Data Mining with Weka

Class 4 – Lesson 1

Classification boundaries

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 4.1 Classification boundaries

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

Class 5
Putting it all together

Lesson 4.1 Classification boundaries

Lesson 4.2 Linear regression

Lesson 4.3 Classification by regression

Lesson 4.4 Logistic regression

Lesson 4.5 Support vector machines

Lesson 4.6 Ensemble learning

Lesson 4.1 Classification boundaries

Weka's Boundary Visualizer for OneR

- Open **iris.2D.arff**, a 2D dataset
  - *(could create it yourself by removing sepallength and sepalwidth attributes)*
- Weka GUI Chooser: **Visualization>BoundaryVisualizer**
  - *open iris.2D.arff*
  - *Note: petallength on X, petalwidth on Y*
  - *choose rules>OneR*
  - *check Plot training data*
  - *click Start*
  - *in the Explorer, examine OneR's rule*

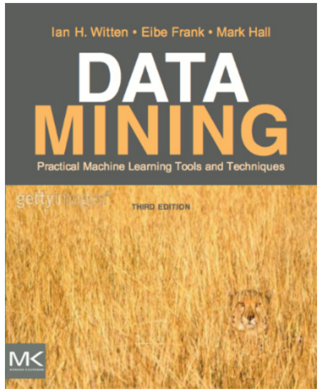
Lesson 4.1 Classification boundaries

Visualize boundaries for other schemes

- Choose **lazy>IBk**
  - *Plot training data; click Start*
  - *k = 5, 20; note mixed colors*
- Choose **bayes>NaiveBayes**
  - *set useSupervisedDiscretization to true*
- Choose **trees>J48**
  - *relate the plot to the Explorer output*
  - *experiment with minNumbObj = 5 and 10: controls leaf size*

Lesson 4.1 Classification boundaries

- Classifiers create boundaries in instance space
- Different classifiers have different biases
- Looked at OneR, IBk, NaiveBayes, J48
- Visualization restricted to numeric attributes, and 2D plots



Course text

- Section 17.3 *Classification boundaries*

Data Mining with Weka

Class 4 – Lesson 2

Linear regression

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 4.2: Linear regression

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

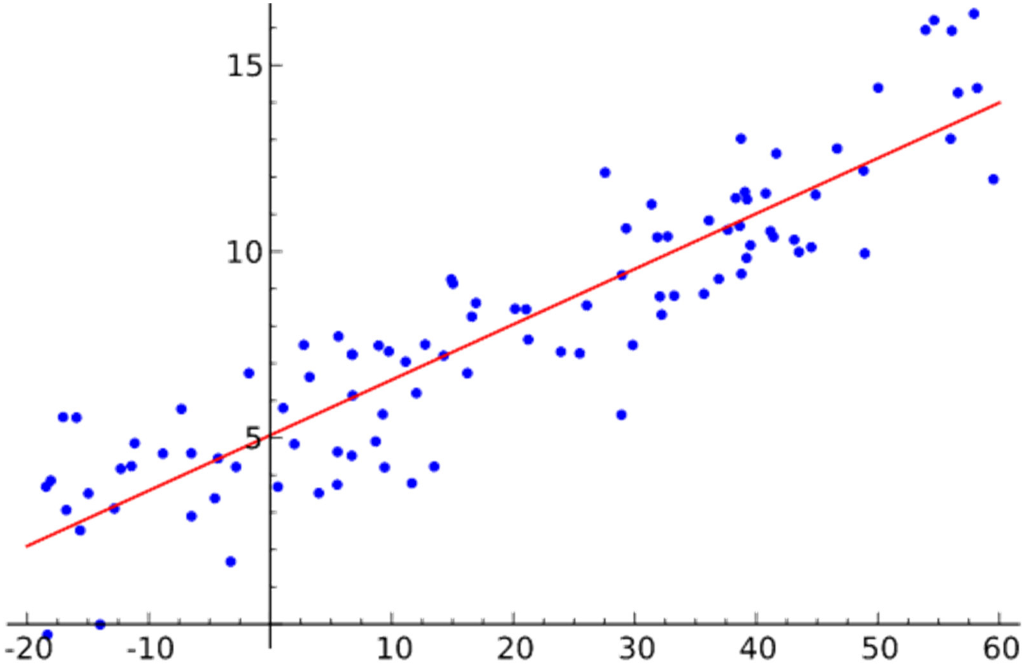
- Lesson 4.1 Classification boundaries
- Lesson 4.2 Linear regression
- Lesson 4.3 Classification by regression
- Lesson 4.4 Logistic regression
- Lesson 4.5 Support vector machines
- Lesson 4.6 Ensemble learning

Class 5
Putting it all together

Lesson 4.2: Linear regression

Numeric prediction (called “regression”)

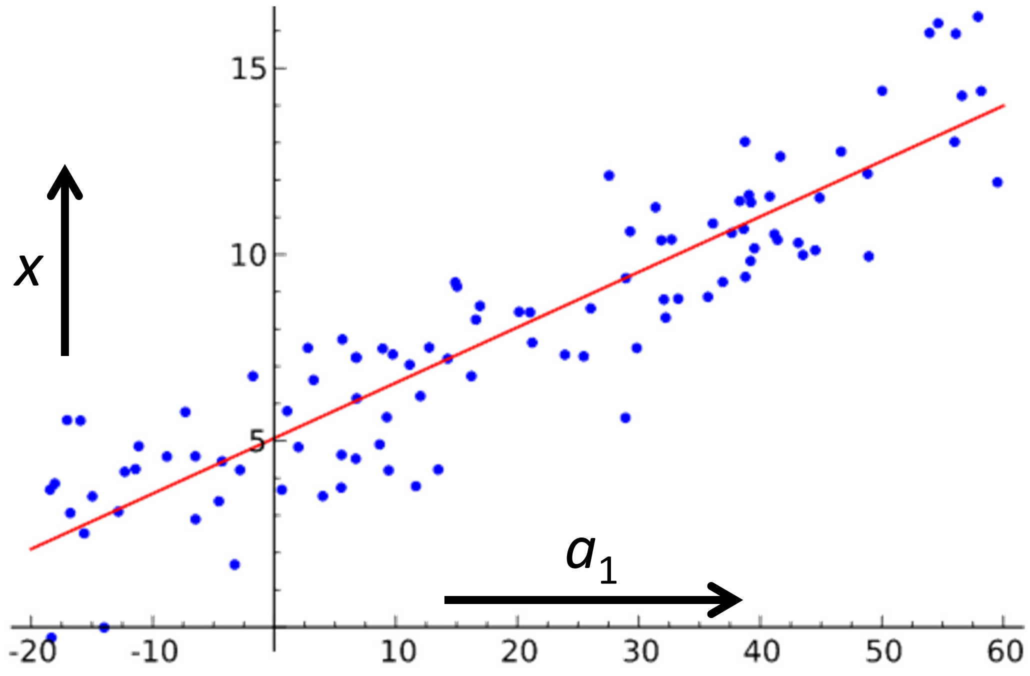
- ❖ Data sets so far: nominal and numeric attributes, but only nominal classes
- ❖ Now: numeric classes
- ❖ Classical statistical method (from 1805!)



