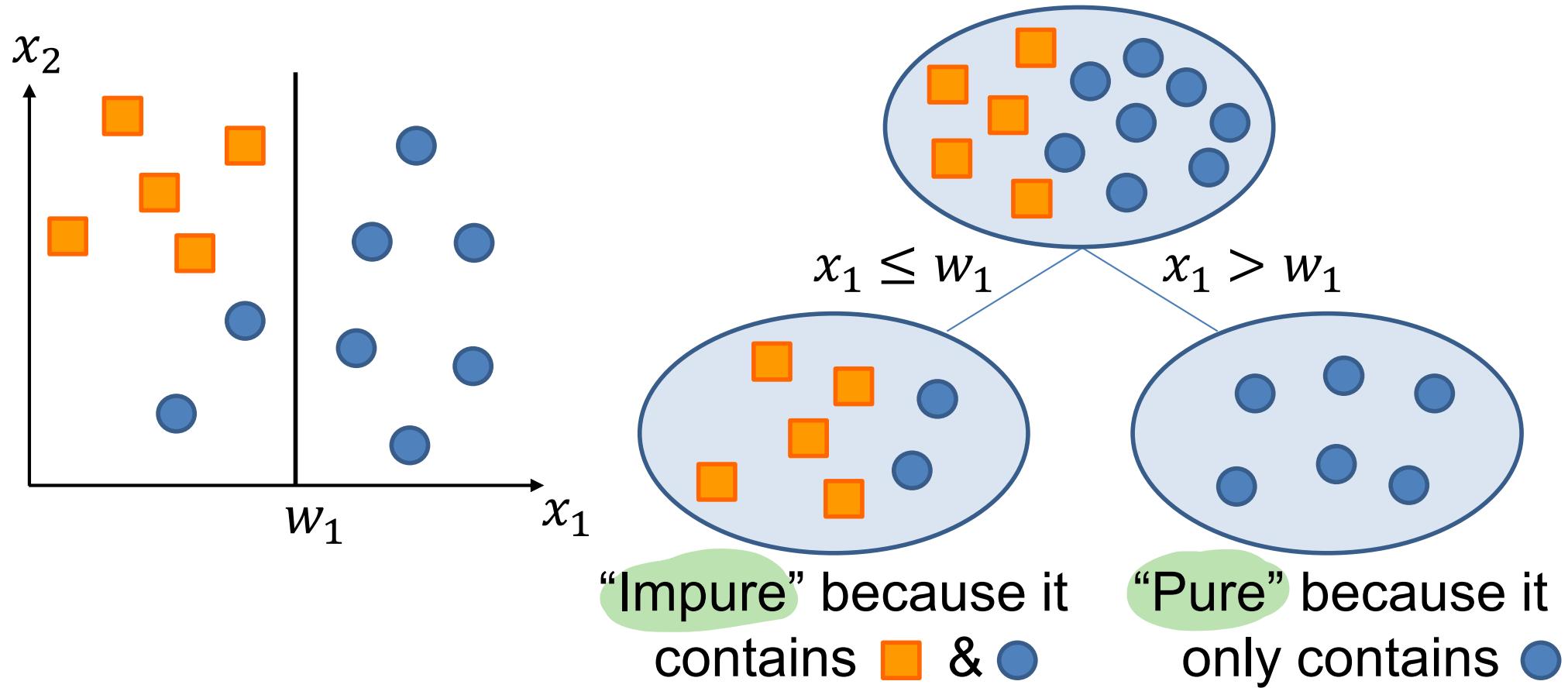
Node Impurity



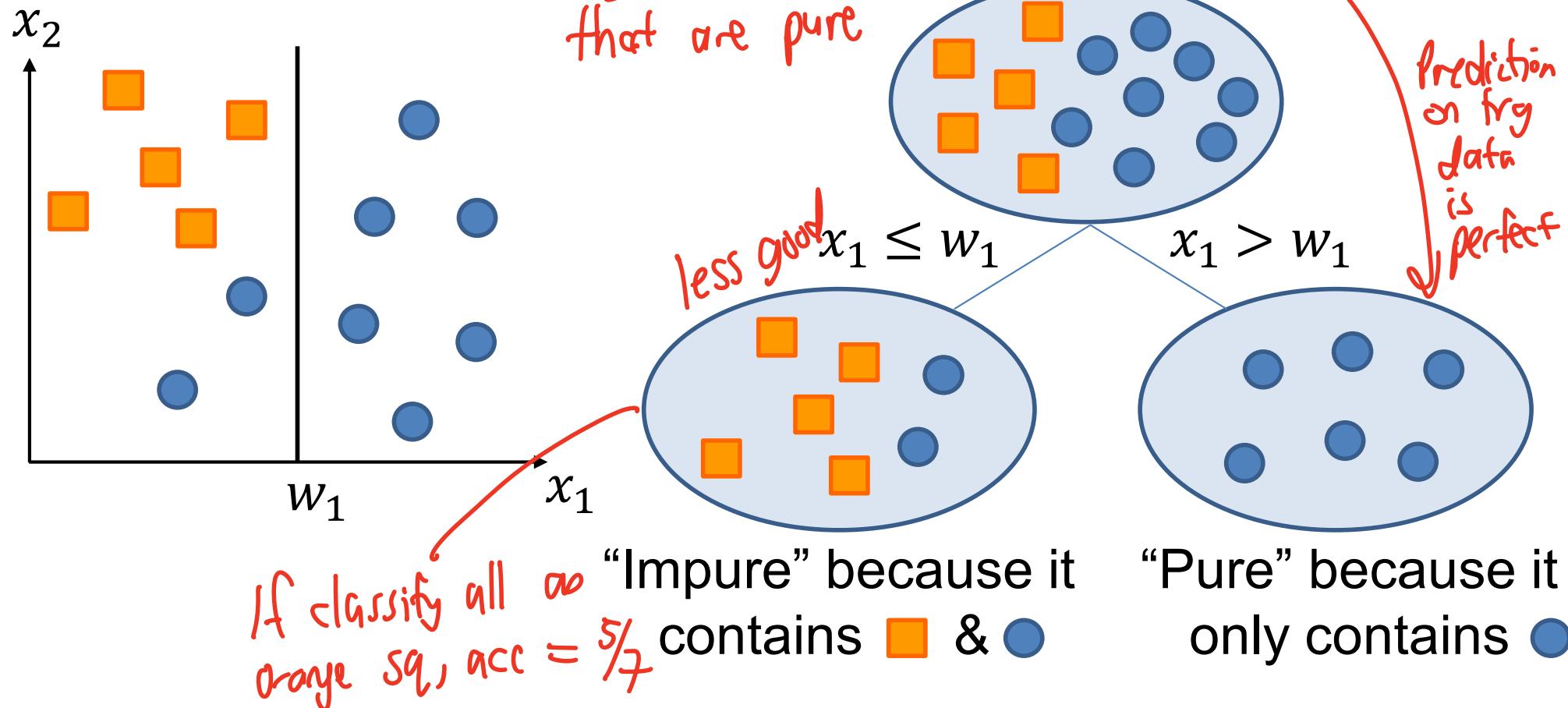
Node Impurity



Node Impurity



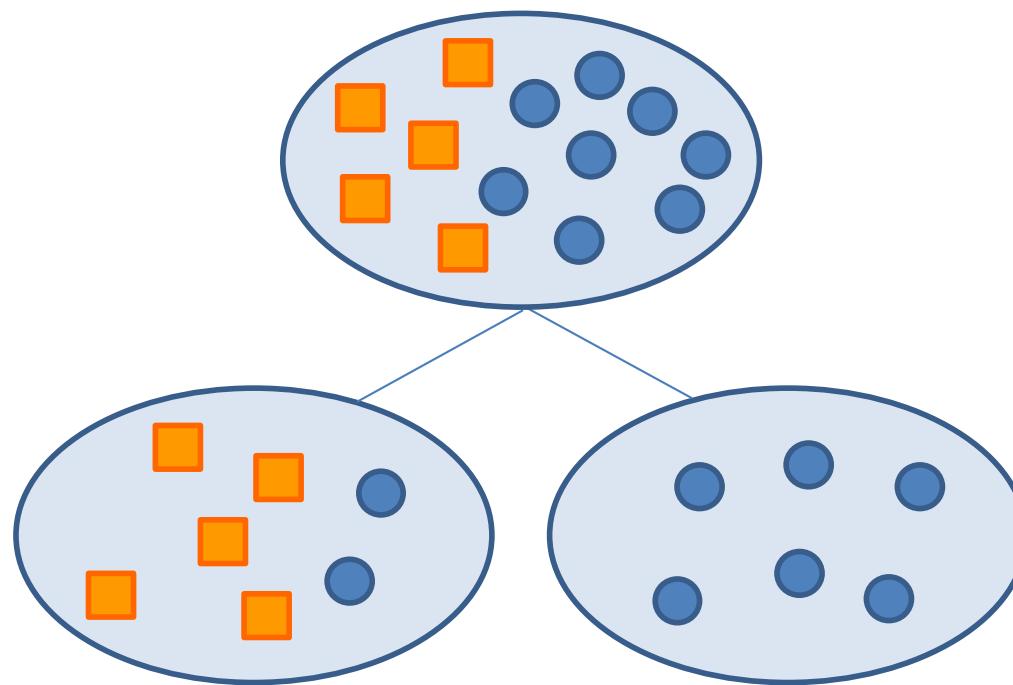
Node Impurity



- “Purity” is desirable because if a node contains only training data from one class, then prediction for training data in the node is perfect

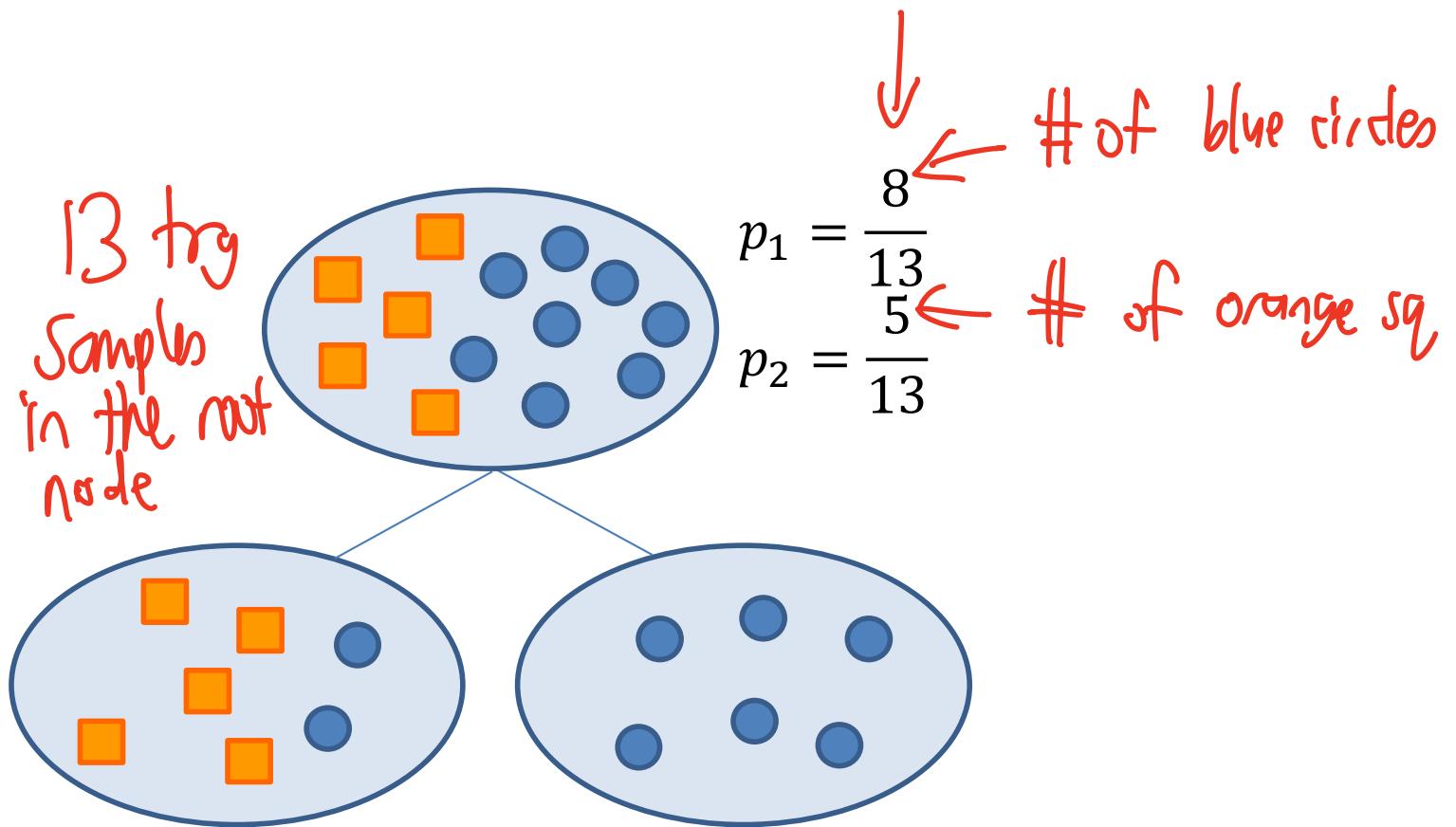
Node Impurity Measures

- Let  be class 1 &  be class 2



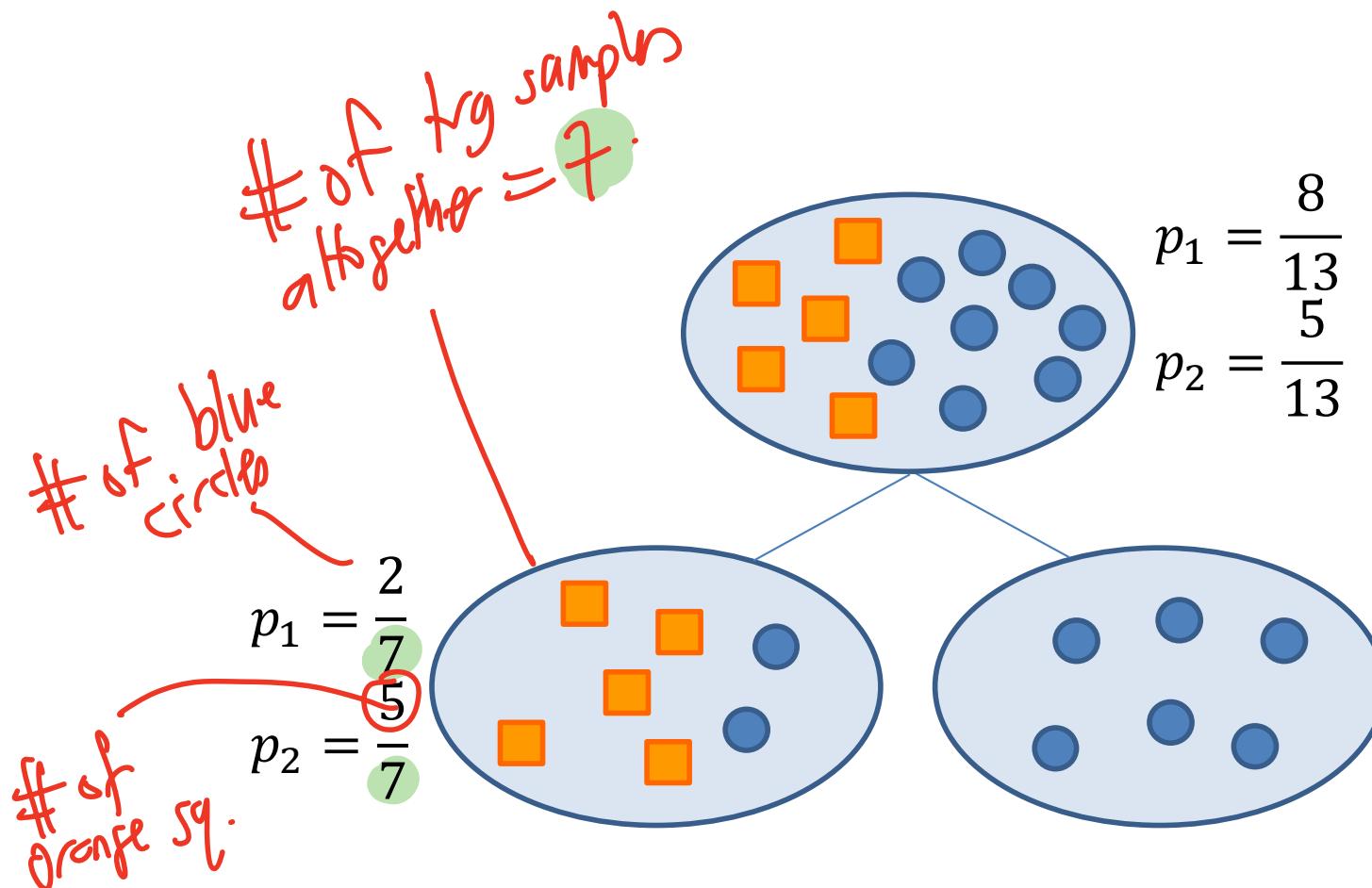
Node Impurity Measures

- Let ● be class 1 & ■ be class 2
- For particular node m , let p_i be the fraction (or probability) of data samples in node m belonging to class i



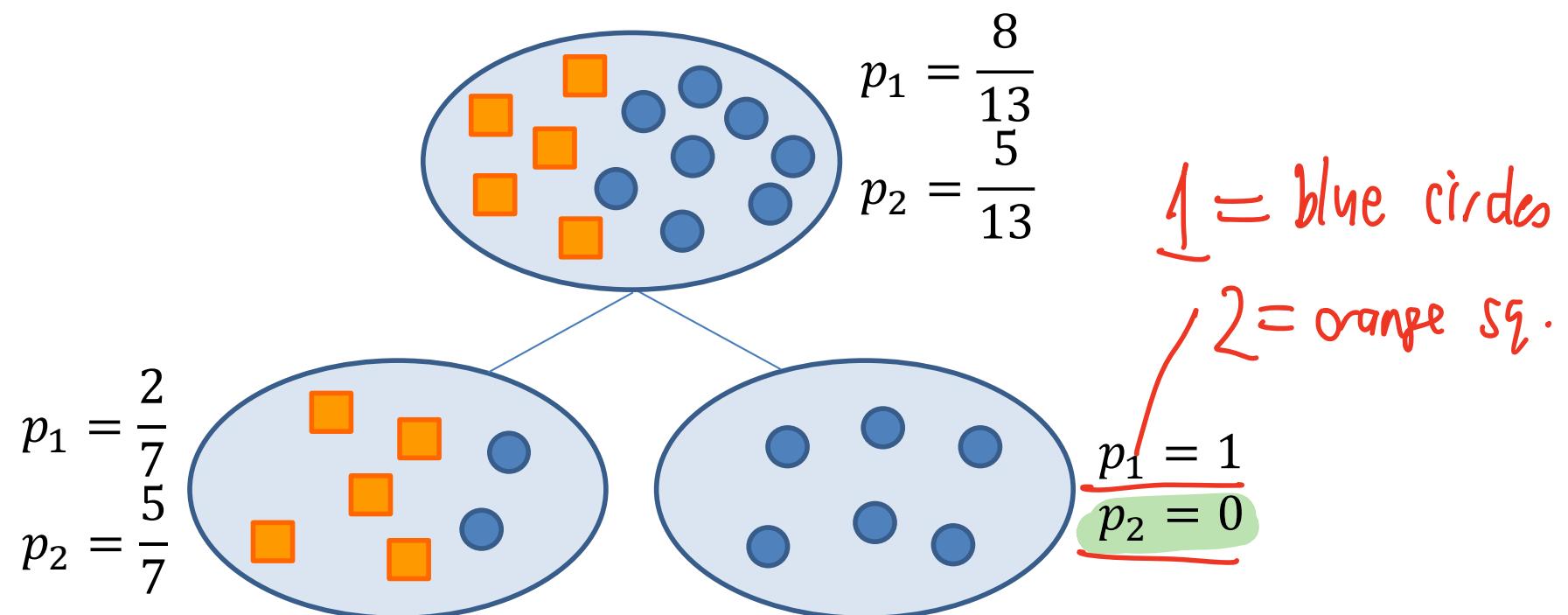
Node Impurity Measures

- Let ● be class 1 & ■ be class 2
- For particular node m , let p_i be the fraction (or probability) of data samples in node m belonging to class i



