University of Southampton

COMP6237 Data Mining
Lecture 7: Decision Trees

Zhiwu Huang

Zhiwu.Huang@soton.ac.uk

Lecturer (Assistant Professor) @ VLC of ECS

University of Southampton

Lecture slides available here:

http://comp6237.ecs.soton.ac.uk/zh.html

(Thanks to Prof. Jonathon Hare and Dr. Jo Grundy for providing the lecture materials used to develop the slides.)

Recap – Document Filtering

Posterior $\propto$ Likelihood Prior

Naive Bayes rules

$$P(c|d) = \frac{P(d|c)\,P(c)}{p(d)}$$

where $p(d|c) = \prod p(f|c)$ Feature independence

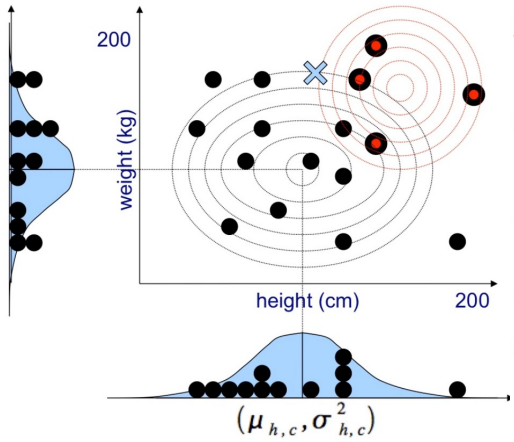
The **prior** can be calculated:

- Empirically, using the total No. docs in c divided by the total number
- or assuming all classes to be equally probably

$p(d)$ is constant, so therefore irrelevant, as same for all categories

1. Continuous features, e.g. heights and weights for cats/dogs, assume features are Gaussian distributed

$$P(a)=\frac{4}{4+12}=0.25\,;\,P(c)=0.75$$



$$p(h_x|c)=\frac{1}{\sqrt{2\pi\sigma^2_{h,c}}}\exp-\frac{1}{2}\left(\frac{(h_x-\mu_{h,c})^2}{\sigma^2_{h,c}}\right)$$

$$p(w_x|c)=\frac{1}{\sqrt{2\pi\sigma^2_{w,c}}}\exp-\frac{1}{2}\left(\frac{(w_x-\mu_{w,c})^2}{\sigma^2_{w,c}}\right)$$

$$p(h_x|a)=\frac{1}{\sqrt{2\pi\sigma^2_{h,a}}}\exp-\frac{1}{2}\left(\frac{(h_x-\mu_{h,a})^2}{\sigma^2_{h,a}}\right)$$

$$p(w_x|a)=\frac{1}{\sqrt{2\pi\sigma^2_{w,a}}}\exp-\frac{1}{2}\left(\frac{(w_x-\mu_{w,a})^2}{\sigma^2_{w,a}}\right)$$

$$P(x|a)=p(h_x|a)\,p(w_x|a)$$

$$P(x|c)=p(h_x|c)\,p(w_x|c)$$

2. Discrete features, e.g. frequencies of words in hams/spams, assume features are Binomial distributed

n_c *is the total number of documents in category* c *(e.g., spam,* n_c*=30)*

	money	viagra	tea	you	dear	…	Total
Spam	15	8	0	19	15	…	30
Ham	2	0	15	20	12	…	70

n_{fc}*: the number of documents with feature* f *(e.g., dear for spam,* n_{fc}*=15) in category* c

strength of assumed, commonly initialized as 1 — *assumed probability, commonly initialized as 0.5* — *frequencies of feature* f *occurs across categories*

$$p_w(f|c) = \frac{weight \times assumed + count \times p_{raw}(f|c)}{count + weight}$$

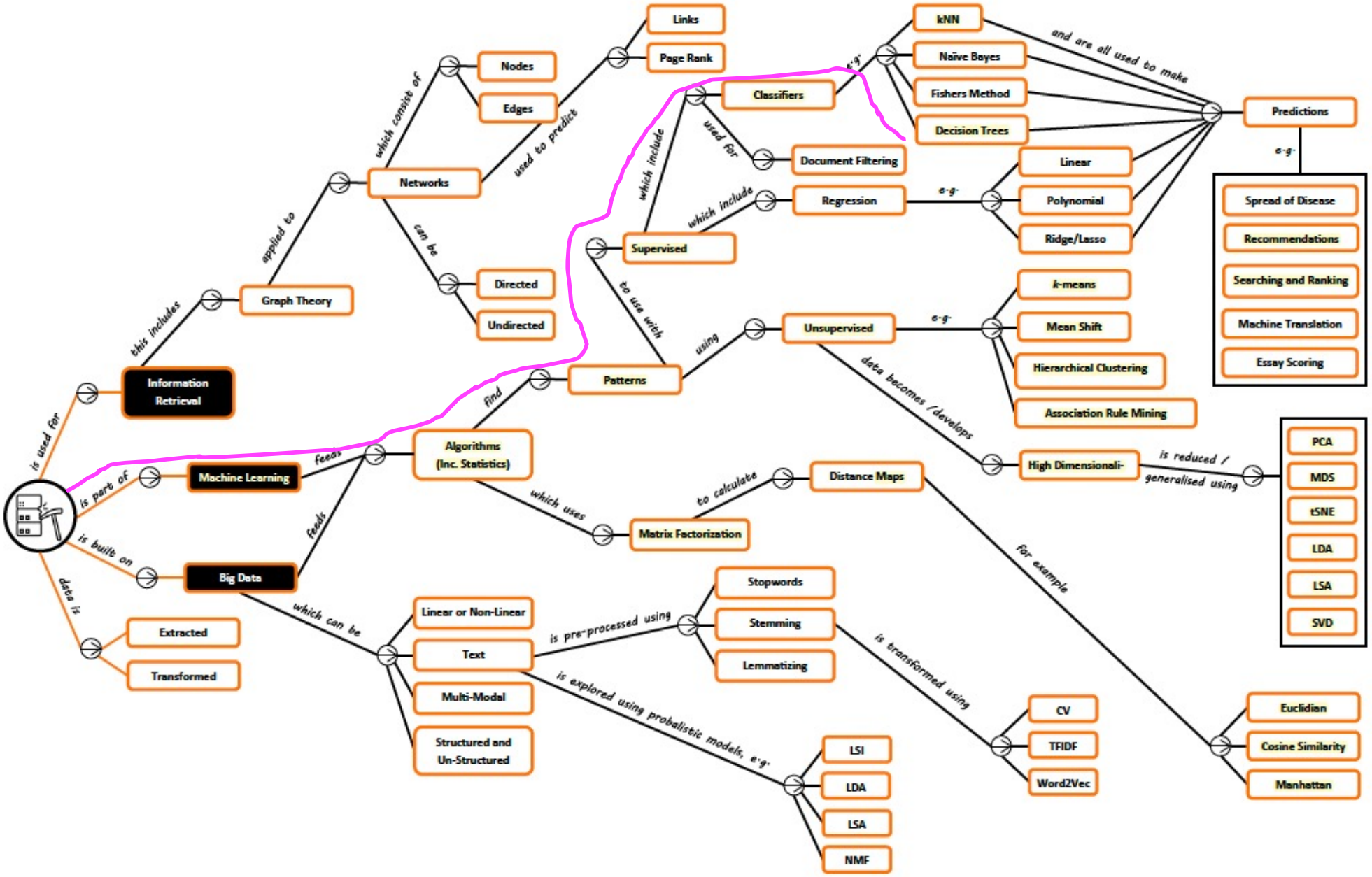
$p_{raw}(f|c) = \frac{n_{fc}}{n_c}$

If a given test document x contains features money (f_1) and viagra (f_2) , then $p(x|c) = p(f_1|c) * p(f_2|c)$

Maximum a Posteriori (MAP): *choose the category with the* ***largest posterior***

$$c = argmax_{c_i}\, p(c_i|x) \propto argmax_{c_i}\, p(x|c_i) * p(c_i)$$

Decision Trees – **Roadmap**



Decision Trees – **Textbook**

PART FOUR CLASSIFICATION

The classification task is to predict the label or class for a given unlabeled point. Formally, a classifier is a model or function M that predicts the class label $\hat{y}$ for a given input example $\mathbf{x}$, that is, $\hat{y} = M(\mathbf{x})$, where $\hat{y} \in \{c_1, c_2, \ldots, c_k\}$ and each c_i is a class label (a categorical attribute value). Classification is a *supervised learning* approach,