Lesson 4.2: Linear regression

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

(Works most naturally
with numeric attributes)



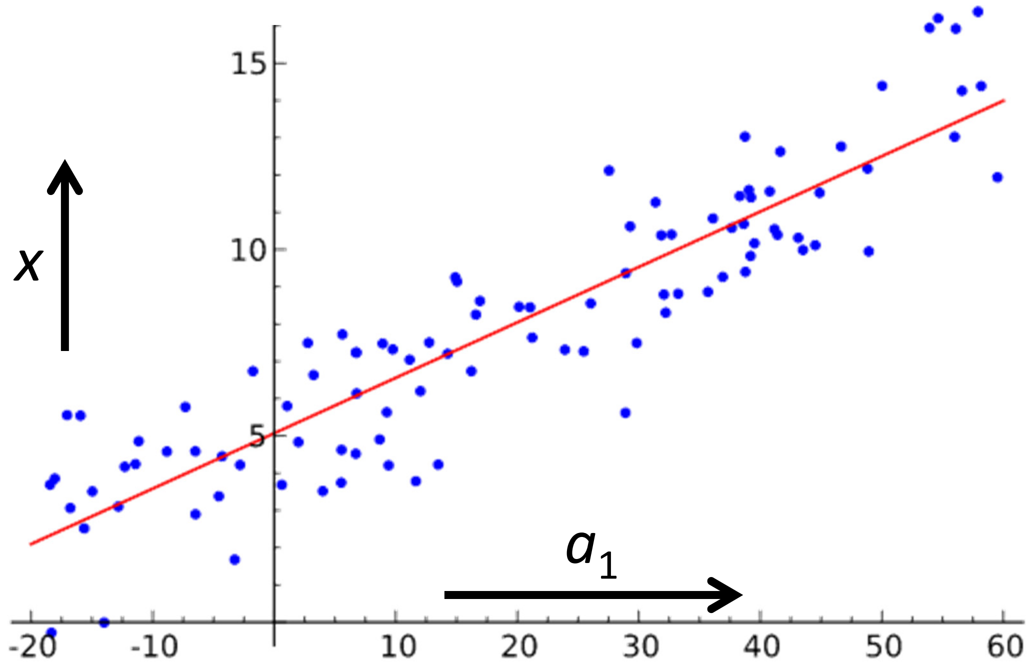
Lesson 4.2: Linear regression

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

- Calculate weights from training data

- Predicted value for first training instance $a^{(1)}$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$



Lesson 4.2: Linear regression

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

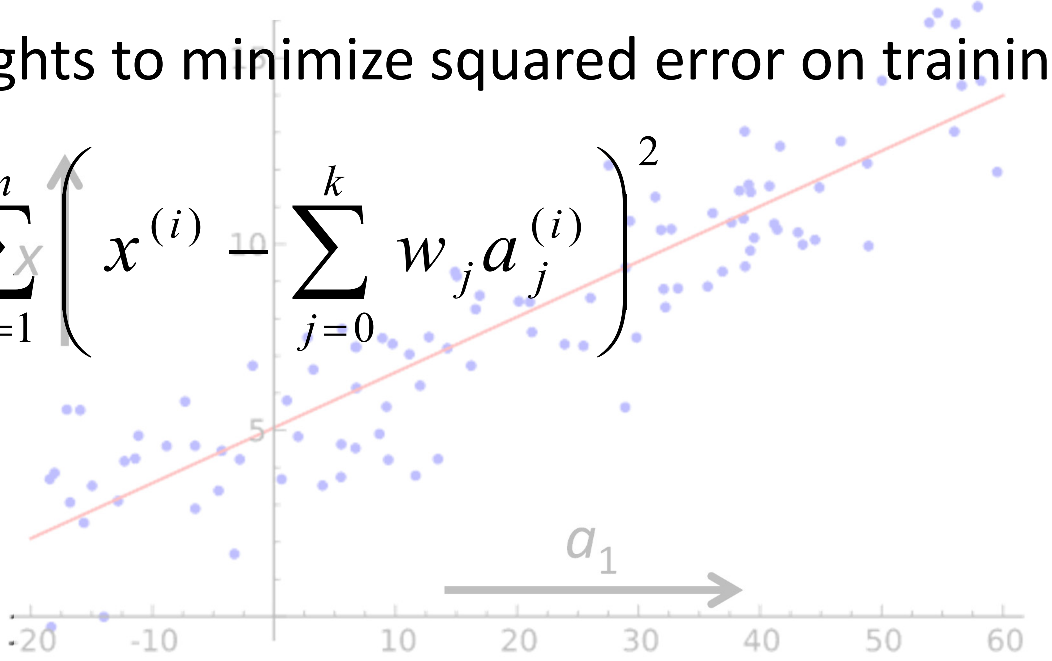
❖ Calculate weights from training data

❖ Predicted value for first training instance $a^{(1)}$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

