CO544 Machine Learning and Data Mining

Sampath Deegalla

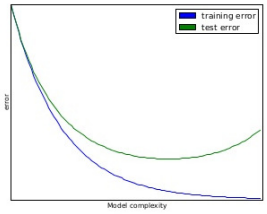
22.09.2017

Overview

- Data transformation
  - Attribute Selection (Feature Selection)
  - Projections (Dimensionality Reduction)

Why feature selection?

- Overfitting
- Interpretability
- Computational Complexity



Source: https://www.slideshare.net/zombiecalypse/feature-select

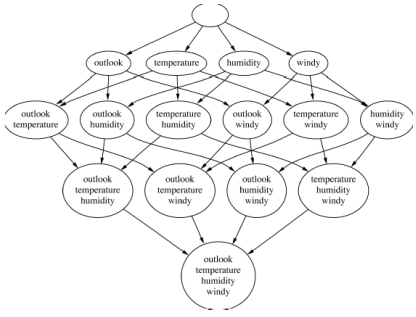
Attribute Selection

- Also known as feature selection
- Adding a random (i.e. irrelevant) attribute can significantly degrade performance of the learning algorithm (e.g. C4.5)
- Instance based learning algorithms (e.g. Nearest Neighbor) 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

Scheme-independent attribute selection

- Filter approach: assess based on general characteristics of the data
- One method: find smallest subset of attributes that separates data
- Another method: use different learning scheme
- e.g. use attributes selected by C4.5 and 1R, or coefficients of linear model (e.g. SVM), possibly applied recursively. It builds a model, ranks the attributes based on the coefficients, removes the lowest-ranked one, and repeats the process until all attributes have been removed. (recursive feature elimination)

Attribute subsets for weather data



Searching attribute space

- Number of attribute subsets is exponential in number of attributes
- Common greedy approaches (Greedy search):
  - *forward selection: Start with no attributes, then add one at a time (top to bottom)*
  - *backward elimination: Start with all the attributes, then delete one at a time (bottom to top)*

Searching attribute space (cont.)

- More sophisticated strategies:
  - *Bidirectional search: use both top-to-bottom and bottom-to-top strategies at the same time (selection + elimination)*
  - *Best-first search: can find optimum solution (start with empty set and add feature with highest performance)*
  - *Beam search: approximation to best-first search*
  - *Genetic algorithms: generate solutions to optimization problems (optimal attribute set) using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.*

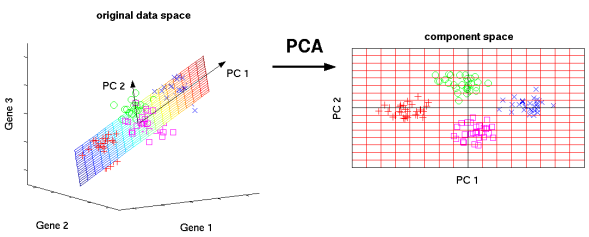
Scheme-specific selection

- Wrapper approach to attribute selection
- Implement "wrapper" around learning scheme
  - Evaluation criterion: cross-validation performance
- Time consuming
  - greedy approach, k attributes → k^2 x time
  - prior ranking of attributes → linear in k
- Can use significance test to stop cross-validation for subset early if it is unlikely to "win" (race search)
  - can be used with forward, backward selection, prior ranking, or special-purpose schemata search
- Learning decision tables: scheme-specific attribute selection essential
- Efficient for decision tables and Naïve Bayes

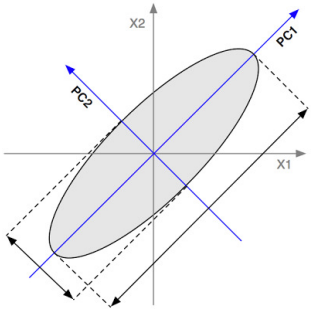
Projections

- 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

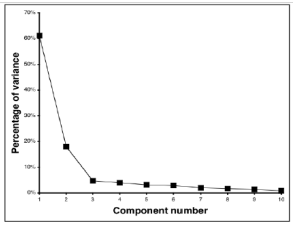


Principal component analysis

- 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

Example: 10-dimensional data

Axis	Variance	Cumulative
1	61.2%	61.2%
2	18.0%	79.2%
3	4.7%	83.9%
4	4.0%	87.9%
5	3.2%	91.1%
6	2.9%	94.0%
7	2.0%	96.0%
8	1.7%	97.7%
9	1.4%	99.1%
10	0.9%	100.0%

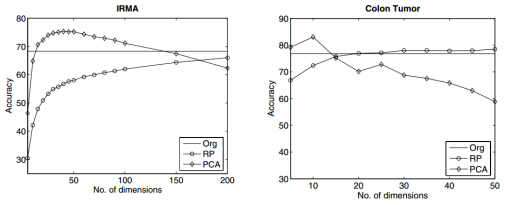


- Can transform data into space given by components
- Data is normally standardized for PCA
- Could also apply this recursively in tree learner

Random projections

- PCA is nice but expensive: cubic in number of attributes
- Alternative: use random directions (projections) instead of principle components
- Surprising: random projections preserve distance relationships quite well (on average)
  - Can use them to apply *kD-trees to high-dimensional*data
  - Can improve stability by using ensemble of models based on different projections

Example: PCA vs RP



S. Deegalla and H. Boström. Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification. In Proceedings of the Fifth International Conference on Machine Learning and Applications, pages 245–250, 2006

One-Class Learning

- Usually training data is available for all classes
- Some problems exhibit only a single class at training time
  - Test instances may belong to this class or a new class not present at training time
- One-class classification
  - Predict either *target or unknown*
- Some problems can be re-formulated into two-class ones
- Other applications truly don't have negative data
  - Eg password hardening in information security

Outlier detection

- One-class classification is often called *outlier/novelty* detection
- Generic approach: identify outliers as instances that lie beyond distance *d from percentage p of the training* data
- Alternatively, estimate density of the target class and mark low probability test instances as outliers
  - Threshold can be adjusted to obtain a suitable rate of outliers

Generating artificial data

- Another possibility is to generate artificial data for the outlier class
  - Can then apply any off-the-shelf classifier
  - Can tune rejection rate threshold if classifier produces probability estimates
- Generate uniformly random data
  - Too much will overwhelm the target class!
- Can be avoided if learning accurate probabilities rather than minimizing classification error
  - Curse of dimensionality – as # attributes increase it becomes infeasible to generate enough data to get good coverage of the space

Transforming multiple classes to binary ones

- Some learning algorithms only work with two class problems
  - Sophisticated multi-class variants exist in many cases but can be very slow or difficult to implement
- A common alternative is to transform multi-class problems into multiple two-class ones
- Simple methods
  - Discriminate each class against the union of the others – *one-vs.-rest*
- Build a classifier for every pair of classes – *pairwise classification*