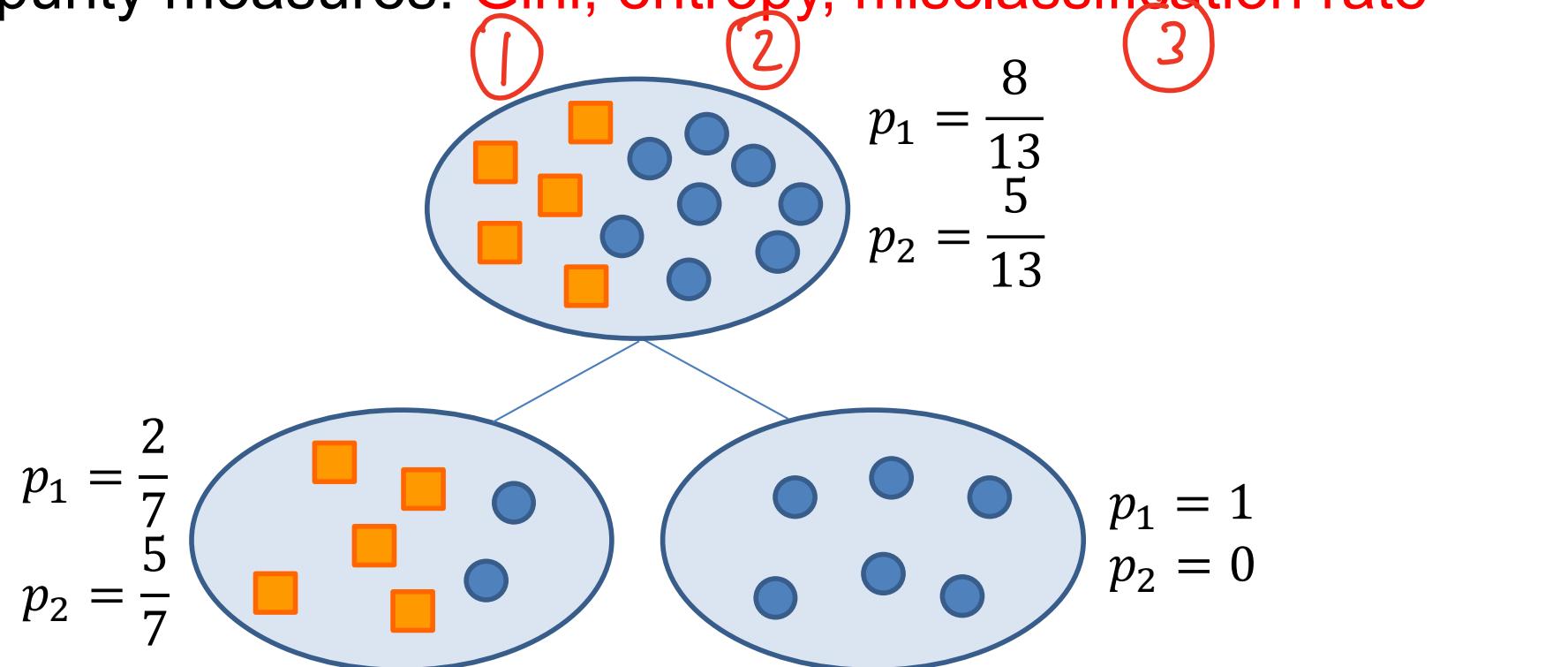
Node Impurity Measures

- Let ● be class 1 & ■ be class 2
- For particular node m , let p_i be the fraction (or probability) of data samples in node m belonging to class i

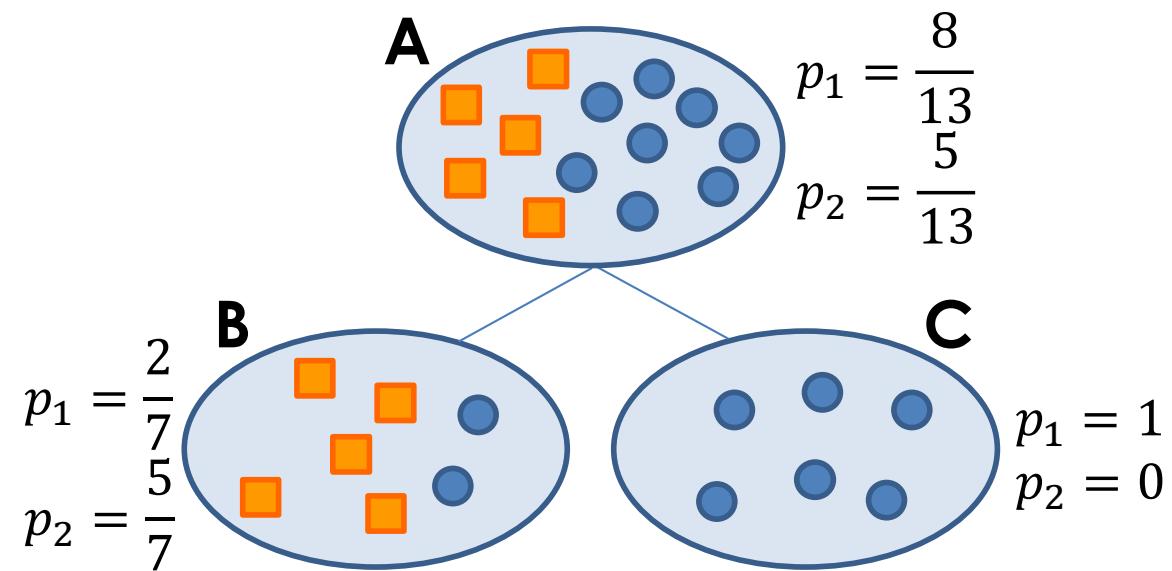


Node Impurity Measures

- Let ● be class 1 & ■ be class 2
- For particular node m , let p_i be the fraction (or probability) of data samples in node m belonging to class i
- Let Q_m be impurity of node m ← These impurities depend on p_i 's.
- 3 impurity measures: Gini, entropy, misclassification rate

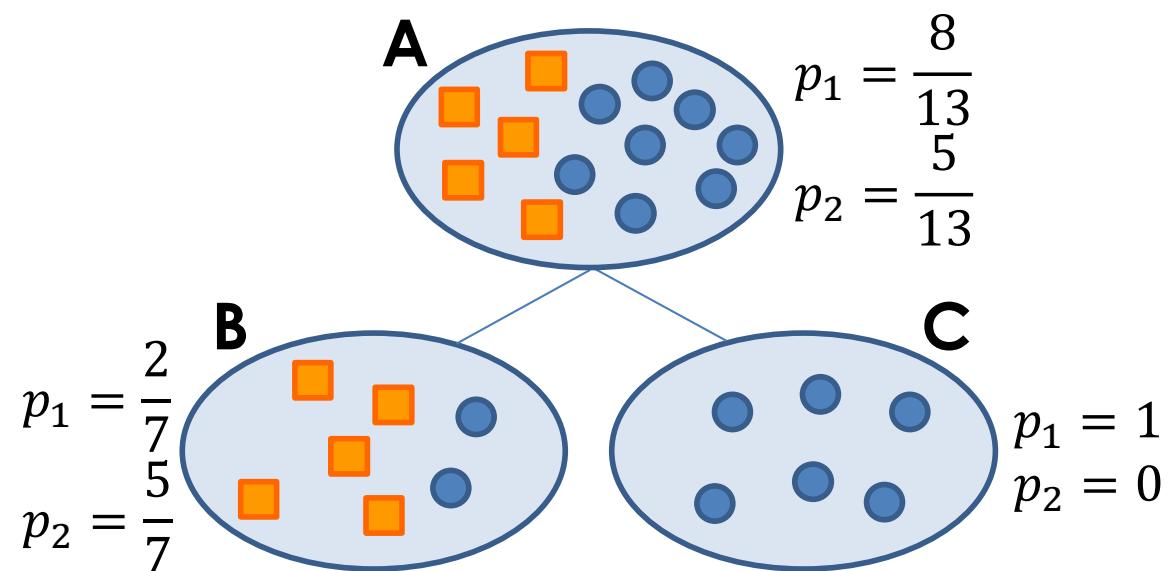


Gini Impurity



Gini Impurity $\Leftarrow 1 - \sum_{i=1}^K p_i^2$

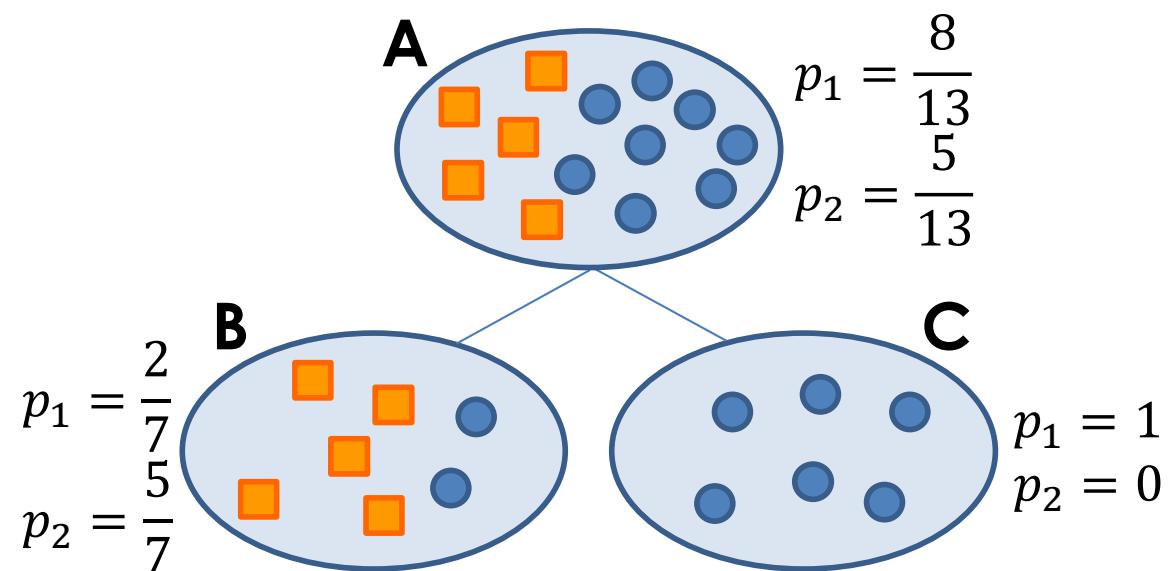
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2$



Gini Impurity

$$(p_1^2 + p_2^2)$$

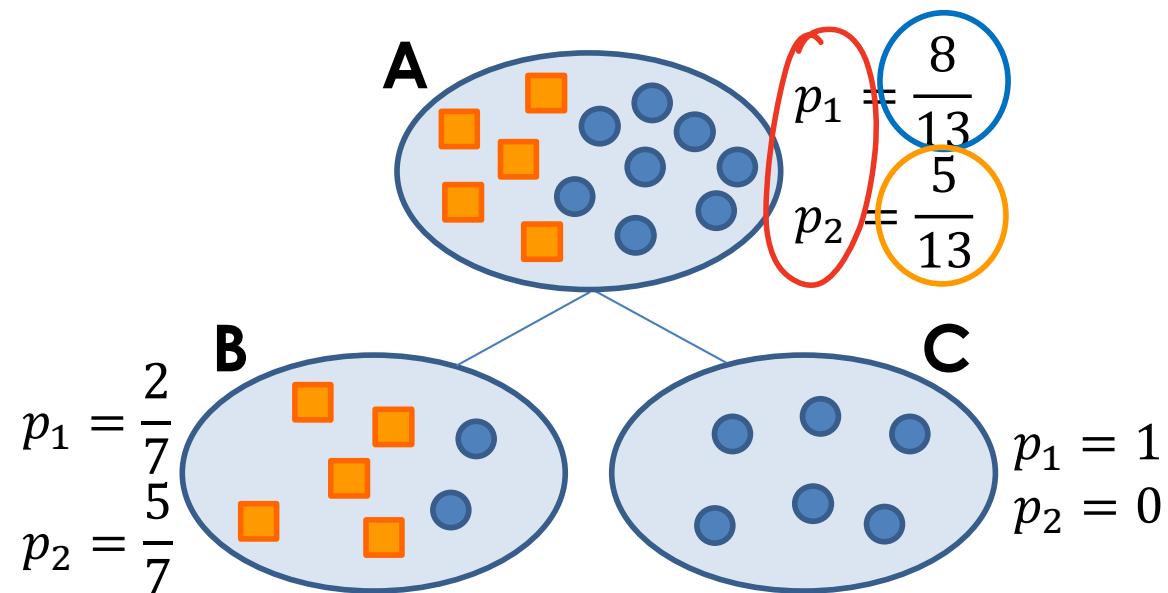
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = \underline{1 - p_1^2 - p_2^2}$



Gini Impurity

$$Q_m = 1 - \sum_{i=1}^K p_i^2$$

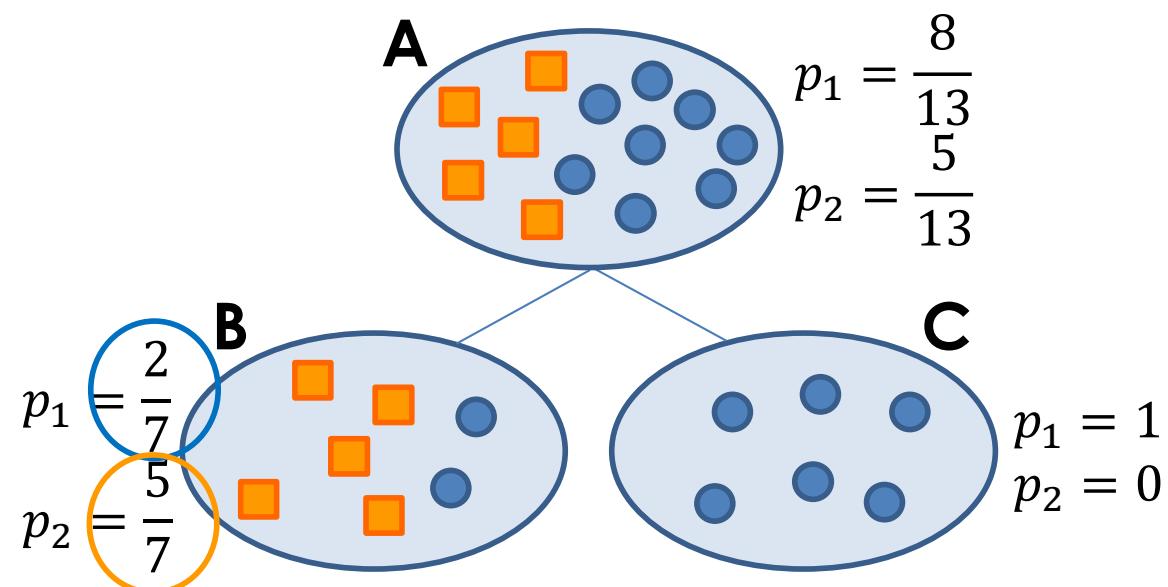
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$



Gini Impurity

- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$
- Node B: $Q_B = 1 - (2/7)^2 - (5/7)^2 = \underline{0.4082}$

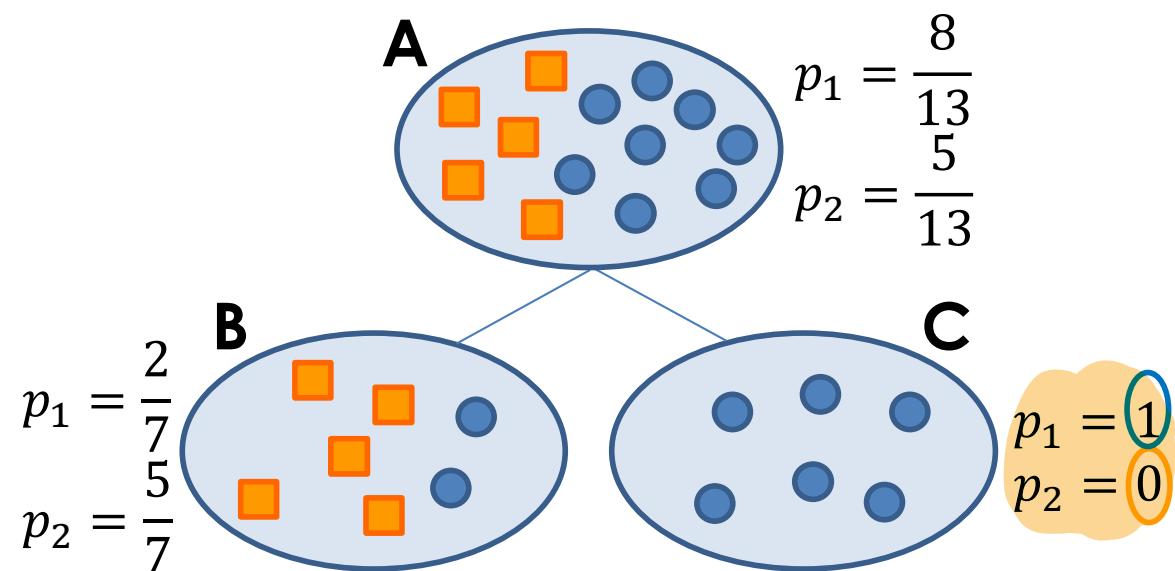
$$p_1 \quad p_2$$



Gini Impurity

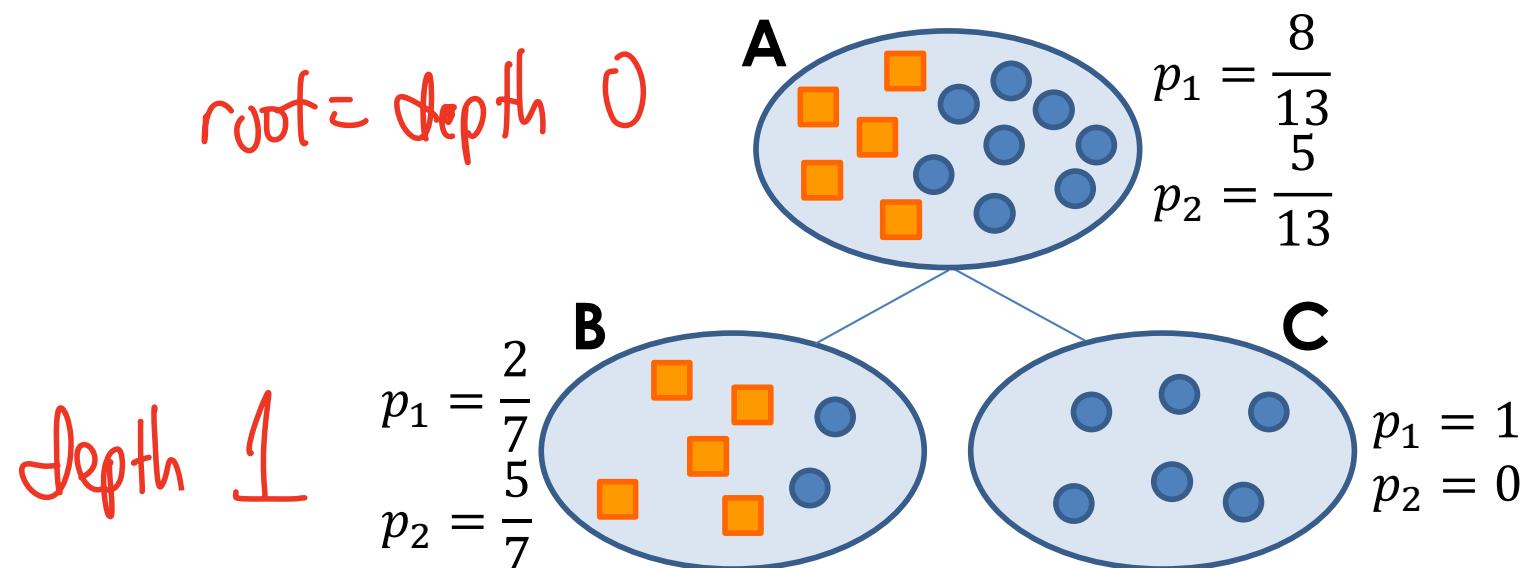
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$
- Node B: $Q_B = 1 - (2/7)^2 - (5/7)^2 = 0.4082$
- Node C: $Q_C = 1 - 1^2 - 0^2 = 0$

