THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

More Data Mining with Weka

Class 2 – Lesson 1

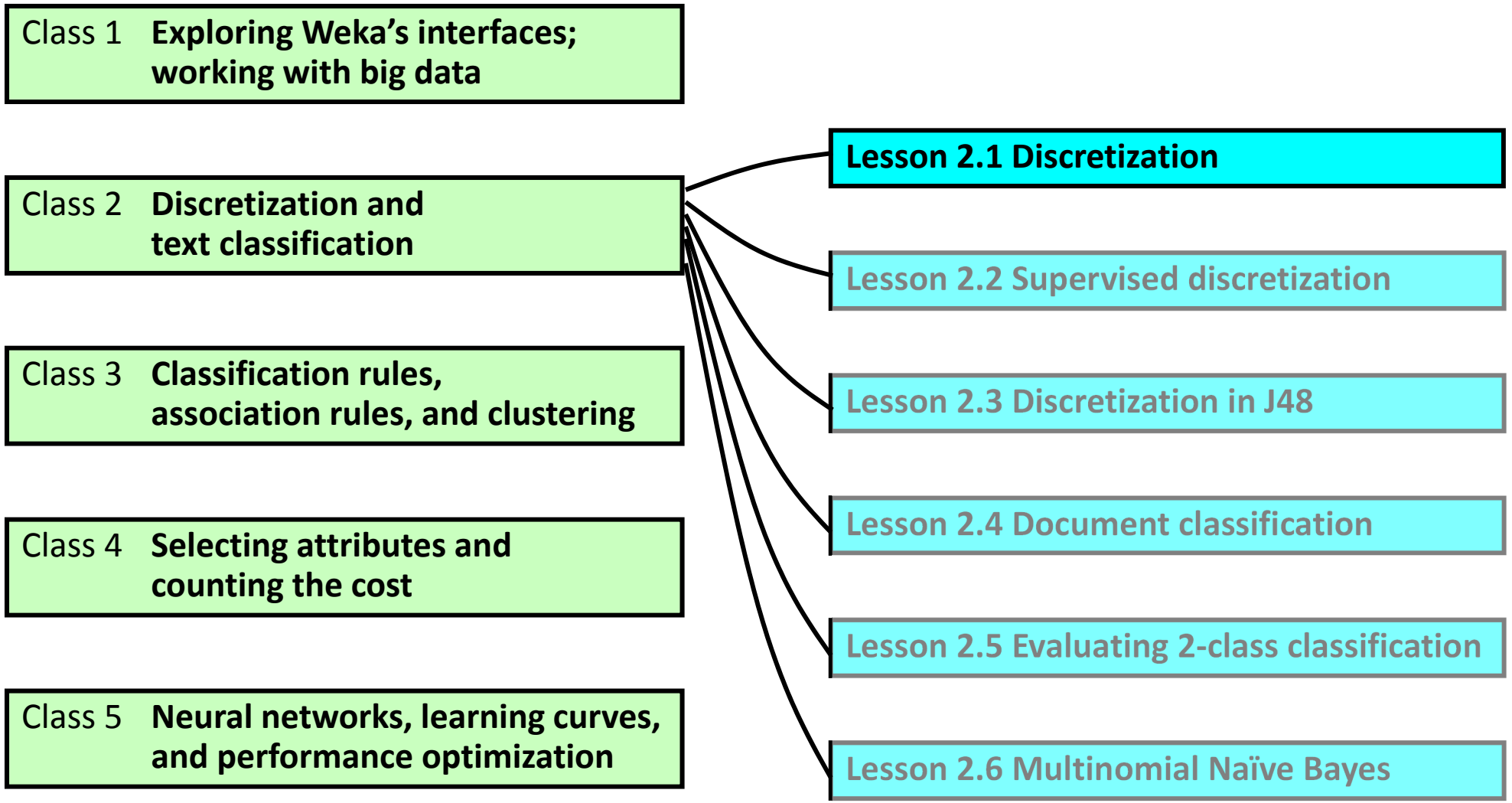
Discretizing numeric attributes

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 2.1: Discretizing numeric attributes



Lesson 2.1: Discretizing numeric attributes

Transforming numeric attributes to nominal

- Equal-width binning
- Equal-frequency binning (“histogram equalization”)
- How many bins?
- Exploiting ordering information?

Lesson 2.1: Discretizing numeric attributes

Equal-width binning

- Open ionosphere.arff; use J48 — 91.5% (35 nodes)
  - *a01: values –1 (38) and +1 (313) [check with Edit…]*
  - *a03: scrunched up towards the top end*
  - *a04: normal distribution?*
- unsupervised>attribute>discretize: examine parameters
- 40 bins; all attributes; look at values — 87.7% (81 nodes)
  - *a01:*
  - *a03:*
  - *a04: looks normal with some extra –1's and +1's*
- 10 bins — 86.6% (51 nodes)
- 5 bins — 90.6% (46 nodes)
- 2 bins — 90.9% (13 nodes)

Lesson 2.1: Discretizing numeric attributes

Equal-frequency binning

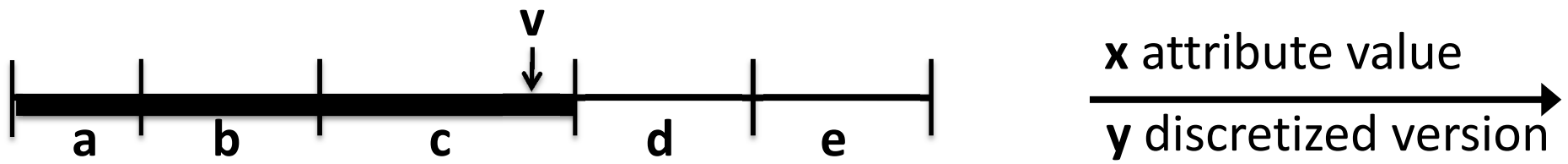
- ionosphere.arff; use J48 — 91.5% (35 nodes)
- equal-frequency, 40 bins — 87.2% (61 nodes)
  - *a01: only 2 bins*
  - *a03: flat with peak at +1 and small peaks at –1 and 0 (check Edit… window)*
  - *a04: flat with peaks at –1, 0, and +1*
- 10 bins — 89.5% (48 nodes)
- 5 bins — 90.6% (28 nodes)
- 2 bins (look at attribute histograms!) — 82.6% (47 nodes)
- How many bins?

$$\propto \sqrt{number\ of\ instances}$$

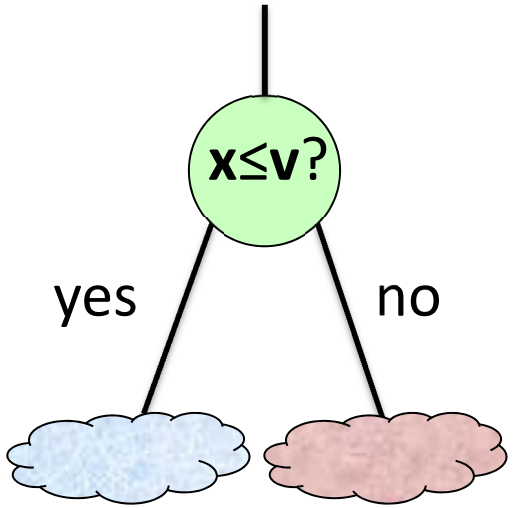
(called “proportional k-interval discretization”)

Lesson 2.1: Discretizing numeric attributes

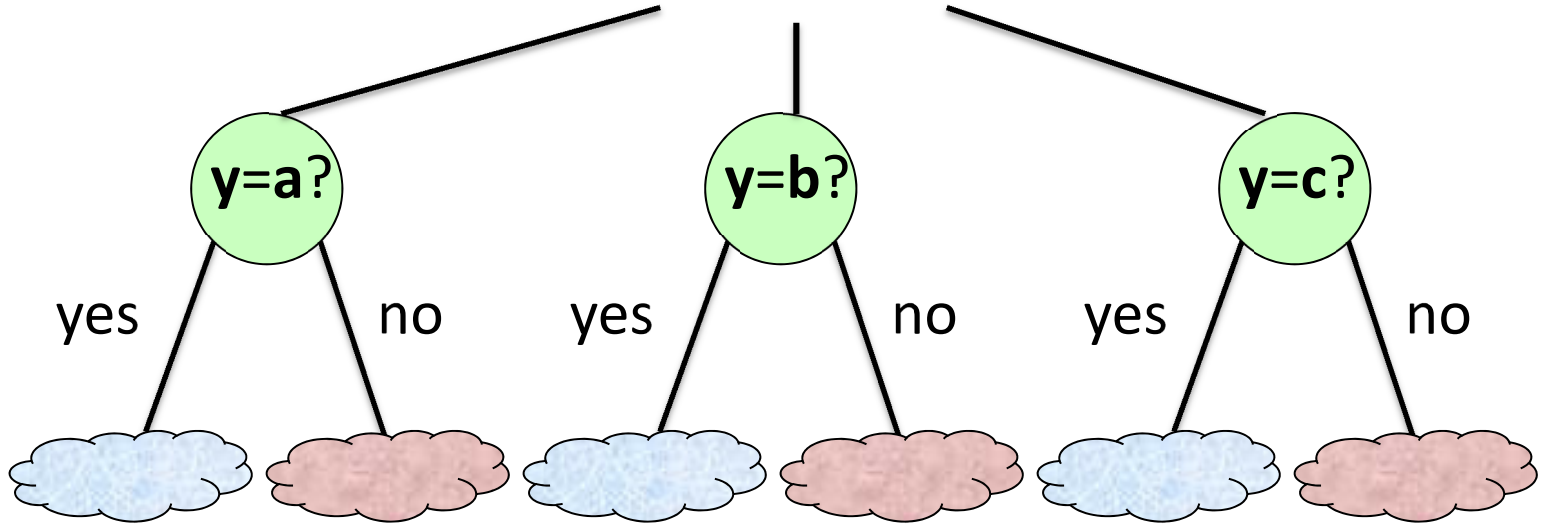
How to exploit ordering information? – what's the problem?



