Data Mining with Weka

Ian H. Witten

Computer Science Department
Waikato University
New Zealand

http://www.cs.waikato.ac.nz/~ihw
http://www.cs.waikato.ac.nz/ml/weka

The problem

Classification ("supervised")
Given
- A set of classified examples "instances"

Produce
- A way of classifying new examples

Instances: described by fixed set of features "attributes"
Classes: discrete or continuous "classification" "regression"

Interested in:
Results? (classifying new instances)
Model? (how the decision is made)

Association rules
Look for rules that relate features to other features

Clustering ("unsupervised")
There are no classes

Simplicity first!

- Simple algorithms often work very well!
- There are many kinds of simple structure, eg:
  - One attribute does all the work
  - All attributes contribute equally and independently
  - A decision tree involving tests on a few attributes
  - Rules that assign instances to classes
  - Distance in instance space from a few class "prototypes"
  - Result depends on a linear combination of attributes
- Success of method depends on the domain

Agenda

- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + Association rules
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and clustering
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

One attribute does all the work

- Learn a 1-level decision tree
  - i.e., rules that all test one particular attribute
- Basic version
  - One branch for each value
  - Each branch assigns most frequent class
  - Error rate: proportion of instances that don't belong to the majority class of their corresponding branch
  - Choose attribute with smallest error rate

```
For each attribute,
  For each value of the attribute, make a rule as follows:
    count how often each class appears
    find the most frequent class
    make the rule assign that class to this attribute-value
  Calculate the error rate of this attribute's rules
Choose the attribute with the smallest error rate
```

Example

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Attribute	Rules	Errors	Total errors
Outlook	Sunny → No	2/5	4/14
	Overcast → Yes	0/4	
	Rainy → Yes	2/5	
Temp	Hot → No*	2/4	5/14
	Mild → Yes	2/6	
	Cool → Yes	1/4	
Humidity	High → No	3/7	4/14
	Normal → Yes	1/7	
Wind	False → Yes	2/8	5/14
	True → No*	3/6	

* indicates a tie

Complications: Missing values

- Omit instances where the attribute value is missing
- Treat "missing" as a separate possible value

"Missing" means what?

- Unknown?
- Unrecorded?
- Irrelevant?

Is there significance in the fact that a value is missing?

Complications: Overfitting

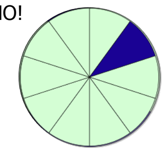
- Nominal vs numeric values for attributes

Outlook	Temp	Humidity	Wind	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
…	…	…	…	…

Attribute	Rules	Errors	Total errors
Temp	85 → No	0/1	0/14
	80 → No	0/1	
	83 → Yes	0/1	
	75 → Yes	0/1	
	…	…	

- Memorization vs generalization
- Do not evaluate rules on the training data
- Here, independent test data shows poor performance
- To fix, use
  - Training data — to form rules
  - Validation data — to decide on best rule
  - Test data — to determine system performance

Evaluating the result

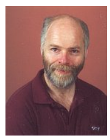


- Evaluate on training set? — NO!
- Independent test set
- Cross-validation
- Stratified cross-validation
- Stratified 10-fold cross-validation, repeated 10 times
- Leave-one-out
- The "Bootstrap"

One attribute does all the work

- This incredibly simple method was described in a 1993 paper
  - An experimental evaluation on 16 datasets
  - Used *cross-validation* so that results were representative of performance on new data
  - Simple rules often outperformed far more complex methods
- Simplicity first pays off!

"Very Simple Classification Rules Perform Well on Most Commonly Used Datasets"
Robert C. Holte, Computer Science Department, University of Ottawa



Agenda

- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + Association rules
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and clustering
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

Statistical modeling

One attribute does all the work?

