COMP30120

Feature Selection

Derek Greene

School of Computer Science and Informatics
Autumn 2015

Overview

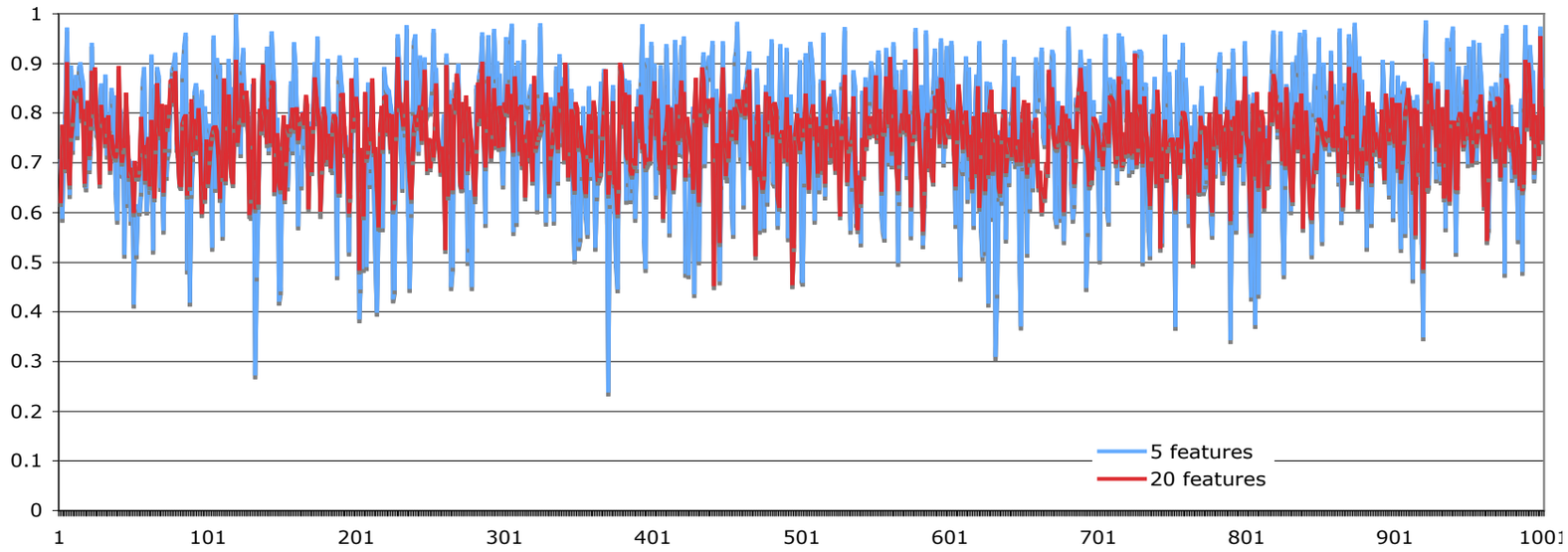
- Dimension Reduction
  - The Curse of Dimensionality
  - Feature Transformation v Selection
- Feature Selection in Supervised Learning
  - Filter v Wrapper strategy
  - Filter approaches
  - Wrapper approaches
  - Stochastic search
  - Overfitting in feature selection

Curse of Dimensionality

- Real data often represented by many features ⇒ high-dimensional
  e.g. Google N-gram corpus: 13,588,391 unique words, after discarding words that appear less than 200 times.
- To build a model from data, the number of examples required per feature increases exponentially with number of features.
- High-dimensional spaces tend to be very *sparse*, so every point is equally far away from virtually every other point, and distances tend to be uninformative.

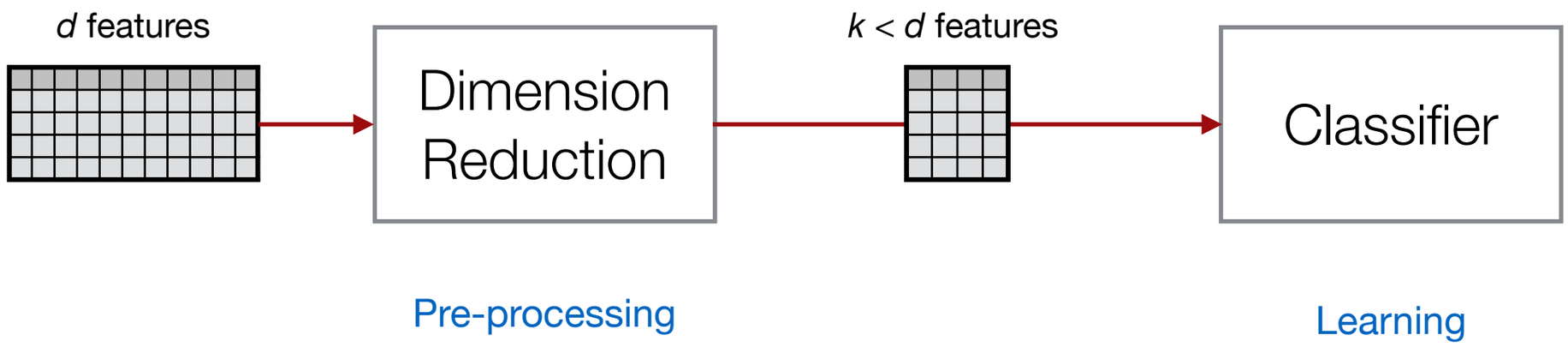
➡ The more dimensions you have, the more similar things appear.

Similarities as number of dimensions increases



Curse of Dimensionality

- In practice, the curse of dimensionality means that, for a given number of samples, there is a maximum number of features beyond which the performance of a classifier will degrade rather than improve (Bellman, 1961).
- Often try to beat the curse of dimensionality by applying pre-processing techniques to reduce the number of features.
- In most cases, the additional information that is lost by removing some features is (more than) compensated by higher accuracy in the lower dimensional space.



Feature Transformation v Selection

Feature Transformation (Feature Extraction)

- Transforms the original features of a data set to a new, smaller, more compact feature set, while retaining as much information as possible.
  e.g. Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA)

Feature Selection

- Tries to find a minimum subset of the original features that optimises one or more criteria, rather than producing an entirely new set of dimensions for the data.