before



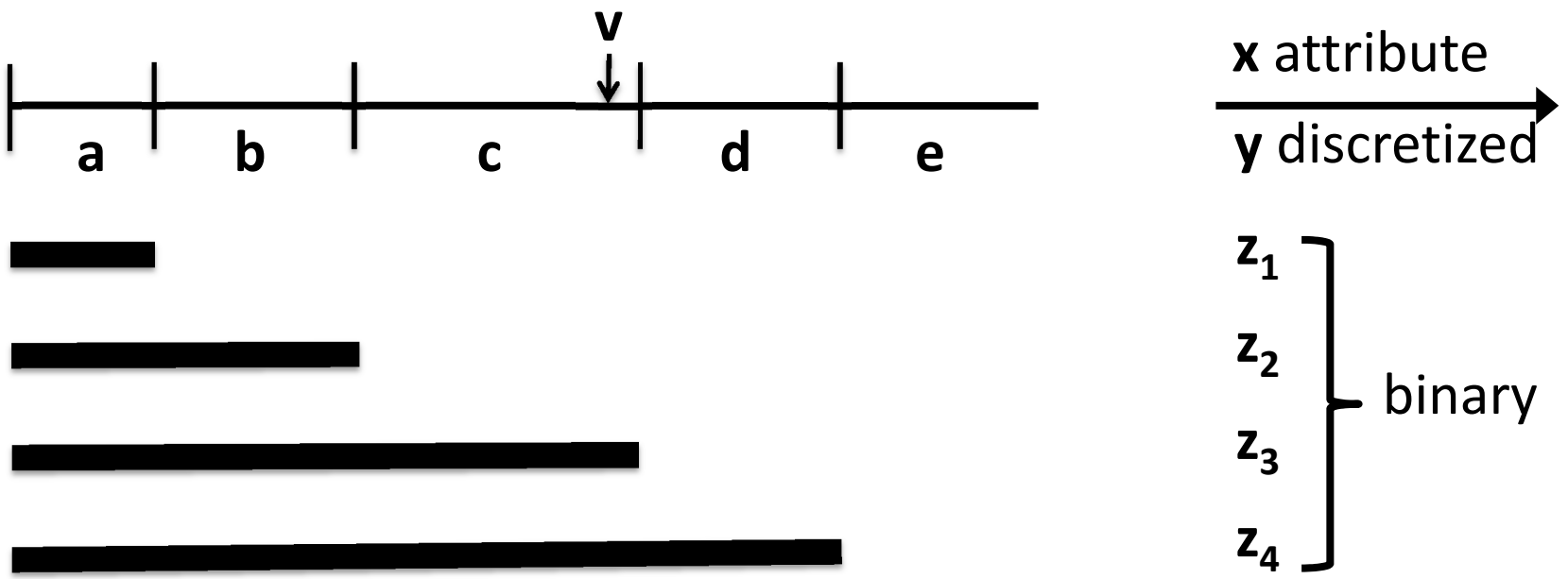
after



Lesson 2.1: Discretizing numeric attributes

How to exploit ordering information? – a solution

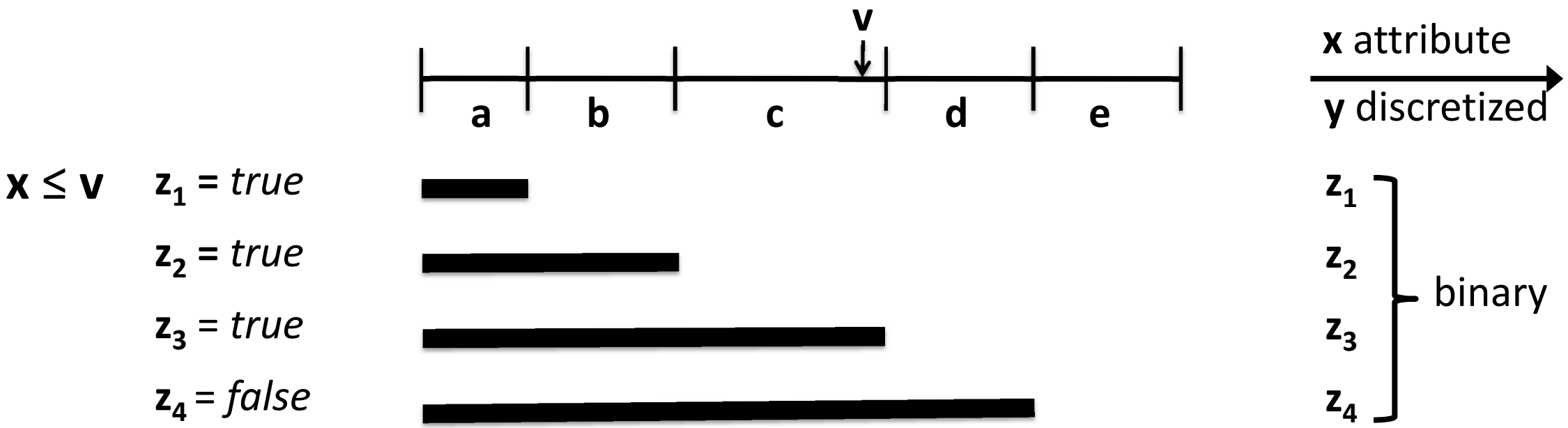
- Transform a discretized attribute with *k* values into *k*–1 binary attributes
- If the original attribute's value is *i* for a particular instance, set the first *i* binary attributes to *true* and the remainder to *false*



Lesson 2.1: Discretizing numeric attributes

How to exploit ordering information? – a solution

- Transform a discretized attribute with *k* attributes into *k*–1 binary attributes
- If the original attribute's value is *i* for a particular instance, set the first *i* binary attributes to *true* and the remainder to *false*

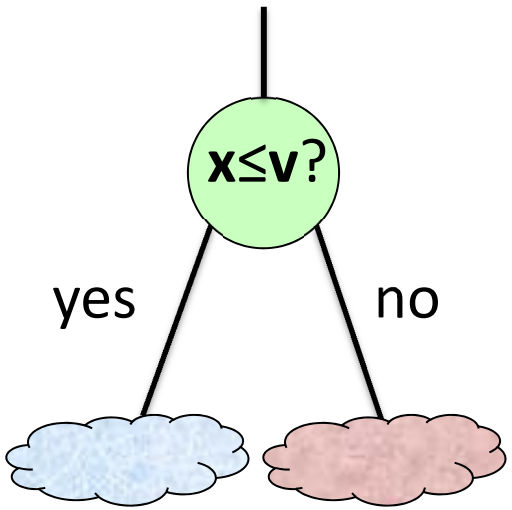


Lesson 2.1: Discretizing numeric attributes

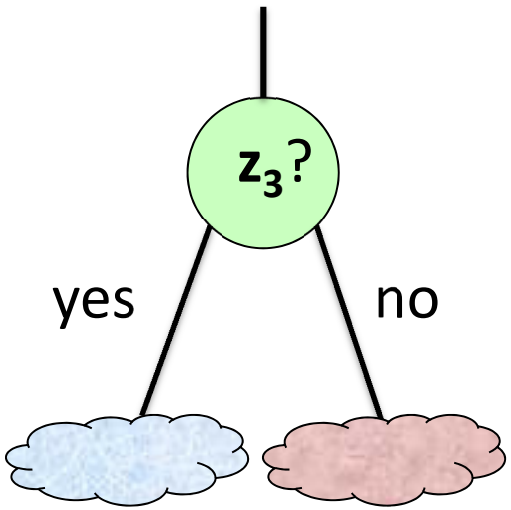
How to exploit ordering information? – a solution

- Transform a discretized attribute with *k* attributes into *k*–1 binary attributes
- If the original attribute's value is *i* for a particular instance, set the first *i* binary attributes to *true* and the remainder to *false*

before



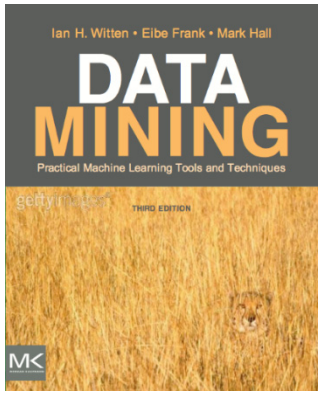
after



Lesson 2.1: Discretizing numeric attributes

- Equal-width binning
- Equal-frequency binning (“histogram equalization”)
- How many bins?
- Exploiting ordering information

- Next … take the class into account (“supervised” discretization)



