CS430/910: Foundations of Data Analytics

Classification | Dr Greg Watson

Objectives

- To understand the concept of classifiers and their construction
- See how decision tree classifiers are constructed
- Statistical classifiers: Naive Bayes
- Linear separation via SVM classifiers
- Construct and compare classifiers using Weka

Part A: Classification: Introduction 1

Classification

- Regression concerns creating a model for **numeric values**.
- Classification is about creating models for **categoric values**.
- Many different approaches to classification.
- *The* core problem in machine learning (see CS342, CS429/CS909).

Classification

Uses

- Many different uses for classification:
  - Use classification to fill in missing values
  - Predict future values based on partial information
  - …

Classification

Applications

- Many applications of classification in practice:
  - Determine whether a loan is likely to be repaid (credit scoring)
  - Fraud detection: identify when a transaction is fraudulent
  - Medical diagnosis: initial classification of test results
  - Identify text from writing (optical character recognition)
- Determine whether an email is spam (spam filter)
- Predict whether someone will like a product (collaborative filtering)
- Determine what is present in an image (object detection)
- and many more...

Classification

- Many different choices in how to model data.
- Central concept: **data = model + error**
    - Attempt to extract a model from data (error is unknown)
    - Error comes from noise, measurement error, random variation
    - We try to avoid modelling error: leads to **overfitting**
    - Overfitting possible when model has too many parameters
    - The model only describes the data and does not **generalise**
    - Converse problem is underfitting: model cannot capture behaviour

Classification

Overfitting

- ***Overfitting***: Fitting a particular dataset very closely, and therefore possibly failing to fit or generalise to other data reliably.
- Contains more parameters than necessary.
- Figure to the right: Black line represents a good fit, green line represents an overfitted model.

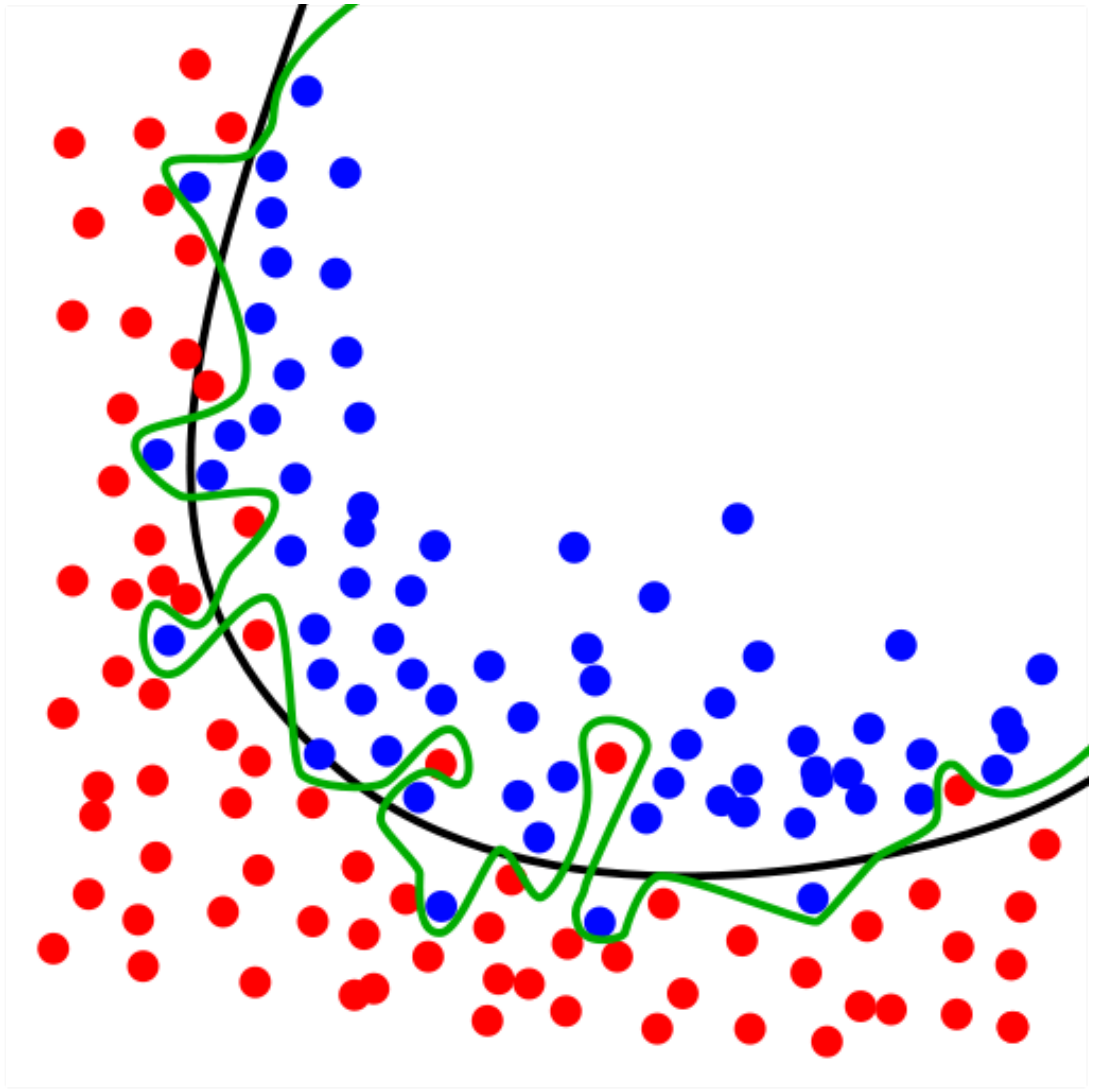
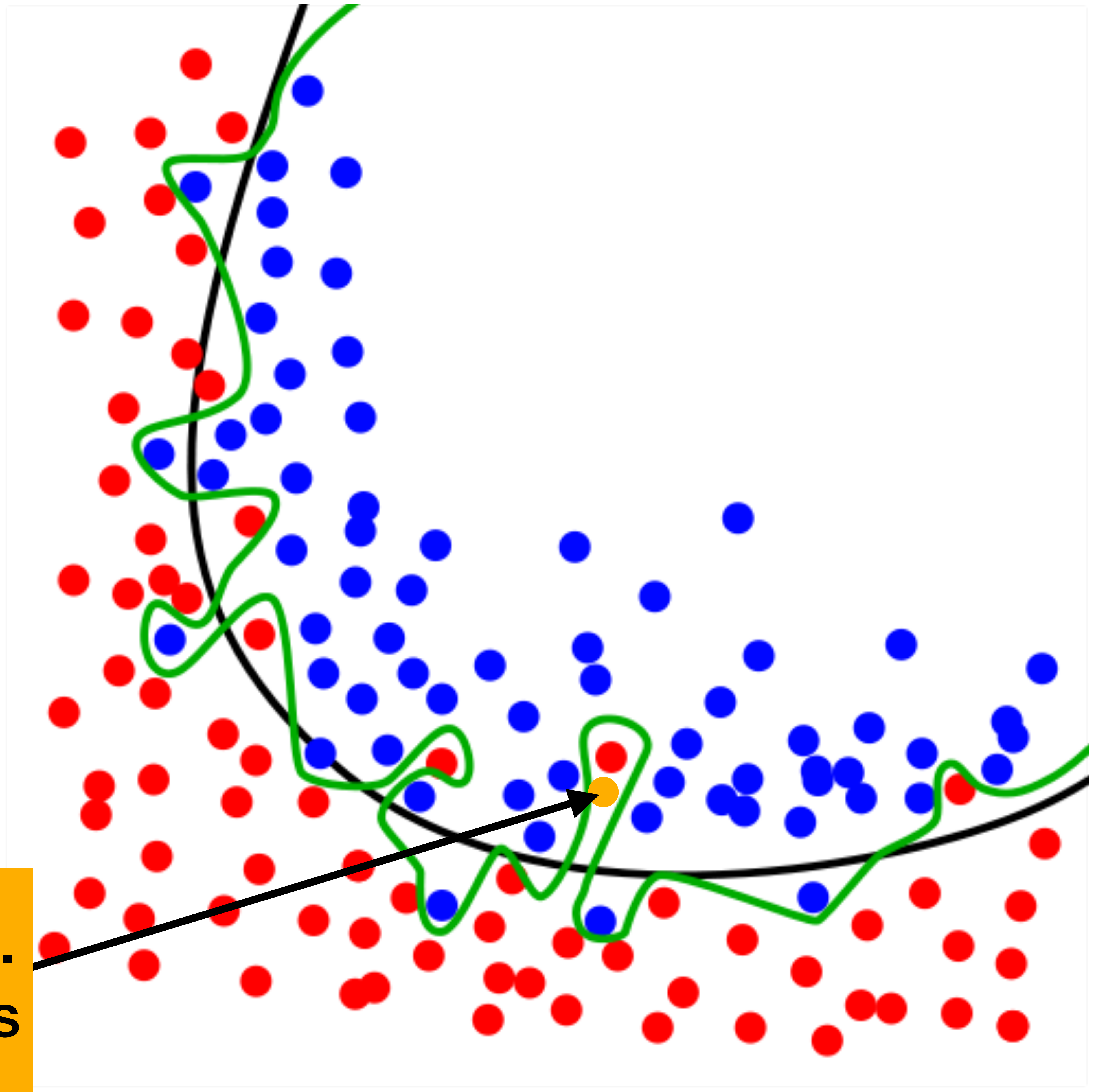


Figure by: Chabacano, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Classification

Overfitting

- ***Overfitting***: Fitting a particular dataset very closely, and therefore possibly failing to fit or generalise to other data reliably.
- Contains more parameters than necessary.
- Figure to the right: Black line represents a good fit, green line represents an overfitted model.



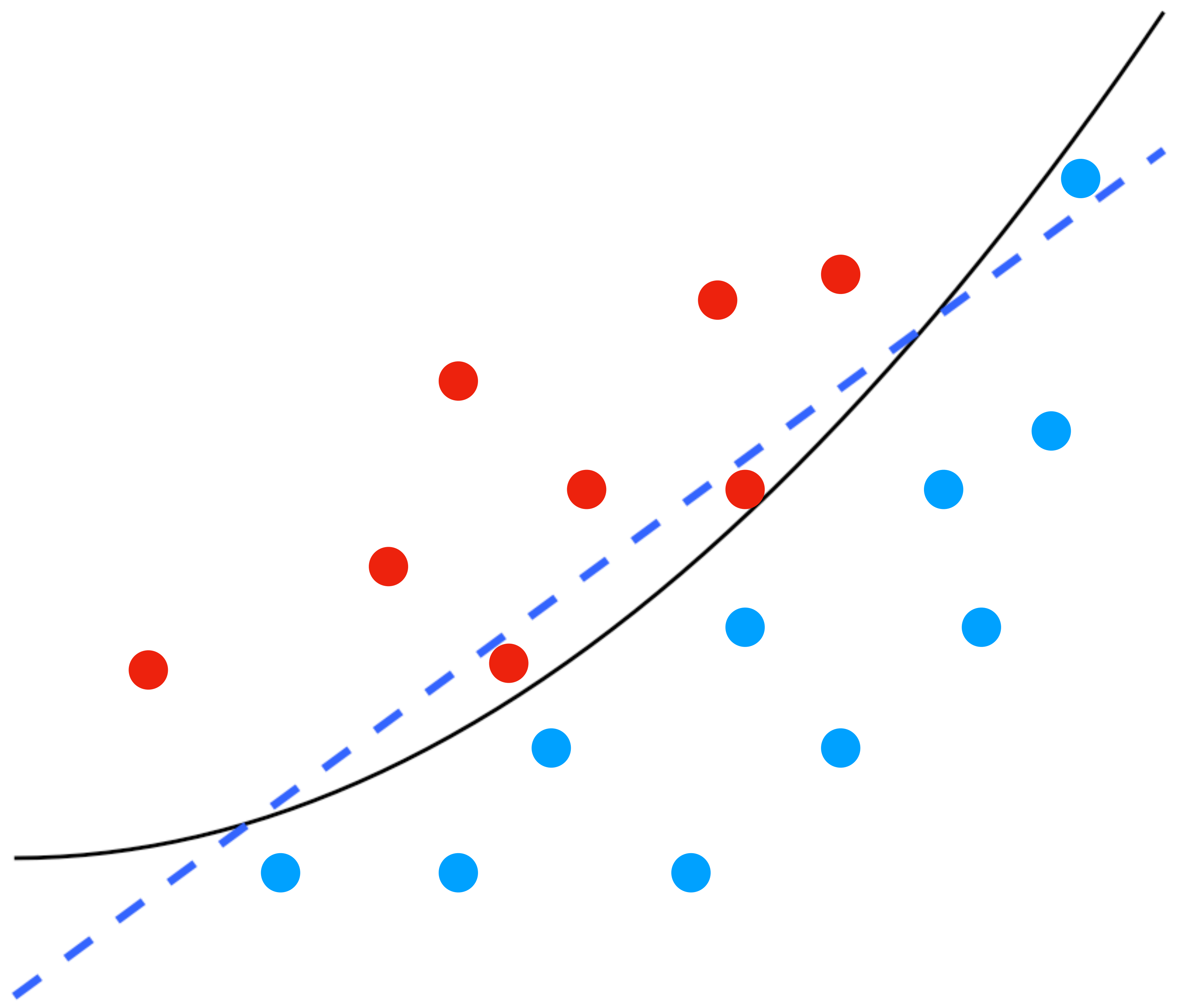
Consider a new point here. How would the two models classify this point?

Figure by: Chabacano, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Classification

Underfitting

- Underfitting is the opposite:
  - The model has not captured the underlying structure of the data adequately.
  - Example: fitting a linear model to non-linear data.
- Figure to the right: Blue line represents a model which has underfit.

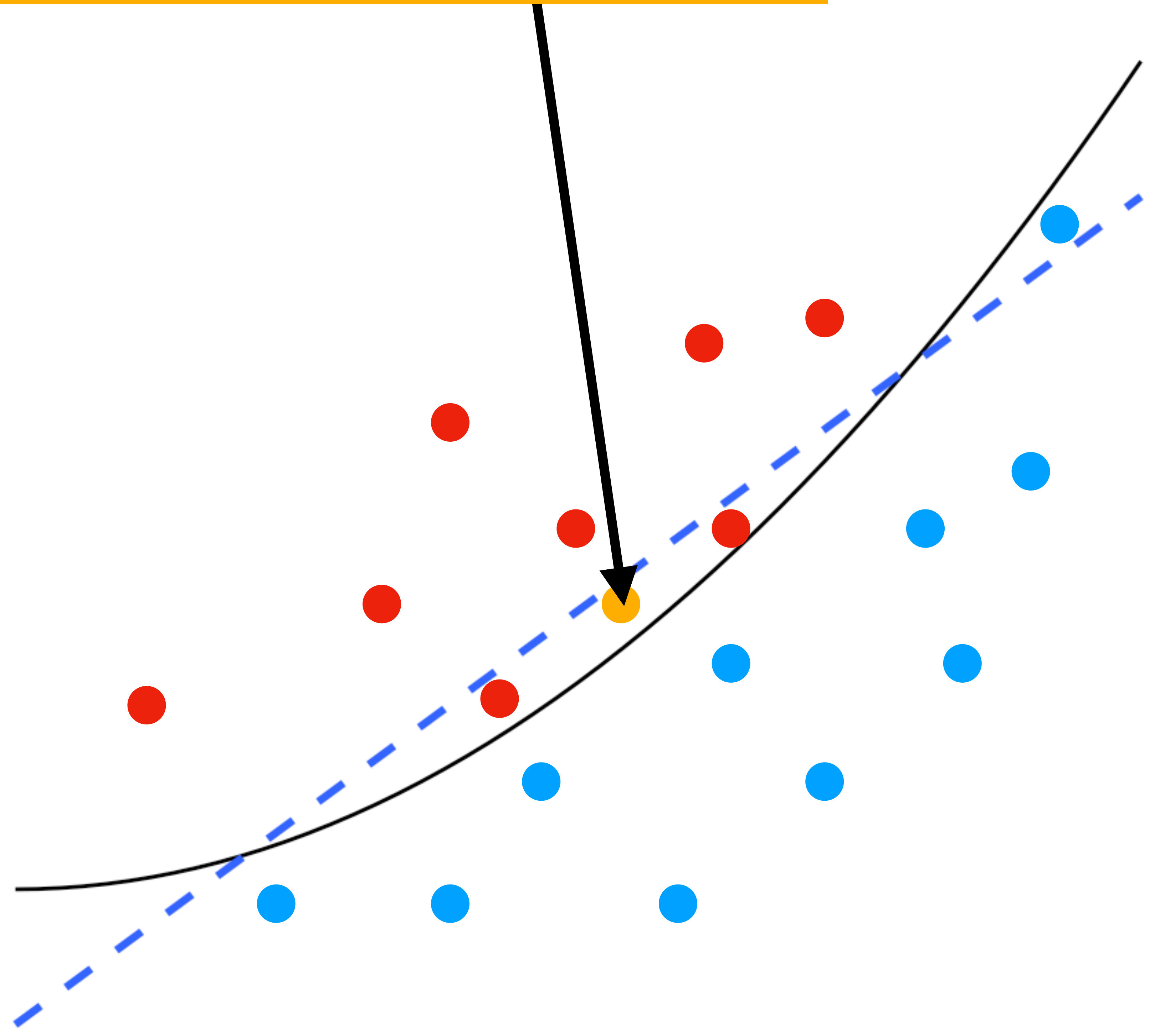


Classification

Underfitting

- Underfitting is the opposite:
    - The model has not captured the underlying structure of the data adequately.
    - Example: fitting a linear model to non-linear data.
- Figure to the right: Blue line represents a model which has underfit.

Consider a new point here. How would the two models classify this point?



Classification Process

Three Phases

- Phase 1: *Training*
    - Obtain a data set with examples and a target attribute (class)
    - Use **training data** to build a model of the data
- Phase 2: *Evaluation*
    - Apply the model to **test data** with hidden class value
    - Evaluate quality of model by comparing prediction to hidden value
    - Accept model, or reject if not sufficiently accurate

Classification Process

Three Phases

- Phase 3: *Application*
  - Apply the model to new data with unknown class value
  - Monitor accuracy: look for concept drift (Concept drift: properties of target class change over time).

Approaches to Evaluation

- Many approaches to evaluation of classifier
- Apply classifier to training data (test data = training data)
    - Risk of overfitting here**:** classifier works well on training data but does not generalise
    - Some classifiers achieve zero error when applied to training data

Approaches to Evaluation

- Hold back a subset of the training data for testing
  - Holdout method: typically use $\frac{2}{3}$ for training, $\frac{1}{3}$ for testing
    - Balance: enough data to train, but enough to test also
  - Have seen this with the UCI adult data set:
    - Adult.data:32K training examples, adult.test: 16K test examples
  - To evaluate a new classifier, may repeat with different random sets

Approaches to Evaluation

k-Fold Cross Validation

- k-fold cross-validation: robust checking of model:
  - Divide data into k equal pieces (folds).
  - Typical choice: $k = 10$
  - Use $(k - 1)$ folds to form the training set
  - Evaluate model accuracy on the 1 remaining fold
  - Repeat for each of the k choices of training set

Approaches to Evaluation

Stratified Cross Validation

- Additionally, ensure that each fold has same class distribution.
- Likely to hold if folds chosen randomly.

Fold 1
Fold 2
Fold 3
Fold 4

Measures of Accuracy

- For each test example, the result is either right or wrong:
  - Typically, a “near miss” is not counted…
  - Accuracy counts the fraction of examples correctly classified
- Refresher: there are several other measures of quality:
  - True Positive Rate, False Positive Rate, Area Under Curve (AUC)...

Measures of Accuracy

- True Positives (TP): These refer to positive examples that were correctly labeled by the classifier.
- True Negatives (TN): These refer to negative examples that were correctly labeled by the classifier.
- False Positives (FP): These refer to negative examples that were incorrectly labeled by the classifier as positive.
- False Negatives (FN): These refer to positive examples that were incorrectly labeled by the classifier as negative.

Predicted Class

		+	-
True Class	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

Predicted Class

		+	-
True Class	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

- True Positives (TP): These refer to positive examples that were correctly labeled by the classifier.
- True Negatives (TN): These refer to negative examples that were correctly labeled by the classifier.
- False Positives (FP): These ref examples that were incorrectl classifier as positive.
- False Negatives (FN): These refer to positive examples that were incorrectly labeled by the classifier as negative.

We can summarise this information in a *Confusion Matrix*.