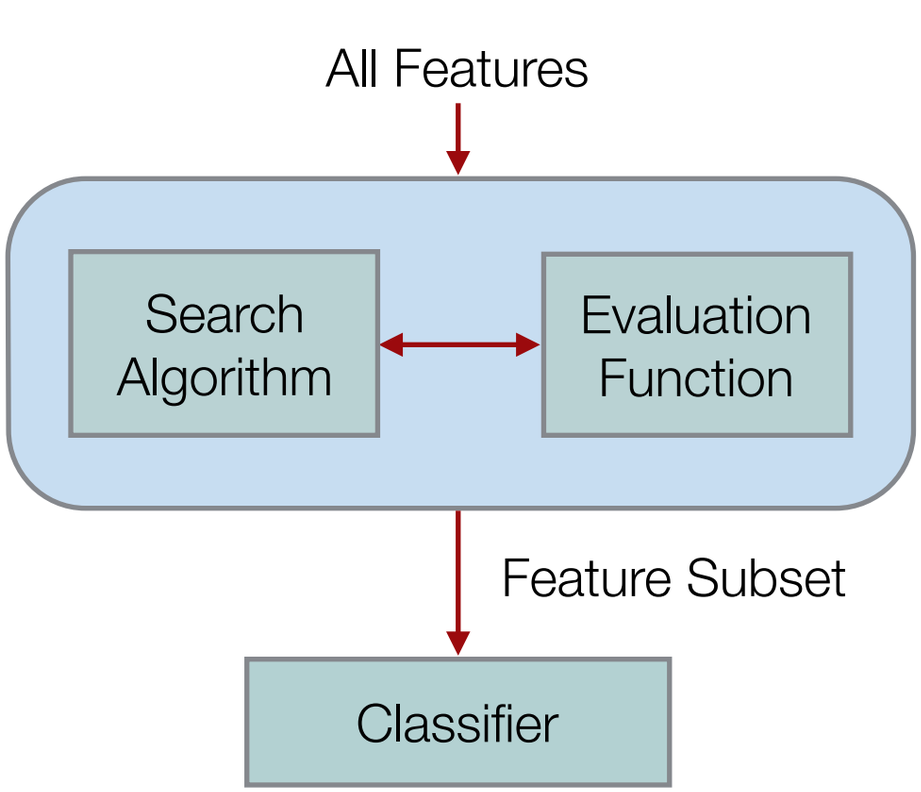
Feature Selection

- Feature Subset Selection: Find the best subset of all available features, which contains the least number of dimensions that most contribute to accuracy. Discard the remaining, unimportant dimensions.
- Why select subset of original features?
  1. Building a better classifier - Redundant or noisy features damage accuracy.
  2. Knowledge discovery - Identifying useful features helps us learn about the domain.
  3. Features expensive to obtain - Test a large number of features, select a few for the final system (e.g. sensors, manufacturing).
  4. Interpretability - Selected features still have meaning. We can extract meaningful rules from the classifier.

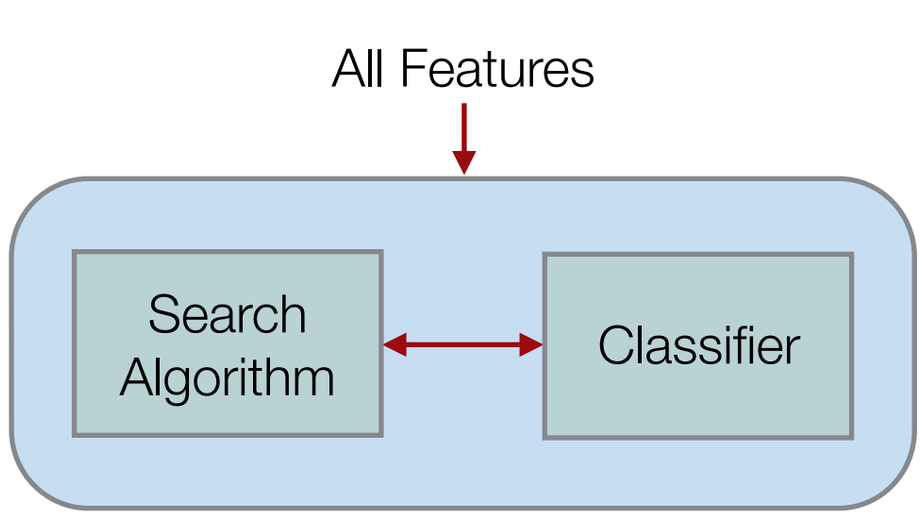
Feature Selection Strategies

- Feature selection is in general NP-hard.
- Brute force evaluation of all feature subsets involves $\binom{d}{k}$ combinations if k is fixed, or 2^d combinations if not fixed.
- Two broad strategies for feature selection:

Filter



Wrapper



Filters

- Pre-processing step that ranks and “filters” features independently of the choice of classifier that will be subsequently applied.
- **Evaluation function:** How does a filter algorithm score different feature subsets to produce an overall ranking?
- Generally score the predictiveness of the features.
  - Information theoretic analysis

    e.g. Information Gain, Breiman’s Gini index
  - Statistical tests

    e.g. Chi-square statistic
  - Relief algorithm

    Filter for binary classification, based on the nearest-neighbour classification algorithm (Kira & Rendell, 1992).

Relief Algorithm

- **Motivation:** Want to find important features of example *X* in a class. So look at an example *Y* similar to *X* from the same class, and a different example *Z* from the other class. Features that are similar between *X* and *Y*, but different between X and *Z,* are likely to be important.

- For each example, two neighbours are selected:
  1. *Nearest hit*: its nearest neighbour from the same class
  2. *Nearest miss*: its nearest neighbour from other class
- To score *i*-th feature:
  1. Measure distances using only that feature.
  2. Score feature based on average over all examples of difference between distance to nearest hit and distance to nearest miss.
- Return a ranked list of highest-scoring features or the top *k* features based on a given score threshold.

Information Gain

- Given a set of training examples S, where p is the proportion of positive examples, q is the proportion of negative examples.

Entropy $$H(S) = -p\log_2(p) - q\log_2(q)$$

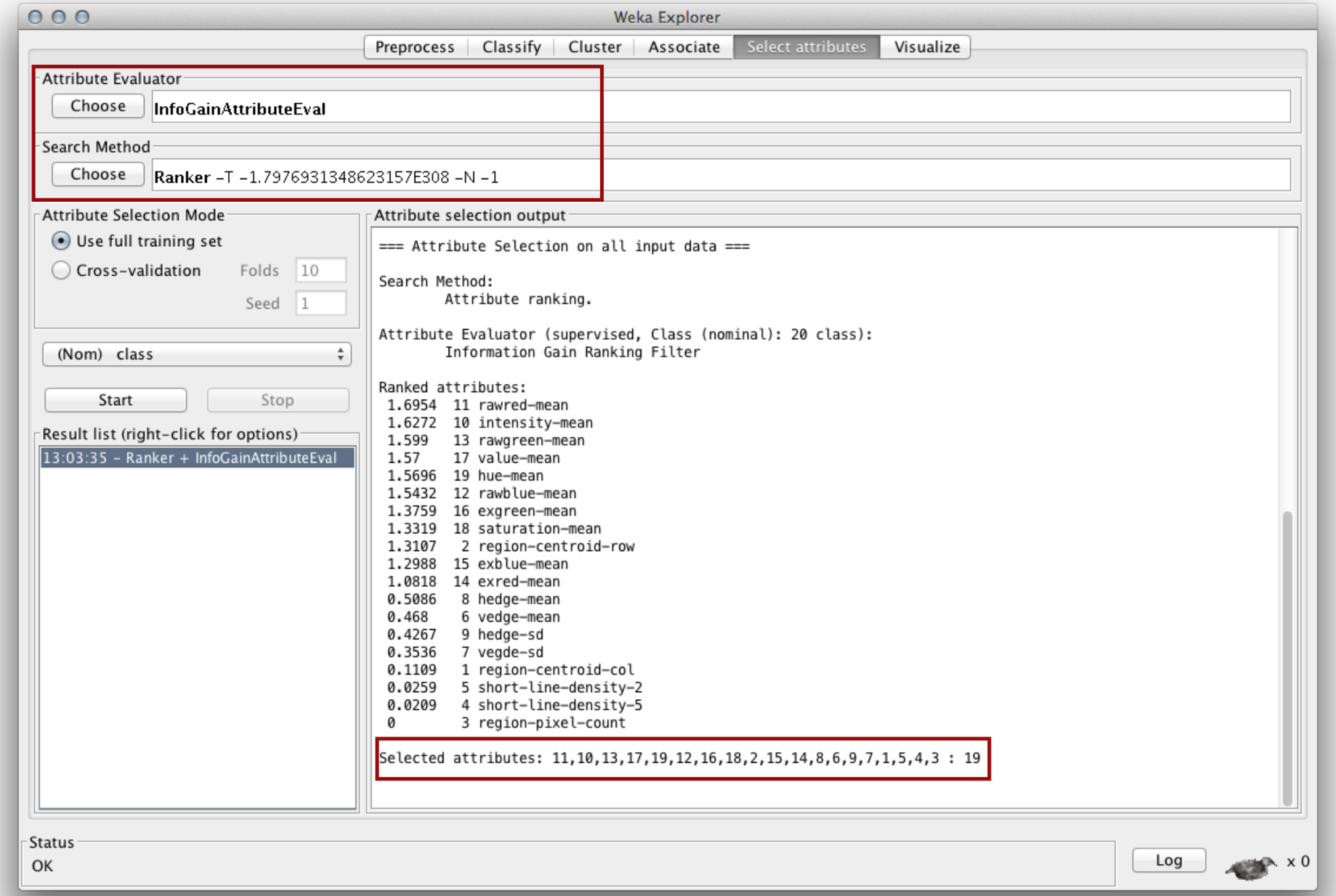
- A feature f that is predictive of a class will give significant Information Gain (i.e. a reduction in uncertainty):