Course text

- Section 7.2 *Discretizing numeric attributes*

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

More Data Mining with Weka

Class 2 – Lesson 2

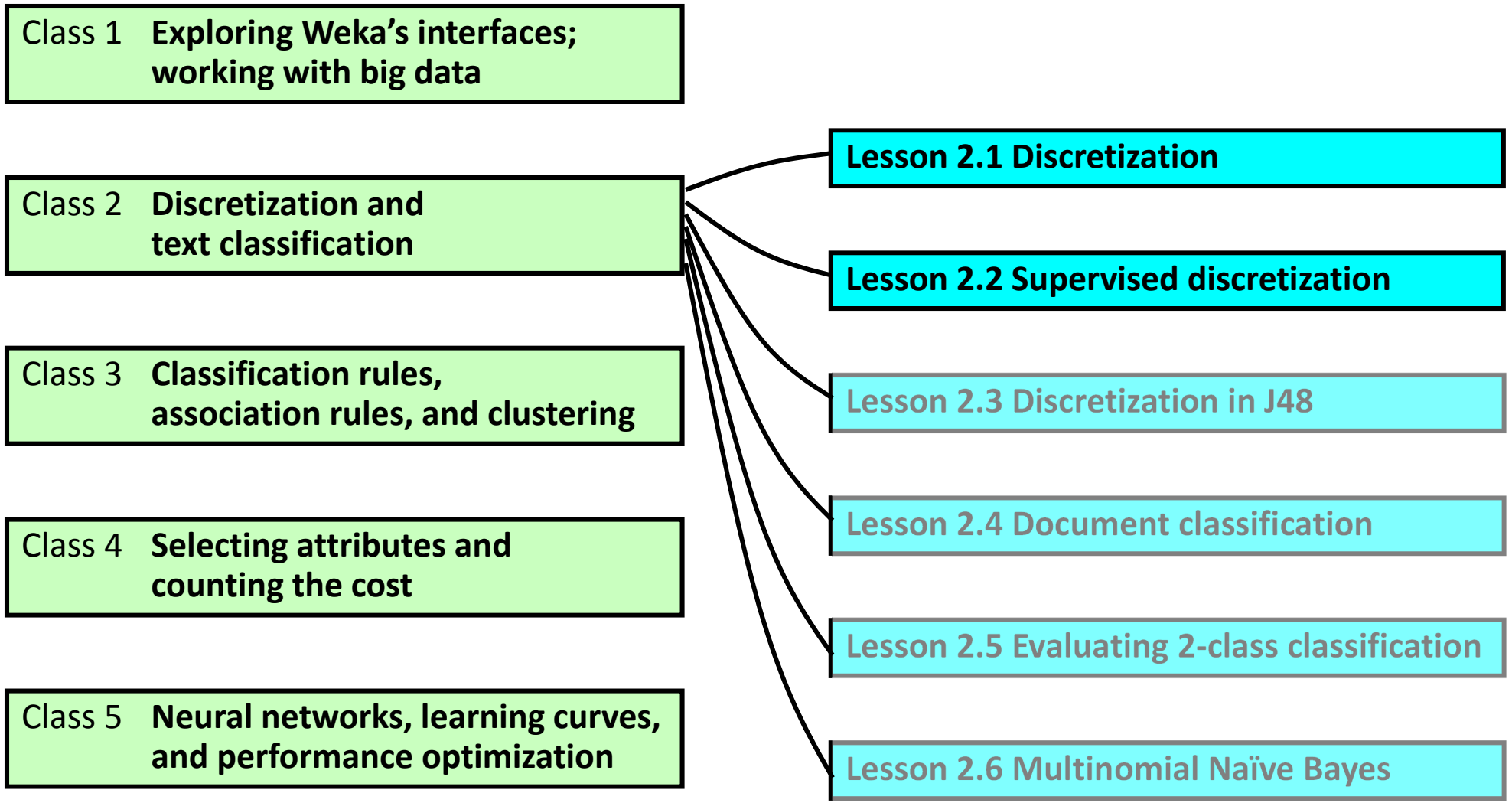
Supervised discretization and the FilteredClassifier

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

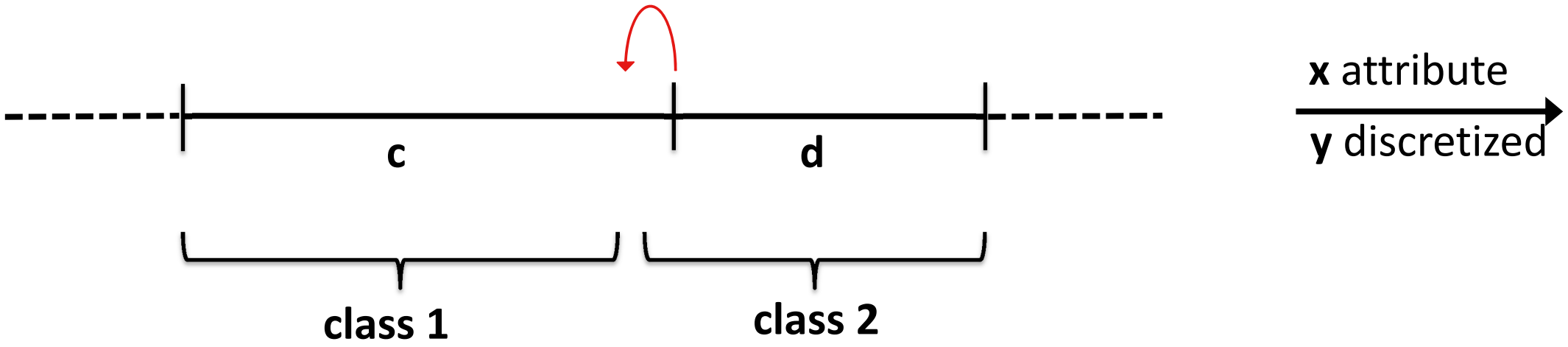
Lesson 2.2: Supervised discretization and the FilteredClassifier



Lesson 2.2: Supervised discretization and the FilteredClassifier

Transforming numeric attributes to nominal

- What if all instances in a bin have one class, and all instances in the next higher bin have another class except for the first, which has the original class?



- Take the class values into account – supervised discretization

Lesson 2.2: Supervised discretization and the FilteredClassifier

Transforming numeric attributes to nominal

- Use the entropy heuristic (pioneered by C4.5 – J48 in Weka)
- e.g. *temperature* attribute of weather.numeric.arff dataset

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	no	yes	yes	no
						yes	yes				

4 yes, 1 no | 5 yes, 4 no

entropy = 0.934 bits

amount of information required to specify the individual values of *yes* and *no* given the split

- Choose split point with smallest entropy (largest information gain)
- Repeat recursively until some stopping criterion is met

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	no	yes	yes	no
						yes	yes				

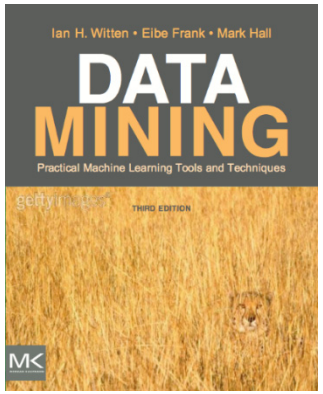
Lesson 2.2: Supervised discretization and the FilteredClassifier

Supervised discretization: information-gain-based

- ionosphere.arff; use J48 — 91.5% (35 nodes)
- supervised>attribute>discretize: examine parameters
- apply filter: attributes range from 1–6 bins
- Use J48? – but there's a problem with cross-validation!
  - *test set has been used to help set the discretization boundaries – cheating!!!*
- (undo filtering)
- meta>FilteredClassifier: examine "More" info
- set up filter and J48 classifier; run: — 91.2% (27 nodes)
- configure filter to set makeBinary — 92.6% (17 nodes)
- cheat by pre-discretizing using makeBinary — 94.0% (17 nodes)

Lesson 2.2: Supervised discretization and the FilteredClassifier

- Supervised discretization
  - take class into account when making discretization boundaries
- For test set, must use discretization determined by training set
- How can you do this when cross-validating?
- FilteredClassifier: designed for exactly this situation
- Useful with other supervised filters too



Course text

- Section 7.2 *Discretizing numeric attributes*
- Section 11.3 *Filtering algorithms*, subsection “Supervised filters”