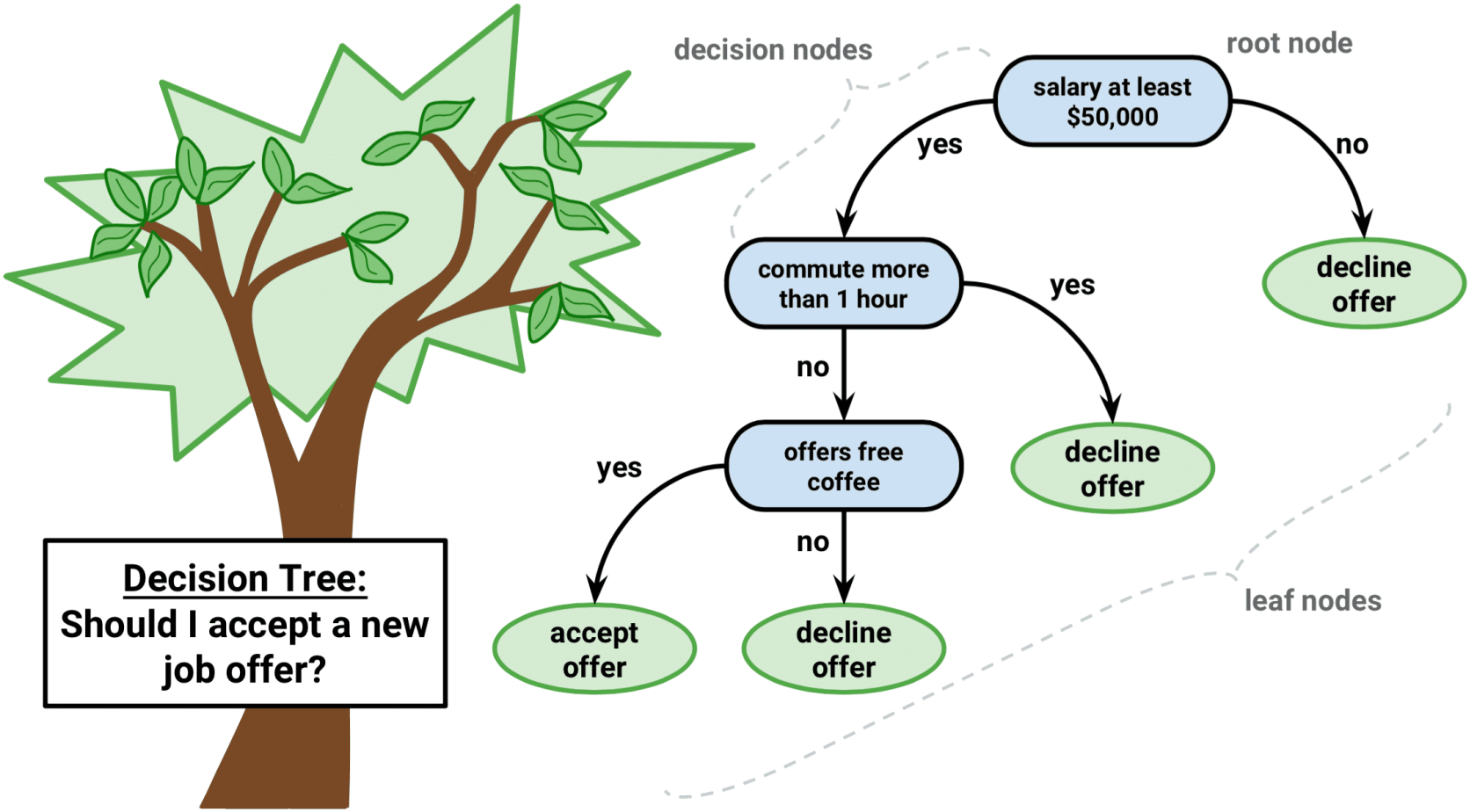
- Data Mining and Machine Learning: Fundamental Concepts and Algorithms M. L. Zaki and W. Meira

4 Classification: Alternative Techniques

The previous chapter introduced the classification problem and presented a technique known as the decision tree classifier. Issues such as model overfitting and model evaluation were also discussed. This chapter presents alternative techniques for building classification models—from simple techniques such as rule-based and nearest neighbor classifiers to more sophisticated techniques such as artificial neural networks, deep learning, support vector machines, and ensemble methods. Other practical issues such as the class imbalance and multiclass problems are also discussed at the end of the chapter.

- Introduction to Data Mining, *P. Tan et al*

Decision Trees – **Overview (1/3)**

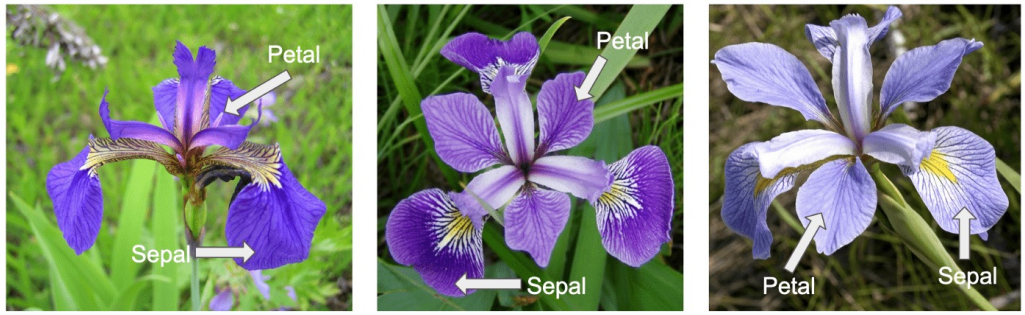


Source: http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/

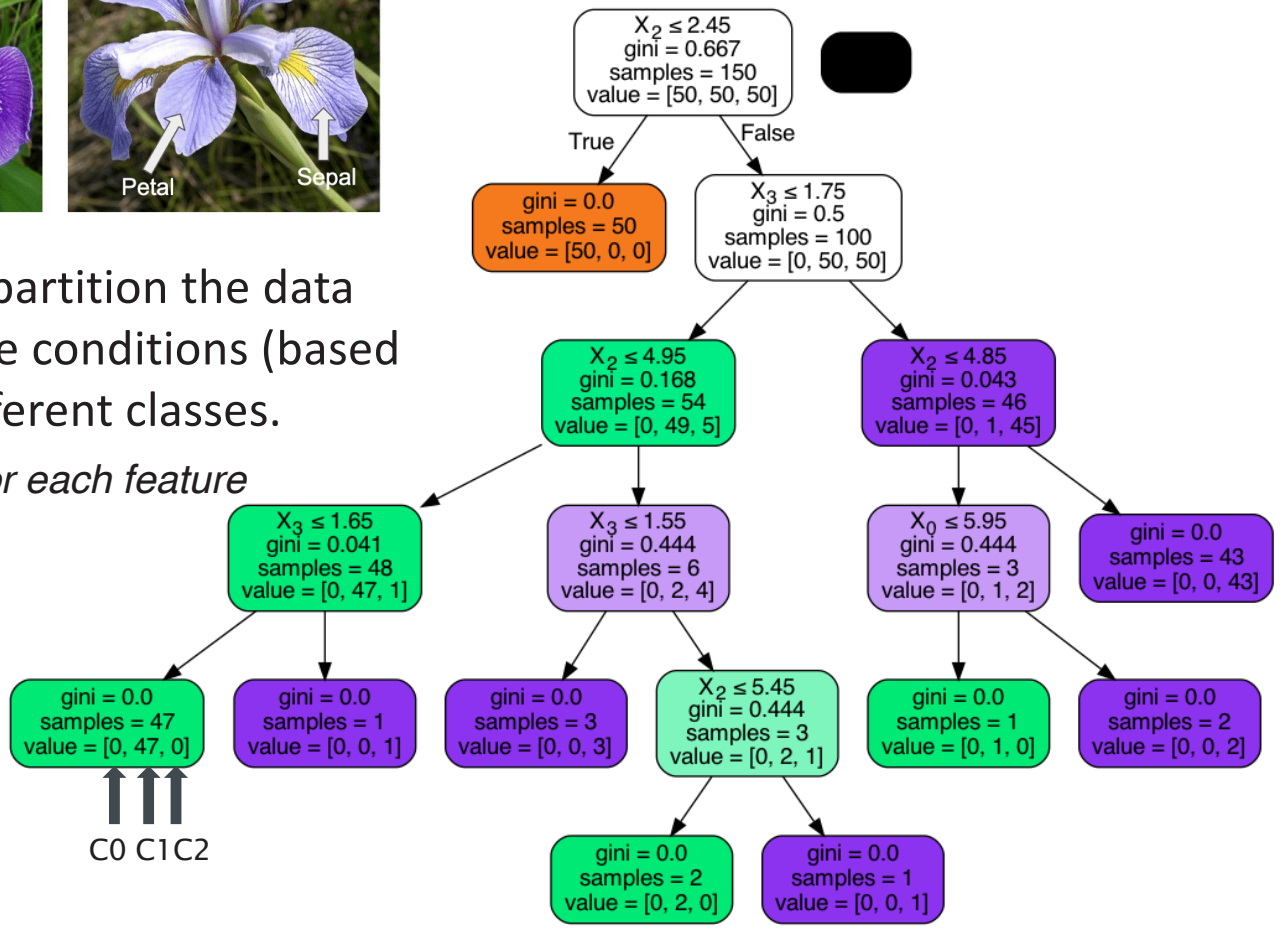
Credit: Zhaoxia Wang

Decision Trees – **Overview (2/3)**

Iris setosa *Iris versicolor* *Iris virginica*



A decision tree is like a flow chart
For example, a tree on the Iris dataset



The objective is to efficiently partition the data based on the most informative conditions (based on features) that separate different classes.