❖ Choose weights to minimize squared error on training data

$$\sum_{i=1}^{n} \left(x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)} \right)^2$$



Lesson 4.2: Linear regression

- Standard matrix problem
  - *Works if there are more instances than attributes*
    roughly speaking

- Nominal attributes
  - *two-valued: just convert to 0 and 1*
  - *multi-valued … will see in end-of-lesson Activity*

Lesson 4.2: Linear regression

- Open file cpu.arff: all numeric attributes and classes
- Choose functions>LinearRegression
- Run it
- Output:
  - *Correlation coefficient*
  - *Mean absolute error*
  - *Root mean squared error*
  - *Relative absolute error*
  - *Root relative squared error*
- Examine model

$$\frac{|p_1 - a_1| + \ldots + |p_n - a_n|}{n}$$

$$\sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}}$$

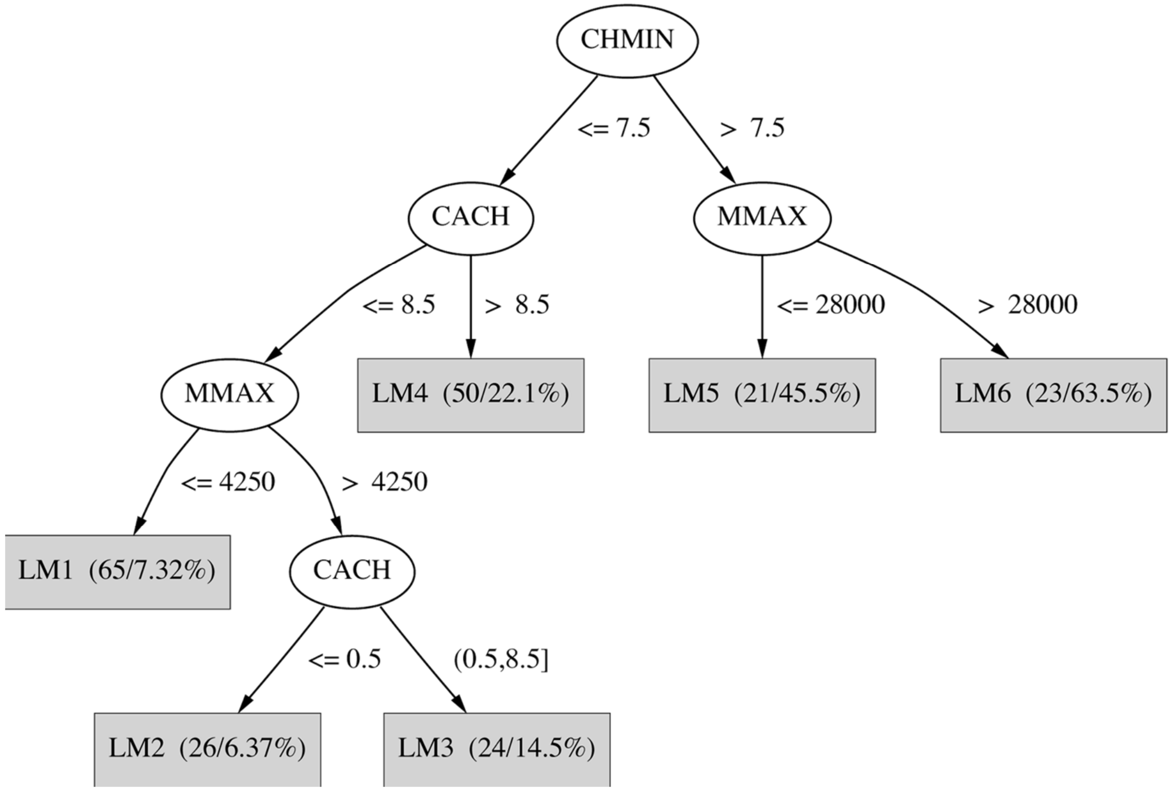
$$\frac{|p_1 - a_1| + \ldots + |p_n - a_n|}{|a_1 - \bar{a}| + \ldots + |a_n - \bar{a}|}$$

$$\sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + \ldots + (a_n - \bar{a})^2}}$$

Lesson 4.2: NON Linear regression

Model tree

- Each leaf has a linear regression model
- Linear patches approximate continuous function



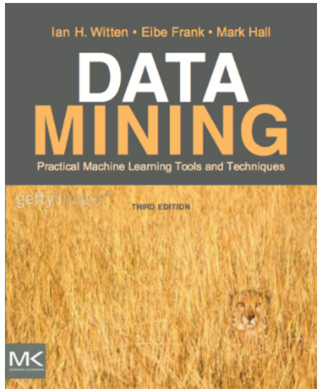
Lesson 4.2: NON Linear regression

- Choose **trees>M5P**
- Run it
- Output:
  - *Examine the linear models*
  - *Visualize the tree*

- Compare performance with the **LinearRegression** result: you do it!

Lesson 4.2: Linear regression

- Well-founded, venerable mathematical technique: **functions>LinearRegression**
- Practical problems often require non-linear solutions
- **trees>M5P** builds trees of regression models