$$IG(S, f) = H(S) - \sum_{v \in values(f)} \frac{|S_v|}{S} H(S_v)$$

- **IG Filter approach**: Rank all features based on IG score, select the top ranked features.
- This is the key idea from the ID3 (C4.5) algorithm for building Decision Trees.

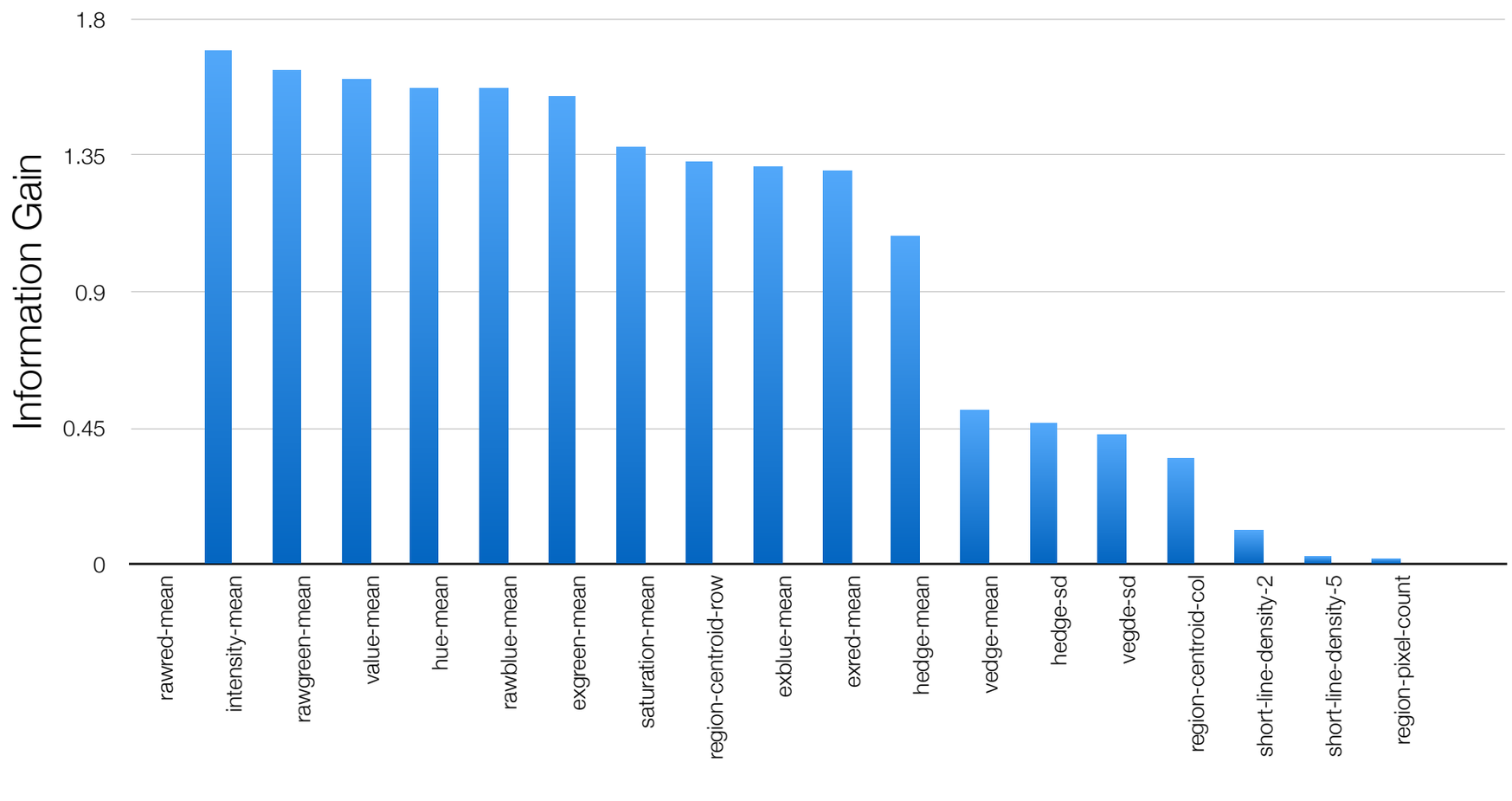
Information Gain Filter in Weka

In Weka *Select attributes*" tab, choose *InfoGainAttributeEval* as the evaluator, *Ranker* as the method.