***Key idea**: Greedily find best split for each feature*

C0: 5
C1: 5

Non-homogeneous,

High degree of impurity

C0: 9
C1: 1

Homogeneous,

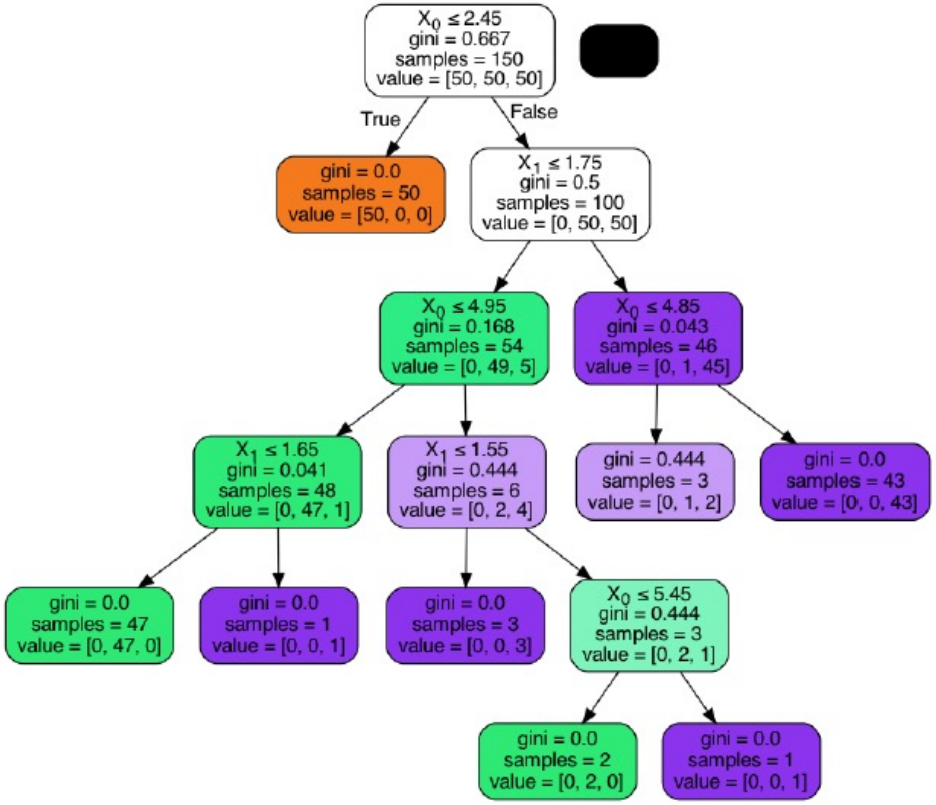
Low degree of impurity

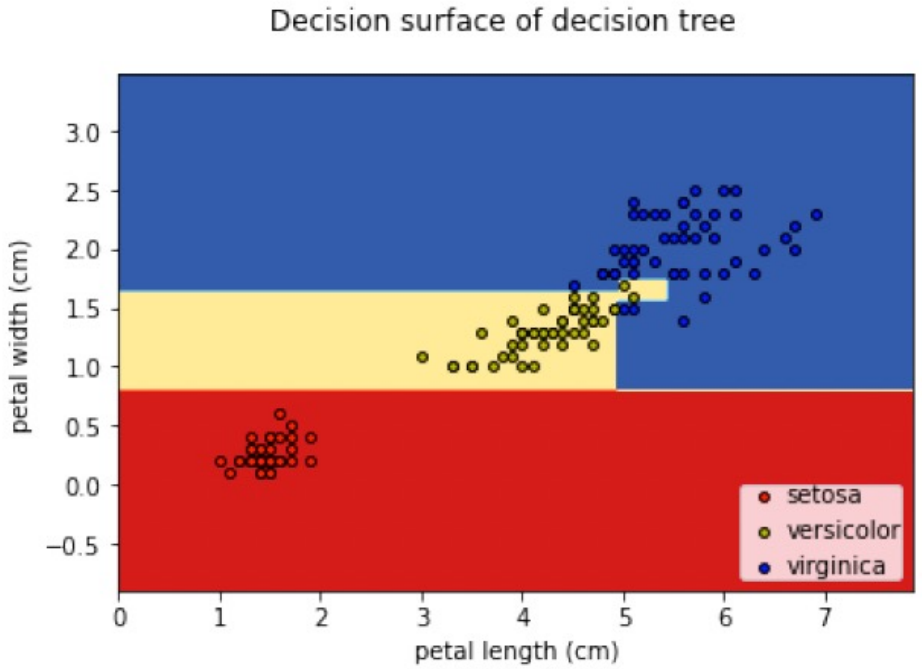
Four features, only three are used here, and one is only used once.

Credit: Jo Grundy

Decision Trees – **Overview (3/3)**

2D iris dataset:





Credit: Jo Grundy

Decision Trees – **Learning Outcomes**

- **LO1**: Demonstrate an understanding of decision tree fundamentals, such as (exam)
  - Calculating impurity and constructing a decision tree using algorithms like ID3, C4.5, etc, given a dataset and distance metric
  - Addressing overfitting in decision trees like pruning
  - Understanding ensemble methods using decision trees like bagging, boosting, random forests
- **LO2**: **Implement** the learned algorithms using decision trees (course work)

Assessment hints: Multi-choice Questions (single answer: concepts, calculation etc)

- *Textbook Exercises: textbooks (Programming + Mining)*
- *Other Exercises: https://www-users.cse.umn.edu/~kumar001/dmbook/sol.pdf*
- *ChatGPT or other AI-based techs*

Decision Trees – Introduction

Decision Trees can be 'hand crafted' by experts
They can also be built up using machine learning techniques

They are **interpretable**, it is easy to see how they made a certain decision

They are used in a wide range of contexts, for example:

- Medicine
- Financial Analysis
- Astronomy

Especially in medicine, the explicit reasoning in decision trees means experts can understand why the algorithm has made its decision.

Credit: Jo Grundy

Decision Trees – CART

The CART algorithm was published by Breiman *et al* in 1984

- Find best split for each feature - minimises impurity measure
- Find feature that minimises impurity the most
- Use the best split on that feature to split the node
- Do the same for each of the leaf nodes

The CART algorithm depends on an impurity measure. It uses Gini impurity

Gini impurity measures how often a randomly chosen element from a set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the set. The probabilities for each label are summed up.

Credit: Jo Grundy

Decision Trees – CART

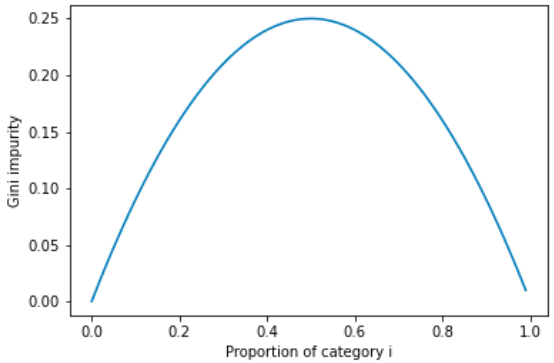
- Gini Index for a given node t :

The smaller the purer

$$GINI(t) = 1 - \sum_{j}^{n_c} [p(j|t)]^2$$

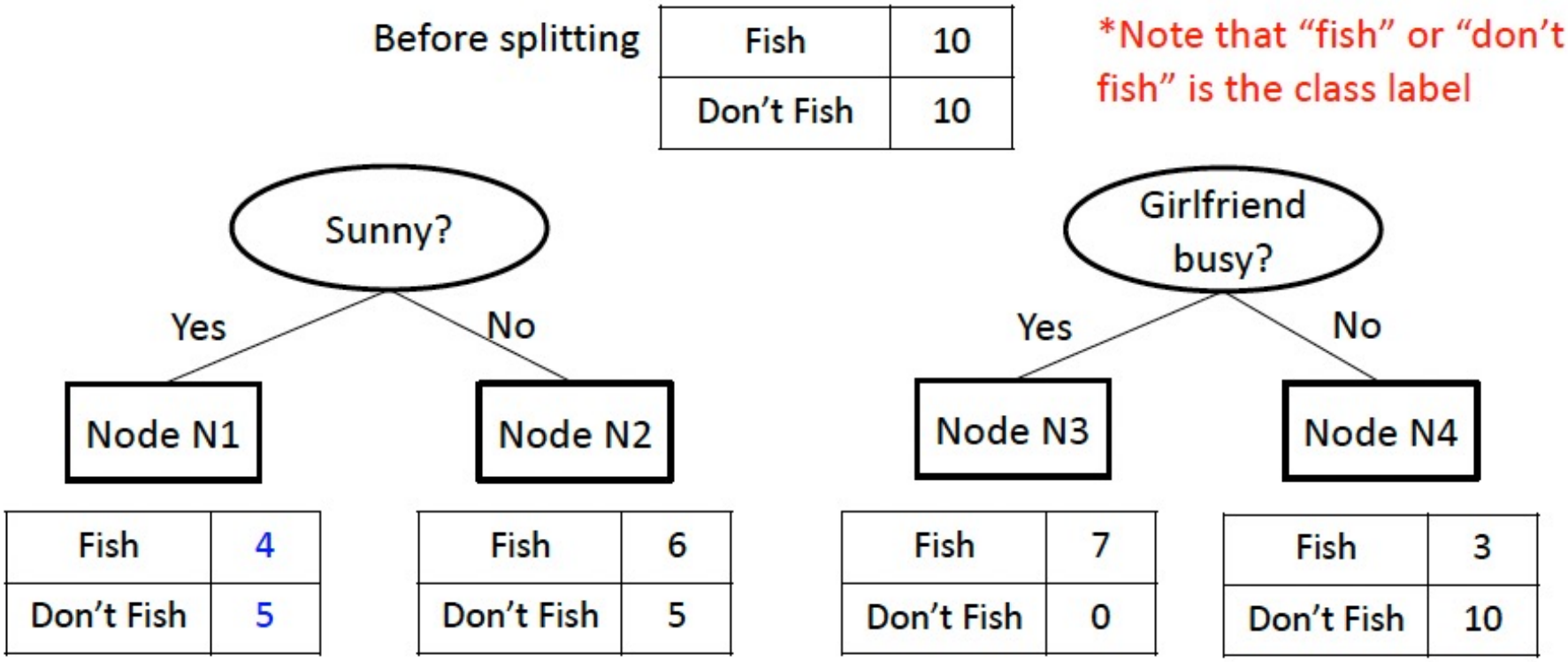
(where: n_c *is the number of classes, and p(j | t)* is the relative frequency of class j at node t).