Course text

- Section 4.6 *Numeric prediction: Linear regression*

Data Mining with Weka

Class 4 – Lesson 3

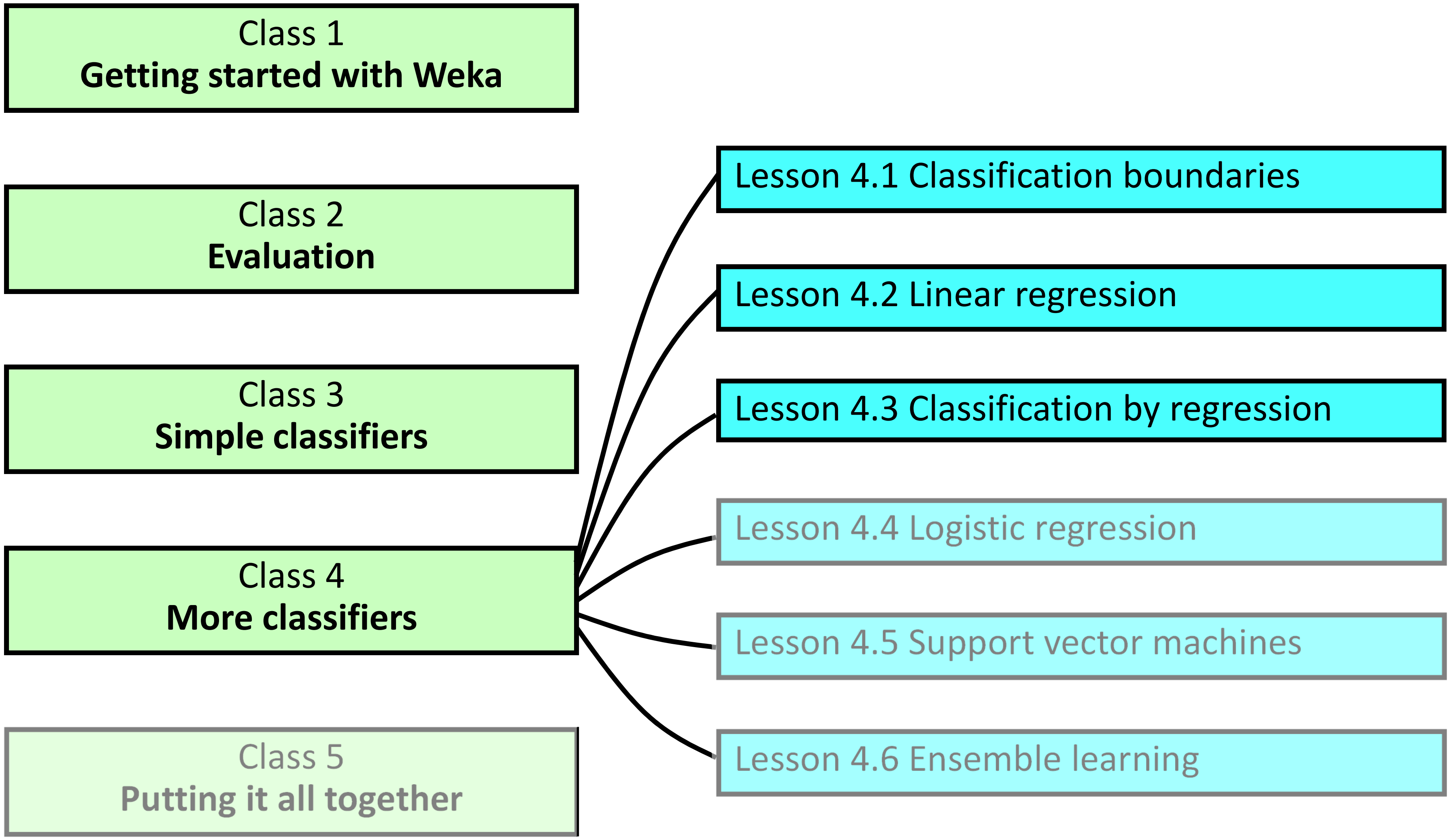
Classification by regression

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 4.3: Classification by regression



Lesson 4.3: Classification by regression

Can a regression scheme be used for classification? Yes!

Two-class problem

- Training: call the classes 0 and 1
- Prediction: set a threshold for predicting class 0 or 1

Multi-class problem: “multi-response linear regression”

- Training: perform a regression for each class
  - *Set output to 1 for training instances that belong to the class,*
    0 for instances that don’t
- Prediction: choose the class with the largest output

… or use “pairwise linear regression”, which performs a regression for every pair of classes

Lesson 4.3: Classification by regression

Investigate two-class classification by regression

- Open file **diabetes.arff**
- Use the **NominalToBinary** attribute filter to convert to numeric
  - *but first set Class: class (Nom) to No class,*
    because attribute filters do not operate on the class value
- Choose **functions>LinearRegression**
- Run
- Set **Output predictions** option

Lesson 4.3: Classification by regression

More extensive investigation

Why are we doing this?

- It's an interesting idea
- Will lead to quite good performance
- Leads in to "Logistic regression" (next lesson), with excellent performance
- Learn some cool techniques with Weka

Strategy

- Add a new attribute ("classification") that gives the regression output
- Use OneR to optimize the split point for the two classes
  (first restore the class back to its original nominal value)

Lesson 4.3: Classification by regression

- Supervised attribute filter AddClassification
  - *choose functions>LinearRegression as classifier*
  - *set outputClassification to true*
  - *Apply; adds new attribute called "classification"*
- Convert class attribute back to nominal
  - *unsupervised attribute filter NumericToNominal*
  - *set attributeIndices to 9*
  - *delete all the other attributes*
- Classify panel
  - *unset Output predictions option*
  - *change prediction from (Num) classification to (Nom) class*
- Select rules>OneR; run it
  - *rule is based on classification attribute, but it's complex*
- Change minBucketSize parameter from 6 to 100
  - *simpler rule (threshold 0.47) that performs quite well: 76.8%*

Lesson 4.3: Classification by regression

- Extend linear regression to classification
  - *Easy with two classes*
  - *Else use multi-response linear regression, or pairwise linear regression*
- Also learned about
  - *Unsupervised attribute filter NominalToBinary, NumericToNominal*
  - *Supervised attribute filter AddClassification*
  - *Setting/unsetting the class*
  - *OneR's minBucketSize parameter*
- But we can do better: Logistic regression
  - *next lesson*