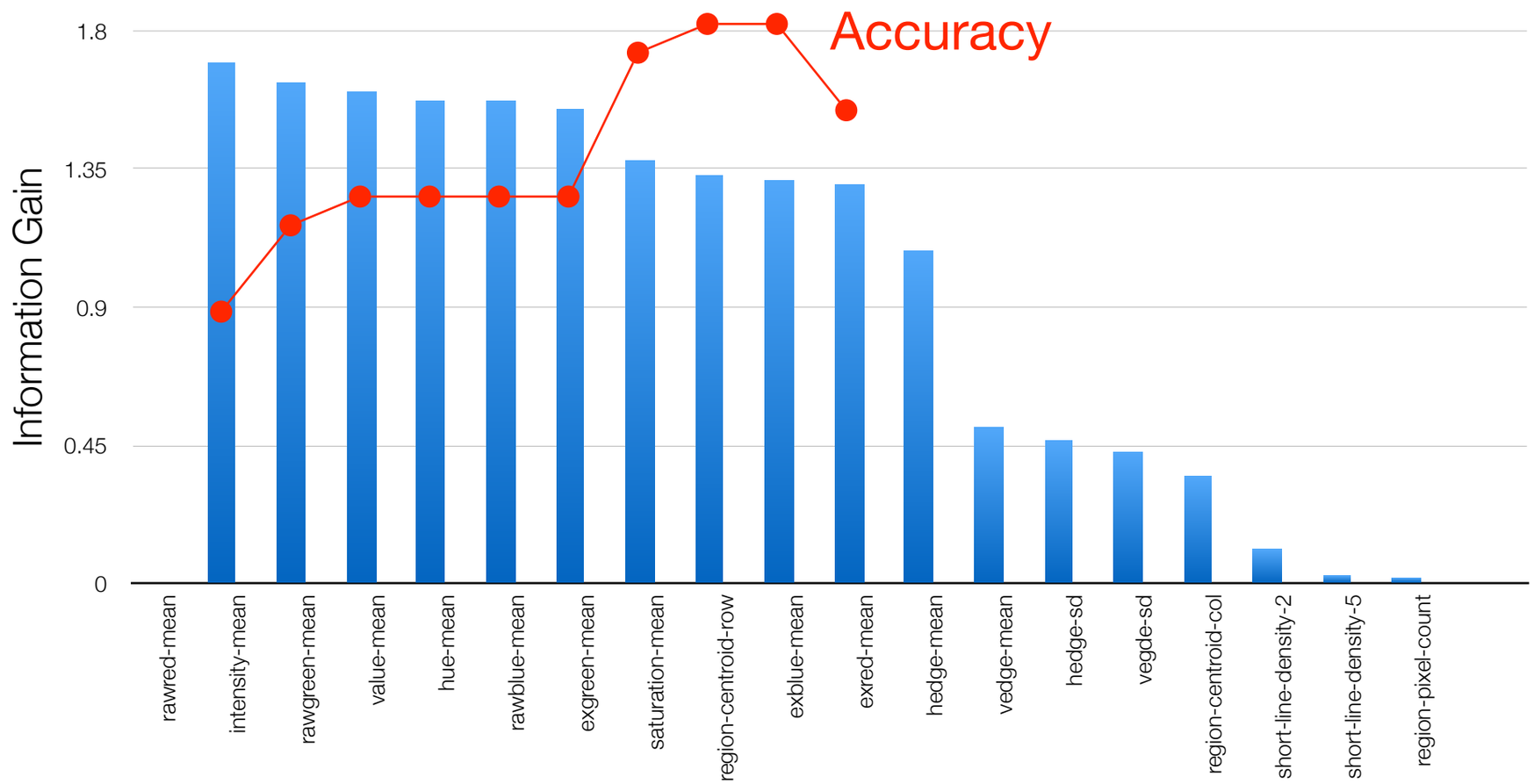
Filters - Top Features

- What to do with ranked list of features?
  - Select the top ranked k features.
  - Select top 50%.
  - Select features with IG > 50% of max IG score.
  - Subset of features with non-zero IG scores.



Filters - Top Features

- Strategy for selecting k top features - test accuracy of subsets of increasing size.
- Start with feature with highest Information Gain, add next feature.
- Test accuracy for each subset using cross-validation.

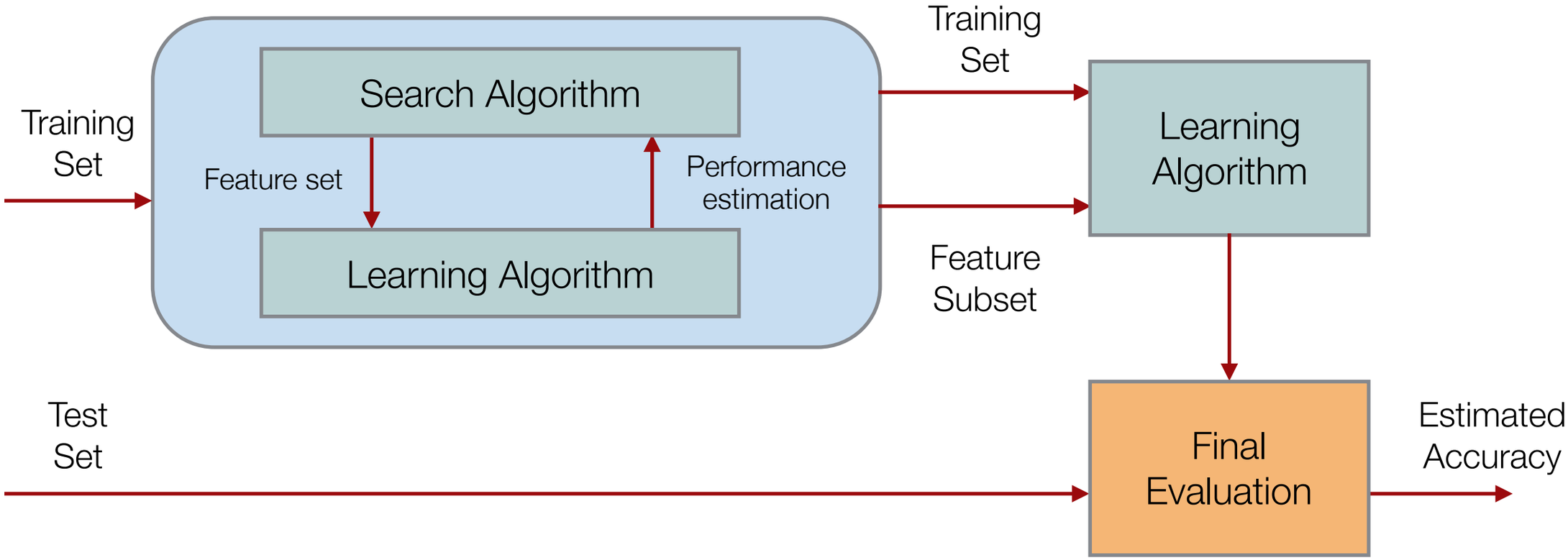


Filters

- Problems with filter feature selection approaches:
  - **Model Bias**
    - Different features may suit different learning algorithms (Neural networks, Decision Trees, K-NN, etc.).
  - **Dependencies**
    - Features are considered in isolation from one another, not considered in context.
    - In some cases, a filter might select two predictive but correlated features, where one would be sufficient.
    - In other cases, one feature needs another feature to boost accuracy.

Wrappers

- The learning algorithm (classifier) is “wrapped” in the feature selection mechanism. Feature subsets are evaluated directly based on their performance when used with that specific algorithm.
- Advantages:
  - Takes bias of specific learning algorithm into account.
  - Considers features in context - i.e. dependencies.