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

More Data Mining with Weka

Class 2 – Lesson 3

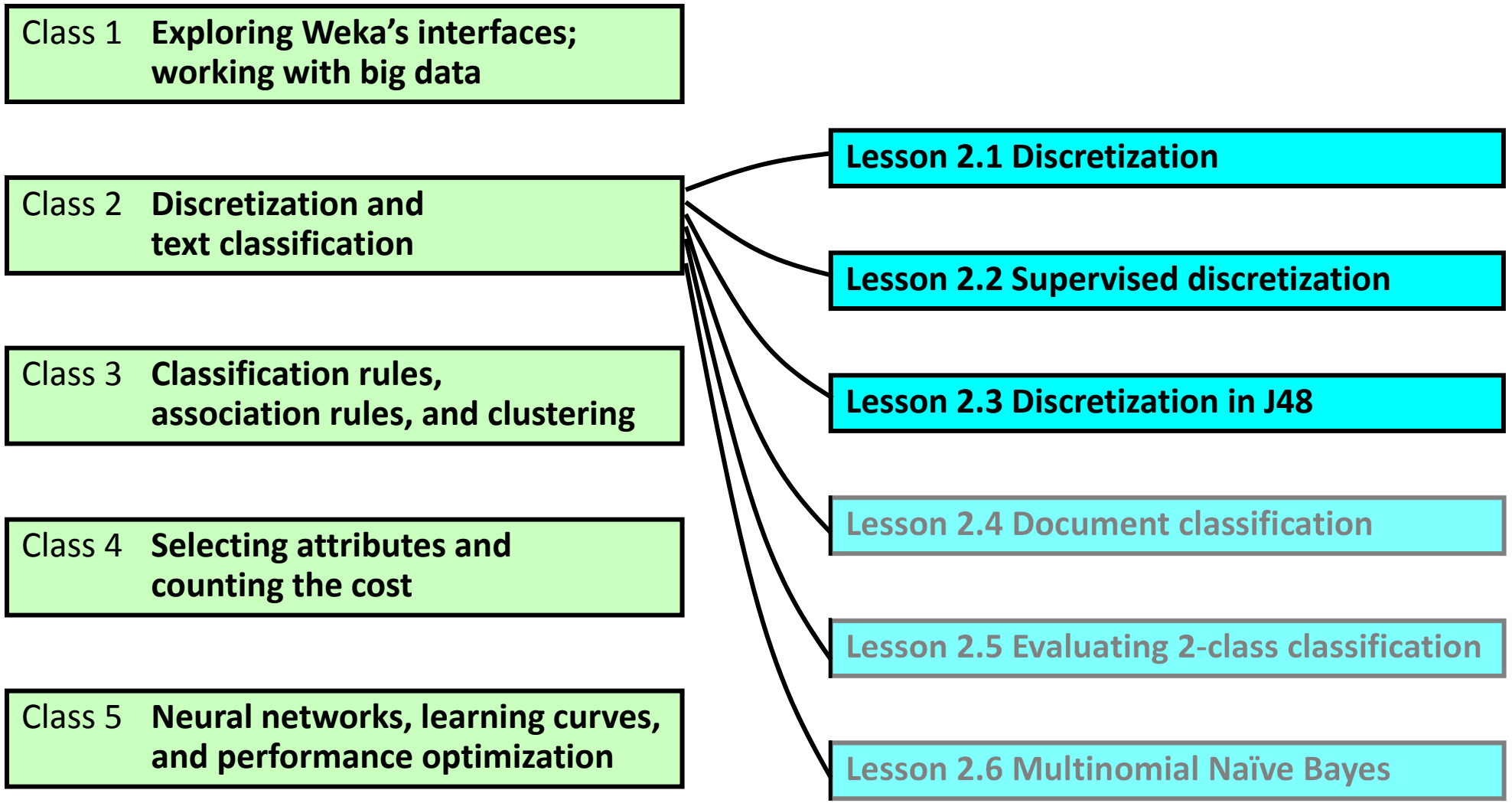
Discretization in J48

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 2.3: Discretization in J48

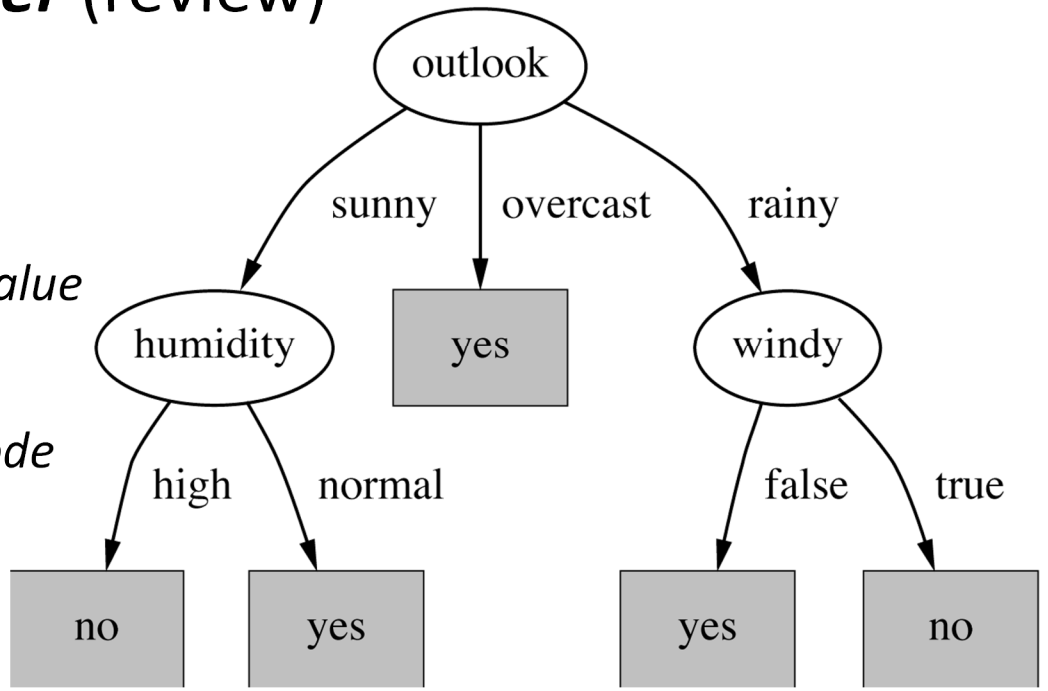


Lesson 2.3: Discretization in J48

How does J48 deal with numeric attributes?

Top-down recursive divide-and-conquer (review)

- **Select** attribute for root node
  - *Create branch for each possible attribute value*
- **Split** instances into subsets
  - *One for each branch extending from the node*
- **Repeat** recursively for each branch
  - *using only instances that reach the branch*



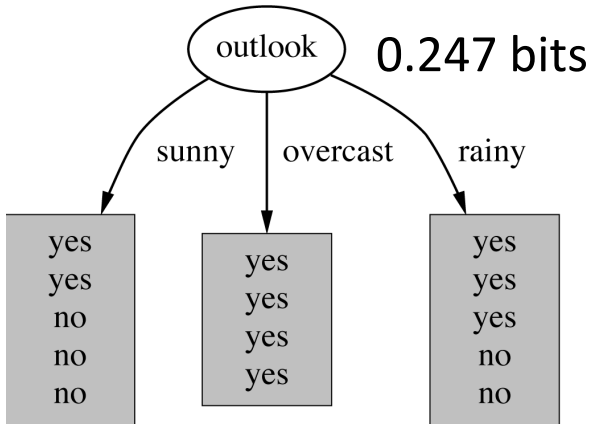
Lesson 2.3: Discretization in J48

Q: Which is the best attribute to split on?

A (J48)**: The one with the greatest "information gain"**

Information gain

- Amount of information gained by knowing the value of the attribute
- (Entropy of distribution before the split) – (entropy of distribution after it)
- $\text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$



Lesson 2.3: Discretization in J48

Information gain for the *temperature* attribute

- Split-point is a number … and there are infinitely many numbers!
- Split mid-way between adjacent values in the training set
- *n*–1 possibilities (*n* is training set size); try them all!

9 yes, 5 no

entropy before the split = 0.940 bits

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no	yes	no	yes	yes	no
						yes	yes				

4 yes, 1 no 5 yes, 4 no

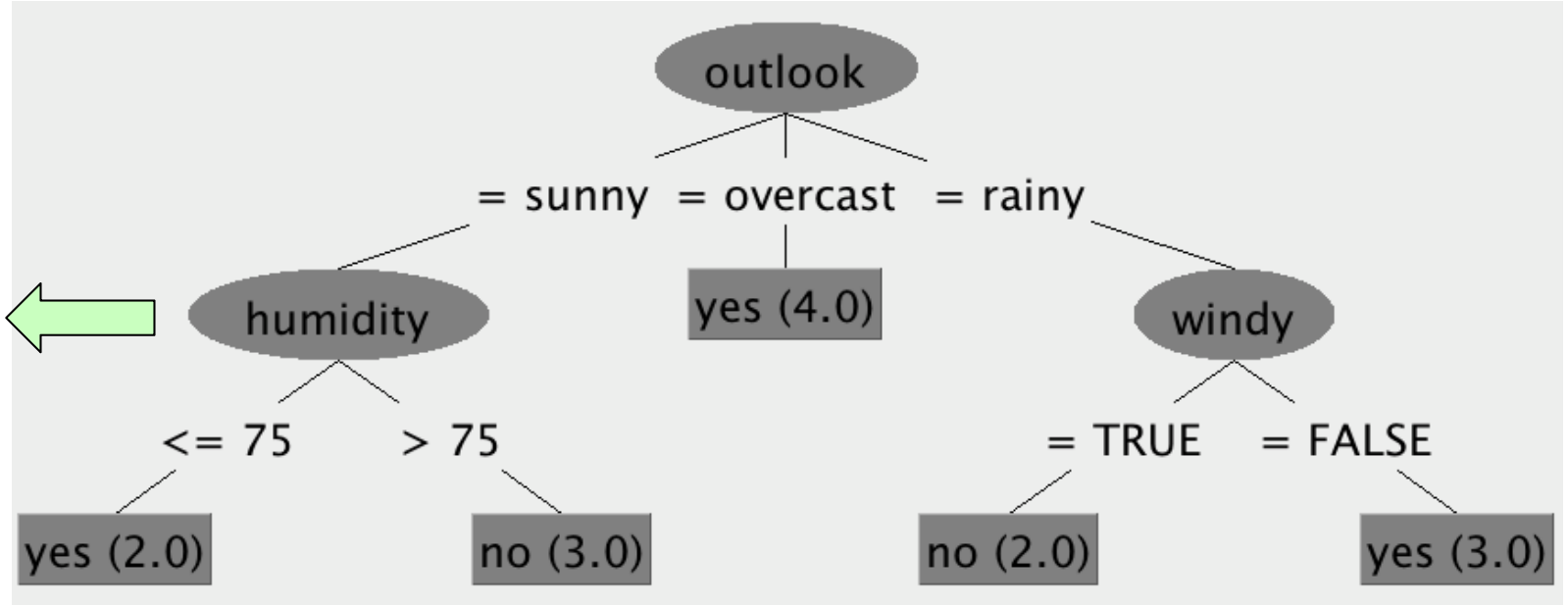
entropy after the split = 0.939 bits

information gain = 0.001 bits

Lesson 2.3: Discretization in J48

Further down the tree, split on *humidity*

Outlook	Temp	Humidity	Wind	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Sunny	72	95	False	No
Sunny	69	70	False	Yes
Sunny	75	70	True	Yes



humidity separates *no*'s from *yes*'s

split halfway between {70,70} and {85}, i.e. 75 (!)

Lesson 2.3: Discretization in J48

Discretization when building a tree vs. in advance

- Discretization boundaries are determined in a more specific context
- But based on a small subset of the overall information

  … particularly lower down the tree, near the leaves

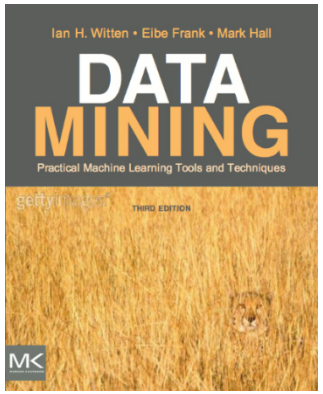
- For every internal node, the instances that reach it must be sorted separately for every numeric attribute

  … and sorting has complexity O(n log n)

  … but repeated sorting can be avoided with a better data structure

Lesson 2.3: Discretization in J48

- C4.5/J48 incorporated discretization early on
- Pre-discretization is an alternative, developed/refined later
  - Supervised discretization uses essentially the same entropy heuristic
  - Can retain the ordering information that numeric attributes imply
- Will J48 internal discretization outperform pre-discretization?
  - arguments both for and against
- An experimental question – you will answer it in the activity!
  - and for other classifiers too