$$1 - 1^2 - 0^2 = 0$$



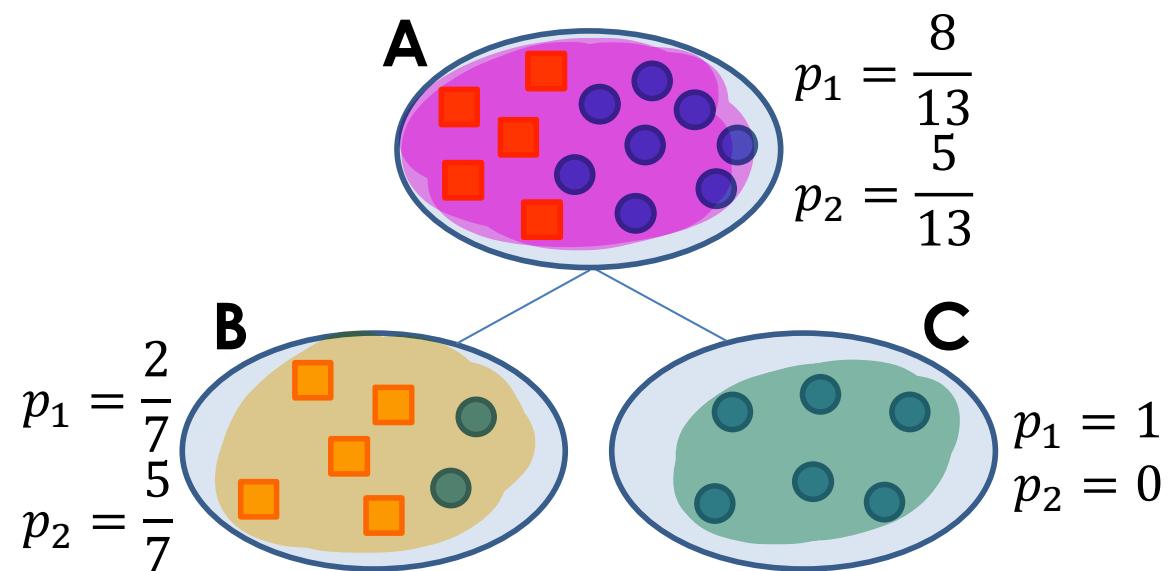
Gini Impurity

- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$
- Node B: $Q_B = 1 - (2/7)^2 - (5/7)^2 = 0.4082$
- Node C: $Q_C = 1 - 1^2 - 0^2 = 0$
- Overall Gini (depth 1) = fraction of data samples in node B $\times Q_B +$ fraction of data samples in node C $\times Q_C$**



Gini Impurity

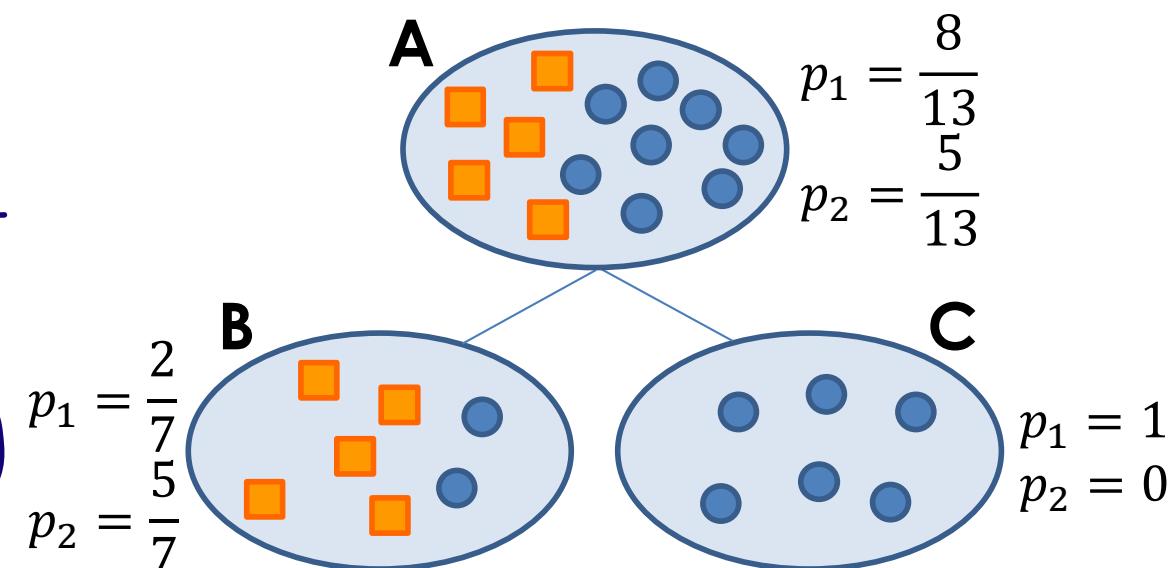
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$
- Node B: $Q_B = 1 - (2/7)^2 - (5/7)^2 = 0.4082$
- Node C: $Q_C = 1 - 1^2 - 0^2 = 0$
- Overall Gini (depth 1) = fraction of data samples in node B $\times Q_B +$
fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times 0.4082 + \left(\frac{6}{13}\right) \times 0 = 0.2198$



Gini Impurity

- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$
- Node B: $Q_B = 1 - (2/7)^2 - (5/7)^2 = 0.4082$
- Node C: $Q_C = 1 - 1^2 - 0^2 = 0$
- Overall Gini (depth 1) = fraction of data samples in node B $\times Q_B +$
fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times 0.4082 + \left(\frac{6}{13}\right) \times 0 = 0.2198$
- Observe lower impurity at depth 1 compared with root

Fact: The overall Gini at depth $m+1 \leq$ overall Gini at depth m (for all $m \geq 0$)

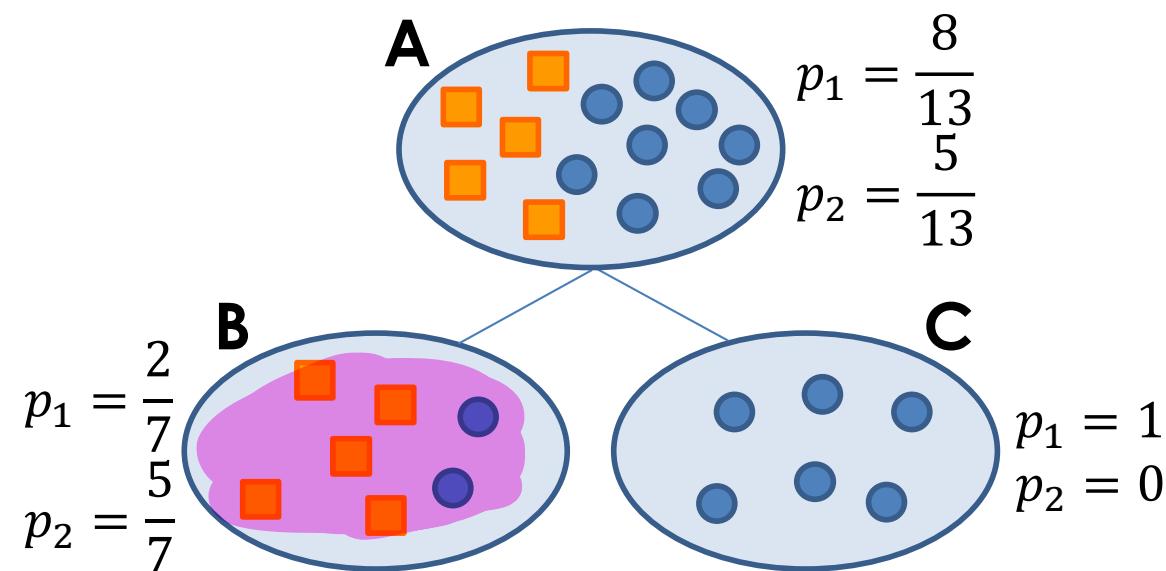


Gini Impurity

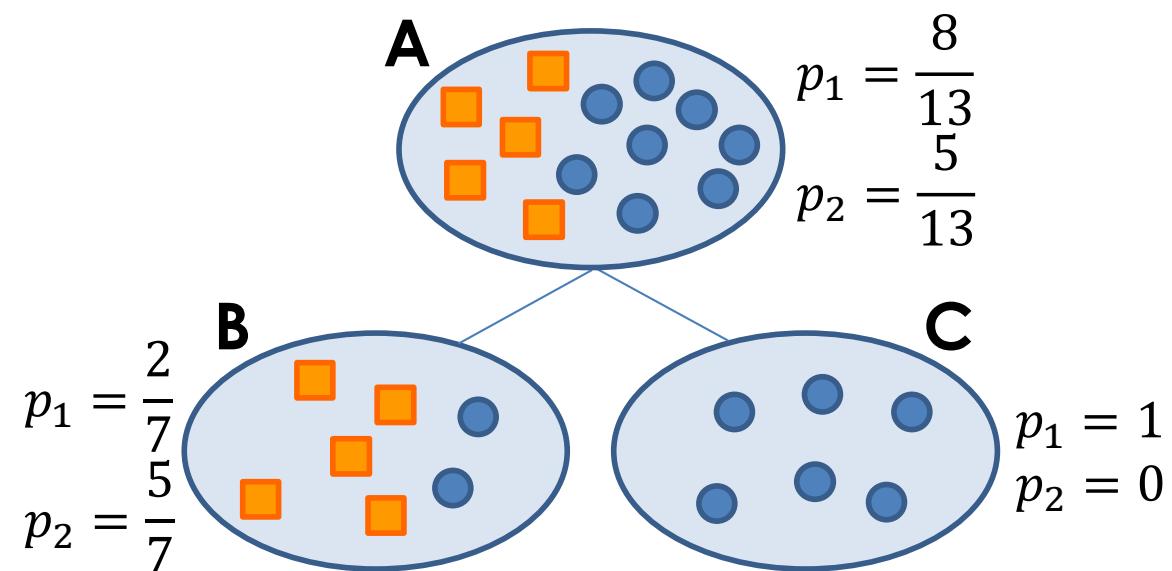
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \sum_{i=1}^K p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$
- Node B: $Q_B = 1 - (2/7)^2 - (5/7)^2 = 0.4082$
- Node C: $Q_C = 1 - 1^2 - 0^2 = 0$
- Overall Gini (depth 1) = fraction of data samples in node B $\times Q_B +$
fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times 0.4082 + \left(\frac{6}{13}\right) \times 0 = 0.2198$
- Observe lower impurity at depth 1 compared with root
- Same Gini formula for more than 2 classes:

$$Q_m = 1 - \sum_{i=1}^K p_i^2$$

$\frac{\# \text{ of samples in } B}{B} = \frac{\# \text{ of samples in } B}{\# \text{ of samples in } B + \# \text{ of samples in } C}$



Entropy



Entropy $\underline{= 2}$

- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i$

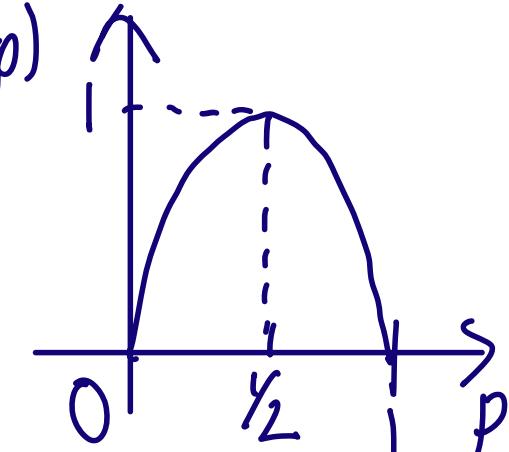
$$\sum_{i=1}^K p_i = 1$$

$$-\sum_{i=1}^K p_i \log_2 p_i$$

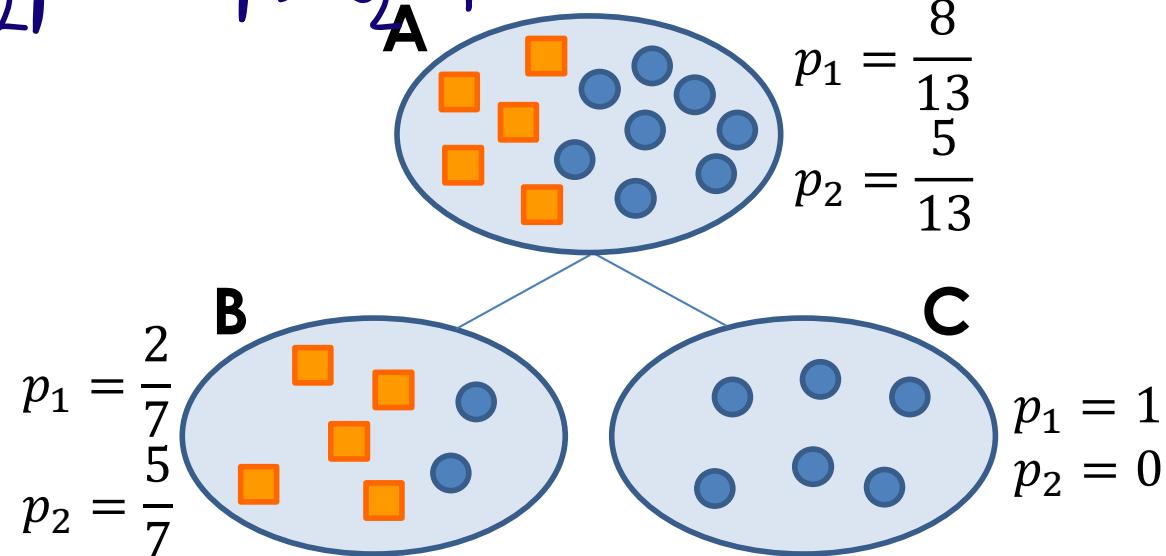
$$p_1 = p$$

$$p_2 = 1-p$$

bit $Q_m(p)$



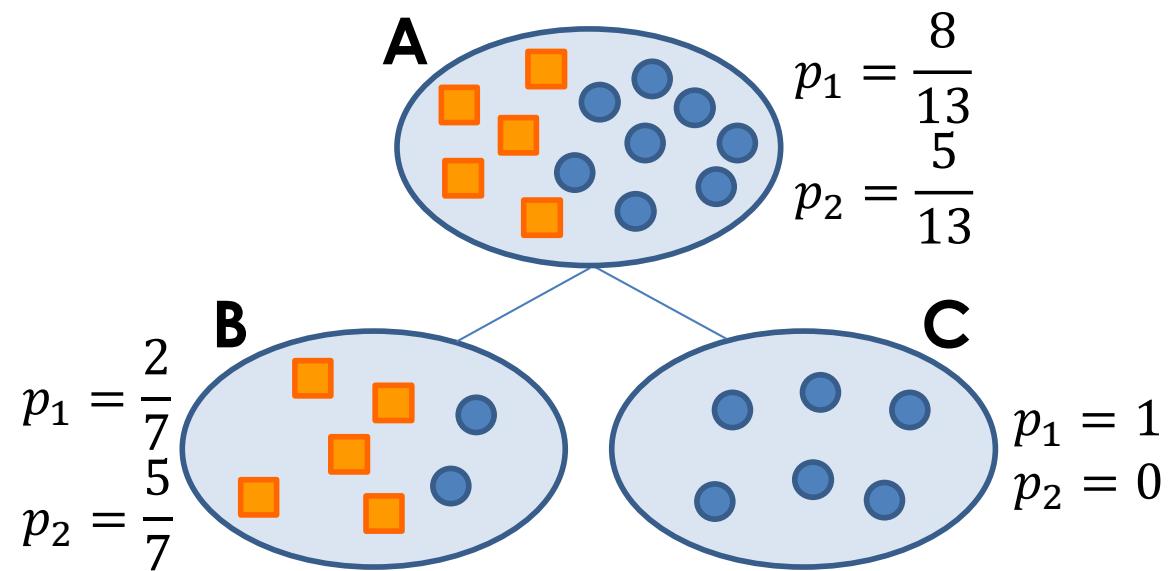
$$Q_m(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$



Entropy

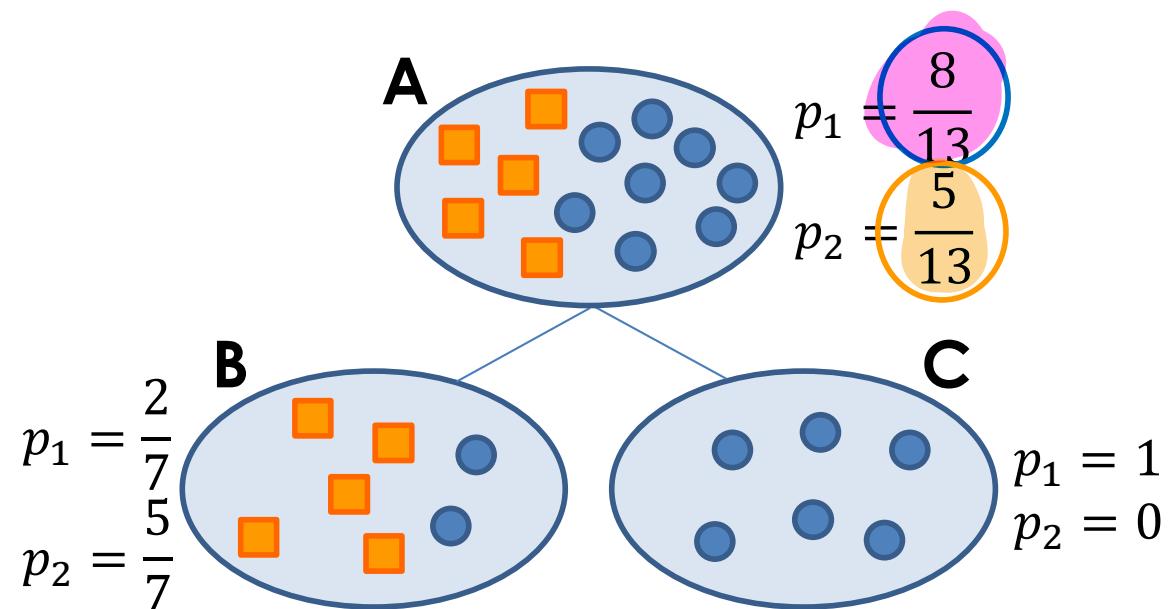
$$P_C = 1 - P_1$$

- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - (1-p_1) \log_2 (1-p_1)$



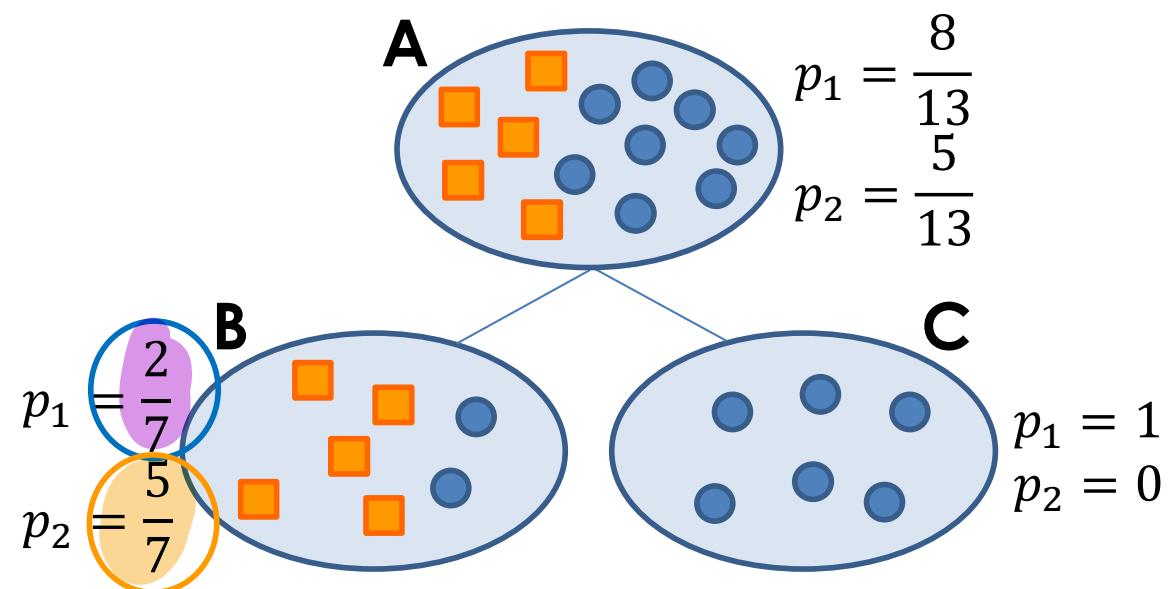
Entropy

- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_A = -(8/13) \log_2(8/13) - (5/13) \log_2(5/13) = 0.9612$



