EE2211 Introduction to Machine Learning

Lecture 9

Semester 2
2024/2025

Yueming Jin
ymjin@nus.edu.sg

Electrical and Computer Engineering Department
National University of Singapore

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Probability and Statistics
- Fundamental Machine Learning Algorithms I (Yueming)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Yueming)
  - 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

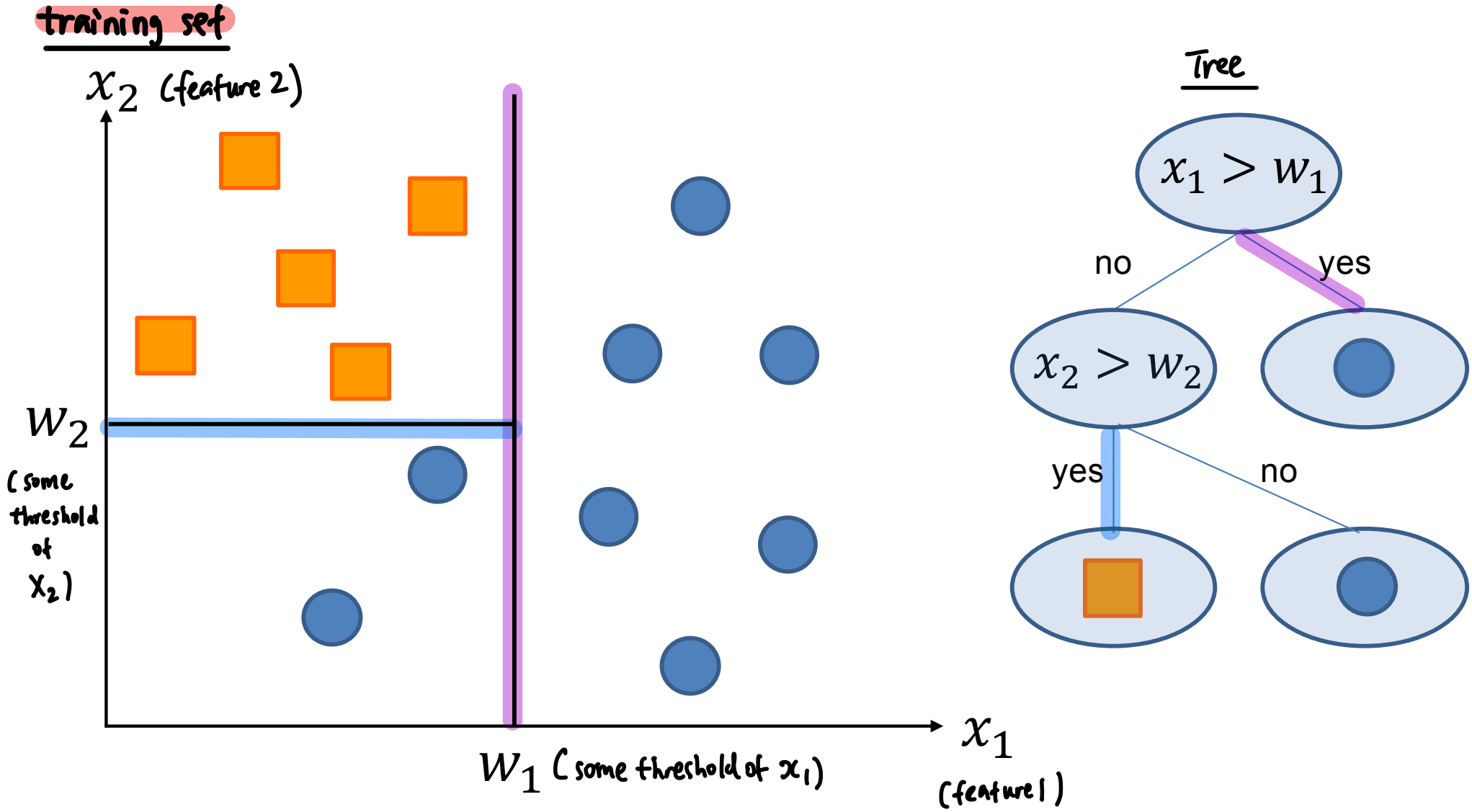
- Overfitting, underfitting and model complexity
- Bias-variance trade-off
- Regularization
- Loss function
- Optimization
- Gradient descent
- Decision trees
- Random forest

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"
- Previous lectures used linear models for regression (and classification)
  - Nonlinearity added by using polynomial regression or other learning models
- New approach today: trees