Course text

- Section 6.1 *Decision trees*

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

More Data Mining with Weka

Class 2 – Lesson 4

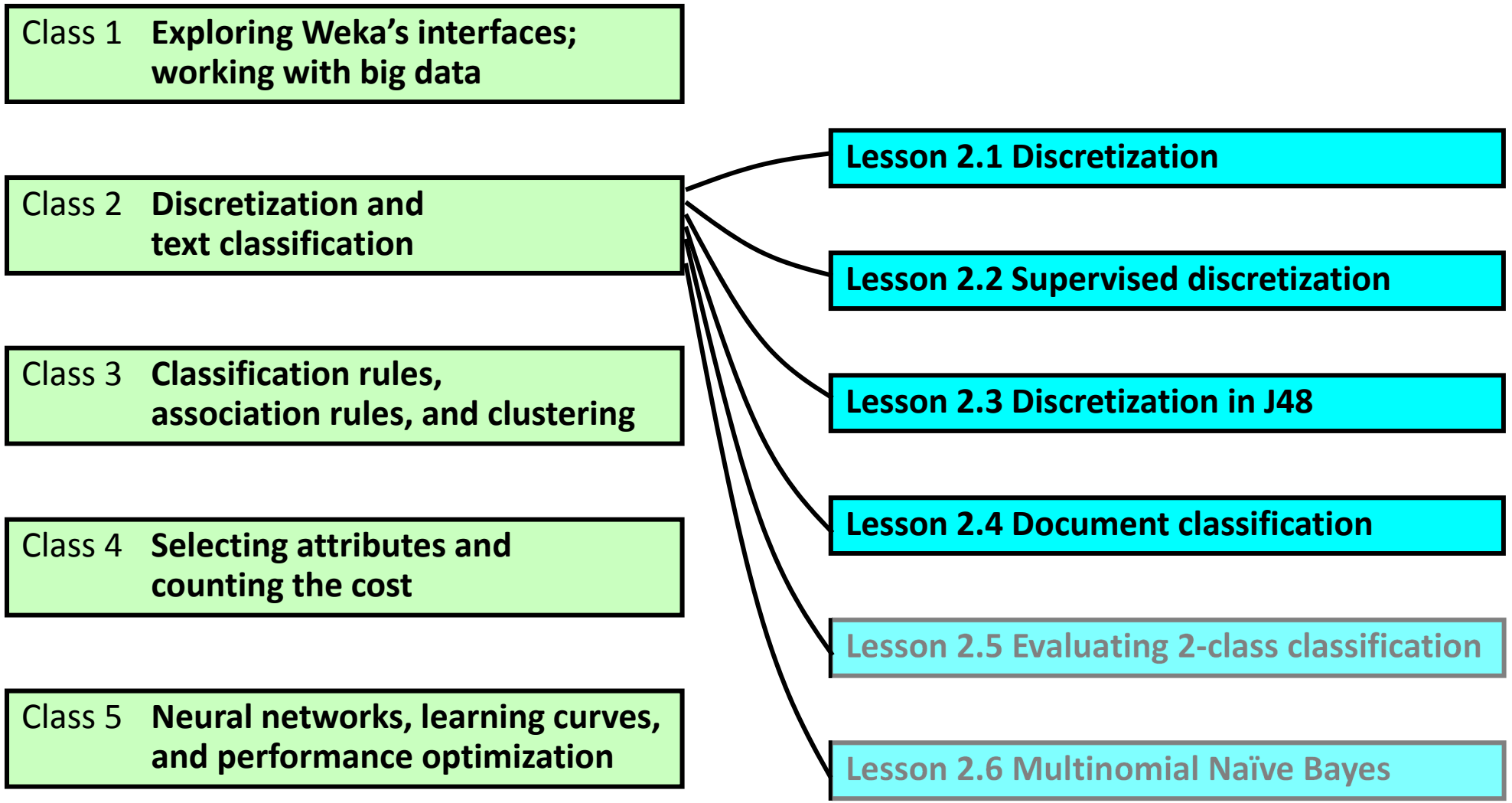
Document classification

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 2.4: Document classification



Lesson 2.4: Document classification

Some training data

Document text	Classification
The price of crude oil has increased significantly	yes
Demand of crude oil outstrips supply	yes
Some people do not like the flavor of olive oil	no
The food was very oily	no
Crude oil is in short supply	yes
Use a bit of cooking oil in the frying pan	no

```
@relation 'training text'

@attribute text string
@attribute type {yes, no}

@data
'The price of crude oil has increased significantly', yes
'Demand of crude oil outstrips supply', yes
'Some people do not like the flavor of olive oil', no
```

Lesson 2.4: Document classification

- Load into Weka; note “string” attributes
- Apply StringToWordVector (unsupervised attribute filter)
- Creates 33 new attributes
  - *Crude, Demand, The, crude, has, in, increases, is, of, oil, …*
- Binary, numeric
- Use J48 (must set the class attribute)
- Evaluate on training set
- Visualize the tree

Lesson 2.4: Document classification

- Supplied test set
  - *set "Output predictions"*
- Problem evaluating classifier
- Apply StringToWordVector to test file?
  - *still get "Problem evaluating classifier"*
- Solution: FilteredClassifier
  - *StringToWordVector creates attributes from training set*
  - *FilteredClassifier uses same attributes for test set*
- Result:
  - *document 1 is "yes"; Documents 2, 3, 4 are "no"*
  - *(though document 3 should be "yes")*

Some test data

Document text	Classification
Oil platforms extract crude oil	Unknown
Canola oil is supposed to be healthy	Unknown
Iraq has significant oil reserves	Unknown
There are different types of cooking oil	Unknown

Lesson 2.4: Document classification

- Take a look at the dataset: ReutersCorn-train.arff
  - *it's big: 1554 instances, 2 attributes*
- Apply StringToWordVector
  - *it's huge: 1554 instances, 2234 attributes (!)*
- Test set: ReutersCorn-test.arff
- FilteredClassifier with StringToWordVector and J48
  - *(takes a while)*
- 97% classification accuracy
- Look at model
- Look at confusion matrix:
  - *classification accuracy on 24 corn-related documents: 15/24 = 62%*
  - *on remaining 580 documents: 573/580 = 99%*
- Is the overall classification accuracy really the right thing to optimize?

Lesson 2.4: Document classification

- String attributes
- StringToWordVector filter: creates many attributes
- Check options for StringToWordVector
- J48 models for text data
- Overall classification accuracy
  - *is it really what we care about?*
  - *perhaps not*

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

More Data Mining with Weka

Class 2 – Lesson 5

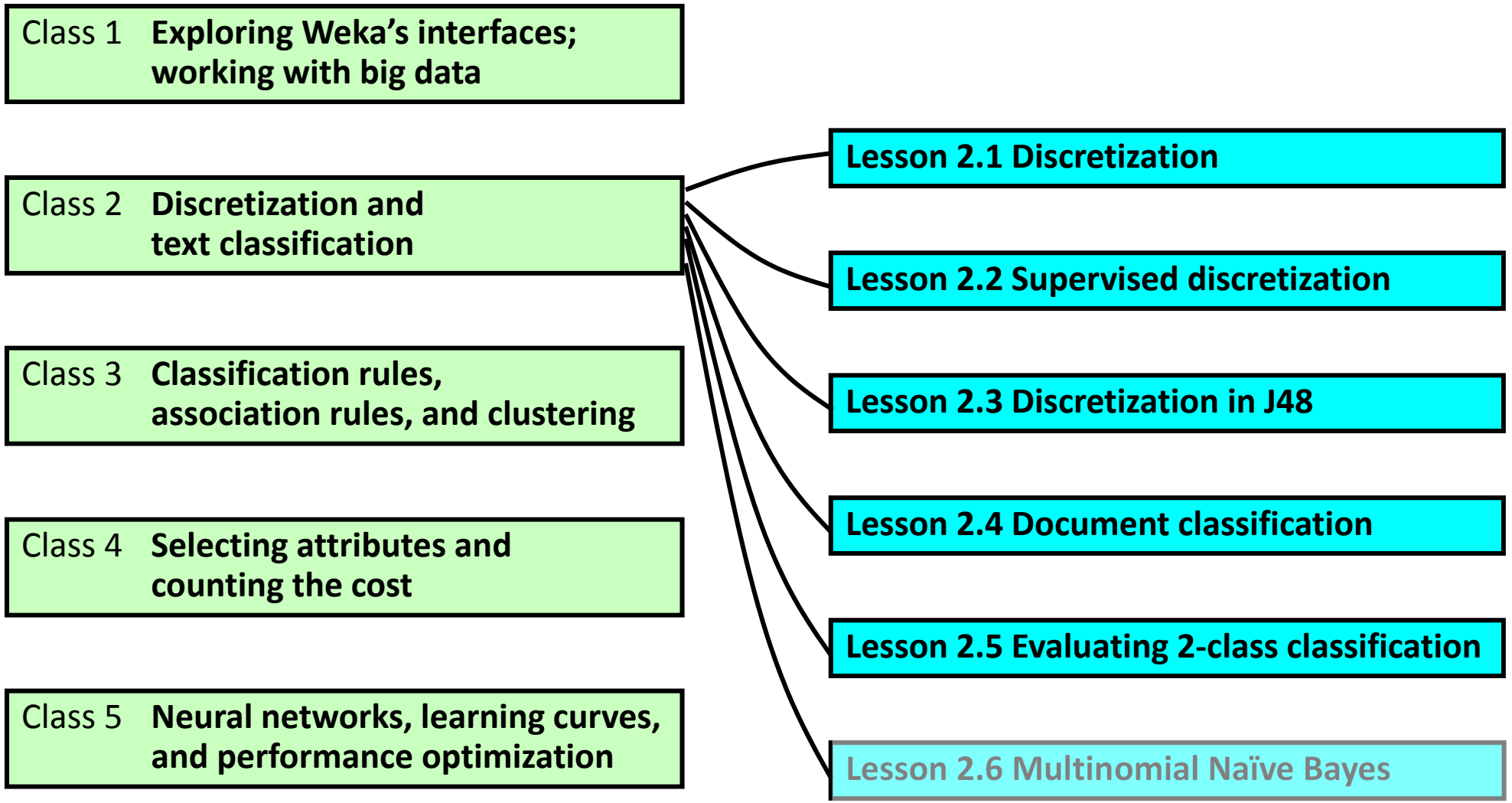
Evaluating 2-class classification

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 2.5: Evaluating 2-class classification