Predicted Class

True Class		+	-
	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

Predicted Class

True Class		+	-
	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

True Positive Rate

- True Positive Rate = $\dfrac{TP}{TP + FN}$
- Also known as recall, or sensitivity
- Tells us what fraction of positive values are reported as positive.

		Predicted Class	
		+	-
True Class	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

		Predicted Class	
		+	-
True Class	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

False Positive Rate

- False Positive Rate = $\dfrac{FP}{FP + TN}$
- Can also be considered *false alarms.*
- Tells us what fraction of negative values are reported as positive.

Predicted Class

True Class		+	-
	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

Predicted Class

True Class		+	-
	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

True Negative Rate

- Specificity $= \dfrac{TN}{FP + TN}$
- Also known as *specificity*.
- Tells us what fraction of negative values are correctly reported as negative.

Predicted Class

		+	-
True Class	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

Predicted Class

		+	-
True Class	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

Precision

- $\text{Precision} = \dfrac{TP}{TP + FP}$
- Tells us what fraction of those reported as positive, are actually positive.

Predicted Class

True Class		+	-
	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

Predicted Class

True Class		+	-
	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

Accuracy

- Accuracy = $\dfrac{TP + TN}{TP + TN + FN + FP}$
- Also known as the *recognition rate*.
- Tells us what fraction are reported correctly.

Predicted Class

		+	-
True Class	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

- Error Rate = 1-Accuracy = $\dfrac{FP + FN}{TP + TN + FN + FP}$
- Also known as the *misclassification rate*.

Predicted Class

		+	-
True Class	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

F1 Measure

- F1 Measure $= \dfrac{2TP}{2TP + FP + FN}$
- Harmonic mean of precision and recall.

Predicted Class

		+	-
True Class	+	True Positive (TP)	False Negative (FN)
	-	False Positive (FP)	True Negative (TN)

Predicted Class

		+	-
True Class	+	1	6
	-	9	84

7 +'s; classifier only gets 1,
93 –'s; classifier gets 84

Measures of Accuracy

Example

		Predicted Class		
	Student	Yes	No	Total
True Class	Yes	345	67	412
	No	128	942	1070
	Total	473	1009	1482

Measures of Accuracy

Example

		Predicted Class		
	Student	Yes	No	Total
True Class	Yes	345 (TP)	67 (FN)	412
	No	128 (FP)	942 (TN)	1070
	Total	473	1009	1482

- True Positive Rate = $\dfrac{TP}{TP+FN} = \dfrac{345}{345+67} = \dfrac{345}{412} = 0.837$
- False Positive Rate = $\dfrac{FP}{FP+TN} = \dfrac{128}{128+942} = \dfrac{128}{1070} = 0.120$
- Specificity = $\dfrac{TN}{FP+TN} = \dfrac{942}{128+942} = \dfrac{942}{1070} = 0.880$
- Precision = $\dfrac{TP}{TP+FP} = \dfrac{345}{345+128} = \dfrac{345}{473} = 0.729$

Measures of Accuracy

Example

		Predicted Class		
	Student	**Yes**	**No**	**Total**
True Class	**Yes**	345 (TP)	67 (FN)	412
	No	128 (FP)	942 (TN)	1070
	Total	473	1009	1482

- Accuracy =
$$\frac{TP + TN}{TP + TN + FN + FP} = \frac{345 + 942}{345 + 942 + 67 + 128} = \frac{1287}{1482} = 0.868$$
- Error Rate = 1-Accuracy =
$$\frac{FP + FN}{TP + TN + FN + FP} = \frac{128 + 67}{345 + 942 + 67 + 128} = \frac{195}{1482} = 0.132$$
- F1 Measure = $\frac{2TP}{2TP + FP + FN} = \frac{2 \times 345}{(2 \times 345) + 128 + 67} = \frac{690}{885} = 0.780$

Comparing Classifiers

- There are other ways to compare classifiers:

  1. **Speed**: The computational cost of generating and using the given classifier.

     - Some classifiers take a prohibitively long time to train!

  2. **Robustness**: The ability of the classifier to make correct predictions given noisy data or data with missing values.

Comparing Classifiers

- There are other ways to compare classifiers:

3. **Scalability**: The ability to construct the classifier efficiently given large amounts of data.

4. **Interpretability**: The level of understanding and insight that is provided by the classifier (some are easy to understand and interpret, whilst some are complex).

    - Can a human make sense of the model? Can we identify the most important attribute(s)?

    - This is subjective!

Comparing Models

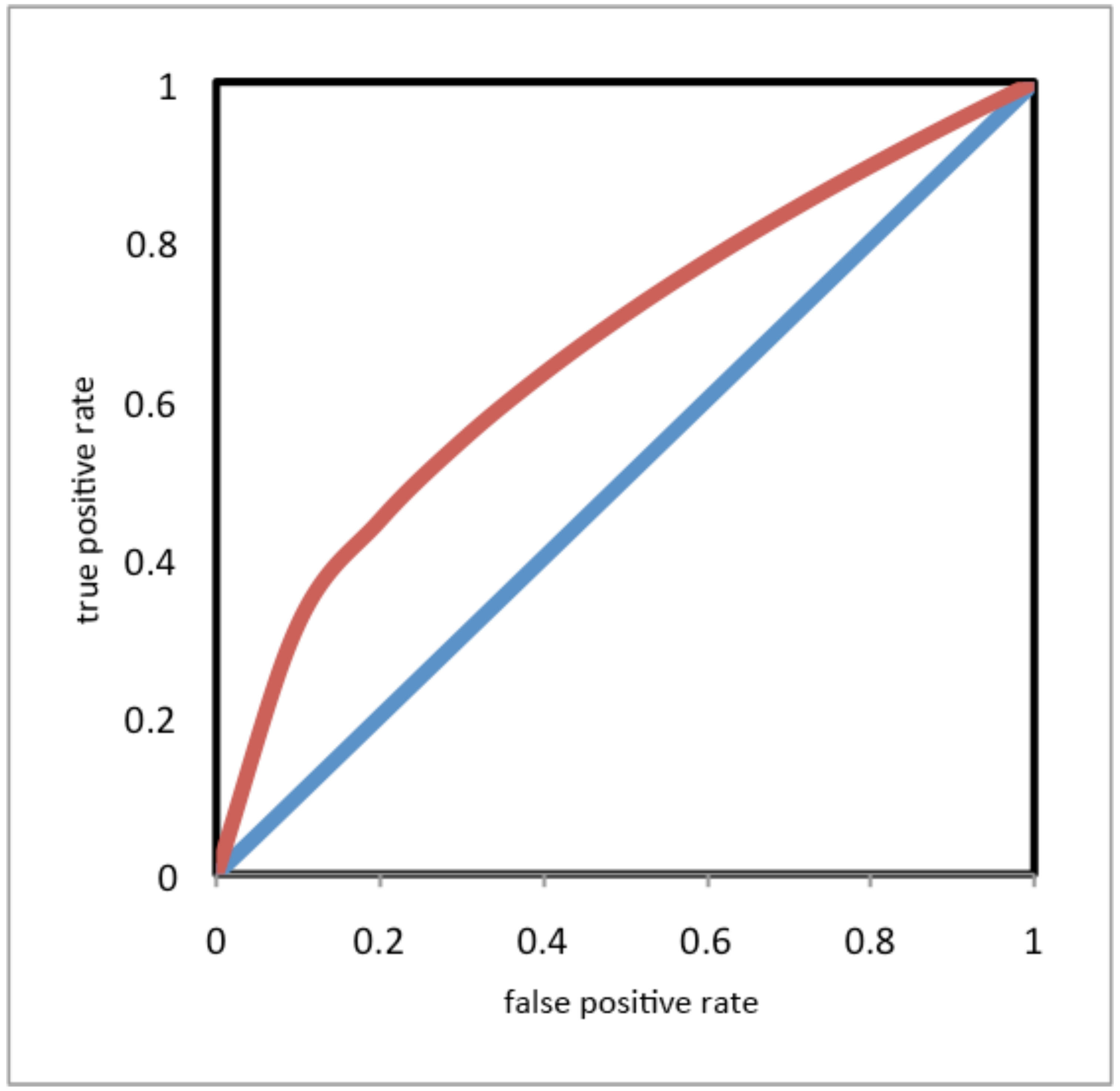
Receiver Operating Characteristic (ROC) Curves

- ROC Curves are a useful tool for comparing two classification models.
- Shows the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR).
- Assume each prediction has a confidence p between 0 and 1
  - Can pick a threshold r, round all confidences $< r$ to 0, $> r$ to 1
  - For each choice of r, we get a true positive rate and false positive rate
  - Different choices of r give a different tradeoff

Comparing Models

Receiver Operating Characteristic (ROC) Curves

- Receiver Operating Characteristic (ROC) curve:
  - Closer to top-left is better
  - Area Under Curve (AUC) measures overall quality
  - Compute AUC by stepping through sorted confidence values
  - Random guessing gives diagonal line
    - AUC is 0.5



Comparing Models

Receiver Operating Characteristic (ROC) Curves

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

True Class.

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

Comparing Models

Receiver Operating Characteristic (ROC) Curves

Probability returned by classifier.

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

Let’s start with tuple 1.

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

Prob = 0.9, let's use that as our threshold, t.

If Prob(tuple #)$\geq$0.9, predict positive, otherwise predict negative.

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	[illegible]	[illegible]	[illegible]	[illegible]
5	P	0.55	4	1	[illegible]	[illegible]	[illegible]	[illegible]
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	[illegible]	[illegible]	[illegible]	[illegible]
9	P	0.50	5	4	[illegible]	[illegible]	[illegible]	[illegible]
10	N	0.40	5	5	0	0	1.0	1.0

So tuple 1 is predicted as positive, and rest are predicted as negative.

This leaves us with 1 TP, 0 FP, 5 TN, 4 FN, a TPR of 0.2, and a FPR of 0.

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

Prob = 0.8, let’s use that as our threshold, t.

If Prob(tuple #)$\geq$0.8, predict positive, otherwise predict negative.

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1				
5	P	0.55	4	1				
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4				
9	P	0.50	5	4				
10	N	0.40	5	5	0	0	1.0	1.0

So tuple 1 and 2 are predicted as positive, and rest are predicted as negative.

This leaves us with 2 TP, 0 FP, 5 TN, 3 FN, a TPR of 0.4, and a FPR of 0.

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

Prob = 0.7, let's use that as our threshold, t.

If Prob(tuple #)$\geq$0.7, predict positive, otherwise predict negative.

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4					
6	N	0.54	4					
7	N	0.53	4					
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5					1.0

So tuple 1, 2 and 3 are predicted as positive, and rest are predicted as negative.

This leaves us with 2 TP, 1 FP, 4 TN, 3 FN, a TPR of 0.4, and a FPR of 0.2.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

Continue for all tuples…

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

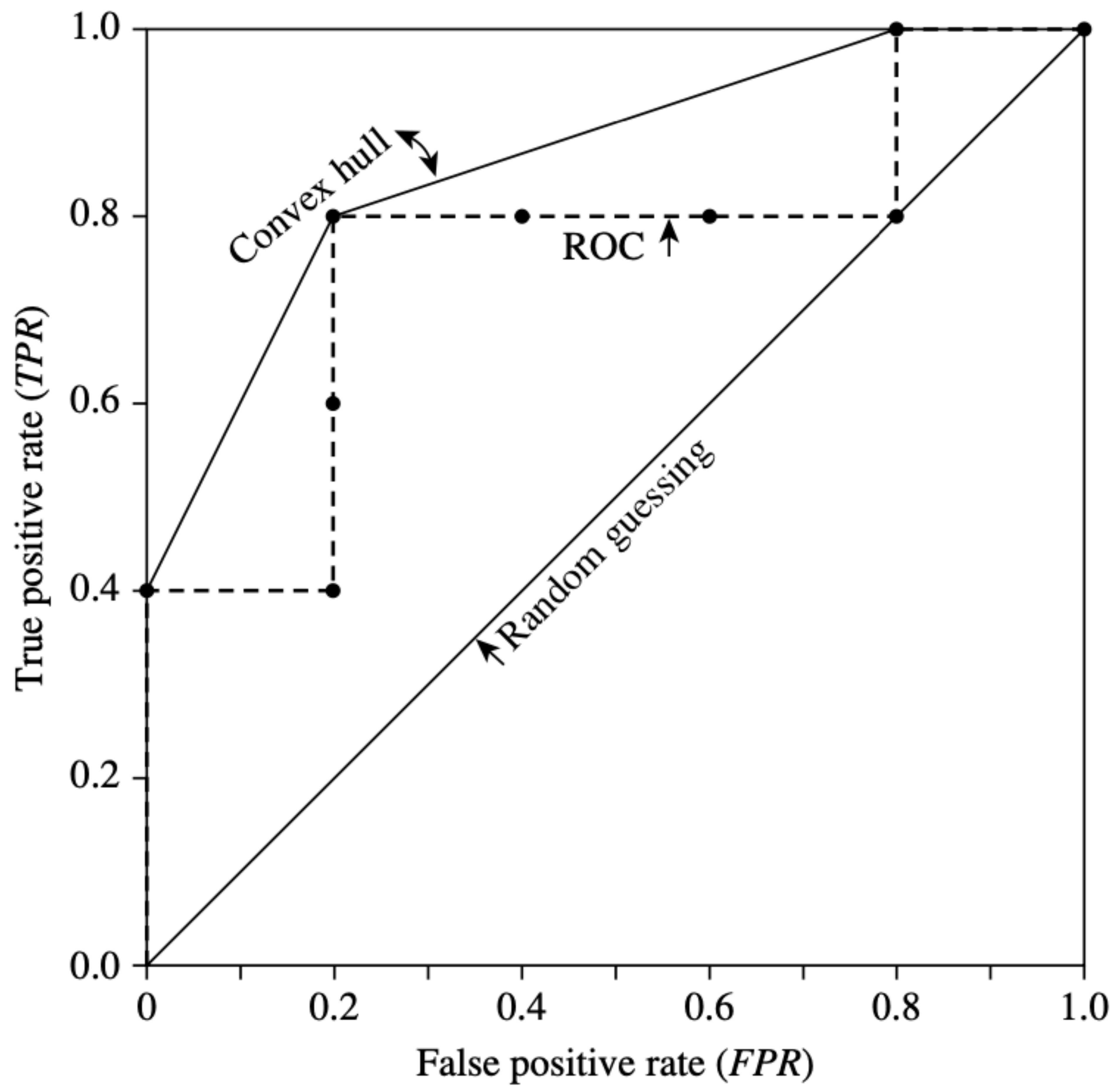
Plot TPR against FPR…

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparing Models

Receiver Operating Characteristic (ROC) Curves

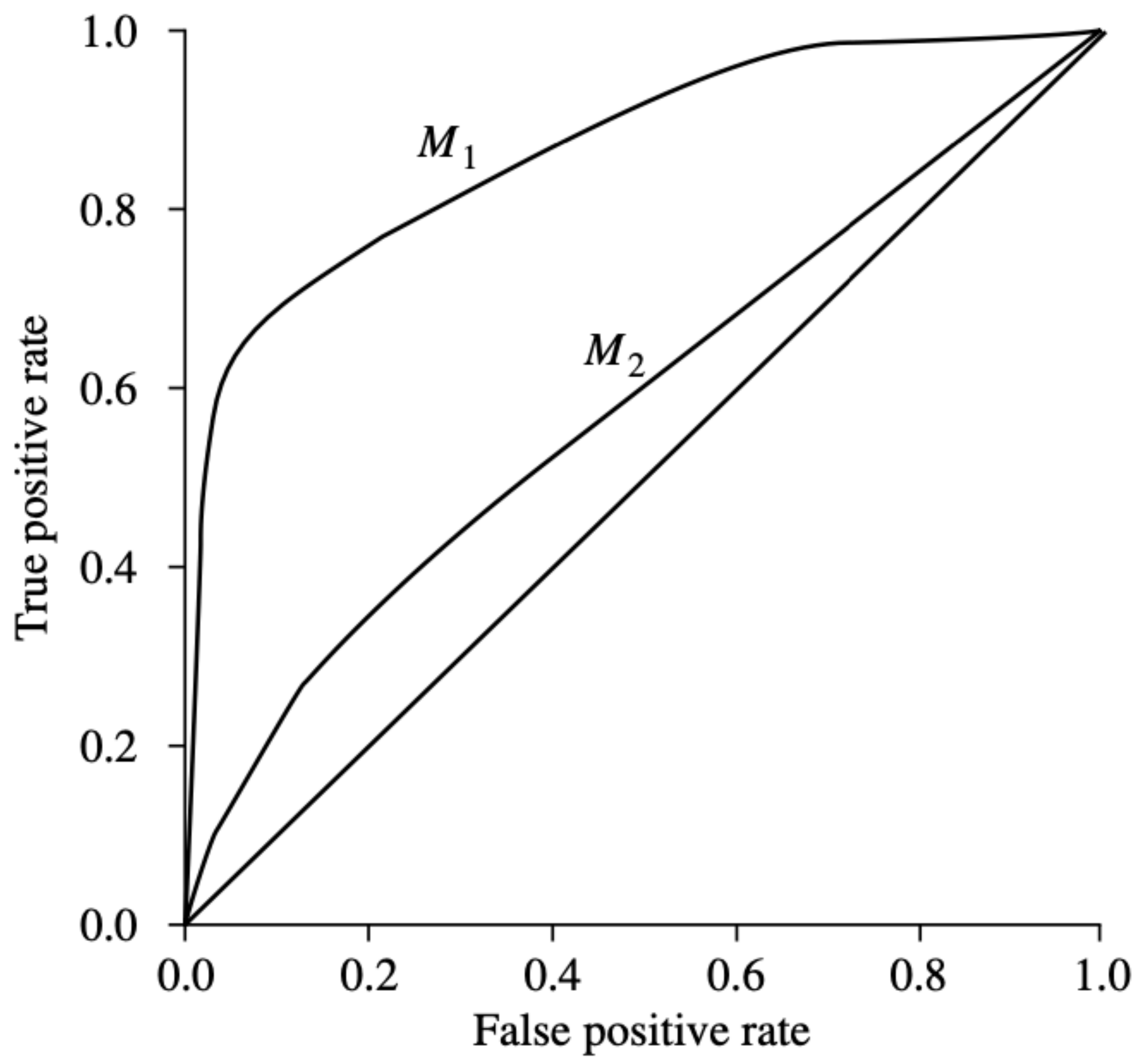


Plot TPR against FPR (jagged line).

There are many methods to obtain a curve, the most common being a *convex hull*.

Comparing Models

Receiver Operating Characteristic (ROC) Curves



- The figure shows the ROC curves of two models (M_1 and M_2).
- The closer the ROC curve to the diagonal line (random guessing) = the worse the model.
- If the model is really good, we are initially more likely to encounter true positives = steeper gradient.
- Can also use Area Under The Curve:
  - The closer the area to 0.5, the less accurate the corresponding model.
  - A model with perfect accuracy will have an area of 1.0.

Acknowledgements

- Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.
- G Cormode [Warwick, CS910]
- F Ciucu [Warwick, CS430/CS910]

Part B: Classification: Introduction 2

Class Imbalance

- Consider a classifier which aims to predict a certain type of cancer, which is absent in 98% of the population.
- The classifier can predict *positive* or *negative* based on some input data.
- We could really easily get a 98% accuracy rate just by predicting *negative*!
- Although, this is not acceptable, as we would miss the 2% of people who do have cancer…

Class Imbalance

Definition

- Class imbalance describes the situation where the classes present within the data are not equally represented.
- Can occur when there are only two classes (see cancer example), but can also occur in multi class examples (where one or more classes are much rarer than others in the dataset).

Class Imbalance

Solutions

1. True Positive Rate (Sensitivity) and True Negative Rate (Specificity) report the proportion of positive examples correctly identified, and the proportion of negative examples correctly identified, respectively.

2. Oversampling

3. Undersampling

4. Threshold Moving

5. Ensemble Techniques

Class Imbalance

Oversampling

- Consider a training data set which contains 500 positive examples and 5000 negative examples…
- **Class Imbalance** - Model can just always predict negative to get a fairly high accuracy rate.
- **Oversampling** - Replicate the examples of the rarer class to form a new training set with 5000 positive examples and 5000 negative examples.
- Could use data augmentation rather than simply replicating existing data.

Class Imbalance

Undersampling

- Consider a training data set which contains 500 positive examples and 5000 negative examples…
- **Class Imbalance** - Model can just always predict negative to get a fairly high accuracy rate.
- **Undersampling** - Randomly eliminate the negative examples until the number of negative examples is equal to the number of positive examples (in our example, 500 positive and 500 negative).