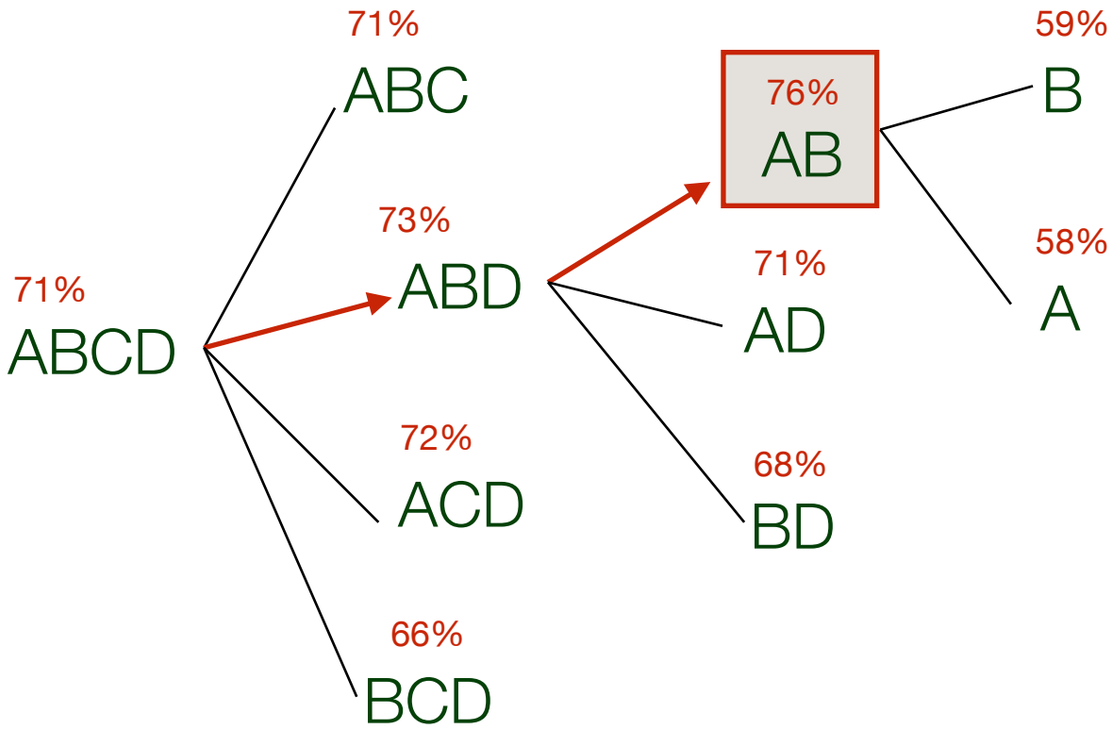


Wrappers

- Testing all features requires searching space of 2^d subsets ⇒ exhaustive search is not feasible.
- Sequential search algorithms use a heuristic stepwise approach.
  - Forward Sequential Selection
    - Start with an empty subset.
    - Find the most informative feature and add it to the subset.
    - Repeat until there is no improvement by adding features.
  - Backward Elimination
    - Start with the complete set of features.
    - Remove the least informative feature.
    - Repeat until there is no improvement by dropping features.
- Backward elimination tends to find better models - can find subsets with interacting features.
- Forward selection starts with small subsets, so cheaper if stopped early.

Wrappers

- Performance estimation (classification accuracy) guides the sequential search process.
- For robustness, performance estimation should use k-fold cross validation.
- Requires retraining the model main different times, can be very time consuming.
- Wrappers usually far slower than filters, but trying to solve the “real problem”.



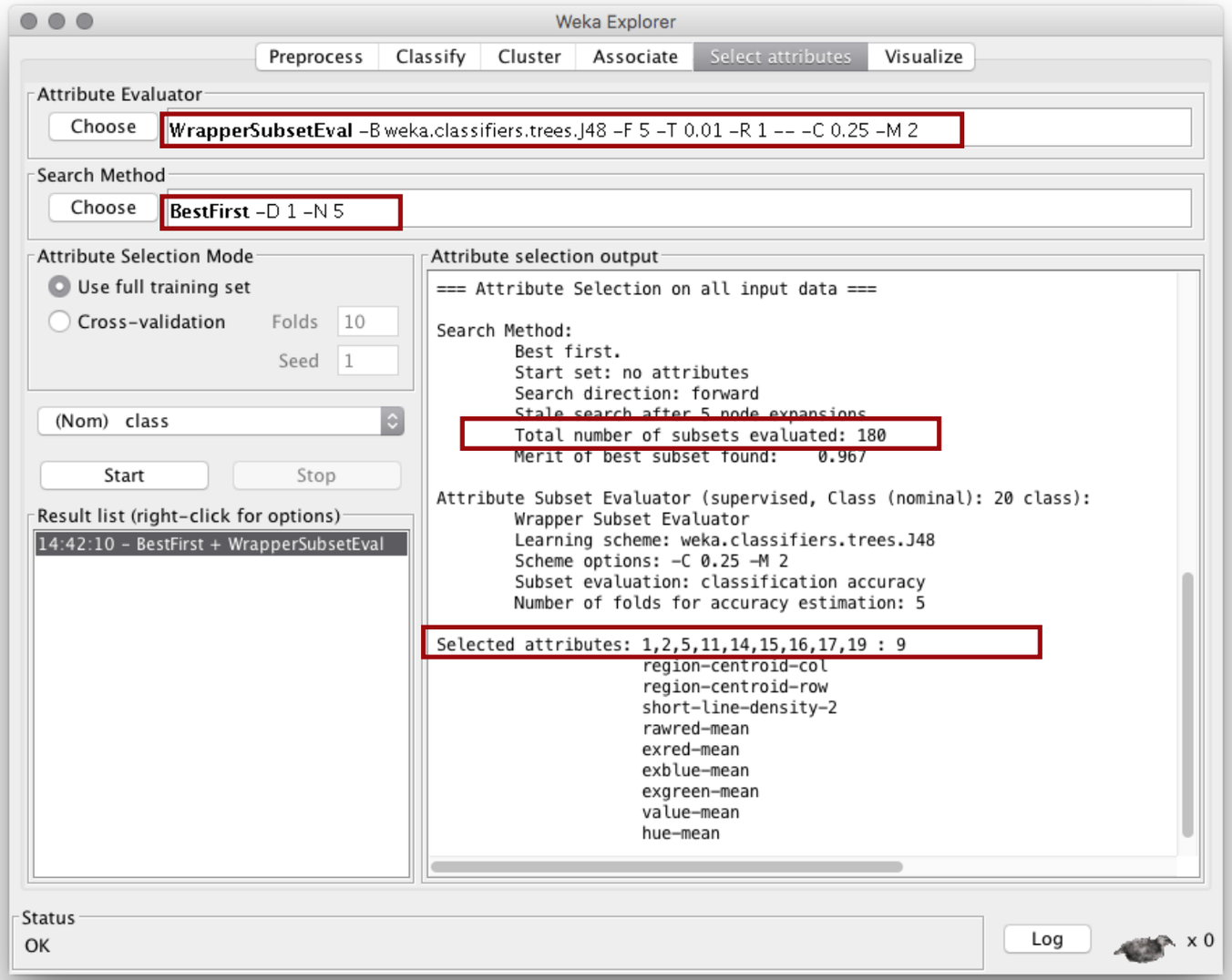
Example: Backward elimination from 4 features (ABCD) to 2 features (AB).

Wrappers in Weka

- For Forward Selection: In *Select attributes* tab, choose *WrapperSubsetEval* as the evaluator with J48 as classifier. Choose *BestFirst* as the search method, change direction to *Forward* (1).

