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

- 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

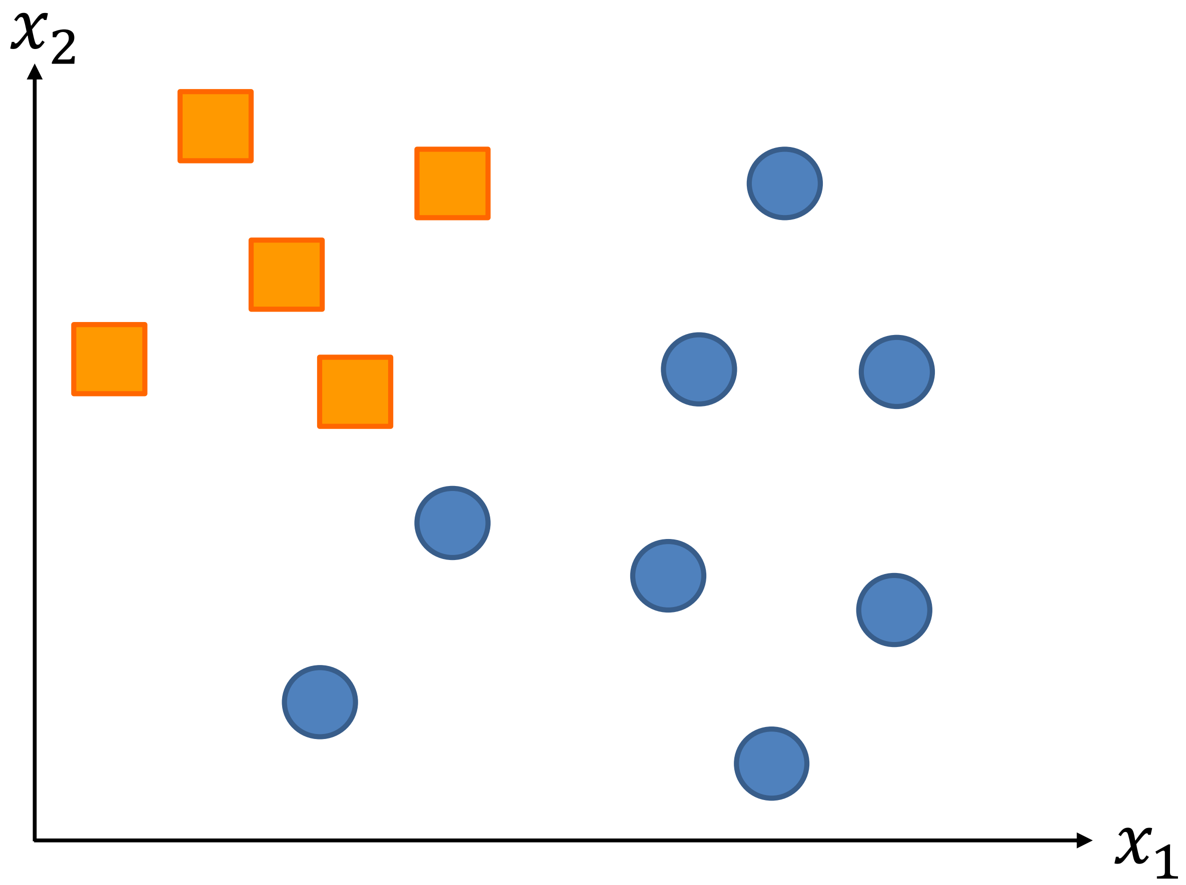
- Supervised learning: given feature(s) x, we want to predict target y to be some $f(x)$
  - If y is continuous, problem is called "regression"
  - If y is discrete, problem is called "classification"
- Previous lectures used linear models for regression (and classification)
  - Nonlinearity added by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

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

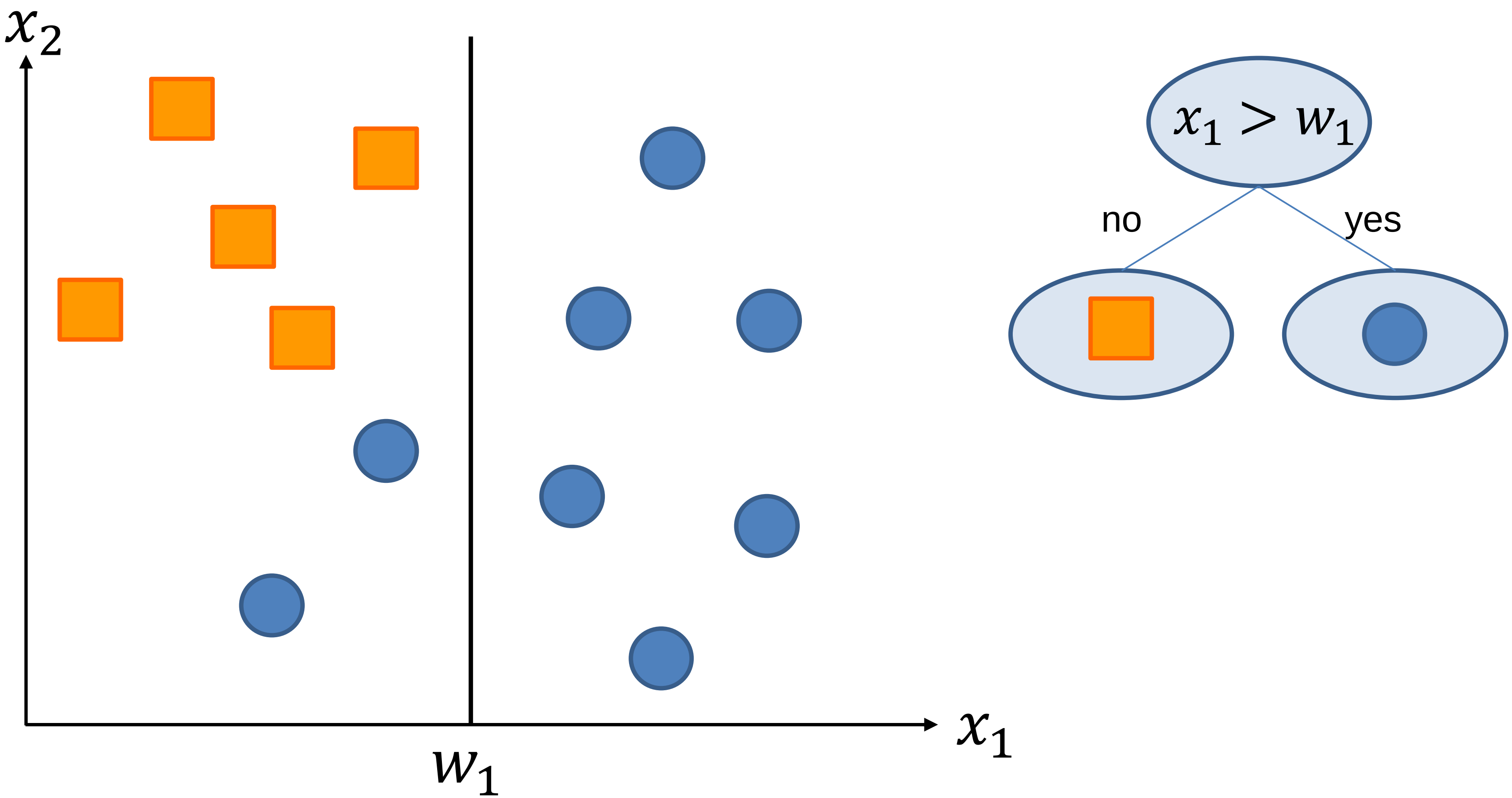
Decision Tree Classification Example

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



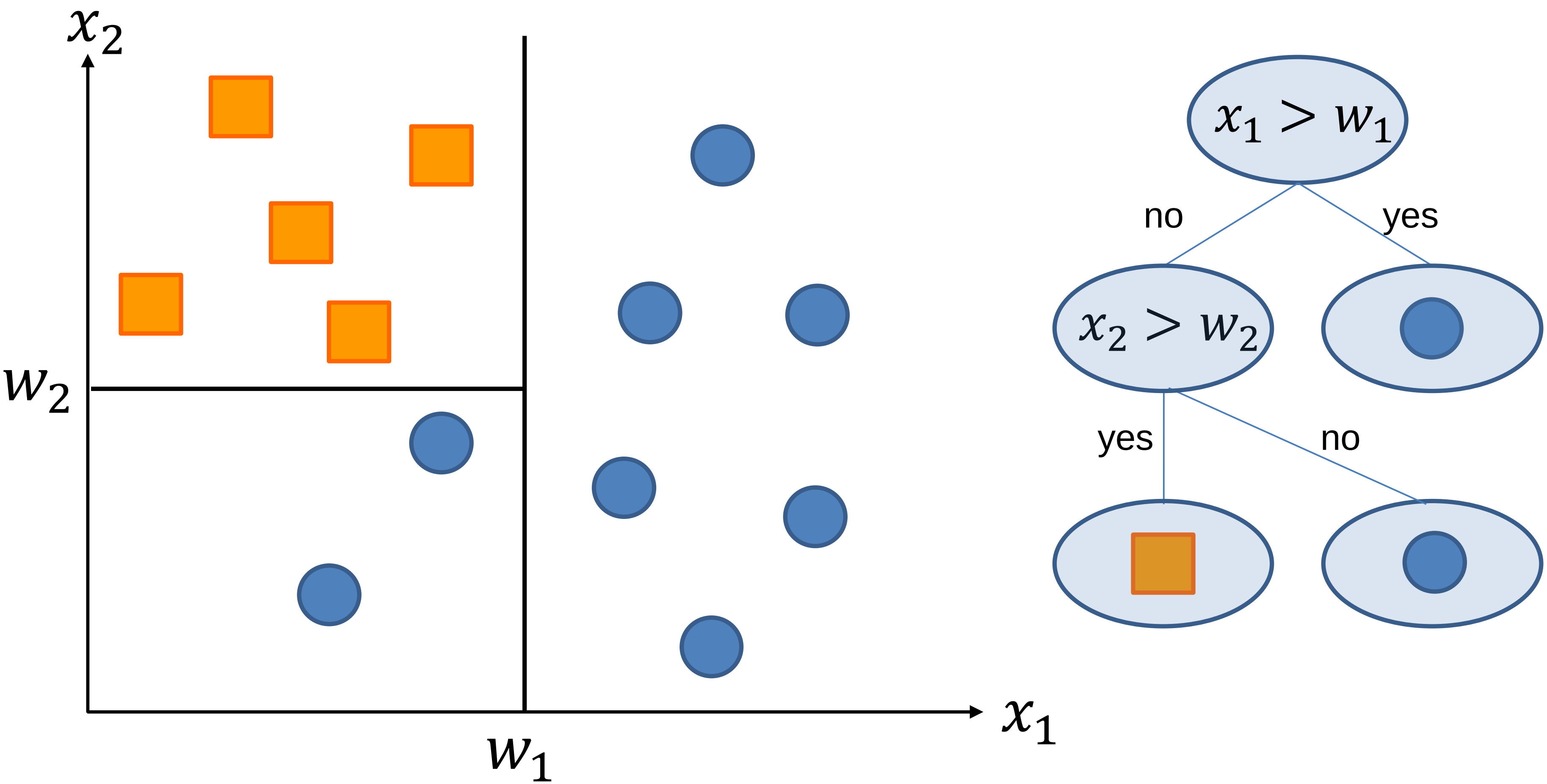
Decision Tree Classification Example

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



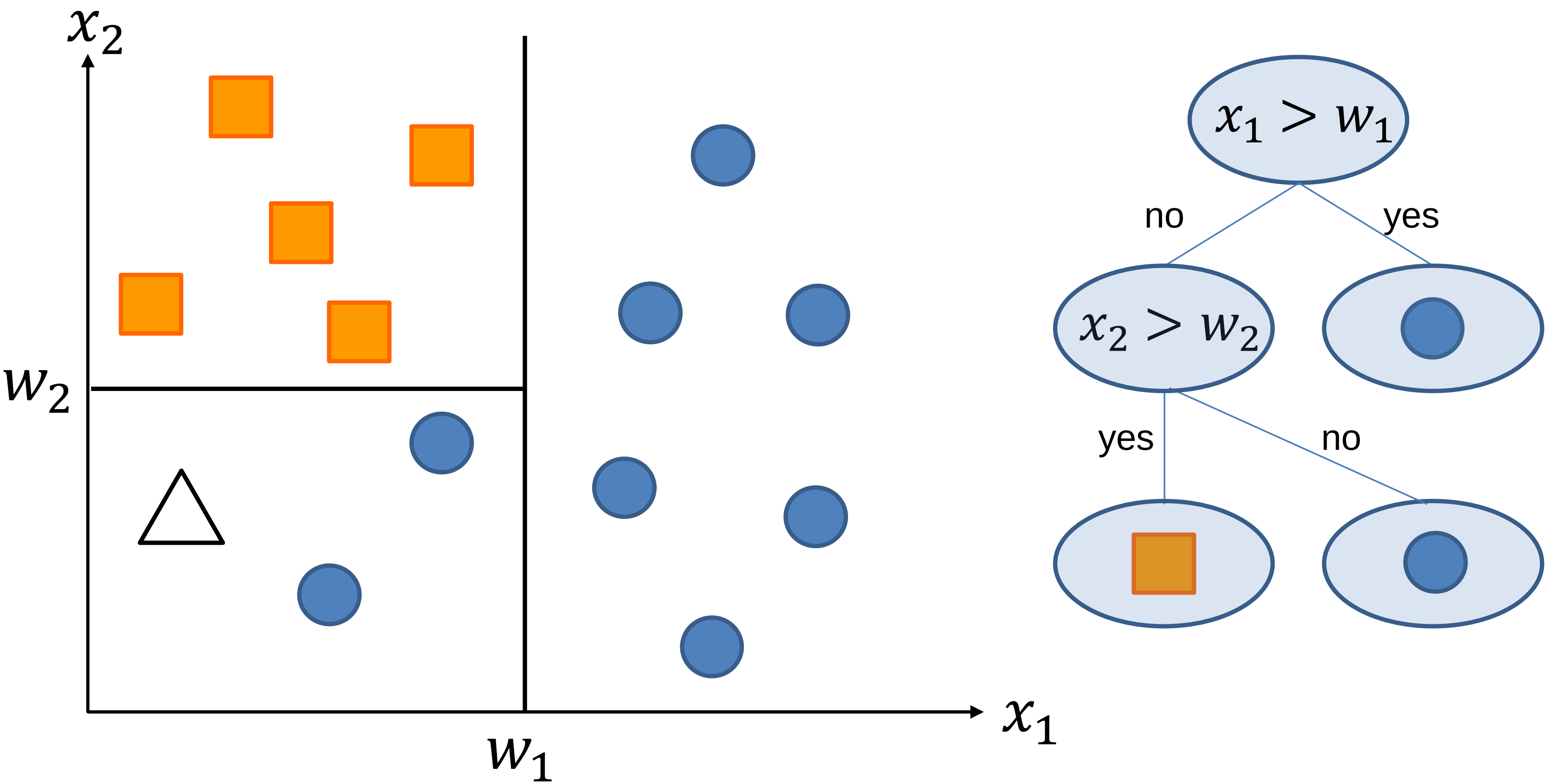
Decision Tree Classification Example

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



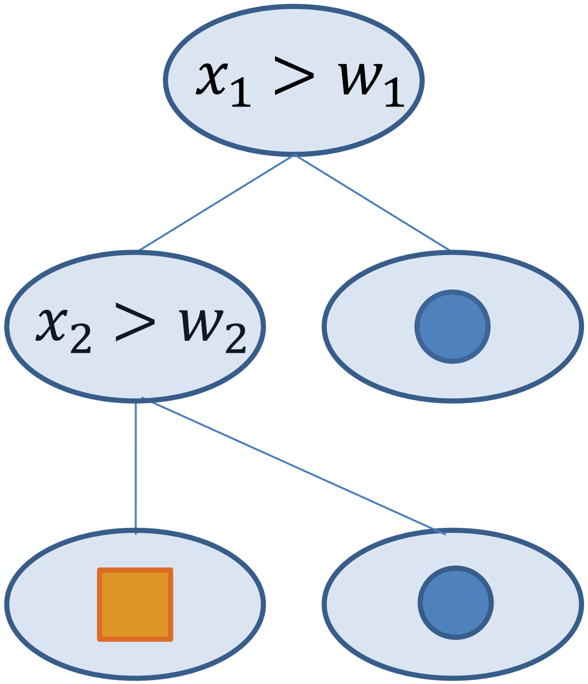
Decision Tree Classification Example

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

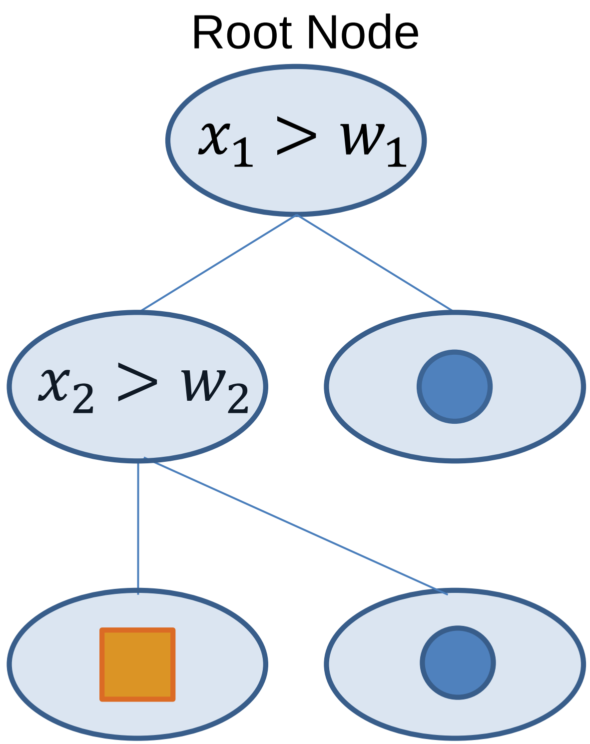


Questions?