Entropy

- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_A = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$
- Node B: $Q_B = -(2/7)\log_2(2/7) - (5/7)\log_2(5/7) = 0.8631$



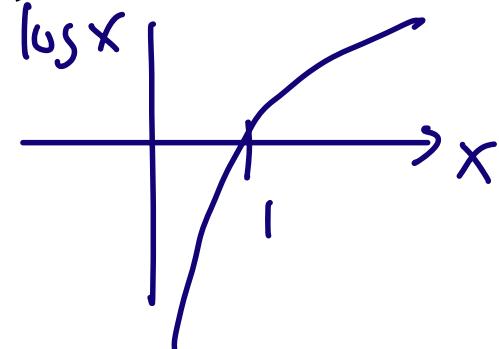
Entropy

$$\lim_{\varepsilon \downarrow 0} \varepsilon \log \varepsilon = 0.$$

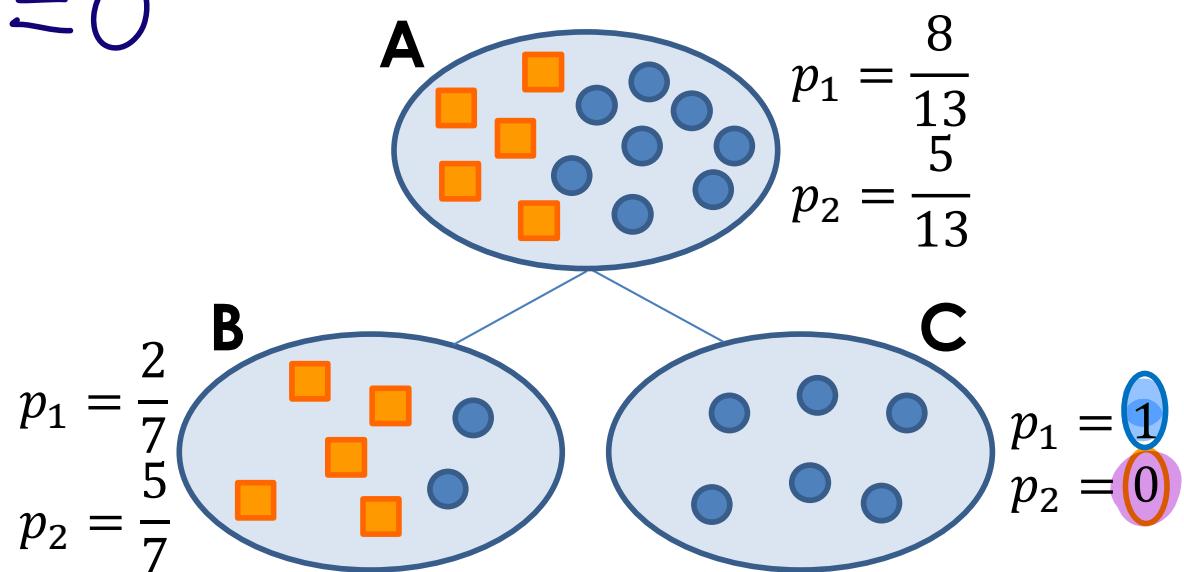
L'Hospital's rule.

- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_A = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$
- Node B: $Q_B = -(2/7)\log_2(2/7) - (5/7)\log_2(5/7) = 0.8631$
- Node C: $Q_C = -1 \log_2 1 - 0 \log_2 0 = 0$

$$0 \log_2 1 + 0 \log_2 0 = 0$$



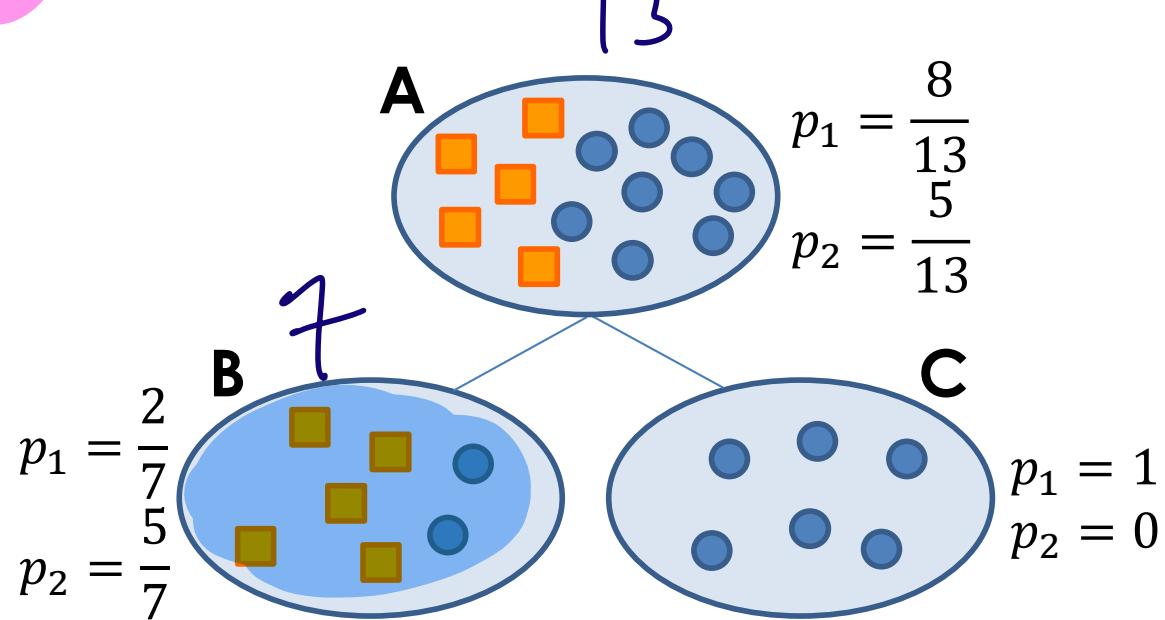
Convention: $0 \log 0 = 0$



Entropy

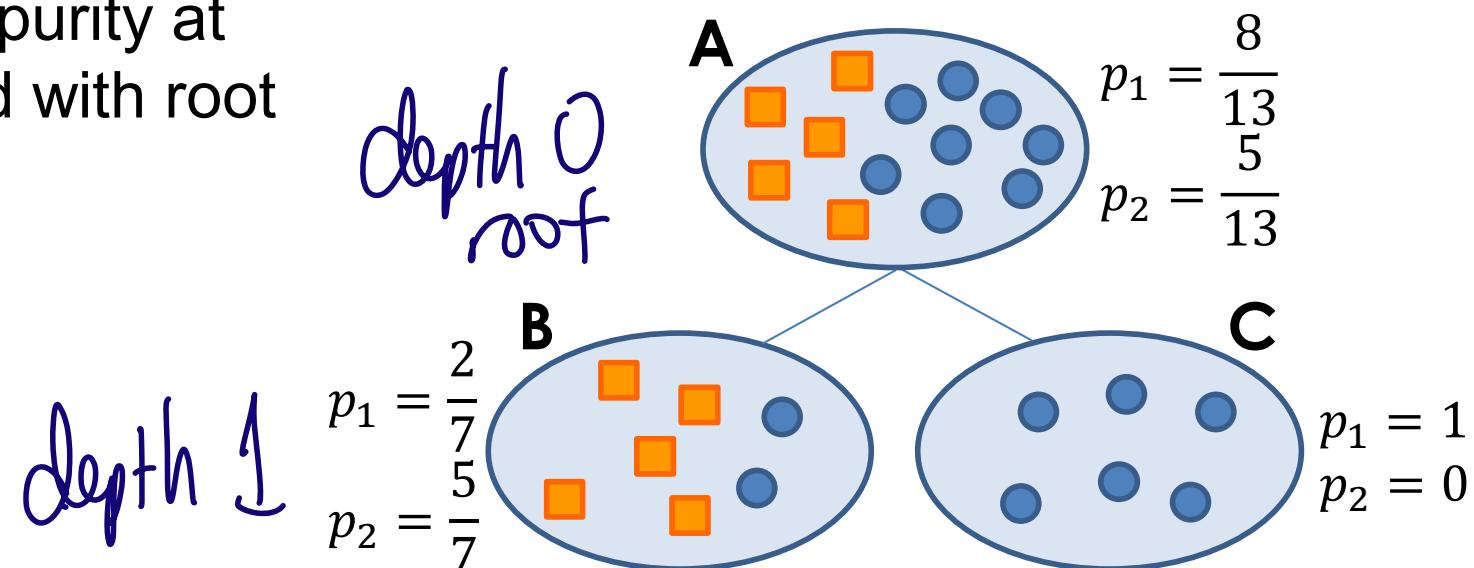
- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_A = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$
- Node B: $Q_B = -(2/7)\log_2(2/7) - (5/7)\log_2(5/7) = 0.8631$
- Node C: $Q_C = -1 \log_2 1 - 0 \log_2 0 = 0$
- Overall entropy (depth 1)** = fraction of data samples in node B $\times Q_B$
 + fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times 0.8631 + \left(\frac{6}{13}\right) \times 0 = 0.4648$

Check: Overall entropy does not increase!



Entropy

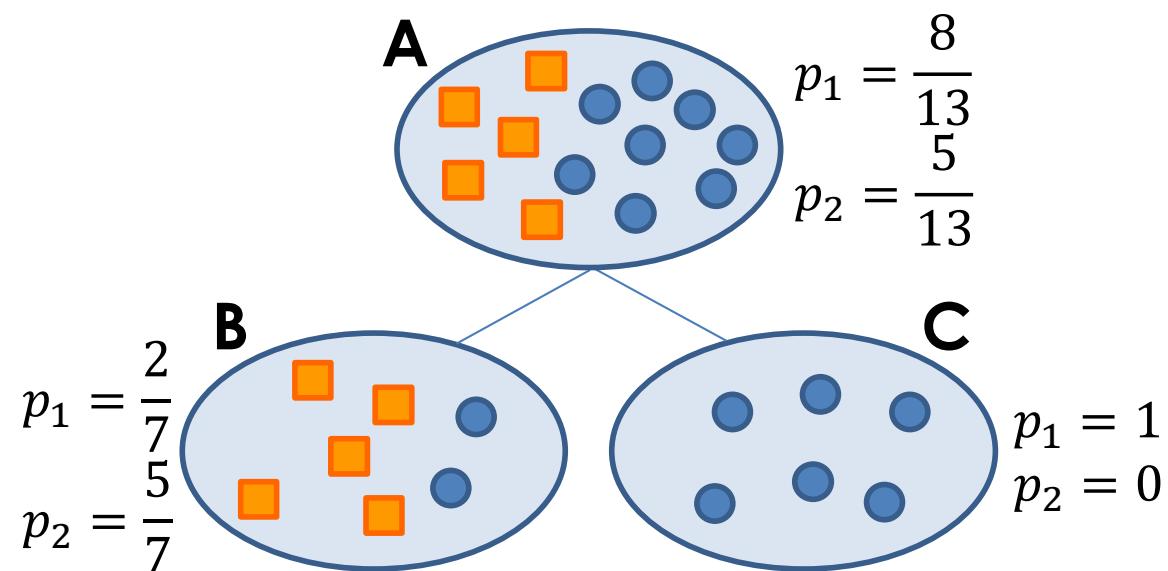
- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_A = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$
- Node B: $Q_B = -(2/7)\log_2(2/7) - (5/7)\log_2(5/7) = 0.8631$
- Node C: $Q_C = -1 \log_2 1 - 0 \log_2 0 = 0$
- Overall entropy (depth 1) = fraction of data samples in node B $\times Q_B$
 + fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times 0.8631 + \left(\frac{6}{13}\right) \times 0 = 0.4648$
- Observe lower impurity at depth 1 compared with root



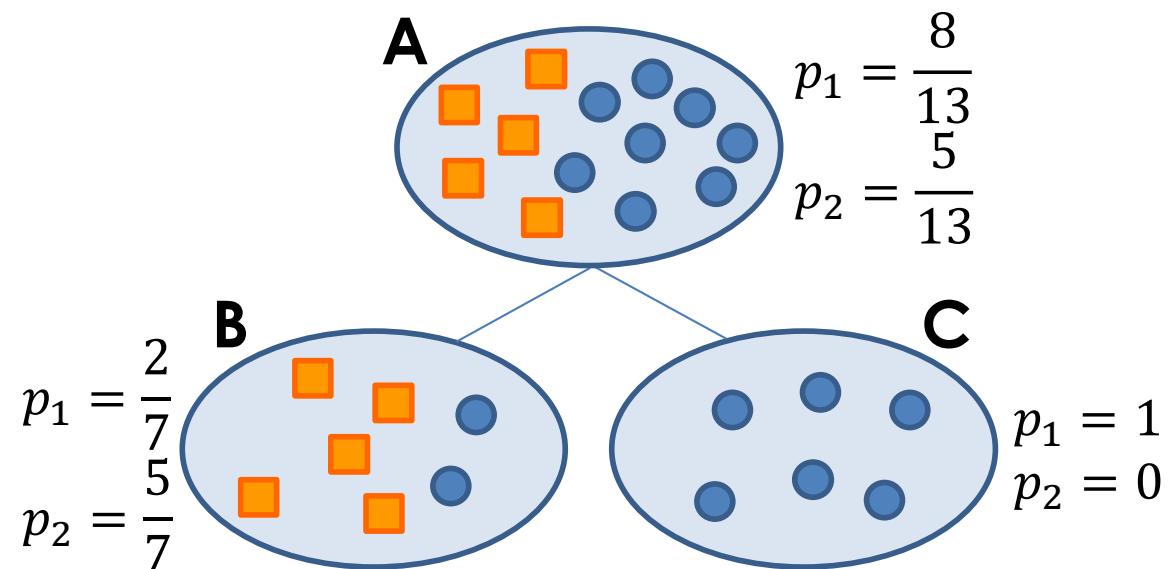
Entropy

- Let $K = \# \text{ classes}$, define $Q_m = -\sum_{i=1}^K p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_A = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$
- Node B: $Q_B = -(2/7)\log_2(2/7) - (5/7)\log_2(5/7) = 0.8631$
- Node C: $Q_C = -1 \log_2 1 - 0 \log_2 0 = 0$
- Overall entropy (depth 1) = fraction of data samples in node B $\times Q_B$
 + fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times 0.8631 + \left(\frac{6}{13}\right) \times 0 = 0.4648$
- Observe lower impurity at depth 1 compared with root
- Same entropy formula for more than 2 classes:

$$Q_m = -\sum_{i=1}^K p_i \log_2 p_i$$



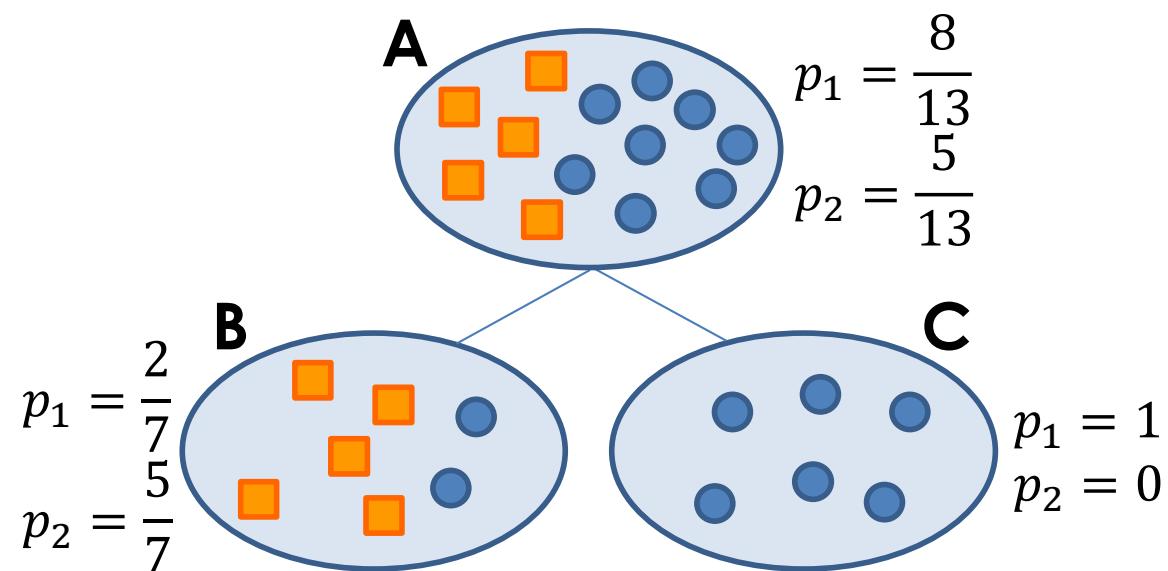
Misclassification rate



Misclassification rate

- Let $K = \# \text{ classes}$, define

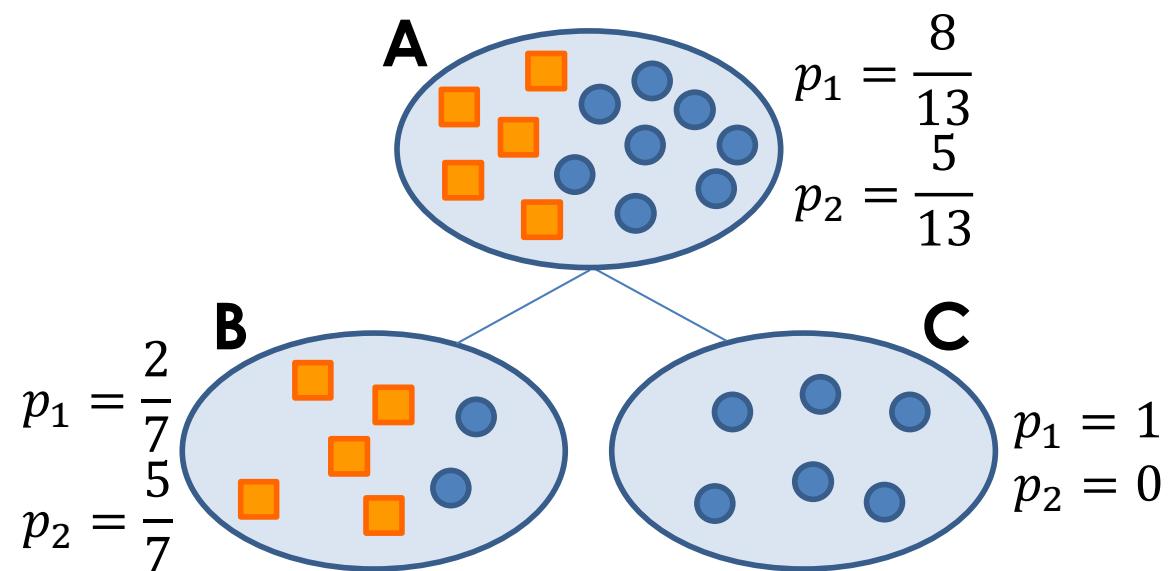
$$Q_m = 1 - \max_i p_i$$



Misclassification rate

$$\max_{i=1,2}$$

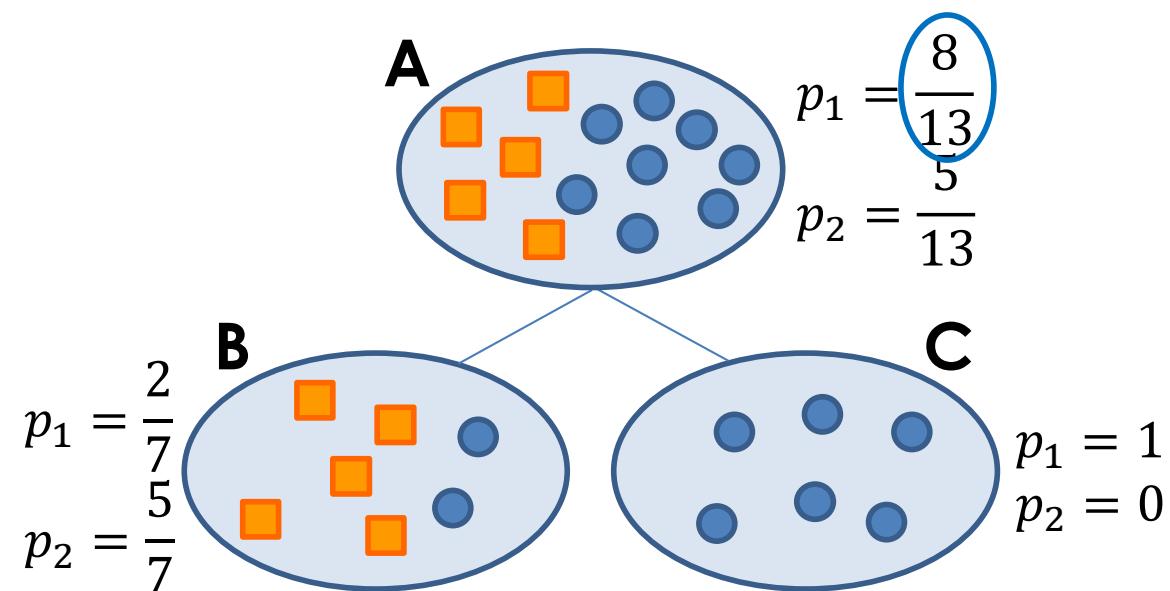
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \max_i p_i = \underline{1 - \max(p_1, p_2)}$