- Opposite strategy: use *all* the attributes
- Two assumptions: Attributes are
  - *equally important a priori*
  - *statistically independent* (given the class value)

    I.e., knowing the value of one attribute says nothing about the value of another (*if the class is known*)
- Independence assumption is never correct!
- But … often works well in practice

Bayes's rule

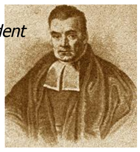
- Probability of event *H* given evidence *E*

$$\Pr[H \mid E] = \frac{\Pr[E \mid H]\Pr[H]}{\Pr[E]}$$

class instance

- *A priori* probability of *H* $\Pr[H]$
  - Probability of event *before* evidence is seen
- *A posteriori* probability of *H* $\Pr[H \mid E]$
  - Probability of event *after* evidence is seen
- "Naïve" assumption:
  - Evidence splits into parts that are *independent*

$$\Pr[H \mid E] = \frac{\Pr[E_1 \mid H]\Pr[E_1 \mid H]\ldots\Pr[E_n \mid H]\Pr[H]}{\Pr[E]}$$



Thomas Bayes
British mathematician and Presbyterian minister
Born 1702 Died 1761

Weather data: probabilities

Outlook	Yes	No	Temperature	Yes	No	Humidity	Yes	No	Wind	Yes	No	Play Yes	Play No
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Weather data: probabilities

Outlook	Yes	No	Temperature	Yes	No	Humidity	Yes	No	Wind	Yes	No	Play Yes	Play No
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/14	5/14
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5		
Rainy	3/9	2/5	Cool	3/9	1/5								

- A new day:

Outlook	Temp.	Humidity	Wind	Play
Sunny	Cool	High	True	?

Likelihood of the two classes

For "yes" = 2/9 × 3/9 × 3/9 × 3/9 × 9/14 = 0.0053

For "no" = 3/5 × 1/5 × 4/5 × 3/5 × 5/14 = 0.0206

Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

Weather data: probabilities

Outlook	Temp.	Humidity	Wind	Play
Sunny	Cool	High	True	?

← ***Evidence E***

Probability of class "yes"

$$\begin{aligned}\Pr[yes \mid E] = {} & \Pr[Outlook = Sunny \mid yes] \\ & \times \Pr[Temperature = Cool \mid yes] \\ & \times \Pr[Humidity = High \mid yes] \\ & \times \Pr[Windy = True \mid yes] \\ & \times \frac{\Pr[yes]}{\Pr[E]} \\ = {} & \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{\Pr[E]}\end{aligned}$$

Complications

Zero frequencies

- An attribute value doesn't occur with every class
  - Probability will be zero! $\Pr[Humidity = High \mid yes] = 0$

Missing values

- Training: do not include instance in frequency count for attribute value-class combination
- Classification: omit attribute from calculation
- Example:

Outlook	Temp.	Humidity	Wind	Play
?	Cool	High	True	?

Likelihood of "yes" = 3/9 × 3/9 × 3/9 × 9/14 = 0.0238

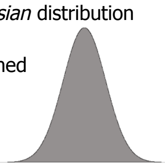
Likelihood of "no" = 1/5 × 4/5 × 3/5 × 5/14 = 0.0343

P("yes") = 0.0238 / (0.0238 + 0.0343) = 41%

P("no") = 0.0343 / (0.0238 + 0.0343) = 59%

Numeric attributes

- Often assume attributes have a *Gaussian* distribution (given the class)
- Its *probability density function* is defined by two parameters:
  - *Sample mean* $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
  - *Standard deviation* $\sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2$

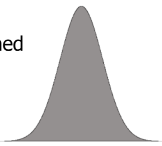


- The density function is $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Carl Friedrich Gauss
German mathematician and scientist
"The prince of mathematicians"
Born 1777 Died 1855

Numeric attributes

- Often assume attributes have a *Gaussian* distribution (given the class)
- Its *probability density function* is defined by two parameters:
  - *Sample mean* $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$
  - *Standard deviation* $\sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2$



- The density function is $f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
- A new day:

Outlook	Temp.	Humidity	Wind	Play
Sunny	66	90	true	?