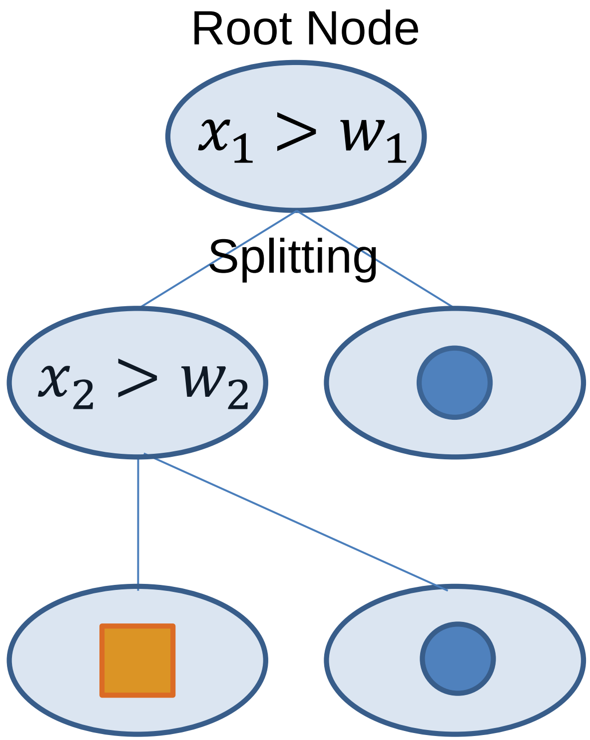
Basic Terminologies



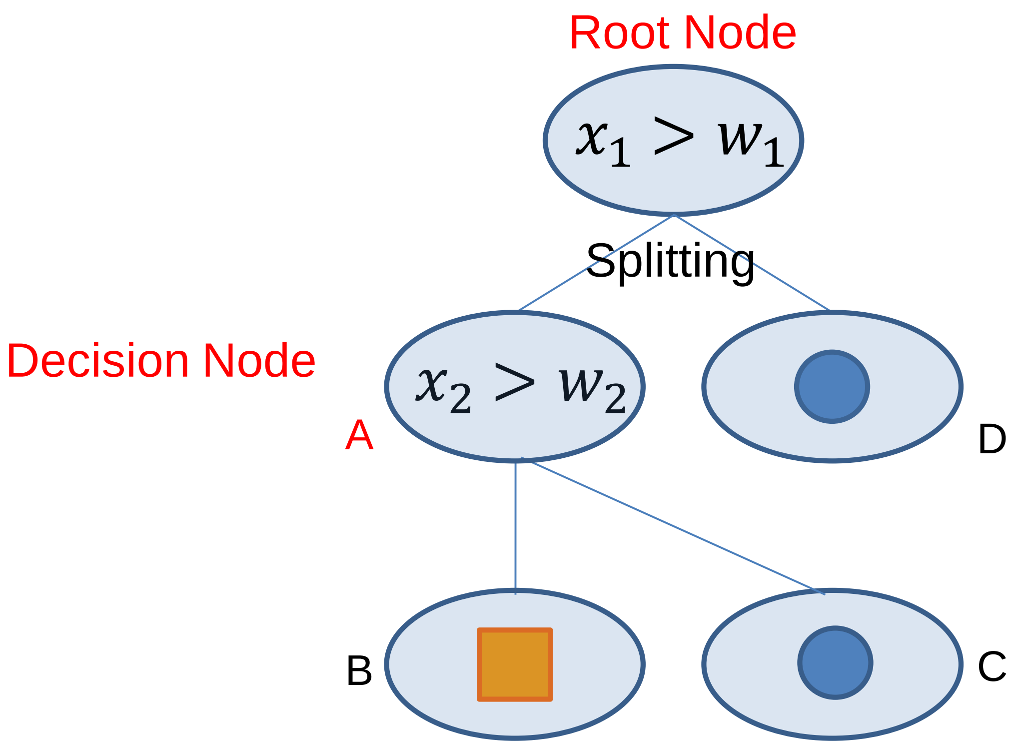
Basic Terminologies



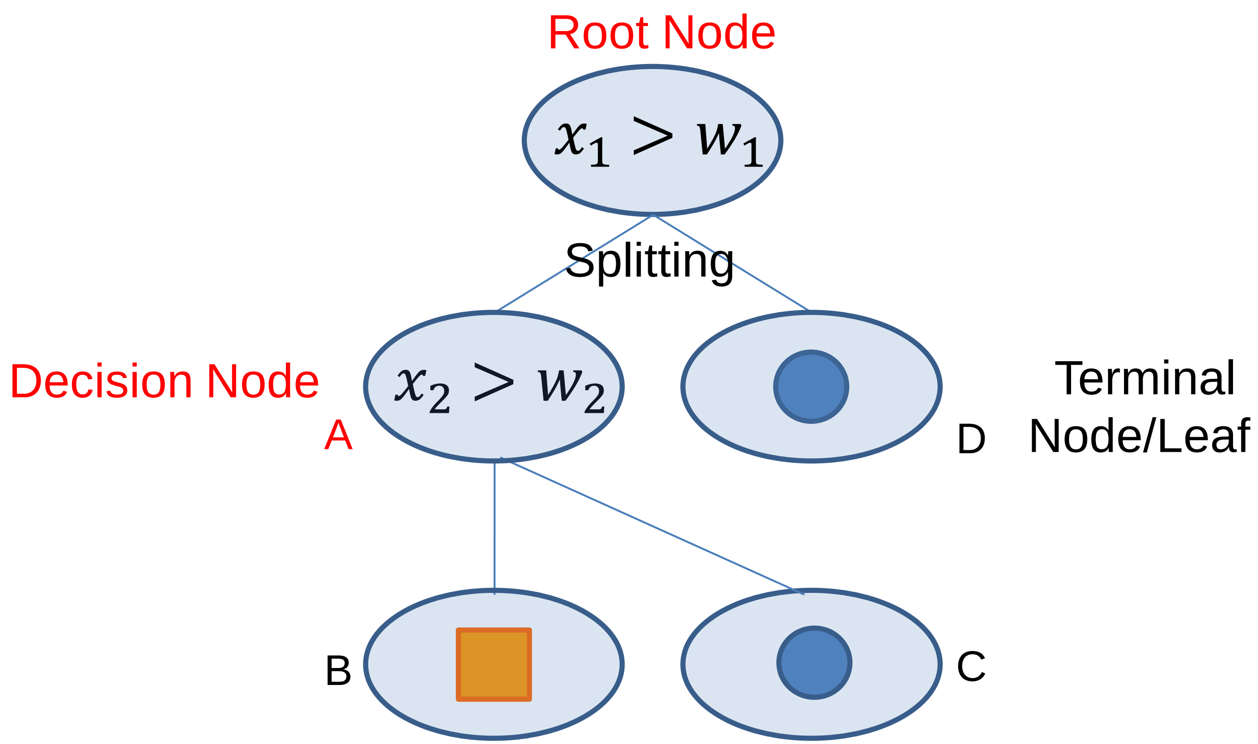
Basic Terminologies



Basic Terminologies

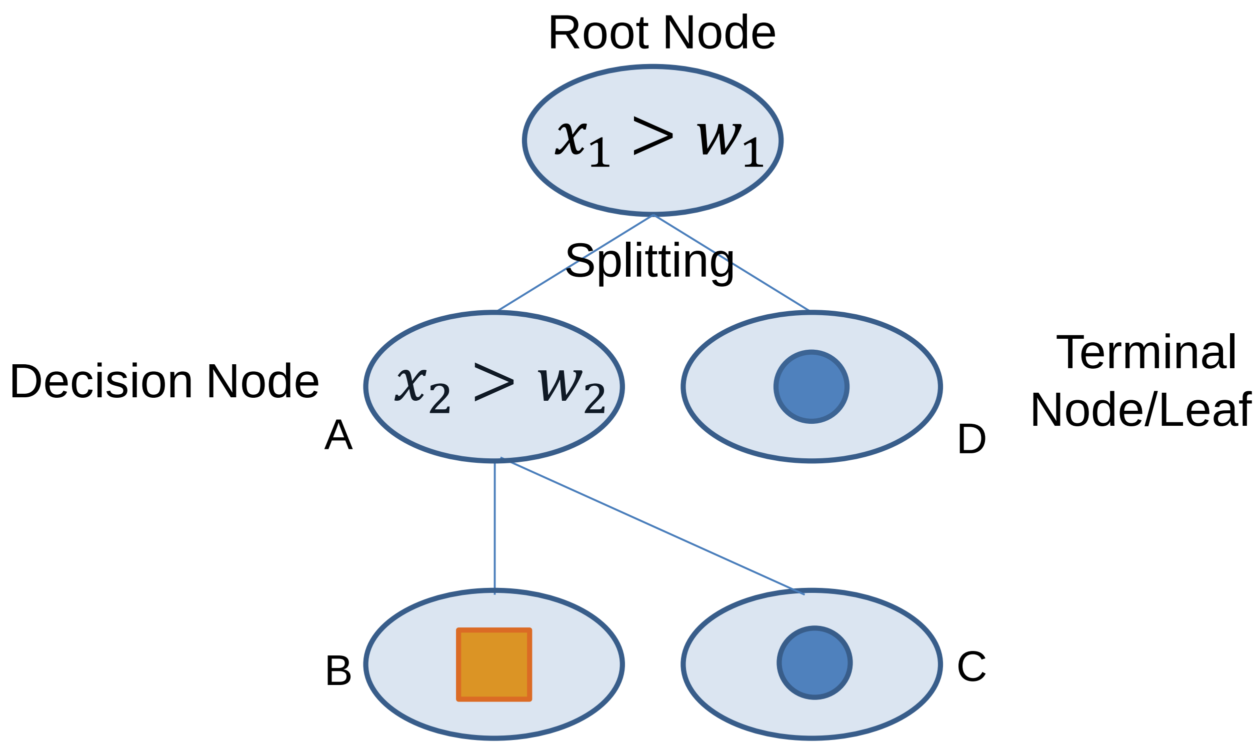


Basic Terminologies



B and C are Terminal Nodes or Leaves

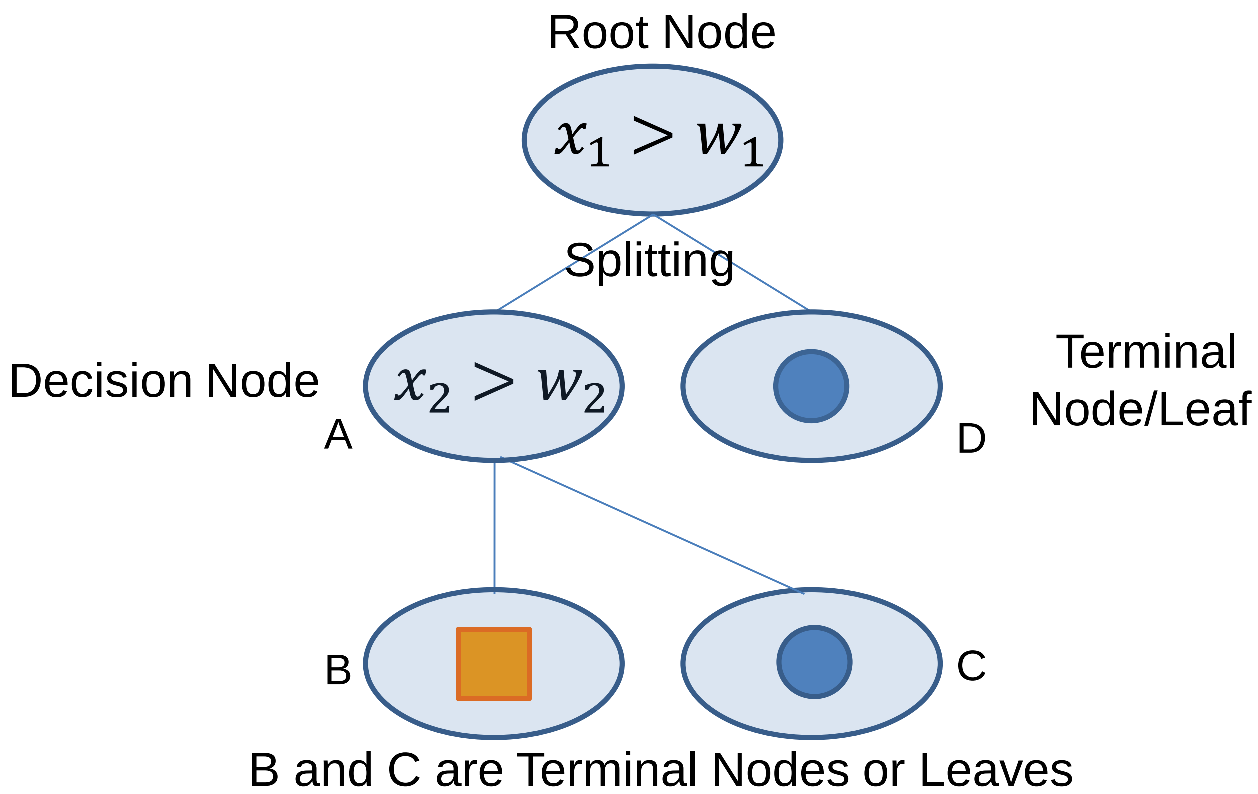
Basic Terminologies



B and C are Terminal Nodes or Leaves

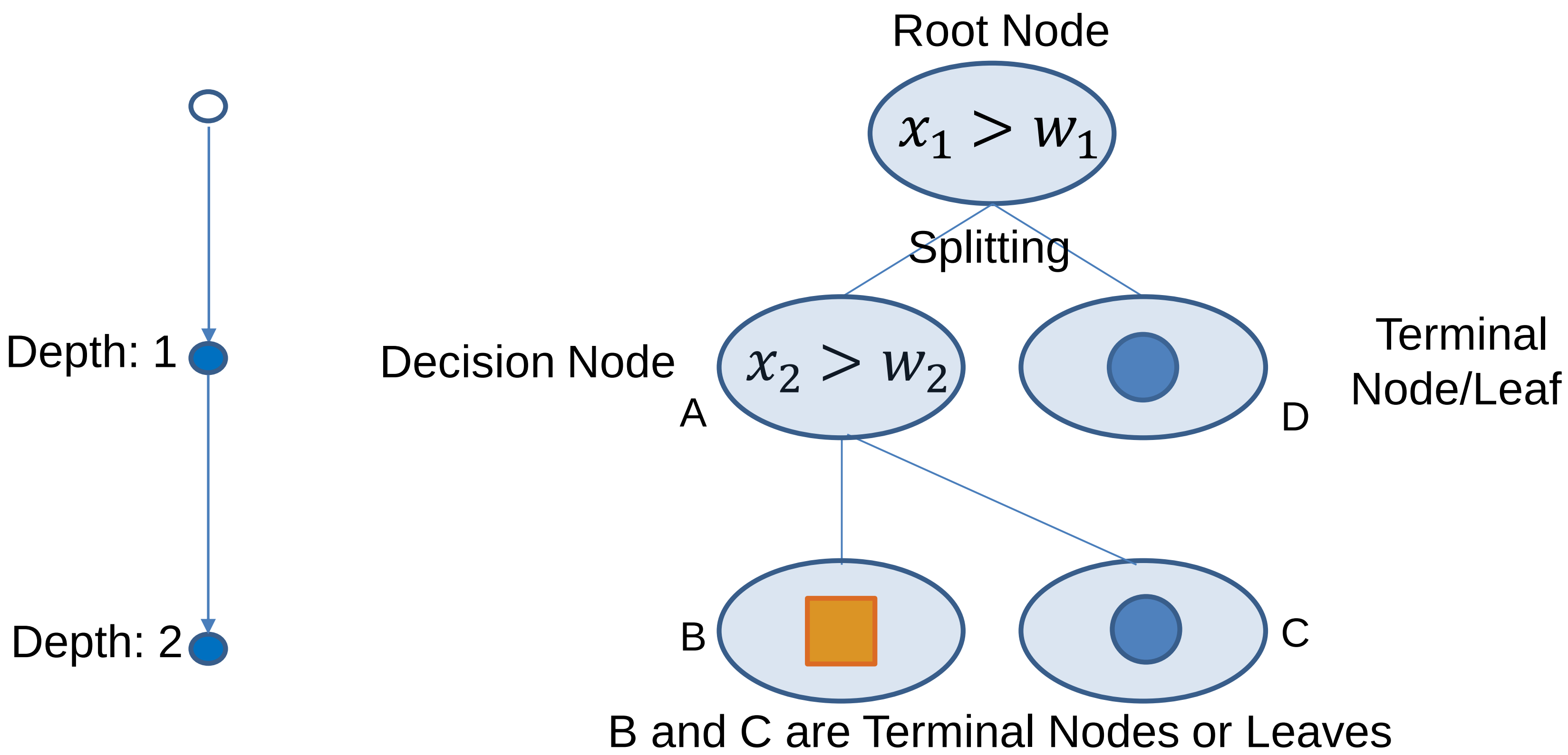
A-B-C forms a **sub-tree** or **branch**.

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.