Misclassification rate

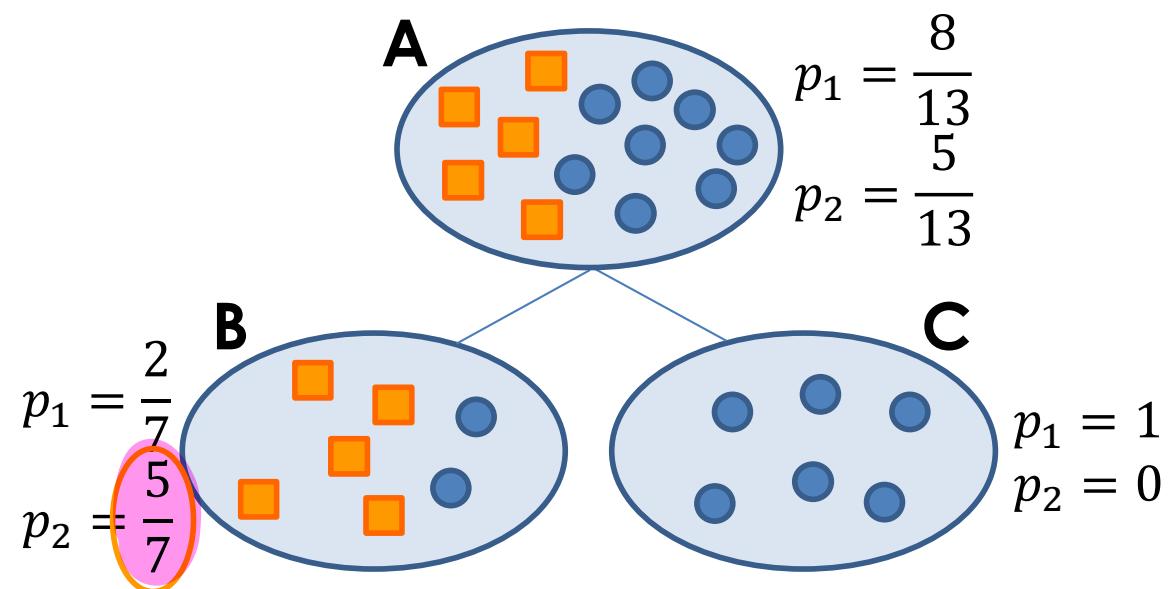
$$Q_m \geq 0$$

- Let $K = \# \text{ classes}$, define $Q_m = 1 - \max_i p_i = 1 - \max(p_1, p_2)$
- Node A: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_A = 1 - \frac{8/13}{p_1} = \frac{5/13}{p_1}$
 $\frac{5/13}{\frac{8}{13}} \parallel \frac{5}{8}$
 attains the max



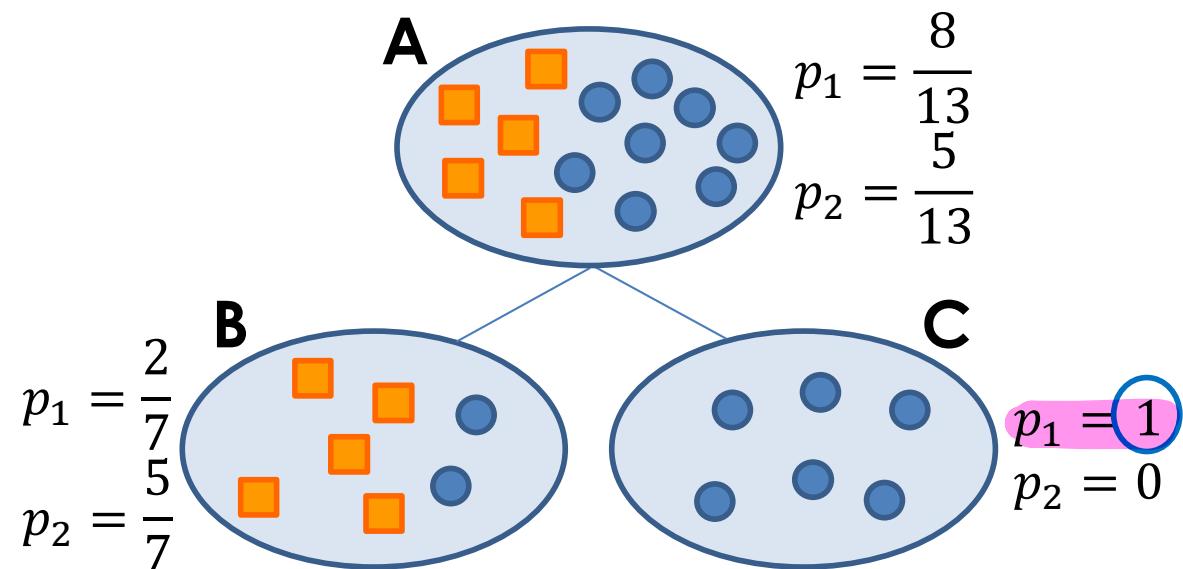
Misclassification rate

- Let $K = \# \text{ classes}$, define $Q_m = 1 - \max_i p_i = 1 - \max(p_1, p_2)$
- Node A: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_A = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = **class 2** $\Rightarrow Q_B = 1 - 5/7 = 2/7$



Misclassification rate

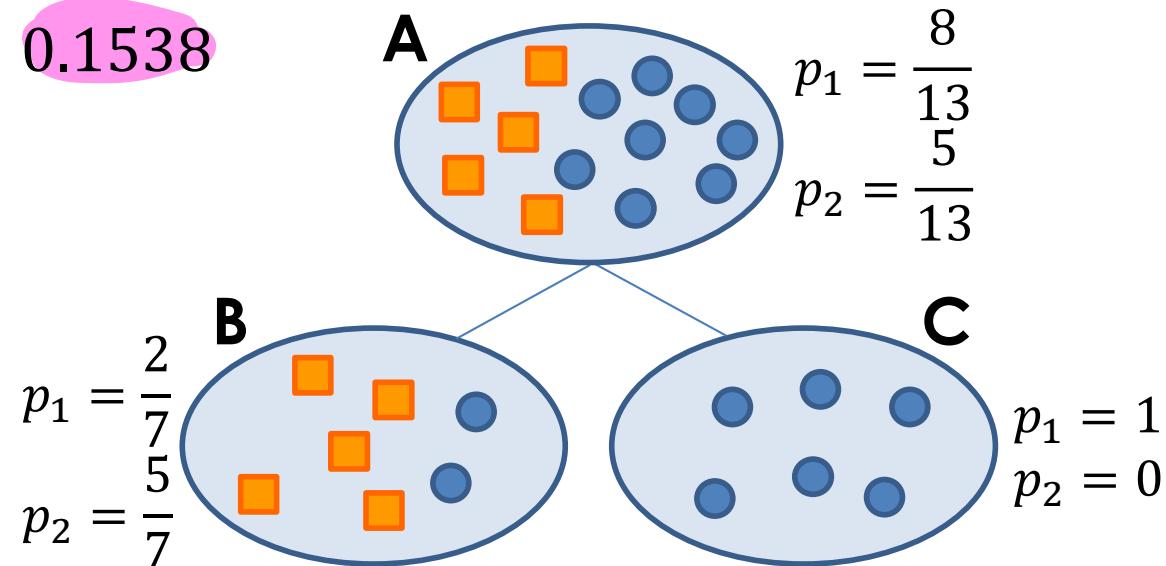
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \max_i p_i = 1 - \max(p_1, p_2)$
- Node A: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_A = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 $\Rightarrow Q_B = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = **class 1** $\Rightarrow Q_C = 1 - 1 = 0$



Misclassification rate

- Let $K = \# \text{ classes}$, define $Q_m = 1 - \max_i p_i = 1 - \max(p_1, p_2)$
- Node A: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_A = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 $\Rightarrow Q_B = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_C = 1 - 1 = 0$
- Overall misclassification rate (depth 1) = fraction of data samples in node B $\times Q_B +$ fraction of data samples in node C $\times Q_C$**

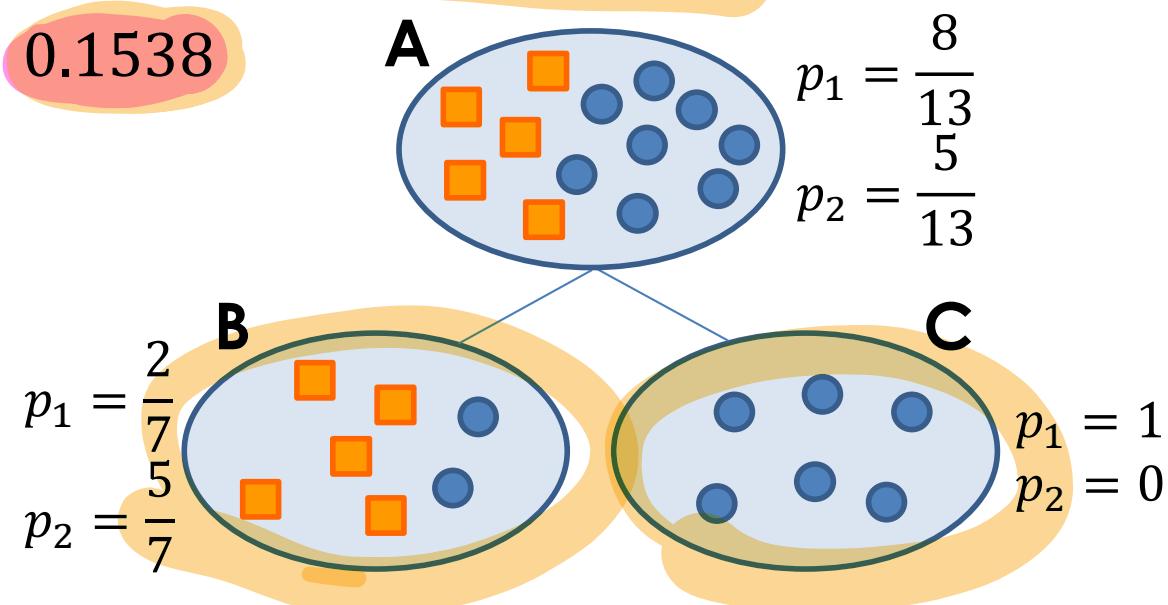
$$\cdot \left(\frac{7}{13} \right) \times \left(\frac{2}{7} \right) + \left(\frac{6}{13} \right) \times 0 = 0.1538$$



Misclassification rate

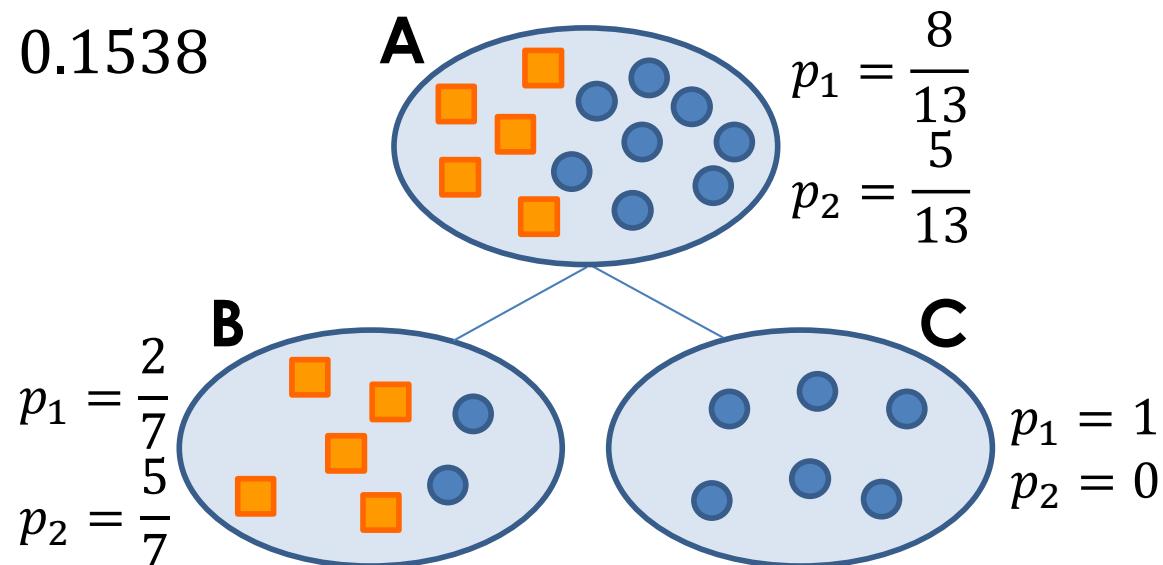
- Let $K = \# \text{ classes}$, define $Q_m = 1 - \max_i p_i = 1 - \max(p_1, p_2)$
- Node A: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_A = 1 - 8/13 = 5/13 \approx 0.38 \dots$
- Node B: $p_2 > p_1$, so best classification = class 2 $\Rightarrow Q_B = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_C = 1 - 1 = 0$
- Overall misclassification rate (depth 1) = fraction of data samples in node B $\times Q_B +$ fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times \left(\frac{2}{7}\right) + \left(\frac{6}{13}\right) \times 0 = 0.1538$
- Observe lower impurity at depth 1 compared with root

$$\frac{7}{13} \times \frac{2}{7} + \frac{6}{13} \times 0 < \frac{5}{13}.$$



Misclassification rate

- Let $K = \# \text{ classes}$, define $Q_m = 1 - \max_i p_i = 1 - \max(p_1, p_2)$
- Node A: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_A = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 $\Rightarrow Q_B = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = class 1 $\Rightarrow Q_C = 1 - 1 = 0$
- Overall misclassification rate (depth 1) = fraction of data samples in node B $\times Q_B +$ fraction of data samples in node C $\times Q_C$
 - $\left(\frac{7}{13}\right) \times \left(\frac{2}{7}\right) + \left(\frac{6}{13}\right) \times 0 = 0.1538$
- Observe lower impurity at depth 1 compared with root
- Same misclassification rate formula for more than 2 classes: $Q_m = 1 - \max_i p_i$



$$P_1 = \frac{2}{7}, P_2 = \frac{1}{7}, \frac{4}{7} = P_3.$$

$$Q_m = 1 - \max(P_1, P_2, P_3)$$

= 1 - \frac{4}{7} = \frac{3}{7}

(at the min).

miscalcif.

Questions?

Classification Tree Learning

Gini, Entropy, Misclassif.

Algorithm: Classification Tree Learning

Input: Impurity measure Q , parameter \max_depth &
 training set

Output: Tree $\{(x_i, y_i)\}$ $y_i \in \{1, \dots, k\}$

Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q , parameter max_depth & training set

Output: Tree

1 root \leftarrow all training samples

Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q , parameter max_depth & training set

Output: Tree

```
1 root ← all training samples
2 for  $d \leftarrow 1$  to  $\text{max\_depth}$  do
3   |
4   |
5   |
6 return tree
```

Classification Tree Learning

Algorithm: Classification Tree Learning

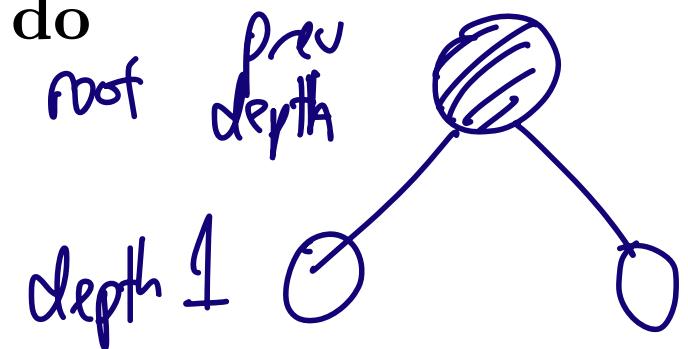
Input: Impurity measure Q , parameter max_depth & training set

Output: Tree

```

1 root ← all training samples
2 for  $d \leftarrow 1$  to  $\text{max\_depth}$  do
3   for each leaf node  $m$  at depth  $d - 1$  do
4   |
5   |
6 return tree
  
```

Go thru leaf nodes
from prev depth



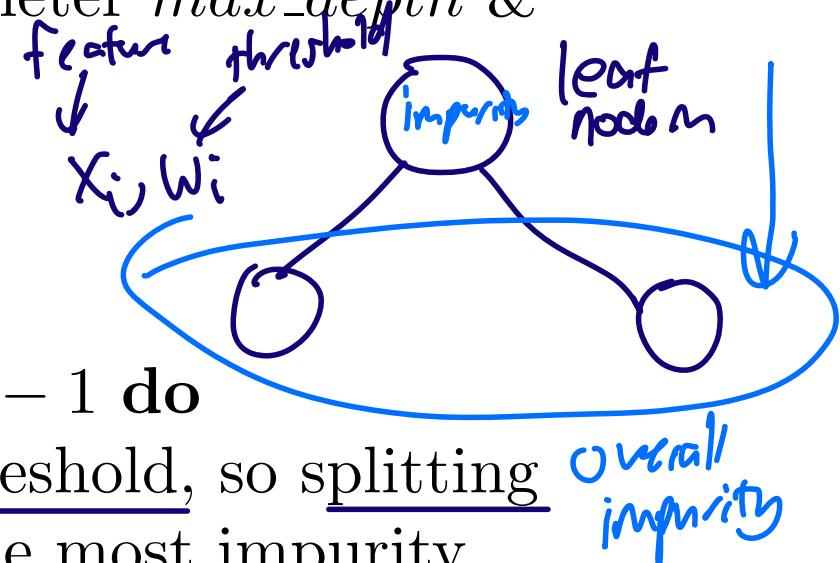
Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q , parameter max_depth & training set