Likelihood of "yes" = 2/9 × 0.0340 × 0.0221 × 3/9 × 9/14 = 0.000036
Likelihood of "no" = 3/5 × 0.0291 × 0.0380 × 3/5 × 5/14 = 0.000136
P("yes") = 0.000036 / (0.000036 + 0. 000136) = 20.9%
P("no") = 0.000136 / (0.000036 + 0. 000136) = 79.1%

"Naïve" statistical model

Naïve = assume attributes are independent

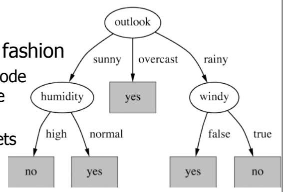
- Naïve Bayes works surprisingly well
  - even if independence assumption is clearly violated
- Why?
  - Because classification doesn't require accurate probability estimates
  - *so long as the greatest probability is assigned to the correct class*
- But: adding redundant attributes causes problems
  - e.g. identical attributes
- And: numeric attributes may not be normally distributed
  - → *kernel density estimators*

Agenda

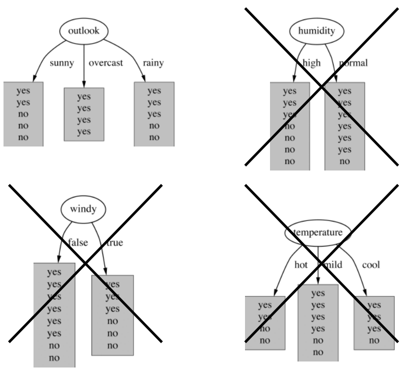
- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + Association rules
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and clustering
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

Constructing decision trees

- Strategy: top down
  Recursive *divide-and-conquer* fashion
  - First: select attribute for root node
    Create branch for each possible attribute value
  - Then: split instances into subsets
    One for each branch extending from the node
  - Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class



Which attribute to select?

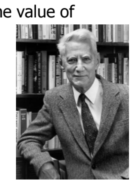


Which is the best attribute?

- Criterion: want to get the smallest tree
- Heuristic
  - choose the attribute that produces the "purest" nodes
  - I.e. the greatest *information gain*
- Information theory: measure information in bits

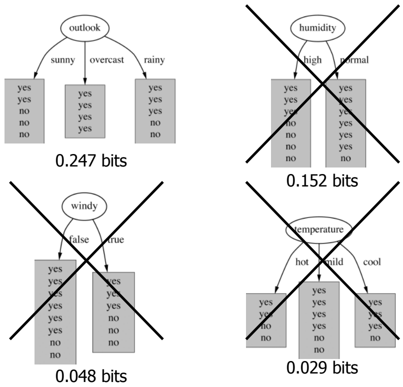
$$\text{entropy}(p_1, p_2, \ldots, p_n) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n$$

- *Information gain*
  - Amount of information gained by knowing the value of the attribute
  - Entropy of distribution before the split – entropy of distribution after it

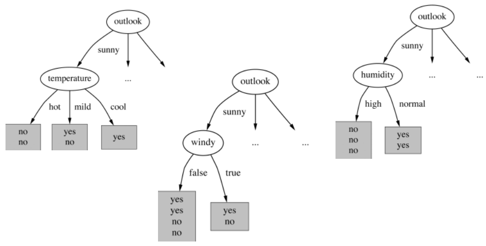


Claude Shannon
American mathematician and scientist
"The father of information theory"
Born 1916 Died 2001

Which attribute to select?



Continuing to split

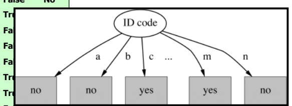


gain(*temperature*) = 0.571 bits
gain(*humidity*) = 0.971 bits
gain(*windy*) = 0.020 bits

Complications

- Highly-branching attributes
  - Extreme case: ID code

ID code	Outlook	Temp	Humidity	Wind	Play
a	Sunny	Hot	High	False	No
b	Sunny	Hot	High	Tr	
c	Overcast	Hot	High	Fa	
d	Rainy	Mild	High	Fa	
e	Rainy	Cool	Normal	Fa	
f	Rainy	Cool	Normal	Tr	
g	Overcast	Cool	Normal	Tr	
h	Sunny	Mild	High	False	No
i	Sunny	Cool	Normal	False	Yes
j	Rainy	Mild	Normal	False	Yes
k	Sunny	Mild	Normal	True	Yes
l	Overcast	Mild	High	True	Yes
m	Overcast	Hot	Normal	False	Yes
n	Rainy	Mild	High	True	No



Info gain is maximal
(0.940 bits)