Class Imbalance

Threshold-Moving

- Recall a few slides ago, we mentioned a classifier that returned an output value [0,1], and the instance was considered positive if the output value was $\geq t$.
- Consider $t = 0.98$. Difficult for example to be considered positive as output value would need to be ≥ 0.98.
- Threshold-moving involves moving the value of the threshold, t, to make it easier for rarer classes to be classified.

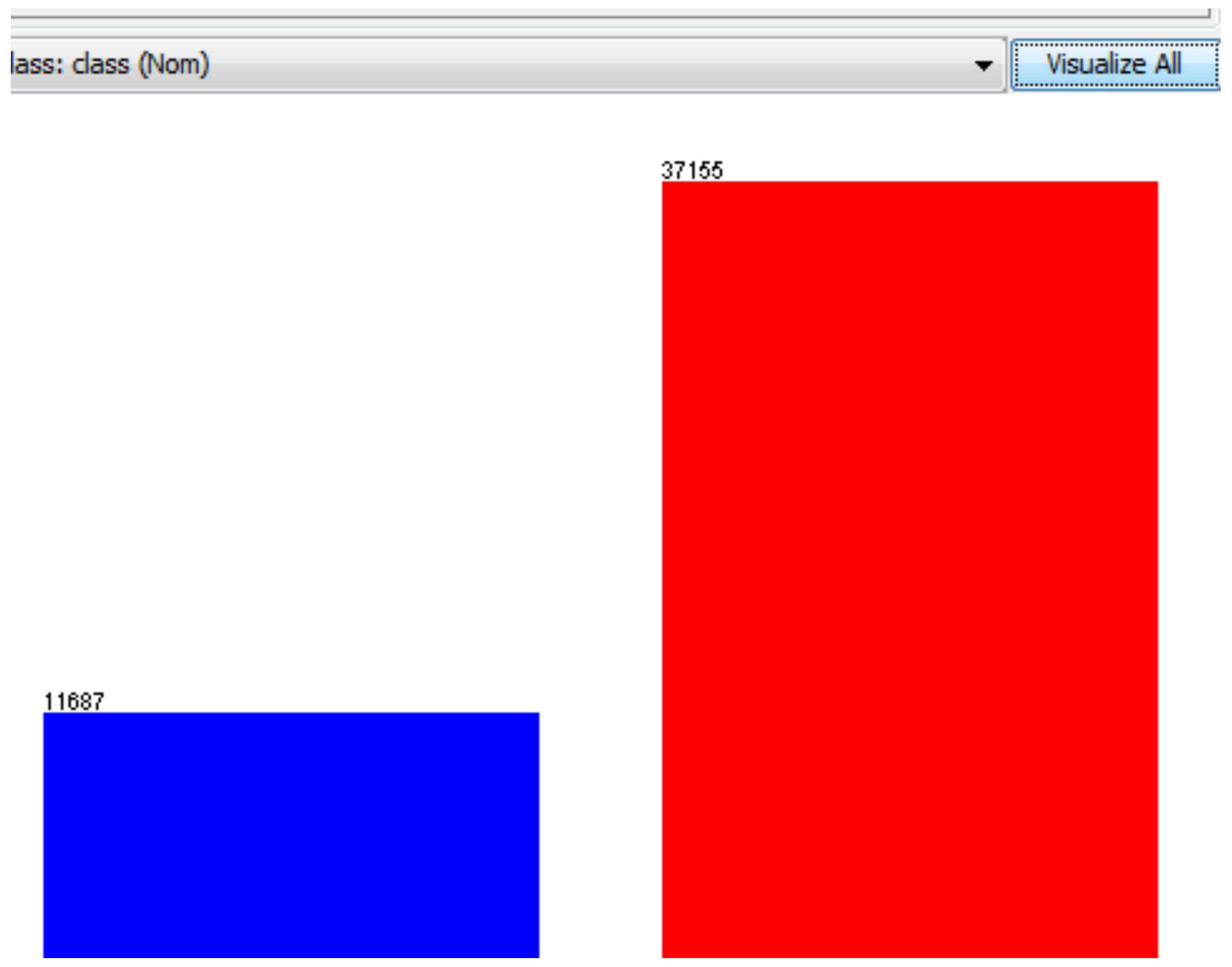
Class Imbalance

Ensemble Techniques

- Combines multiple models ($M_1, M_2, \ldots M_k$), with the aim of creating an improved, composite classification model, $M*$.
- Individual classifiers within the ensemble may include version of the approaches described previously (oversampling, undersampling, threshold moving).

Majority Class Classifier

- A trivial baseline: always predict the majority class
  - More generally, predict the distribution of the class values
- Suppose the majority class occurs a p fraction of the time
  - Then should achieve (approximately) p accuracy
- Not a useful or meaningful classifier
  - But an important benchmark: any good model should clearly outperform the majority class classifier
- Implemented as "ZeroR" model in Weka: **classifiers.rules.ZeroR**
  - Generalises to numeric data: predicts mean



Weka Results for Majority Class Classifier

```
Relation:     adult
Instances:    48842
Attributes:   15
Test mode:split 66.0% train, remainder test
=== Classifier model (full training set) ===
ZeroR predicts class value: <=50K
Time taken to build model: 0.01 seconds
=== Evaluation on test split ===
Correctly Classified Instances       12728               76.647  %
Incorrectly Classified Instances      3878               23.353  %
Kappa statistic                          0
Mean absolute error                      0.3626
Root mean squared error                  0.4232
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances            16606
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area  Class
                 0         0         0          0         0          0.5       >50K
                 1         1         0.766      1         0.868      0.5       <=50K
Weighted Avg.    0.766     0.766     0.587      0.766     0.665      0.5
=== Confusion Matrix ===
     a     b   <-- classified as
     0  3878 |     a = >50K
     0 12728 |     b = <=50K
```

Nearest Neighbour Classifier

- Assume that examples that are “close” should behave alike
    - To classify an example, find most similar training example
- k-nearest neighbour classifier: find the k closest examples
    - And predict the majority vote
    - N-nearest neighbour classifier: equivalent to Majority-class

Nearest Neighbour Classifier

- How to determine which is the closest?
  - Use an appropriate distance, as discussed earlier (L_1/ L_2 distance)
  - Finding the nearest neighbour can be slow: scan through all points
  - Some specialised data structures help, but still not fast

Nearest Neighbour Classifier

In Weka

- Listed as an “instance based classifier” (classifiers.lazy.IBk)
  - Class of classifiers that use training data as model
  - IB1: very simple 1-nearest neighbour classifier
  - IBk: more flexible k-nearest neighbour classifier
    - User provides k (default: 1)
    - Or can use cross-validation to pick best value of k
    - Pick search algorithm to use (default: linearscan)
    - Pick distance to use (default: Euclidean; L_1, L_∞ also available)
  - Fast to build (just look at data), but slow to evaluate/apply

Weka Results for adult.data *k*NN

```
Relation:      adult
Instances:     48842
Attributes:    15
Test mode:split 66.0% train, remainder test
=== Classifier model (full training set) ===
IB1 instance-based classifier using 10 nearest neighbour(s) for classification
Time taken to build model: 0.11 seconds
=== Evaluation on test split ===
Correctly Classified Instances       13851               83.4096 %
Incorrectly Classified Instances      2755               16.5904 %
Kappa statistic                          0.5359
Mean absolute error                      0.2122
Root mean squared error                  0.3392
Relative absolute error                 58.5032 %
Root relative squared error             80.1583 %
Total Number of Instances            16606
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.643     0.108      0.645     0.643     0.644      0.874    >50K
                 0.892     0.357      0.891     0.892     0.892      0.874    <=50K
Weighted Avg.    0.834     0.299      0.834     0.834     0.834      0.874
=== Confusion Matrix ===
     a     b   <-- classified as
  2492  1386 |     a = >50K
  1369 11359 |     b = <=50K
```

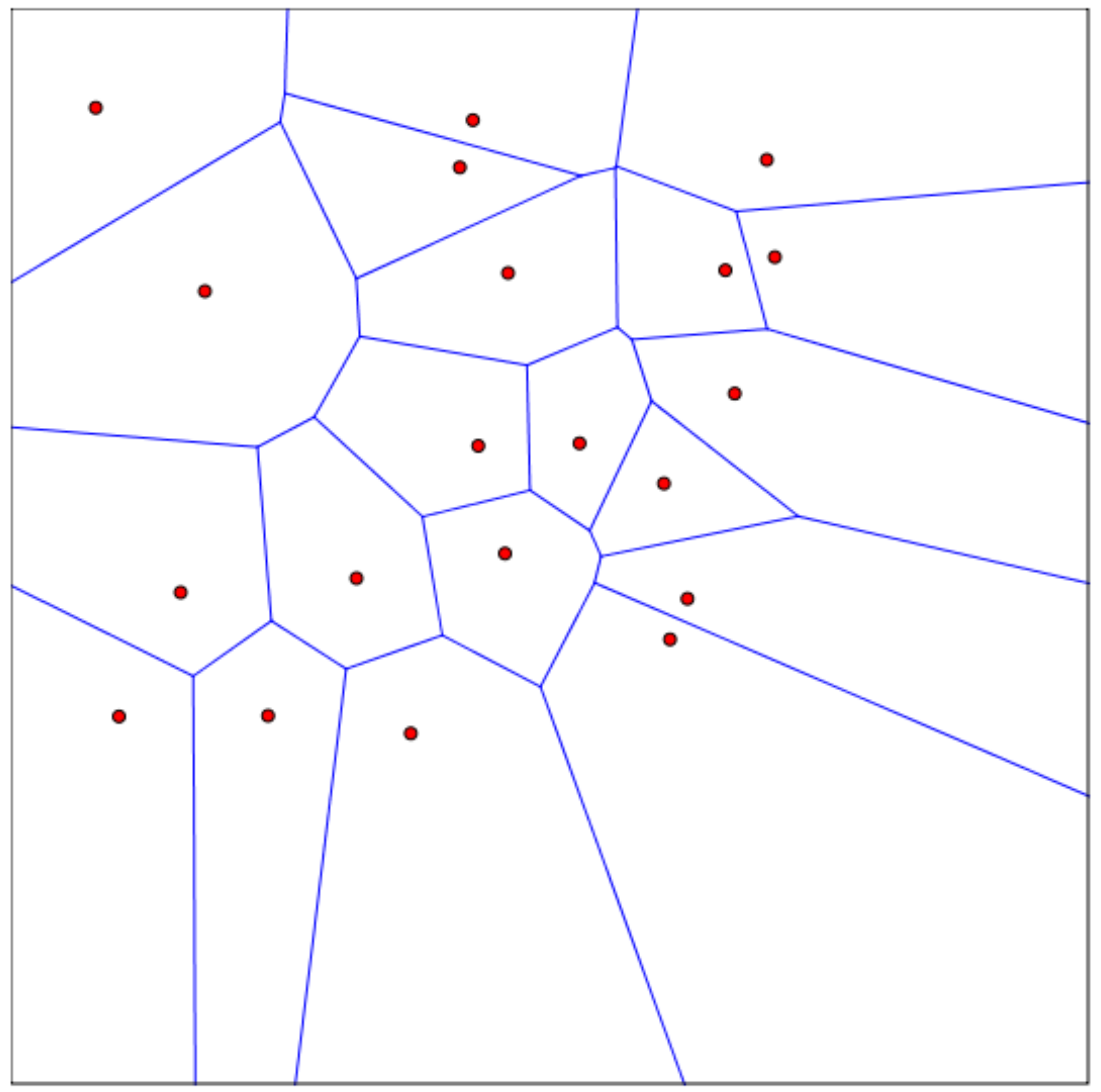
Acknowledgements

- Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.
- G Cormode [Warwick, CS910]
- F Ciucu [Warwick, CS430/CS910]

Part C: Decision Trees

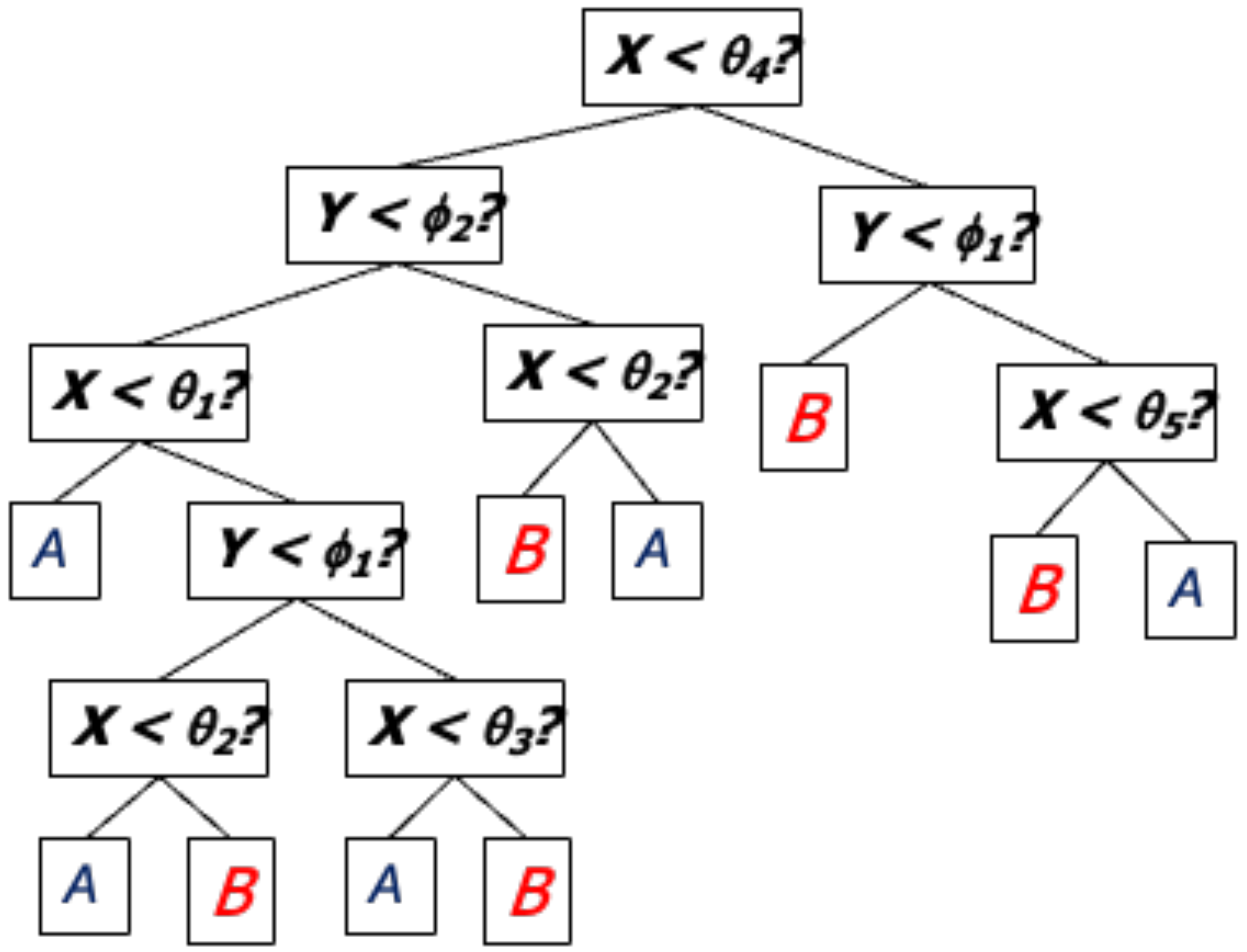
From Points to Regions

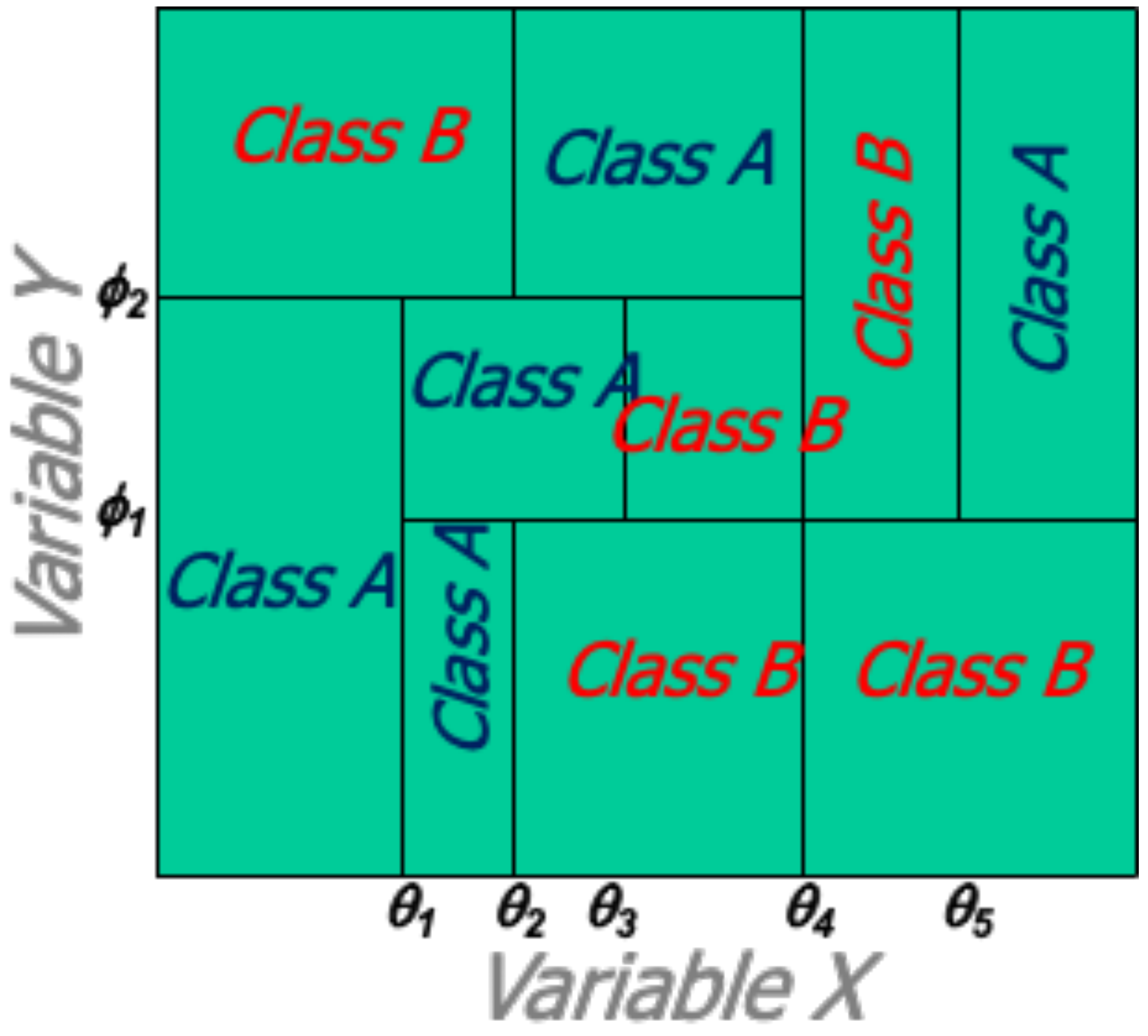
- The principle behind the nearest neighbour classifier is sound
  - Examples with similar class are “close” in the data space
- The partition of space induced by nearest neighbours is complex
  - Difficult to search/index
  - Very detailed description
  - Technically a voronoi diagram
- Other classifiers aim for a simpler description of the space:
  - Divide space into “rectangles”
  - Divide hierarchically for easy indexing