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

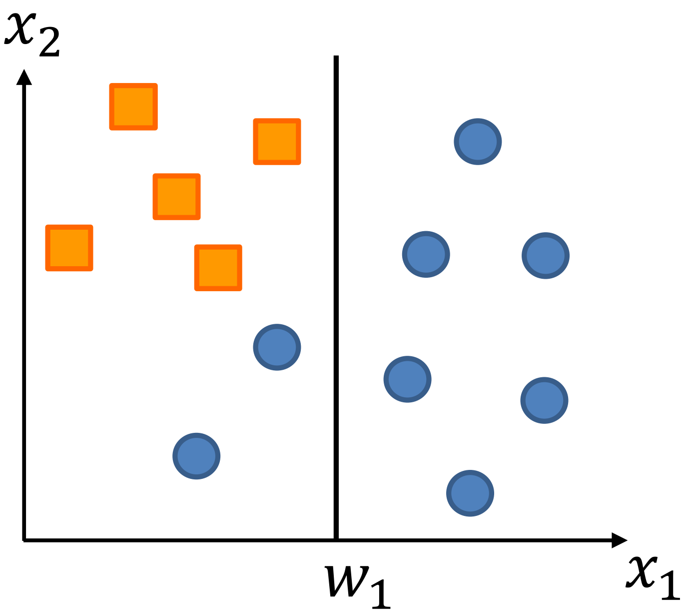
Building a Classification Decision Tree

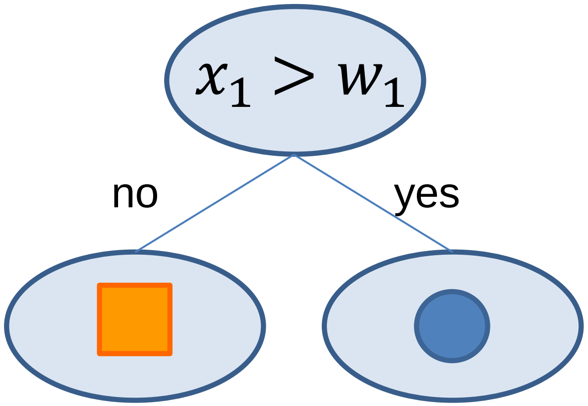
- Classification tree learning (or tree induction or tree growing) is the construction of a classification tree given training data
- For a given training set, there can be many trees with 0 training error, so we prefer less “complex” trees
- Complexity can be defined as number of nodes in the tree
-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

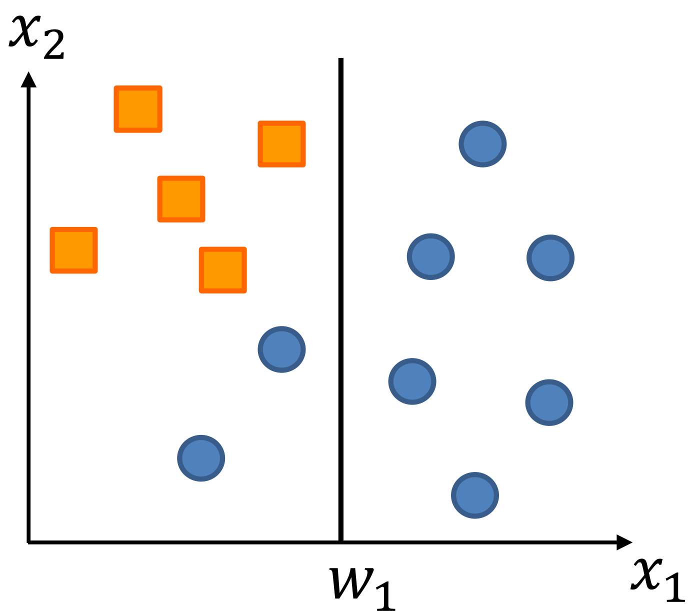
- Classification tree learning (or tree induction or tree growing) is the construction of a classification tree given training data
- For a given training set, there can be many trees with 0 training error, so we prefer less “complex” trees
- Complexity can be defined as number of nodes in the tree
-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

