

COMP47460

Dimension Reduction

Aonghus Lawlor
Deepak Anjwani

School of Computer Science
Autumn 2019

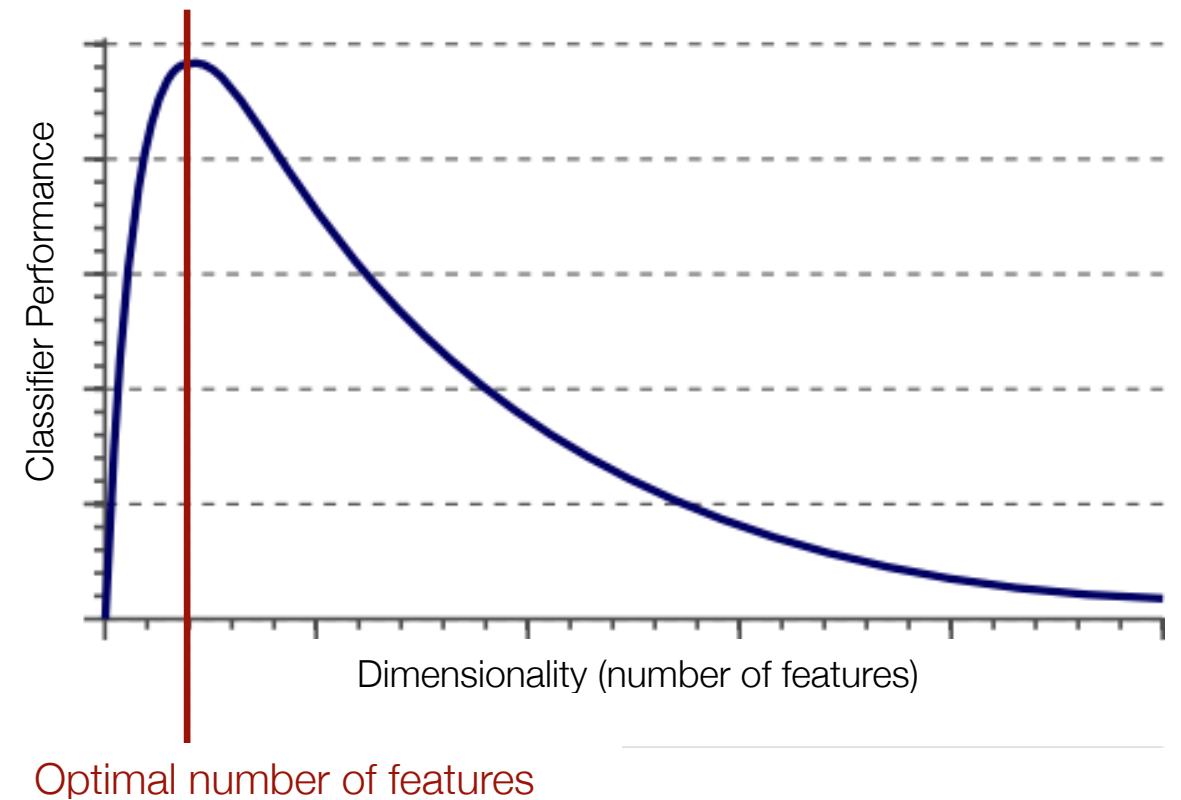


Overview

- The Curse of Dimensionality
- Dimension Reduction
- Feature Transformation v Selection
- Feature Selection in Supervised Learning
 - Filter Methods
 - Wrapper Methods
- Feature Transformation
 - Linear Transformations
 - Projection Methods
 - Principal Component Analysis (PCA)
 - PCA in Weka

Curse of Dimensionality

- Intuitively adding more features (dimensions) to a dataset should provide more information about each example, making prediction easier.
- In reality, we often reach a point where adding more features no longer helps, or can even reduce predictive power.
- **Curse of dimensionality**: the phenomenon whereby many machine learning algorithms can perform poorly on high-dimensional data (Bellman, 1961).
- In practice, to build a model, the number of examples required per feature increases exponentially with number of features.

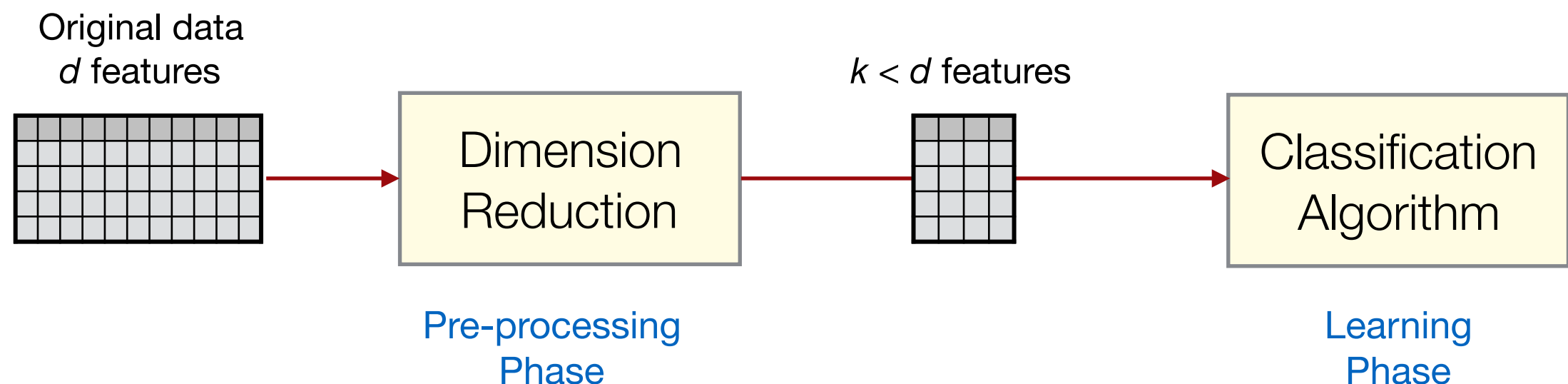


Dimension Reduction

- There are often other reasons why we might want to reduce the number of features/dimensions used to represent data:
 1. **Computational cost:** For many algorithms, having a large number of features can significantly increase running times and memory usage.
 2. **Financial cost:** In certain domains (e.g. clinical medicine, manufacturing), running experiments to generate feature values can be expensive. In such cases, we only want to generate the minimum number of features required to build a good model.
 3. **Interpretability:** A feature set which is more compact can help to give a better understanding of the underlying process that generated the data.
- ➔ Understanding which features in our data are informative and which are not is an important knowledge discovery task.