Partition of Space

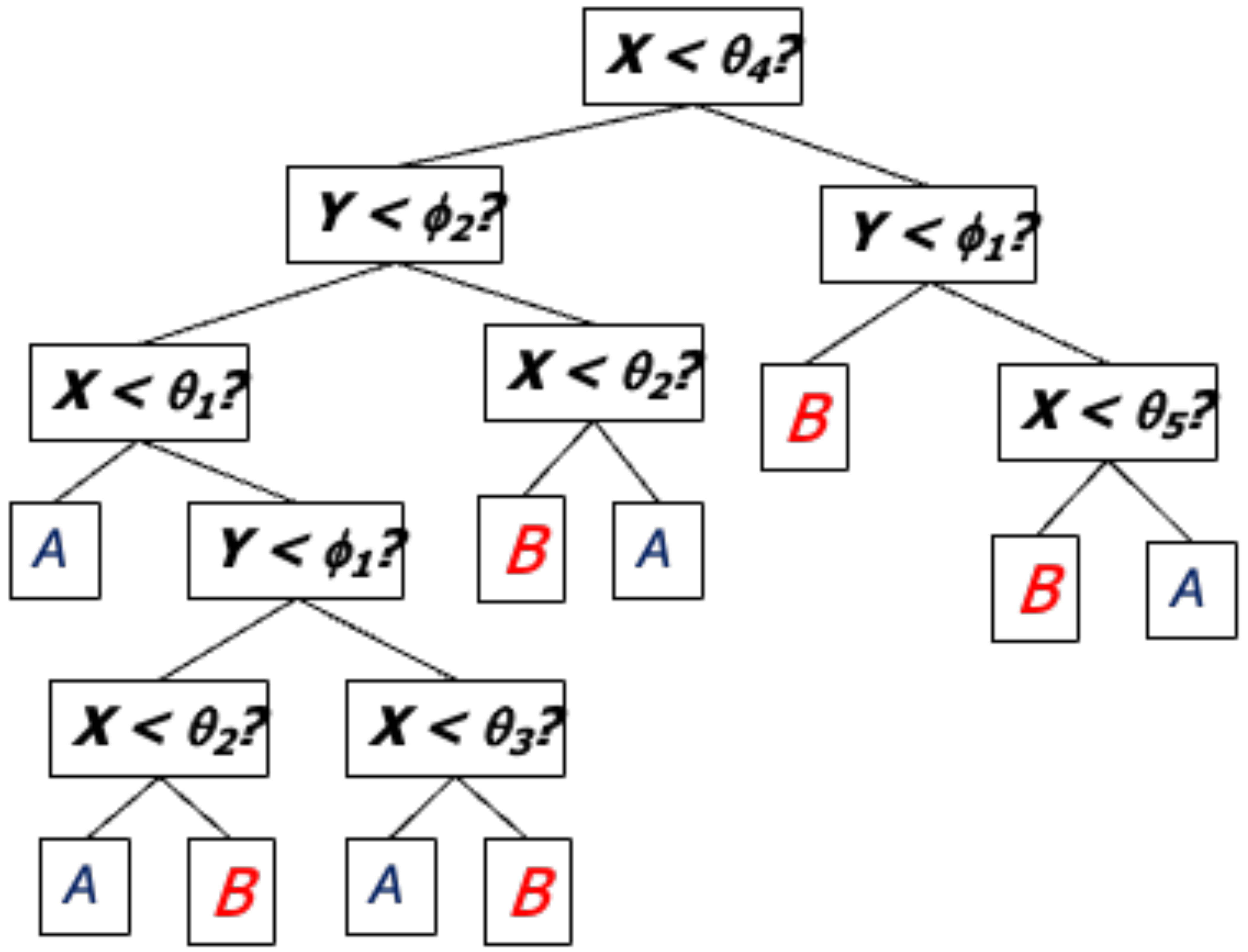
- Hierarchical divisions of space are easy to navigate (for classification)
  - Can be represented as a tree





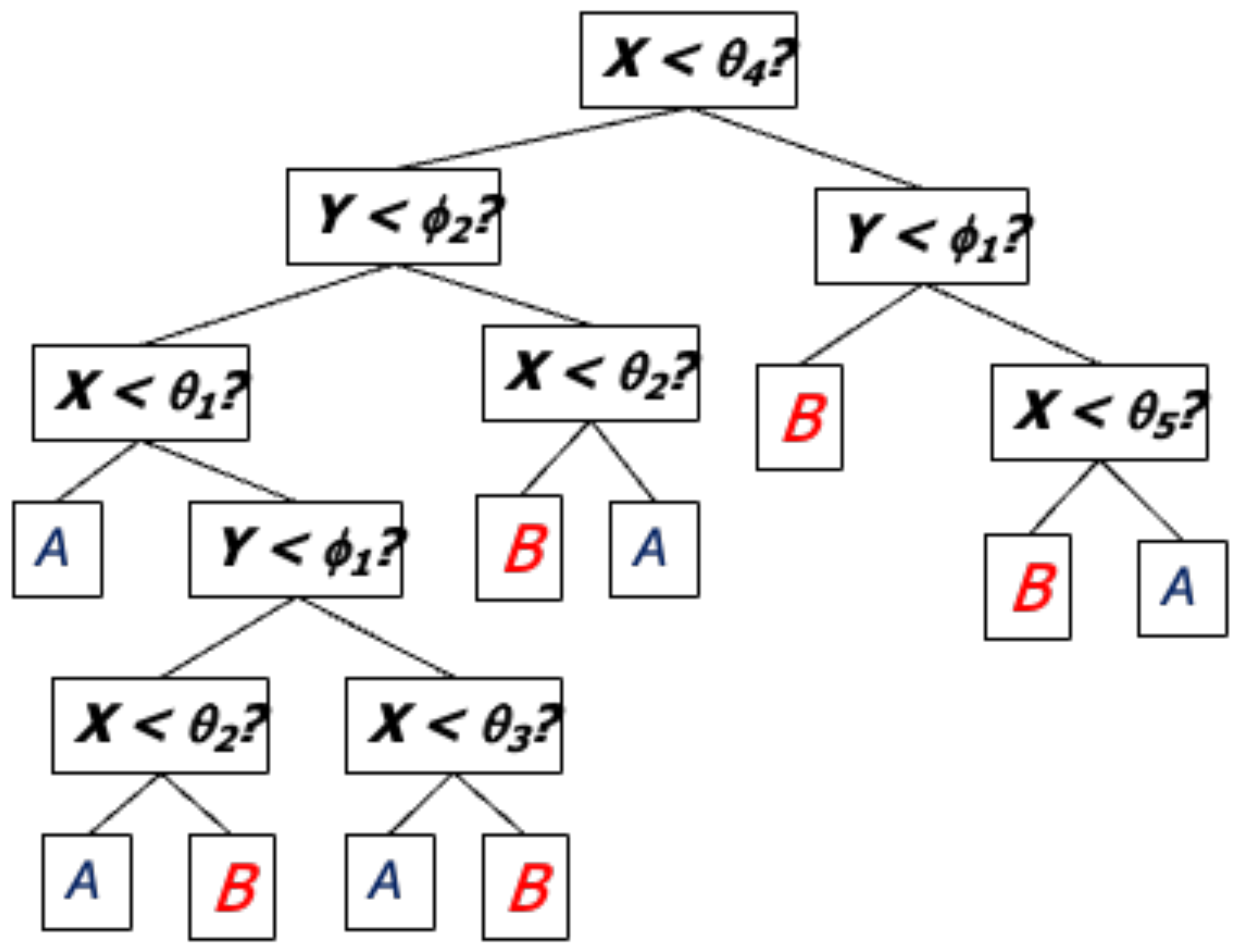
Partition of Space

- A *Decision Tree*:
  - Convention:
    - True to left, False to right
    - Many ways to construct



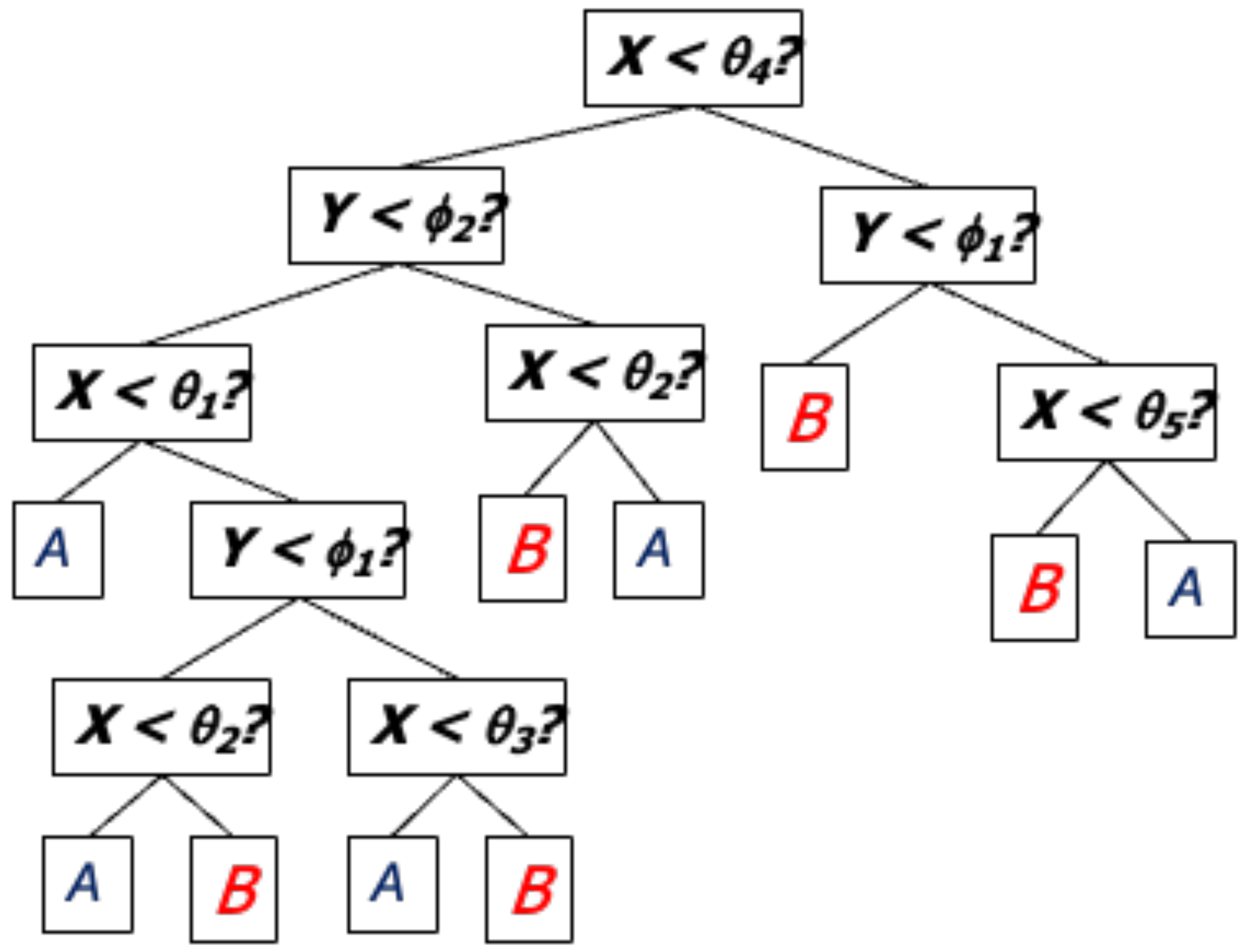
Building a Decision Tree

- Some definitions:
  - **Internal node**: A non-leaf node which denotes a test on an attribute.
  - **Leaf node** (or **terminal node**): Holds a class label.
  - **Root node**: Topmost node.
- For a given example, we test its attribute values against the decision tree.
  - Trace a path to the leaf node to determine the class.



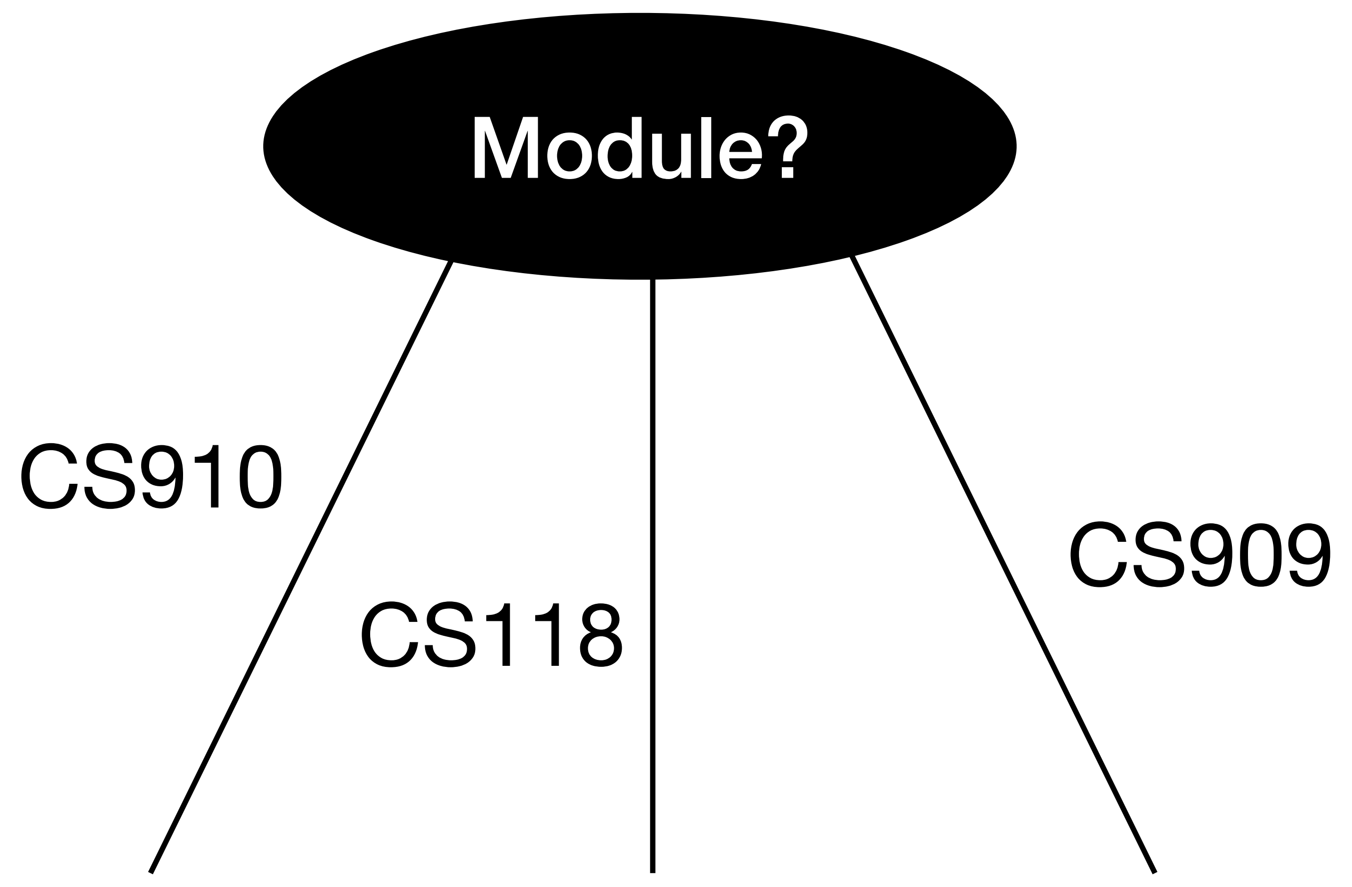
Building a Decision Tree

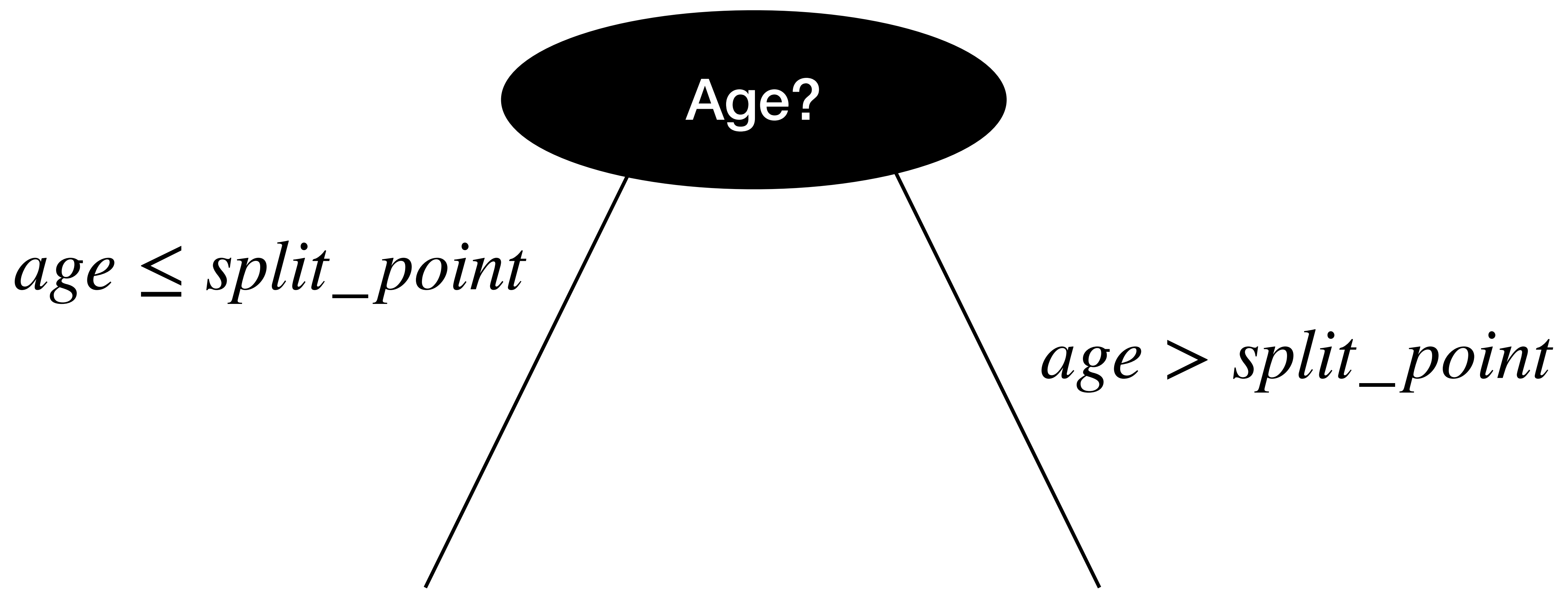
- Many choices in building a decision tree
  - Start with all examples at ‘root’ of tree
  - Choose an attribute to partition on, and a split value
  - Recursively apply the algorithm on both halves
  - Continue splitting until some terminating condition is met:
    - All examples in a partition belong to the same class
    - There is only one example in the partition
  - Label each ‘leaf’ of the tree with the majority class in the leaf
    - Or with the class distribution
- Main difference between algorithms is **how to choose splits**



Building a Decision Tree

Choice of Splits





Module is discrete-valued

Age is continuous-valued

Building a Decision Tree

Choice of Splits

- **Binary attributes** (e.g. Yes/No): only one way to split!
- **Categoric attributes**, e.g. {Single, Married, Widowed, Divorced}
  - Each node could have multiple children (depending on attribute selection measure)
  - Could pick one attribute value versus rest to split on
    - n choices if attribute has n possibilities
  - Could pick any subset of attribute values to split into two
    - ~2^{n-1} choices if attribute has n possibilities
      - e.g. {S}, {S, M}, {S, W}, {S, D}, {S, M, W}, {S, M, D}, {S, W, D}

Building a Decision Tree

Choice of Splits

- **Ordered attributes**, e.g. Grade = (A, B, C, D, E)
  - Each node could have multiple children (depending on attribute selection measure)
  - Or, pick some point to make a split into two.

How to Choose Splits

Attribute Selection Measures

- An attribute selection measure is a heuristic for selecting the best splitting criterion, i.e. that which best separates a given data partition, D, of class-labeled training tuples (examples) into individual classes.
- Ideal scenario: We divide D into smaller partitions according to the outcomes of the splitting criterion, and each partition (at leaf level) would be pure.
  - **Pure**: All the tuples that fall into the given partition would fall into the same class.
- Attribute selection measures are sometimes referred to as *splitting rules*.

How to Choose Splits

Attribute Selection Measures

- Attribute selection measures provide a ranking for each attribute describing the given training tuples.
- The attribute having the “best” score (highest or lowest depending on the method) is chosen as the splitting attribute for the given tuples.
- Three popular attribute selection measures:

  1. **Information Gain**
  2. **Gain Ratio**
  3. **Gini Index**

How to Choose Splits

Notation

- D = Data Partition (training set of class-labelled tuples).
- Let the class label have m distinct classes, C_i (for $i = 1,\ldots,m$).
- Let $C_{i,D}$ be the set of tuples of class C_i in D.
- $|D|$ = Number of tuples in D.
- $|C_{i,D}|$ = Number of tuples in $C_{i,D}$.

Information Gain

- The attribute with the highest information gain is chosen as the splitting attribute for node N, for a data partition D.
  - Goal is to minimise the information needed to classify the tuples in the resulting partitions.
- The expected information needed to classify a tuple in D is given by:

$$Info(D) = -\sum_{i=1}^{m} p_i log_2(p_i)$$

where p_i is the nonzero probability that an arbitrary tuple in D belongs to class C_i, and is estimated by $\frac{|C_{i,D}|}{|D|}$.