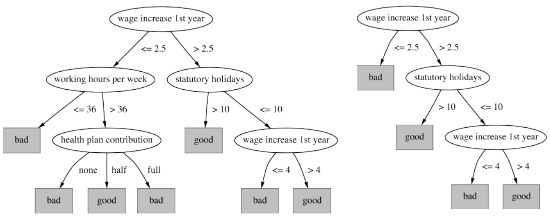
Complications

- Highly-branching attributes
  - Extreme case: ID code
- Overfitting: need to prune

Attribute	Type	1	2	3	…	40
Duration	(Number of years)	1	2	3		2
Wage increase first year	Percentage	2%	4%	4.3%		4.5
Wage increase second year	Percentage	?	5%	4.4%		4.0
Wage increase third year	Percentage	?	?	?		?
Cost of living adjustment	{none,tcf,tc}	none	tcf	?		none
Working hours per week	(Number of hours)	28	35	38		40
Pension	{none,ret-allw, empl-cntr}	none	?	?		?
Standby pay	Percentage	?	13%	?		?
Shift-work supplement	Percentage	?	5%	4%		4
Education allowance	{yes,no}	yes	?	?		?
Statutory holidays	(Number of days)	11	15	12		12
Vacation	{below-avg,avg,gen}	avg	gen	gen		avg
Long-term disability assistance	{yes,no}	no	?	?		yes
Dental plan contribution	{none,half,full}	none	?	full		full
Bereavement assistance	{yes,no}	no	?	?		yes
Health plan contribution	{none,half,full}	none	?	full		half
Acceptability of contract	{good,bad}	bad	good	good		good

Complications

- Highly-branching attributes
  - Extreme case: ID code
- Overfitting: need to prune



Complications

- Highly-branching attributes
  - Extreme case: ID code
- Overfitting: need to prune
  - Prepruning vs postpruning
- Missing values
  - During training
  - During testing: "fractional instances"
- Numeric attributes
  - Choose best "split point" for attribute
  - E.g. temp < 25

Constructing decision trees

Top-down induction of decision trees

- The most extensively studied method of machine learning used in data mining
- Different criteria for attribute selection
  - rarely make a large difference
- Different pruning methods
  - mainly change the size of the pruned tree
- Univariate vs multivariate decision trees
  - Single vs compound tests at the nodes
- C4.5 and CART

Ross Quinlan
Australian computer scientist
University of Sydney

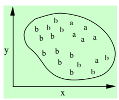
Agenda

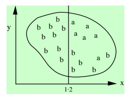
- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + Association rules
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and clustering
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

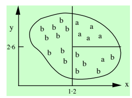
Constructing rules

- Convert (top-down) decision tree into a rule set
  - Straightforward, but rule set overly complex
  - More effective conversions are not trivial
- Alternative: (bottom-up) covering method
  - for each class in turn find rule set that covers all instances in it
    (excluding instances not in the class)
- *Separate-and-conquer* method
  - First identify a useful rule
  - Then separate out all the instances it covers
  - Finally "conquer" the remaining instances
- Cf *divide-and-conquer* methods:
  - No need to explore subset covered by rule any further

Generating a rule







```
If true
  then class = a
If x > 1.2
  then class = a
If x > 1.2 and y > 2.6
  then class = a
```

- Possible rule set for class "b":

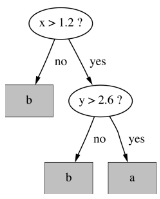
```
If x ≤ 1.2 then class = b
If x > 1.2 and y ≤ 2.6 then class = b
```

- Could add more rules, get "perfect" rule set

Rules vs. trees

- Corresponding decision tree: (produces exactly the same predictions)

```
If x ≤ 1.2 then class = b
If x > 1.2 and y ≤ 2.6 then class = b
```



- Rule sets *can* be more perspicuous
  - E.g. when decision trees contain replicated subtrees
- Also: in multiclass situations,
  - covering algorithm concentrates on one class at a time
  - decision tree learner takes all classes into account

