

# Miscellaneous Topics

---

- Attribute Selection;
- Data cleansing;
- Dealing with massive data set;
- Discretizing numeric attributes;

**Reading material:** Chapter 7,9 of the text-book by Witten.

# Attribute selection

---

**Theoretical illusion:** irrelevant attributes won't be selected in the process of learning, because they play no role in the final decision

**In practice:** irrelevant attributes impact the performance of the learning scheme

- Adding a binary irrelevant attribute cause C4.5 to deteriorate (from 5% to 10%).

- Divide-conquer tree learners and rule learners suffer from irrelevant attributes

- Instance-based learners are susceptible to those attributes because they might change the neighborhoods

- Naive Bayes model does not suffer from this, but suffer from redundant attributes.

**Surprising:** relevant attributes might also be harmful!

**Manual selection** is desirable, but too time-consuming

**Automatic methods:** delete unsuitable attributes and reduce the dimensionality of the data, resulting in more compact and easily interpretable representation

---

# Scheme-independent selection

---

**The filter method** filters the attribute set to produce the most promising set

- Assessment based on general characteristics of the data

How about finding a subset of attributes that is enough to separate all the instances?

- Expensive and overfitting

**Alternative:** use one learning scheme(i.e. 1R) to select attributes and use the resulting attribute set for another scheme.

**Issue:** Algorithms (IBL-based weighting techniques) can't find redundant or irrelevant attributes.

**Searching the attribute space:** Number of possible attribute subsets is exponential in the number of attributes

**Greedy approaches:** forward selection and backward elimination

**Other strategies** include: Bidirectional search, Best-first search, Beam search and Genetic algorithms

---

# Scheme-specific selection

---

**The Wrapper method:** attribute selection implemented as wrapper around learning scheme

- Evaluation via cross-validation

**Time Complexity:** adds factor  $k^2$  even for greedy approaches with  $k$  attributes

- Linearity in  $k$  requires prior ranking of attributes

Forward selection might reduce the number of attributes without much influence on the accuracy, it might help us to understand the decision structure

Scheme-specific attribute selection essential for learning decision tables (DTs)

Can be done efficiently for DTs

Forward selection works well for Naive Bayes, because irrelevant attribute won't degrade the performance of the scheme, while redundant attributes can be more likely caught by forward selection.

# Discretizing numeric attributes

---

**Discretizing numeric attributes** can avoid making normality assumption in Naive Bayes and Clustering

Simple discretization scheme is used in 1R

C4.5 performs local discretization

Global discretization can be advantageous because it's based on more data

- Learner can be applied to discretized attribute
- The discretized attribute is ordinal. Thus the learner that able to deal with ordinal attributes is a favorable choice.

**Unsupervised discretization** generates intervals without looking at class labels

- Only possible way in clustering

**Two main strategies:**

- Equal-interval binning
- Equal-frequency binning based on the distribution

Inferior to supervised schemes in classification tasks

# Entropy-based discretization

---

**Supervised method** that builds a decision tree with pre-pruning on the attribute being discretized

- Entropy used as splitting criterion
- MDLP used as stopping criterion

## Implementing MDL:

- "Theory": the splitting point( $\log_2(N - 1)$ ) + class distribution in each subset
- DL before/after splitting is compared

## Example

64 65 • 68 69 70 • 71 72 72 • 75 • 80 • 81 83 • 85  
Y no F Y Y Y E no no Y D YY C no B Y Y A no

## Assumptions:

- N instances in  $S = S_1 + S_2$ .
- k classes and entropy E in S;
- $k_1$  classes and  $E_1$  entropy in  $S_1$ ;
- $k_2$  classes and  $E_2$  entropy in  $S_2$

If

$$gain > \frac{\log_2(N-1) + \log_2(3^k - 2) - kE + k_1E_1 + k_2E_2}{N},$$

then split the data set. Otherwise not.

Applying this criteria leads to no discretization for the temperature attribute at all.