Information Gain

- A log function to the base 2 is used, because the information is encoded in bits.
- $Info(D)$ is the average amount of information needed to identify the class label of a tuple in D, and is also known as the *entropy* of D.
- Say we were to partition the tuples in D on some attribute A having v distinct values, $\{a_1, a_2, \ldots, a_v\}$.
  - If A is discrete-valued, attribute A can be used to split D into v partitions, $\{D_1, D_2, \ldots, D_v\}$, where D_j contains the tuples in D that have outcome a_j of A. These partitions would correspond to the branches grown from node N.

Information Gain

- Ideally, these partitions would be pure (i.e. all the examples that fall into the given partition would fall into the same class), but this is **unlikely**.
- How much more information would we need after the previous partitioning to arrive at an exact classification? This is measured by:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- $Info_A(D)$ is the expected information required to classify a tuple from D based on the partitioning from A.
    - The smaller the information still required, the greater the purity of the partitions.

Information Gain

- Information Gain is the difference between the two:

$$Gain(A) = Info(D) - Info_A(D)$$

- Higher value implies that $Info_A(D)$ is smaller, hence we choose attribute with the highest $Gain(A)$.

Information Gain

- D = Data Partition (training set of class-labelled tuples).
- $m = 2$, (plays_sports has 2 classes, C_1 and C_2).
- $|D|$ = Number of tuples in D = 6.
- Remember, $C_{i,D}$ is the set of tuples of class C_i in D, $|D|$ is the number of tuples in D, and $|C_{i,D}|$ is the number of tuples in $C_{i,D}$.

ID	**income**	**is_student**	**Class:plays_sports**
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Information Gain

- Expected information needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i log_2(p_i)$$

$$= -\frac{4}{6}log_2\left(\frac{4}{6}\right) - \frac{2}{6}log_2\left(\frac{2}{6}\right)$$

$$= 0.9183 \text{ bits}$$

Where p_i us the probability that an arbitrary tuple in D belongs to C_i (estimated by $\frac{|C_{i,D}|}{|D|}$).

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Information Gain

- Expected information requirement for $income$:

$$Info_{income}(D) = = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{2}{6} \times \left(- \frac{1}{2} log_2\left(\frac{1}{2}\right) - \frac{1}{2} log_2\left(\frac{1}{2}\right)\right) +$$

$$\frac{1}{6} \times \left(- \frac{1}{1} log_2\left(\frac{1}{1}\right)\right) + \frac{3}{6} \times \left(- \frac{2}{3} log_2\left(\frac{2}{3}\right) - \frac{1}{3} log_2\left(\frac{1}{3}\right)\right)$$

$= 0.7925$ bits

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Information Gain

- The gain in information from partitioning on income would be:

$$\begin{aligned} Gain(income) &= Info(D) - Info_{income}(D) \\ &= 0.9183 - 0.7925 \\ &= 0.1258 \text{ bits} \end{aligned}$$

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Information Gain

- Expected information requirement for $is\_student$:

$$Info_{is\_student}(D) = = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$= \frac{4}{6} \times \left(-\frac{2}{4} log_2\left(\frac{2}{4}\right) - \frac{2}{4} log_2\left(\frac{2}{4}\right)\right) +$$

$$\frac{2}{6} \times \left(-\frac{2}{2} log_2\left(\frac{2}{2}\right)\right)$$

$= 0.66666$ bits

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Information Gain

- The gain in information from partitioning on is_student would be:

$$Gain(is\_student) = Info(D) - Info_{is\_student}(D)$$
$$= 0.9183 - 0.6666$$
$$= 0.2517 \text{ bits}$$

- $is\_student$ has the highest Information Gain, so it is chosen as the splitting attribute.

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Decision Tree Algorithms and Split Rules

- Different decision tree algorithms use different split rules:
  - ID3 uses **Information Gain**
  - C4.5 uses **Gain Ratio**
  - CART (classification and regression tree) uses the **Gini Index**

Gain Ratio

- Problem with Information Gain: It is biased towards tests with many outcomes.
    - Consider an attribute user_ID. Splitting on user_ID would result in a large number of partitions (equal to number of values), each containing one tuple.
        - Each partition would be pure, so Information Gain would prefer this outcome!
- However, this partition is useless…

Gain Ratio

- Gain Ratio is an extension of Information Gain, which overcomes this problem.
- It is used by C4.5, a successor of ID3.
- Gain Ratio is defined as:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

Gain Ratio

- Gain Ratio is an extension of Information Gain, which overcomes this problem.
- It is used by C4.5, a successor of ID3.
- Gain Ratio is defined as:

Information Gain

$$GainRatio(A) = \frac{\boxed{Gain(A)}}{SplitInfo_A(D)}$$

Gain Ratio

- Gain Ratio is an extension of Information Gain, which overcomes this problem.
- It is used by C4.5, a successor of ID3.
- Gain Ratio is defined as:

$$GainRatio(A) = \frac{Gain(A)}{\boxed{SplitInfo_A(D)}}$$

Normalisation

Gain Ratio

- Gain Ratio is an extension of Information Gain, which overcomes this problem.
- It is used by C4.5, a successor of ID3.
- Gain Ratio is defined as:

$$GainRatio(A) = \frac{Gain(A)}{\boxed{SplitInfo_A(D)}}$$

Normalisation

How do we define $SplitInfo_A(D)$?

Gain Ratio

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right)$$

- Note that for each outcome $(1,\ldots,v)$, it considers the number of tuples having that outcome, with respect to the total number of tuples in D.
- The attribute with the **maximum Gain Ratio** is selected as the splitting attribute.

Gain Ratio

Example

- As shown previously:

$$Gain(is\_student) = 0.2517 \text{ bits}$$

$$SplitInfo_{is\_student}(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right)$$

$$= -\frac{4}{6} log_2\left(\frac{4}{6}\right) - \frac{2}{6} log_2\left(\frac{2}{6}\right)$$

$$= 0.9183$$

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Gain Ratio

Example

$$
\begin{aligned}
GainRatio(is\_student) &= \frac{Gain(is\_student)}{SplitInfo_{is\_student}(D)} \\
&= \frac{0.2517}{0.9183} \\
&= 0.2741
\end{aligned}
$$

- The attribute with the **maximum** Gain Ratio is taken as the splitting attribute.

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Gini Index

- The Gini Index is used by CART.
- It measures the impurity of D, as:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

- p_i is the probability that a tuple in D belongs to class C_i, and is estimated by $\frac{|C_{i,D}|}{|D|}$. It is then computed over m classes.

Gini Index

- For each attribute, the Gini Index considers a binary split.
- Consider the case where A is a discrete-valued attribute with v distinct values, $\{a_1, a_2, \ldots, a_v\}$, occurring in D.
- To determine the best binary split, we examine all possible subsets of known values of A.
  - If A has v possible values, then there are 2^v possible subsets.

Gini Index

- Let $module$ have three possible values, $\{CS118, CS430, CS910\}$.
  - The possible subsets are:

    $\{CS118, CS430, CS910\}$, $\{CS118, CS430\}$, $\{CS118, CS910\}$, $\{CS430, CS910\}$, $\{CS118\}$, $\{CS430\}$, $\{CS910\}$ and , $\{\}$.
  - $\{CS118, CS430, CS910\}$ and $\{\}$ (the empty set) are excluded, as they do not represent a split.
    - Hence, there are $\dfrac{(2^v - 2)}{2}$ possible ways to form two partitions of the data D from the binary split on A.

Gini Index

- If a binary split on A partitions D into D_1 and D_2, the Gini Index of D given that partitioning is:

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2)$$

- For each attribute, each of the possible binary splits is considered.

- For a discrete-valued, the subset that gives the **minimum** Gini index for that attribute is selected as the splitting subset.

Gini Index

- For continuous-valued attributes, the midpoint between each pair of (sorted) values is taken as a possible split-point.
- The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute.
- Remember, for a given split point of A, D_1 is the set of tuples in D satisfying $A \leq split\_point$, and D_2 is the set of tuples in D satisfying $A > split\_point$.

Gini Index

- The reduction in impurity incurred by a binary split on a discrete or continuous-valued attribute A is:

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximises the reduction in impurity is selected as the splitting attribute.
    - This attribute and its splitting subset (or splitting point for a continuous-valued splitting attribute) together form the *splitting criterion*.

Gini Index

Example

- We must first compute $Gini(D)$ as follows:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

$$= 1 - \left(\frac{4}{6}\right)^2 - \left(\frac{2}{6}\right)^2$$

$$= 0.444...$$

- Now, we must compute the Gini Index for each attribute…

ID	**income**	**is_student**	**Class:plays_sports**
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Gini Index

Example

- $income$ binary subsets are $\{\{low, medium\}, \{high\}\}$, $\{\{low, high\}, \{medium\}\}$, $\{\{medium, high\}, \{low\}\}$.

$$Gini_{income \in \{low, medium\}}(D)$$
$$= \frac{3}{6}Gini(D_1) + \frac{3}{6}Gini(D_2)$$
$$= \frac{3}{6}\left(1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2\right) +$$
$$\frac{3}{6}\left(1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2\right)$$
$$= 0.444... = Gini_{income \in \{high\}}(D)$$

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Gini Index

Example

- $income$ binary subsets are $\{\{low, medium\}, \{high\}\}$, $\{\{low, high\}, \{medium\}\}$, $\{\{medium, high\}, \{low\}\}$.

$$
\begin{aligned}
&Gini_{income \in \{low,high\}}(D) \\
&= \frac{5}{6} Gini(D_1) + \frac{1}{6} Gini(D_2) \\
&= \frac{5}{6}\left(1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2\right) + \\
&\quad \frac{1}{6}\left(1 - \left(\frac{1}{1}\right)^2\right) \\
&= 0.4 = Gini_{income \in \{medium\}}(D)
\end{aligned}
$$

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Gini Index

Example

- $income$ binary subsets are $\{\{low, medium\}, \{high\}\}$, $\{\{low, high\}, \{medium\}\}$, $\{\{medium, high\}, \{low\}\}$.

$$Gini_{income \in \{medium, high\}}(D)$$
$$= \frac{4}{6} Gini(D_1) + \frac{2}{6} Gini(D_2)$$
$$= \frac{4}{6}\left(1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2\right) +$$
$$\frac{2}{6}\left(1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2\right)$$
$$= 0.41666... = Gini_{income \in \{low\}}(D)$$

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Gini Index

Example

- $is\_student$ binary subsets are $\{\{yes\}, \{no\}\}$.

$$
\begin{aligned}
&Gini_{is\_student \in \{yes\}}(D) \\
&= \frac{4}{6} Gini(D_1) + \frac{2}{6} Gini(D_2) \\
&= \frac{4}{6}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) + \\
&\quad \frac{2}{6}\left(1 - \left(\frac{2}{2}\right)^2\right) \\
&= 0.333... = Gini_{is\_student \in \{no\}}(D)
\end{aligned}
$$

ID	income	is_student	Class:plays_sports
1	medium	yes	yes
2	high	no	yes
3	low	yes	no
4	low	yes	yes
5	high	yes	no
6	high	no	yes

Gini Index

Example

- Summary:

$$Gini_{income \in \{low, medium\}}(D) = Gini_{income \in \{high\}}(D) = 0.444...$$
$$Gini_{income \in \{low, high\}}(D) = Gini_{income \in \{medium\}}(D) = 0.4$$
$$Gini_{income \in \{medium, high\}}(D) = Gini_{income \in \{low\}}(D) = 0.41666...$$
$$Gini_{is\_student \in \{yes\}}(D) = Gini_{is\_student \in \{no\}}(D) = 0.333...$$

- Therefore, the best binary split on $income$ is $\{low, high\}$ (or $\{medium\}$), as it minimises the Gini index. The best binary split on $is\_student$ is $\{yes\}$ (or $\{no\}$), because the attribute itself is binary.

Gini Index

Example

Minimum Gini Index overall.

- Summary:

$$Gini_{income \in \{low, medium\}}(D) = Gini_{income \in \{high\}}(D) = 0.444...$$

$$Gini_{income \in \{low, high\}}(D) = Gini_{income \in \{medium\}}(D) = 0.4$$

$$Gini_{income \in \{medium, high\}}(D) = Gini_{income \in \{low\}}(D) = 0.41666...$$

$$Gini_{is\_student \in \{yes\}}(D) = Gini_{is\_student \in \{no\}}(D) = 0.333...$$

- Therefore, the best binary split on $income$ is $\{low, high\}$ (or $\{medium\}$), as it minimises the Gini index. The best binary split on $is\_student$ is $\{yes\}$ (or $\{no\}$), because the attribute itself is binary.

Gini Index

Example

- Recall:
  - The reduction in impurity incurred by a binary split on a discrete or continuous-valued attribute A is:

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- Therefore:

$$\begin{aligned}\Delta Gini(A) &= Gini(D) - Gini_A(D) \\ &= 0.444... - 0.333... \\ &= 0.111...\end{aligned}$$

- $is\_student$ and its splitting subset form the splitting criterion.

Split Rules

- The three measures, in general, return good results, but:
  - **Information Gain**: biased towards multivalued attributes
  - **Gain Ratio**: tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini Index**: bias to multivalued attributes, struggles when number of classes is large
- Many other split rules have been suggested, including:
  - **CHAID**: uses the statistical χ^2 test to choose split
  - **C-SEP**
  - **G-statistic**
  - **Minimum Description Length**: seeks the simplest overall tree
    - Requires fewest bits to describe

Pruning Trees

- Building a tree down to leaves that are pure is likely to overfit
  - **Pure**: only contain members of a single class
- Better results can be obtained by pruning the tree
  - Removing some branches from the tree to reduce size/ complexity

- **Prepruning**: add additional stopping conditions when building
  - E.g. limit the depth of the tree (length of path)
  - E.g. stop when the number of examples in a subtree is small
- **Postpruning**: build full tree, then decide which branches to prune
  - Easier to measure difference before/after pruning

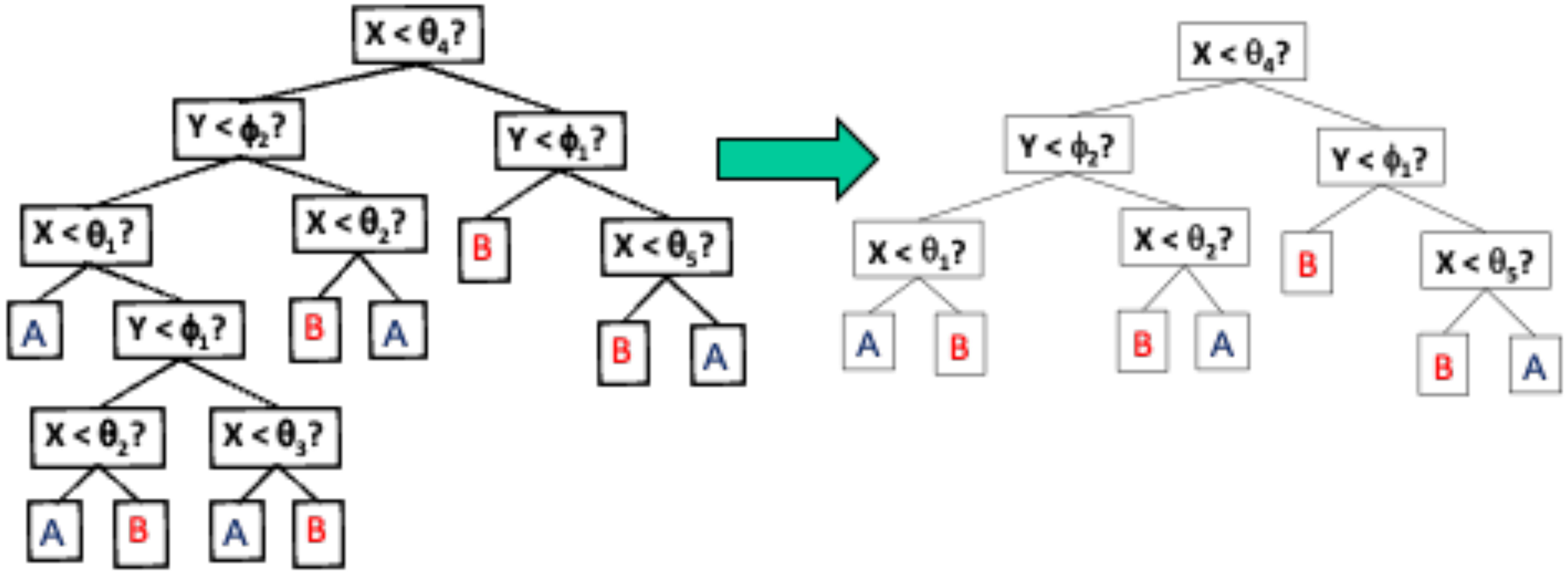
Pruning Trees

More Information

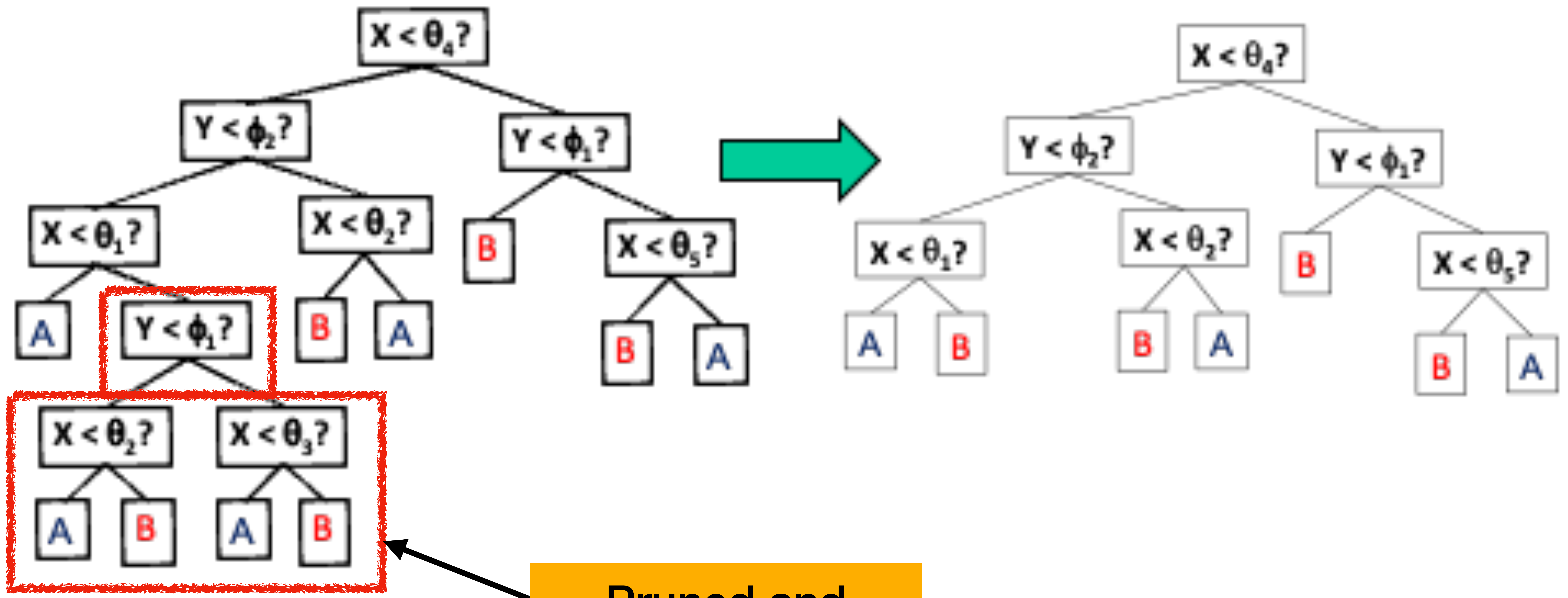
- **Prepruning**: Following application of measure such as Gini Index, Information Gain, etc., if a split falls below a certain threshold, we may instead halt further partitioning.

- **Postpruning**: A subtree at a given node is pruned and replaced with a leaf node. The leaf node is labeled with the most frequent class.

Pruning Trees



Pruning Trees



Pruned and replaced with B.