Data Mining with Weka

Class 4 – Lesson 4

Logistic regression

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 4.4: Logistic regression

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

Class 5
Putting it all together

Lesson 4.1 Classification boundaries

Lesson 4.2 Linear regression

Lesson 4.3 Classification by regression

Lesson 4.4 Logistic regression

Lesson 4.5 Support vector machines

Lesson 4.6 Ensemble learning

Lesson 4.4: Logistic regression

Can do better by using prediction probabilities

Probabilities are often useful anyway …

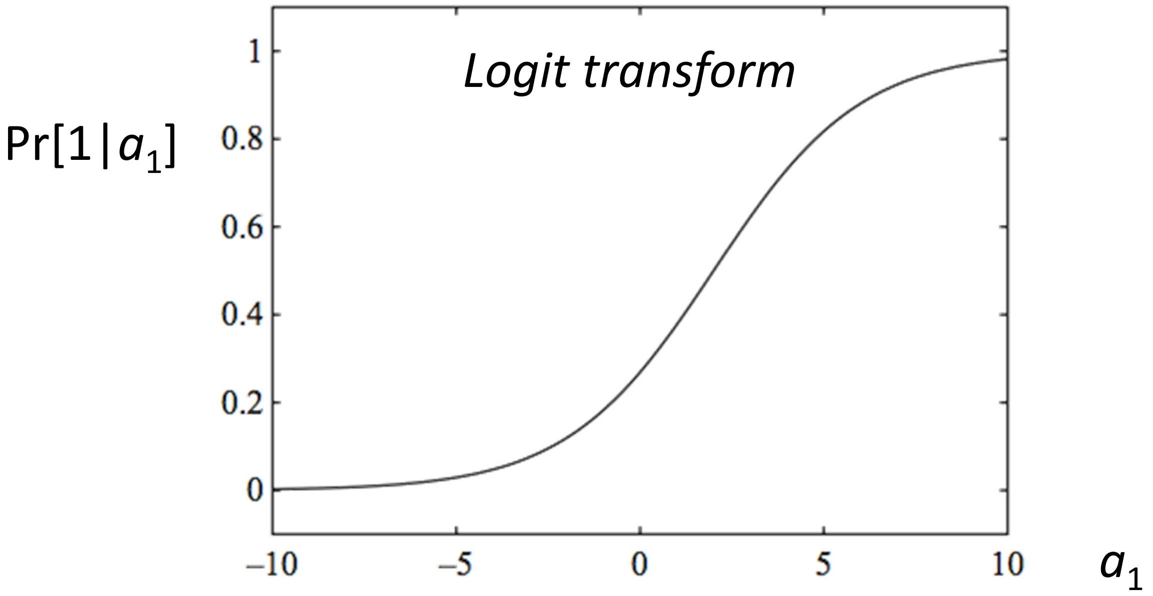
- Naïve Bayes produces them (obviously)
  - *Open **diabetes.arff** and run **Bayes>NaiveBayes*** **with 90% percentage split**
  - *Look at columns: actual, predicted, error, prob distribution*
- Other methods produce them too …
  - *Run **rules>ZeroR**. Why probabilities* [*0.648, 0.352*] for [*tested_negative, tested_positive*]?
  - *90% training fold has 448 negatve, 243 positive instances*
  - *(448+1)/(448+1 + 243+1) = 0.648* [*cf. Laplace correction, Lesson 3.2*]
  - *Run **trees>J48***
  - *J48 uses probabilities internally to help with pruning*

Make linear regression produce probabilities too!

Lesson 4.4: Logistic regression

- Linear regression: calculate a linear function and then a threshold
- Logistic regression: estimate class probabilities directly

$$\Pr[1 \mid a_1, a_2, \ldots, a_k] = 1/(1+\exp(-w_0 - w_1 a_1 - \ldots - w_k a_k))$$



- Choose weights to maximize the log-likelihood (not minimize the squared error):

$$\sum_{i=1}^{n} (1-x^{(i)}) \log(1-\Pr[1 \mid a_1^{(1)}, a_2^{(2)}, \ldots, a_k^{(k)}]) + x^{(i)} \log(\Pr[1 \mid a_1^{(1)}, a_2^{(2)}, \ldots, a_k^{(k)}])$$

Lesson 4.4: Logistic regression

- Open file **diabetes.arff**
- Classification-by-regression 76.8% mean of 10 runns

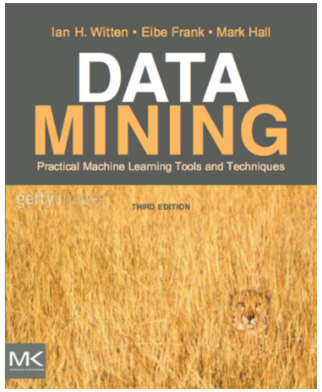
cf	ZeroR	65.1%	65.1%
	Naïve Bayes	76.3%	75.8%
	J48	73.8%	74.5%

- **Apply functions>Logistic** 77.2% 77.5%

- Extension to multiple classes …
  - Perform a regression for each class?
    (like multi-response regression)
  - No. Probabilities won't sum to 1
  - Can be tackled as a joint optimization problem

Lesson 4.4: Logistic regression

- Logistic regression is popular and powerful
- Uses logit transform to predict probabilities directly
  - like Naïve Bayes
- Also learned about
  - Prediction probabilities from other methods
  - How to calculate probabilities from ZeroR



Course text

- Section 4.6 *Numeric prediction: Logistic regression*

Data Mining with Weka

Class 4 – Lesson 5

Support vector machines

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 4.5 Support vector machines