Constructing rules

For each class C
- Initialize E to the instance set
- While E contains instances in class C
  - Create a rule R that predicts class C
    (with empty left-hand side)
  - Until R is perfect
    (or there are no more attributes to use)
    - For each attribute A not mentioned in R, and each value v,
      - Consider adding the condition A = v to the left-hand side of R
      - Select A and v to maximize the accuracy p/t
        (break ties by choosing the condition with the largest p)
    - Add A = v to R
  - Remove the instances covered by R from E

More about rules

- Rules are order-dependent
  - Two rules might assign different classes to an instance
- Work through the classes in turn
  - generating rules for that class
- For each class a "decision list" is generated
  - Subsequent rules are designed for instances that are not covered by previous rules
  - But: order doesn't matter because all rules predict the same class
- Problems: overlapping rules
- For better rules: globalization optimization

Association rules

- … can predict any attribute an
- … are not intended to be used
- Problem: immense number of
  - Output needs to be restricted predictive associations

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Define
  - Support: number of instances
  - Confidence: correct predictior
- Examples

```
If temperature = cool then humidity = normal
```

Support = 4, confidence = 100%

```
If Wind = false and play = no
   then outlook = sunny and humidity = high
```

Support = 2, confidence = 100%

- Specify minimum support and confidence
  - e.g. 58 rules with support ≥ 2 and confidence ≥ 95%

Constructing association rules

- To find association rules:
  - Use separate-and-conquer
  - Treat every possible combination of attribute values as a separate class
- Two problems:
  - Computational complexity
  - Huge number of rules (which would need pruning on the basis of support and confidence)
- But: we can look for high support rules directly!
  - Generate frequent "item sets"

```
Temperature = Cool, Humidity = Normal, Wind = False, Play = Yes (2)
```

  - From them, generate and test possible rules

```
Temperature = Cool, Wind = False ⇒ Humidity = Normal, Play = Yes
Temperature = Cool, Wind = False, Humidity = Normal ⇒ Play = Yes
Temperature = Cool, Wind = False, Play = Yes ⇒ Humidity = Normal
```

(all have support 2, confidence = 100%)

Example association rules

- Rules with support ≥ 2 and confidence 100%:

	Association rule		Sup.	Conf.
1	Humidity=Normal Wind=False	⇒ Play=Yes	4	100%
2	Temperature=Cool	⇒ Humidity=Normal	4	100%
3	Outlook=Overcast	⇒ Play=Yes	4	100%
4	Temperature=Cold Play=Yes	⇒ Humidity=Normal	3	100%
	...	...	...	...
58	Outlook=Sunny Temperature=Hot	⇒ Humidity=High	2	100%

support=4	3 rules
support=3	5 rules
support=2	50 rules
total	58

Association rules: discussion

- Market basket analysis: huge data sets

```
Buy beer ⇒ buy chips
Day = Thursday, buy beer ⇒ buy diapers
```

- May not fit in main memory
  - Different algorithms necessary
  - Minimize passes through the data
- Practical issue: generating a certain number of rules
  - e.g. by incrementally reducing minimum support
- Confidence is not necessarily the best measure
  - e.g. milk occurs in almost every supermarket transaction
  - Other measures have been devised (e.g. lift)

Agenda

- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + Association rules
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and clustering
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

Linear models

"Regression" = predicting a numeric quantity

- Standard technique: linear regression
  - Works most naturally with numeric attributes
  - Outcome is linear combination of attributes

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

- Calculate weights from the training data
- Predicted value for first training instance $\mathbf{a}^{(1)}$

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

- Choose weights to minimize squared error on the training data

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

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

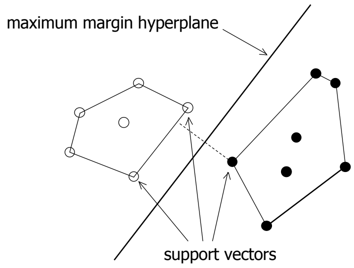
Classification by regression

- **Method 1**: *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 those that don't
  - Prediction: predict class that produces the largest output
- **Method 2**: *Pairwise linear regression*
  - Find a regression function for every *pair* of classes, using only instances from these two classes
    - Assign output of +1 to one class, −1 to the other
  - Prediction: use voting
    - Class that receives most votes is predicted
    - Alternative: "don't know" if there is no agreement
- **Method 3**: *Logistic regression*
  - Alternative to linear regression, designed for classification
  - Tries to estimate the class probabilities directly