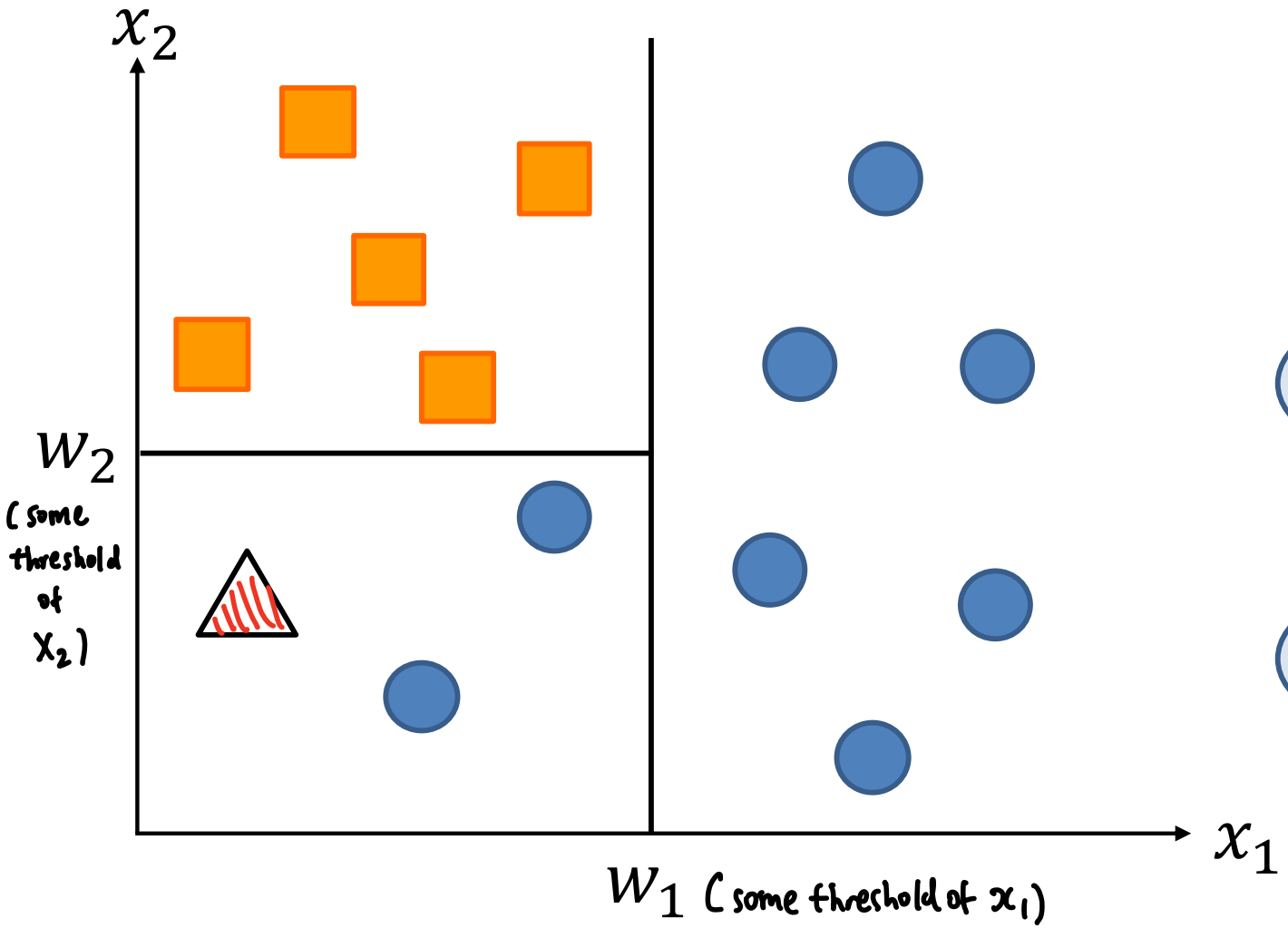
Decision Tree Classification Example

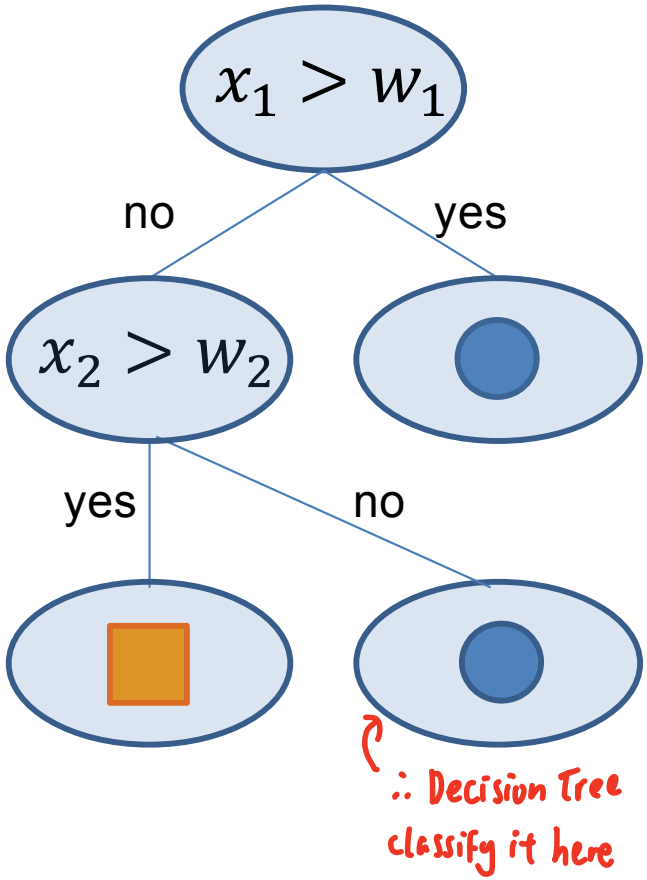
- Goal: predict class labels using two features x_1 & x_2



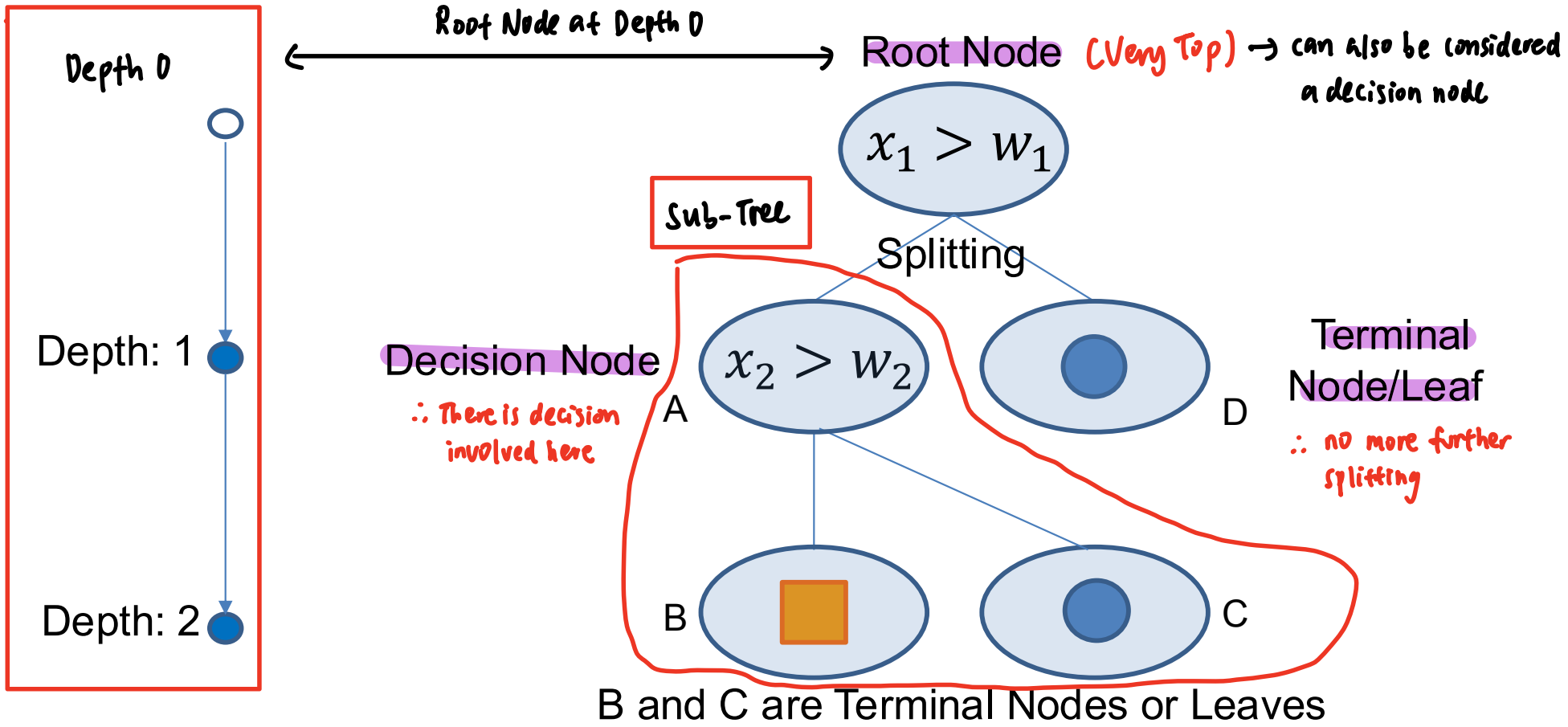
Decision Tree Classification Example

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





Basic Terminologies

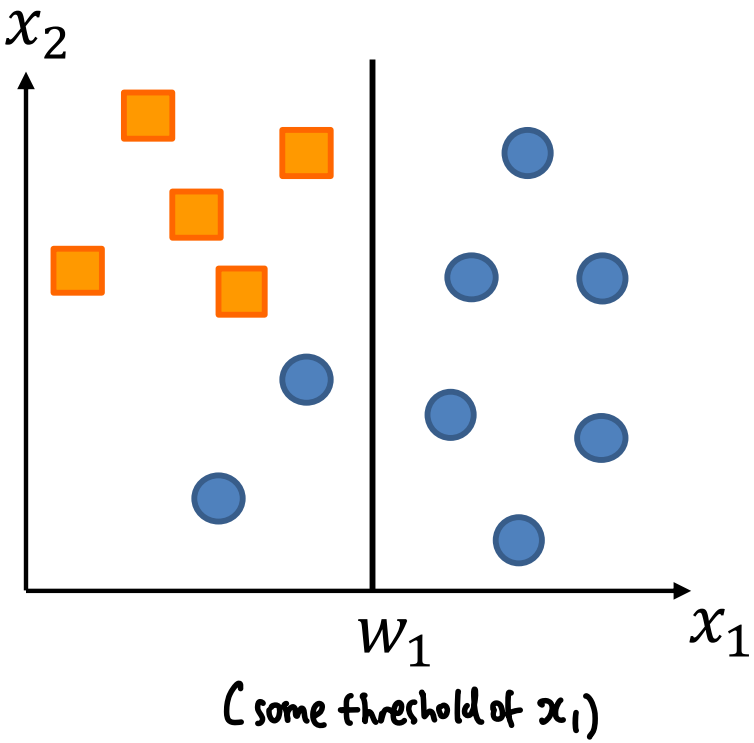


A-B-C forms a **sub-tree** or **branch**.
A is **parent node** of B and C; B and C are **children nodes** of A.