Output: Tree

- 1 root \leftarrow all training samples
- 2 **for** $d \leftarrow 1$ **to** max_depth **do**
- 3 **for** each leaf node m at depth $d - 1$ **do**
- 4 Find best feature & best threshold, so splitting node m into two reduces the most impurity
- 5
- 6 **return** tree



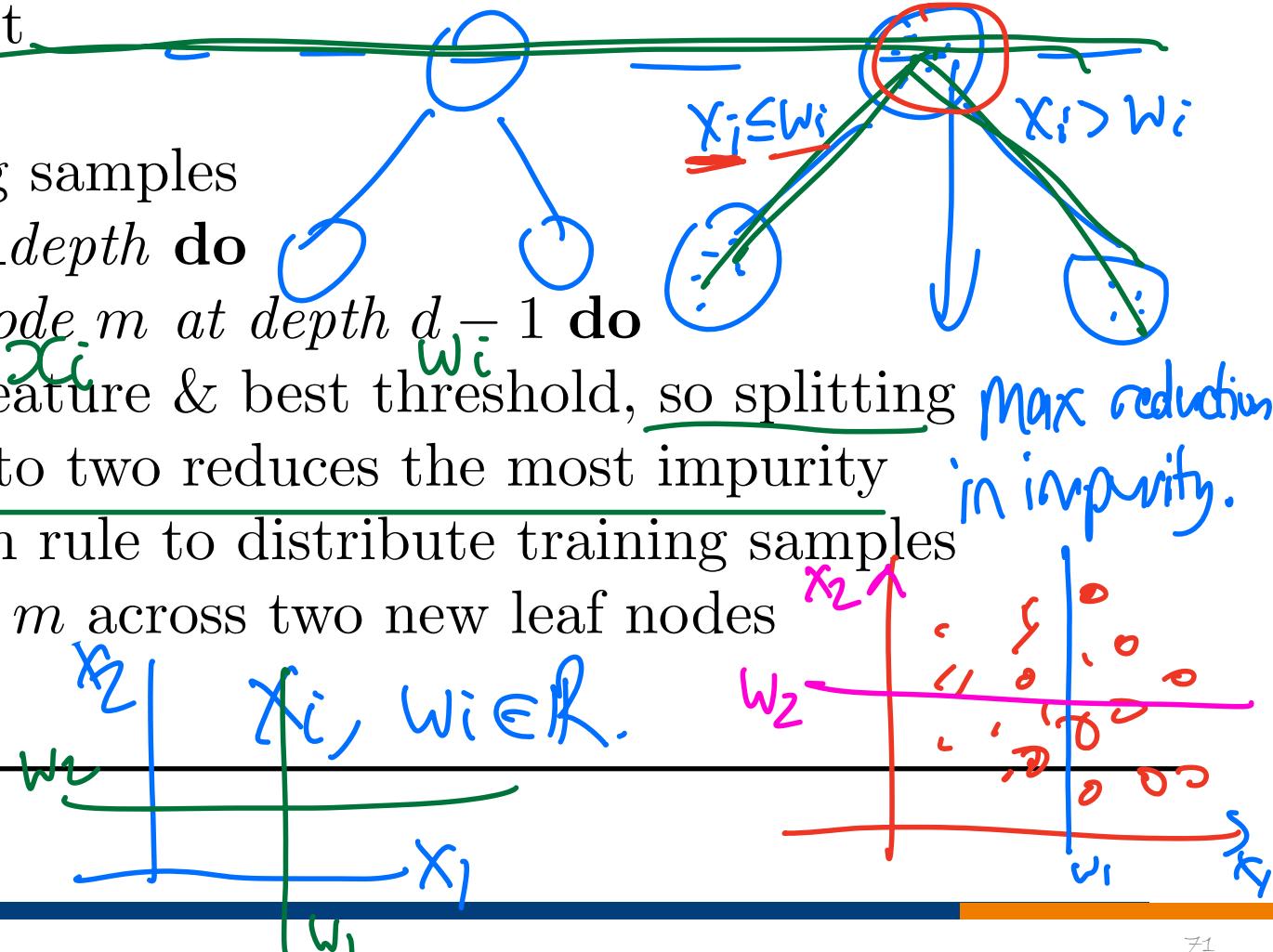
Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q , parameter max_depth & training set

Output: Tree

- 1 root \leftarrow all training samples
- 2 **for** $d \leftarrow 1$ **to** max_depth **do**
- 3 **for** each leaf node m at depth $d - 1$ **do**
- 4 Find best feature x_i & best threshold w_i , so splitting node m into two reduces the most impurity Max reduction in impurity.
- 5 Use decision rule to distribute training samples from node m across two new leaf nodes
- 6 **return tree**



Questions?

Advantages / Disadvantages

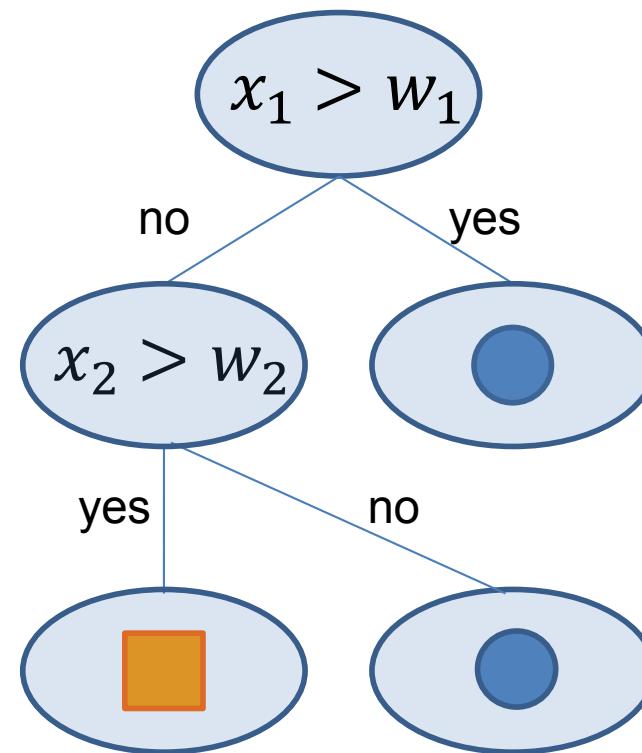
Advantages

- Easy to visualize & understand tree

Easy to understand.

Visualize

Interpretable.



Advantages / Disadvantages

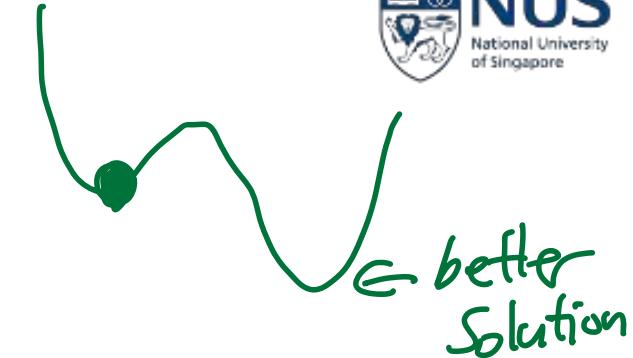
Advantages

- Easy to visualize & understand tree
- Can work with a mix of continuous and discrete data
- Less data cleaning required (i.e., less affected by outliers)

Advantages / Disadvantages

Advantages

- Easy to visualize & understand tree
- Can work with a mix of continuous and discrete data
- Less data cleaning required (i.e., less affected by outliers)
- Makes less assumptions about the relationship between features & target



$$y = \text{sgn}(w_0 + w_1 x + w_2 x^2)$$

Advantages / Disadvantages

Advantages

- Easy to visualize & understand tree
- Can work with a mix of continuous and discrete data
- Less data cleaning required
- Makes less assumptions about the relationship between features & target

Disadvantages

- Trees can become overly complex resulting in overfitting

Advantages / Disadvantages

Advantages

- Easy to visualize & understand tree
- Can work with a mix of continuous and discrete data
- Less data cleaning required
- Makes less assumptions about the relationship between features & target

Disadvantages

- Trees can become overly complex resulting in overfitting
- Trees can be unstable, e.g., small changes in training data can result in very different trees

random forests.

To reduce overfitting...

- One or more of the following can help reduce overfitting
 - Set maximum depth for the tree

To reduce overfitting...

- One or more of the following can help reduce overfitting
 - Set maximum depth for the tree
 - Set minimum number of samples for splitting a leaf node, e.g., if leaf node has less than 10 samples, then do not split node

To reduce overfitting...

- One or more of the following can help reduce overfitting
 - Set maximum depth for the tree
 - Set minimum number of samples for splitting a leaf node, e.g., if leaf node has less than 10 samples, then do not split node
 - Set minimum decrease in impurity, e.g., if selecting the best feature & threshold does not improve impurity by at least 1%, then do not split the leaf node

To reduce overfitting...

- One or more of the following can help reduce overfitting
 - Set maximum depth for the tree
 - Set minimum number of samples for splitting a leaf node, e.g., if leaf node has less than 10 samples, then do not split node
 - Set minimum decrease in impurity, e.g., if selecting the best feature & threshold does not improve impurity by at least 1%, then do not split the leaf node
 - Instead of looking at all features when considering how to split a leaf node, we can randomly look at a subset (e.g., square root of the total number of features)

$$\cancel{f(x_1, x_2) \geq w_{1,2}}$$

$$x_i \geq w_i$$

$$x_i < w_i$$

Questions?

Regression Trees

- Classification trees seek to predict discrete variables (i.e., classification)
- Regression trees seek to predict continuous variables (i.e., regression)

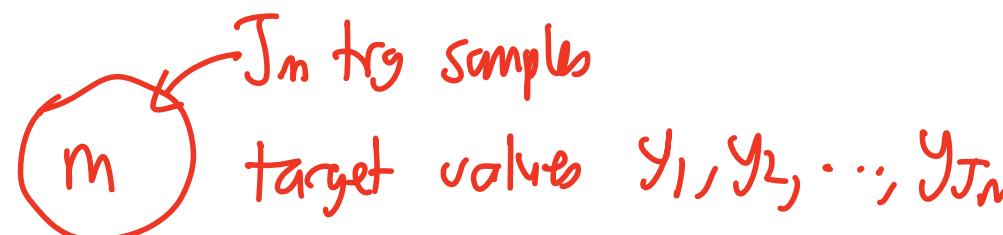
Regression Trees

- Classification trees seek to predict discrete variables (i.e., classification)
- Regression trees seek to predict **continuous** variables (i.e., regression)
- Can use same approach as before, but instead of minimizing impurity, we can try to minimize mean square error (MSE)

Gini, entropy
discrete variable $\{p_i\}$

Regression Trees

- Classification trees seek to predict discrete variables (i.e., classification)
- Regression trees seek to predict **continuous** variables (i.e., regression)
- Can use same approach as before, but instead of minimizing impurity, we can try to **minimize mean square error (MSE)**
- Suppose there are J_m training samples in a leaf node m of the regression tree with target values y_1, y_2, \dots, y_{J_m}



 Our prediction $\hat{y}_m = \text{Average}(y_1, y_2, \dots, y_{J_m})$

Regression Trees

- Classification trees seek to predict discrete variables (i.e., classification)
- Regression trees seek to predict continuous variables (i.e., regression)
- Can use same approach as before, but instead of minimizing impurity, we can try to minimize mean square error (MSE)
- Suppose there are J_m training samples in a leaf node m of the regression tree with target values y_1, y_2, \dots, y_{J_m}
 - We can predict $\hat{y}_m = \frac{1}{J_m} \sum_{j=1}^{J_m} y_j$

Regression Trees

- Classification trees seek to predict discrete variables (i.e., classification)
- Regression trees seek to predict continuous variables (i.e., regression)
- Can use same approach as before, but instead of minimizing impurity, we can try to minimize mean square error (MSE)
- Suppose there are J_m training samples in a leaf node m of the regression tree with target values $\underline{y_1, y_2, \dots, y_{J_m}}$

- We can predict $\hat{y}_m = \frac{1}{J_m} \sum_{j=1}^{J_m} y_j$

- Then MSE of node m is given by $S_m = \frac{1}{J_m} \sum_{j=1}^{J_m} (y_j - \hat{y}_m)^2$

prediction
 \downarrow
 $\overbrace{}^{\text{Q}_m: \text{entropy, Gini}}$

Regression Trees

- Classification trees seek to predict discrete variables (i.e., classification)
 - Regression trees seek to predict continuous variables (i.e., regression)
 - Can use same approach as before, but instead of minimizing impurity, we can try to minimize mean square error (MSE)
 - Suppose there are J_m training samples in a leaf node m of the regression tree with target values y_1, y_2, \dots, y_{J_m}
 - We can predict $\hat{y}_m = \frac{1}{J_m} \sum_{j=1}^{J_m} y_j$
 - Then MSE of node m is given by $S_m = \frac{1}{J_m} \sum_{j=1}^{J_m} (y_j - \hat{y}_m)^2$
 - Across all leaf nodes, $\underline{\text{total MSE } S = \sum_m \frac{J_m}{N} S_m}$, where N is the total number of data samples
- \uparrow weighted sum.

Regression Tree Learning

- Algorithm is basically the same as classification tree learning

Algorithm: Regression Tree Learning

Input: parameter max_depth & training set

Output: Tree

```
1 root ← all training samples
2 for  $d \leftarrow 1$  to  $\text{max\_depth}$  do
3   for each leaf node  $m$  at depth  $d - 1$  do
4     Find best feature & best threshold, so splitting
      node  $m$  into two reduces MSE the most
5     Use decision rule to distribute training samples
      from node  $m$  across two new leaf nodes
6 return tree
```

Regression Tree Learning

- Algorithm is basically the same as classification tree learning
- Various approaches to reduce overfitting also apply here

Algorithm: Regression Tree Learning

Input: parameter max_depth & training set

Output: Tree

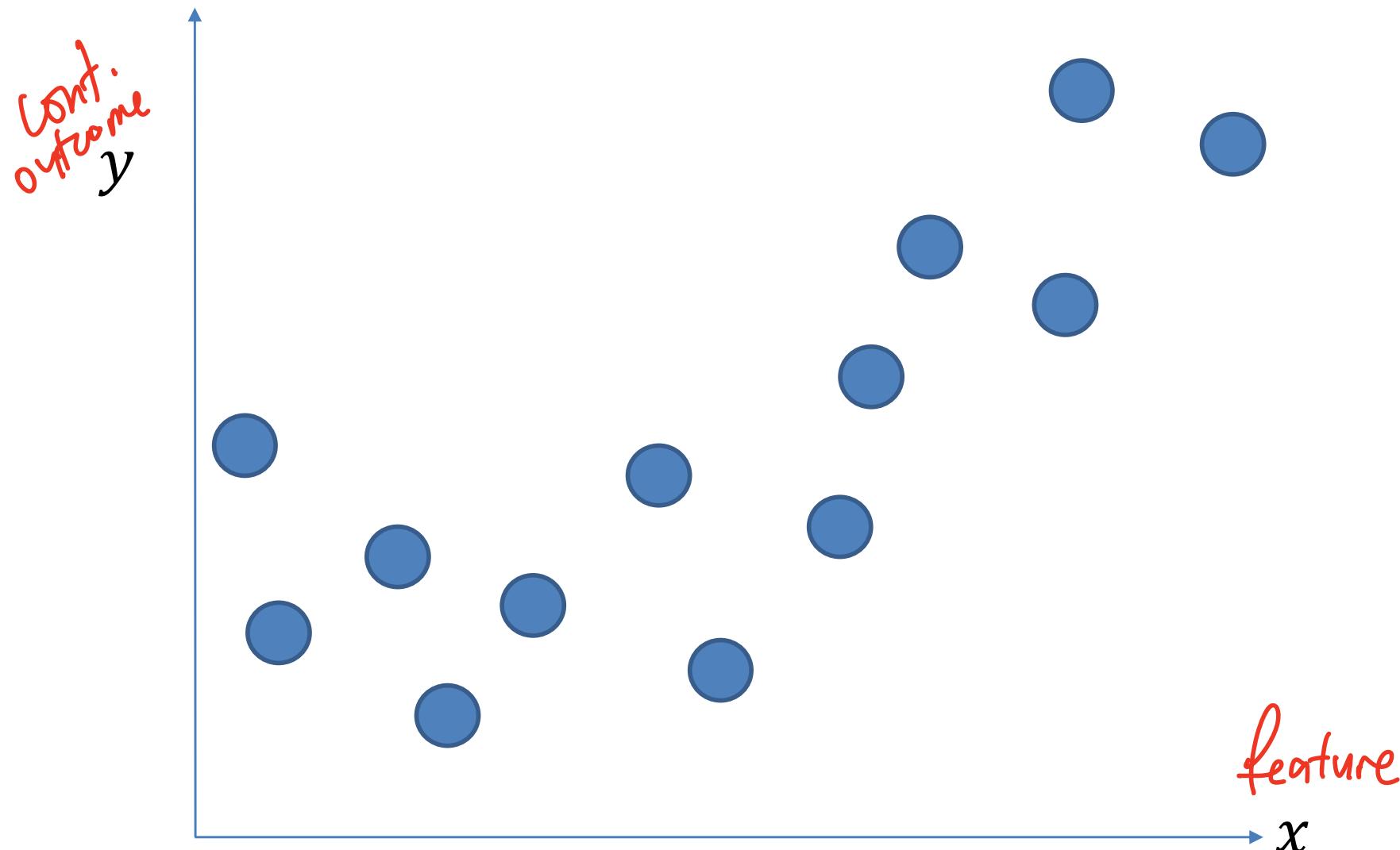
- 1 root \leftarrow all training samples
- 2 **for** $d \leftarrow 1$ **to** max_depth **do**
- 3 **for** each leaf node m at depth $d - 1$ **do**
- 4 Find best feature & best threshold, so splitting node m into two reduces **MSE** the most
- 5 Use decision rule to distribute training samples from node m across two new leaf nodes
- 6 **return** tree

Adv & Disadv

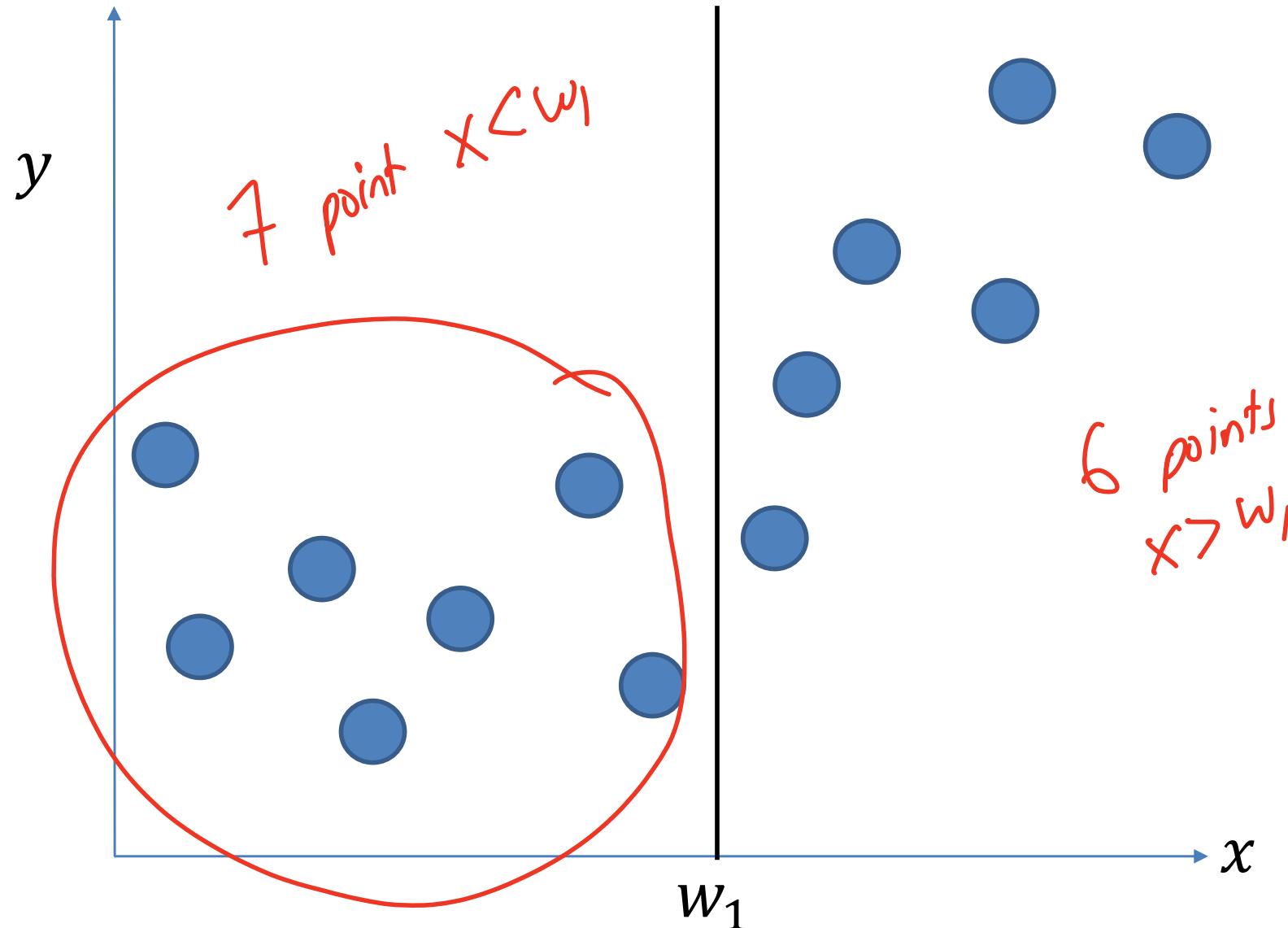
*Use same techniques
to reduce overfitting.*

Questions?