Pruning Trees

- Prune based on tradeoff between complexity and error
    - Complexity: measure of the complexity of the subtree
    - Error: measured on pruning set (separate from training and test set)
- Reduced error pruning is simple and fast
    - Start at leaves, replace node with majority class
    - Accept if this does not affect accuracy on pruning set

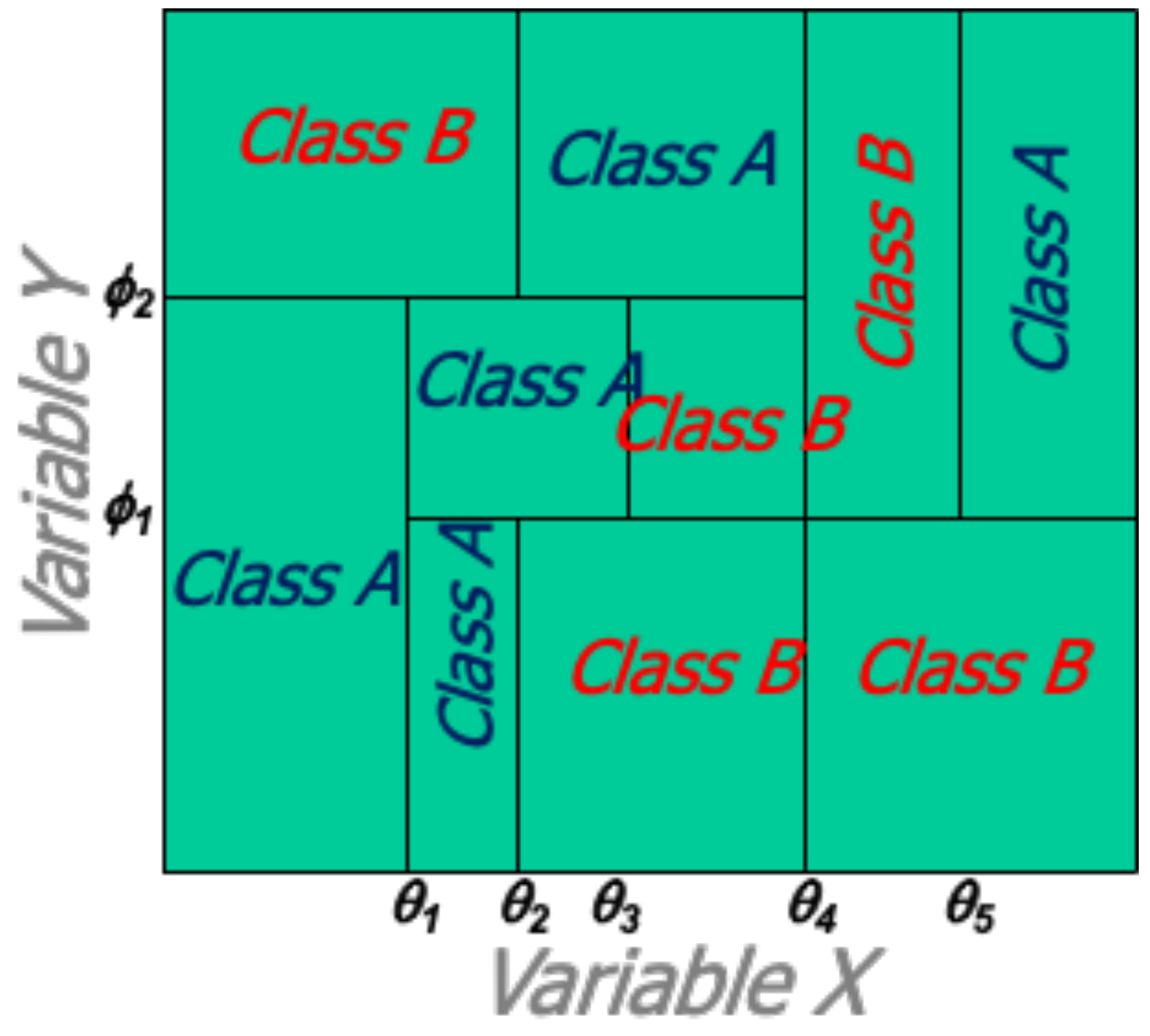
Pruning Trees

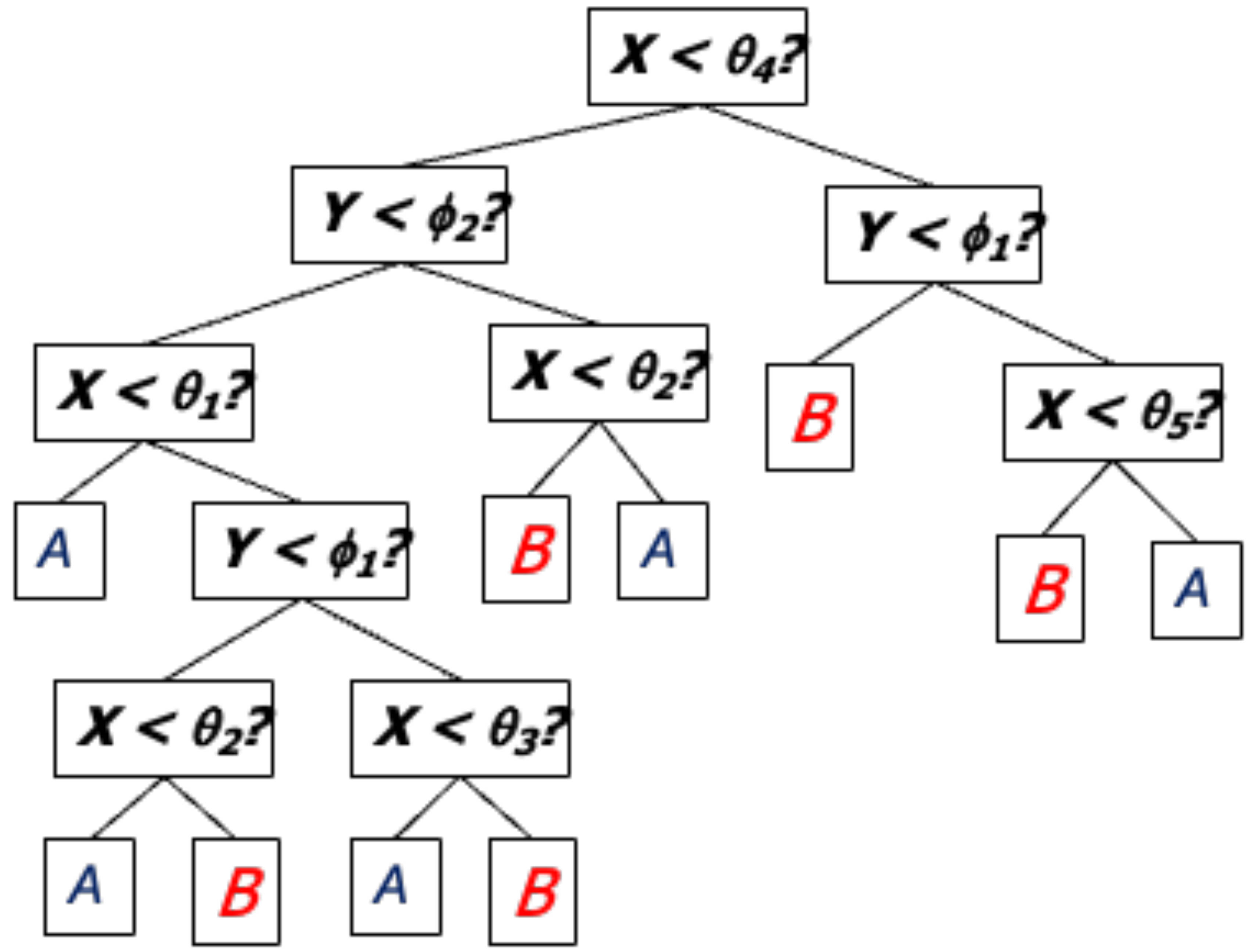
- Cost complexity pruning (CART[1]) uses a more complex function
  - Given current tree T (with $|T|$ leaves) and T' with a node removed
  - Measure E = Error of T on pruning set, E' = error of T' on pruning set
  - Select T' that minimises $\dfrac{E' - E}{|T| - |T'|}$
  - Can evaluate all generated trees on test set, pick the best

[1]Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J., 2017. *Classification and regression trees*. Routledge.

Classifying with a Decision Tree

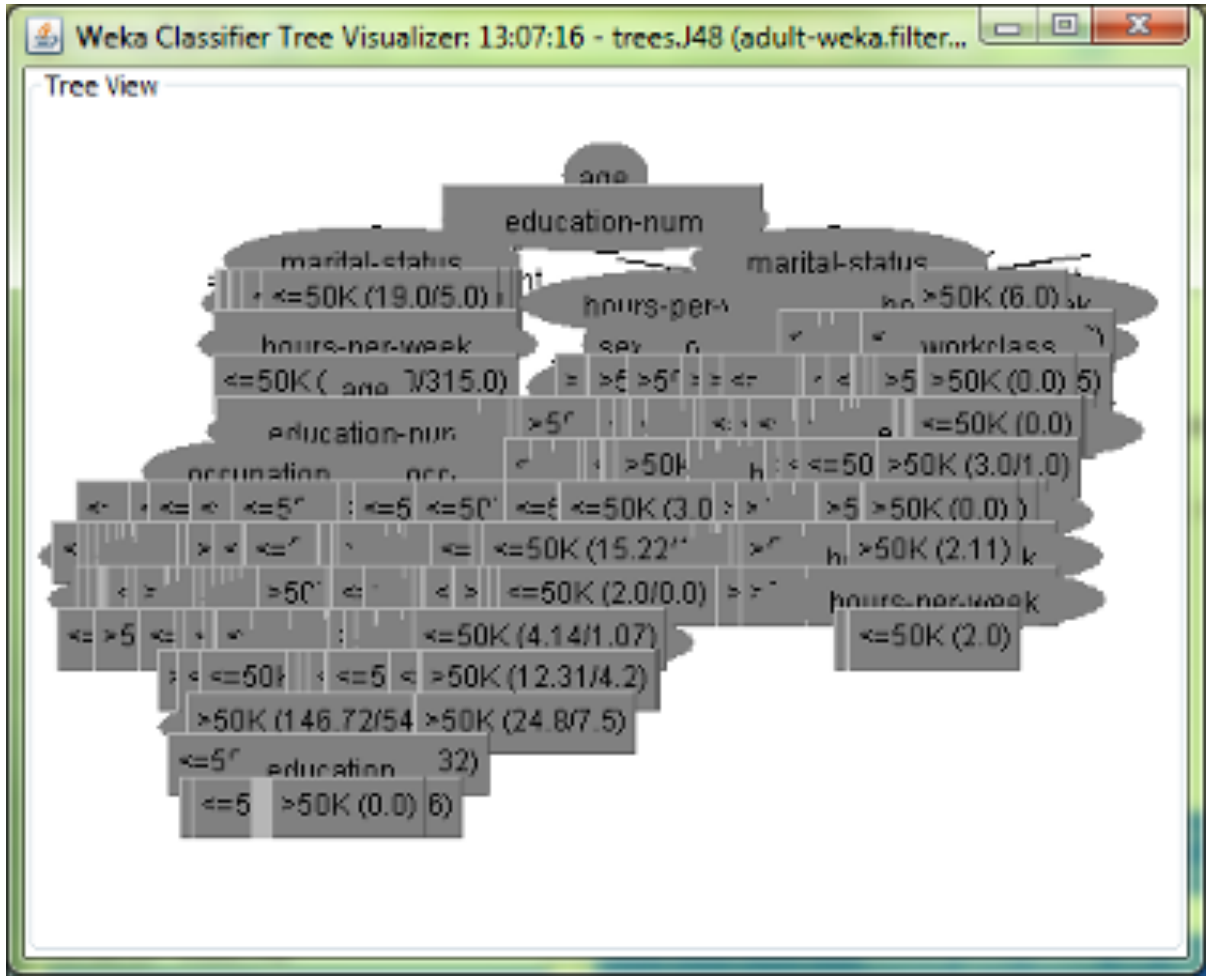
- Classifying a new example with a decision tree is straightforward
    - Start at the root
    - Apply the first test, and determine which branch to follow
    - Iterate until a leaf is reached
    - Apply the label at the leaf





Decision Trees in Weka

- **trees.id3** implements ID3 (Information Gain)
    - Not robust: does not allow missing attributes
- **trees.J48** implements C4.5 (Information Gain scaled by Split Info)



Weka Results for adult.data via J48

```
J48 pruned tree
------------------
age <= 27: <=50K (12012.0/369.0)
age > 27
|   education-num <= 12
|   |   marital-status = Married-civ-spouse
|   |   |   education-num <= 8: <=50K (2323.0/304.0)
|   |   |   education-num > 8
|   |   |   |   hours-per-week <= 35: <=50K (1410.0/315.0)
|   |   |   |   hours-per-week > 35
|   |   |   |   |   age <= 35: <=50K (2760.0/833.0)
|   |   |   |   |   age > 35
|   |   |   |   |   |   education-num <= 9
|   |   |   |   |   |   |   occupation = Tech-support
|   |   |   |   |   |   |   |   workclass = Private: >50K (55.88/23.32)
|   |   |   |   |   |   |   |   workclass = Self-emp-not-inc: <=50K (1.03/0.01)
|   |   |   |   |   |   |   |   workclass = Self-emp-inc: >50K (0.0)
|   |   |   |   |   |   |   |   workclass = Federal-gov: >50K (10.35/3.24)
|   |   |   |   |   |   |   |   workclass = Local-gov: >50K (1.03/0.02)
|   |   |   |   |   |   |   |   workclass = State-gov: <=50K (5.17/1.05)
|   |   |   |   |   |   |   |   workclass = Without-pay: >50K (0.0)
|   |   |   |   |   |   |   |   workclass = Never-worked: >50K (0.0)
```

Weka Results for adult.data via J48

```
Attributes:   age, workclass, education, education-num, marital-status, occupation,
    relationship, race, sex, hours-per-week, native-country, class
Test mode:split 66.0% train, remainder test
Number of Leaves  : 549
Size of the tree :      682
Time taken to build model: 8.4 seconds
=== Evaluation on test split ===
Correctly Classified Instances       13938               83.9335 %
Incorrectly Classified Instances      2668               16.0665 %
Kappa statistic                          0.5337
Mean absolute error                      0.2263
Root mean squared error                  0.3399
Relative absolute error                 62.4059 %
Root relative squared error             80.3307 %
Total Number of Instances            16606
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.602     0.088      0.675     0.602     0.636      0.865    >50K
                 0.912     0.398      0.883     0.912     0.897      0.865    <=50K
Weighted Avg.    0.839     0.326      0.834     0.839     0.836      0.865

=== Confusion Matrix ===
a     b   <-- classified as
  2335  1543 |     a = >50K
  1125 11603 |     b = <=50K
```

Acknowledgements

- Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.
- G Cormode [Warwick, CS910]
- F Ciucu [Warwick, CS430/CS910]

Part D: Naïve Bayes & SVM

Bayesian Classification Methods

- Bayesian classifiers are statistical classifiers based on Bayes' theorem.
  - Predicts the probability that a given tuple belongs to a particular class.
- An example of a simple Bayesian classifier is the Naïve Bayesian classifier:
  - Assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called *class-conditional independence*.

Bayes Theorem

- Let $\mathbf{X}$ be a data tuple, which in Bayesian terms is considered *evidence*, described by measurements made on a set of n attributes.
- Let H be some hypothesis, such as that data tuple $\mathbf{X}$ belongs to class C.
- For classification problems, we want to determine $P(H \mid \mathbf{X})$ - the probability that the hypothesis H holds, given the evidence $\mathbf{X}$.
    - Or, the probability that tuple $\mathbf{X}$ belongs to class C, given that we know the attribute description of $\mathbf{X}$.

Bayes Theorem

- $P(H \mid \mathbf{X})$ is the *posterior probability* (or *a posteriori probability*) of H conditioned on $\mathbf{X}$.
  - Example: Let our data tuples describe people by area and income, and that $\mathbf{X}$ is a person living in Coventry with an income of £25,000. Let H be the hypothesis that the person will attend *Folklorica* at the Warwick Arts Centre. $P(H \mid \mathbf{X})$ reflects the probability that person $\mathbf{X}$ will attend *Folklorica* given that we know their area and income.

Bayes Theorem

- $P(H)$ is the *prior probability* (or *a priori probability*) of H.
  - From the previous example, $P(H)$ is the probability that any person will attend *Folklorica*, regardless of location or income.
- $P(\mathbf{X} \mid H)$ is the posterior probability of $\mathbf{X}$ conditioned on H.
  - From the previous example, $P(\mathbf{X} \mid H)$ is the probability that a person, $\mathbf{X}$, lives in Coventry and has an income of £25,000, given that we know that they will attend *Folklorica*.

Bayes Theorem

- $P(\mathbf{X})$ is the *prior probability* (or *a priori probability*) of $\mathbf{X}$.
    - From the previous example, $P(\mathbf{X})$ is the probability that our person lives in Coventry and earns £25,000.

- $P(H)$, $P(\mathbf{X} \mid H)$ and $P(\mathbf{X})$ can be estimated from the given data, $P(H \mid \mathbf{X})$ can be created using Bayes' Theorem:

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

Naïve Bayes Classifier

- Also known as the *Simple Bayesian classifier*.

1. Let D be a training set of tuples and their associated class labels. Each tuple is an n-dimensional attribute vector $\mathbf{X} = (x_1, x_2, \ldots, x_n)$, depicting n measurements from n attributes $(A_1, A_2, \ldots, A_n)$.

2. Let there be m classes $(C_1, C_2, \ldots, C_m)$. For a tuple $\mathbf{X}$, the classifier will predict that $\mathbf{X}$ belongs to the class having the highest posterior probability, conditioned on $\mathbf{X}$, i.e. the classifier will predict that $\mathbf{X}$ belongs to class C_i if:

$$P(C_i \mid \mathbf{X}) > P(C_j \mid \mathbf{X}) \text{ for } 1 \le j \le m, j \neq i$$

Naïve Bayes Classifier

- Reminder, we can calculate $P(C_i \mid \mathbf{X})$ by:

$$P(C_i \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_i)P(C_i)}{P(\mathbf{X})}$$

- The class C_i for which $P(C_i \mid \mathbf{X})$ is maximised, is called the *maximum posteriori hypothesis*.

Naïve Bayes Classifier

- Reminder, we can calculate $P(C_i \mid \mathbf{X})$ by:

$$P(C_i \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_i)P(C_i)}{\boxed{P(\mathbf{X})}}$$

Constant for all classes.

- The class C_i for which $P(C_i \mid \mathbf{X})$ is maximised, is called the *maximum posteriori hypothesis*.

Naïve Bayes Classifier

3. As $P(\mathbf{X})$ is constant for all classes, only $P(\mathbf{X} \mid C_i)P(C_i)$ needs to be maximised.

- If the class prior probabilities ($P(C_i)$) are not known, then we commonly assume that all classes are equally likely (i.e. $P(C_1) = P(C_2) = \ldots = P(C_m)$).=, meaning we would only need to maximise $P(\mathbf{X} \mid C_i)$.

- Class prior probabilities can be estimated by $P(C_i) = \dfrac{|C_{i,D}|}{|D|}$, where $|C_{i,D}|$ is the number of training tuples of class C_i in D.

Naïve Bayes Classifier

4. Due to the naive assumption of class-conditional independence (which presumes that the attributes' values are conditionally independent of one another, given the class label of the tuple). We can calculate $P(\mathbf{X} \mid C_i)$ as follows:

$$\begin{aligned} P(\mathbf{X} \mid C_i) &= \prod_{k=1}^{n} P(x_k \mid C_i) \\ &= P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i) \end{aligned}$$

- Assuming A_k is categorical, $P(x_k \mid C_i)$ is the number of tuples of class C_i in D having the value x_k for A_k, divided by $|C_{i,D}|$ (the number of tuples of class C_i in D).

Naïve Bayes Classifier

5. We now compute $P(\mathbf{X} | C_i)P(C_i)$ for each class C_i. The classifier predicts that the class label of tuple $\mathbf{X}$ is the class C_i if and only if:

   $$P(\mathbf{X} | C_i)P(C_i) > P(\mathbf{X} | C_j)P(C_j) \text{ for } 1 \leq j \leq m, j \neq i$$

   i.e. the predicted class label is the class C_i for which $P(\mathbf{X} | C_i)P(C_i)$ is the maximum.