Advanced linear models

- Linear model inappropriate if data exhibits non-linear dependencies
- But: can serve as building blocks for more complex schemes
- Support vector machine
  - Resilient to overfitting
  - Learn a particular kind of decision boundary
- Multilayer perceptron
  - Network of linear classifiers can approximate any target concept
  - An example of an artificial neural network
- Model tree
  - Decision tree with linear model at the nodes

Support vector machine

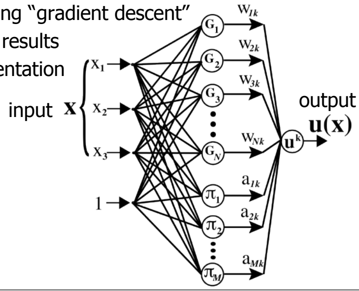


The support vectors define the maximum margin hyperplane

All other instances can be deleted without changing it!

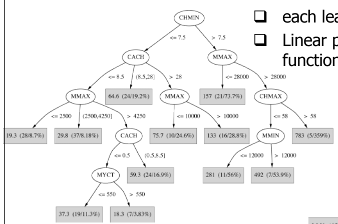
Multilayer perceptron

- Network of linear classifiers
  - Input layer, hidden layer(s), and output layer
- Parameters are found by backpropagation
- Minimize error using "gradient descent"
- Can get excellent results
- Involves experimentation

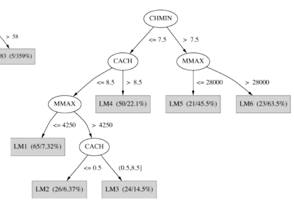


Trees for numeric prediction

- *Regression tree*
  - each leaf predicts a numeric quantity
  - Predict the average value of training instances at the leaf



- *Model tree*
  - each leaf has a linear regression models
  - Linear patches approximate continuous function



Discussion of linear models

- Linear regression: well-founded mathematical technique
- Can be used for classification in situations that are "linearly separable"
- … but very susceptible to noise
- Support vector machines yield excellent performance
  - particularly in situations with many redundant attributes
- Multilayer perceptrons ("neural nets") can work well
  - but often require much experimentation
- Regression/model trees grew out of decision trees
  - Regression trees were introduced in CART
  - Model trees were developed by Quinlan

Agenda

- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + Association rules
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and clustering
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

Instance-based learning

"Rote learning" = simplest form of learning

- Search training set for instance that's most like the new one
  - The instances themselves represent the "knowledge"
  - Noise will be a problem
- Similarity function defines what's "learned"
  - Euclidean distance
  - Nominal attributes? Set to 1 if different, 0 if same
  - Weight the attributes?
- *Lazy* learning: do nothing until you have to
- Methods:
  - *nearest-neighbor*
  - *k-nearest-neighbor*

Instance-based learning

- Often very accurate
- … but slow:
  - scan entire training data to make each prediction?
  - sophisticated data structures can make this much faster
- Assumes all attributes are equally important
  - Remedy: attribute selection or weights
- Remedies against noisy instances:
  - Majority vote over the k nearest neighbors
  - Weight instances according to their prediction accuracy
  - Identify reliable "prototypes" for each class
- Statisticians have used k-NN since 1950s
  - If $n \to \infty$ and $k/n \to 0$, error approaches minimum

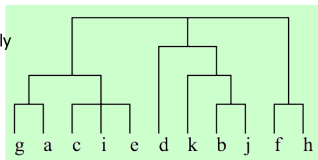
Clustering

Unsupervised vs supervised learning (classification)

- No target value to predict
- Differences between models/algorithms:
  - Exclusive vs. overlapping
  - Hierarchical vs. flat
  - Incremental vs. batch learning
  - Deterministic vs. probabilistic
- Evaluation?
  - Usually by inspection
  - Clusters-to-classes evaluation?
  - Probabilistic density estimation can be evaluated on test data