Dimension Reduction

- Basic idea to try to beat the curse of dimensionality:
 1. Apply pre-processing techniques to reduce the number of features used to represent a dataset.
 2. Then build a model on the smaller feature set.
- In many (but not all) cases, the additional information that is lost by removing some features is (more than) compensated by higher classifier accuracy in the lower dimensional space.



Feature Transformation v Selection

There are two general strategies for dimension reduction:

1. Feature Transformation (Feature Extraction)

Transforms the original features of a dataset to a completely new, smaller, more compact feature set, while retaining as much information as possible.

e.g. Principal Components Analysis (PCA), Linear Discriminant Analysis (LDA)

2. 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.

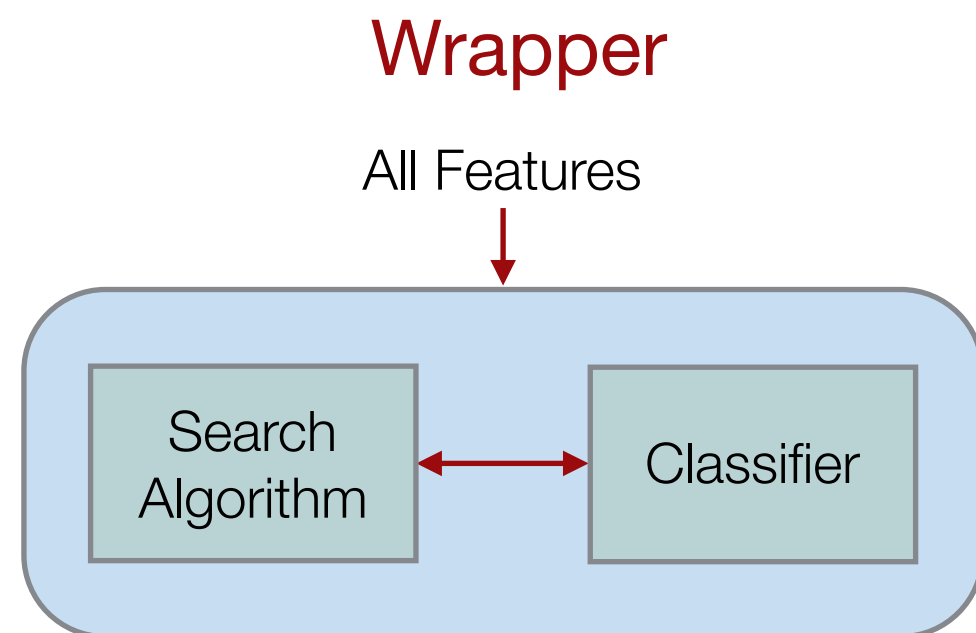
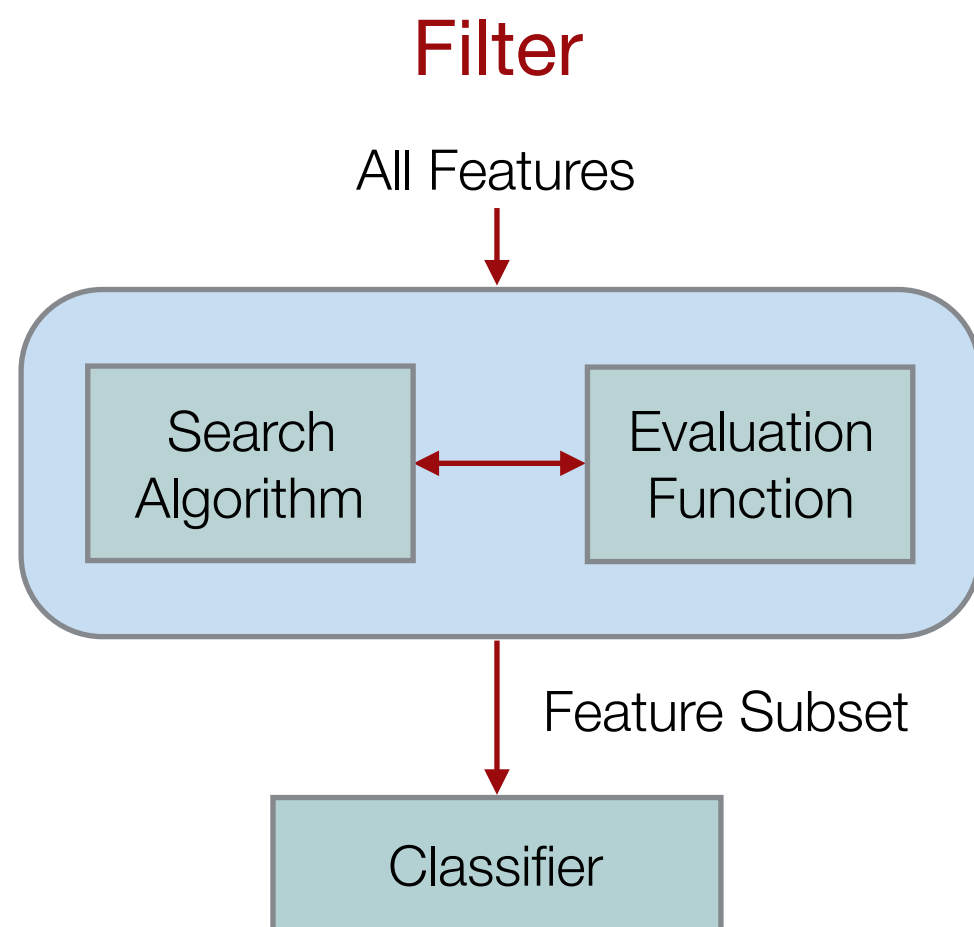
e.g. Information Gain filter, Wrapper with sequential forward selection