Naïve Bayes Classifier

- A further point to raise…
  - Consider the situation where $P(x_k \mid C_i) = 0$ in the training set (e.g., from our previous example $P(area = Coventry \mid visits\_Folklorica = yes) = 0$).
  - Recall the following equation:

$$P(\mathbf{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

$$= P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i)$$

Naïve Bayes Classifier

- A further point to raise…
  - Consider the situation where $P(x_k \mid C_i) = 0$ in the training set (e.g., from our previous example $P(area = Coventry \mid visits\_Folklorica = yes) = 0$).
  - Recall the following equation:

If this is 0…

$$\boxed{P(\mathbf{X} \mid C_i)} = \prod_{k=1}^{n} P(x_k \mid C_i)$$

$$= \boxed{P(x_1 \mid C_i)} \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i)$$

…so is this!

Naïve Bayes Classifier

- Simple solution:
  - We assume our training data, D, is so large that adding one to each count would make a negligible difference, yet would avoid the problem on the previous slide.
    - e.g. add one tuple where a person from Coventry visits Folklorica.
    - $P(area = Coventry \mid visits\_Folklorica = yes)$ is no longer 0, but is very small!
  - This process is known as *Laplacian Correction* (more info in book).

Naïve Bayes Classifier

Example

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	low	no	excellent	no

Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Naïve Bayes Classifier

Example

- Let's say we want to classify $X = (age = youth, income = medium, student = yes, credit\_rating = fair)$

- Let's first calculate $P(C_i)$ for all classes:

  - $P(buys\_computer = yes) = \frac{9}{14} = 0.643$
  - $P(buys\_computer = no) = \frac{5}{14} = 0.357$

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Naïve Bayes Classifier

Example

- To compute $P(\mathbf{X}|C_i)$, for $i = 1,2$, we need:
  - $P(age = youth | buys\_computer = yes) = \frac{2}{9} = 0.222$
  - $P(age = youth | buys\_computer = no) = \frac{3}{5} = 0.6$
  - $P(income = medium | buys\_computer = yes) = \frac{4}{9} = 0.444$
  - $P(income = medium | buys\_computer = no) = \frac{2}{5} = 0.400$

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Naïve Bayes Classifier

Example

- To compute $P(\mathbf{X}|C_i)$, for $i = 1,2$, we need:
  - $P(student = yes|buys\_computer = yes) = \frac{6}{9} = 0.667$
  - $P(student = yes|buys\_computer = no) = \frac{1}{5} = 0.200$
  - $P(credit\_rating = fair|buys\_computer = yes) = \frac{6}{9} = 0.667$
  - $P(credit\_rating = fair|buys\_computer = no) = \frac{2}{5} = 0.400$

$$P(C_i \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C_i)P(C_i)}{P(\mathbf{X})}$$

Naïve Bayes Classifier

Example

- Next, compute $P(\mathbf{X} \mid buys\_computer = yes)$:

$$\begin{aligned} P(\mathbf{X} \mid buys\_computer = yes) = {} & P(age = youth \mid buys\_computer = yes) \\ & \times P(income = medium \mid buys\_computer = yes) \\ & \times P(student = yes \mid buys\_computer = yes) \\ & \times P(credit\_rating = fair \mid buys\_computer = yes) \\ & = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044 \end{aligned}$$

$$P(\mathbf{X} \mid buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019$$

Naïve Bayes Classifier

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

Example

- Finally, find the class which maximises $P(\mathbf{X}|C_i)P(C_i)$:

$$P(\mathbf{X}|buys\_computer = yes)P(buys\_compter = yes) = 0.044 \times 0.643 = 0.028$$

$$P(\mathbf{X}|buys\_computer = no)P(buys\_compter = no) = 0.019 \times 0.357 = 0.007$$

- Therefore, the classifier predicts $buys\_computer = yes$ for tuple $\mathbf{X}$.

Naïve Bayes in Weka

```
Attribute                          >50K   <=50K
                                  (0.24)  (0.76)
===============================================
age
  mean                            44.2752 36.8722
  std. dev.                       10.5585 14.1039
  weight sum                        11687   37155
  precision                             1       1

workclass
  Private                          7388.0 26520.0
  Self-emp-not-inc                 1078.0  2786.0
  Self-emp-inc                      939.0   758.0
  Federal-gov                       562.0   872.0
  Local-gov                         928.0  2210.0
  State-gov                         531.0  1452.0
  Without-pay                         3.0    20.0
  Never-worked                        1.0    11.0
  [total]                         11430.0 34629.0
…
Time taken to build model: 0.51 seconds
```

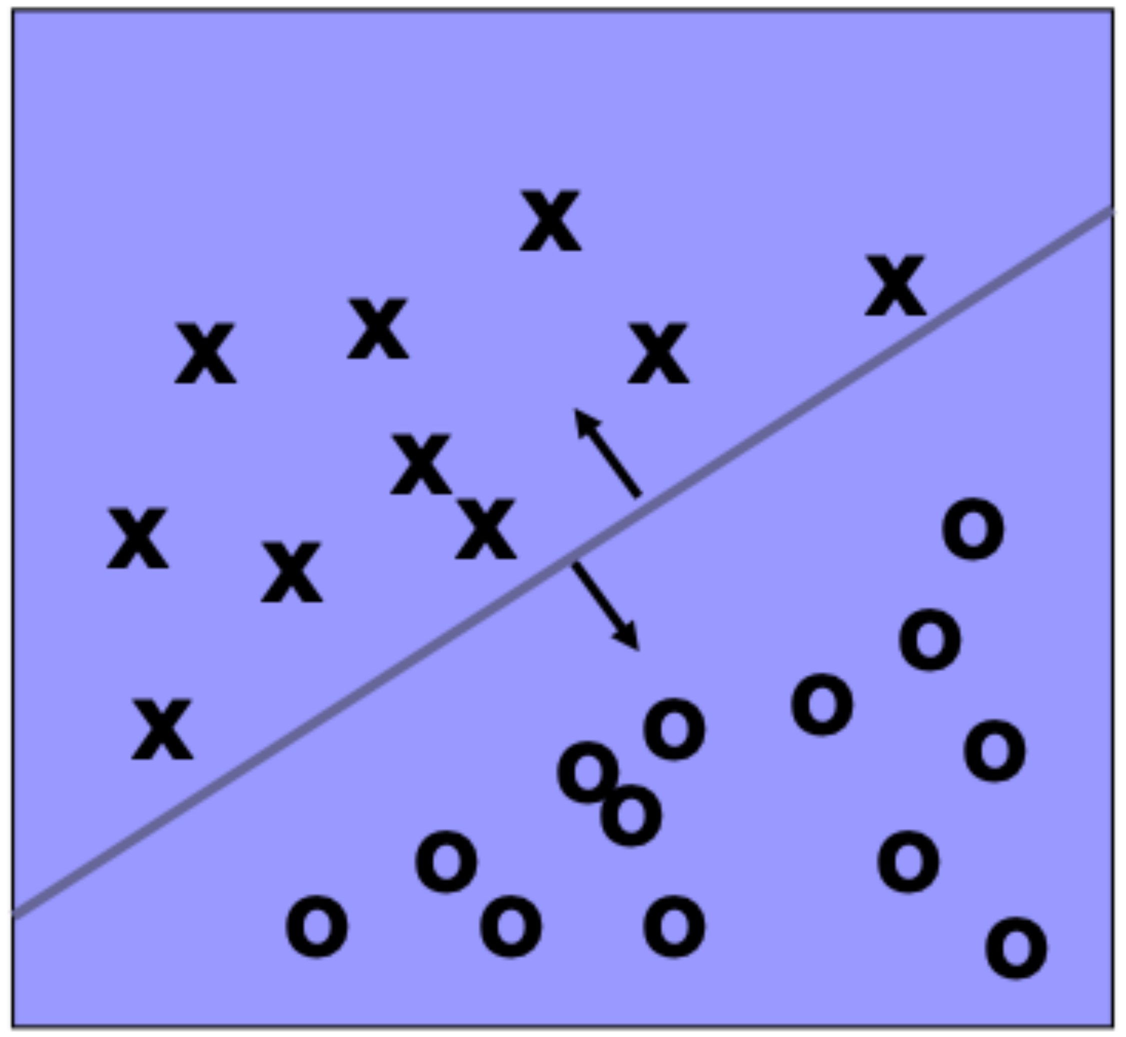
Note: Weka applies the Laplacian Correction to categorical attributes.

Naïve Bayes in Weka

```
Scheme:weka.classifiers.bayes.NaiveBayes
Relation:     adult-weka.filters.unsupervised.attribute.Remove-R3,11-12
Instances:    48842
Test mode:split 66.0% train, remainder test
=== Evaluation on test split ===
Correctly Classified Instances       13519               81.4103 %
Incorrectly Classified Instances      3087               18.5897 %
Kappa statistic                          0.5295
Mean absolute error                      0.2043
Root mean squared error                  0.3681
Relative absolute error                 56.3263 %
Root relative squared error             86.9948 %
Total Number of Instances            16606
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.751     0.167      0.579     0.751     0.654      0.885    >50K
                 0.833     0.249      0.917     0.833     0.873      0.885    <=50K
Weighted Avg.    0.814     0.23       0.838     0.814     0.822      0.885
=== Confusion Matrix ===
     a     b   <-- classified as
  2913   965 |     a = >50K
  2122 10606 |     b = <=50K
```

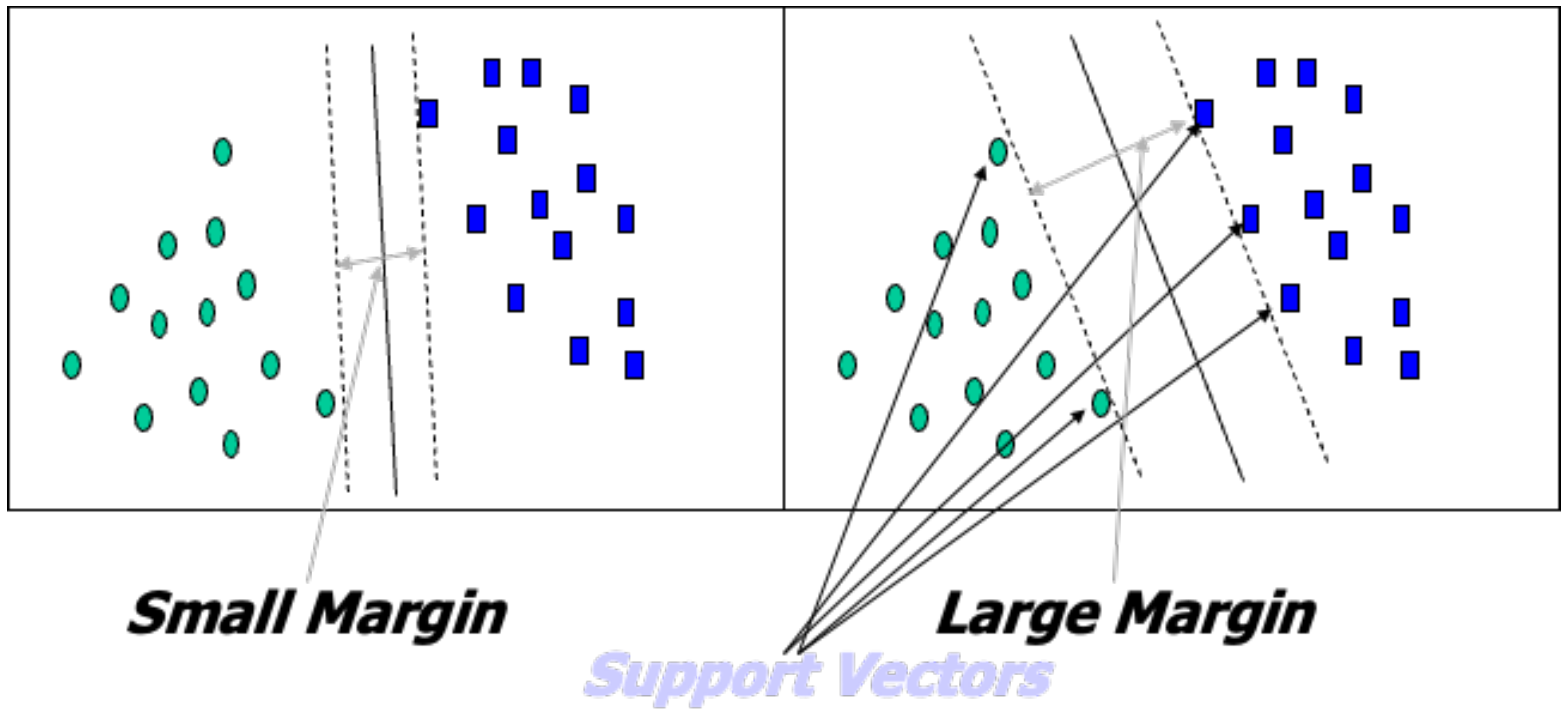
Linearly Separable Data

- Models seen so far have tried to partition up the data space
    - **Decision Trees**: partition into (hyper)rectangles (generalisation to higher dimension)
- Sometimes the data is *linearly separable*
    - Examples of one class fall one side of a line / hyperplane
    - Other class is on other side of the plane
- *Linear classifiers* aim to find 'best' plane
    - Allows a simple description of the model
        - Defined by the separating hyperplane
    - Early examples: *winnow algorithm*, *perceptron* (used in neural networks)



Support Vector Machines (SVM)

- Support Vector Machines (SVM) seek the best plane.
  - When the data is linearly separable, many hyperplanes can separate.
  - SVM seeks the plane that maximises the margin between the classes.



Support Vector Machines (SVM)

1. SVM begins by transforming the original training data into a higher dimension.

2. Once the data has been transformed, SVM searches for a hyperplane/decision boundary which separates the tuples of one class from another.

Support Vector Machines (SVM)

As an Optimisation Problem

- A hyperplane is written as $\mathbf{W} \cdot \mathbf{X} + b = 0$
    - $\mathbf{W}$: (d-dimensional) weight vector, where d is the number of attributes
    - $\mathbf{X}$: points in (d-dimensional) space
    - b: bias
- Any $\mathbf{X}$ "above" hyperplane has $\mathbf{W} \cdot \mathbf{X} + b > 0$
    - $\mathbf{W} \cdot \mathbf{X} + b < 0$: $\mathbf{X}$ is "below" the hyperplane
- Find weights $\mathbf{W}$ so that
    - $\mathbf{W} \cdot \mathbf{X} + b \geq 1$ for $y = +1$
    - $\mathbf{W} \cdot \mathbf{X} + b \leq -1$ for $y = -1$
- Combining the above gives us: $y(\mathbf{W} \cdot \mathbf{X} + b) \geq 1$ for all (y, X)

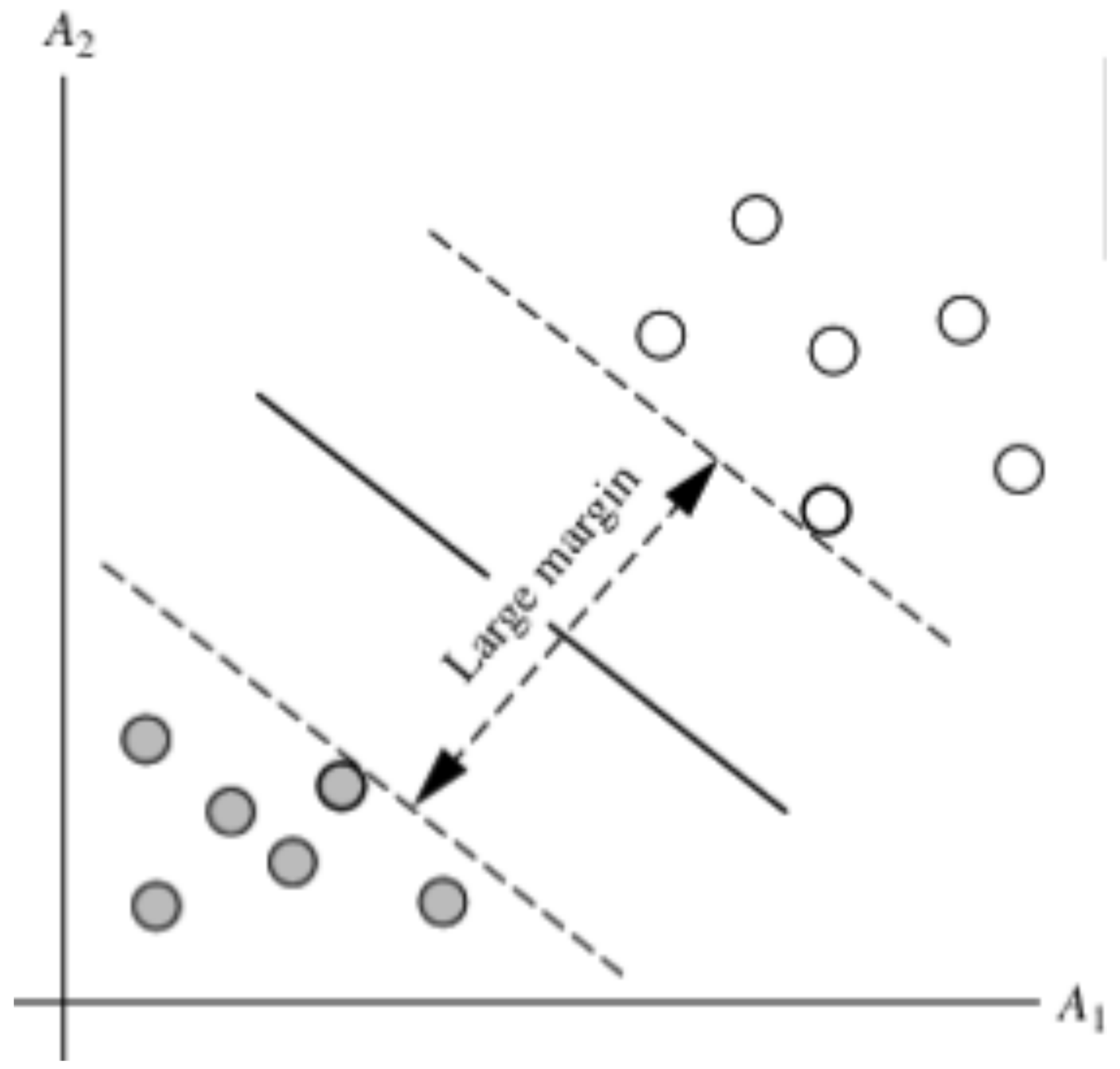


Figure from: Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Support Vector Machines (SVM)

As an Optimisation Problem

- Training tuples that fall on the hyperplanes (lets call them H_1 and H_2) are known as *Support Vectors*.

- The distance of a point on H_1 (or H_2) from the plane is $\frac{1}{\boxed{\|\mathbf{W}\|}}$ ← Euclidean norm of $\mathbf{W}$.

- Hence, the maximal margin is $\frac{2}{\|W\|}$.