J48: C4.5 Decision Tree algorithm

BestFirst D=1: Forward selection



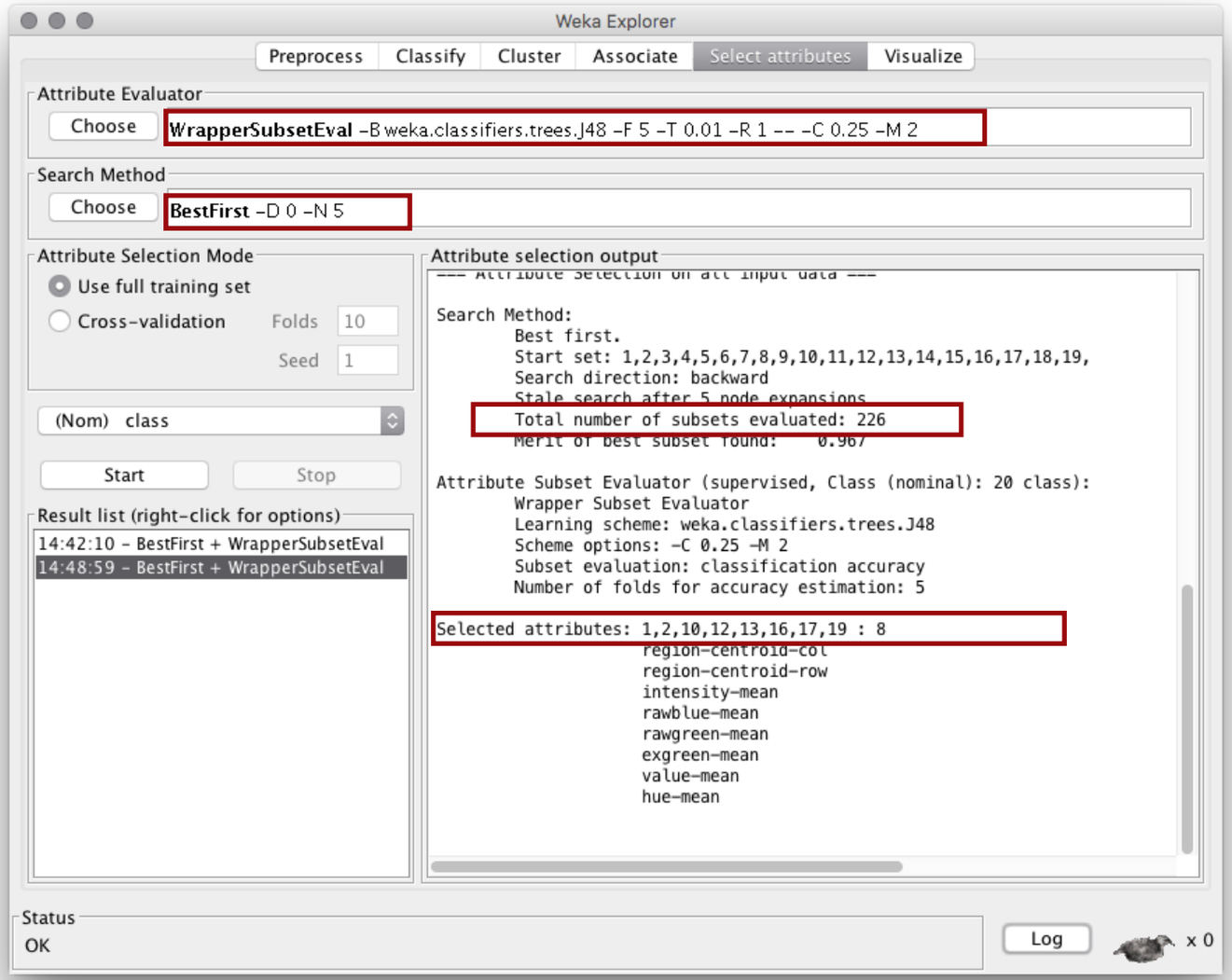
Selected 9/19 features after checking 180 subsets

Example: segment.arff

Wrappers in Weka

- For Backward Elimination: In *Select attributes* tab, choose *WrapperSubsetEval* as the evaluator with J48 as classifier. Change options for search direction to *Backward* (0).

J48: C4.5 Decision Tree algorithm



BestFirst D=0: Backward elimination

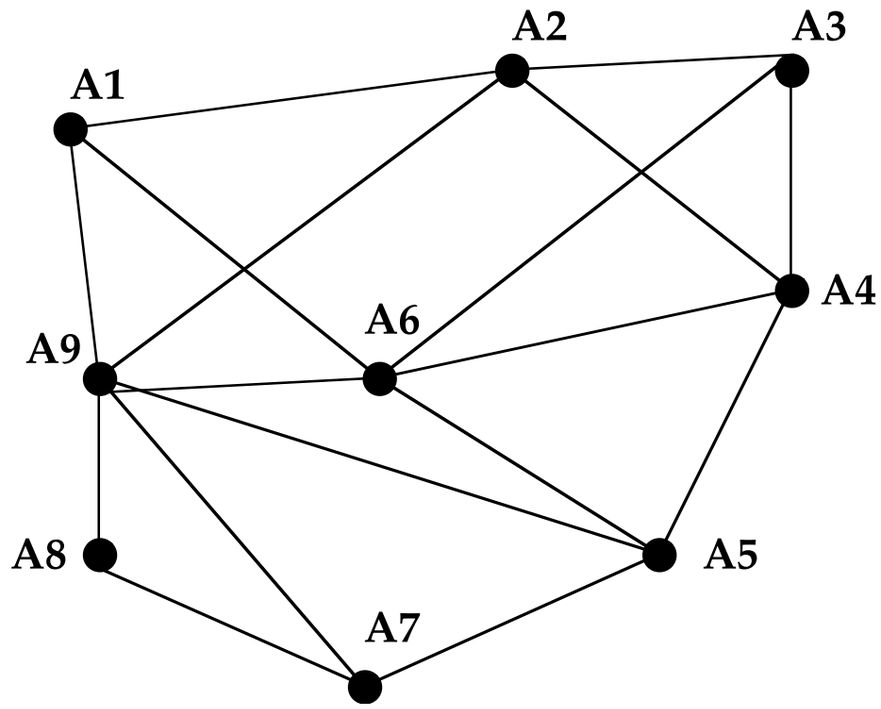
Selected 8/19 features after checking 226 subsets

Example: segment.arff

Stochastic Search

- Stochastic search: methods for solving many hard combinatorial problems (e.g. feature subset selection, finding shortest tours).
  - Evolutionary Algorithms (Genetic Algorithms), Simulated Annealing, Tabu Search, Ant Colony Optimisation…

- **Travelling Salesman Problem (TSP)**
  - Classic NP-hard test-bed for optimisation algorithms.
  - Given *n* cities and the distances between them, what is the shortest possible tour that visits each city exactly once and returns to the starting city?