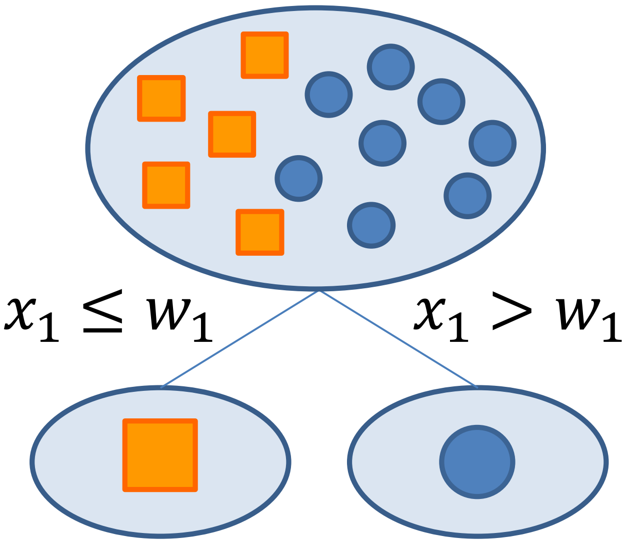
Node Impurity



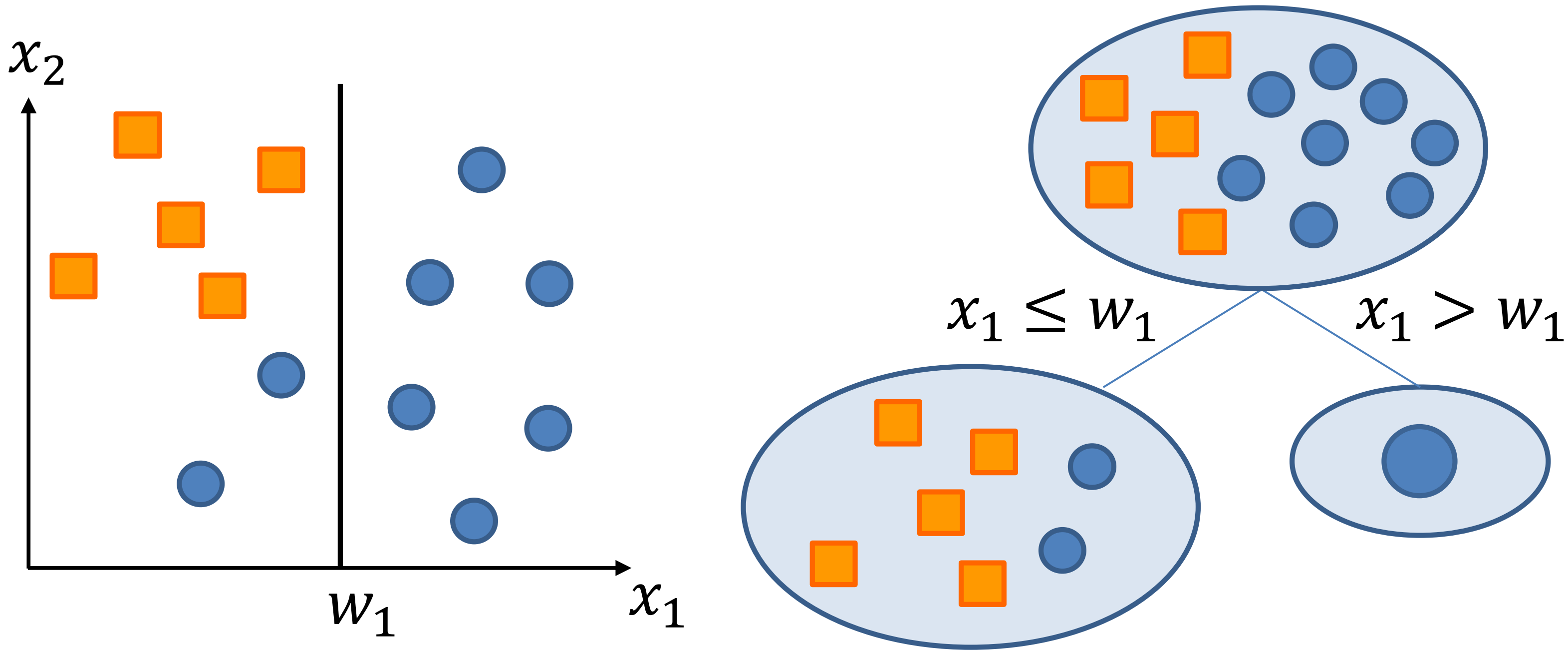


Node Impurity

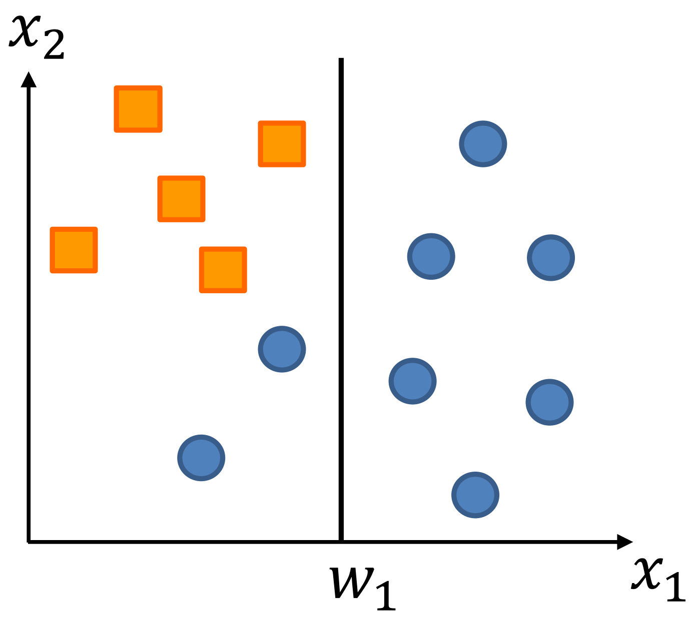


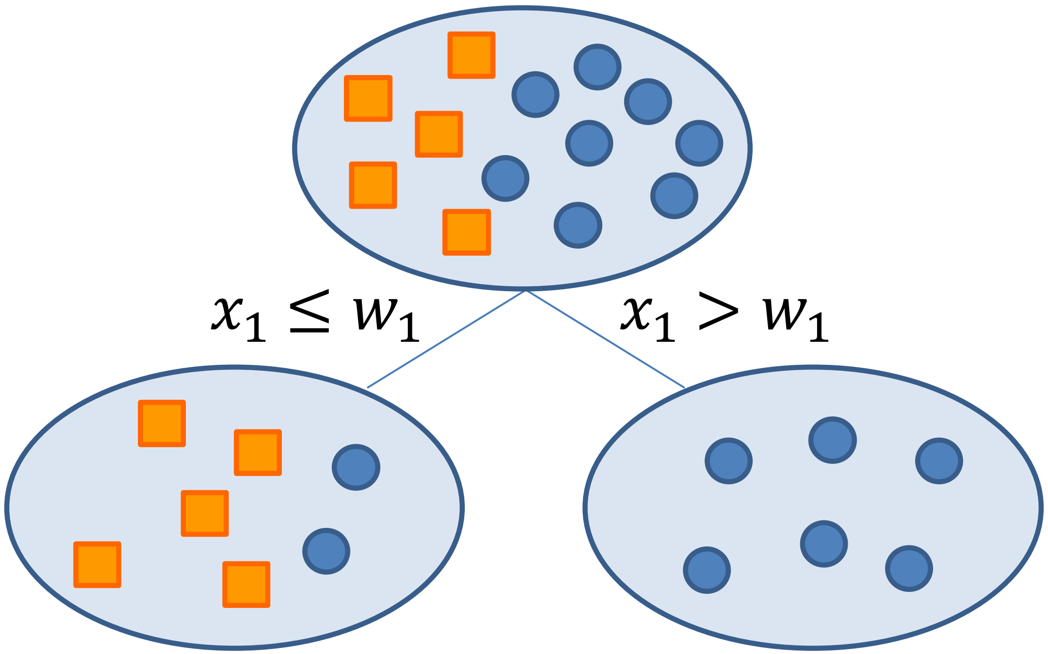


Node Impurity

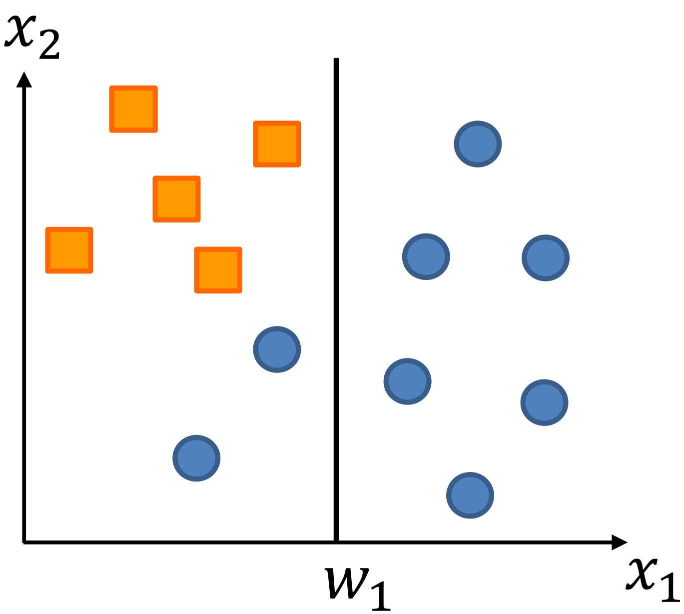


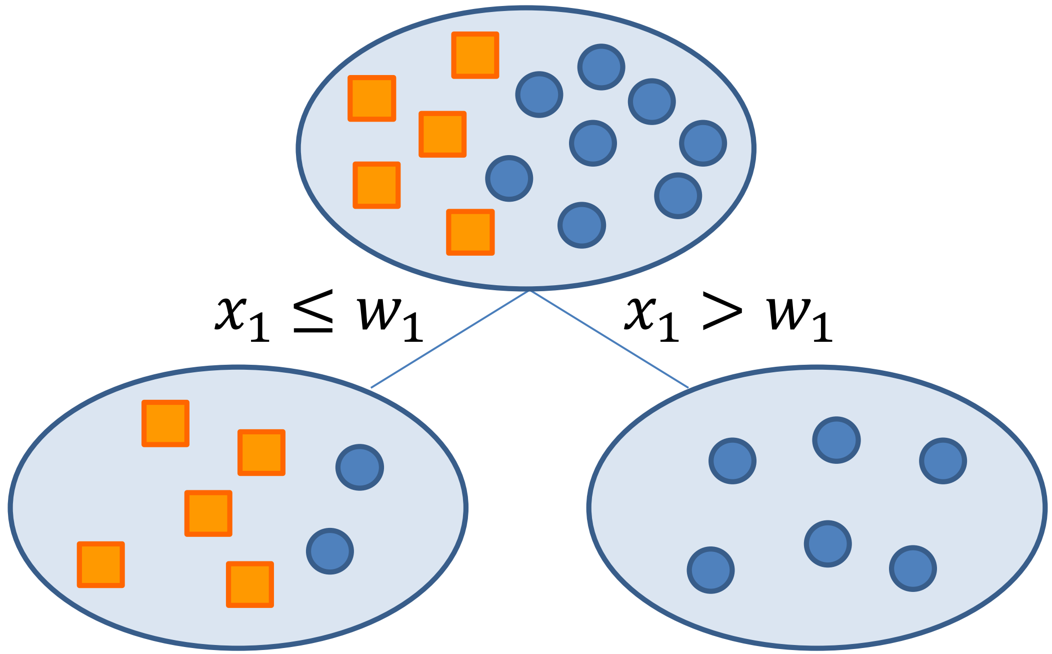
Node Impurity





Node Impurity

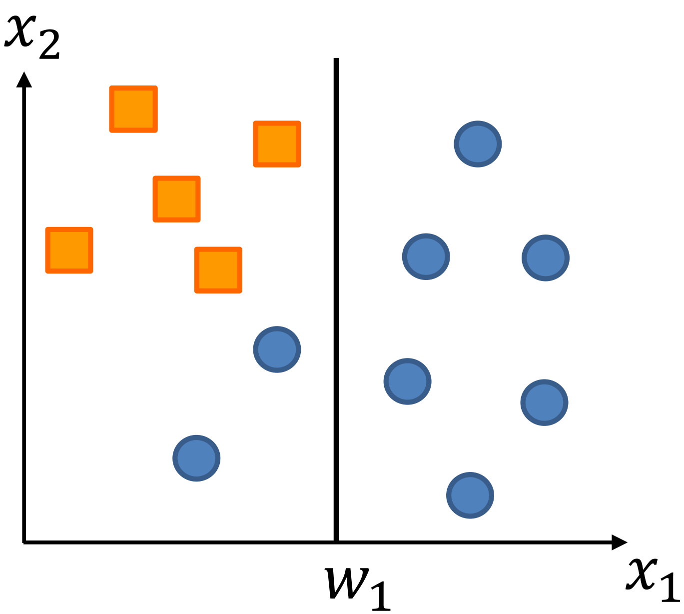


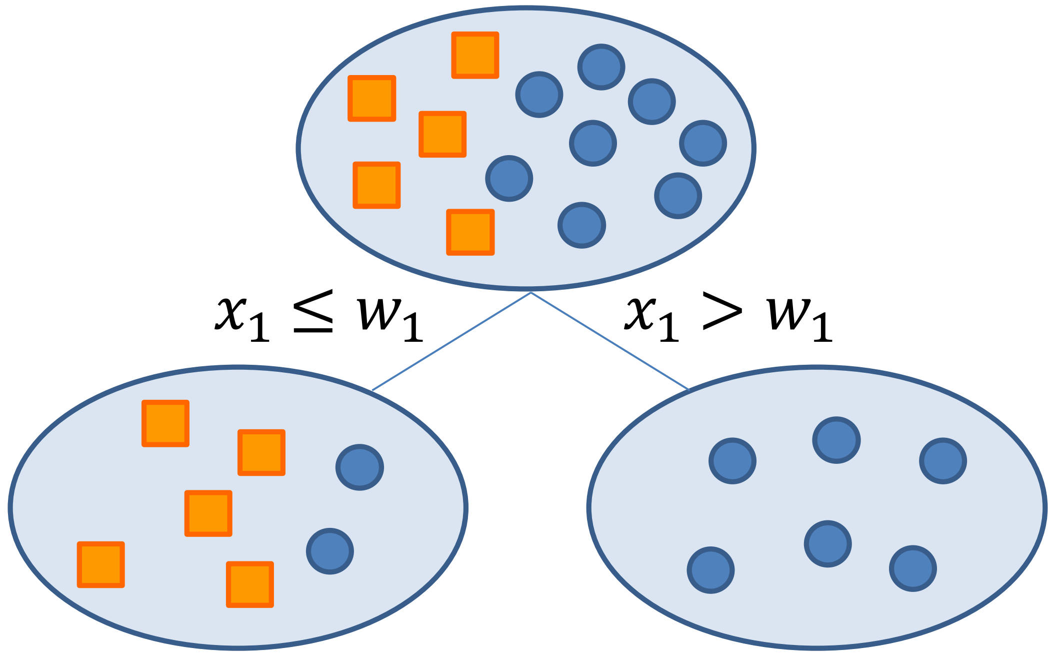


"Impure" because it contains ■ & ●

"Pure" because it only contains ●

Node Impurity





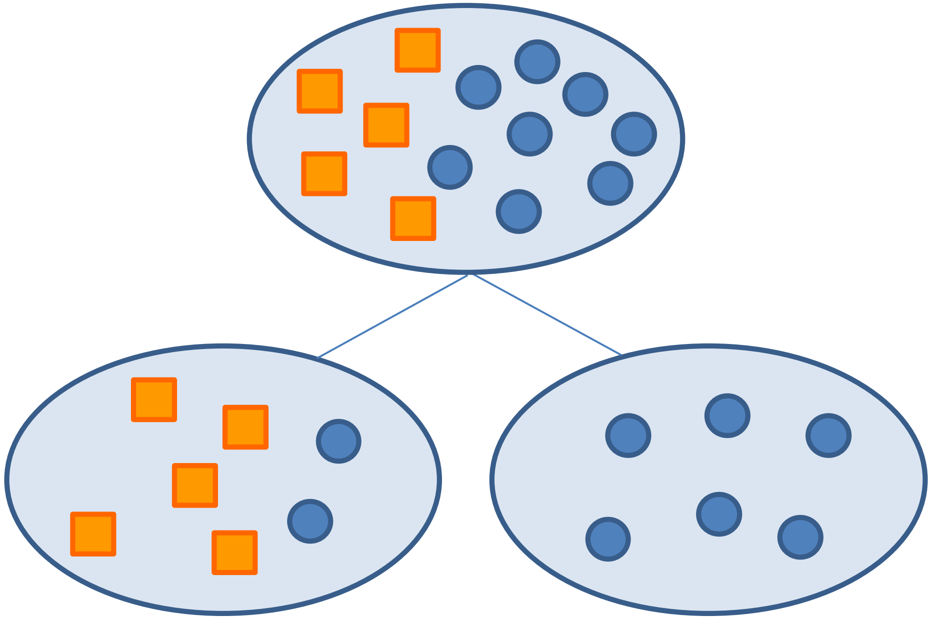
“Impure” because it contains &

“Pure” because it only contains

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

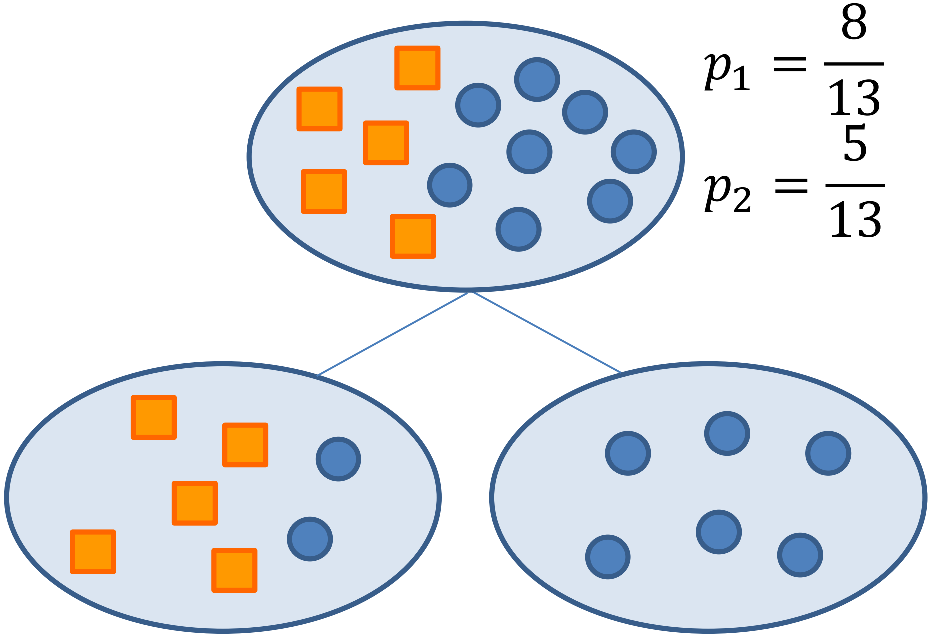
Node Impurity Measures

- 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

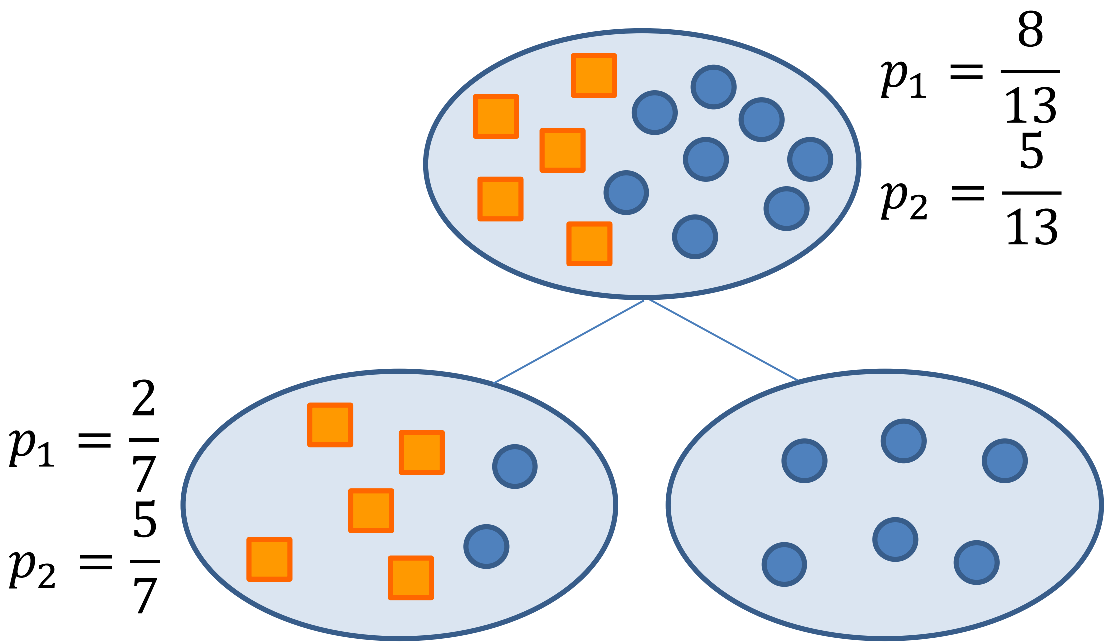


$$p_1 = \frac{8}{13}$$

$$p_2 = \frac{5}{13}$$

Node Impurity Measures

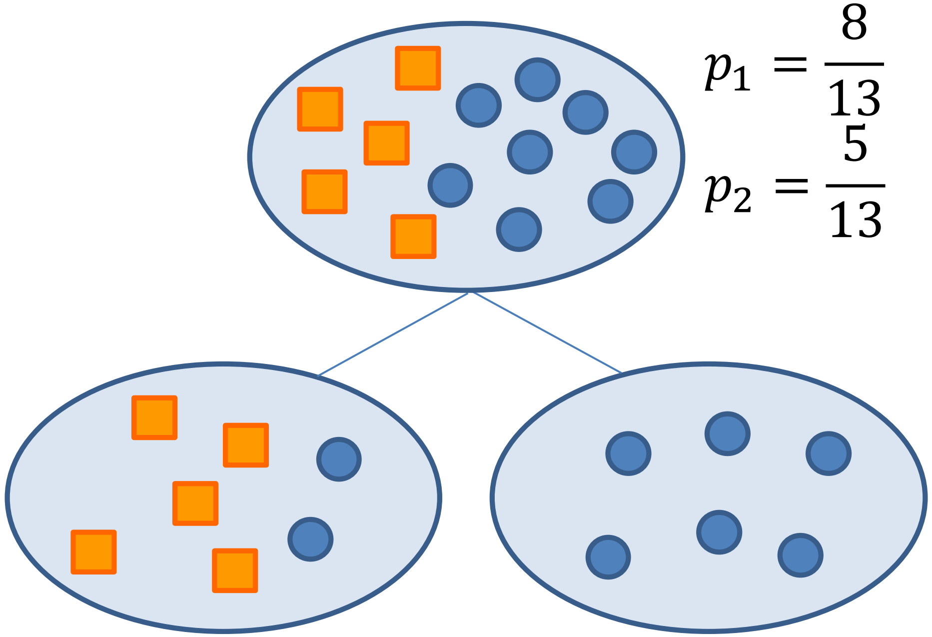
- Let ● be class 1 & ■ be class 2
- For particular node m, let p_i be the fraction (or probability) of data samples in node m belonging to class i



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

$$p_1 = \frac{8}{13}$$
$$p_2 = \frac{5}{13}$$

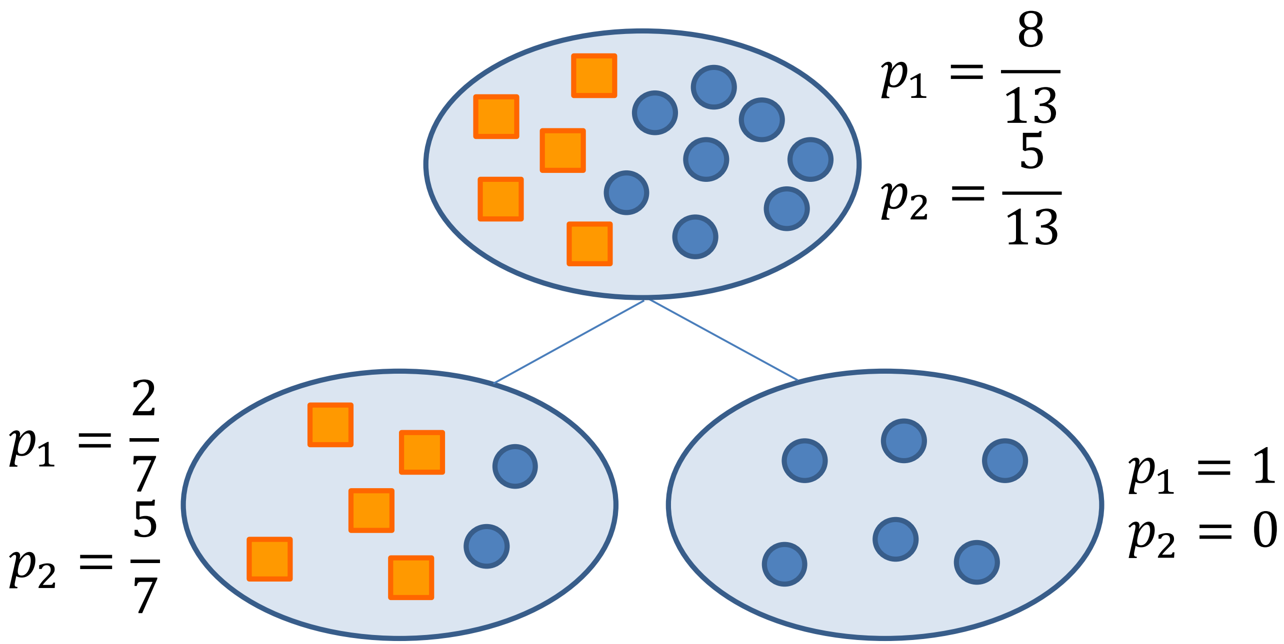


$$p_1 = \frac{2}{7}$$
$$p_2 = \frac{5}{7}$$

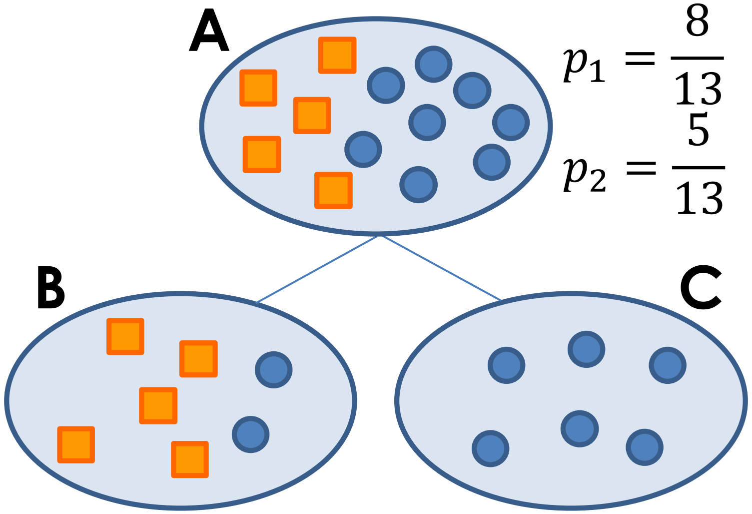
$$p_1 = 1$$
$$p_2 = 0$$

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
- 3 impurity measures: Gini, entropy, misclassification rate