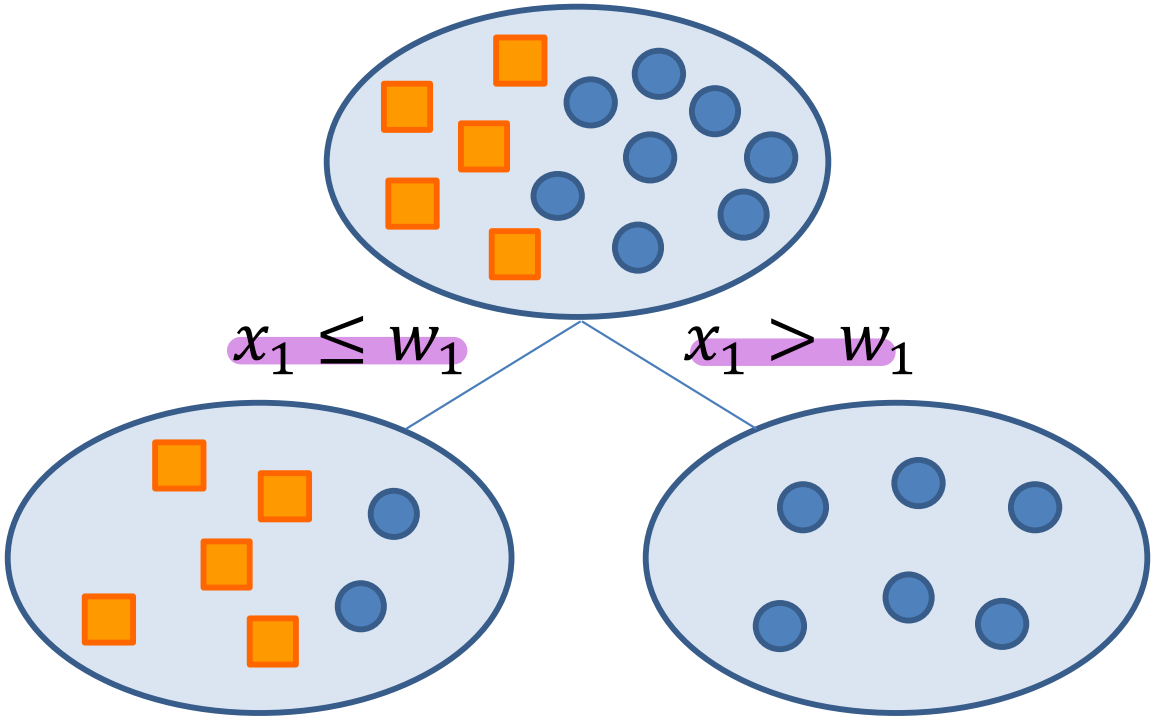
Building a Classification Decision Tree

- 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



∴ Replace the root node with all the training data



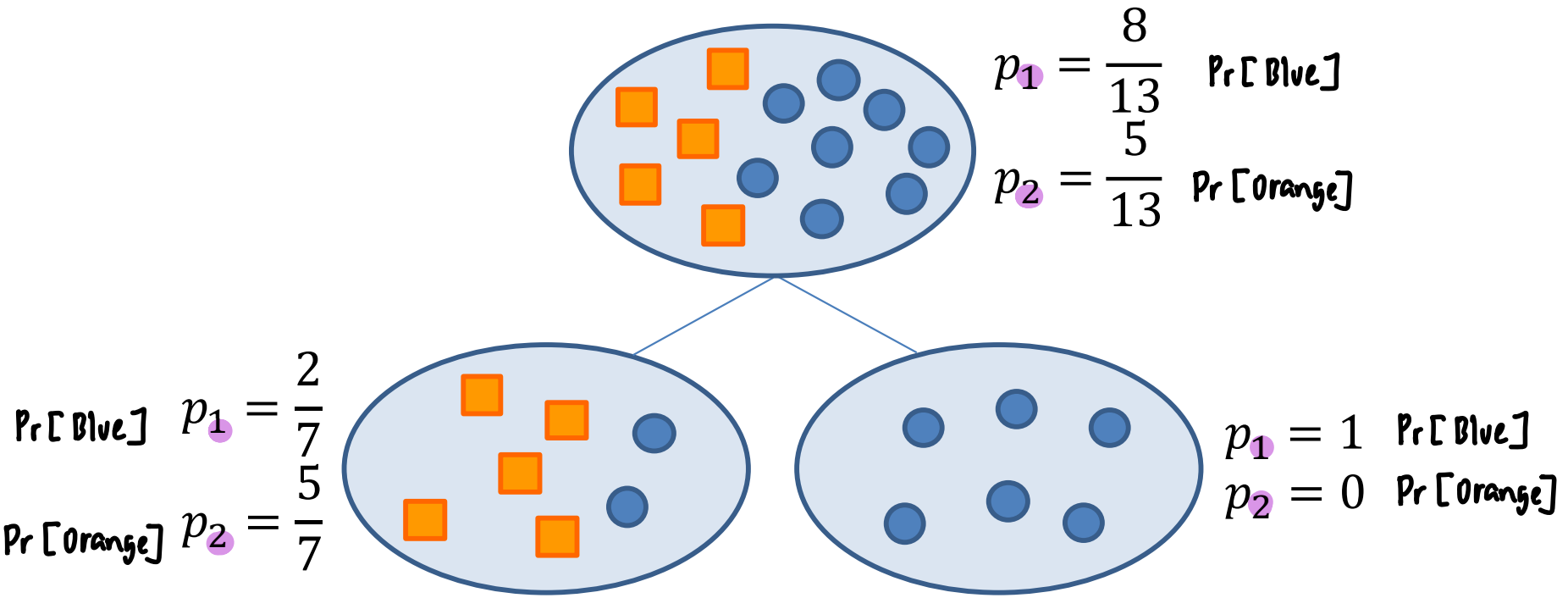
"Impure" because it contains ■ & ●

"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 → 100% accuracy

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

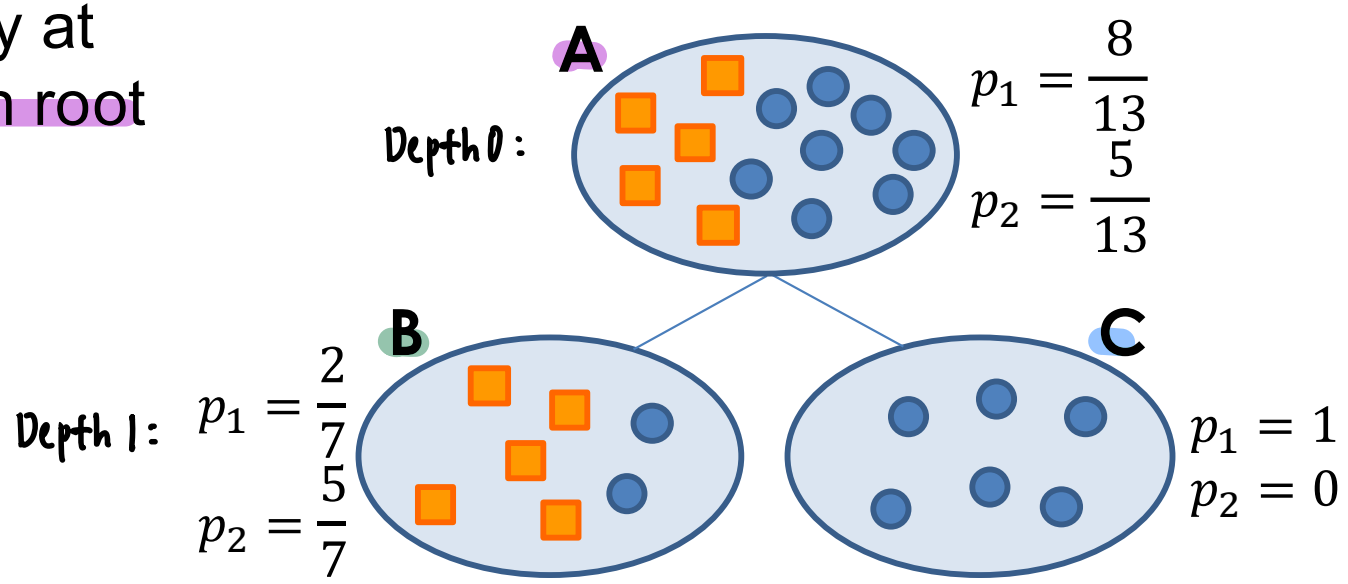


Gini Impurity

in this example, only two - classes

- Let K = # classes, define $Q_m = 1 - \Sigma_{i=1}^{K} p_i^2 = 1 - p_1^2 - p_2^2$
- Node A: $Q_A = 1 - (8/13)^2 - (5/13)^2 = 0.4734$ → Depth 0
- Node B: $Q_B = 1 - (2/7)^2 - (5/7)^2 = 0.4082$
- Node C: $Q_C = 1 - 1^2 - 0^2 = 0$ ∴ lowest, ie most pure
- Overall Gini (depth 1) = fraction of data samples in node B x Q_B + fraction of data samples in node C x Q_C ∴ something like weighted sum, bcos node B has more elements (7) than Node C (6)
  - $\left(\frac{7}{13}\right) \times 0.4082 + \left(\frac{6}{13}\right) \times 0 = 0.2198$
- Observe lower impurity at depth 1 compared with root
- Same Gini formula for more than 2 classes:

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



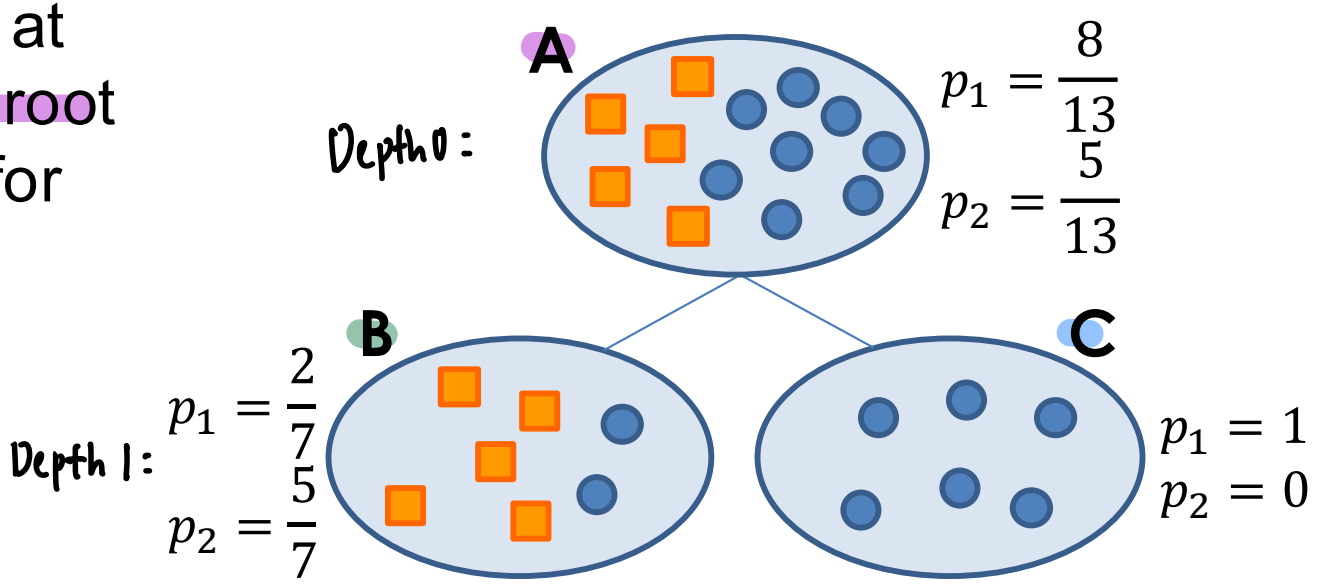
Entropy