  - Maximum: (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
  - Minimum: (0.0) when all records belong to one single class, implying most interesting information



It reaches its minimum when all cases in the node fall into a single category

Decision Trees – CART



Using Gini Index, evaluate which test condition is better

Credit: Zhaoxia Wang

Decision Trees – CART

$$GINI(t) = 1 - \sum_{j}^{n_c} [p(j|t)]^2$$

Before splitting

Fish	10
Don't Fish	10

Gini = 1 – [p(Fish)² + p(Don't Fish)²]

= 1 – [(10/20)² + (10/20)²]

= 1 – (0.25 + 0.25) = 0.5

Split by "Sunny?"

Yes

Fish	4
Don't Fish	5

P(Fish) = 4/9 P(Don't Fish) = 5/9

Gini = 1 – [(4/9)² +(5/9)²]= 0.494 updated value

No

Fish	6
Don't Fish	5

P(Fish) = 6/11 P(Don't Fish) = 5/11

Gini = 1 – [(6/11)² +(5/11)²]= 0.496

Question: How to aggregate the branch-based values for the whole split?

Credit: Zhaoxia Wang

Decision Trees – CART

- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i) \quad (1)$$

where k is number of branches

$GINI(i)$ is the i -th branch-based GINI

n_i = number of records at child i,

n = number of records at node p.

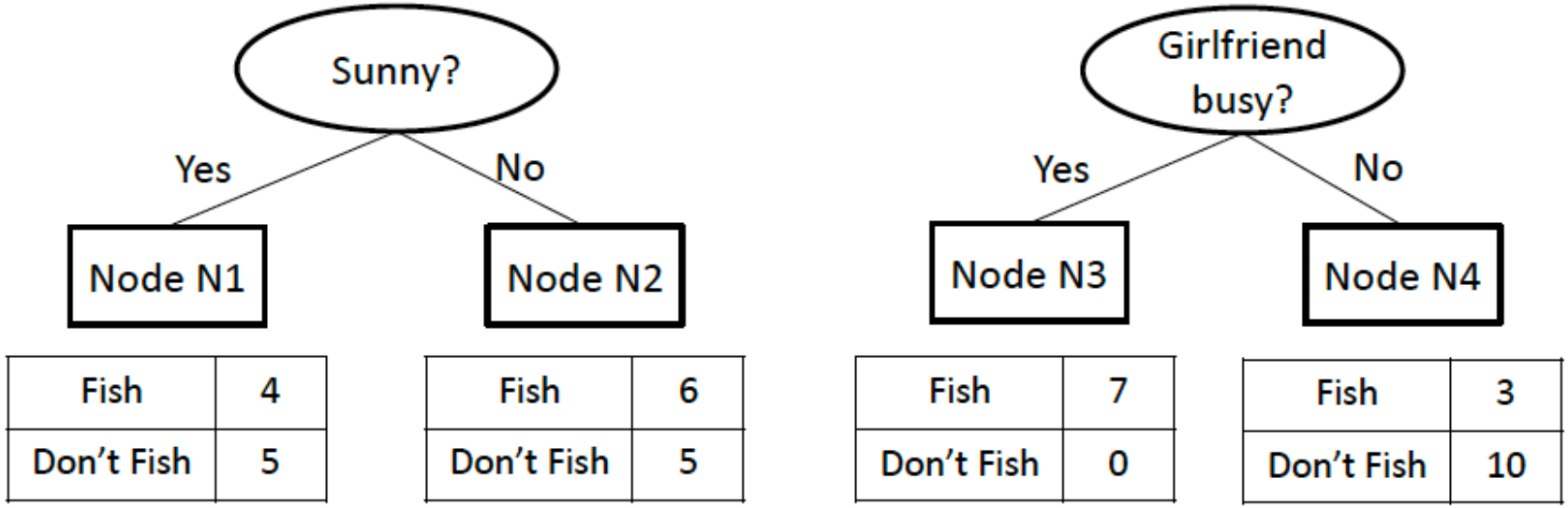
Information gain: $Gain_{split} = GINI_{original} - GINI_{split}$,
The higher gain, the better split

Credit: Zhaoxia Wang

Decision Trees – CART

Before splitting		
	Fish	10
	Don't Fish	10

*Note that "fish" or "don't fish" is the class label



Fish	4	Fish	6	Fish	7	Fish	3
Don't Fish	5	Don't Fish	5	Don't Fish	0	Don't Fish	10

Gini(N1) = 0.494 (updated value)

Gini(N2) = 0.496

Gini(N3) = 0.0

Gini(N4) = 0.355

$Gini_{sunny}$ = (9/20 * 0.494) + (11/20 * 0.496) = 0.495

$Gini_{girlfriend}$ = (7/20 * 0.0) + (13/20 * 0.355) = 0.23

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i) \quad (1)$$

$$Gain_{split} = GINI_{original} - GINI_{split} = 0.5 - GINI_{split}$$

Credit: Zhaoxia Wang

Decision Trees – ID3

Similar to CART, Iterative Dichotomy 3 (ID3) minimises entropy

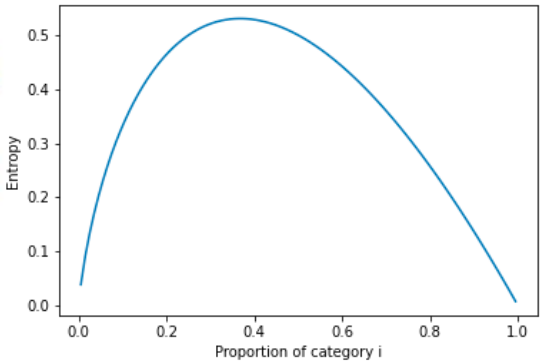
- Entropy at a given node t:

The smaller the purer

$$Entropy(t) = -\sum_{j=1}^{n_c} p(j|t) \log_2 p(j|t) \quad (1)$$

(where: n_c *is the number of classes, and p(j | t)* is the relative frequency of class j at node t).

  - Measures homogeneity of a node.
    - Maximum ($\log_2 n_c$) when records are equally distributed among all classes implying the **least** information
    - Minimum (0.0) when all records belong to one class, implying the **most** information
  - Entropy based computations are similar to the GINI index computations



Decision Trees – ID3

University of Southampton

The higher the better

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)\right) \quad (1)$$

where $Entropy(p)$: entropy of Parent Node, $Entropy(i)$: entropy of the i-th branch/partition, n is split into k partitions; n_i is number of records in partition i

  - Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3 and C4.5

Decision Trees – CART

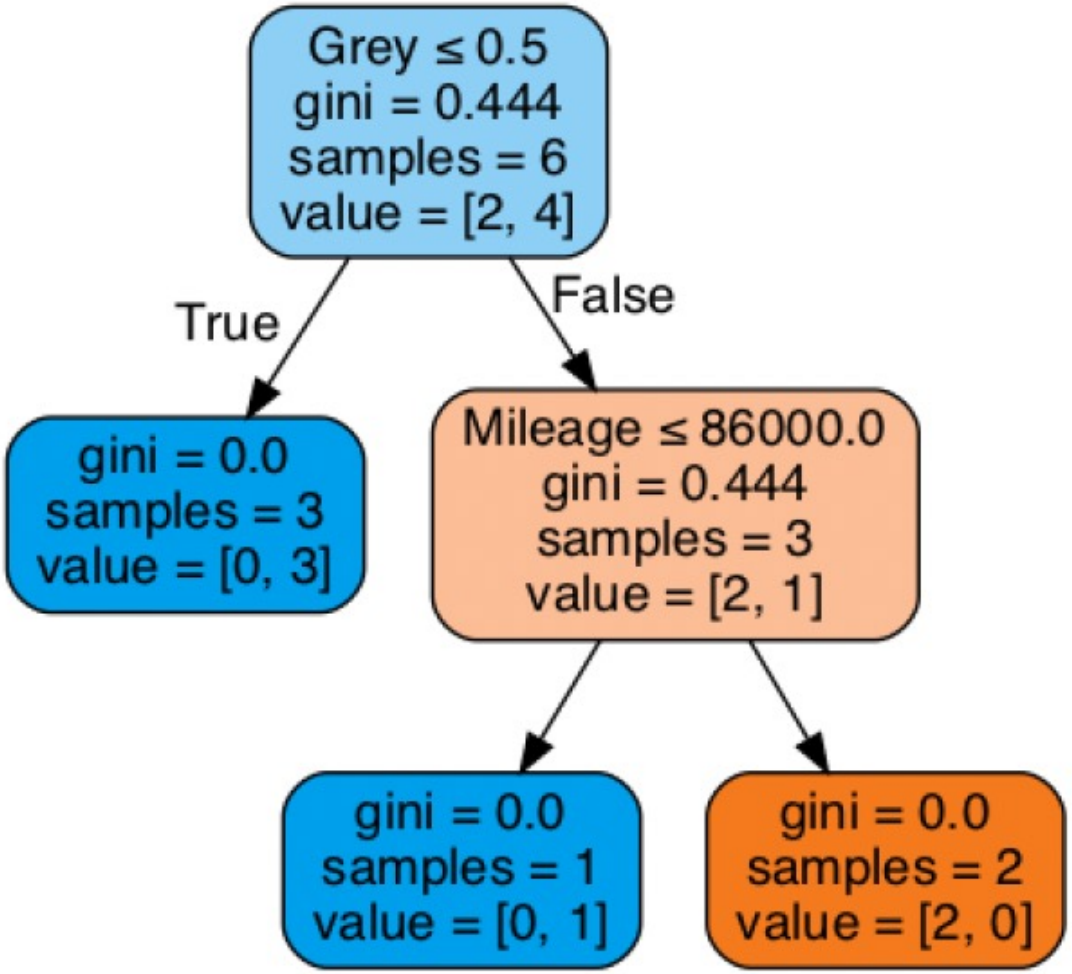
Example:

Car Make	Type	Colour	Price	Mileage	Bought?
VW	Polo	Grey	£2000	82000	Yes
Ford	Fiesta	Purple	£1795	95000	Yes
Ford	Fiesta	Grey	£1990	90000	No
VW	Golf	Red	£1800	120000	Yes
VW	Polo	Grey	£900	150000	No
Ford	Ka	Yellow	£1400	100000	Yes

Can go through and calculate best split for each feature.