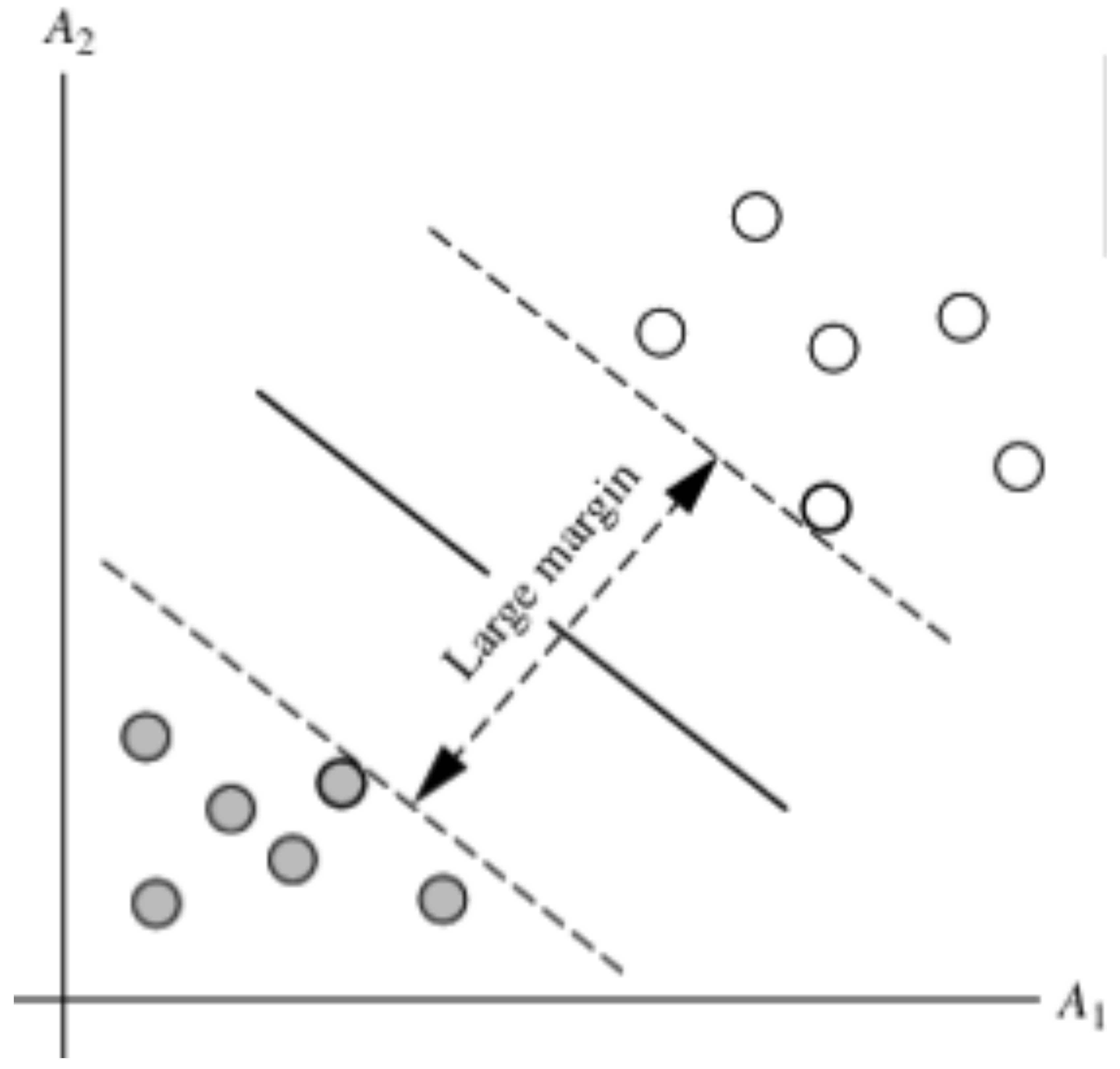


Figure from: Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Support Vector Machines (SVM)

As an Optimisation Problem

- This then becomes an optimisation problem…
- We want to maximise $\frac{2}{||W||}$.
    - Subject to $y(\mathbf{W} \cdot \mathbf{X} + b) \geq 1$, for all (y, X)

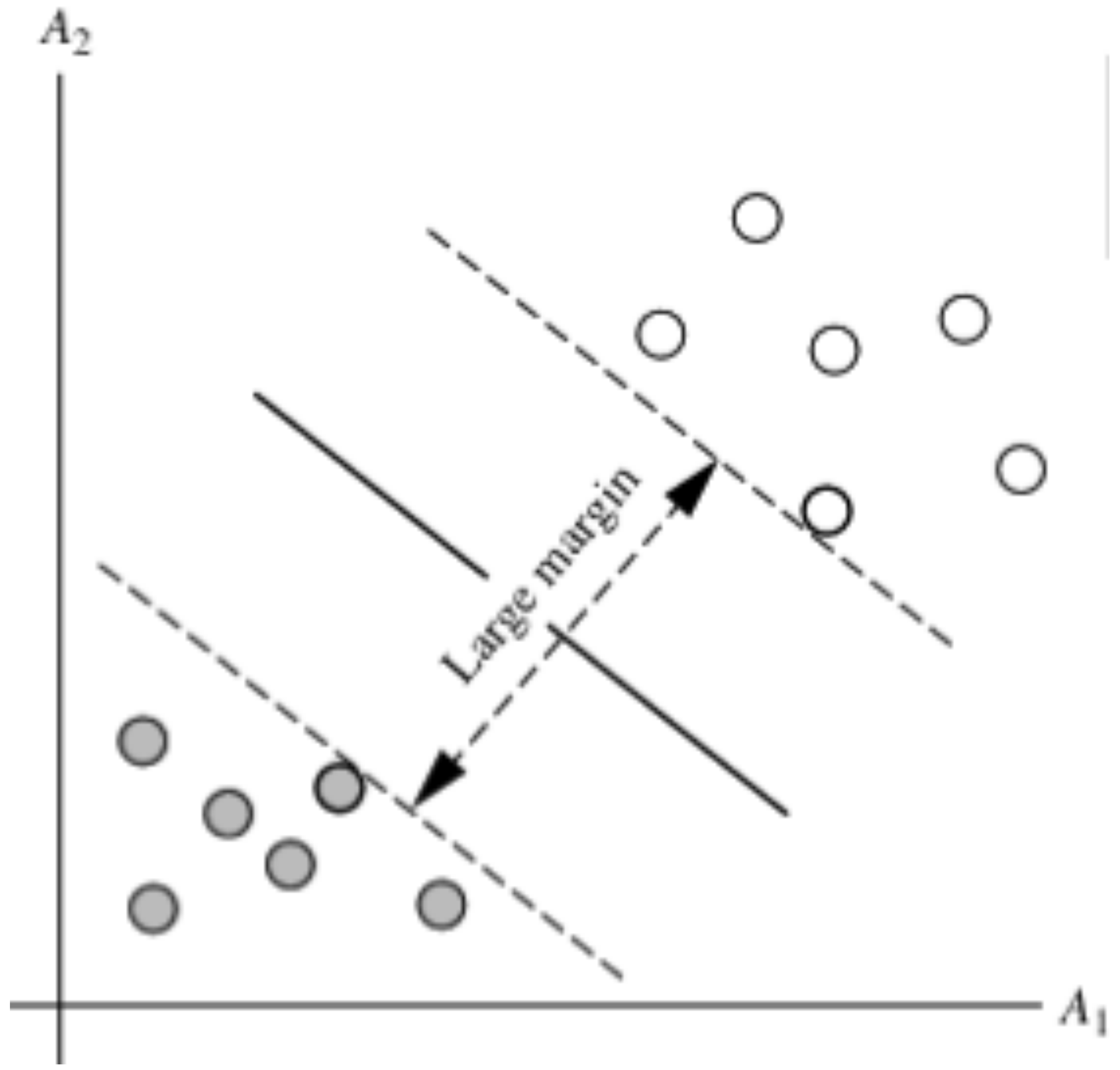


Figure from: Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Support Vector Machines (SVM)

Advantages

- Advantages:
  - Given $\mathbf{W}$ and b, SVM is very fast to apply to new examples.
    - If $\mathbf{W} \cdot \mathbf{X} + b > 0$, output $+1$, else output -1
  - Accuracy is generally good
    - Margin determined by only the few support vectors.
  - Overfitting is avoided

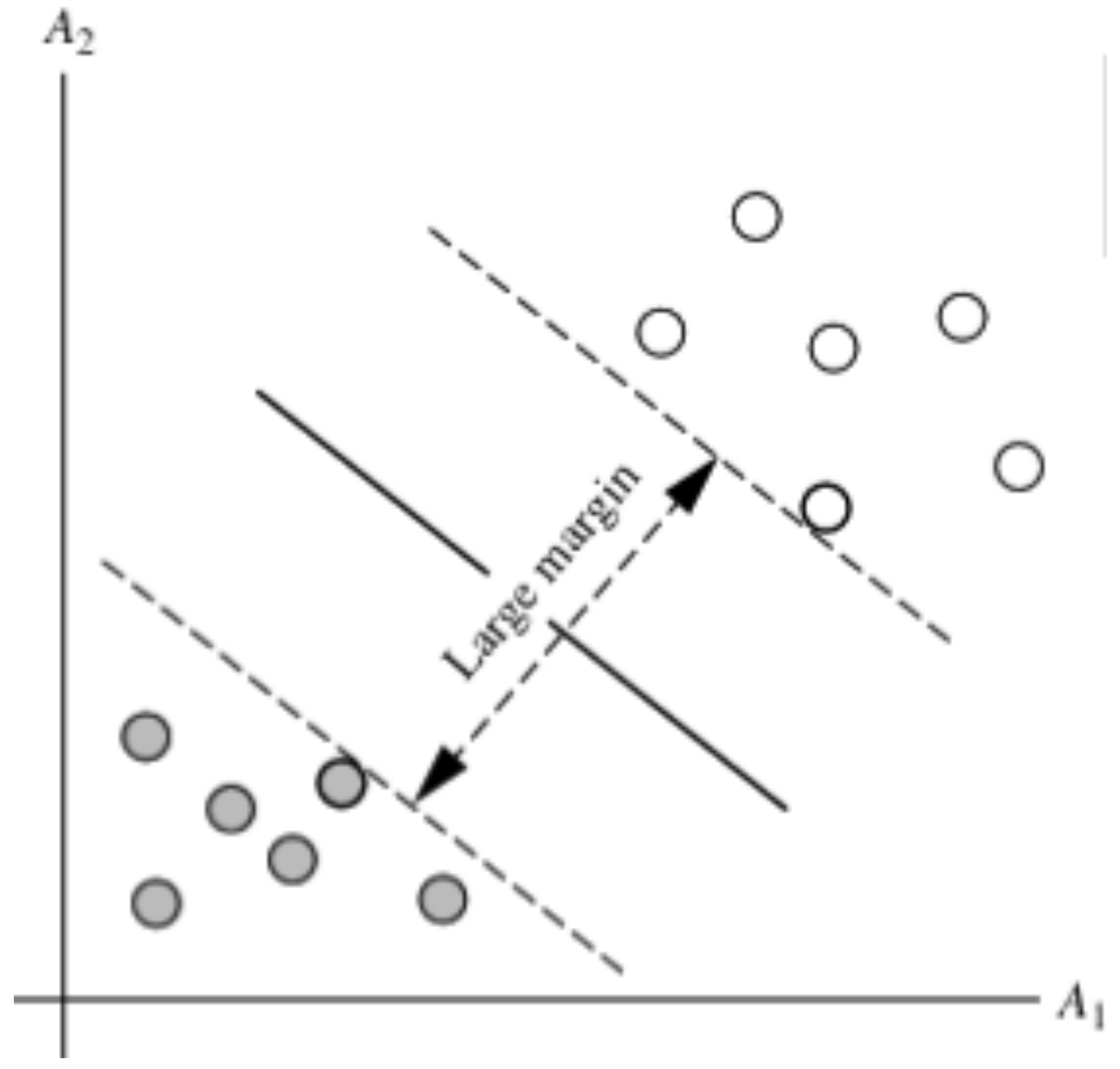


Figure from: Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Support Vector Machines (SVM)

Disadvantages

- Disadvantages:
  - Whilst SVM is fast to apply, it is slow to train!
  - Defined only for two classes (need e.g. one-vs.-all for more classes)
  - Assumes data is linearly separable, but this is not true in general.

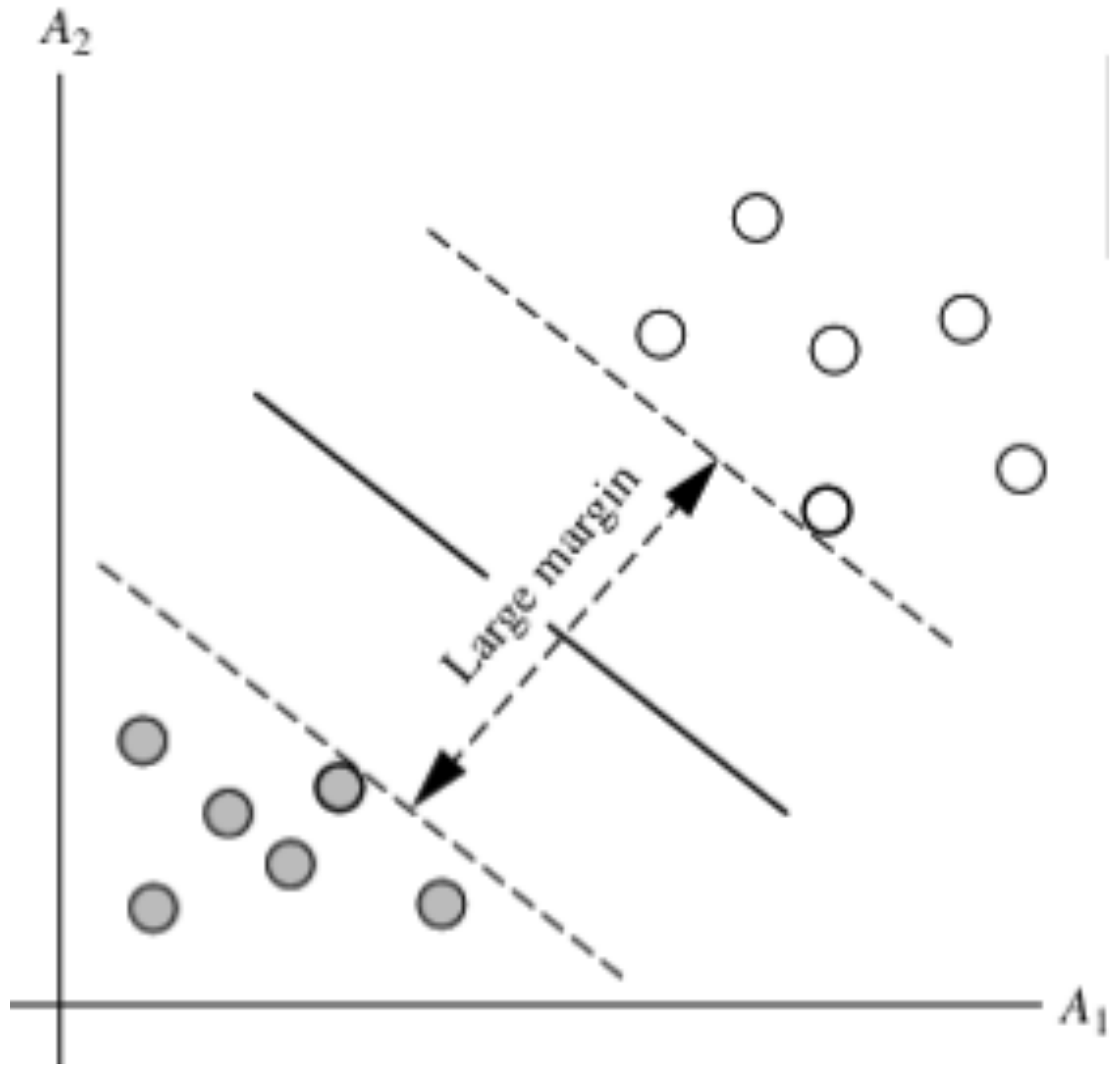


Figure from: Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Support Vector Machines (SVM)

Evaluation on adult.data in Weka

```
Scheme:weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K
    "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"
Relation:     adult-weka.filters.unsupervised.attribute.Remove-R3,11-12
Instances:    48842
Attributes:   12
Test mode:split 66.0% train, remainder test
=== Classifier model (full training set) ===
Kernel used:  Linear Kernel: K(x,y) = <x,y>
Classifier for classes: >50K, <=50K
Time taken to build model: 3472.87 seconds
=== Evaluation on test split ===
Correctly Classified Instances       13912               83.7769 %
Incorrectly Classified Instances      2694               16.2231 %
Total Number of Instances            16606
=== Detailed Accuracy By Class ===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.556     0.076      0.689     0.556     0.615      0.74     >50K
                 0.924     0.444      0.872     0.924     0.897      0.74     <=50K
Weighted Avg.    0.838     0.358      0.829     0.838     0.831      0.74
=== Confusion Matrix ===
     a     b   <-- classified as
  2155  1723 |     a = >50K
   971 11757 |     b = <=50K
```

Support Vector Machines (SVM)

More Information

- We’ve only scratched the surface of SVM…
- Find out more in **CS429/CS909 (Data Mining)** and **CS342 (Machine Learning)**.

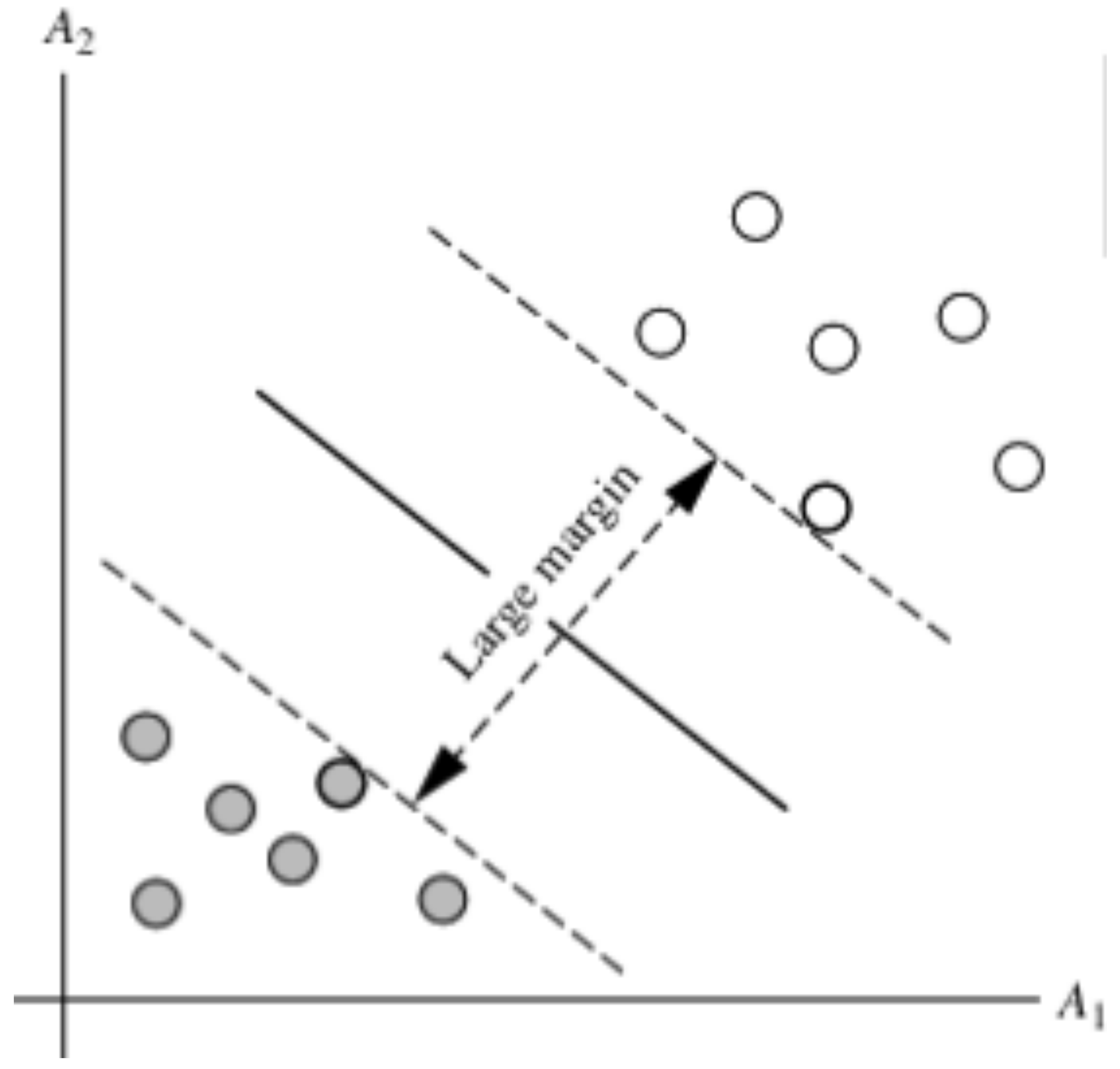


Figure from: Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.

Comparisons of Classifiers on adult.data

Name	Accuracy	AUC	FP Rate	Time to build	Kappa Statistic
Majority Class	0.766	0.5	0.766	0.1s	0
k Nearest Neighbour	0.834	0.874	0.299	0.1s	0.5359
Decision tree (J48)	0.839	0.865	0.326	8.4s	0.5377
Naïve Bayes	0.814	0.885	0.23	0.5s	0.5295
SVM	0.837	0.74	0.358	3472s	0.5141
Logistic Regression	0.837	0.888	0.35	31s	0.5175

- Tradeoffs between accuracy, FP rate, AUC, time to build, time to apply, and interpretability of resulting classifier

- Different data sets (or different objectives) will show different behaviour

Summary of Classifiers

- Many different ways to construct classifiers to predict values.
- **General framework**: build on training data, evaluate on test data.
- **Decision tree** classifiers constructed by recursively splitting data.
- **Naïve Bayes** makes independence assumption between variables.
- **SVM** classifiers find linear separation.
- Can construct and compare classifiers using Weka.

Acknowledgements

- Han, J., Pei, J. and Kamber, M., 2012. *Data Mining: Concepts and Techniques*. Elsevier.
- G Cormode [Warwick, CS910]
- F Ciucu [Warwick, CS430/CS910]