Feature Selection

- **Feature Subset Selection:** Find the best subset of all available features, which contains the smallest number of features that most contribute to accuracy. Discard the remaining, unimportant features.
- **Why select subset of original features?**
 1. Building a better classifier - Redundant or noisy features can damage accuracy.
 2. Knowledge discovery - Identifying useful features helps us learn more about the data.
 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

- Finding the optimal feature subset for a given dataset is difficult.
- 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:



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).

Information Gain Filter

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

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

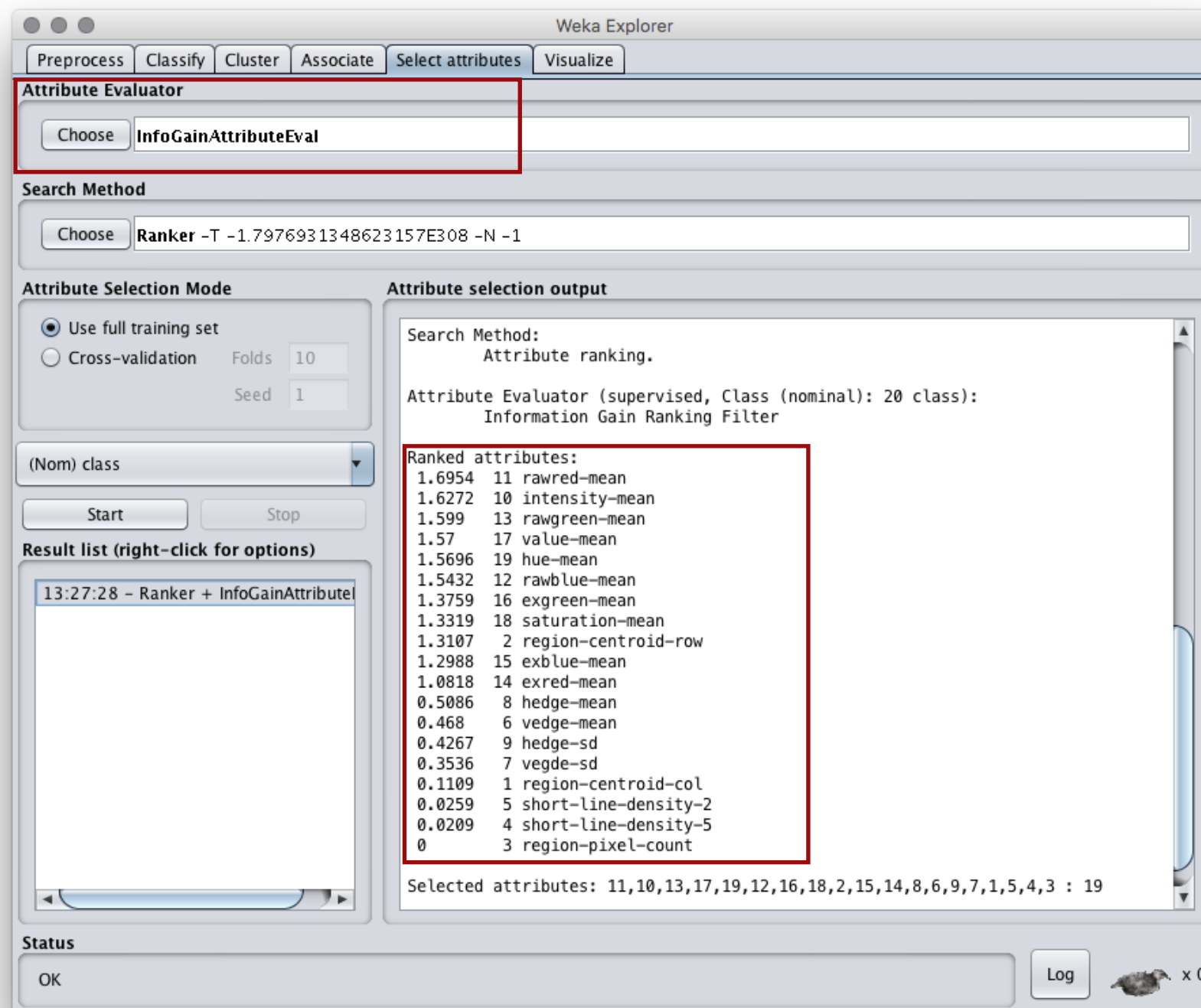
- A feature f that is predictive of a class will give significant Information Gain (i.e. a reduction in uncertainty):

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

- IG Filter approach:**
 - Score all features based on their Information Gain (IG).
 - Rank features based on their scores.
 - Select a subset of the top ranked features to use for classification.