Hierarchical clustering

- Bottom up
  - Start with single-instance clusters
  - At each step, join the two closest clusters
  - How to define the distance between clusters?
    - Distance between the two closest instances?
    - Distance between the means
- Top down
  - Start with one universal cluster
  - Find two clusters
  - Proceed recursively on each subset



Iterative: fixed num of clusters

The k-means algorithm

To cluster data into k groups *(k is predefined)*

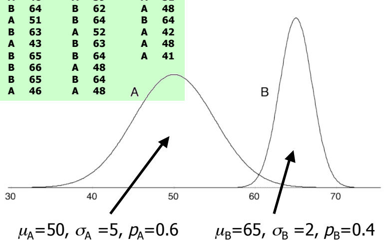
1. Choose k cluster centers ("seeds")
   - e.g. at random
2. Assign instances to clusters
   - based on distance to cluster centroids
3. Compute centroids of clusters
4. Go to step 1
   - until convergence

- Results can depend strongly on initial seeds
- Can get trapped in local minumum
  - Rerun with different seeds?

Probabilistic clustering

- Model data using a *mixture* of normal distributions
- One cluster, one distribution
  - governs probabilities of attribute values in that cluster
- *Finite mixtures* : finite number of clusters

A	51	B	62	B	64	A	48	A	39	A	51
A	43	A	47	A	51	B	64	B	62	A	48
B	62	A	52	A	52	A	51	B	64	B	64
B	64	B	64	B	62	B	63	A	52	A	42
A	45	A	51	A	49	A	43	B	63	A	48
A	42	B	65	A	48	B	65	B	64	A	41
A	46	A	48	B	62	B	66	A	48		
A	45	A	49	A	43	B	65	B	64		
A	45	A	46	A	40	A	46	A	48		



Using the mixture model

- Probability that instance x belongs to cluster A:

$$\Pr[A \mid x] = \frac{\Pr[x \mid A]\Pr[A]}{\Pr[x]} = \frac{f(x;\mu_A,\sigma_A)p_A}{\Pr[x]} \qquad f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(x-\mu)^2}{2\sigma^2}}$$

- *Likelihood* of an instance given the clusters:

$$\Pr[x \mid \text{the distributions}] = \sum_i \Pr[x \mid \text{cluster}_i]\Pr[\text{cluster}_i]$$

- Learn the clusters ⇒
  - determine their parameter, ie mean, standard deviation
- Performance criterion:
  - *likelihood of training data given the clusters*
- Iterative Expection-Maximization (EM) algorithm
  - E step: Calculate cluster probability for each instance
  - M step: Estimate distribution parameters from cluster probabilities
- Finds a local maximum of the likelihood

Extending the mixture model

- More then two distributions: easy
- Several attributes: easy—assuming independence!
- Correlated attributes: difficult
  - Joint model: bivariate normal distribution with a (symmetric) covariance matrix
  - n attributes: need to estimate $n + n(n+1)/2$ parameters
- Nominal attributes: easy (if independent)
- Missing values: easy
- Can use other distributions than normal:
  - "log-normal" if predetermined minimum is given
  - "log-odds" if bounded from above and below
  - Poisson for attributes that are integer counts
- Unknown number of clusters:
  - Use cross-validation to estimate k

Bayesian clustering

- Problem: many parameters ⇒ EM overfits
- *Bayesian approach* : give every parameter a prior probability distribution
  - Incorporate prior into overall likelihood figure
  - Penalizes introduction of parameters
- Eg: Laplace estimator for nominal attributes
- Can also have prior on number of clusters!
- Implementation: NASA's AUTOCLASS

Agenda

- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + Association rules
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and clustering
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

Engineering the input & output

Just apply a learner? – NO!