Credit: Jo Grundy

Decision Trees – CART

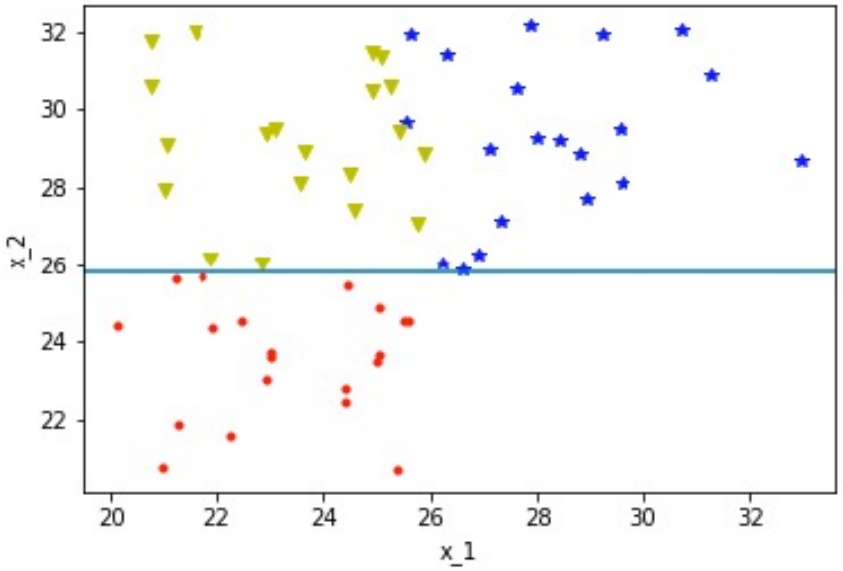


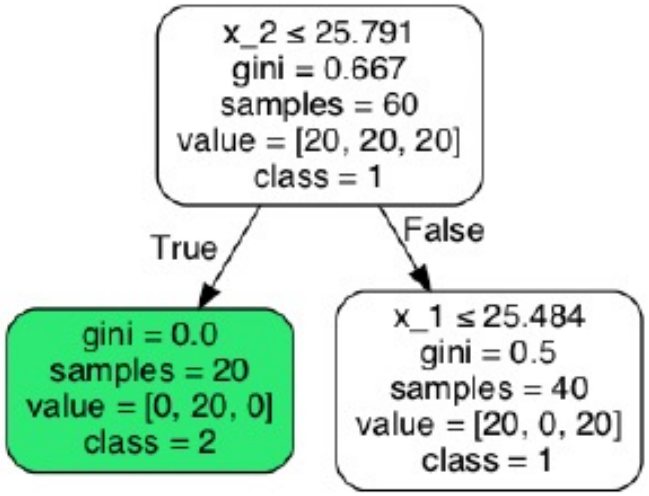
ipynb demo

Credit: Jo Grundy

Decision Trees – CART

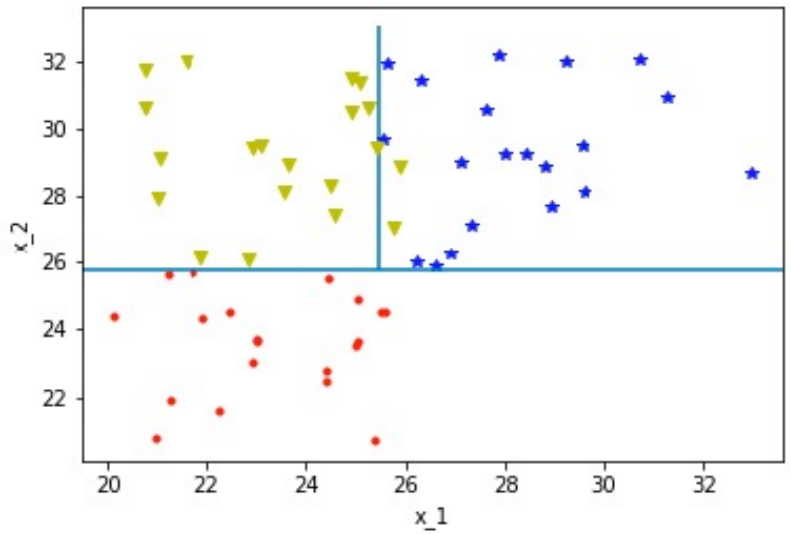
With three classes:

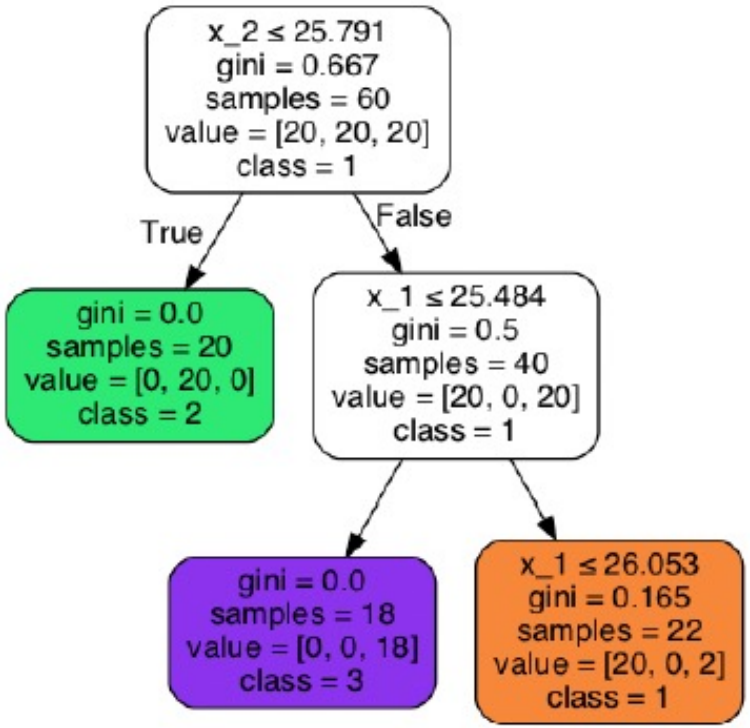




Credit: Jo Grundy

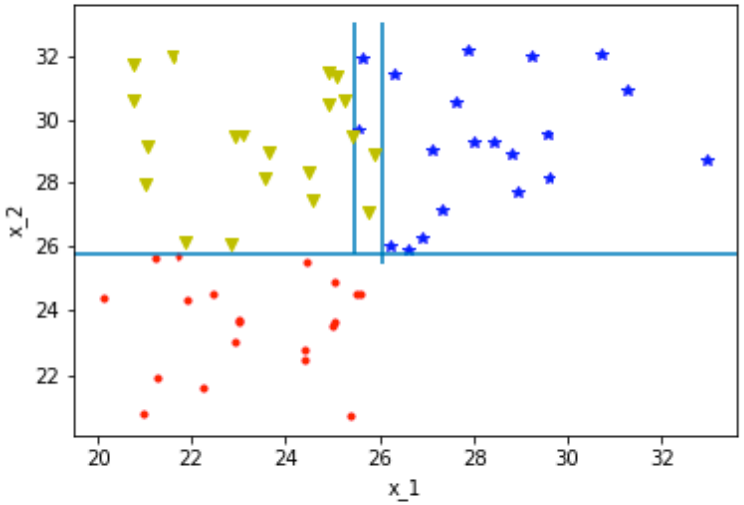
Decision Trees – CART

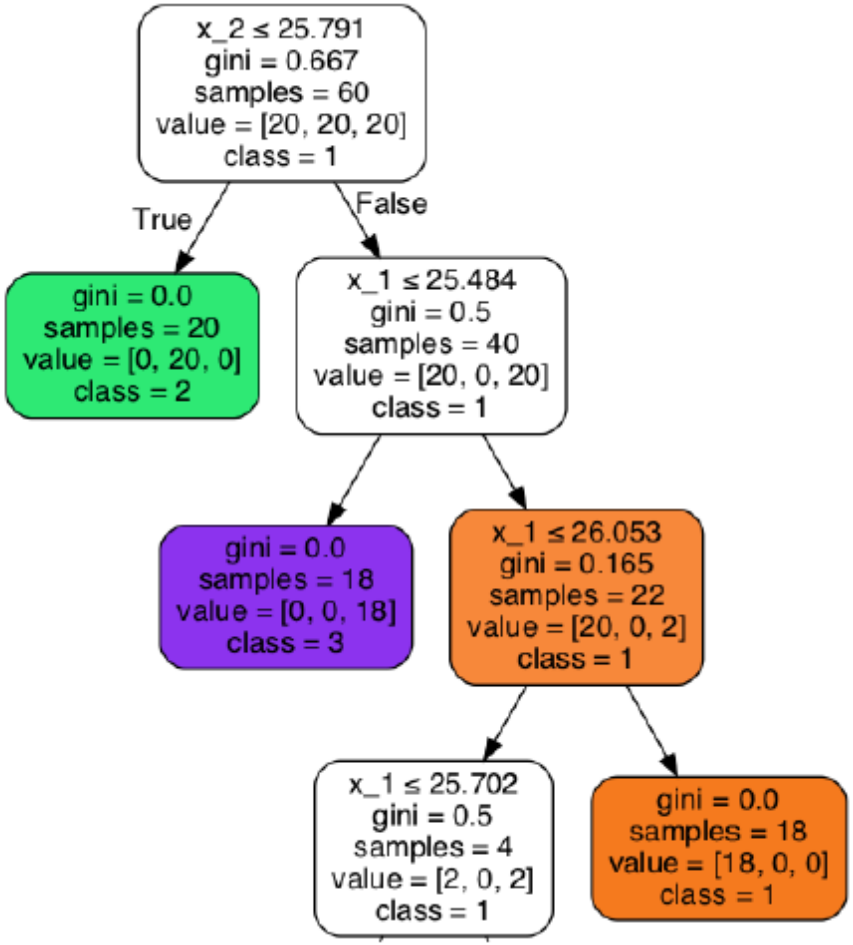




Credit: Jo Grundy

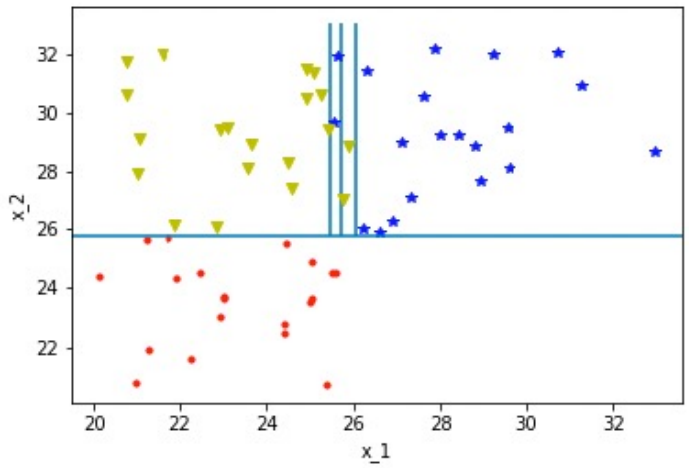
Decision Trees – CART

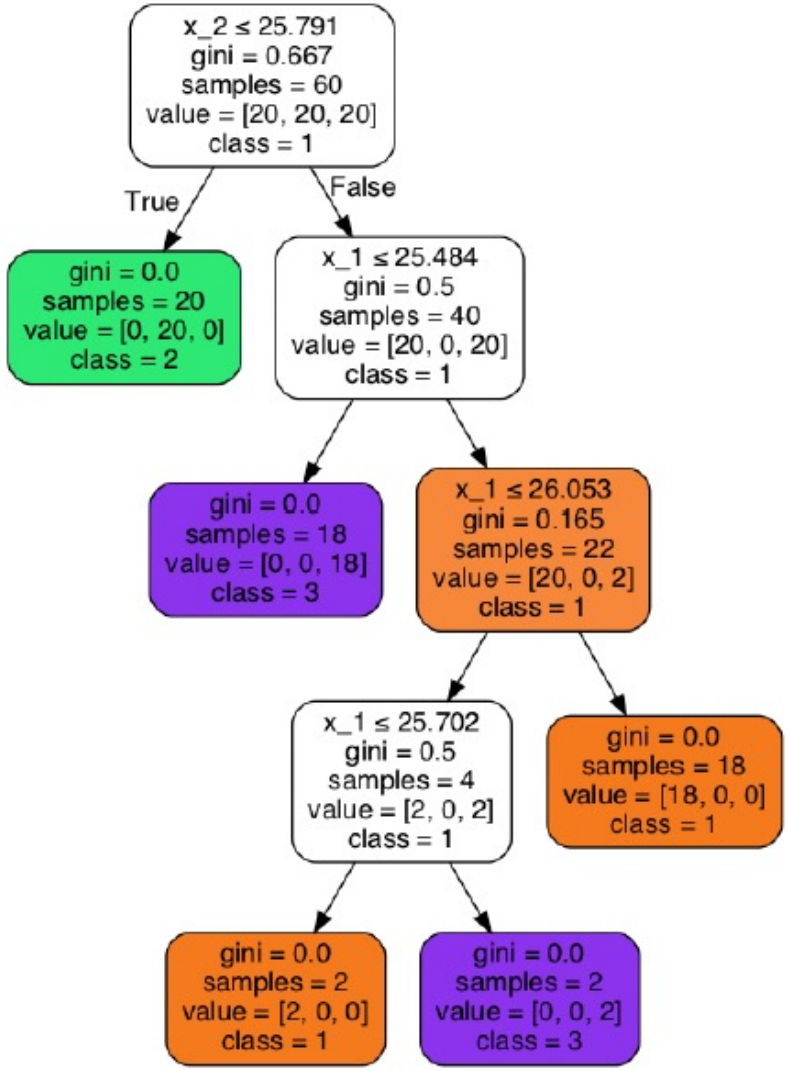




Credit: Jo Grundy

Decision Trees – CART





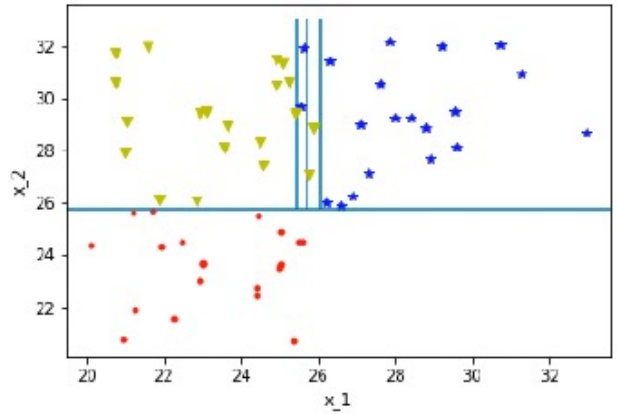
Overfitting..

Credit: Jo Grundy

Decision Trees – Address Overfitting
Pruning

University of Southampton

Overfitting is a serious problem with Decision Trees



.. are four splits really required here?

The trees it creates are too complex.

One solution is **pruning**

This can be done in a variety of ways, including:

- Reduced Error Pruning
- Entropy Based Merging

Credit: Jo Grundy

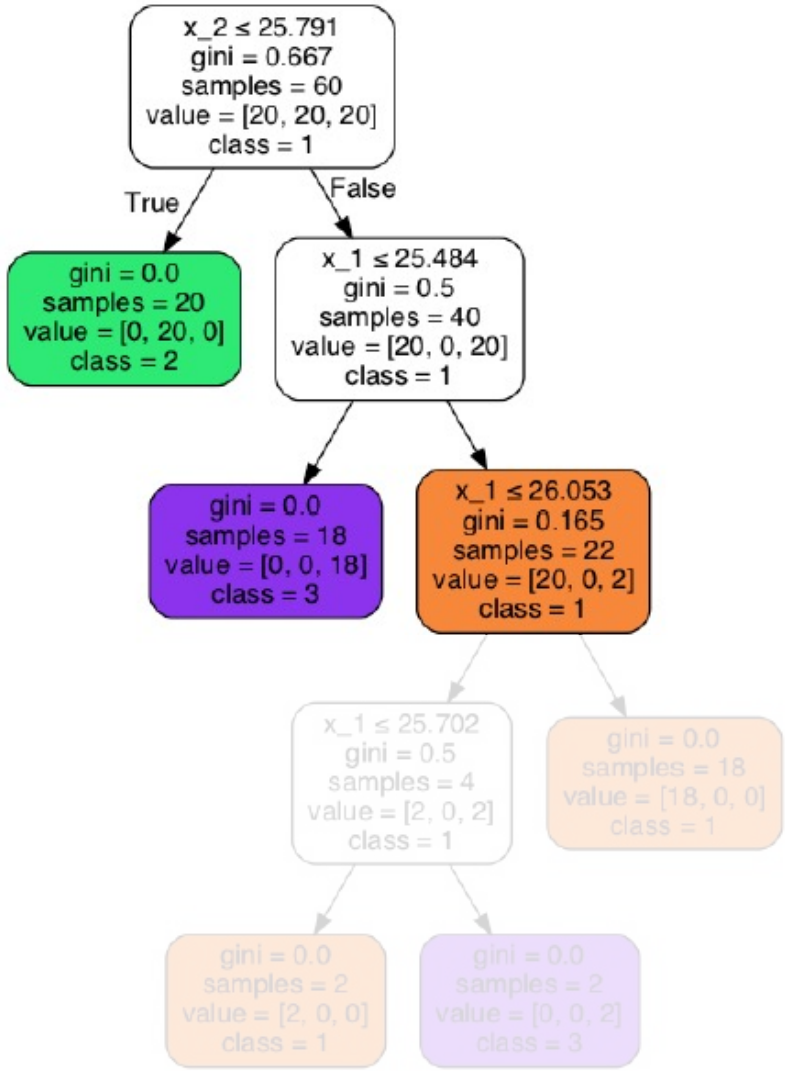
Decision Trees – Address Overfitting
Reduced Error Pruning

University of Southampton

Growing a decision tree fully, then removing branches without reducing predictive accuracy, measured using a *validation set*.