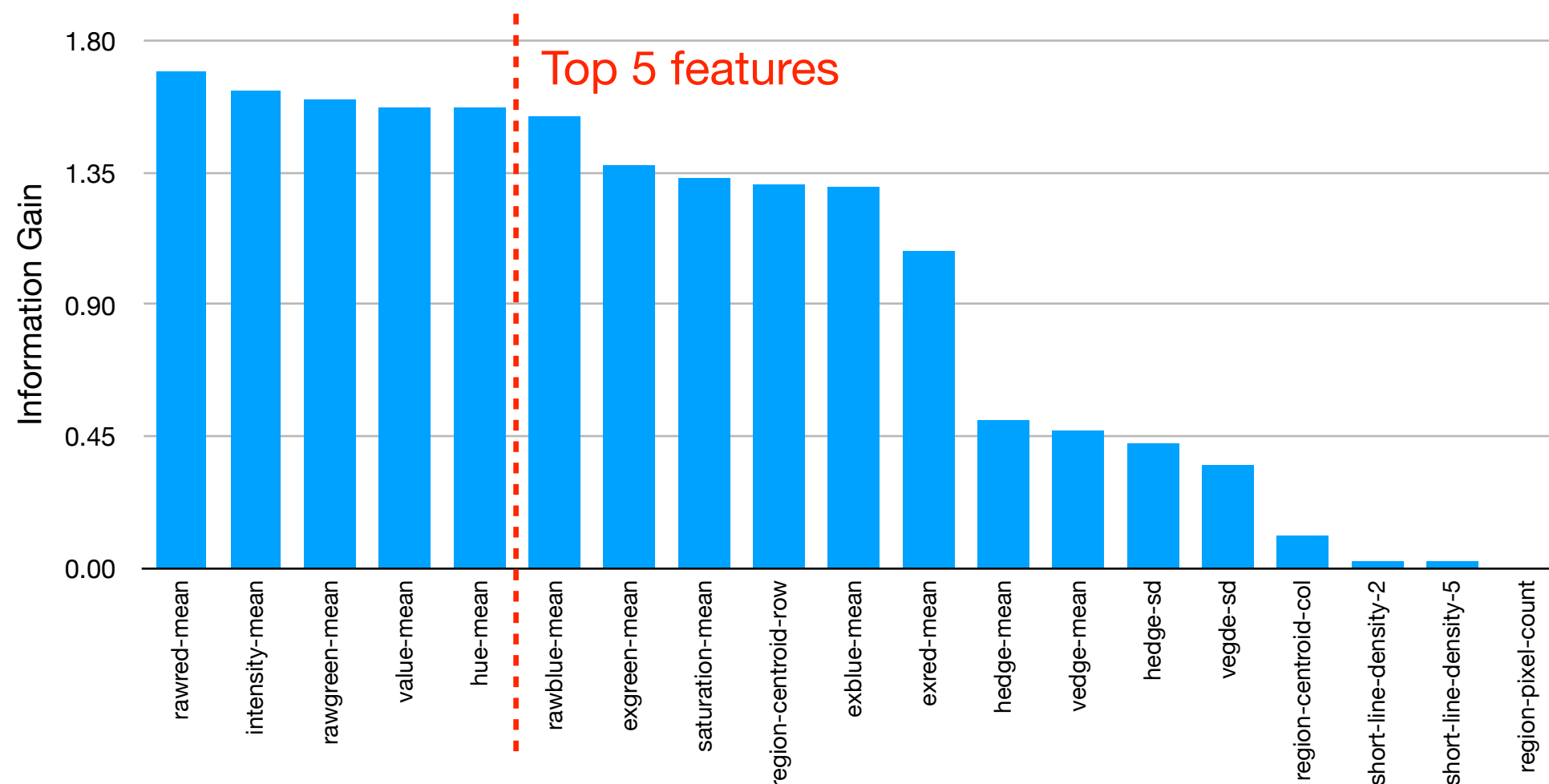
Information Gain Filter in Weka

In Weka *Select attributes* tab, choose *InfoGainAttributeEval* as the evaluator, *Ranker* as the method, and hit *Start*.