↳ higher the entropy, the more random distribution across the classes, which means nodes more impure

in this example, only two-classes

- Let K = # classes, define $Q_m = -\Sigma_{i=1}^{K} p_i \log_2 p_i = -p_1 \log_2 p_1 - p_2 \log_2 p_2$ (sum of)
- Node A: $Q_A = -(8/13)\log_2(8/13) - (5/13)\log_2(5/13) = 0.9612$ → Depth 0
- 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 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:

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

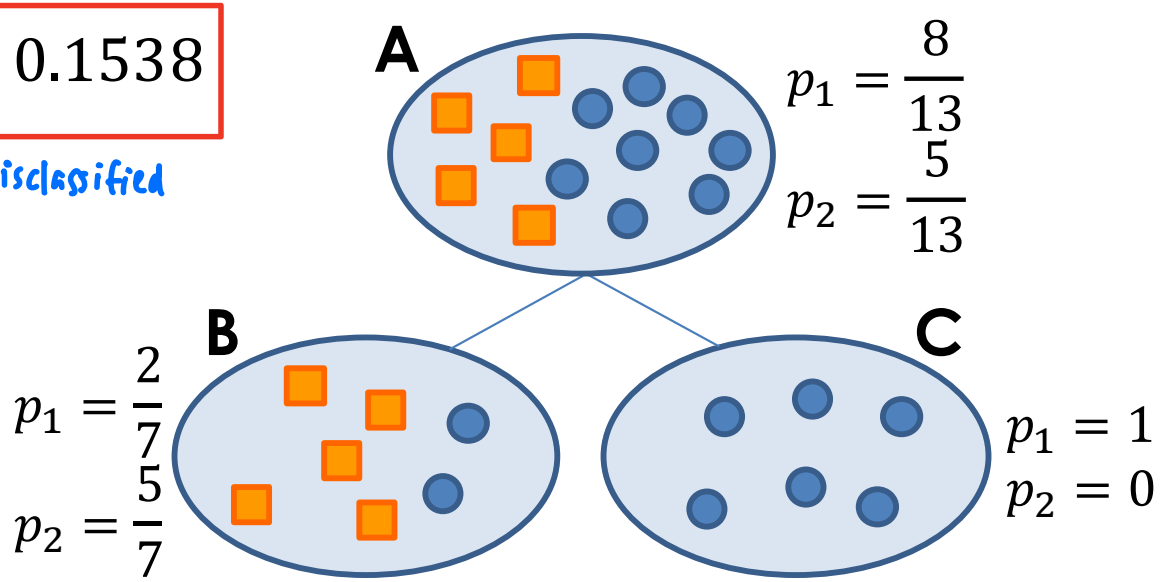


Misclassification rate

- 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 => $Q_A = 1 - 8/13 = 5/13$ → Depth 0
- Node B: $p_2 > p_1$, so best classification = class 2 => $Q_B = 1 - 5/7 = 2/7$ →
- Node C: $p_1 > p_2$, so best classification = class 1 => $Q_C = 1 - 1 = 0$
- 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
  - $\left(\frac{7}{13}\right) \times \left(\frac{2}{7}\right) + \left(\frac{6}{13}\right) \times 0 = 0.1538$

misclassified

- Observe lower impurity at depth 1 compared with root
- Same misclassification rate formula for more than 2 classes: $Q_m = 1 - \max_i p_i$



Classification Tree Learning (discrete)

Algorithm: Classification Tree Learning

Input: Impurity measure Q, parameter max_depth & training set

(Gini / Misclassification rate / Entropy)