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

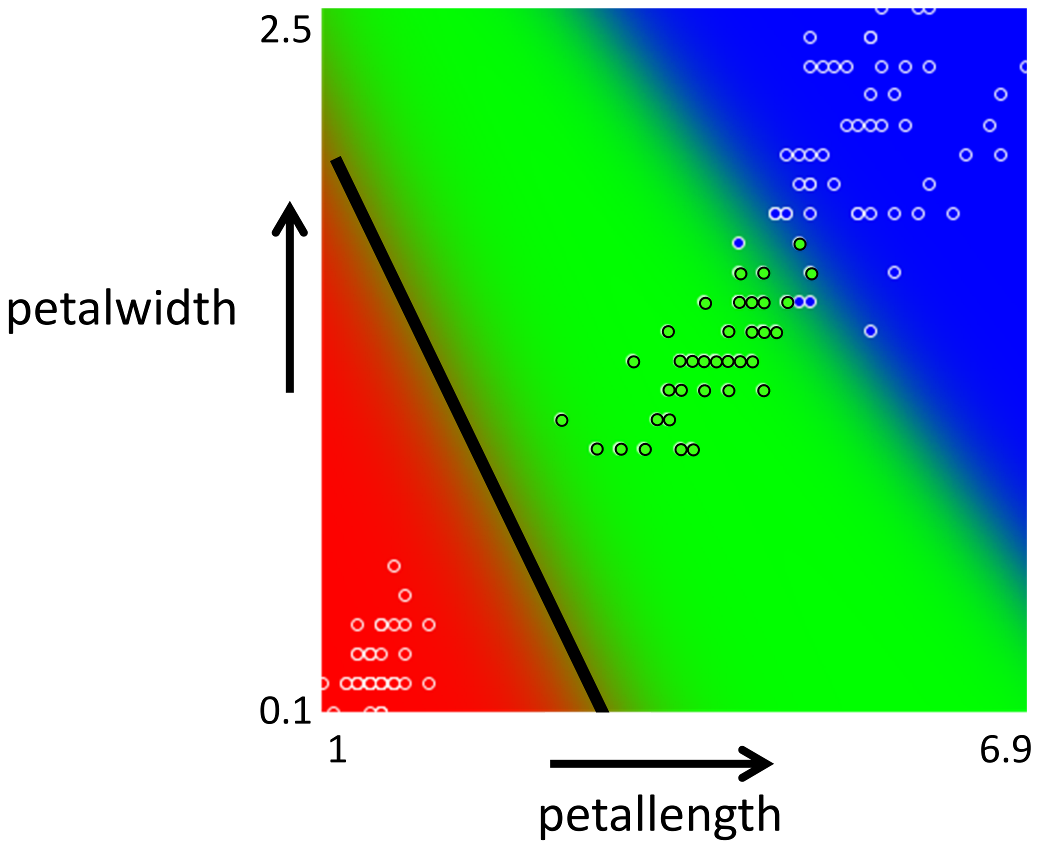
- Lesson 4.1 Classification boundaries
- Lesson 4.2 Linear regression
- Lesson 4.3 Classification by regression
- Lesson 4.4 Logistic regression
- Lesson 4.5 Support vector machines
- Lesson 4.6 Ensemble learning