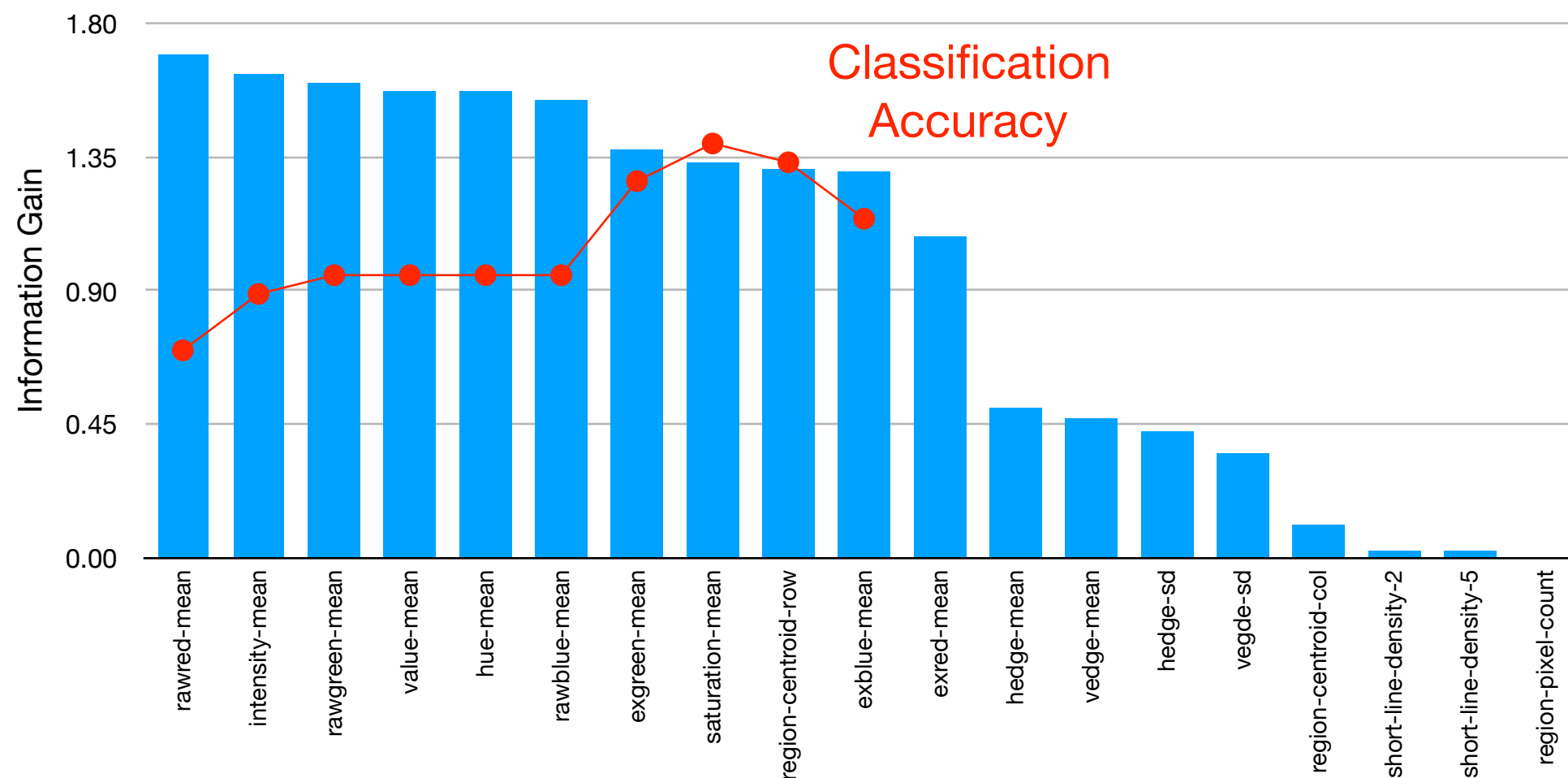
Filters - Top Features

- What to do with ranked list of features? Several basic options...
 - 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

- Better strategy for selecting k top features: evaluate classification performance using feature subsets of increasing size.
- Start with feature with highest Information Gain, then add the next feature. Measure the accuracy for each subset using cross-validation. Choose the final subset giving the highest accuracy.



Filters - Disadvantages

Key problems with filter feature selection:

1. No Model Bias

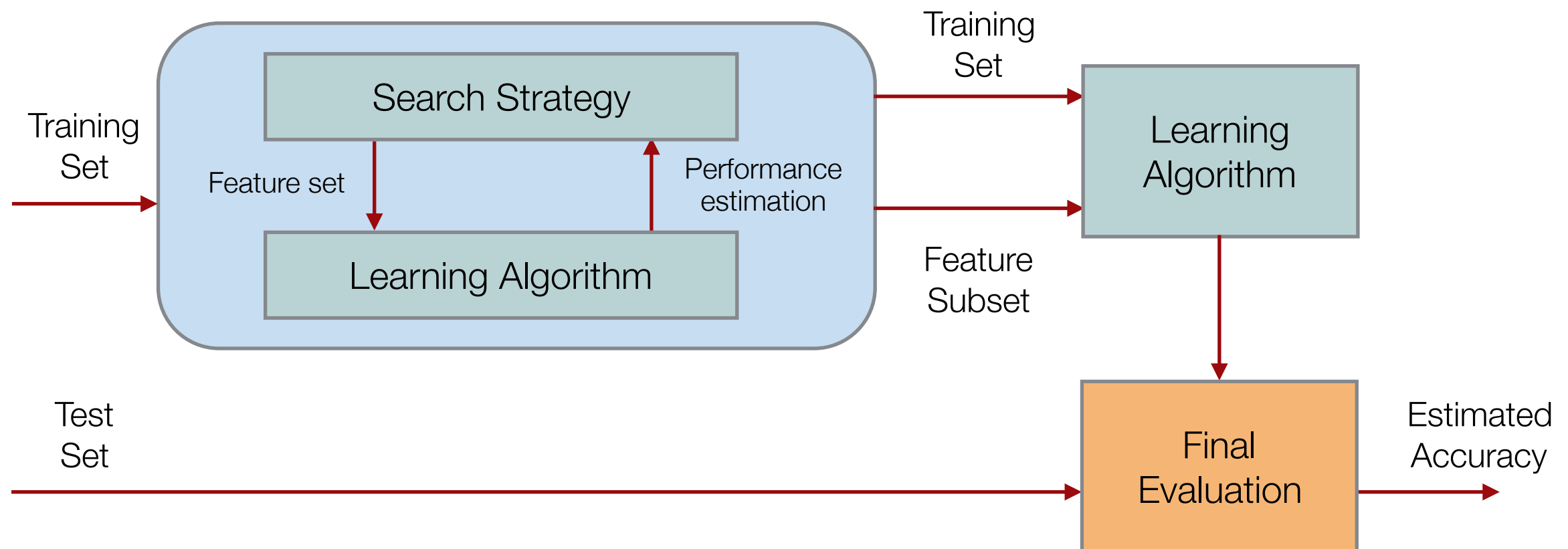
- Different features may suit different learning algorithms (Neural networks, Decision Trees, K-NN, etc.), but filters do not take this into account.

2. No Feature Dependencies

- In filters, the features are considered in isolation from one another, and are 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, but a filter cannot discover this.