Lesson 2.5: Evaluating 2-class classification

Weather data; Naïve Bayes; 10-fold cross-validation

```
===Confusion Matrix ===

 a b   <-- classified as

 7 2 | a = yes
 4 1 | b = no
```

true positives

false negatives

false positives

true negatives

(negative instances that are incorrectly assigned to the positive class)

TP rate: TP / (TP + FN) = 7/9 = 0.78

accuracy on class a

FP rate: FP / (FP + TN) = 4/5 = 0.80

1 – *accuracy on class* b

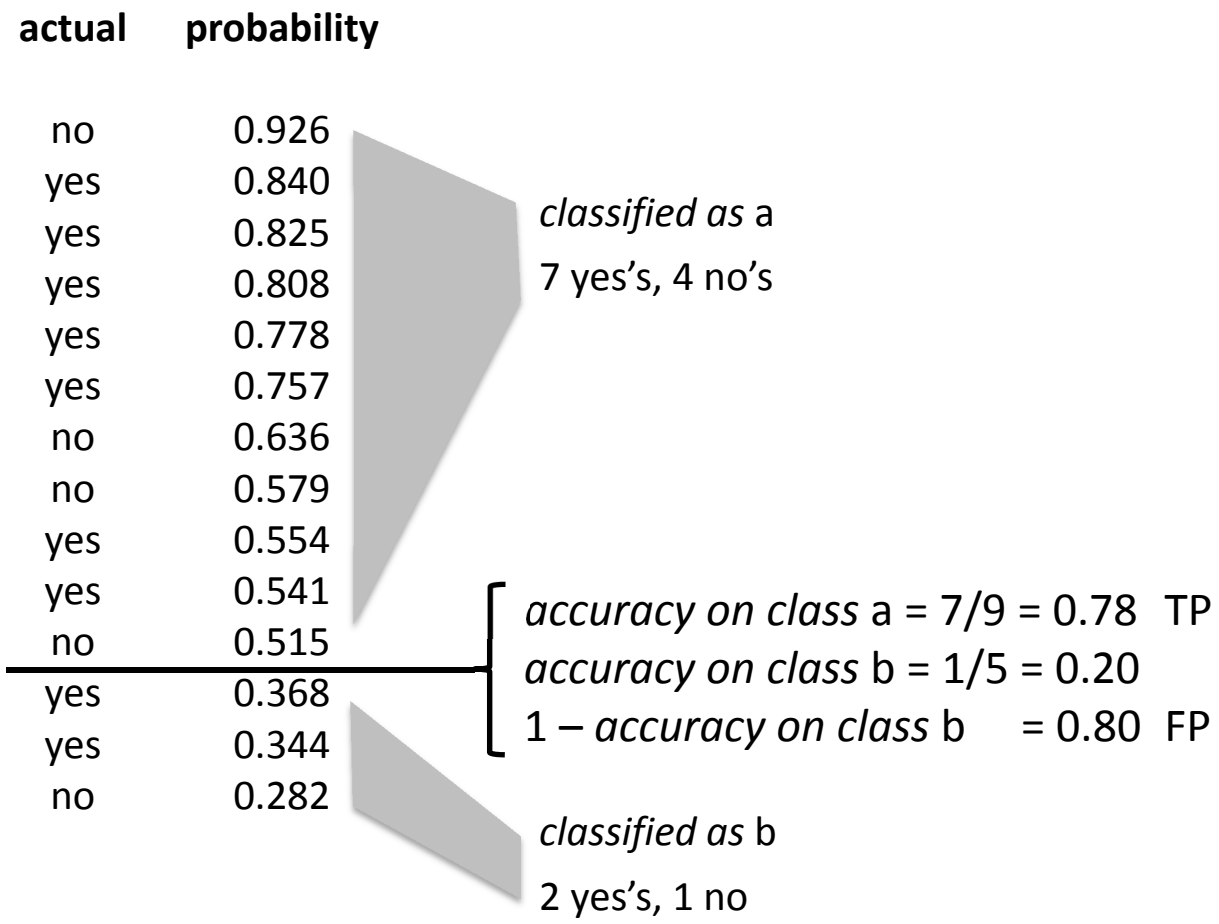
(taking “yes” as the positive class)

Lesson 2.5: Evaluating 2-class classification

Different probability thresholds

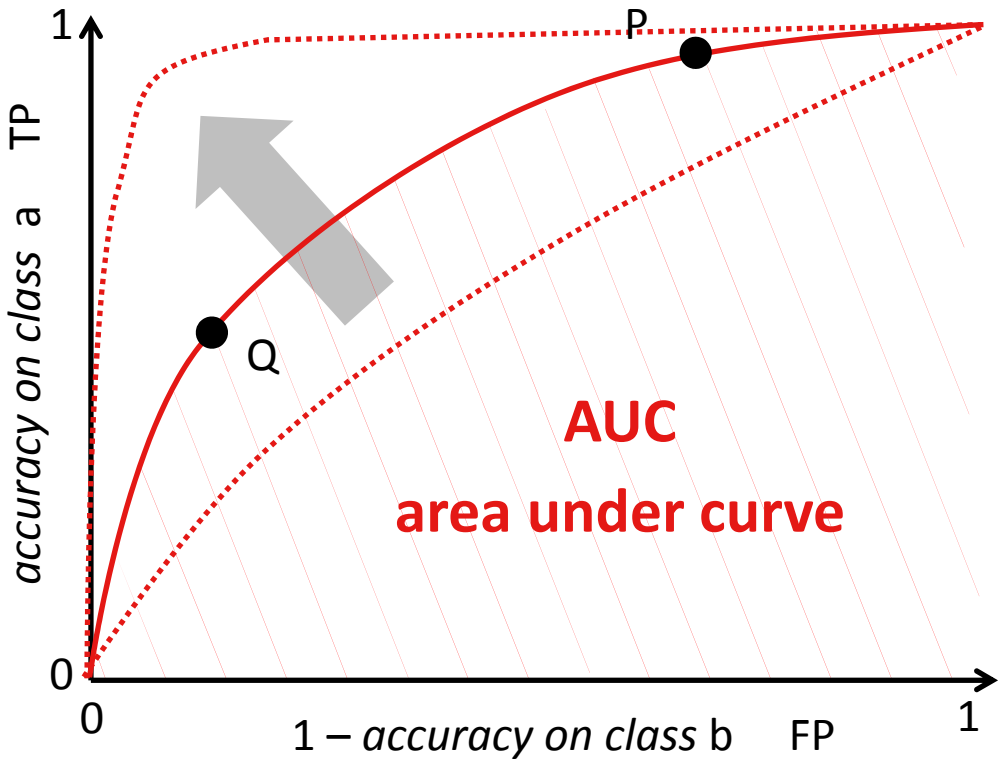
```
=== Predictions on test data ===

inst#,     actual, predicted, error, probability distribution
     1       2:no       1:yes       +   *0.926   0.074
     2      1:yes       1:yes           *0.825   0.175
     1       2:no       1:yes       +   *0.636   0.364
     2      1:yes       1:yes           *0.808   0.192
     1       2:no        2:no            0.282  *0.718
     2      1:yes        2:no       +    0.344  *0.656
     1       2:no       1:yes       +   *0.579   0.421
     2      1:yes       1:yes           *0.541   0.459
     1       2:no       1:yes       +   *0.515   0.485
     1      1:yes        2:no       +    0.368  *0.632
     1      1:yes       1:yes           *0.84    0.16
     1      1:yes       1:yes           *0.554   0.446
     1      1:yes       1:yes           *0.757   0.243
     1      1:yes       1:yes           *0.778   0.222
```

```
 a b   <-- classified as
 7 2 | a = yes
 4 1 | b = no
```