Gini Impurity



$p_1 = \frac{8}{13}$

$p_2 = \frac{5}{13}$

$p_1 = \frac{2}{7}$

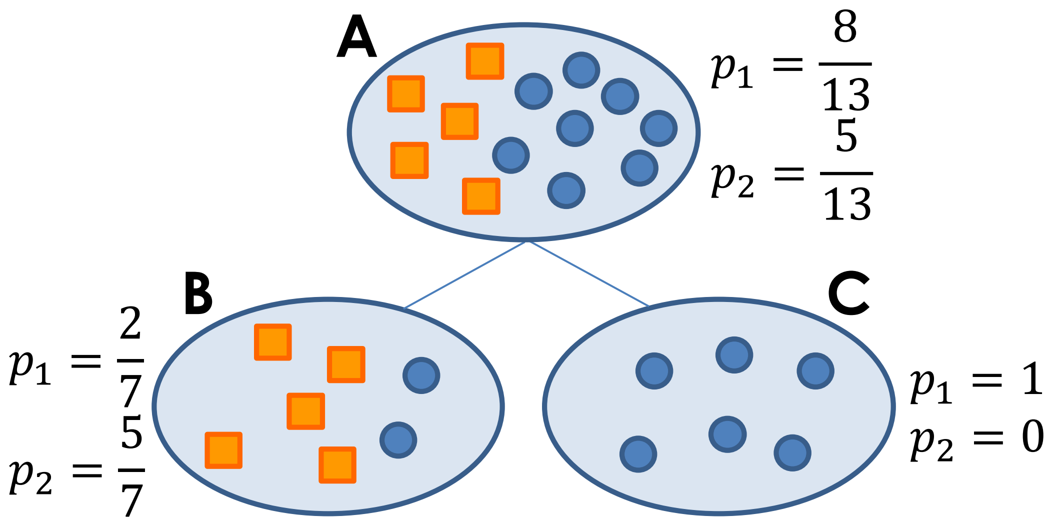
$p_2 = \frac{5}{7}$

$p_1 = 1$

$p_2 = 0$

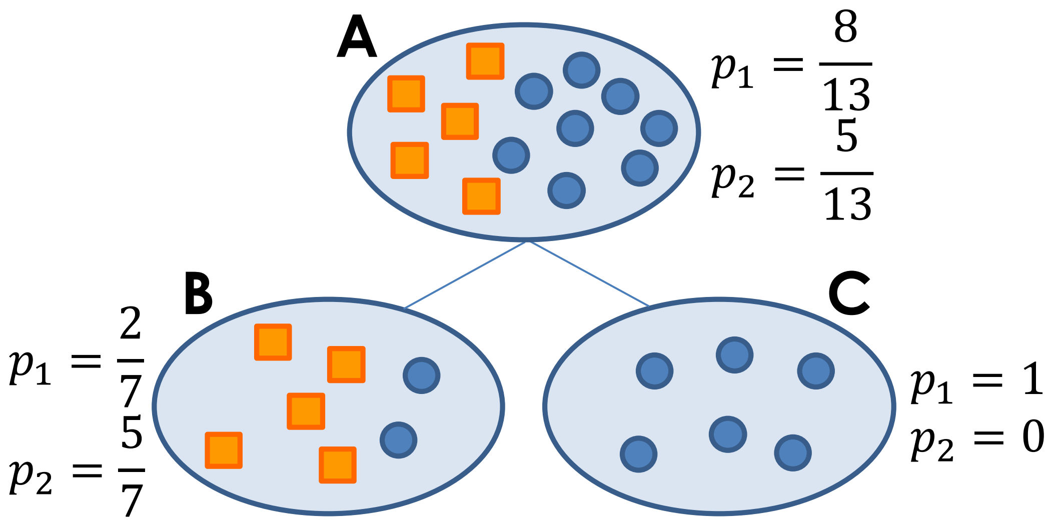
Gini Impurity

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



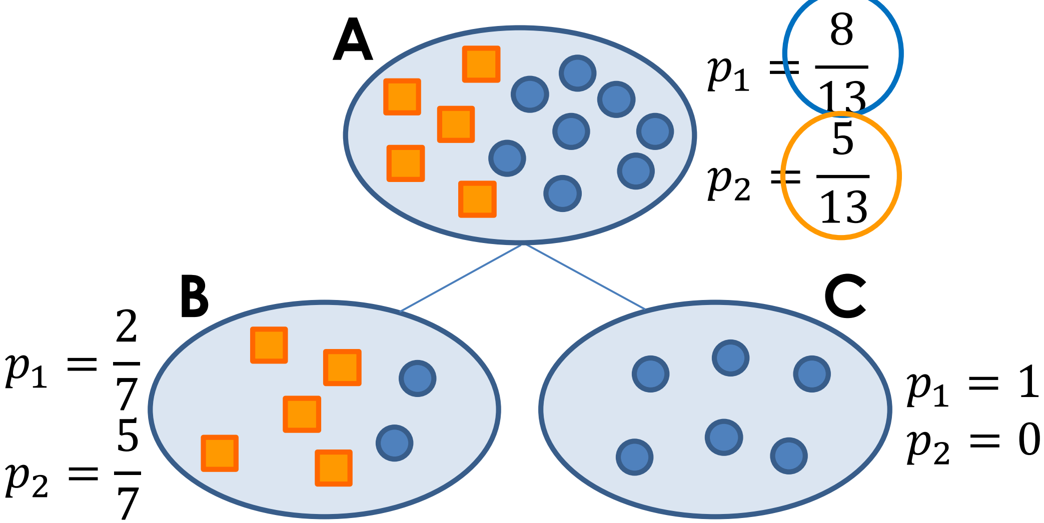
Gini Impurity

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



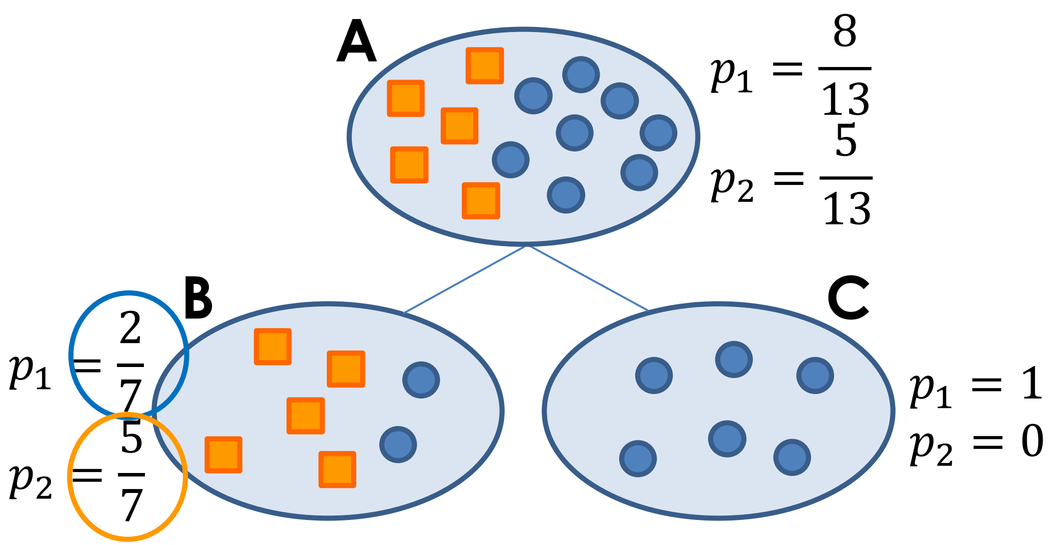
Gini Impurity

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



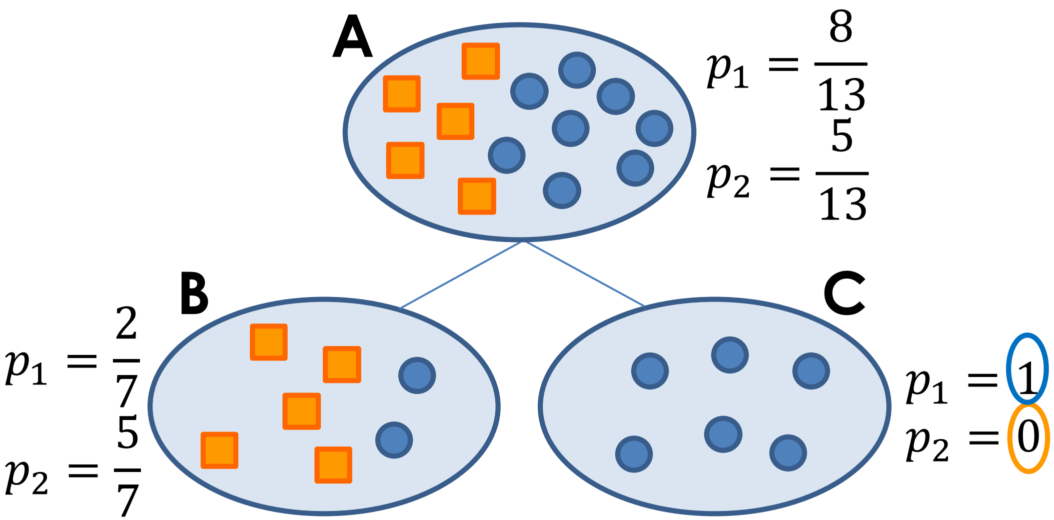
Gini Impurity

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



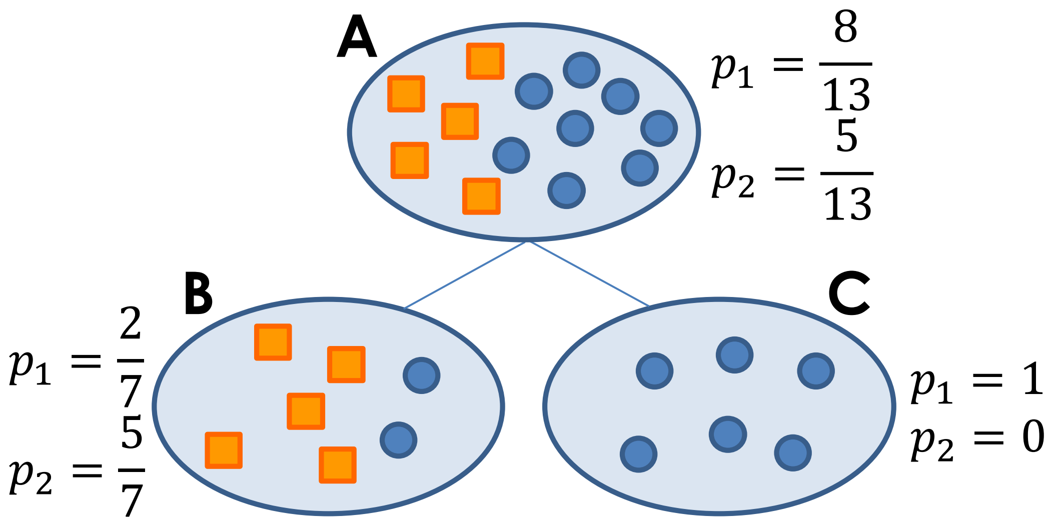
Gini Impurity

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



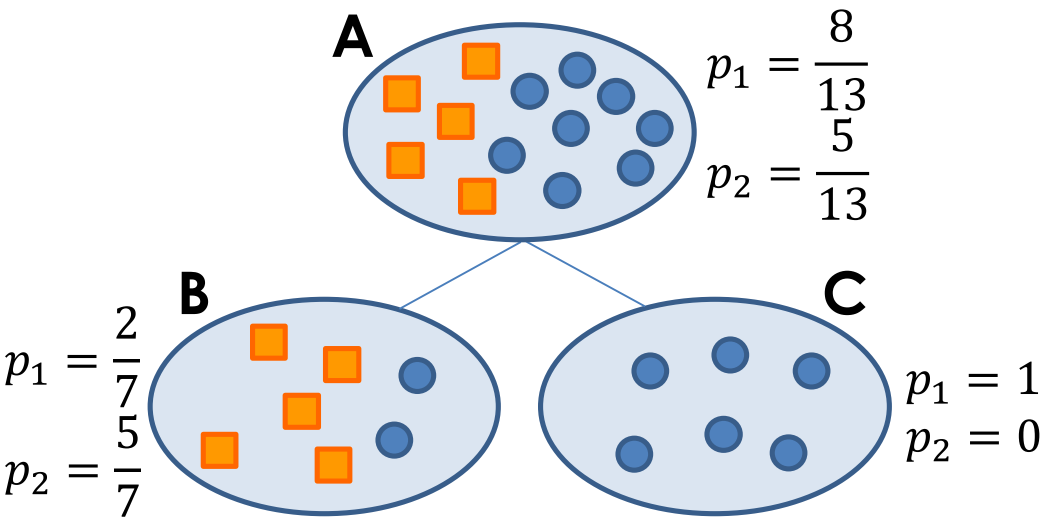
Gini Impurity

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



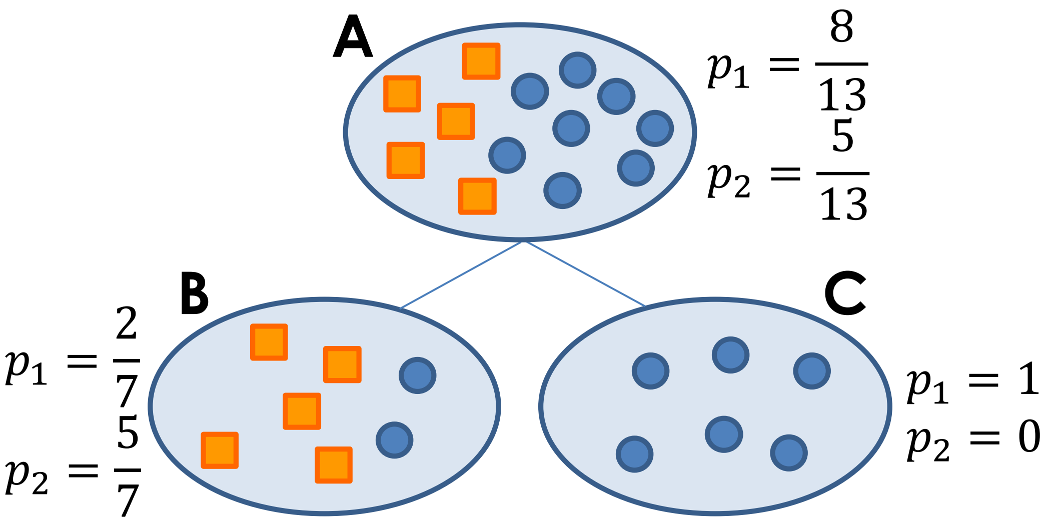
Gini Impurity

- Let K = # classes, define $Q_m = 1 - \Sigma_{i=1}^{K} p_i^2 = 1 - p_1^2 - p_2^2$
- 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 x Q_B + fraction of data samples in node C x Q_C
  - $\left(\frac{7}{13}\right) \times 0.4082 + \left(\frac{6}{13}\right) \times 0 = 0.2198$



Gini Impurity

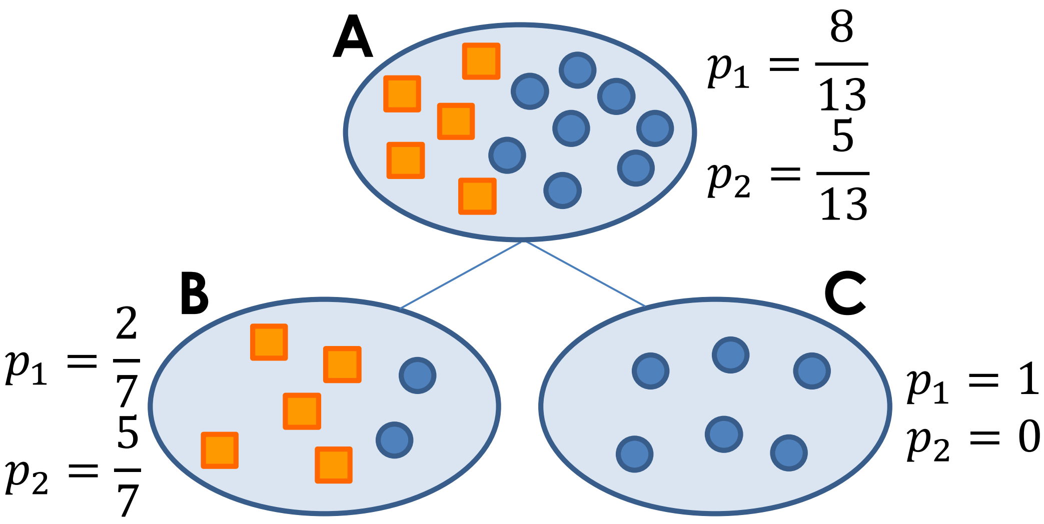
- Let K = # classes, define $Q_m = 1 - \Sigma_{i=1}^{K} p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_\text{A} = 1 - (8/13)^2 - (5/13)^2 = 0.4734$
- Node B: $Q_\text{B} = 1 - (2/7)^2 - (5/7)^2 = 0.4082$
- Node C: $Q_\text{C} = 1 - 1^2 - 0^2 = 0$
- Overall Gini (depth 1) = fraction of data samples in node B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



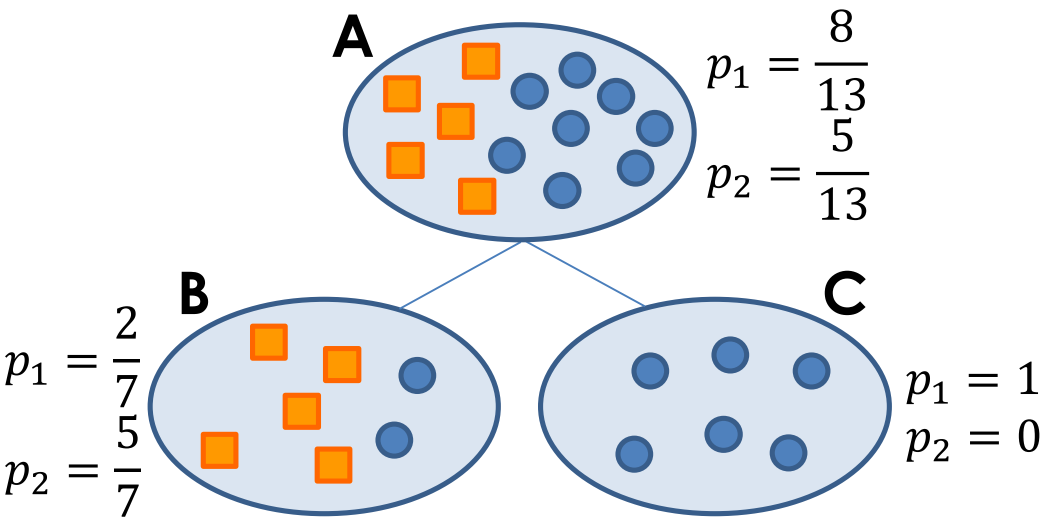
Gini Impurity

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

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

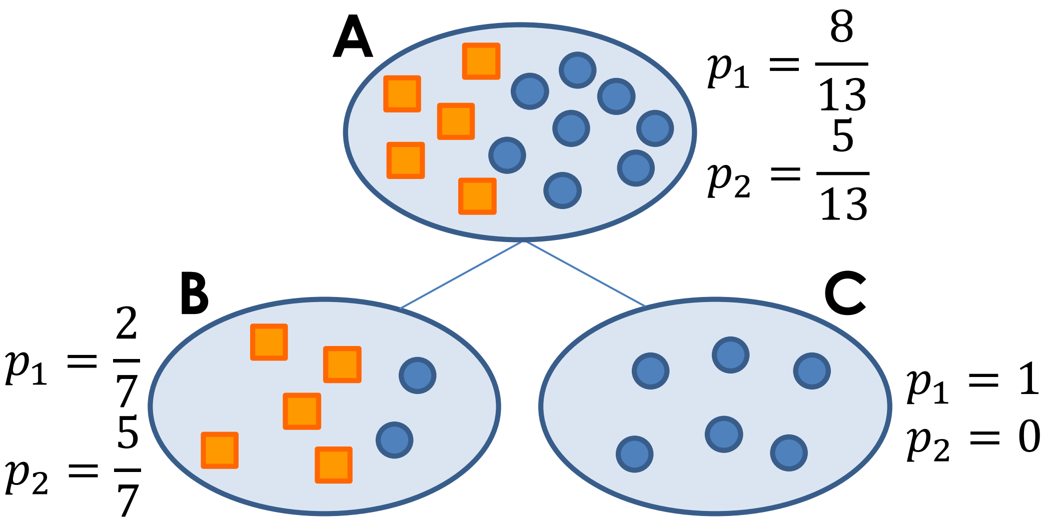


Entropy



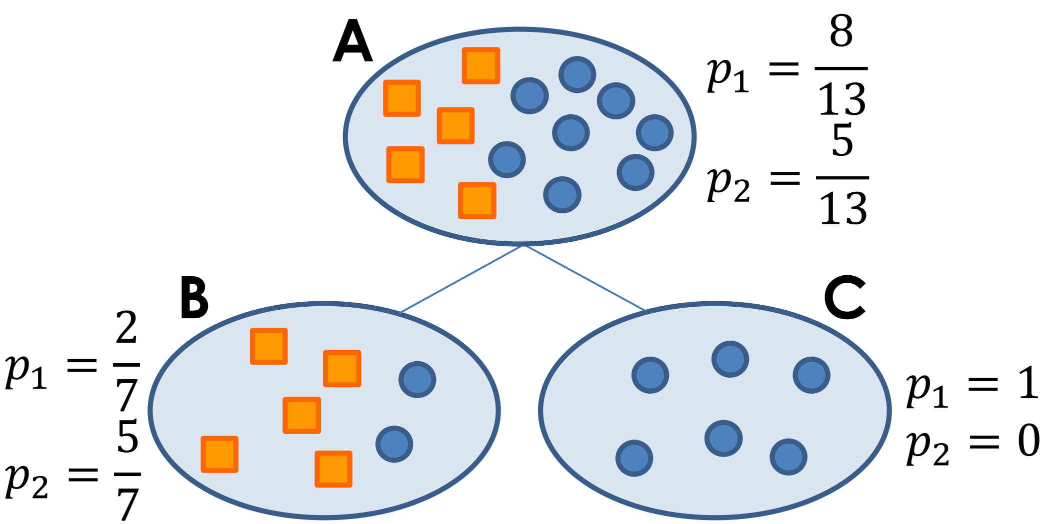
Entropy

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



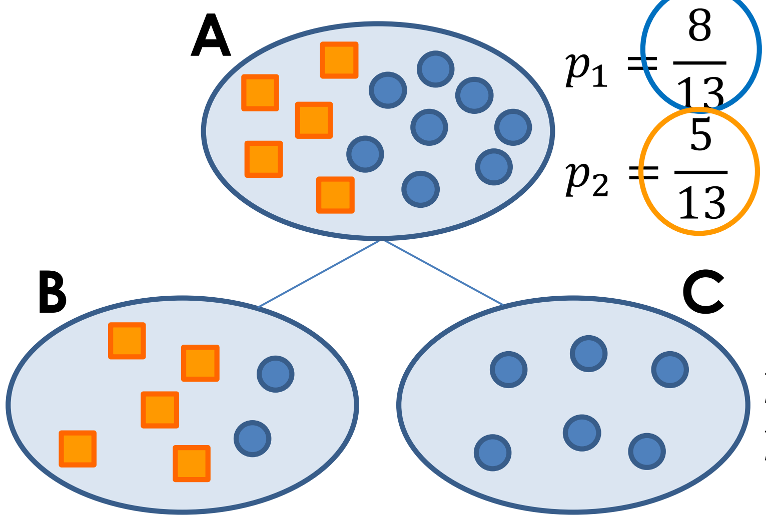
Entropy

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



Entropy

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



$p_1 = \frac{8}{13}$

$p_2 = \frac{5}{13}$

$p_1 = \frac{2}{7}$

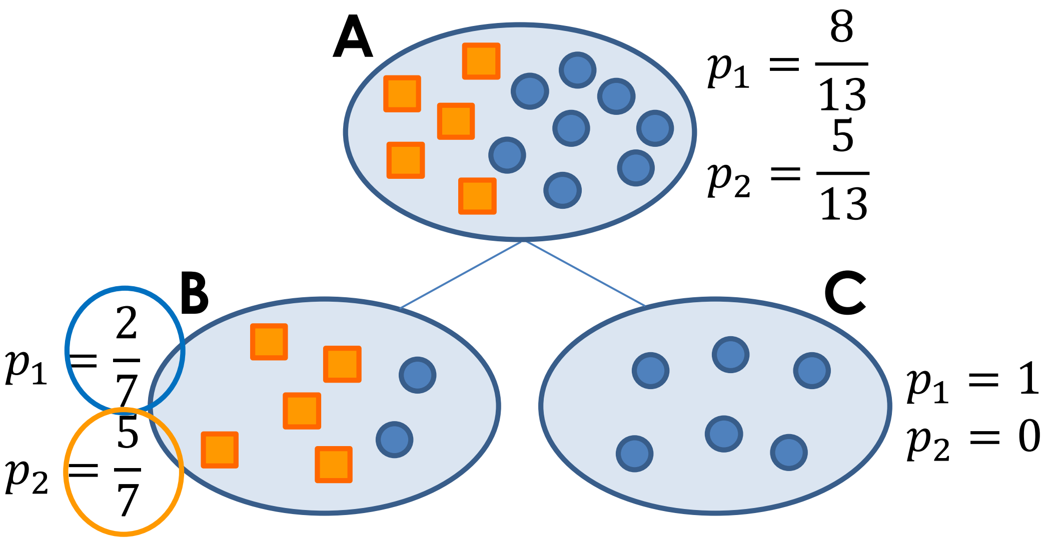
$p_2 = \frac{5}{7}$

$p_1 = 1$

$p_2 = 0$

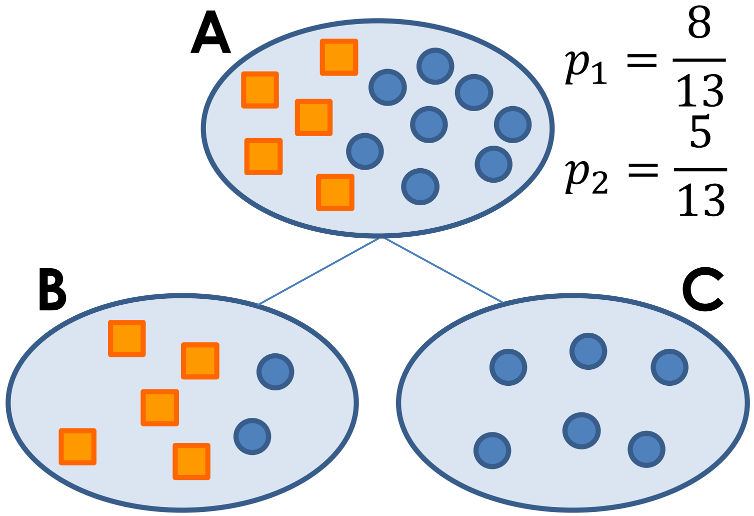
Entropy

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



Entropy

- Let K = # classes, define $Q_m = -\Sigma_{i=1}^{K} p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- 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: $Q_C = -1\log_2 1 - 0\log_2 0 = 0$