Wrappers

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



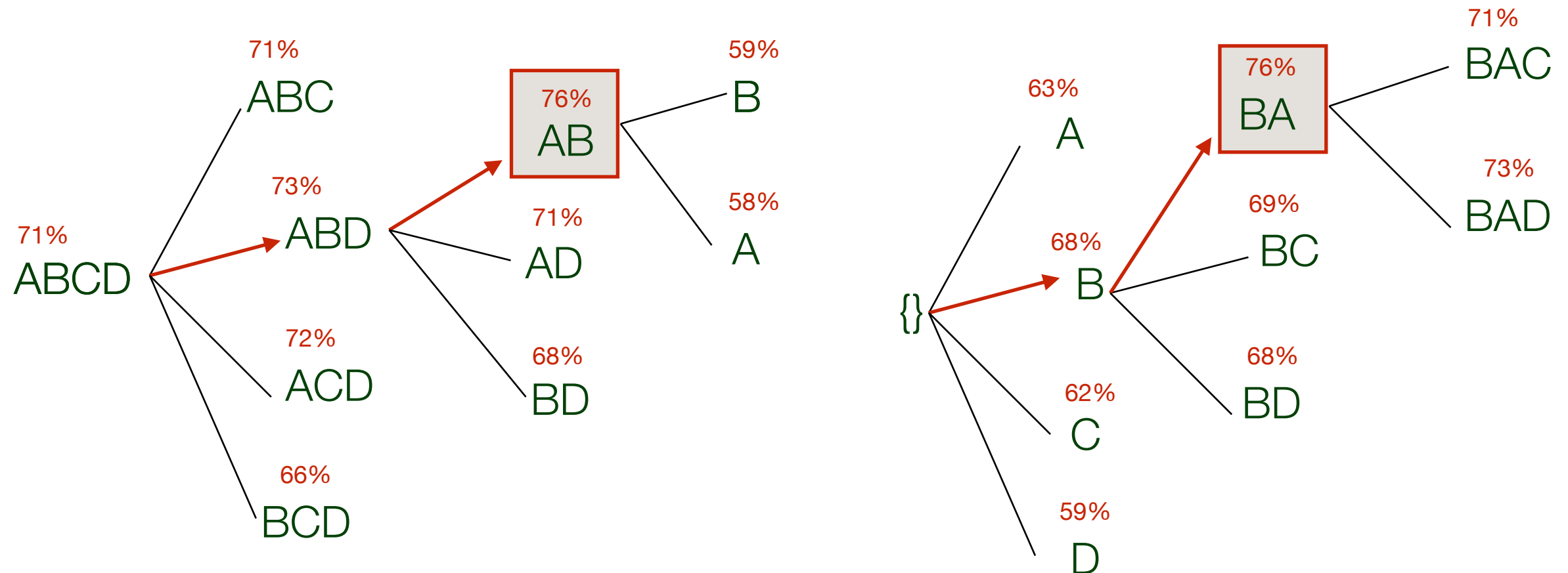
Feature Subset Search

- A key aspect of wrappers is the search strategy which generates the candidate feature subsets for evaluation.
- There are many different ways to search the space of all possible subsets. The choice of a suitable search strategy often depends on the number of features d in the data. Several general approaches:
 - **Exhaustive search**: Evaluate every possible subset of features. For d features there are 2^d potential subsets. For even small values of d , an exhaustive search over this huge space is intractable.
 - **Exponential search**: Returns the optimal feature subset, but uses heuristics so that not every one of the 2^d possible subsets needs to be evaluated.
 - **Sequential search**: Fast search algorithms that choose a subset by adding or removing one feature at a time. Not guaranteed to find the optimal feature subset, but widely used.

Sequential Search

- In sequential search, performance estimation guides the subset search process (e.g. overall classification accuracy).
- The most basic sequential search methods use a heuristic stepwise approach, either:
 - **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.

Sequential Search



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

Example: Forward selection from no features to 2 features (BA).

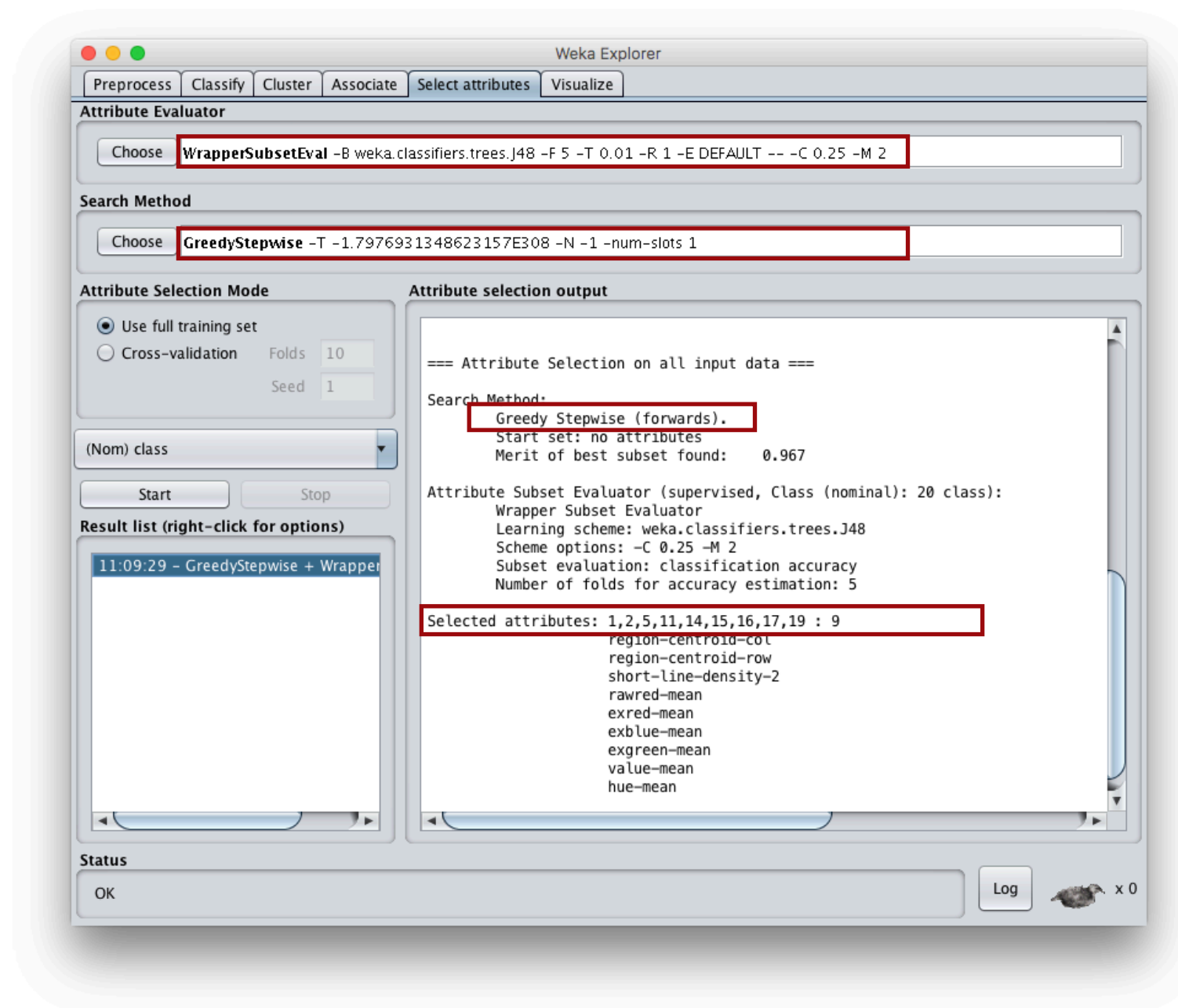
- Backward elimination tends to find better models, can find subsets with interacting features. But tends to be slower.
- Forward selection starts with small subsets, so less require less running time if stopped early.