Lesson 2.5: Evaluating 2-class classification

Different probability thresholds

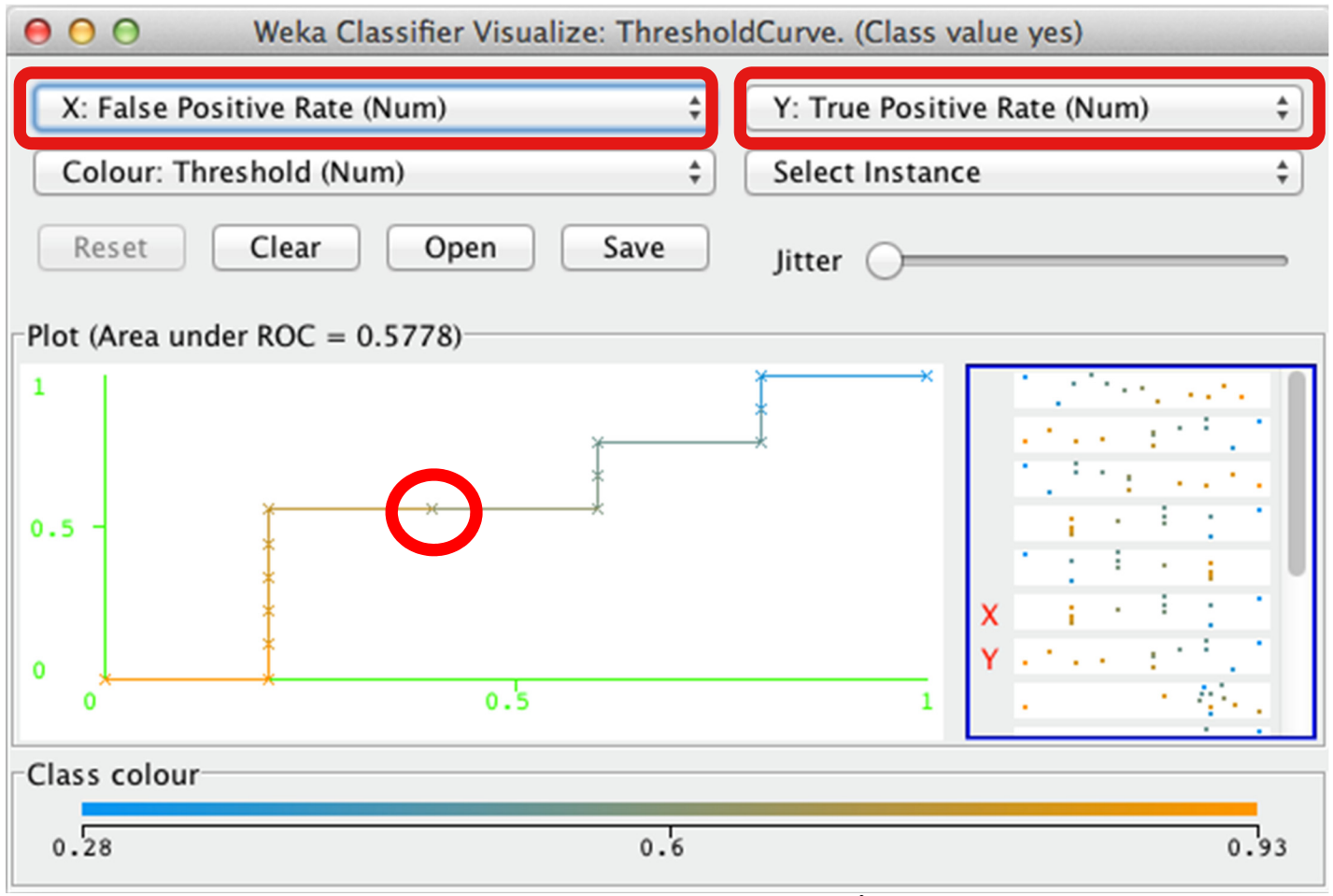


	actual	probability	
	no	0.926	
	yes	0.840	
	yes	0.825	
	yes	0.808	
	yes	0.778	
Q	yes	0.757	*accuracy on class* a = 5/9 = 0.56 TP *accuracy on class* b = 4/5 = 0.80 1 – *accuracy on class* b = 0.20 FP
	no	0.636	
	no	0.579	
	yes	0.554	
	yes	0.541	
P	no	0.515	*accuracy on class* a = 7/9 = 0.78 TP *accuracy on class* b = 1/5 = 0.20 1 – *accuracy on class* b = 0.80 FP
	yes	0.368	
	yes	0.344	
	no	0.282	

… different tradeoffs between accuracy on class a and accuracy on class b

Lesson 2.5: Evaluating 2-class classification

The "ROC" curve (*Receiver Operating Characteristic: historical name*)



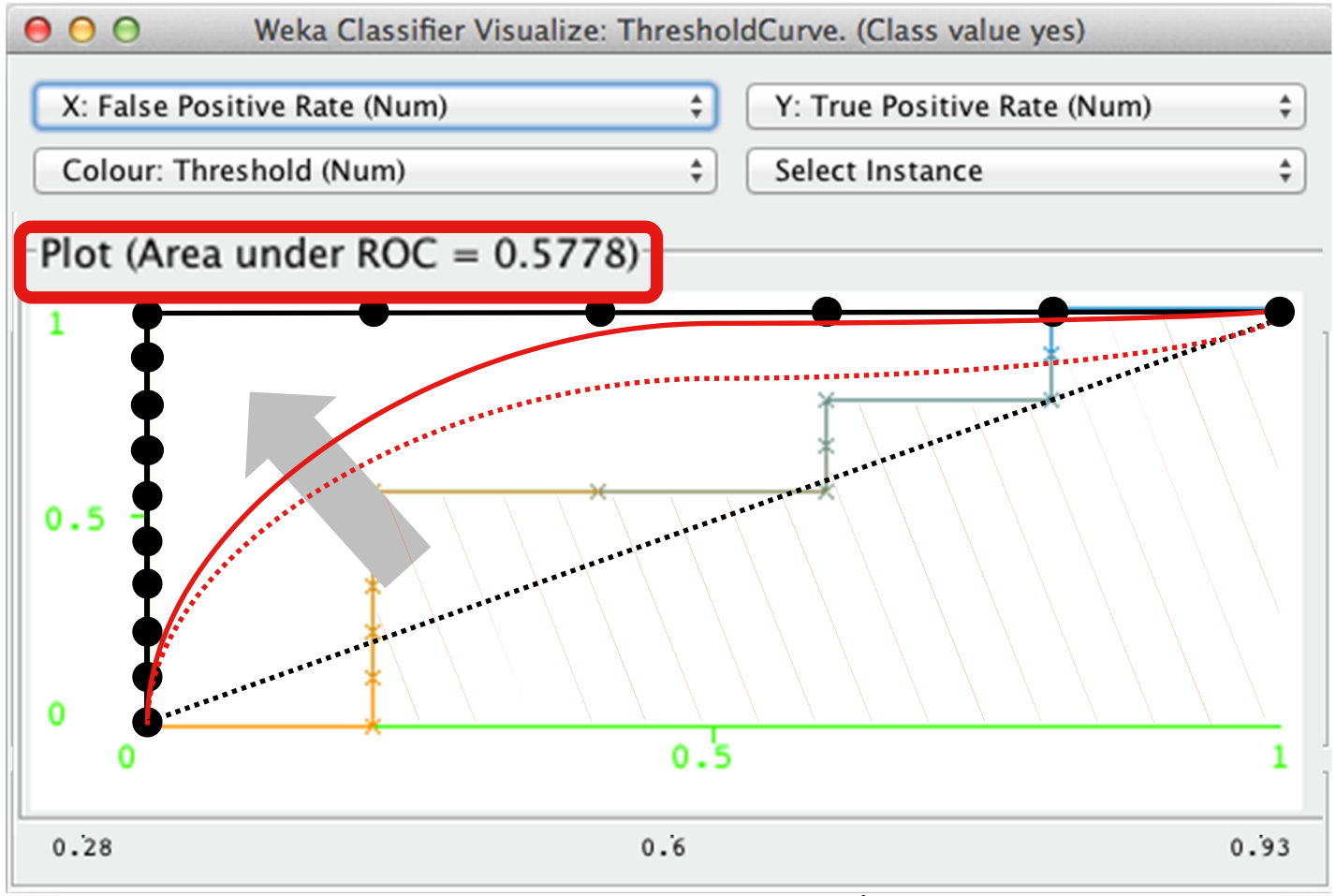
accuracy on class a

1 – *accuracy on class b*

actual	probability	1 – *accuracy on class* b (FP rate)	*accuracy on class* a (TP rate)
		0/5	0/9
no	0.926	1/5	0/9
yes	0.840	1/5	1/9
yes	0.825	1/5	2/9
yes	0.808	1/5	3/9
yes	0.778	1/5	4/9
yes	0.757	1/5	5/9
no	0.636	2/5	5/9
no	0.579	3/5	5/9
yes	0.554	3/5	6/9
yes	0.541	3/5	7/9
no	0.515	4/5	7/9
yes	0.368	4/5	8/9
yes	0.344	4/5	9/9
no	0.282	5/5	9/9

Lesson 2.5: Evaluating 2-class classification

Idealized "ROC" curves

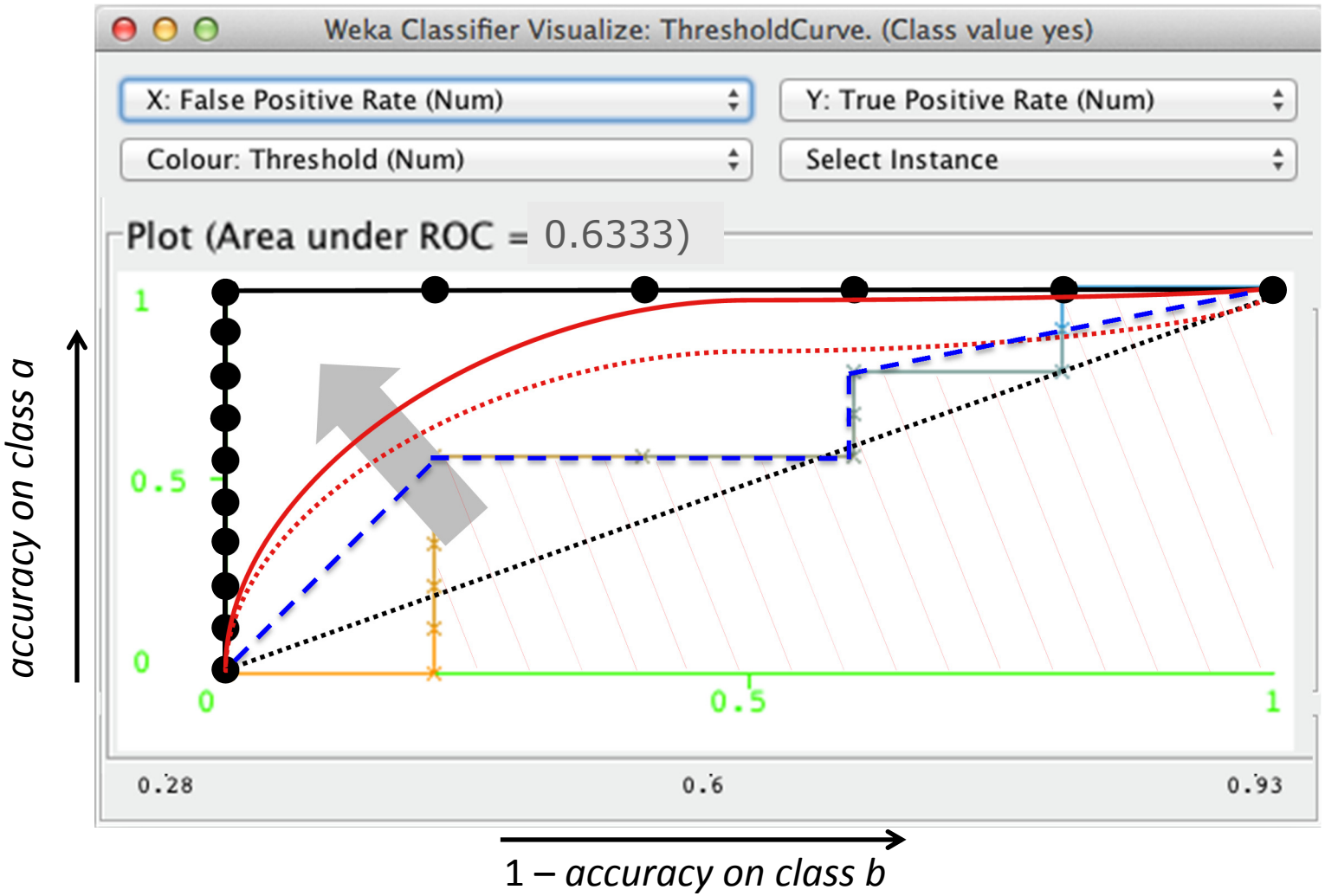


accuracy on class a

1 – *accuracy on class b*

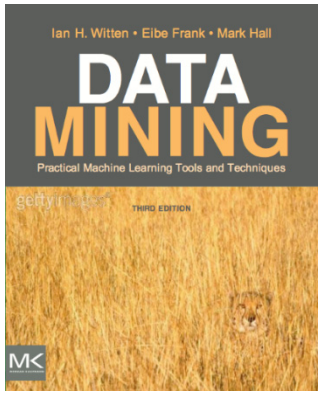
Lesson 2.5: Evaluating 2-class classification