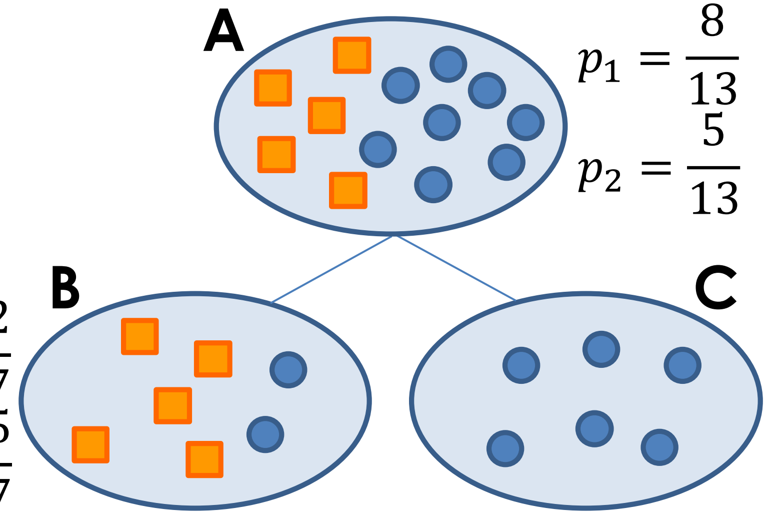
$p_1 = \frac{8}{13}$
$p_2 = \frac{5}{13}$

$p_1 = \frac{2}{7}$
$p_2 = \frac{5}{7}$

$p_1 = 1$
$p_2 = 0$

Entropy

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



$p_1 = \frac{8}{13}$

$p_2 = \frac{5}{13}$

$p_1 = \frac{2}{7}$

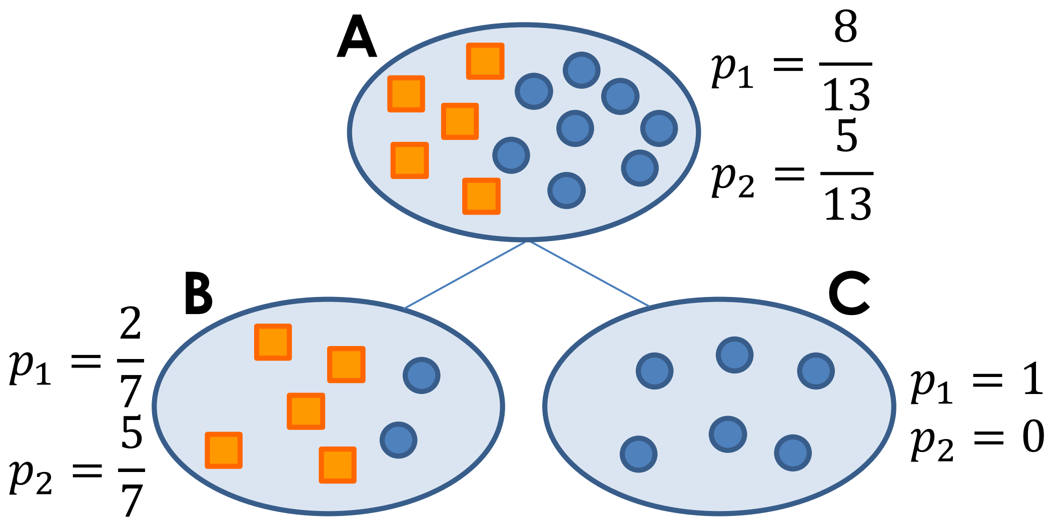
$p_2 = \frac{5}{7}$

$p_1 = 1$

$p_2 = 0$

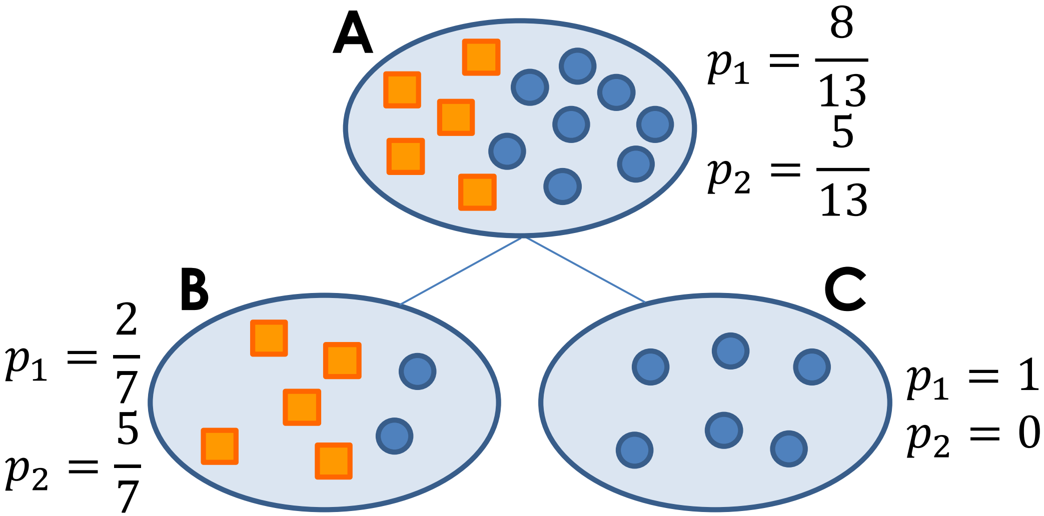
Entropy

- Let K = # classes, define $Q_m = -\Sigma_{i=1}^{K} p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_\mathrm{A} = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$
- Node B: $Q_\mathrm{B} = -(2/7)\log_2(2/7) - (5/7)\log_2(5/7) = 0.8631$
- Node C: $Q_\mathrm{C} = -1\log_2 1 - 0\log_2 0 = 0$
- Overall entropy (depth 1) = fraction of data samples in node B x Q_B + fraction of data samples in node C x 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

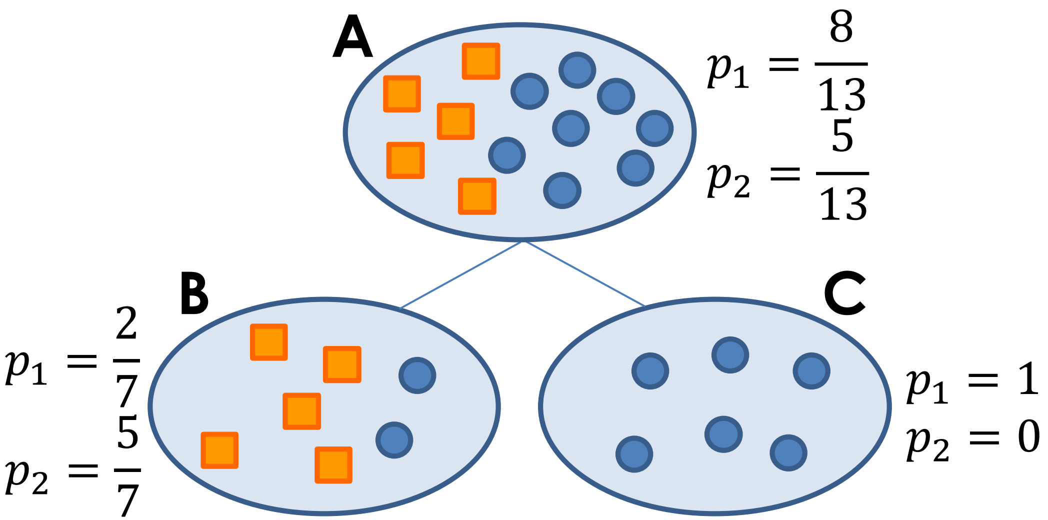


Entropy

- Let K = # classes, define $Q_m = -\Sigma_{i=1}^{K} p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$
- Node A: $Q_\mathrm{A} = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$
- Node B: $Q_\mathrm{B} = -(2/7)\log_2(2/7) - (5/7)\log_2(5/7) = 0.8631$
- Node C: $Q_\mathrm{C} = -1\log_2 1 - 0\log_2 0 = 0$
- Overall entropy (depth 1) = fraction of data samples in node B x Q_B + fraction of data samples in node C x 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: $Q_m = -\Sigma_{i=1}^{K} p_i \log_2 p_i$

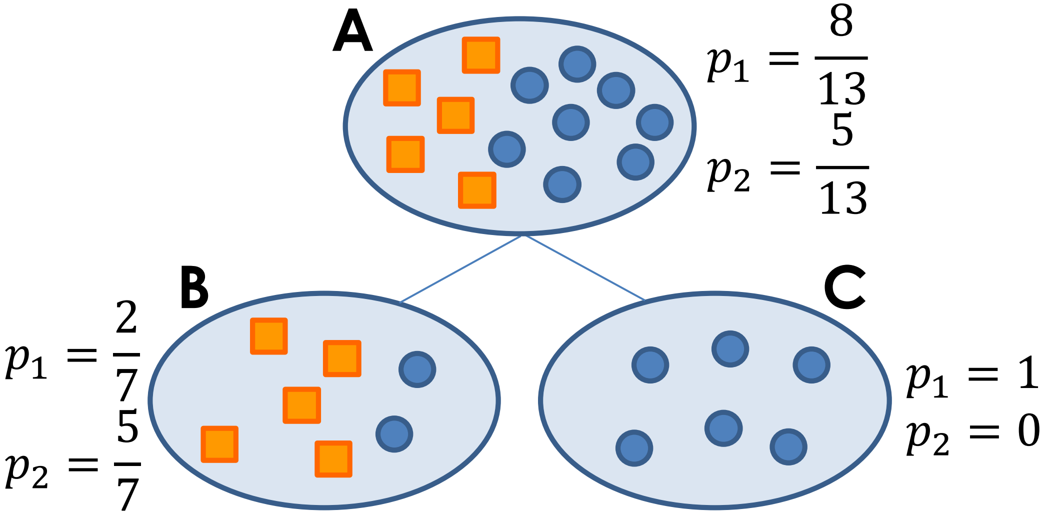


Misclassification rate



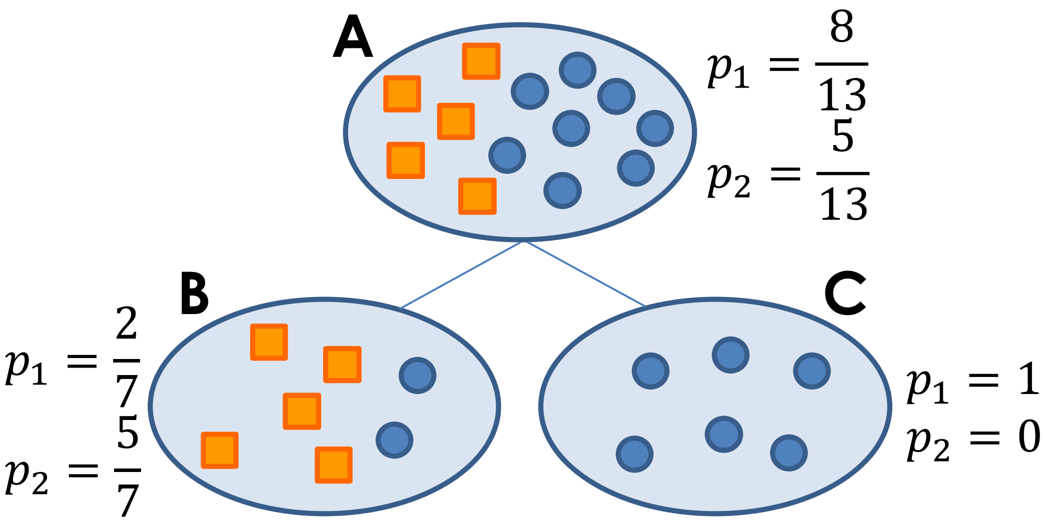
Misclassification rate

- Let K = # classes, define $Q_m = 1 - \max_i p_i$



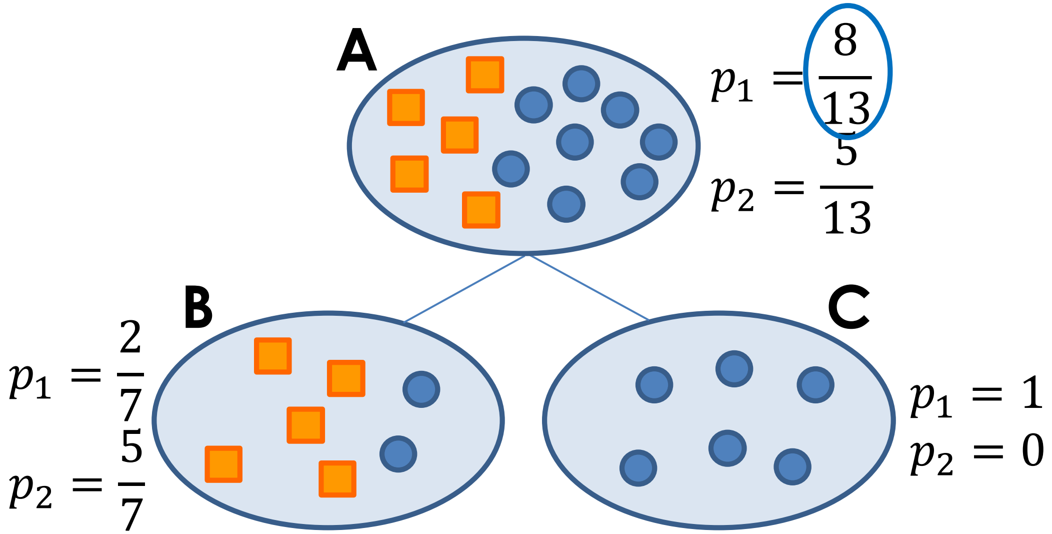
Misclassification rate

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



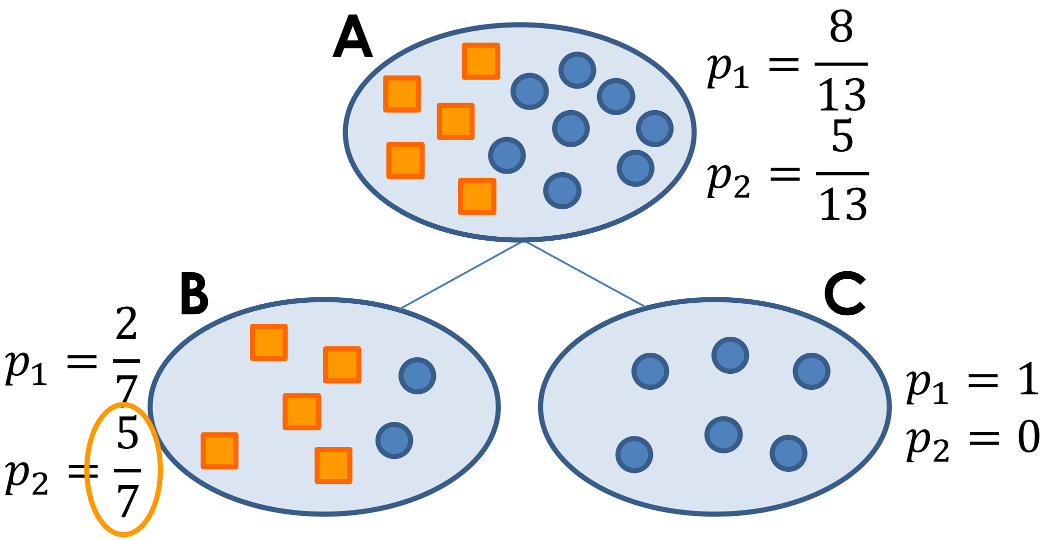
Misclassification rate

- Let K = # 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 => $Q_A = 1 - 8/13 = 5/13$



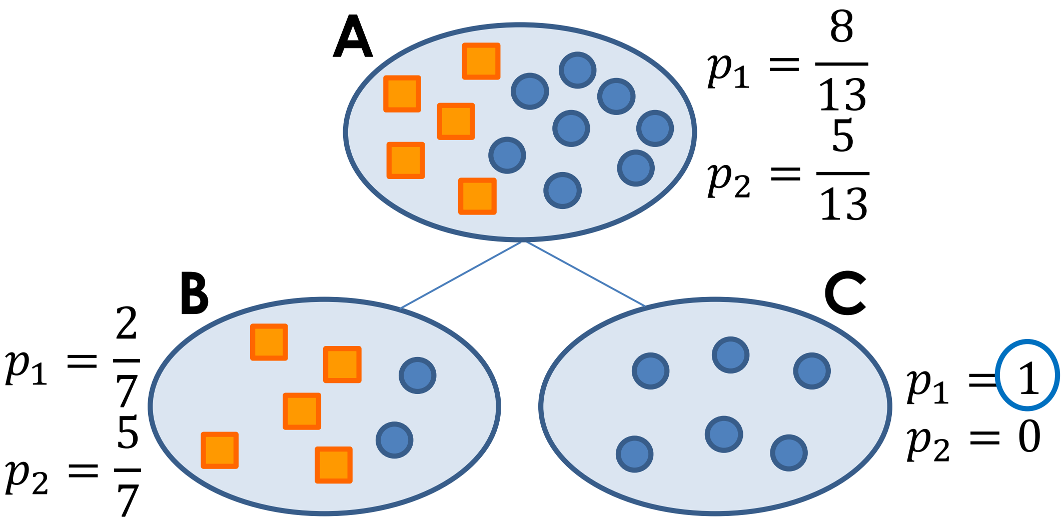
Misclassification rate

- Let K = # 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 => $Q_\mathrm{A} = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 => $Q_\mathrm{B} = 1 - 5/7 = 2/7$



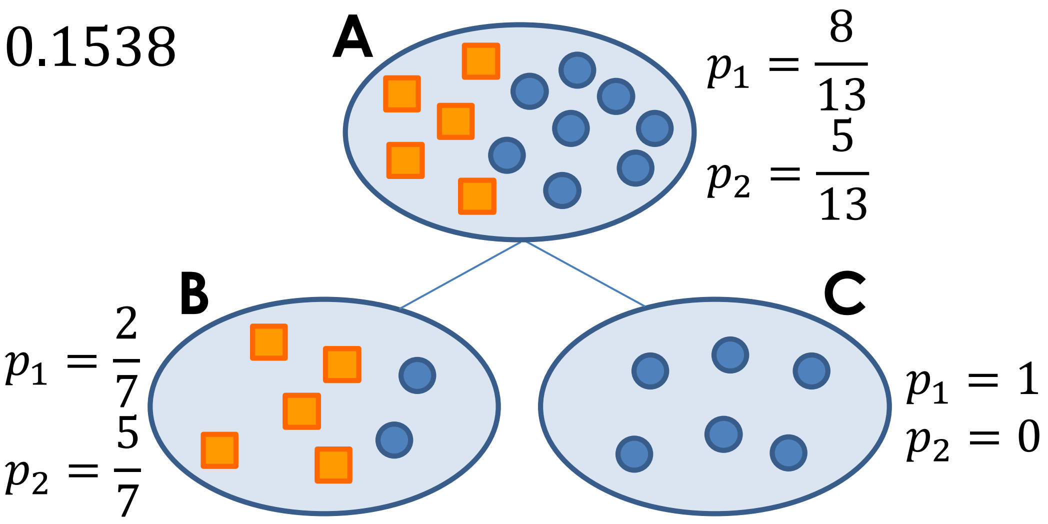
Misclassification rate

- Let K = # 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 => $Q_\mathrm{A} = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 => $Q_\mathrm{B} = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = class 1 => $Q_\mathrm{C} = 1 - 1 = 0$