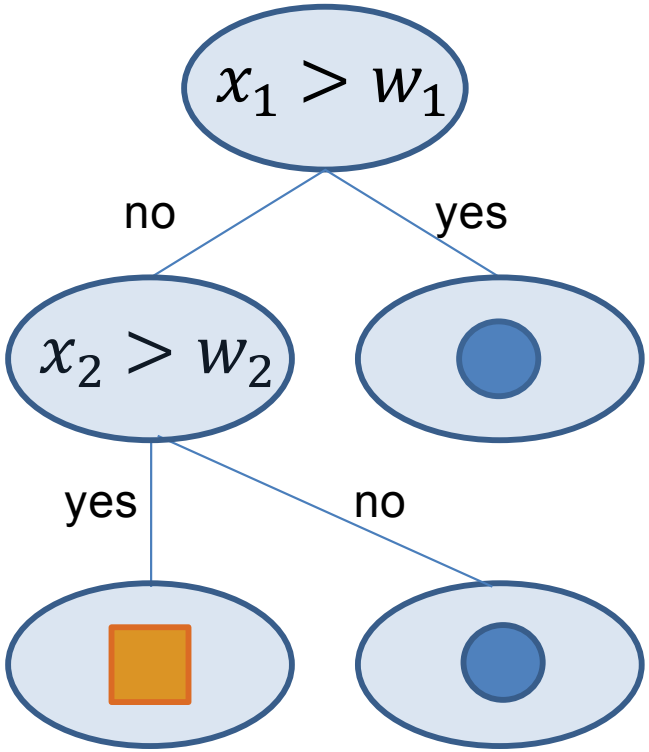
Output: Tree

```
1 root ← all training samples   (assign all training set to root node)
2 for d ← 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         Use decision rule to distribute training samples
          from node m across two new leaf nodes
6 return tree
```

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
- Makes less assumptions about the relationship between features & target

→ Since decision tree only consider one feature at a time, in certain situations, may not be very good as can get trapped in some bad local minimum.

Disadvantages

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

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)

even if not "pure"

Basically instead of making the tree v. complex, we can build simple tree by randomly taking out 10 out of 100 features to pick as the best features to split the tree.

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 $S_m = \frac{1}{J_m} \sum_{j=1}^{J_m} (y_j - \hat{y}_m)^2$ (original target) (predicted)
- 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 (continuous)

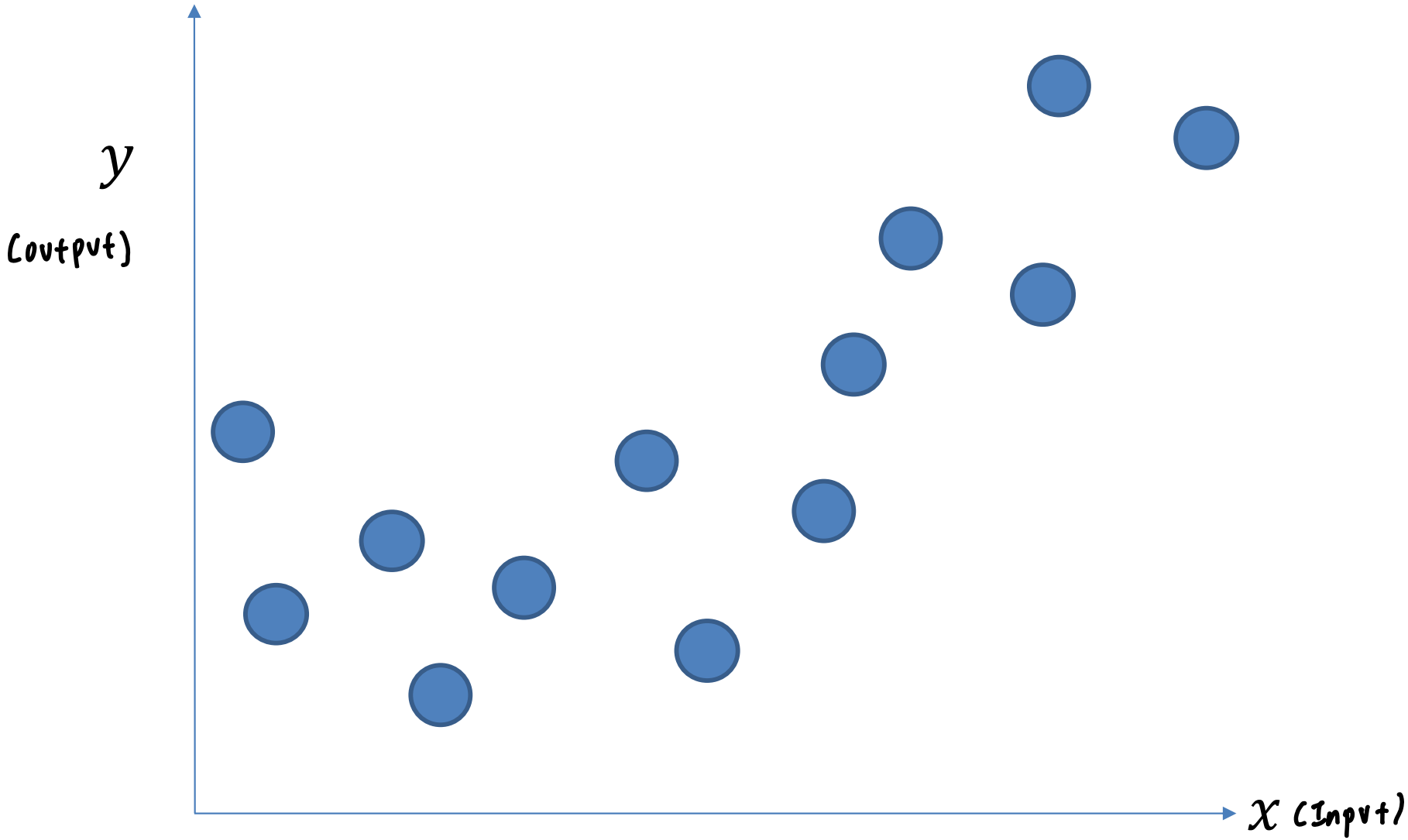
- Algorithm is basically the same as classification tree learning
- Various approaches to reduce overfitting also apply here