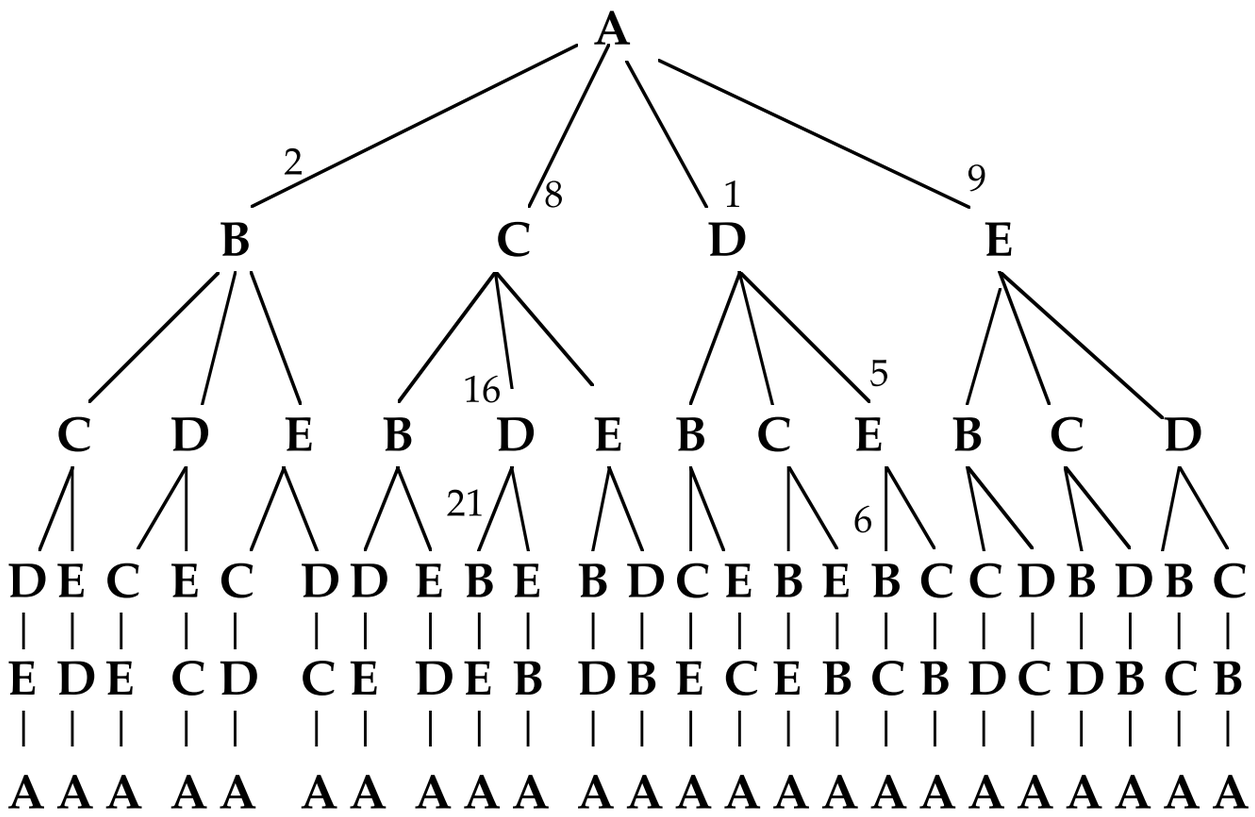
http://en.wikipedia.org/wiki/Travelling_salesman_problem

Travelling Salesman Problem

Q. Why are these problems computationally hard?

- **Example:** Consider simple problem with *n=5* cities…

	A	B	C	D	E
A	-	2	8	1	9
B	2	-	3	5	1
C	8	3	-	8	7
D	1	5	8	-	4
E	9	1	7	4	-



➡ Simple problem has *4x3x2x1 = 24 = (n-1)!* solutions

Travelling Salesman Problem

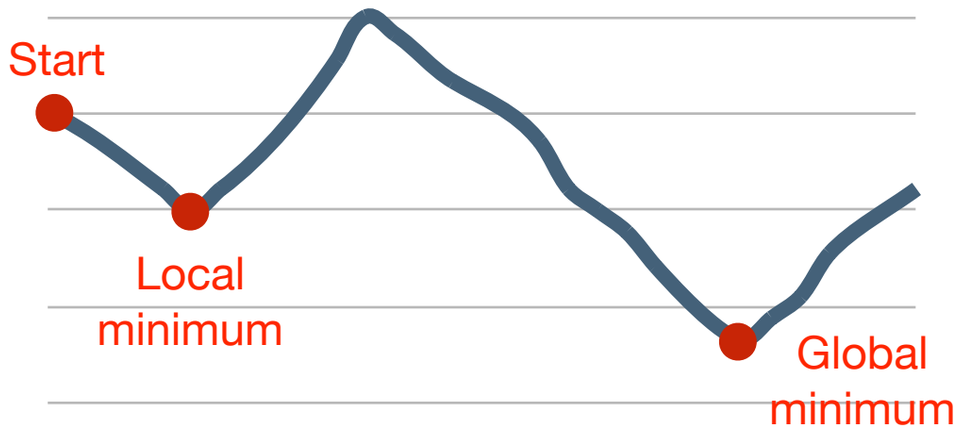
- **Heuristic Algorithm for TSP:**

  1. Generate a random feasible solution *D*
     (i.e. one possible tour among the *n* cities).
  2. Attempt to find an improved feasible solution *D*' by applying some transformation to *D*
     (e.g. reverse a random section of the tour).
  3. If an improved solution is found, then replace *D* by *D*' and repeat from Step 2.
  4. If no improved solution is found then *D* is a local optimum solution. Repeat from Step 1 until time runs out.

- This is a greedy search technique ⇒ the search may not find the global minimum solution.

Genetic Algorithms

- In many problems, the search space may be so large that global optimum cannot be found in a reasonable time.
- Sequential methods can easily get stuck in local minima, especially for more complex problems.



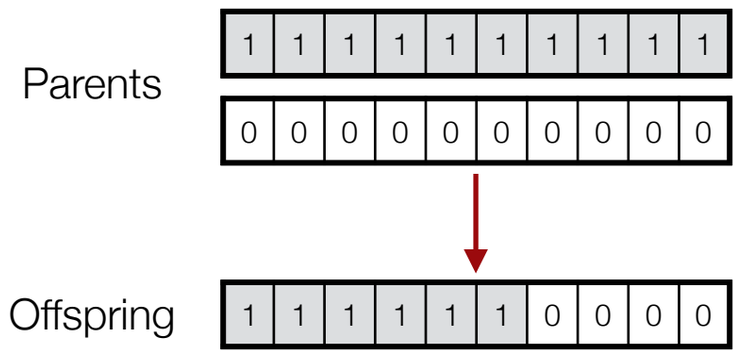
- Genetic Algorithms (GA)
  - Search heuristic that mimics the process of natural selection - a *population* of candidate solutions is evolved toward better solution.
  - Candidates are encoded as a *chromosome* which can be evaluated according to some *fitness function*.
  - Algorithm runs over multiple *generations*. Fitter candidates are more likely to survive to the next generation.
  - Evolution occurs via one or more operators that transform the candidate chromosome - e.g. crossover, mutation.

GA Feature Selection

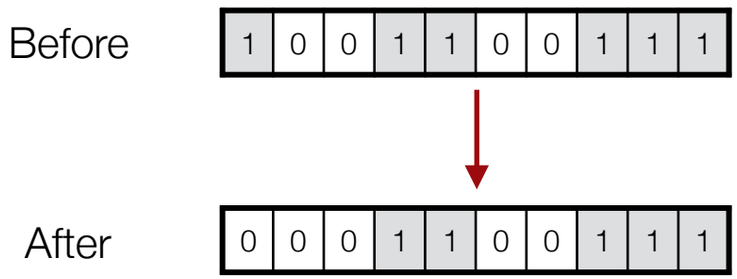
- A genetic algorithm can be used to guide the search process for wrapper feature selection.
  - *Population*: Different candidate feature subsets.
  - *Chromosome*: Bit mask encodes a feature subset, one bit per feature.
  - *Fitness function*: Classification accuracy using a given classifier on each subset. May also consider number of features and diversity.
  - *Operators*: Crossover and mutation (i.e. functions that change which features are present/missing in the subsets).