Wrappers in Weka - Forward Selection

- In *Select attributes* tab, choose *WrapperSubsetEval* as the evaluator with J48 as classifier. Choose *GreedyStepwise* as the search method, with direction *SearchBackwards* = *False*.

Example:
segment.arff
19 features

Greedy Stepwise
(Forwards)
+
J48 Decision
Tree Classifier



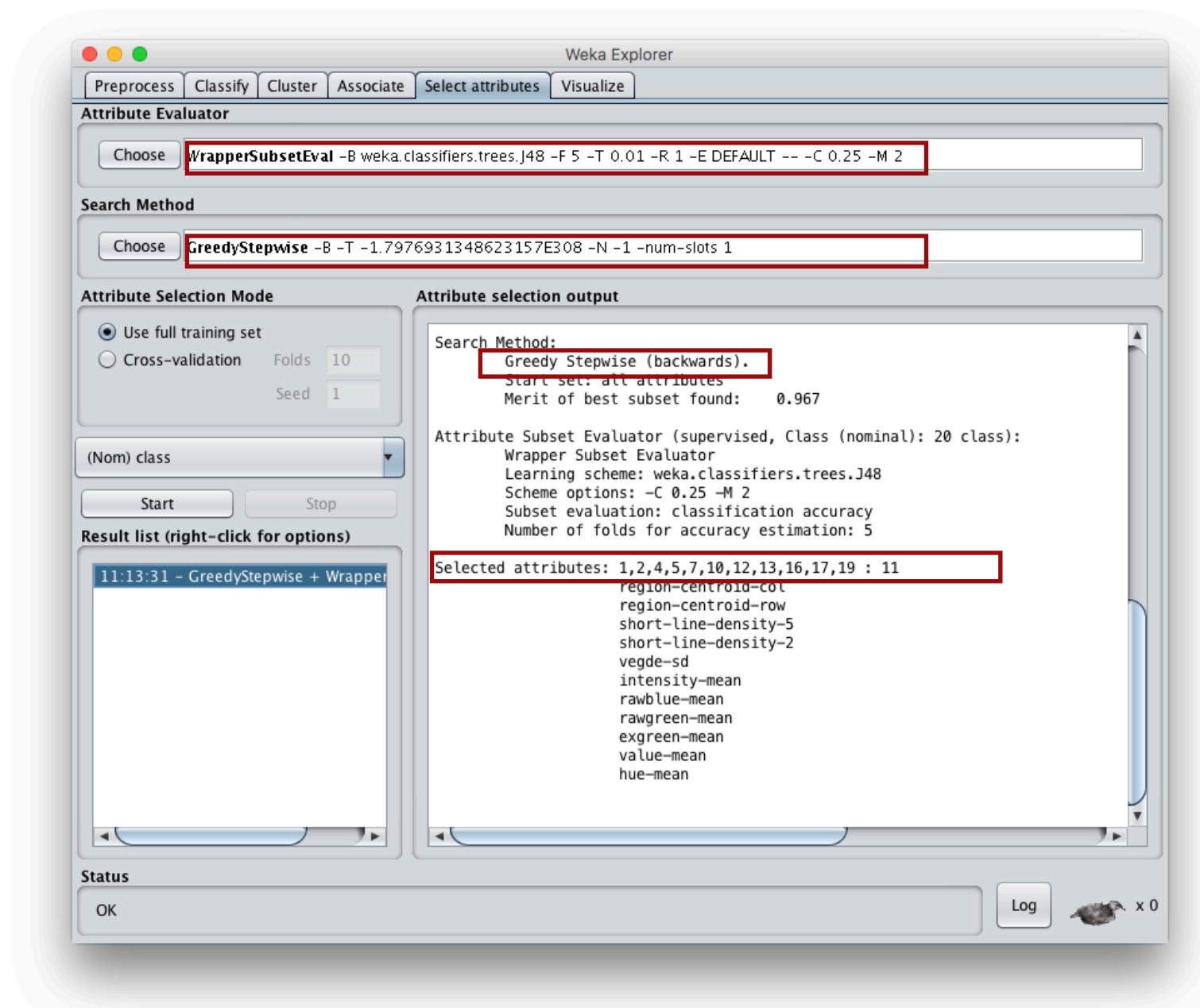
Result:
Selected 9/19
features

Wrappers in Weka - Backward Elimination

- In *Select attributes* tab, choose *WrapperSubsetEval* as the evaluator with J48 as classifier. Choose *GreedyStepwise* as the search method, with direction *SearchBackwards = True*.