- Attribute selection
  - Scheme-independent, scheme-specific
- Attribute discretization
  - Unsupervised, supervised
- Data transformations
  - Ad hoc, Principal component analysis
- Dirty data
  - Data cleansing, robust regression, anomaly detection
- Combining multiple models
  - Bagging, randomization, boosting, stacking
- Using unlabeled data
  - Co-training

Attribute selection

- Adding a random (i.e. irrelevant) attribute can significantly degrade C4.5's performance
  - Problem: attribute selection based on smaller and smaller amounts of data
- IBL very susceptible to irrelevant attributes
  - Number of training instances required increases exponentially with number of irrelevant attributes
- Naïve Bayes doesn't have this problem
- Relevant attributes can also be harmful

Data transformations

- Simple transformations can often make a large difference in performance
- Example transformations (not necessarily for performance improvement):
  - Difference of two date attributes
  - Ratio of two numeric (ratio-scale) attributes
  - Concatenating the values of nominal attributes
  - Encoding cluster membership
  - Adding noise to data
  - Removing data randomly or selectively
  - Obfuscating the data
- Principal component analysis

Principal component analysis

- Method for identifying the important "directions" in the data
- Can rotate data into (reduced) coordinate system that is given by those directions
- Algorithm:
  1. Find direction (axis) of greatest variance
  2. Find direction of greatest variance that is perpendicular to previous direction and repeat
- Implementation: find eigenvectors of covariance matrix by diagonalization
  - Eigenvectors (sorted by eigenvalues) are the directions

Combining multiple models

- Basic idea:
  build different "experts," let them vote
- Advantage:
  - often improves predictive performance
- Disadvantage:
  - usually produces output that is very hard to analyze
  - but: there are approaches that aim to produce a single comprehensible structure
- Methods
  - Bagging
  - Randomization
  - Boosting
  - Stacking

Bagging

- Combining predictions by voting/averaging
  - Simplest way
  - Each model receives equal weight
- "Idealized" version:
  - Sample several training sets of size n
    (instead of just having one training set of size n)
  - Build a classifier for each training set
  - Combine the classifiers' predictions
- Learning scheme is unstable Þ
  almost always improves performance
  - Small change in training data can make big change in model (e.g. decision trees)

Randomization

- Can randomize learning algorithm instead of input
- Some algorithms already have a random component: eg. initial weights in neural net
- Most algorithms can be randomized, eg. greedy algorithms:
  - Pick from the N best options at random instead of always picking the best options
  - Eg.: attribute selection in decision trees
- More generally applicable than bagging: e.g. random subsets in nearest-neighbor scheme
- Can be combined with bagging

Boosting

- Also uses voting/averaging
- Weights models according to performance
- Iterative: new models are influenced by performance of previously built ones
  - Encourage new model to become an "expert" for instances misclassified by earlier models
  - Intuitive justification: models should be experts that complement each other
- Several variants

Stacking

- To combine predictions of base learners, don't vote, use *meta learner*
  - 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
- Hard to analyze theoretically: "black magic"

Using unlabeled data

- Semisupervised learning: attempts to use unlabeled data as well as labeled data
  - The aim is to improve classification performance
- Why try to do this? Unlabeled data is often plentiful and labeling data can be expensive
  - Web mining: classifying web pages
  - Text mining: identifying names in text
  - Video mining: classifying people in the news
- Leveraging the large pool of unlabeled examples would be very attractive

Co-training

- Method for learning from multiple views (multiple sets of attributes), eg:
  - First set of attributes describes content of web page
  - Second set of attributes describes links that link to the web page
- Step 1: build model from each view
- Step 2: use models to assign labels to unlabeled data
- Step 3: select those unlabeled examples that were most confidently predicted (ideally, preserving ratio of classes)
- Step 4: add those examples to the training set
- Step 5: go to Step 1 until data exhausted
- Assumption: views are independent

Agenda

- A very simple strategy
  - Overfitting, evaluation
- Statistical modeling
  - Bayes rule
- Constructing decision trees
- Constructing rules
  - + **Association rules**
- Linear models
  - Regression, perceptrons, neural nets, SVMs, model trees
- Instance-based learning and **Clustering**
  - Hierarchical, probabilistic clustering
- Engineering the input and output
  - Attribute selection, data transformations, PCA
  - Bagging, boosting, stacking, co-training

Data mining with Weka

- There is no magic in data mining
  - Instead, a huge array of alternative techniques
- There is no single universal "best method"
  - Experiment! Which ones work best on your problem?
- The WEKA machine learning workbench
  - *http://www.cs.waikato.ac.nz/ml/weka*
- *Data mining: practical machine learning tools and techniques* by Ian H. Witten and Eibe Frank, 2005