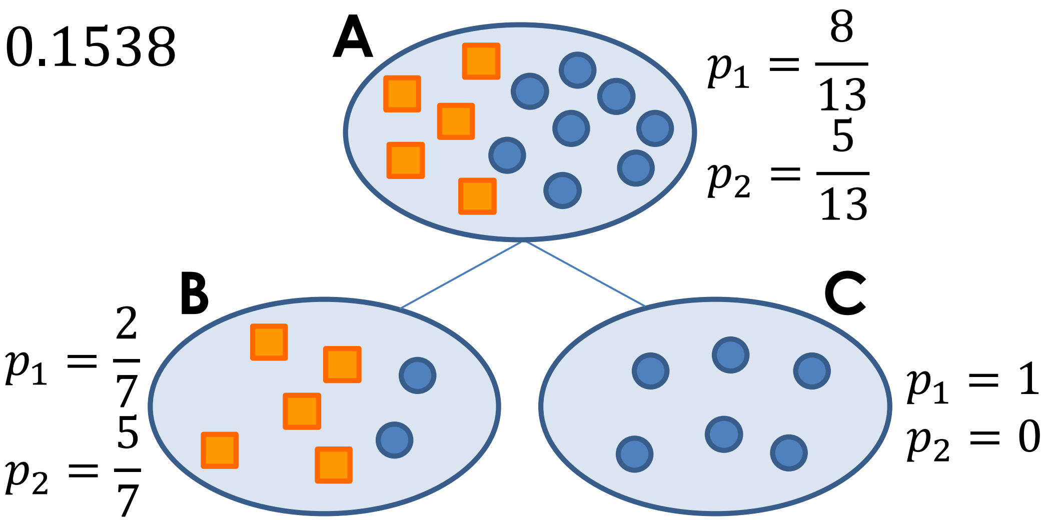
Misclassification rate

- Let K = # 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 => $Q_\mathrm{A} = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 => $Q_\mathrm{B} = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = class 1 => $Q_\mathrm{C} = 1 - 1 = 0$
- Overall misclassification rate (depth 1) = fraction of data samples in node B x Q_B + fraction of data samples in node C x Q_C
    - $\left(\frac{7}{13}\right) \times \left(\frac{2}{7}\right) + \left(\frac{6}{13}\right) \times 0 = 0.1538$



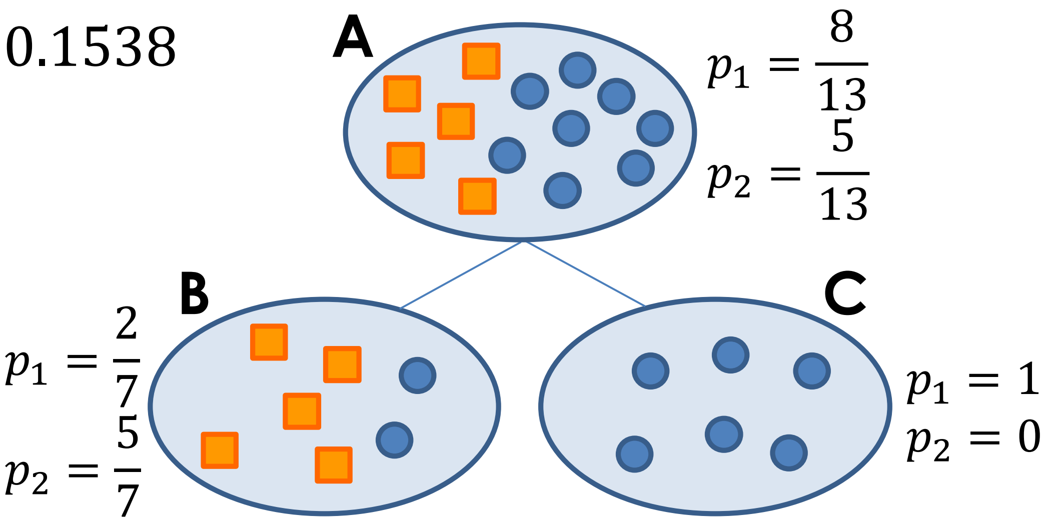
Misclassification rate

- Let K = # 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 => $Q_\mathrm{A} = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 => $Q_\mathrm{B} = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = class 1 => $Q_\mathrm{C} = 1 - 1 = 0$
- Overall misclassification rate (depth 1) = fraction of data samples in node B x Q_B + fraction of data samples in node C x 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



Misclassification rate

- Let K = # 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 => $Q_\mathrm{A} = 1 - 8/13 = 5/13$
- Node B: $p_2 > p_1$, so best classification = class 2 => $Q_\mathrm{B} = 1 - 5/7 = 2/7$
- Node C: $p_1 > p_2$, so best classification = class 1 => $Q_\mathrm{C} = 1 - 1 = 0$
- Overall misclassification rate (depth 1) = fraction of data samples in node B x Q_B + fraction of data samples in node C x 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: $Q_m = 1 - \max_i p_i$



Questions?

Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q, parameter max_depth & training set

Output: 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

Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q, parameter max_depth & training set

Output: Tree

```
root ← all training samples
for d ← 1 to max_depth do

return tree
```

Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q, parameter max_depth & training set

Output: Tree

```
root ← all training samples
for d ← 1 to max_depth do
    for each leaf node m at depth d − 1 do


return tree
```

Classification Tree Learning

Algorithm: Classification Tree Learning

Input: Impurity measure Q, parameter max_depth & training set

Output: Tree

root $\leftarrow$ all training samples

for $d \leftarrow 1$ ***to*** max_depth **do**

    for *each leaf node* m *at depth* $d-1$ **do**

        Find best feature & best threshold, so splitting node m into two reduces the most impurity

return *tree*

Classification Tree Learning

Algorithm: Classification Tree Learning

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

Output: Tree

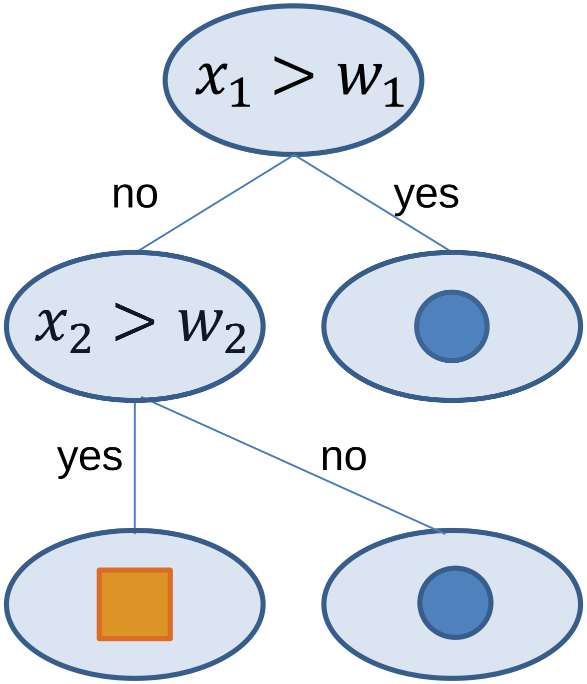
```
root ← all training samples
for d ← 1 to max_depth do
    for each leaf node m at depth d − 1 do
        Find best feature & best threshold, so splitting
          node m into two reduces the most impurity
        Use decision rule to distribute training samples
          from node m across two new leaf nodes
return tree
```

Questions?

Advantages / Disadvantages

Advantages

- Easy to visualize & understand tree



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

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

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)

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)

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 $y_1, y_2, \cdots, y_{J_m}$

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 $y_1, y_2, \cdots, y_{J_m}$
  - 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 $y_1, y_2, \cdots, y_{J_m}$
  - 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 $S_m = \frac{1}{J_m} \sum_{j=1}^{J_m} (y_j - \hat{y}_m)^2$

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 $y_1, y_2, \cdots, y_{J_m}$
  - 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, total MSE $S = \sum_m \frac{J_m}{N} S_m$, where N is the total number of data samples

Regression Tree Learning

- Algorithm is basically the same as classification tree learning

Algorithm: Regression Tree Learning

Input: parameter *max_depth* & training set

Output: Tree

```
root ← all training samples
for d ← 1 to max_depth do
    for each leaf node m at depth d − 1 do
        Find best feature & best threshold, so splitting
          node m into two reduces MSE the most
        Use decision rule to distribute training samples
          from node m across two new leaf nodes
return tree
```

Regression Tree Learning

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

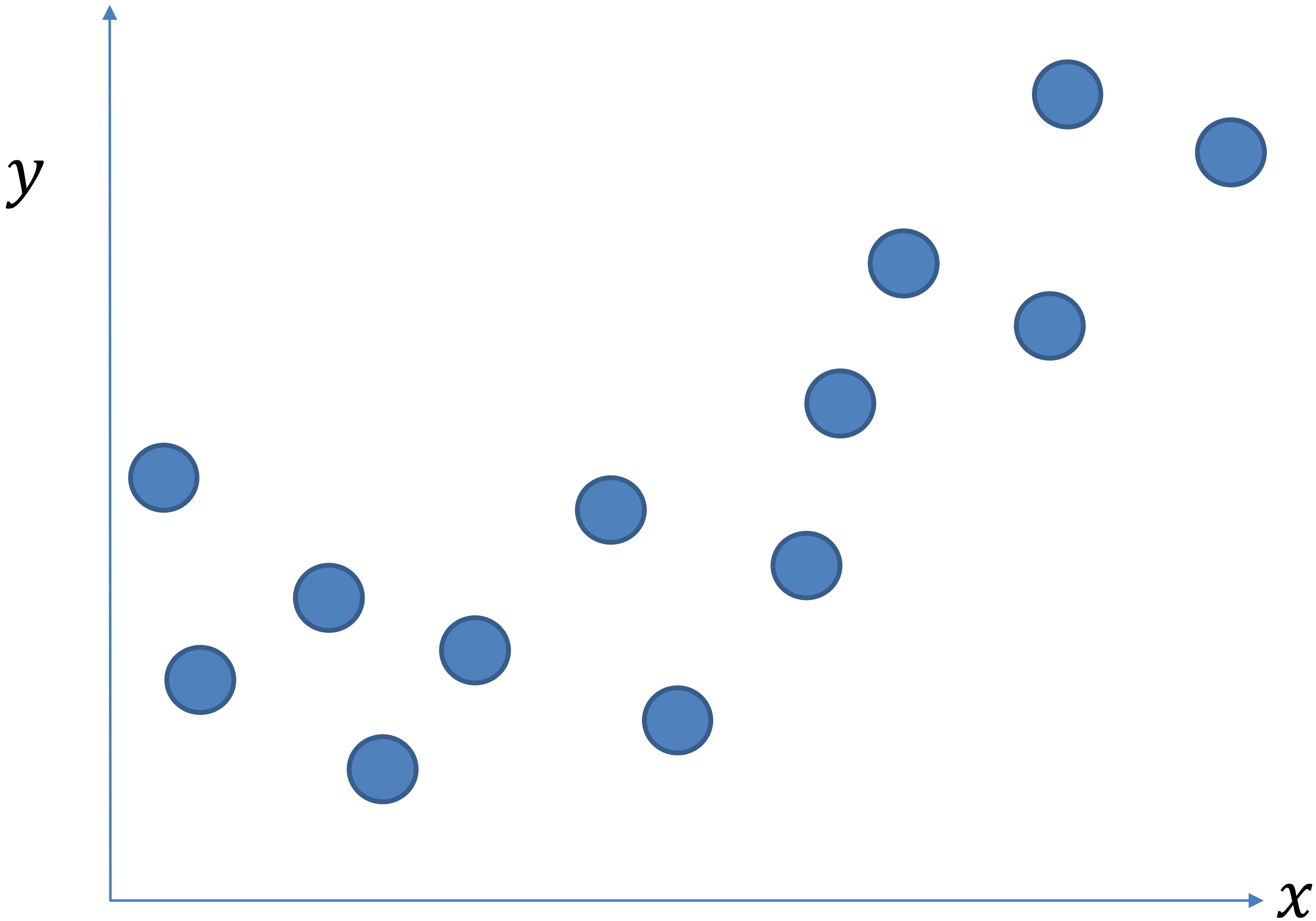
Algorithm: Regression Tree Learning

Input: parameter *max_depth* & training set
Output: Tree

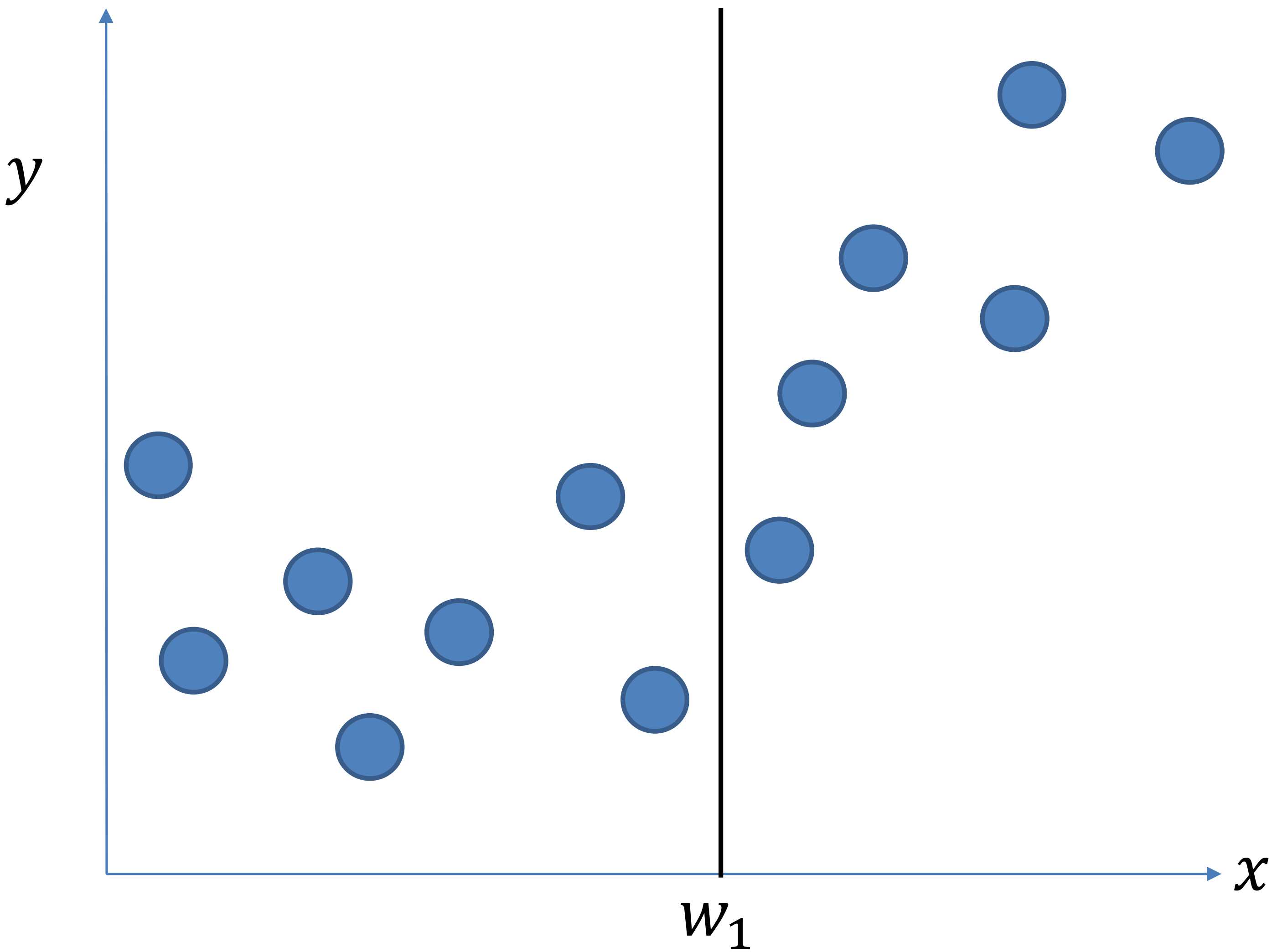
root $\leftarrow$ all training samples
for $d \leftarrow 1$ ***to*** *max_depth* **do**
    for *each leaf node* m *at depth* $d-1$ **do**
        Find best feature & best threshold, so splitting node m into two reduces MSE the most
        Use decision rule to distribute training samples from node m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

Questions?