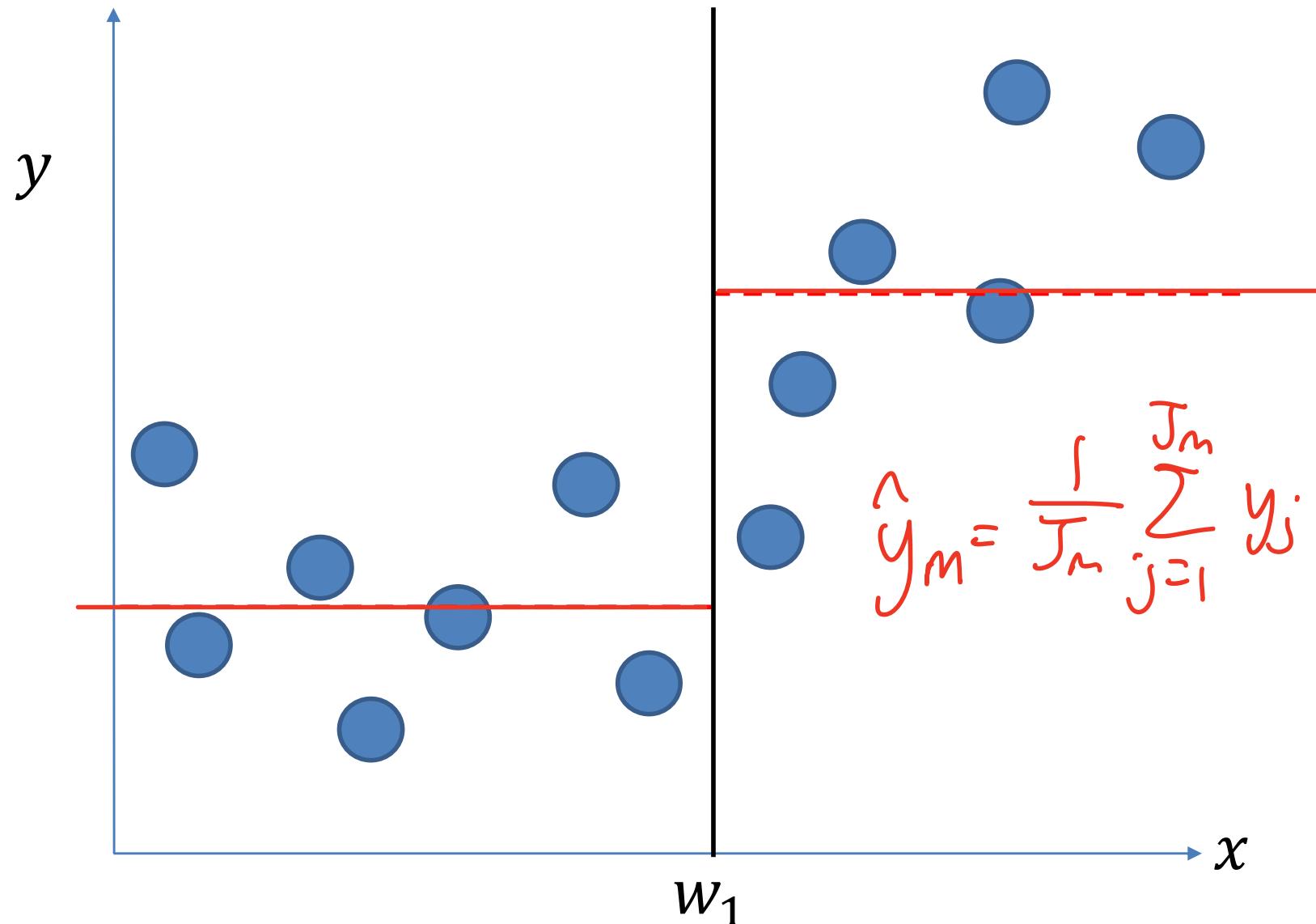
Regression Tree Example



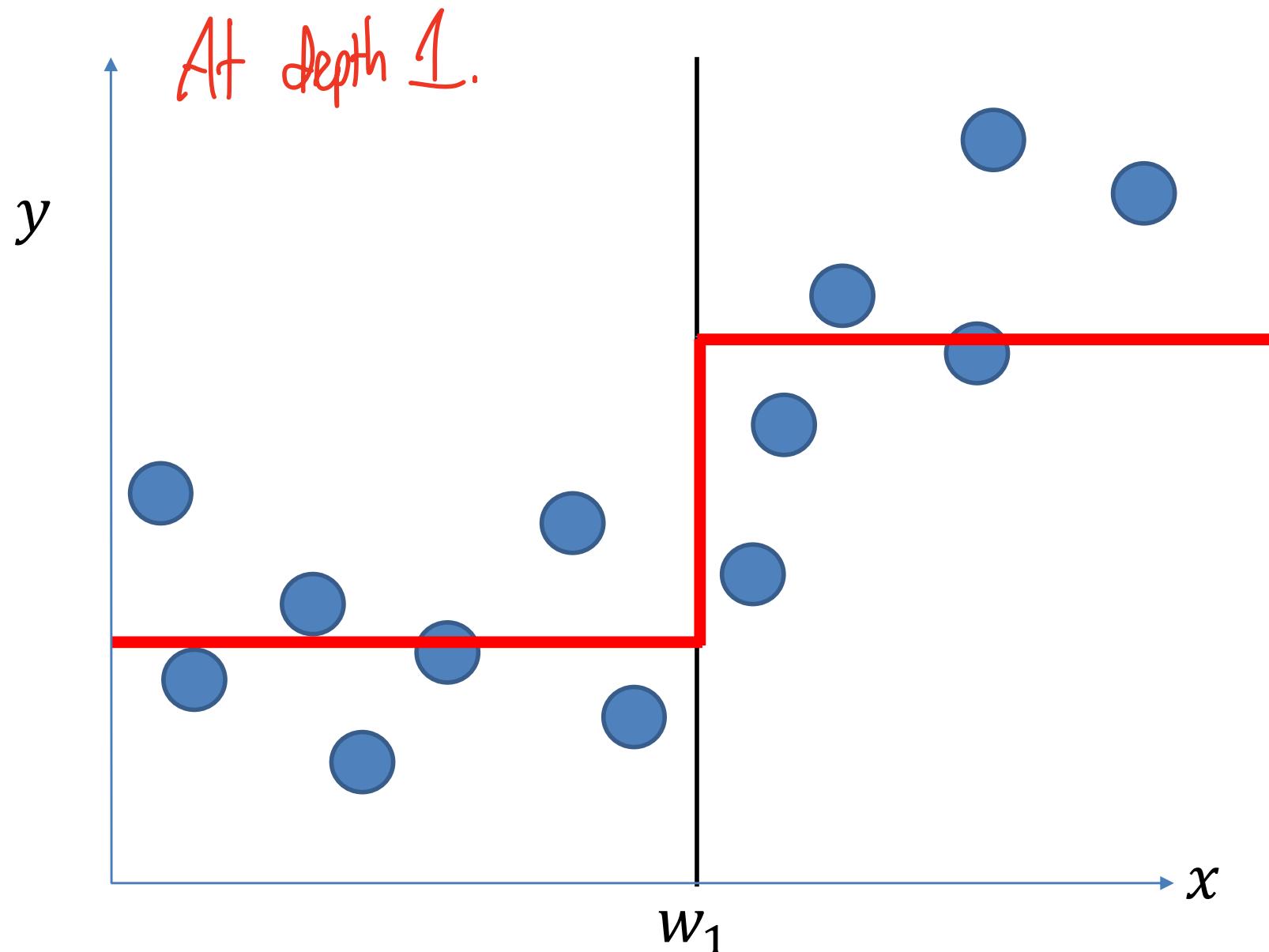
Regression Tree Example



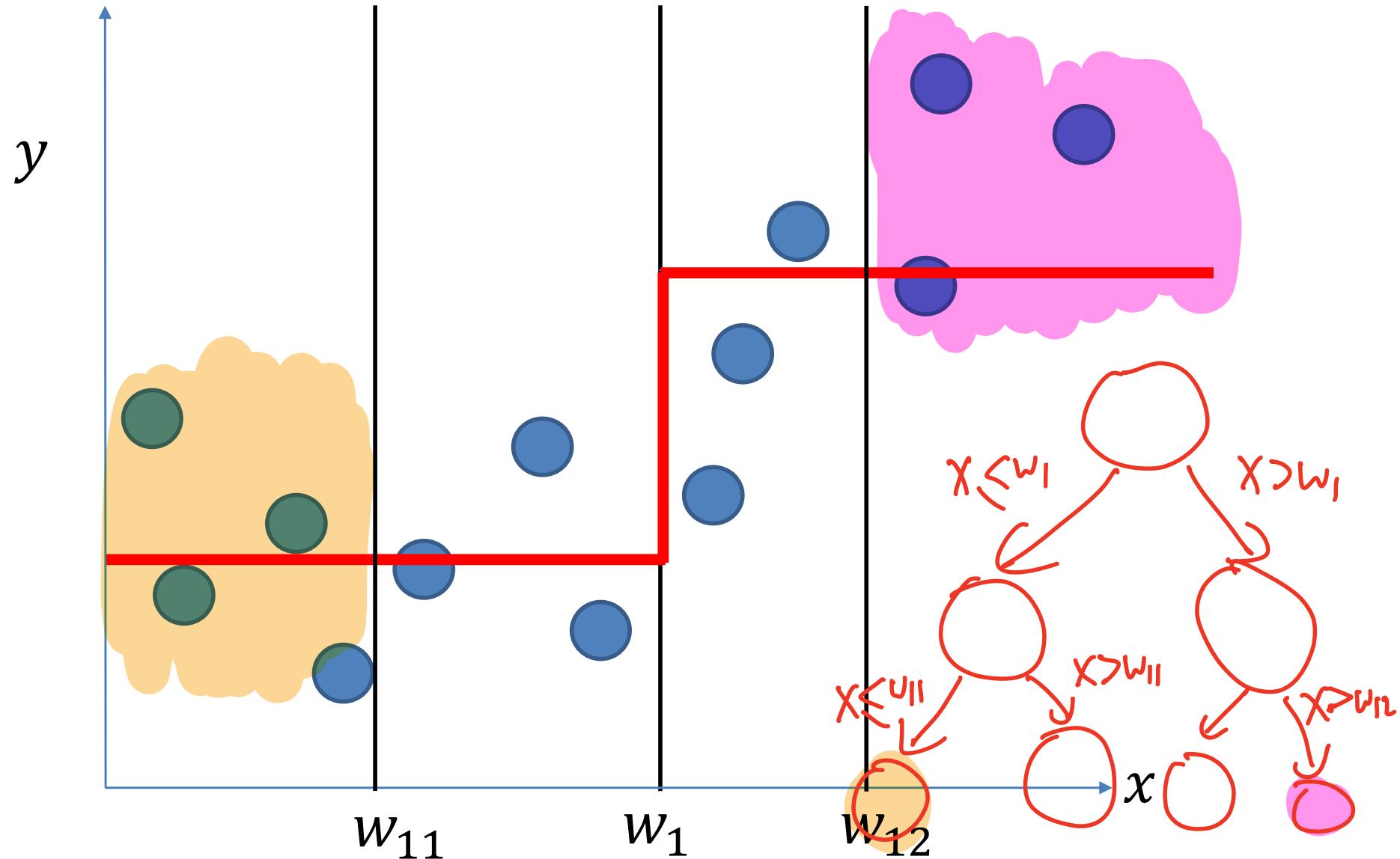
Regression Tree Example



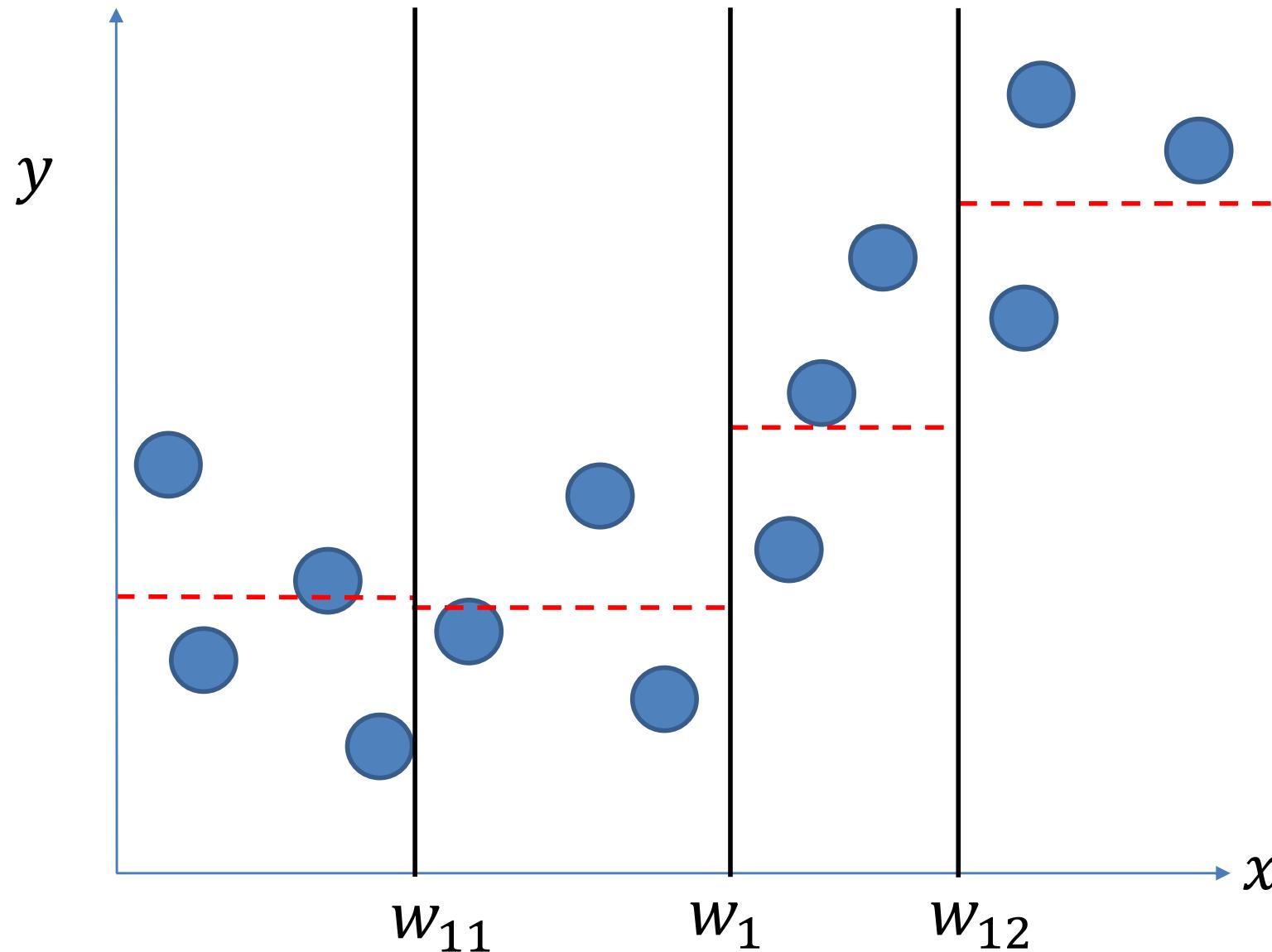
Regression Tree Example



Regression Tree Example

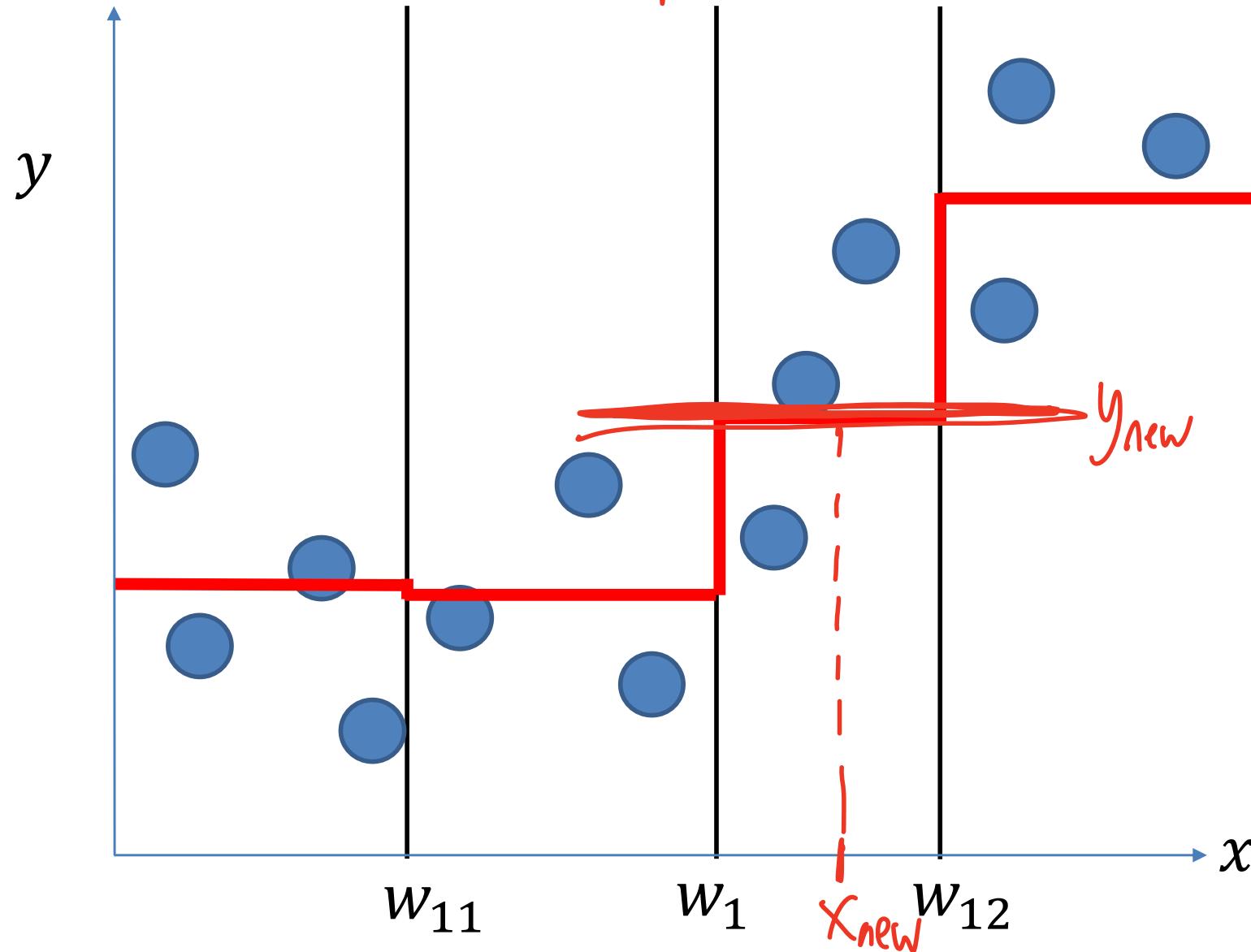


Regression Tree Example



Regression Tree Example

At depth 2



Example of Regression Tree

Example of Regression Tree

- Consider house prices in Singapore.
- Target variable is Price P .
- Attributes are House Size S and Number of Rooms R .