- Start at leaf nodes
- Look up branches at last decision split
- replace with a leaf node predicting the majority class
- If validation set classification accuracy is not affected, then keep the change

This is a simple and fast algorithm that can simplify over complex decision trees



Credit: Jo Grundy

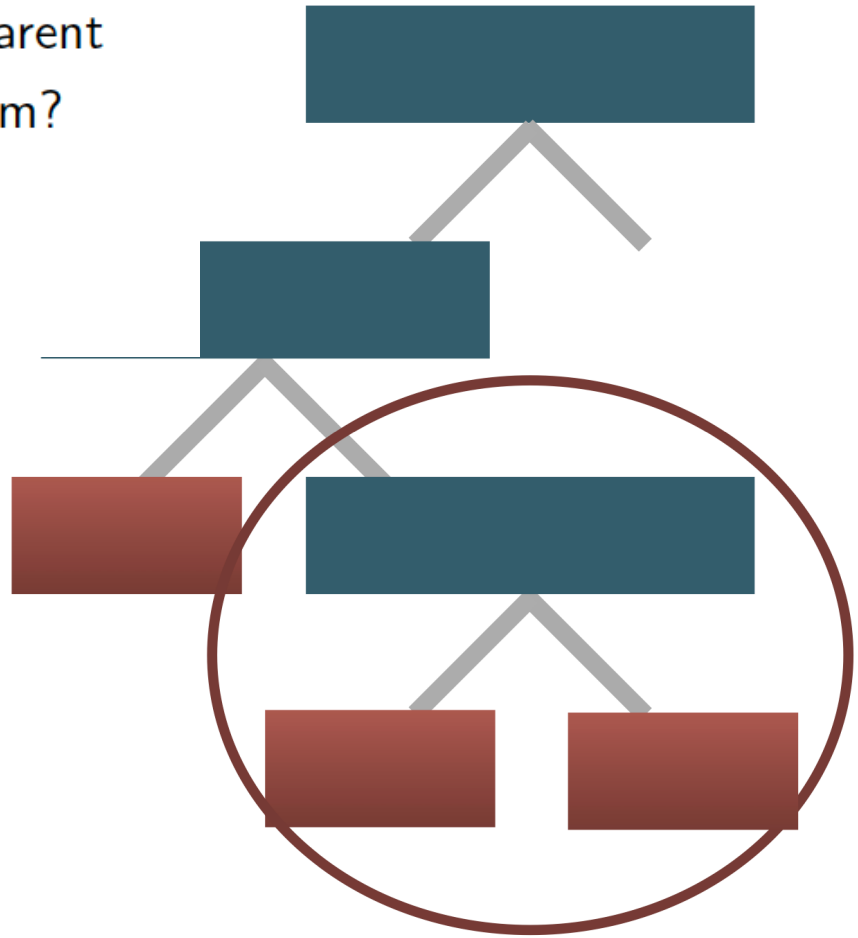
Decision Trees – Address Overfitting
Entropy Based Pruning

University of Southampton

Grow a decision tree fully, then

- Chose a pair of leaf nodes with the same parent
- What is the entropy gain from merging them?
- If lower than a threshold, merge nodes

This doesn't require additional data



Credit: Jo Grundy, Jon Hare

Decision Trees – Address Overfitting
Missing Data

University of Southampton

ID3 ignores missing data, CART generally puts them to the node that has the largest number of the same category

- You can assign a branch specifically to an unknown value
- You can assign it to the branch with the most of the same target value
- You can weight each branch using the known factors and put it in the most similar branch

Referrer	Location	Read FAQ	Pages viewed	Service chosen
Slashdot	USA	Yes	?	None

Credit: Jo Grundy, Jon Hare

Decision Trees – Address Overfitting
Regression

University of Southampton

CART - Classification and Regression Trees How do we use decision trees for regression? i.e. to give numerical values rather than a classification.

Could use classification, but using each numerical value as a class..

Problems?

Credit: Jo Grundy

Decision Trees – Address Overfitting
Regression

University of Southampton

CART - Classification and Regression Trees How do we use decision trees for regression? i.e. to give numerical values rather than a classification.

Could use classification, but using each numerical value as a class..

Problems?

- How would you generalise?
- Loses all meaning of ordering, or similarity

Solution?

Credit: Jo Grundy

Decision Trees – Address Overfitting
Regression

University of Southampton

Maximise Variance Gain:

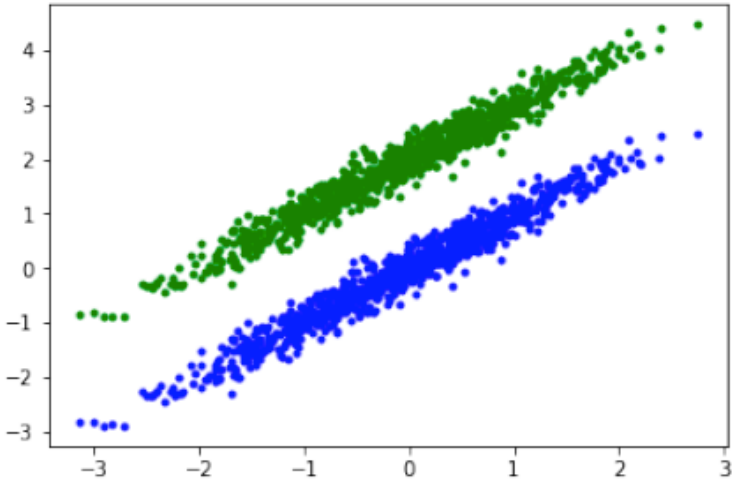
- Split on the feature values that give maximum gain in variance
- Should make similar numbers group together
- I. e. lower numbers on one side, higher on the other

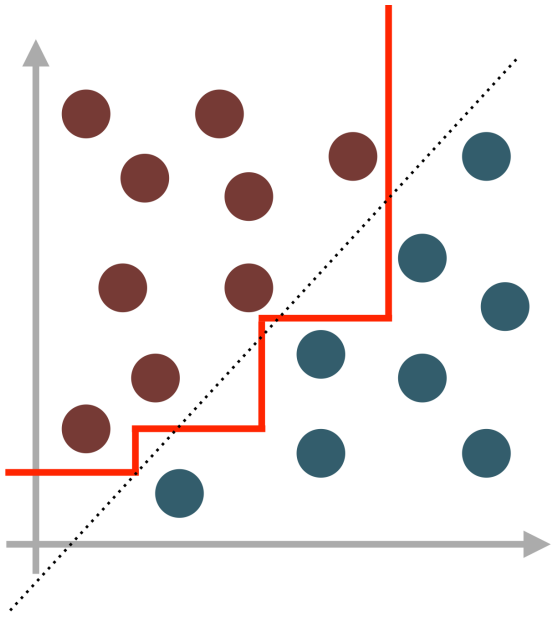
Credit: Jo Grundy

Decision Trees – Address Overfitting

There are problems with Decision Trees:

- Finding an optimal Tree is NP-complete
- They overfit, so don't generalise well - Hence need to prune
- Information Gain is biased to features with more categories
- Splits are axis aligned..





Credit: Jo Grundy, Jon Hare

Decision Trees – Ensemble methods

Bagging: Bootstrap aggregating
Uniformly sample initial dataset with replacement in to m subsets
For example:
if data set has 5 samples, $(s_1, s_2, s_3, s_4, s_5)$
make a whole bunch of similar data sets:

- $(s_5, s_2, s_2, s_1, s_5)$
- $(s_4, s_2, s_1, s_3, s_3)$
- $(s_2, s_5, s_3, s_1, s_1)$
- $(s_3, s_1, s_2, s_4, s_4)$

Train a different decision tree on each set
To classify, apply each classifier and chose the correct one by majority vote

Credit: Jo Grundy

Decision Trees – Ensemble methods

Boosting - Kearns and Valiant (1988):
"Can a set of weak learners create a single strong learner?"

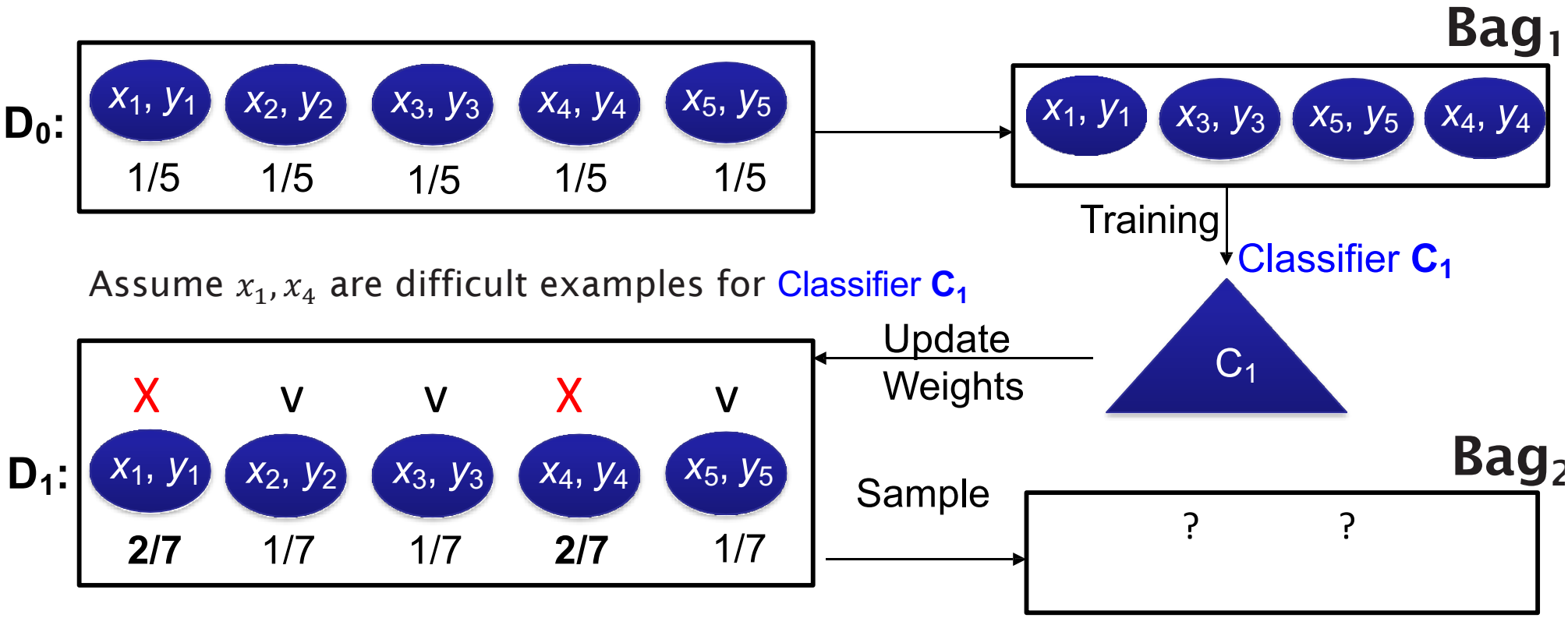
We make a weighted sum of very weak learners
- so long as they all learn different things then it works!
AdaBoost:

- Train a weak learner on one feature
- See what it does well on
- Weight the remaining data more
- repeat

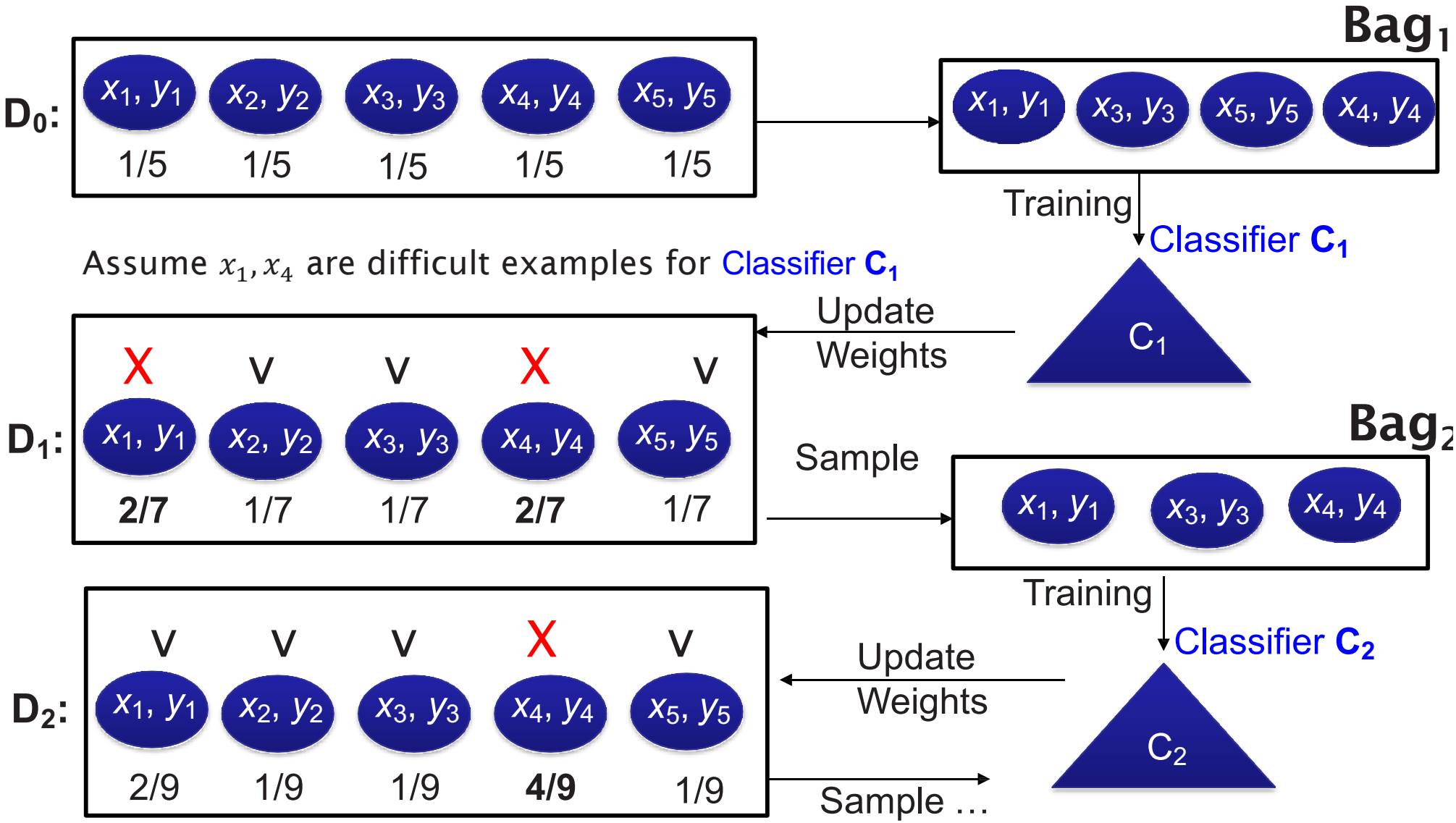
This makes a series of weak learners that have learned how to use different features to discriminate between classes.

Credit: Jo Grundy

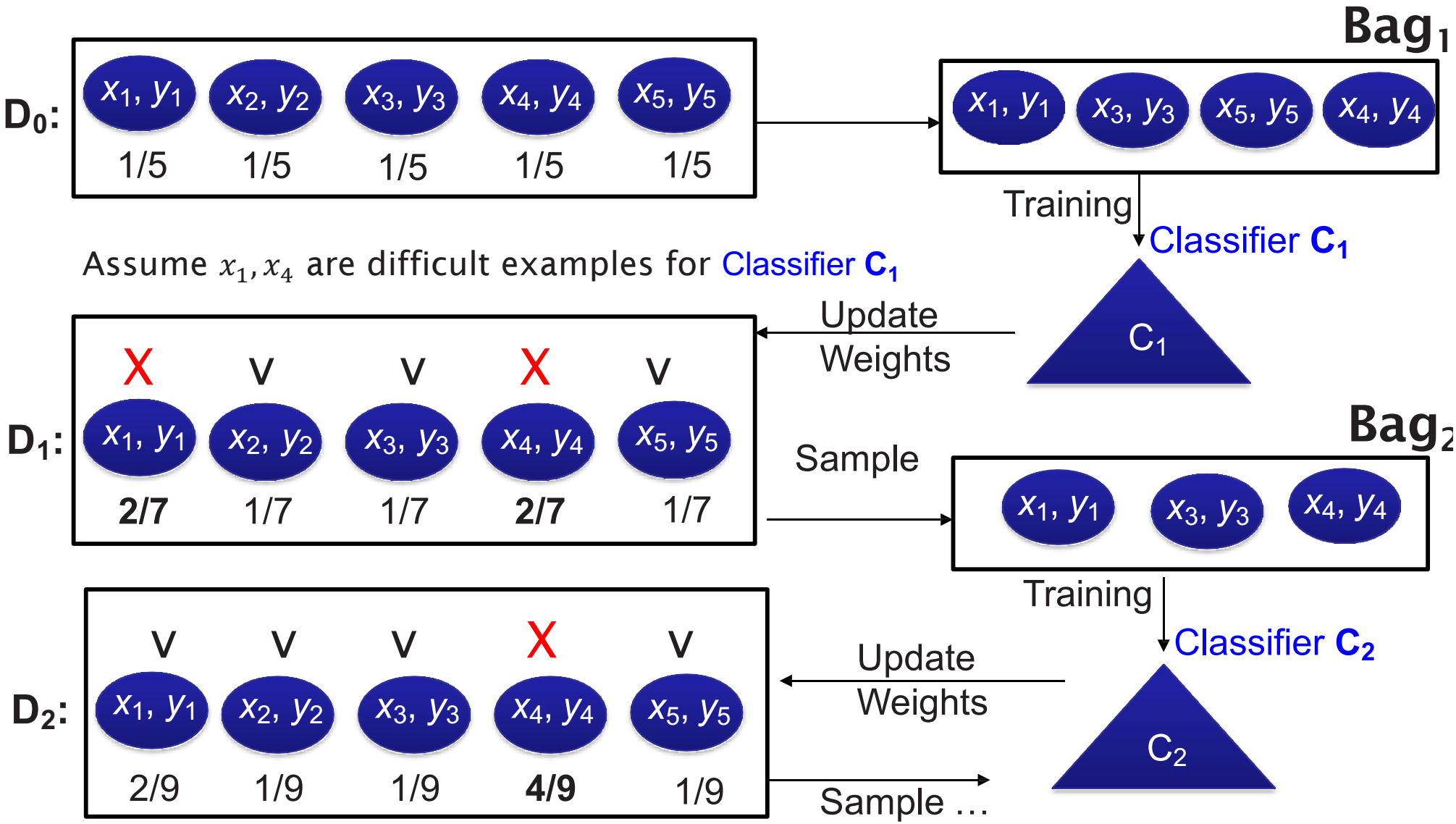
Decision Trees – Ensemble methods



Decision Trees – Ensemble methods



Decision Trees – Ensemble methods



Final Decision: $w_1 * C_1 + w_2 * C_2 + \cdots$, where w_i is the weight for C_i, proportional to its classification performance

Decision Trees – Ensemble methods

Gradient boosting with trees - Friedman (1999):
Generalise Adaboost to Gradient boosting to handle any loss function
Shortcomings are where the residuals are larger
So fit a tree to the residuals:

- $x_1, y_q - f(x_1)$
- $x_2, y_q - f(x_1)$
- $x_3, y_q - f(x_1)$
- $\vdots$
- $x_n, y_q - f(x_n)$

more detail available at

http://www.chengli.io/tutorials/gradient_boosting.pdf

Credit: Jo Grundy

Decision Trees – Ensemble methods

Random Forests

Apply bagging
but when learning the tree for each subset, chose the split by searching over a random sample of the features

Reduces overfitting

Credit: Jo Grundy

Decision Trees – Summary

Advantages:

- Interpretability
- Ability to work with numerical and categorical features
- Good with mixed tabular data

Disadvantages:

- Might not scale effectively for lots of classes
- Features that interact are problematic

Credit: Jo Grundy