Encoding of subset of 6/10 features

Crossover Operator



Mutation Operator



10% mutation rate

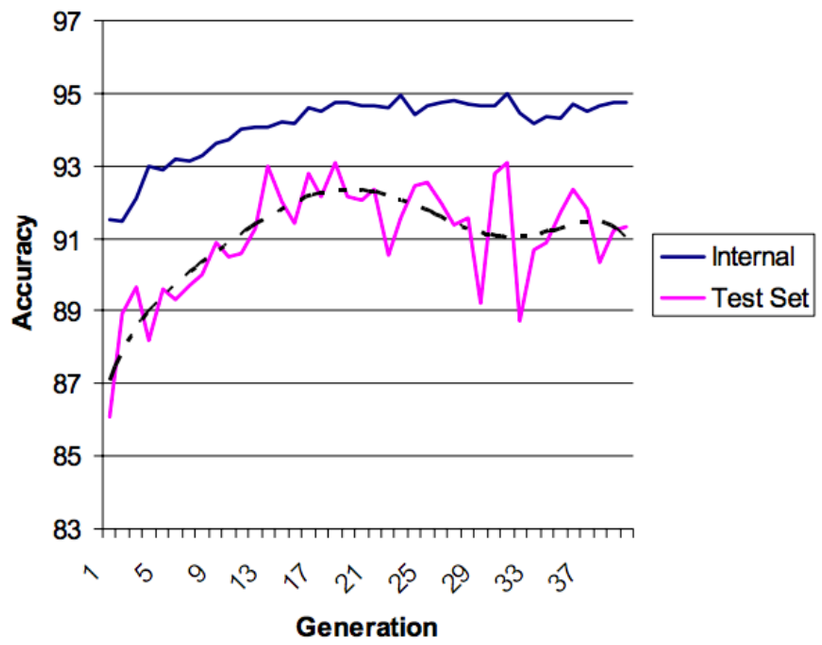
Overfitting in Feature Selection

- Reminder: A classifier is said to overfit to a data set if it models the training data too closely, leading to poor predictions when applied to unseen data.
- Wrappers can be prone to overfitting when the chosen feature subsets are used to build classifiers on unseen data.

➡ We simply found the best feature subset for the training data!

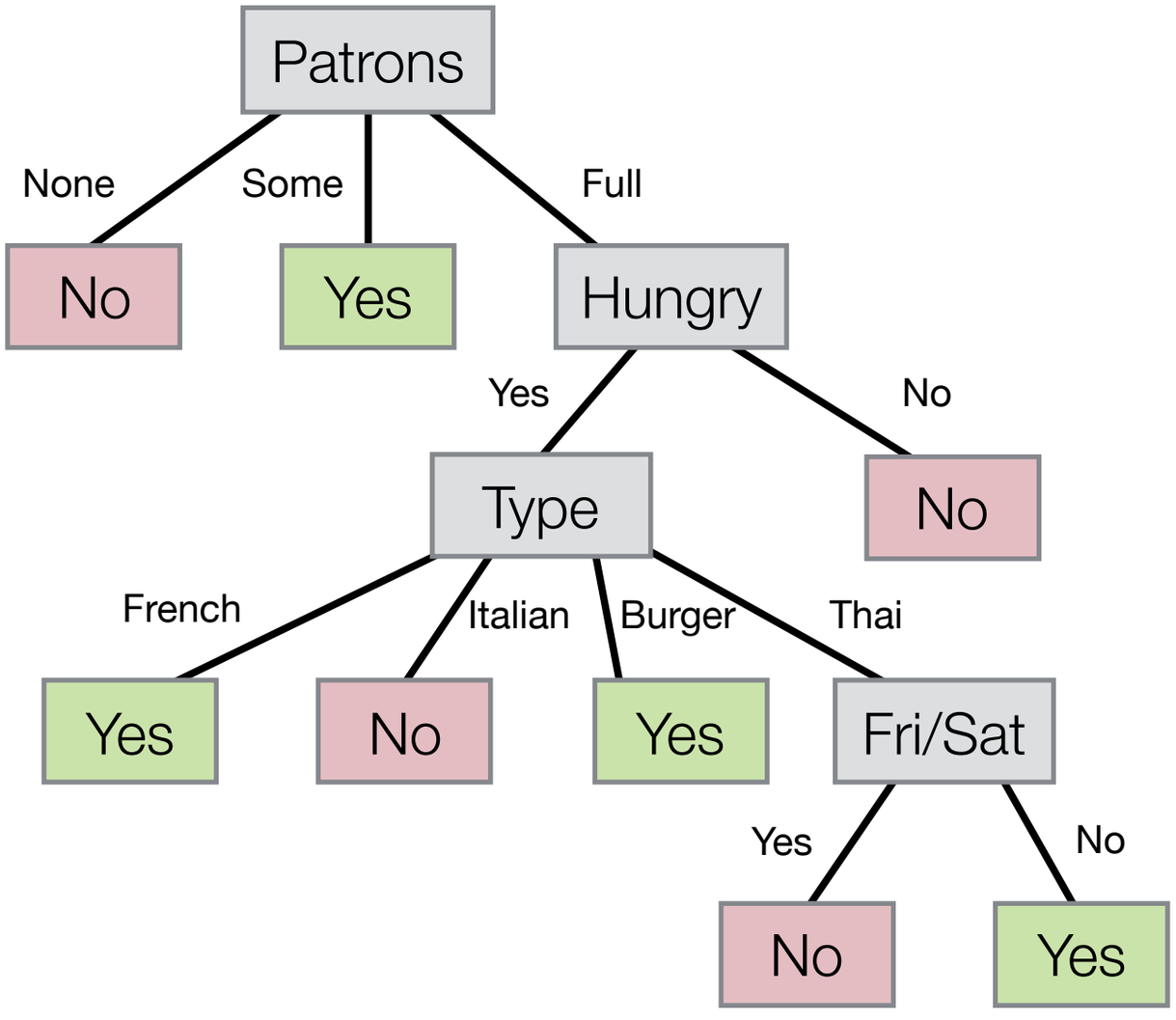
"Overfitting in feature selection appears to be exacerbated by the intensity of the search, since the more feature subsets that are visited the more likely the search is to find a subset that overfits"
- Loughrey & Cunningham, 2004

Example: Feature Selection with GA Search



Embedded Feature Selection

- Some methods do not even look like feature selection as they are “embedded” directly into the learning algorithm.
- Example: Decision Trees have a feature selection mechanism as an integral part of their core operation.



ID3 algorithm applies a feature selection + splitting process until all examples have the same class, or no features are left to split.

Summary

- Dimension Reduction
  - The Curse of Dimensionality
  - Feature Transformation v Selection
- Feature Selection in Supervised Learning
  - Filter v Wrapper Strategies
  - Filter approaches
  - Wrapper approaches
  - Stochastic search
  - Overfitting in feature selection

References

- R. Bellman (1961). “Adaptive control processes: a guided tour”. Princeton University Press.
- E. Alpaydin. "Introduction to Machine Learning", Adaptive Computation and Machine Learning series, MIT press, 2009.
- P. Flach. “Machine Learning: The Art and Science of Algorithms that Make Sense of Data”. Cambridge University Press, 2012.
- K. Kira, L. Rendell. “A practical approach to feature selection”. Proc 9th *i*nternational workshop on Machine Learning, 1992.
- T. Mitchell. “Machine Learning”. McGraw-Hill, 1997.
- L. Zhuo et al. “A genetic algorithm based wrapper feature selection method for classification of hyperspectral images using support vector machine”, Geoinformatics 2008.
- J. Loughrey, P. Cunningham. “Overfitting in Wrapper-Based Feature Subset Selection: The Harder You Try the Worse it Gets”. SGAI 2004