	House Size ('000 sq ft)	Num of Rooms	Price ('000,000 SGD)
1	0.5	2	0.19
2	0.6	1	0.23
3	1.0	3	0.28
4	2.0	5	0.42
5	3.0	4	0.53
6	3.2	6	0.75
7	3.8	7	0.80

- Note that I have arranged the data points in increasing order of P , which so happens to be increasing order of S as well. However, this is not the same order as that of R .

Example of Regression Tree

Mean Squared Error (MSE)

- The MSE for a node m with samples $\{y_i : 1 \leq i \leq J_m\}$ is

$$\text{MSE}_m = \frac{1}{J_m} \sum_{i=1}^{J_m} (y_i - \hat{\mu}_m)^2 \quad \text{where} \quad \hat{\mu}_m = \frac{1}{J_m} \sum_{i=1}^{J_m} y_i.$$

- The overall MSE is $\text{MSE}_P = 0.0520$.

Example of Regression Tree

Calculation of MSE for House Size Split

- Focus first on the House Size attribute S . If we set the threshold at $\tau = 0.75$, then the targets of the two classes are $\{0.19, 0.23\}$ and $\{0.28, 0.42, 0.53, 0.75, 0.80\}$. The individual conditional MSEs are

$$\text{MSE}_{P|S<0.75} = 4 \times 10^{-4} \quad \text{and} \quad \text{MSE}_{P|S \geq 0.75} = 0.0385.$$

- Thus, the averaged conditional MSE with a split of S at 0.75 is

$$\text{MSE}_{P|S(0.75)} = \frac{2}{7}\text{MSE}_{P|S<0.75} + \frac{5}{7}\text{MSE}_{P|S \geq 0.75} = 0.0276.$$

- Sweep through all possible thresholds τ to determine the best threshold for attribute S .

$\text{MSE}_{P S(0.55)}$	$\text{MSE}_{P S(0.75)}$	$\text{MSE}_{P S(1.5)}$	$\text{MSE}_{P S(2.5)}$	$\text{MSE}_{P S(3.1)}$	$\text{MSE}_{P S(3.5)}$
0.0402	0.0276	0.0145	0.0102	0.0116	0.0325

Example of Regression Tree

Calculation of MSE for # Rooms Split

- Rearrange the target variables in order of the house sizes. Doing so we get $(0.23, 0.19, 0.28, 0.53, 0.42, 0.75, 0.80)$. Now we sweep through all possible thresholds τ for R to get the following averaged conditional MSEs.
- We get the following table.

$MSE_{P R(1.5)}$	$MSE_{P R(2.5)}$	$MSE_{P R(3.5)}$	$MSE_{P R(4.5)}$	$MSE_{P R(5.5)}$	$MSE_{P R(6.5)}$
0.0435	0.0276	0.0145	0.0222	0.0116	0.0325

Example of Regression Tree

Where is the First Split?

- Minima of the split of the S and R variables at different thresholds τ are shaded.
- Choose the minimum MSE as doing so and keeping in mind that MSE_P is the same throughout.
- Gain which is

$$\text{Gain}(S(\tau); P) = \text{MSE}_P - \text{MSE}_{P|S(\tau)}$$

or

$$\text{Gain}(R(\tau); P) = \text{MSE}_P - \text{MSE}_{P|R(\tau)}$$

for various τ .

- Minimum MSE is attained for the split of the S attribute at $\tau = 2.5$.

Example of Regression Tree

Where is the First Split?

- We should first split the dataset into two branches, the left branch indicating $S < 2.5$ and the right with $S \geq 2.5$.
- Split the dataset into two sub-datasets and we may decide to stop or split the R feature.
- If we decide to stop, then for any new/test house with a house size of < 2.5 , we will predict that its price is the average of the houses in our training set whose size is < 2.5 , i.e.,

$$(0.19 + 0.23 + 0.28 + 0.42)/4 = 0.28.$$

- For a new/test house with a house size of ≥ 2.5 , we will predict that its price is the average of the houses in our training set whose size is ≥ 2.5 , i.e.,

$$(0.53 + 0.75 + 0.80)/3 = 0.6933.$$

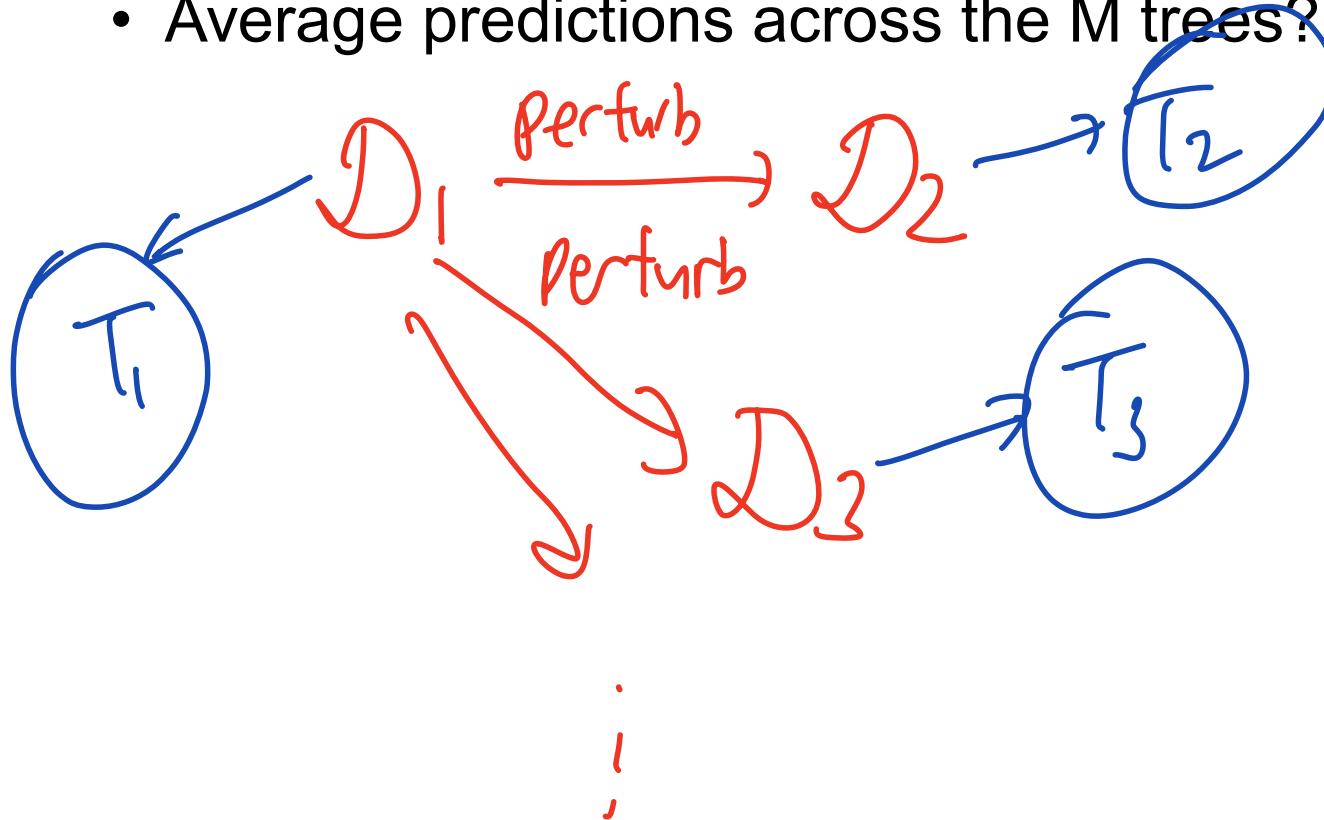
Questions?

To reduce instability...

- For both classification & regression trees, small perturbations to data can result in very different trees => low bias, high variance

To reduce instability...

- For both classification & regression trees, small perturbations to data can result in very different trees => low bias, high variance
- To reduce variance, we can perturb training set to generate M perturbed training sets
 - Train one tree for each perturbed training set
 - Average predictions across the M trees?



To reduce instability...

- For both classification & regression trees, small perturbations to data can result in very different trees => low bias, high variance
- To reduce variance, we can perturb training set to generate M perturbed training sets
 - Train one tree for each perturbed training set
 - **Average predictions across the M trees?**
- For example, if we have 100 regression trees (trained from 100 perturbed training sets)
 - Given features x from new test sample, the i -th tree predicts $f_i(x)$
 - Then final prediction is $\underbrace{\frac{1}{100} \sum_{i=1}^{100} f_i(x)}$

ER

To reduce instability...

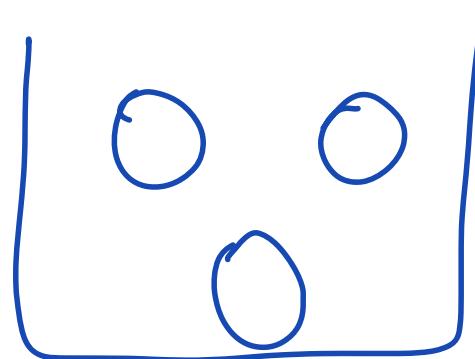
- For both classification & regression trees, small perturbations to data can result in very different trees => low bias, high variance
- To reduce variance, we can perturb training set to generate M perturbed training sets
 - Train one tree for each perturbed training set
 - Average predictions across the M trees?
- For example, if we have 100 regression trees (trained from 100 perturbed training sets)
 - Given features x from new test sample, the i -th tree predicts $f_i(x)$
 - Then final prediction is $\frac{1}{100} \sum_{i=1}^{100} f_i(x)$
- For example, if we have 100 classification trees (trained from 100 perturbed training sets) *majority voting*
 - Given features x from new test sample, the i -th tree predicts $g_i(x)$
 - Then final prediction is the most frequent class among 100 predictions $g_1(x), g_2(x), \dots, g_{100}(x)$

Random Forest

- To perturb data, we can apply “**bootstrapping**” to the training set to create a new “bootstrapped” dataset

Random Forest

- To perturb data, we can apply “bootstrapping” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement



Random Forest

- To perturb data, we can apply “**bootstrapping**” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement
 - For example, given training set with 3 data samples $\{x_1, y_1\}$,
 $\{x_2, y_2\}$, $\{x_3, y_3\}$,

Random Forest

- To perturb data, we can apply “bootstrapping” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement
 - For example, given training set with 3 data samples $\{x_1, y_1\}$, $\{x_2, y_2\}$, $\{x_3, y_3\}$, a bootstrapped training set might comprise sample 1 $\{x_2, y_2\}$,

Random Forest

- To perturb data, we can apply “bootstrapping” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement
 - For example, given training set with 3 data samples $\{x_1, y_1\}$, $\{x_2, y_2\}$, $\{x_3, y_3\}$, a bootstrapped training set might comprise sample 1 $\{x_2, y_2\}$, sample 2 $\{x_2, y_2\}$

Random Forest

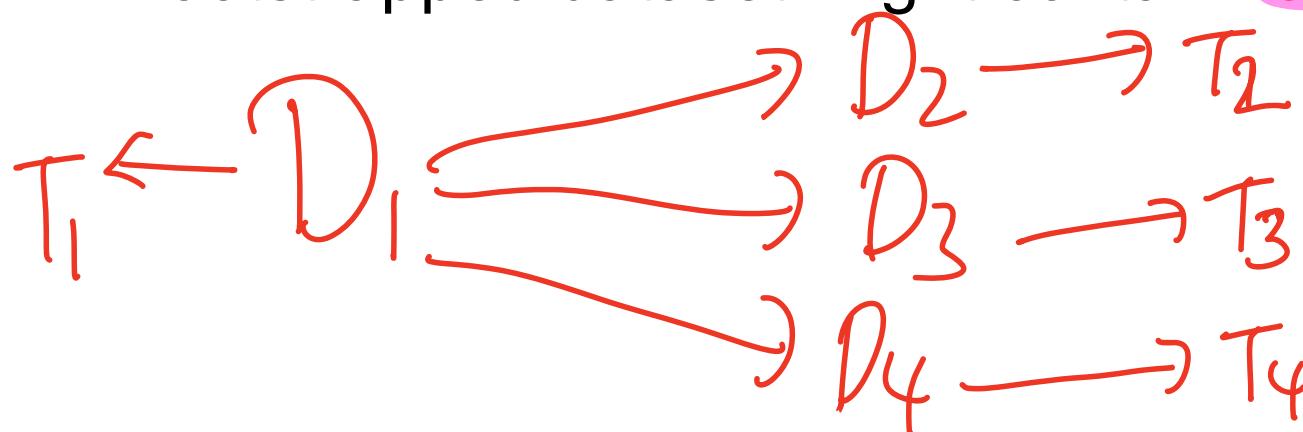
- To perturb data, we can apply “bootstrapping” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement
 - For example, given training set with 3 data samples $\{x_1, y_1\}$, $\{x_2, y_2\}$, $\{x_3, y_3\}$, a bootstrapped training set might comprise sample 1 $\{x_2, y_2\}$, sample 2 $\{x_2, y_2\}$ and sample 3 $\{x_1, y_1\}$

Random Forest

- To perturb data, we can apply “bootstrapping” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement
 - For example, given training set with 3 data samples $\{x_1, y_1\}$, $\{x_2, y_2\}$, $\{x_3, y_3\}$, a bootstrapped training set might comprise sample 1 $\{x_2, y_2\}$, sample 2 $\{x_2, y_2\}$ and sample 3 $\{x_1, y_1\}$
 - Bootstrapped dataset is the **same size as original dataset**

Random Forest

- To perturb data, we can apply “bootstrapping” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement
 - For example, given training set with 3 data samples $\{x_1, y_1\}$, $\{x_2, y_2\}$, $\{x_3, y_3\}$, a bootstrapped training set might comprise sample 1 $\{x_2, y_2\}$, sample 2 $\{x_2, y_2\}$ and sample 3 $\{x_1, y_1\}$
 - Bootstrapped dataset is the same size as original dataset
 - Bootstrapped dataset might contain **repeated samples**



Random Forest

- To perturb data, we can apply “bootstrapping” to the training set to create a new “bootstrapped” dataset
- Bootstrapping is procedure in which we sample data with replacement
 - For example, given training set with 3 data samples $\{x_1, y_1\}$, $\{x_2, y_2\}$, $\{x_3, y_3\}$, a bootstrapped training set might comprise sample 1 $\{x_2, y_2\}$, sample 2 $\{x_2, y_2\}$ and sample 3 $\{x_1, y_1\}$
 - Bootstrapped dataset is the same size as original dataset
 - Bootstrapped dataset might contain repeated samples
 - Bootstrapped dataset might **not** contain some samples from original dataset

Random Forest

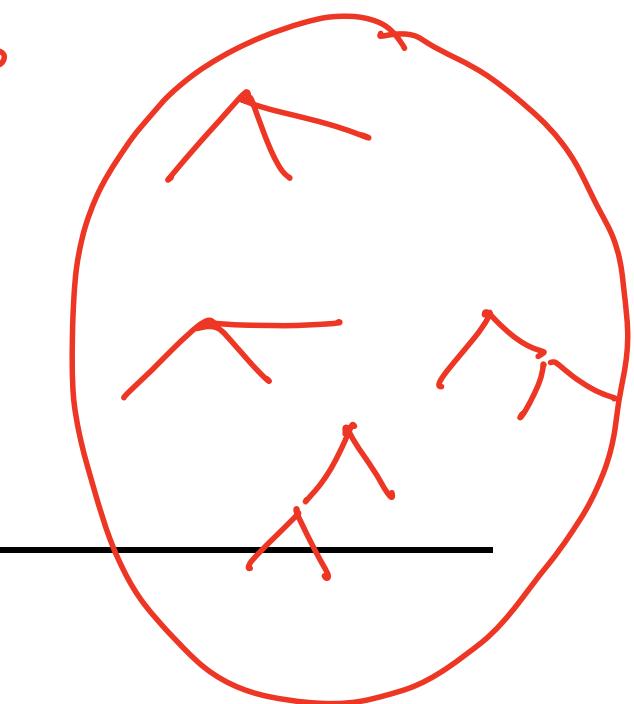
Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples

Output: Forest

$T_1, T_2, T_3 \dots, T_{100}$

1
2
3
4
5



forest

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples

Output: Forest

```
1 for  $t \leftarrow 1$  to  $max\_trees$  do
2   dataset  $\leftarrow$  bootstrap( $N$  training samples)
3
4
5
```

Sampling w/ replacement

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples

Output: Forest

```
1 for  $t \leftarrow 1$  to  $\text{max\_trees}$  do
2   dataset  $\leftarrow$  bootstrap( $N$  training samples)
3   Tt trees[t]  $\leftarrow$  TreeLearning(dataset)
4
5
```

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples

Output: Forest

```
1 for  $t \leftarrow 1$  to  $\text{max\_trees}$  do
2     dataset  $\leftarrow$  bootstrap( $N$  training samples)
3     trees[t]  $\leftarrow$  TreeLearning(dataset)
4 4 forest  $\leftarrow$  average(trees)
5 return forest
```

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples

Output: Forest

```
1 for  $t \leftarrow 1$  to  $max\_trees$  do
2     dataset  $\leftarrow$  bootstrap( $N$  training samples)
3     trees[t]  $\leftarrow$  TreeLearning(dataset)
4 forest  $\leftarrow$  average(trees)
5 return  $forest$ 
```

- To increase randomness, when training the trees, instead of looking at all features when considering how to split a node, we can randomly look at a subset (e.g., square root of the total number of features)

Questions?

Summary

- Decision tree: series of binary decisions to arrive at prediction of target variable

Summary

- Decision tree: series of binary decisions to arrive at prediction of target variable
- Classification tree predicts discrete target class

Summary

- Decision tree: series of binary decisions to arrive at prediction of target variable
- Classification tree predicts discrete target class
- Classification tree learning
 - Impurity measures: Gini, Entropy, Misclassification rate
 - For each leaf node, find best feature and threshold to split node to minimize impurity

Summary

- Decision tree: series of binary decisions to arrive at prediction of target variable
- Classification tree predicts discrete target class
- Classification tree learning
 - Impurity measures: Gini, Entropy, Misclassification rate
 - For each leaf node, find best feature and threshold to split node to minimize impurity
- Regression tree predicts continuous target
 - Minimize MSE instead of impurity

Summary

- **Decision tree**: series of binary decisions to arrive at prediction of target variable
 - **Classification tree** predicts discrete target class
 - Classification tree learning *Check that overall impurity does not ↑*
 - Impurity measures: Gini, Entropy, Misclassification rate
 - For each leaf node, find best feature and threshold to split node to minimize impurity
 - **Regression tree** predicts continuous target
 - Minimize MSE instead of impurity
 - **Random forest**
 - Generate multiple bootstrapped training sets
 - Train on each bootstrapped training set & average
- $D_1 \neq D_2$
 \downarrow
 $T_1 \neq T_2$