Class 5
Putting it all together

Lesson 4.5 Support vector machines

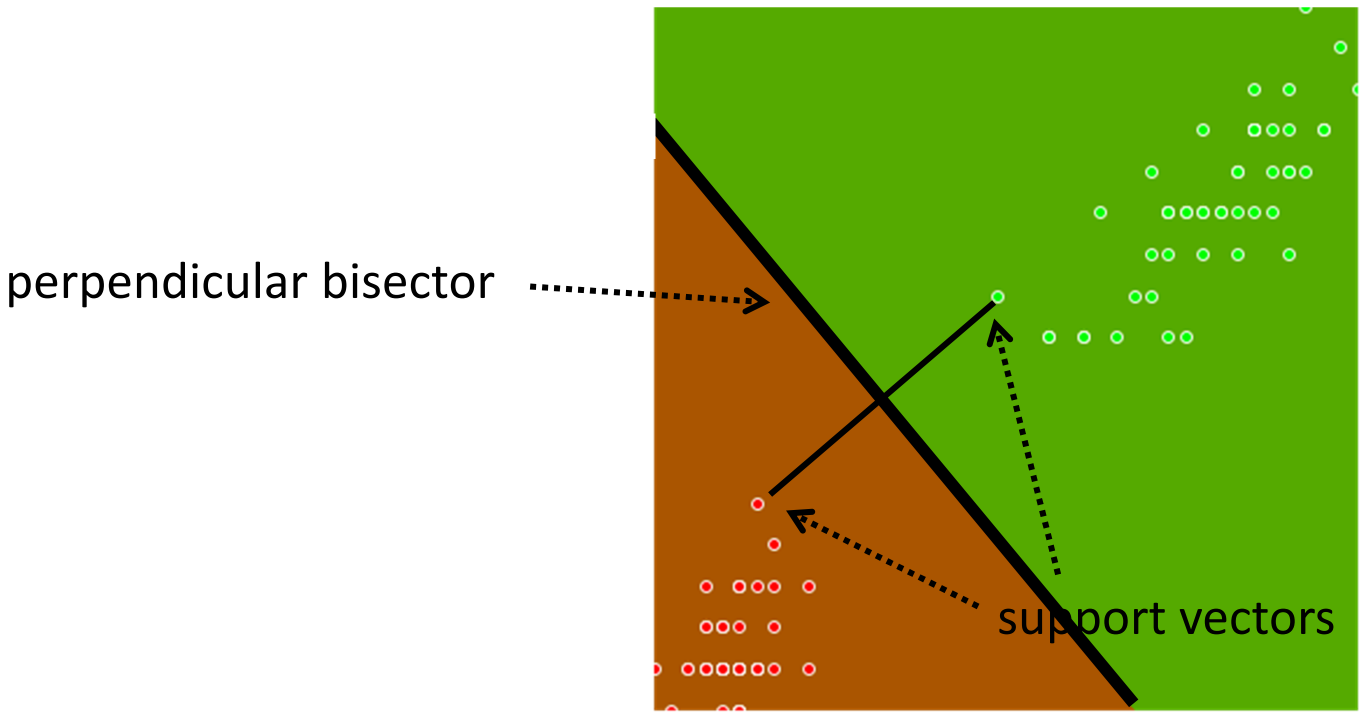
Logistic regression ⇒ linear boundaries

- ❖ Weka's boundary visualizer
  - *from the activity following Lesson 3.6*



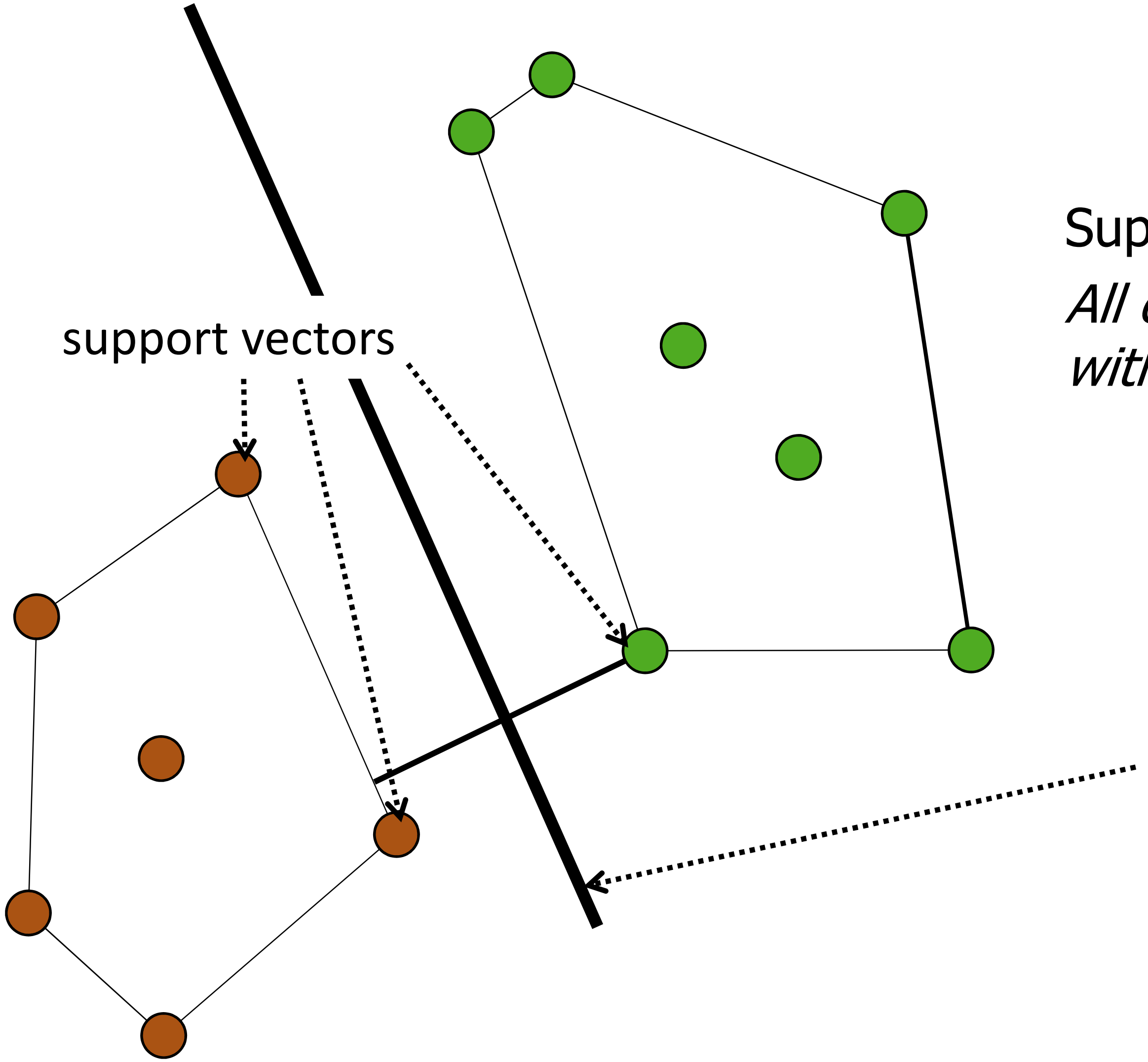
Lesson 4.5 Support vector machines

Support vector geometry



Lesson 4.5 Support vector machines

Maximum margin hyperplane



Support vectors define the boundary

All other instances can be deleted without changing it!

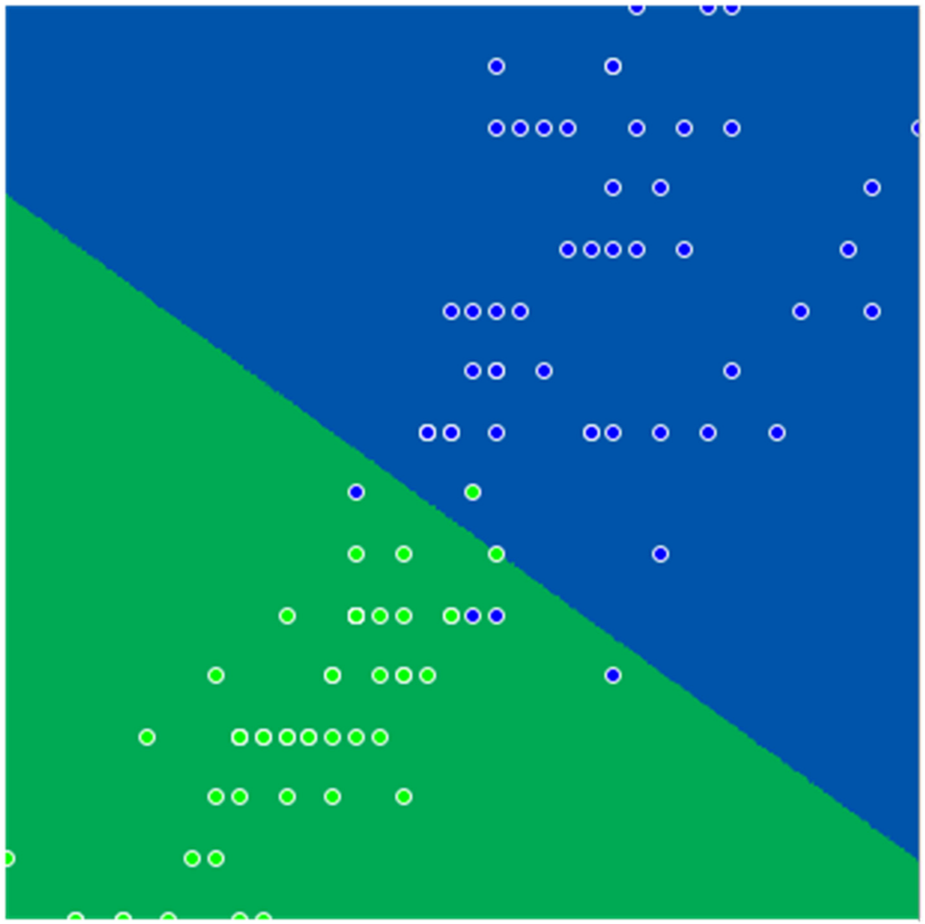
$$x = b + \sum \alpha_i y_i \mathbf{a(i)} \bullet \mathbf{a}$$