# Other discretization methods

---

## Bottom-up method

- Place each instance into its own interval and then merge adjacent intervals;
- $\chi$ -squared test is used for merging

**Dynamic programming:** find optimum k-way split

- Time complexity quadratic in number of instances if entropy is used as criterion
- Linear time if error rate is used

## Converting nominal into numeric

---

**IB1:** indicator attributes

- Doesn't make use of ordering information

**M5** generates ordering of nominal values and codes ordering using binary attributes

This strategy can be used for any ordinal attribute

- Avoids problem of using integer attribute to code ordering: would imply a metric

In general: subsets of attributes coded as binary attributes

---

# Automatic data cleansing

---

**Improving decision trees:** relearn tree with misclassified instances removed

- Taking a pruning decision to its logical conclusion directly

Better let expert check(or correct) misclassified instances

When systematic noise is present, better not to modify the training set only

- The test set is 'noisy' as well;

(Unsystematic) class noise in training set should be eliminated if possible

**Robust regression:** Statistical methods dealing with outliers

- Minimize absolute(not the squared) error
- Remove outliers based on distribution(i.e. 10% of points farthest from the regression plane)

- Minimize median(not mean of squares)
  - Finds narrowest strip covering half the observations

# Detecting anomalies

---

**Visualization:** best, but not easy

**Automatic approach:** committee of different learning schemes

- E.g. C4.5, K-NN and SVM
- Conservative choice: only delete instances that are incorrectly classified by all of them

**Potential issue:** Schemes are biased, might sacrifice instances of a particular class

## Combining multiple models

---

**Meta learning:** build different “experts” and let them vote

- Advantage: improves predictive performance
- Disadvantage: hard to analyze output

**Approaches:** bagging, boosting, stacking, and error-correcting output codes

- The first three can be applied to both classification and numeric prediction problems

# Bagging

---

Simple way of meta learning: voting/averaging

- Each model receives equal weight
- Idealized version of bagging:
  - Sample several training sets of size  $n$  (not just one training set of size  $n$ )
  - Build a classifier for each training set
  - Combine classifiers' predictions

This improves performance if the learning scheme is unstable(i.e., decision tree)

**Bias-variance decomposition:** theoretical tool for analyzing the influence of the training set on classifiers

**Assumption:** infinite classifiers built from different training sets of size  $n$

- A learning scheme's bias is the expected error of its meta version on new data
- The variance of a learning scheme is the expected error on a particular training set
- Total expected error: bias+variance

Compare with bootstrapping in evaluating the performance of a scheme!

---

# Bagging algorithm

---

**Bagging = Bootstrap + Aggregating:**

- Reduces variance and the overall expected error by voting/averaging
  - There are cases where the overall error might increase
  - Usually, the more classifiers the better

**Problem:** we only have one dataset!

**Solution:** Resampling from the original training set.

## Bagging algorithm

---

**Model generation:**

For each of  $t$  iterations:

- Sample  $n$  instances with replacement from training set.
- Apply the algorithm to the sample.
- Store the resulting model.

**Classification:**

For each of the  $t$  models:

- Predict class of instance using model.

Return class by voting.

---

# Boosting

---

**Note:** Bagging is pointless for stable scheme!

**Boosting:** uses voting/averaging where based on weighted models

**Iterative procedure:**

- New models are influenced by performance of previous ones
- New model is biased to instances classified incorrectly by earlier models

**Intuitive justification:** models should be experts that complement each other

**Variants of Boosting:** AdaBoost.M1

- Model generation
- Classification

# AdaBoost.M1

---

## Model generation:

Assign equal weight to each training instance.

For each of  $t$  iterations:

- Apply learning algorithm to weighted dataset and store resulting model.
- Compute error  $e$  of model on weighted dataset and store error. If  $e = 0$ , or  $e \geq 0.5$ :
  - Terminate model generation.
- For each instance in dataset:
  - If instance classified correctly by model:
    - Multiply weight of instance by  $\frac{e}{1-e}$ .
- Normalize weight of all instances.

## Classification:

Assign weight of zero to all classes.

For each of the  $t$  (or less) models:

- Add  $-\log \frac{e}{1-e}$  to weight of class predicted by model.

Return class with highest weight.