Algorithm: Regression Tree Learning

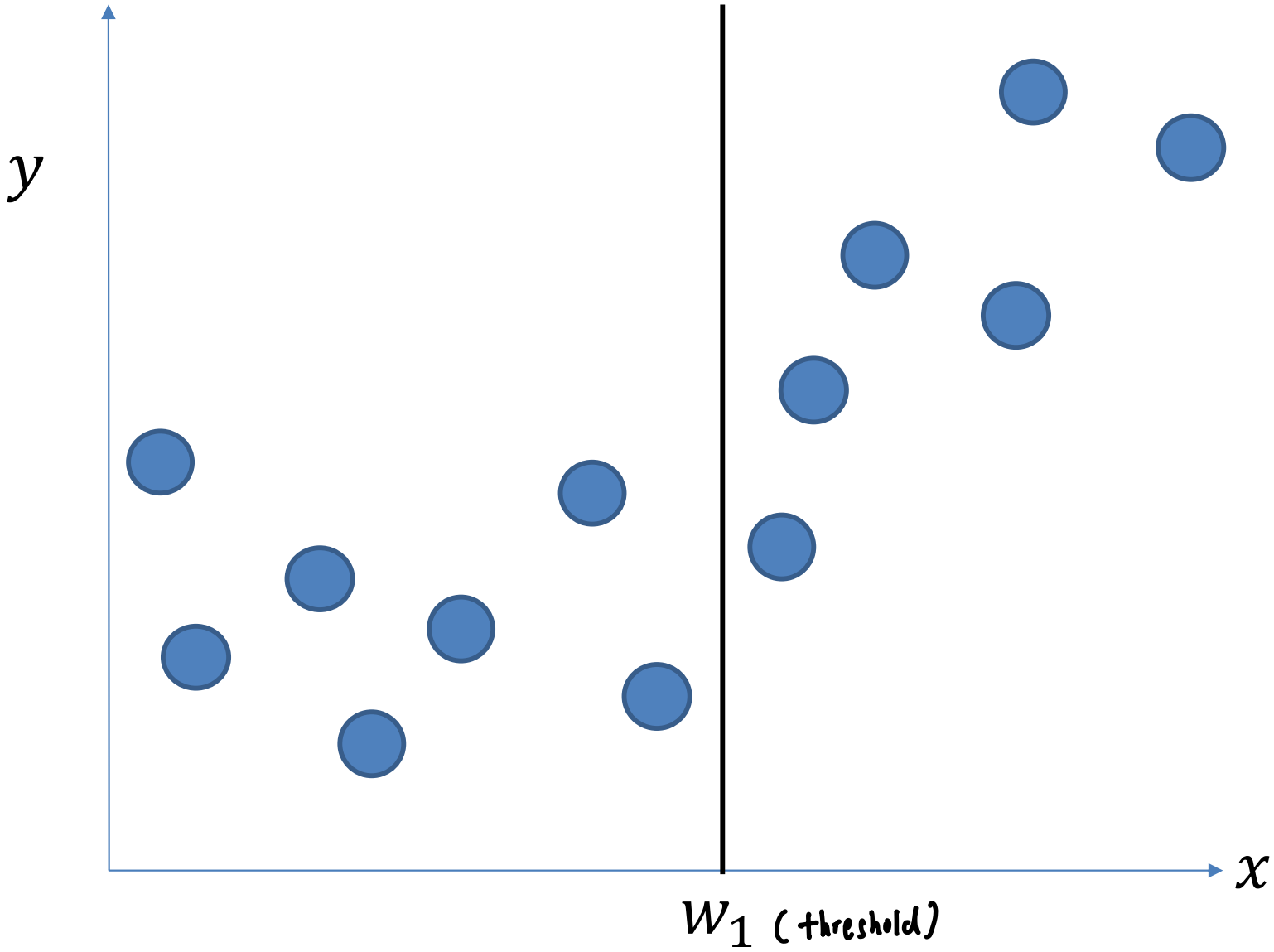
Input: parameter *max_depth* & training set
Output: Tree

```
root ← 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
```