sum over support vectors **a(i)**

Lesson 4.5 Support vector machines

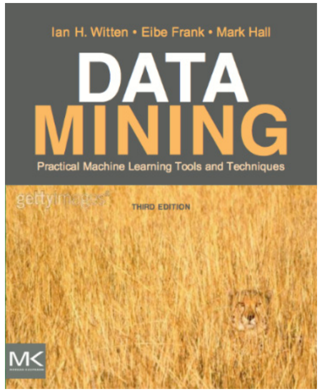
Classes that are not linearly separable

(more complex)



Lesson 4.5 Support vector machines

- Linear decision boundary
  - but can get more complex boundaries with the "Kernel trick"
- Very resilient to overfitting
  - boundary depends on a very few points
- Weka: functions>SMO
  - restricted to two classes
  - so use Multiresponse linear regression … or Pairwise linear regression
- Weka: functions>LibSVM
  - External library for support vector machines
  - faster than SMO, more sophisticated options



Course text

- Section 6.4 *Maximum-margin hyperplane*

Data Mining with Weka

Class 4 – Lesson 6

Ensemble learning

Ian H. Witten

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz

Lesson 4.6: Ensemble learning

Class 1
Getting started with Weka

Class 2
Evaluation

Class 3
Simple classifiers

Class 4
More classifiers

Class 5
Putting it all together

Lesson 4.1 Classification boundaries

Lesson 4.2 Linear regression

Lesson 4.3 Classification by regression

Lesson 4.4 Logistic regression

Lesson 4.5 Support vector machines

Lesson 4.6 Ensemble learning

Lesson 4.6: Ensemble learning

Committee structure: build different "experts," let them vote

- Often improves predictive performance
- Produces output that is hard to analyze
  - *but: there are approaches that aim to produce a single comprehensible structure*
- Methods
  - *Bagging*
  - *Randomization*
  - *Boosting*
  - *Stacking*

Lesson 4.6: Ensemble learning

Bagging

- Several training sets of the same size
  - *produce them by sampling … with replacement*
- Build model for each one
  - *use same machine learning scheme*
- Combine predictions by voting

  (or, for regression, averaging)
- Very suitable for “unstable” learning schemes
  - *small change in training data can make big change in model*
  - *example: decision trees … but not Naïve Bayes or instance-based learning*
- Weka: meta>Bagging
- E.g. with glass.arff
  - *J48* *66.8%*
  - *Bagging (default parameters)* *72.4%*

Lesson 4.6: Ensemble learning

Randomization: random forests

- Randomize the algorithm, not the training data
  - *how you randomize depends on the algorithm*
- Random forests
  - *attribute selection for J48 decision tree: don't pick the best, pick randomly from the k best options*
  - *generally improves decision trees*
- Weka: trees>RandomForests
  - *options: number of trees (default 10); maximum depth of trees; number of attributes*
- E.g. with glass.arff
  - *J48* 66.8%
  - *RandomForests (default parameters)* 75.2%

Lesson 4.6: Ensemble learning

Boosting

- Iterative: new models are influenced by performance of previously built ones
  - *extra weight for instances that are misclassified ("hard" ones)*
  - *encourage new model to become an "expert" for instances misclassified by earlier models*
  - *Intuitive justification: committee members should complement each other's expertise*
- Uses voting (or, for regression, averaging)
  - *but weights models according to their performance*
- Often dramatically improves performance
- Weka: meta>AdaBoostM1
- E.g. with glass.arff
  - *J48* *66.8%*
  - *AdaBoostM1 (using J48)* *74.3%*

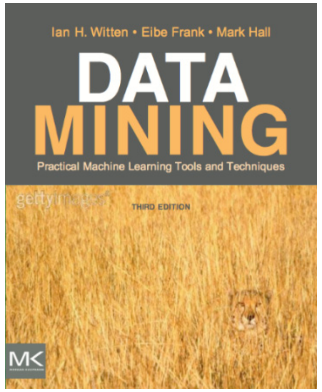
Lesson 4.6: Ensemble learning

Stacking

- Combine predictions of base learners using a *meta learner* (not voting)
  - *base learners: level-0 models*
  - *meta learner: level-1 model*
  - *predictions of base learners are input to meta learner*
- Base learners are usually different schemes
- Can't use predictions on training data to generate data for level-1 model!
  - *Instead use cross-validation-like scheme*
- Weka: meta>Stacking
  - *and StackingC, more efficient version*
  - *allow multiple level-0 models (by specifying a metaclassifier)*
- Quite hard to make stacking work well, but with glass.arff I got
  - *J48* *66.8%*
  - *StackingC, with default metaclassifier and base classifiers IBk, PART, J48* *72.5%*

Lesson 4.6: Ensemble learning

- Combining multiple models into “ensembles”
  - analogy with committees of humans
- Diversity helps, especially with “unstable” learners
  - when small changes in the training data can produce large changes in the learned model
- Create diversity by
  - Bagging: resampling the training set — meta>Bagging
  - Random forests: alternative branches in decision trees — trees>RandomForests
  - Boosting: focus on where the existing model makes errors — meta>AdaBoostM1
  - Stacking: combine results using another learner (instead of voting) — meta>Stacking



Course text

- Chapter 8 *Ensemble learning*

Data Mining with Weka

Department of Computer Science
University of Waikato
New Zealand

weka.waikato.ac.nz