# Comments on boosting

---

If the learning scheme can not cope with weighted instances, resampling with probability determined by weights can be applied

- Disadvantage: low weighted instances might be lost in resampling
- Advantage: resampling can be repeated if error exceeds 0.5

**Theoretical result:** training error decreases exponentially

Works if base classifiers not too complex and their error doesn't increase too quickly

**Puzzling fact:** generalization error for fresh data can decrease long after training error has reached zero

**Reason:** boosting can increase margin

**Margin:** difference between estimated probability of the true class and that of most likely predicted other (not the true) class

Boosting works with weak learners: only condition is that error doesn't exceed 0.5

# Stacking

---

**Stacking:** uses meta learner to combine predictions from base learners

- Predictions of base learners (level-0 models) are used as input for meta learner (level-1 model)

Base learners are usually different learning schemes

Predictions on training data can't be used to generate data for level-1 model!

- Cross-validation scheme is employed

If base learners can output probabilities, then use probabilities as input to meta learner

**Meta learner generation:**

- D. Wolpert: “relatively global, smooth” model
  - Base learners do most of the work
  - Reduces risk of overfitting

Stacking can also be applied to numeric prediction (and density estimation)

# Error-correcting output codes

---

**EOC:** a elegant method of transforming multiclass problem into two-class problem

## Simple Example

vector	vector class
a	1000
b	0100
c	0010
d	0001

## EOCO Example

vector	vector class
a	1111111
b	0000111
c	0011001
d	0101010

**Quiz:** What's the true class if base classifiers predict 1011111?

**Answer:** compare the hamming distance to each class.

Suppose that the minimum hamming distance between any pair of words is  $d$

- Up to  $\frac{d-1}{2}$  bit errors can be corrected

## Measurements:

- Row-separation: minimum distance between rows
- Column-separation: minimum distance between columns (and their complements)

# Exhaustive ECOCs

---

Column separation is necessary:

- If columns are identical, base classifiers will make the same errors
- Error-correction is weakened if errors are correlated

**ECOC** only works for problems with more than 3 classes, but practically infeasible for large  $k$ . Why?

**Exhaustive code for  $k$  classes:**

- Columns might be any possible  $k$ -string except for complements and all zero/ one strings

- Each code word contains  $2^{k-1} - 1$  bits

**Code word for first class:** all ones

**Second class:**  $2^{k-2}$  zeroes followed by  $2^{k-2} - 1$  ones

**The  $i$ -th class:** alternating runs of  $2^{k-i}$  zeroes and ones, the last run being one short

ECOC don't work with NN classifier

- It works if different attribute subsets are used to predict each output bit

# Mining massive data set

---

**What to do** for massive data set?

- No problem for incremental-type learning schemes like naive Bayes and one R
  - Each time we can input one instance, update the model and move to the next one;
- Model tree and instanced-based learning do not work well because they need information for all the instances

## **Complexity issues**

- If the running time is not linear, then the algorithm will become practically infeasible for large data set
  - Sometimes the number of attributes is also crucial
    - Constructing a decision tree for a data set with hundreds attributes requires tremendous work
  - Bayes is linear in number of instances and number of attributes, Top-down decision tree inducer is also linear in attribute's number, but log-linear in number of instances

# Dealing with large dataset

---

## Approaches:

- Subsampling the training set is necessary,
- Evaluating on a different sample set

## Sampling

- Simple random sample with(out) replacement of size  $n$ , out of a set of  $N$  instances;
  - Cluster sample: if the whole data set is grouped, then sample in each cluster;
  - Stratified sample: if the dataset is divided into mutually disjoint parts, then we can get a stratified sample by obtaining SRS in each stratum.

## Design algorithm with lower complexity,

- Stochastic algorithms give nice approximation at relatively lower time complexity.
- Fixing irrelevant and redundant attributes
  - Use simple algorithm (like 1R) for attribute selection
  - Use data compression techniques to reduce the data set size (wavelet transform);
  - Parallelization provides an efficient to reduce the time complexit

# Visualization

---

**Visualization:** easily understandable, helpful for selecting breaking points and attributes.

- A nice visualization of the output is significant for the success in practice.

**Example:** Iris Data in input.

**Domain knowledge** is crucial:

- **A semantic relation** between two attributes indicates that if the first is in a rule, so is the second one.

- For a data base from a supermarket, the attribute 'milk' might appear in the rule together with the address and the name of the suppliers. Semantic relation is associated with the data set as well as the mining task.

- **Causal relations** occurs when one attribute causes another one. A chain might be formulated by causal relations.

**Functional dependence:** like the correlation among attributes.