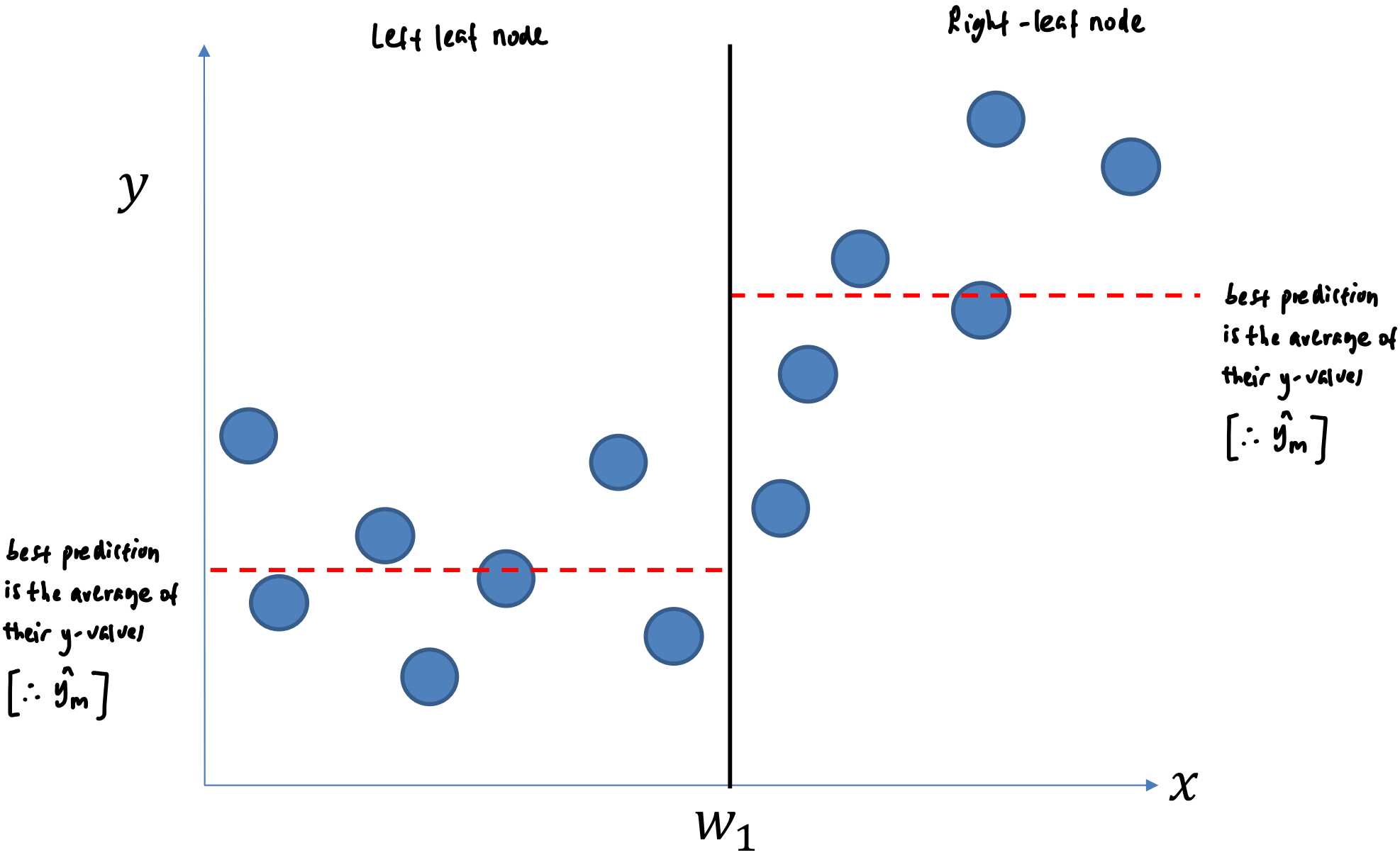
Regression Tree Example



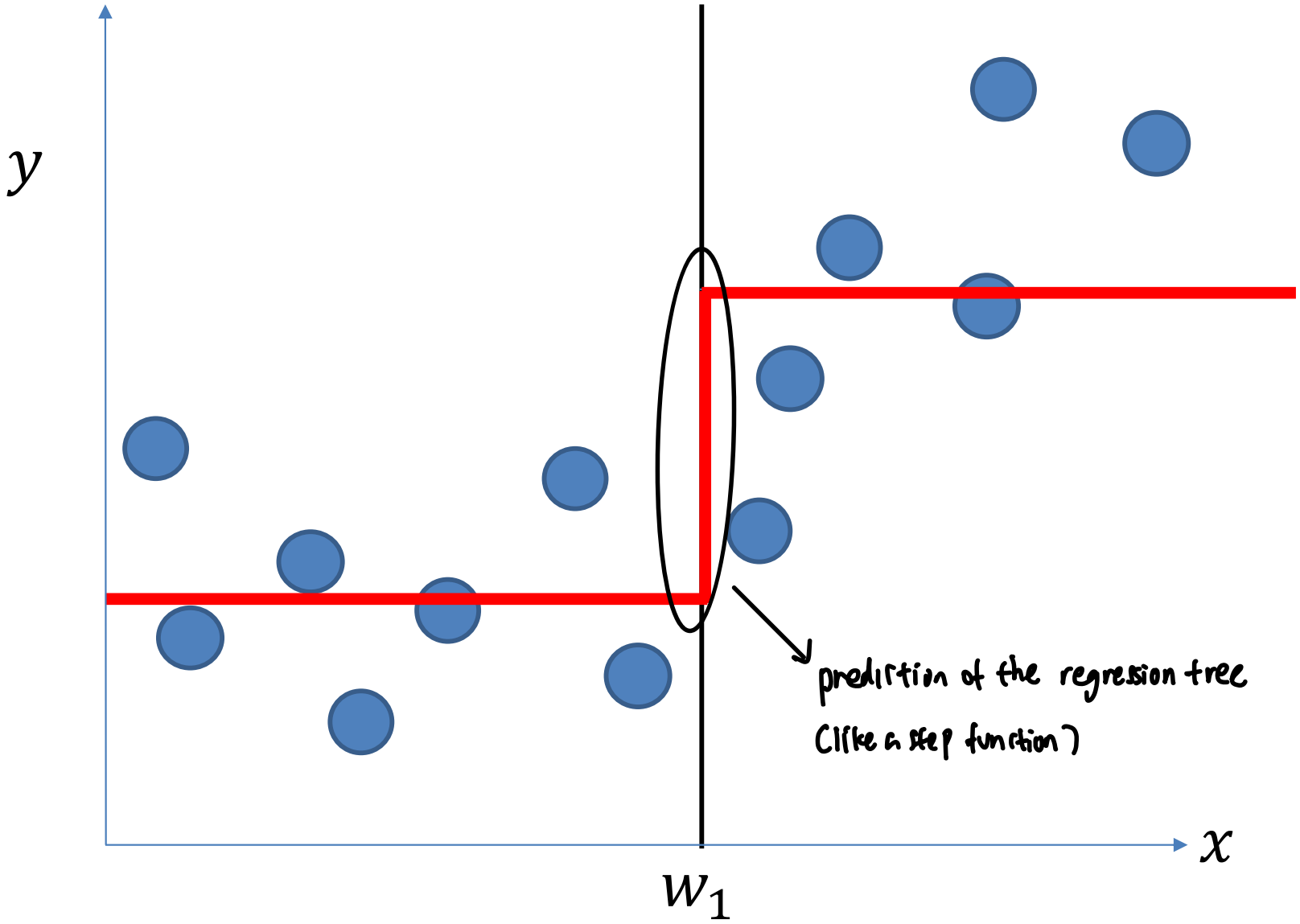
Regression Tree Example



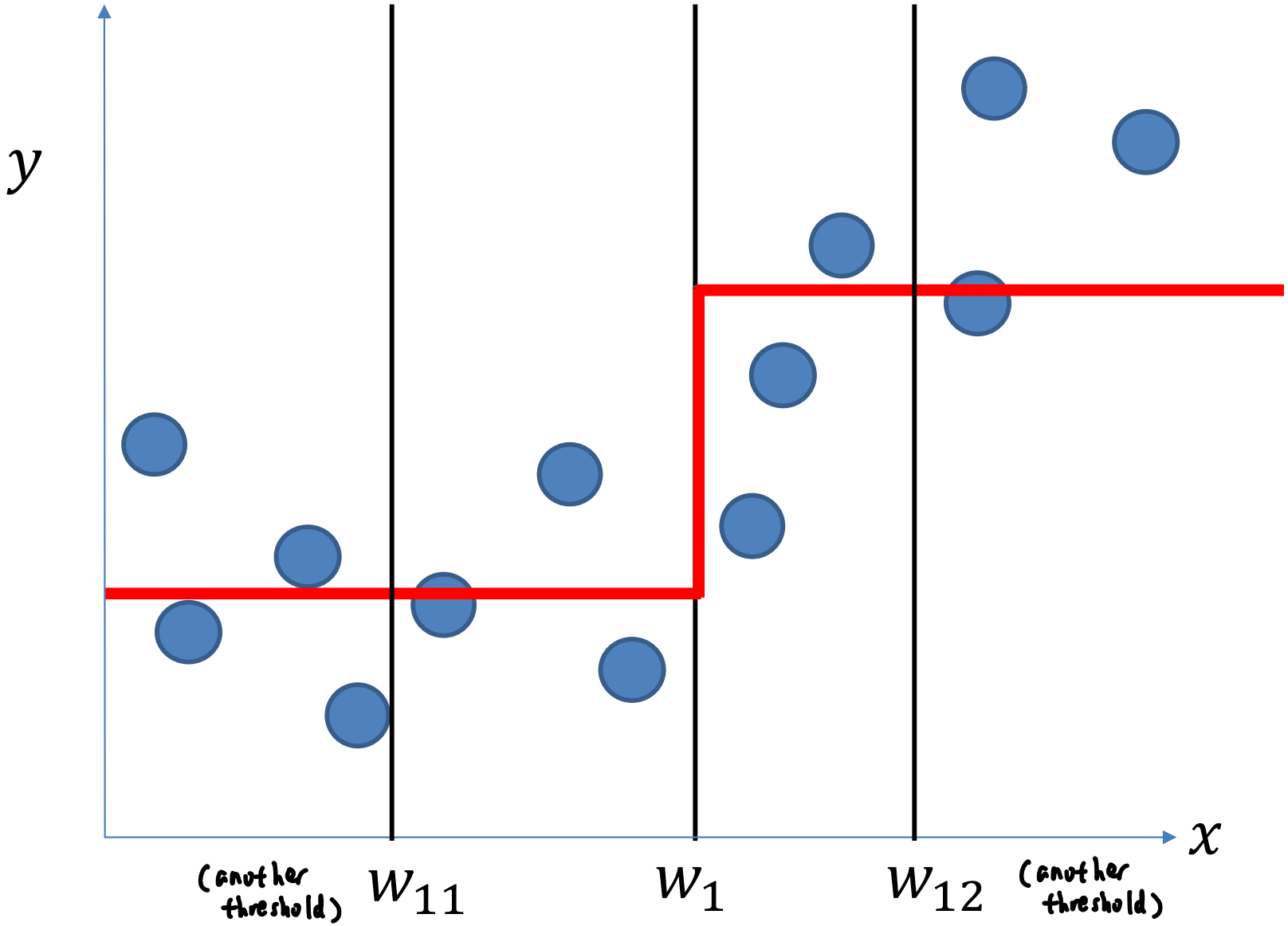
Regression Tree Example



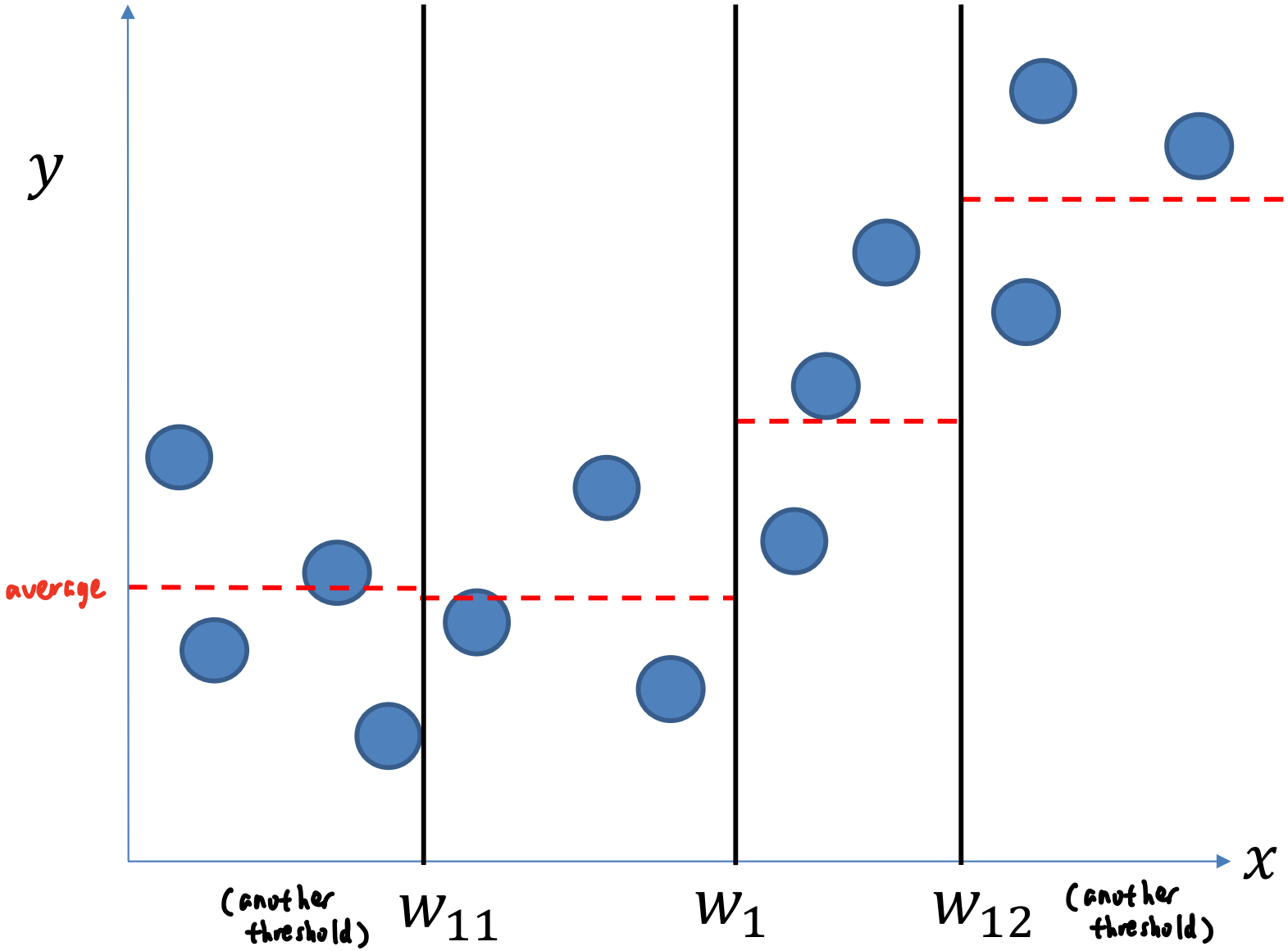
Regression Tree Example



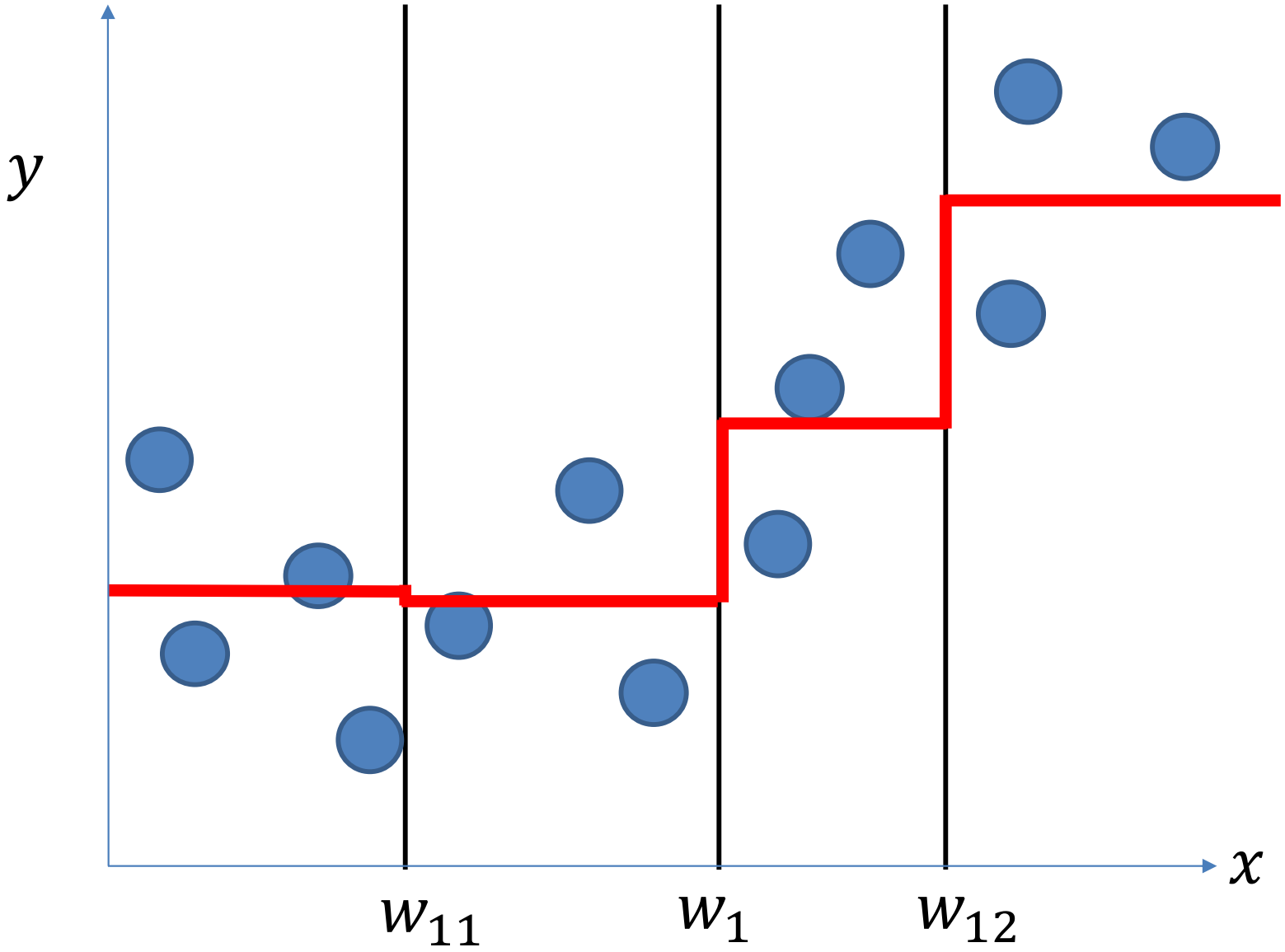
Regression Tree Example



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. (training set)

(Feature 1: S) (Feature 2: R) (Target: P)

	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

Calculation of MSE for House Size Split

0.75 threshold

	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

- 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

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

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

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

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

(minimum MSE)

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

$$S_m = \frac{1}{J_m}\sum_{j=1}^{J_m}(y_j - \hat{y}_m)^2 \quad \text{(original, } \hat{y} \text{ predict)} \qquad \text{MSE } S = \sum_m \frac{J_m}{N} S_m \qquad \hat{y}_m = \frac{1}{J_m}\sum_{i=1}^{J_m} y_j$$

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.

$\text{MSE}_{P\|R(1.5)}$	$\text{MSE}_{P\|R(2.5)}$	$\text{MSE}_{P\|R(3.5)}$	$\text{MSE}_{P\|R(4.5)}$	$\text{MSE}_{P\|R(5.5)}$	$\text{MSE}_{P\|R(6.5)}$
0.0435	0.0276	0.0145	0.0222	0.0116	0.0325

Choose the threshold with minimum MSE

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 $S \geq 2.5$. (shown pg 27) ↳ lowest MSE

	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

$S<2.5$ / $S\geq 2.5$

- Split the dataset into two sub-datasets and we may decide to stop or split the R feature.

↓ if feel that change in MSE is small enough or num of samples in node is reached (based on what we set) or reach max tree depth

- 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.,

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

Python demo

To reduce instability…

- For both classification & regression trees, small perturbations to data can result in very different trees => low bias, high variance (changes)
- To reduce variance, we can perturb training set to generate M perturbed training sets
  - Train one tree for each perturbed training set → output a prediction
  - 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 $\frac{1}{100}\sum_{i=1}^{100} f_i(x)$ → averaged
- 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

(change)

- 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

```
for t ← 1 to max_trees do
    dataset ← bootstrap(N training samples)
    trees[t] ← TreeLearning(dataset)
forest ← 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) → Look at pg 16

Summary

Ass2 submission

Ass3 will release today

- 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