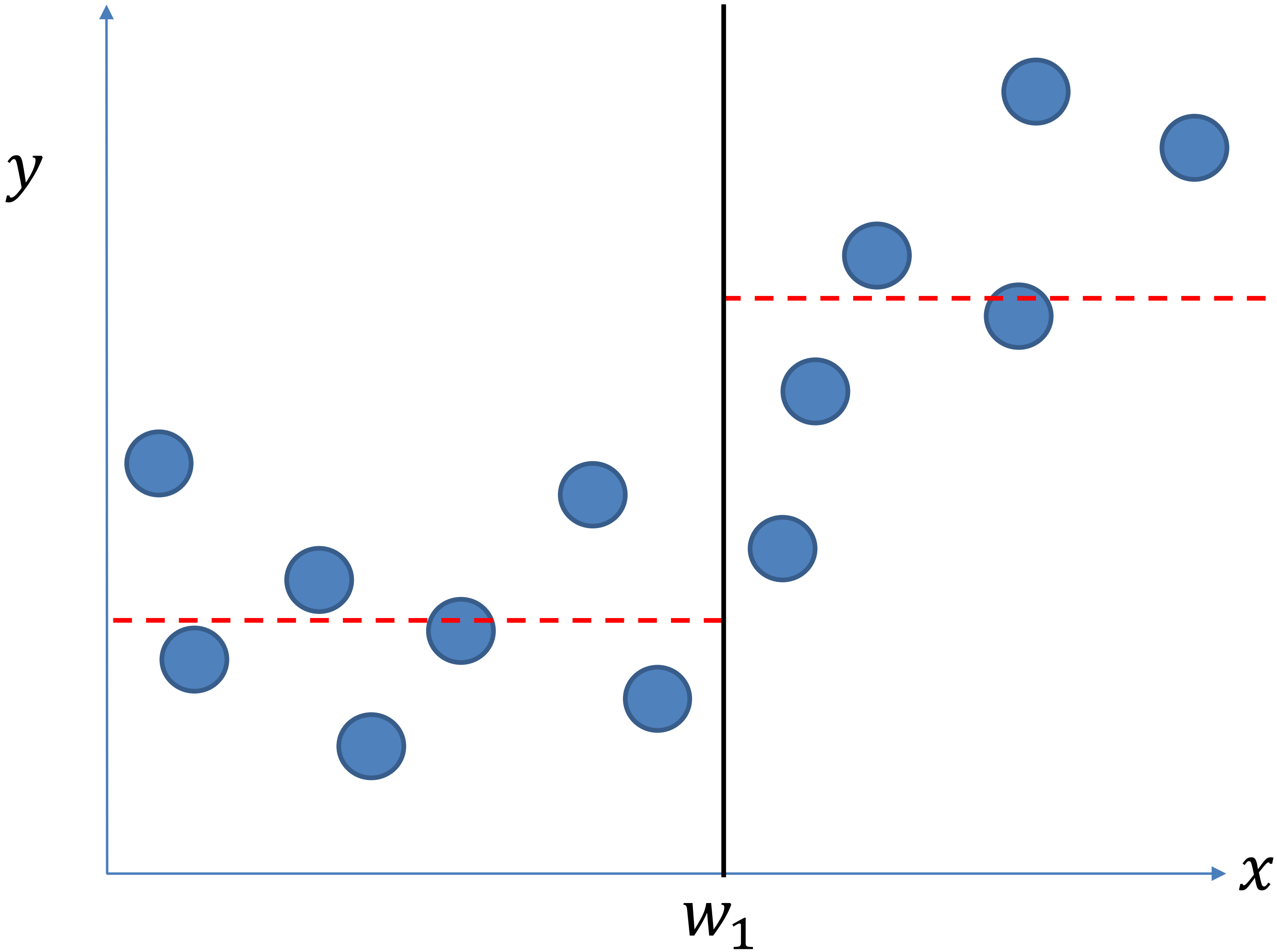
Regression Tree Example



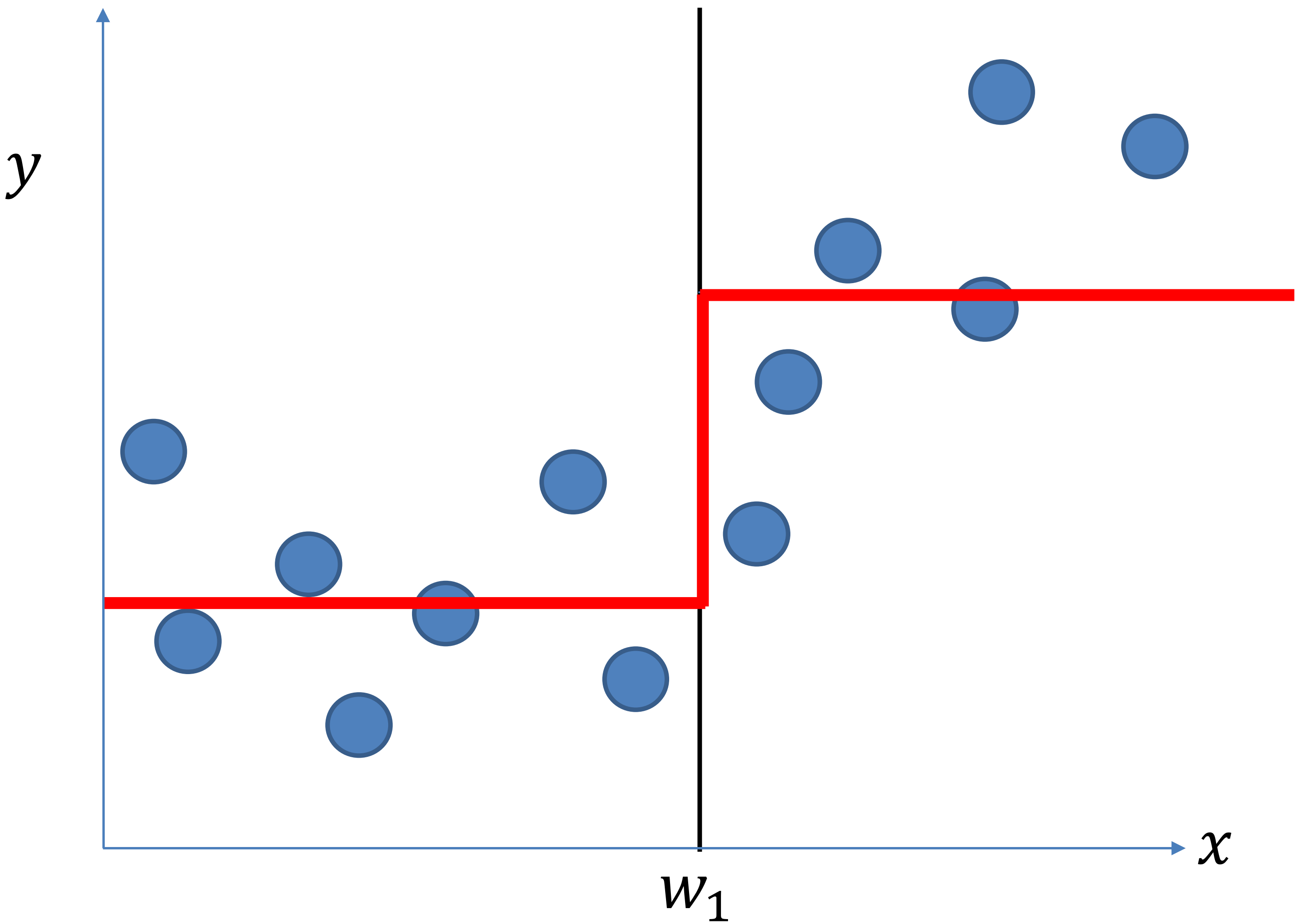
Regression Tree Example



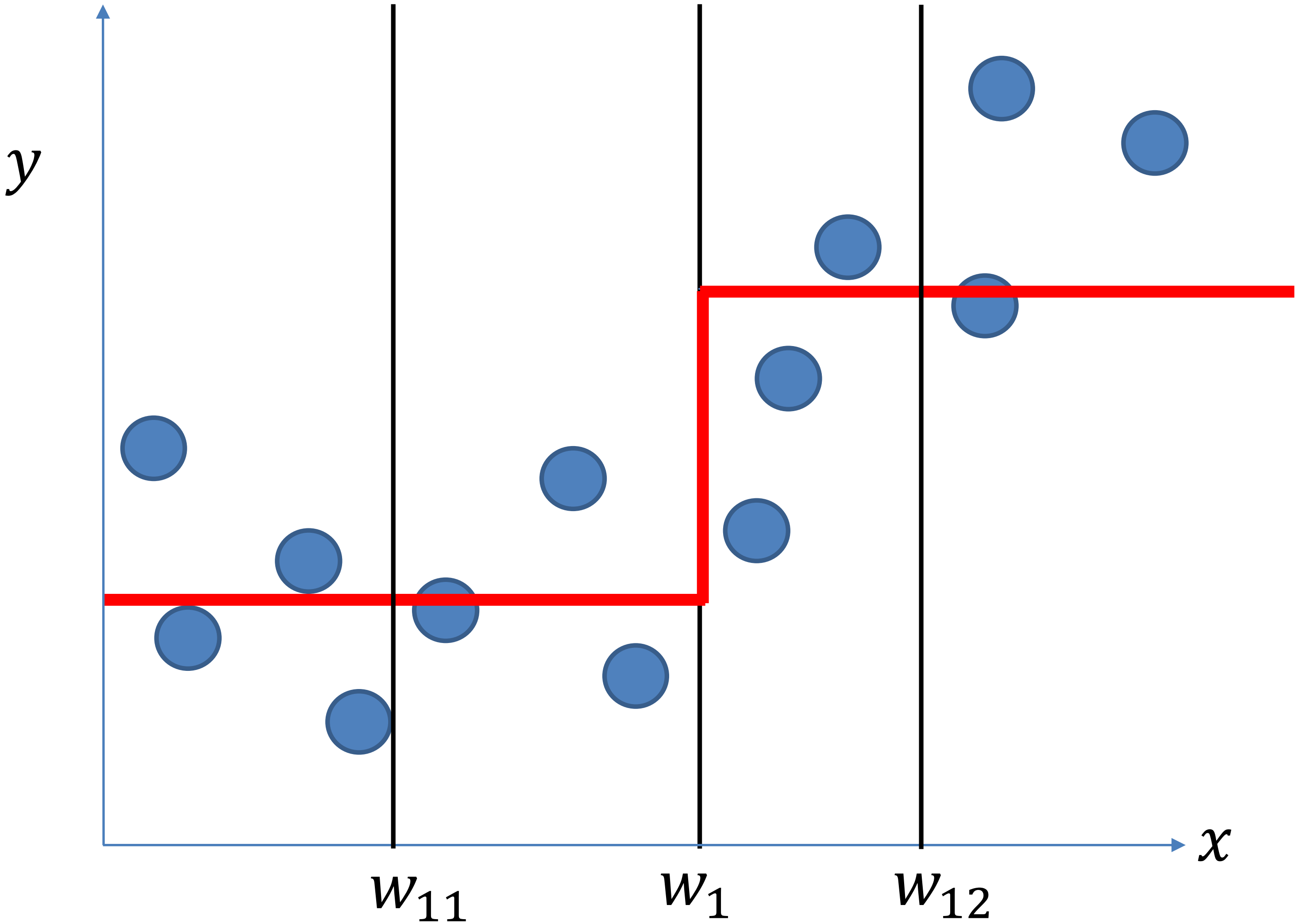
Regression Tree Example



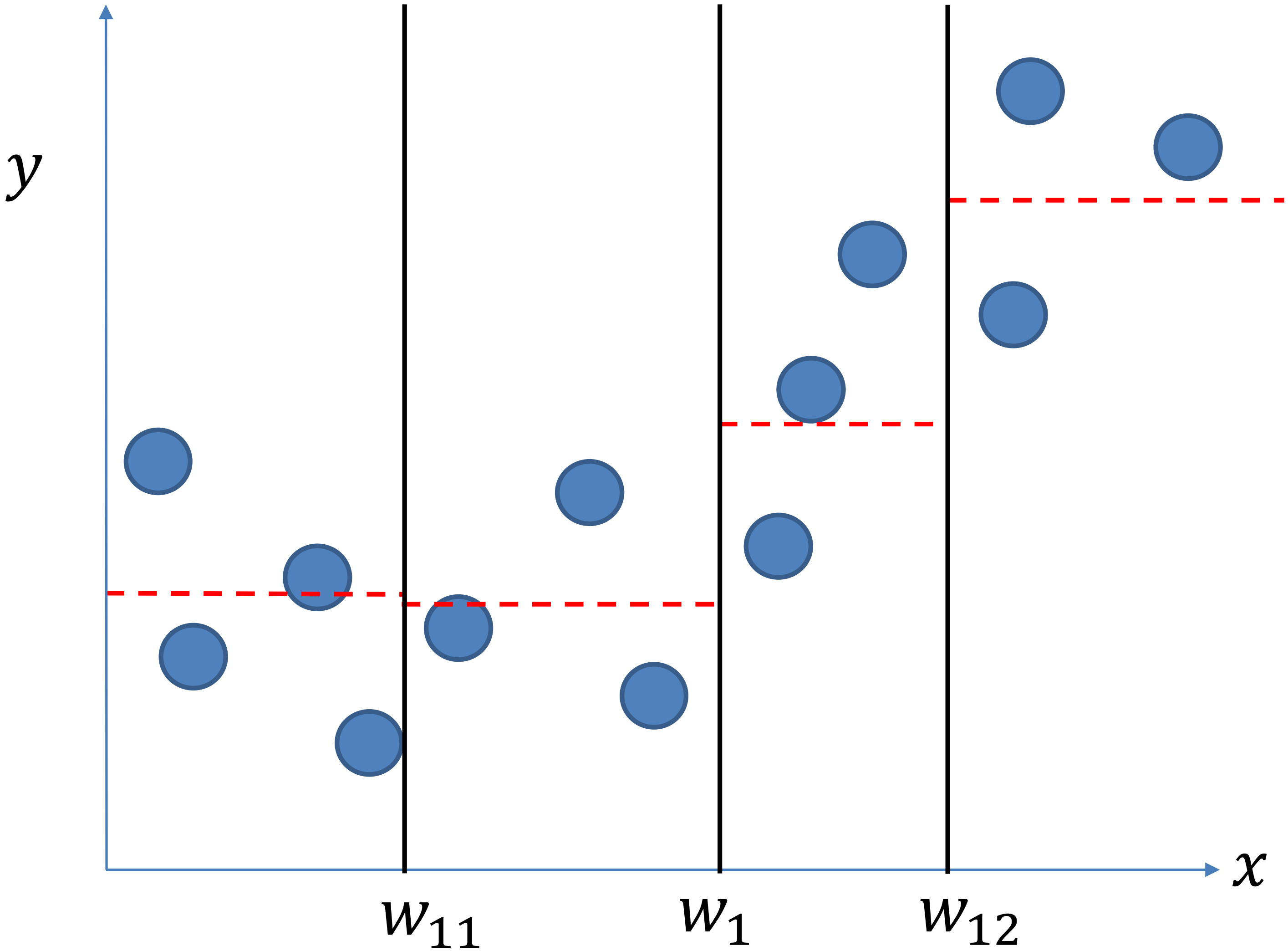
Regression Tree Example



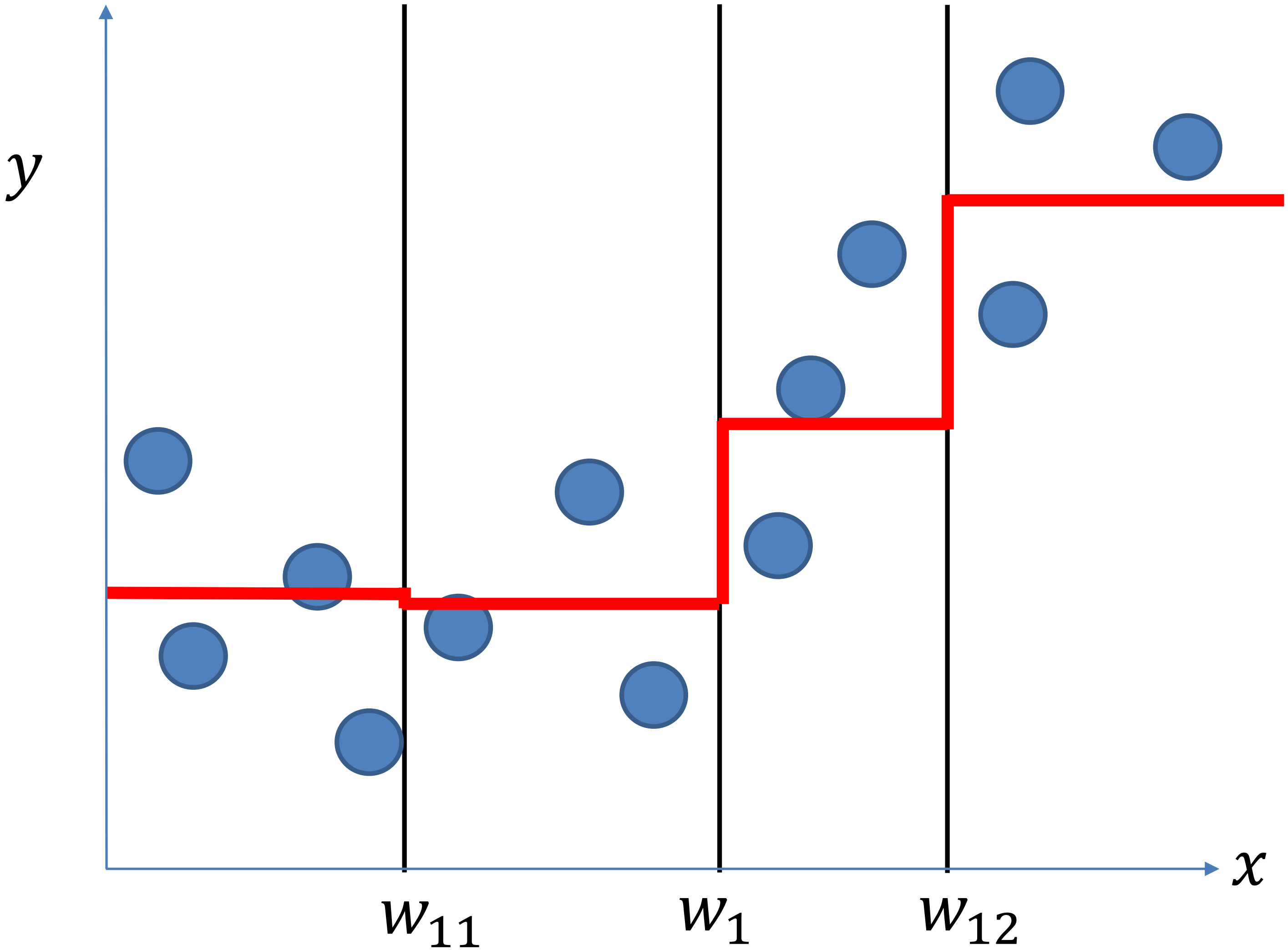
Regression Tree Example



Regression Tree Example



Regression Tree Example



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

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

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

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.

$\mathrm{MSE}_{P\|R(1.5)}$	$\mathrm{MSE}_{P\|R(2.5)}$	$\mathrm{MSE}_{P\|R(3.5)}$	$\mathrm{MSE}_{P\|R(4.5)}$	$\mathrm{MSE}_{P\|R(5.5)}$	$\mathrm{MSE}_{P\|R(6.5)}$
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

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

  or

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

  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 $\frac{1}{100}\sum_{i=1}^{100} f_i(x)$

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)
  - 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 $g_1(x)$, $g_2(x), \cdots, g_{100}(x)$

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 <u>sample data with replacement</u>
  - 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 <u>sample data with replacement</u>
  - 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 <u>sample data with replacement</u>
  - 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

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples
Output: Forest

for $t \leftarrow 1$ ***to*** max_trees **do**
    dataset $\leftarrow$ bootstrap(N training samples)

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples
Output: Forest

```
for t ← 1 to max_trees do
    dataset ← bootstrap(N training samples)
    trees[t] ← TreeLearning(dataset)
```

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples

Output: Forest

for $t \leftarrow 1$ ***to*** max_trees **do**

    dataset $\leftarrow$ bootstrap(N training samples)

    trees[t] $\leftarrow$ TreeLearning(dataset)

forest $\leftarrow$ average(trees)

return $forest$

Random Forest

Algorithm: Random Forest Learning

Input: parameter max_trees & N training samples

Output: Forest

for $t \leftarrow 1$ ***to*** max_trees **do**

  dataset $\leftarrow$ bootstrap(N training samples)

  trees[t] $\leftarrow$ TreeLearning(dataset)

forest $\leftarrow$ average(trees)

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
  - 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