- - - - ROC curve for J48: Area under ROC = 0.6333



Lesson 2.5: Evaluating 2-class classification

- "Per-class accuracy" threshold curves
  - points correspond to different tradeoffs between error types
- ROC curves: TP rate (*y* axis) against FP rate (*x* axis)
  - go from lower left to upper right
  - good ones stretch up towards the top left corner
  - a diagonal line corresponds to a random decision
- AUC (area under the [ROC] curve) – measures overall quality
  - probability that the classifier ranks a randomly chosen +ve test instance above a randomly chosen –ve one



Course text

- Section 5.2 *Counting the cost*, subsection "ROC curves"

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

More Data Mining with Weka

Class 2 – Lesson 6

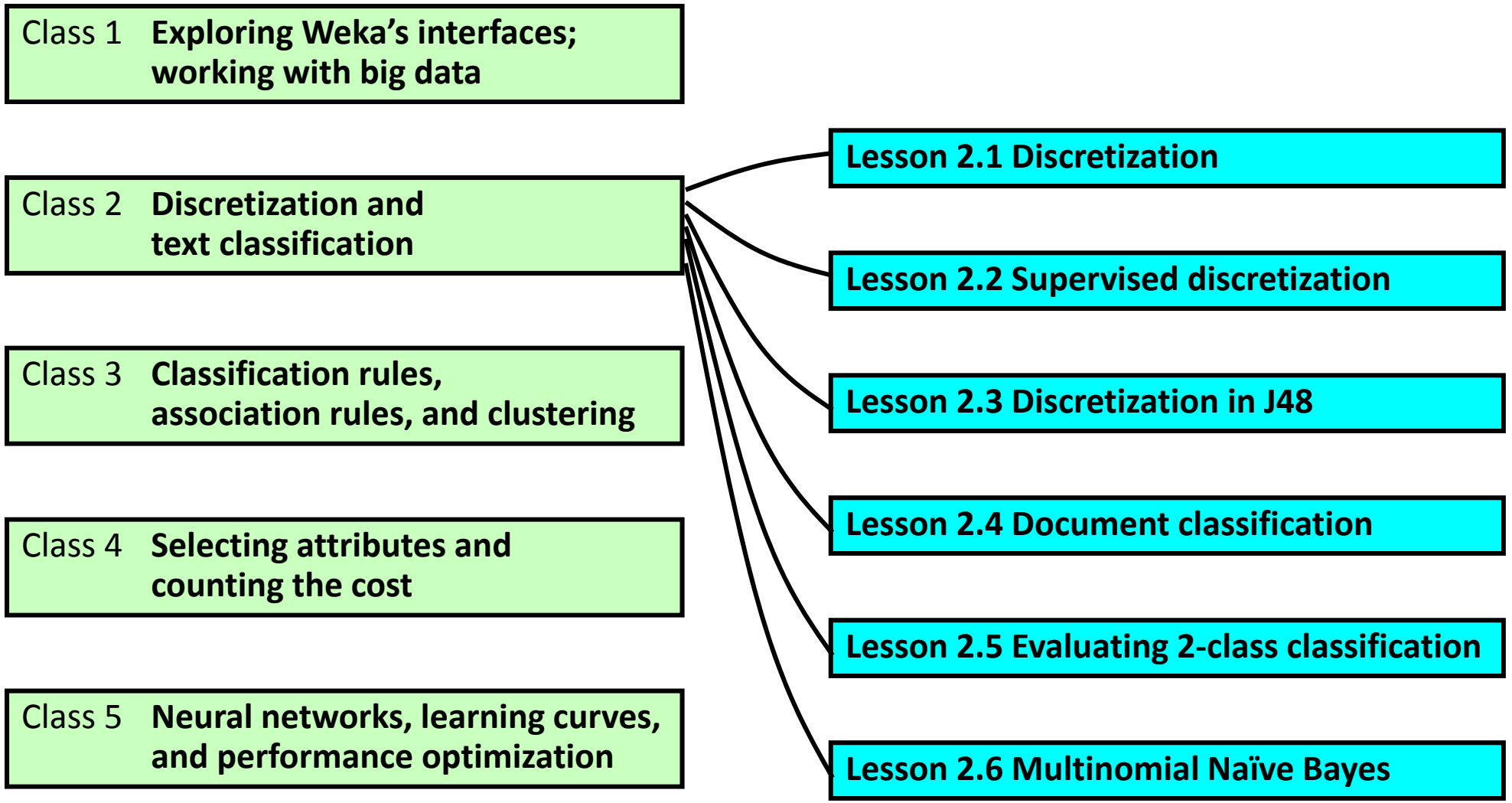
Multinomial Naïve Bayes

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

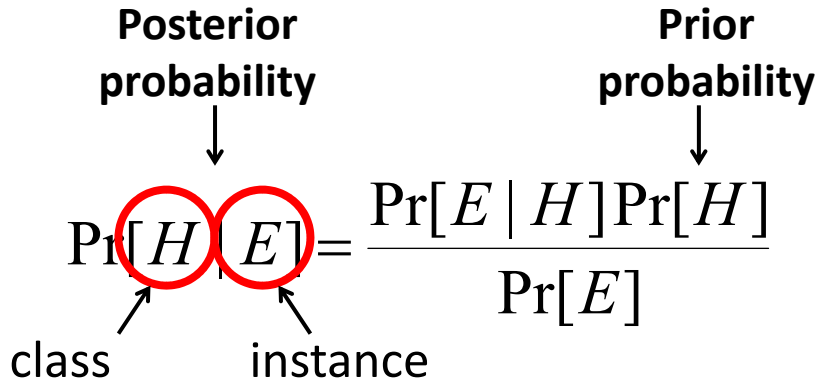
Lesson 2.6: Multinomial Naïve Bayes



Lesson 2.6: Multinomial Naïve Bayes

Remember Naïve Bayes?

- Probability of event *H* given evidence *E*



- Evidence splits into independent parts

$$\Pr[E \mid H] = \Pr[E_1 \mid H]\Pr[E_2 \mid H]...\Pr[E_n \mid H]$$

Document classification: E_i is appearance of word *i*

- But
  - *non-appearance of a word counts just as strongly as appearance*
  - *does not account for multiple repetitions of a word*
  - *treats all words (common ones, unusual ones, …) the same*

Lesson 2.6: Multinomial Naïve Bayes

Multinomial Naïve Bayes

(for the curious)

$$\Pr[E \mid H] = \Pr[E_1 \mid H]\Pr[E_2 \mid H]\ldots\Pr[E_n \mid H]$$

$$= N! \times \prod_{i=1}^{k} \frac{p_i^{n_i}}{n_i!}$$

- p_i is probability of word *i* over all documents in class *H*
- n_i is number of times it appears in this document
- $N = n_1+n_2+\ldots+n_k$ is number of words in this document

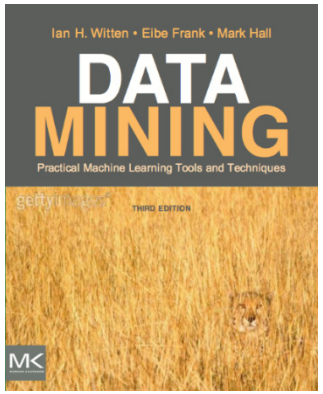
(the factorials "!" are a technicality to account for different word orderings)

Lesson 2.6: Multinomial Naïve Bayes

- Training set: ReutersGrain-train.arff; test set: ReutersGrain-test.arff
- Classifier: FilteredClassifier with StringToWordVector
- J48 gets 96% classification accuracy
  - *38/57 on corn-related documents, 544/547 on others; ROC Area = 0.906*
- NaiveBayes: 80% classification accuracy
  - *46/57 on corn-related documents, 439/547 on others; ROC Area = 0.885*
- NaiveBayesMultinomial: 91% classification accuracy
  - *52/57 on corn-related documents, 496/547 on others; ROC Area = 0.973*
- Set outputWordCounts in StringToWordVector
  NaiveBayesMultinomial: 91% classification accuracy
  - *54/57 on corn-related documents, 496/547 on others; ROC Area = 0.962*
- Set lowerCaseTokens, useStoplist in StringToWordVector
  NaiveBayesMultinomial: 93% classification accuracy
  - *56/57 on corn-related documents, 504/547 on others; ROC Area = 0.978*

Lesson 2.6: Multinomial Naïve Bayes

- Multinomial Naïve Bayes is designed for text
  - based on word appearance only, not non-appearance
  - can account for multiple repetitions of a word
  - treats common words differently from unusual ones
- It's a lot faster than plain Naïve Bayes!
  - ignores words that do not appear in a document
  - internally, Weka uses a sparse representation of the data
- The StringToWordVector filter has many interesting options
  - although they don't necessarily give the results you're looking for!
  - outputs results in "sparse data" format, which MNB takes advantage of



Course text

- Section 4.2 *Statistical modeling*, under "Naïve Bayes for document classification"

More Data Mining with Weka

Department of Computer Science
University of Waikato
New Zealand

Creative Commons Attribution 3.0 Unported License

creativecommons.org/licenses/by/3.0/

weka.waikato.ac.nz