Example:
segment.arff
19 features

Greedy Stepwise
(Backwards)
+
J48 Decision
Tree Classifier



Result:
Selected 11/19
features

Wrappers - Disadvantages

There are two main disadvantages with wrapper methods:

1. Computational Cost

- Significant running time, particularly when the number of features is large. Far slower than filters.
- Computational bottleneck is the requirement to retrain the model many different times on different feature subsets.

2. Overfitting

- Risk of overfitting, particularly when the number of training examples is insufficient.
- We can end up finding the best feature subset for the training data, which does not work well for new unseen data.

Feature Selection v Transformation

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 Transformation (Feature Extraction)

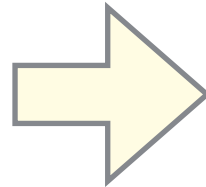
- A popular alternative strategy in data pre-processing is to transform the data into an entirely different format.
- Examples represented by one set of features are transformed to another new set of features.
- Resulting features can be more compact and less noisy, resulting in more accurate predictions.
- Typically involve a linear transformation of the original data.

Feature Selection v Transformation

	<i>region-centroid-col</i>	<i>region-centroid-row</i>	<i>region-pixel-count</i>	<i>vedge-mean</i>	<i>vegde-sd</i>	<i>hedge-mean</i>	<i>hedge-sd</i>	<i>intensity-mean</i>
x1	218.0	178.0	9.0	0.8	0.5	1.1	0.5	59.6
x2	113.0	130.0	9.0	0.3	0.3	0.3	0.4	0.9
x3	202.0	41.0	9.0	0.9	0.8	1.1	1.0	123.0
x4	32.0	173.0	9.0	1.7	1.8	9.0	6.7	43.6
x5	61.0	197.0	9.0	1.4	1.5	2.6	1.9	49.6
x6	149.0	185.0	9.0	1.6	1.1	3.1	1.9	49.3
x7	197.0	229.0	9.0	1.4	1.6	1.2	0.6	17.7
x8	29.0	111.0	9.0	0.4	0.2	0.6	0.2	5.4
...

Feature Selection:

Select subset of the original features, use them to represent the data.

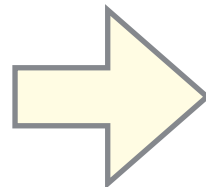


	<i>region-centroid-row</i>	<i>hedge-mean</i>	<i>hedge-sd</i>	<i>intensity-mean</i>
x1	178.0	1.1	0.5	59.6
x2	130.0	0.3	0.4	0.9
x3	41.0	1.1	1.0	123.0
x4	173.0	9.0	6.7	43.6
x5	197.0	2.6	1.9	49.6
x6	185.0	3.1	1.9	49.3
x7	229.0	1.2	0.6	17.7
x8	111.0	0.6	0.2	5.4
...

Feature

Transformation:

Create a new set of dimensions, use them to represent the data.

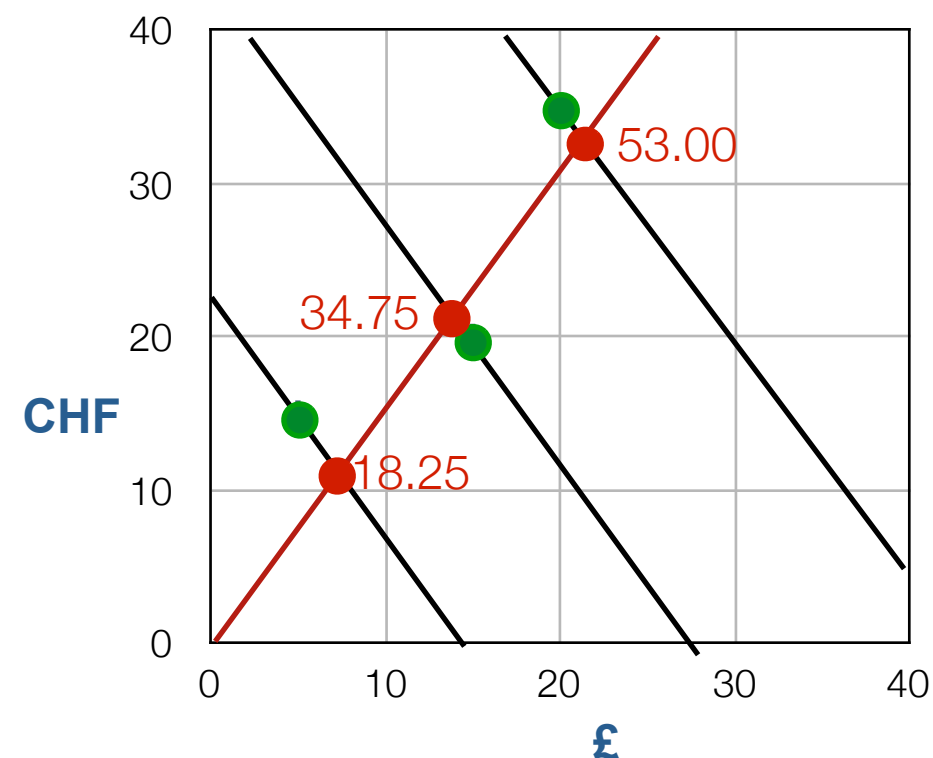


	<i>PC1</i>	<i>PC2</i>	<i>PC3</i>	<i>PC4</i>
x1	95.04	19.51	36.68	37.97
x2	-13.34	11.55	19.53	-29.57
x3	76.50	-2.37	-112.33	41.45
x4	-90.95	5.04	44.07	29.96
x5	-61.17	13.49	61.95	43.10
x6	26.18	16.23	49.07	34.47
x7	74.49	26.46	100.29	21.20
x8	-97.65	5.20	2.54	-29.51
...

Linear Transformations

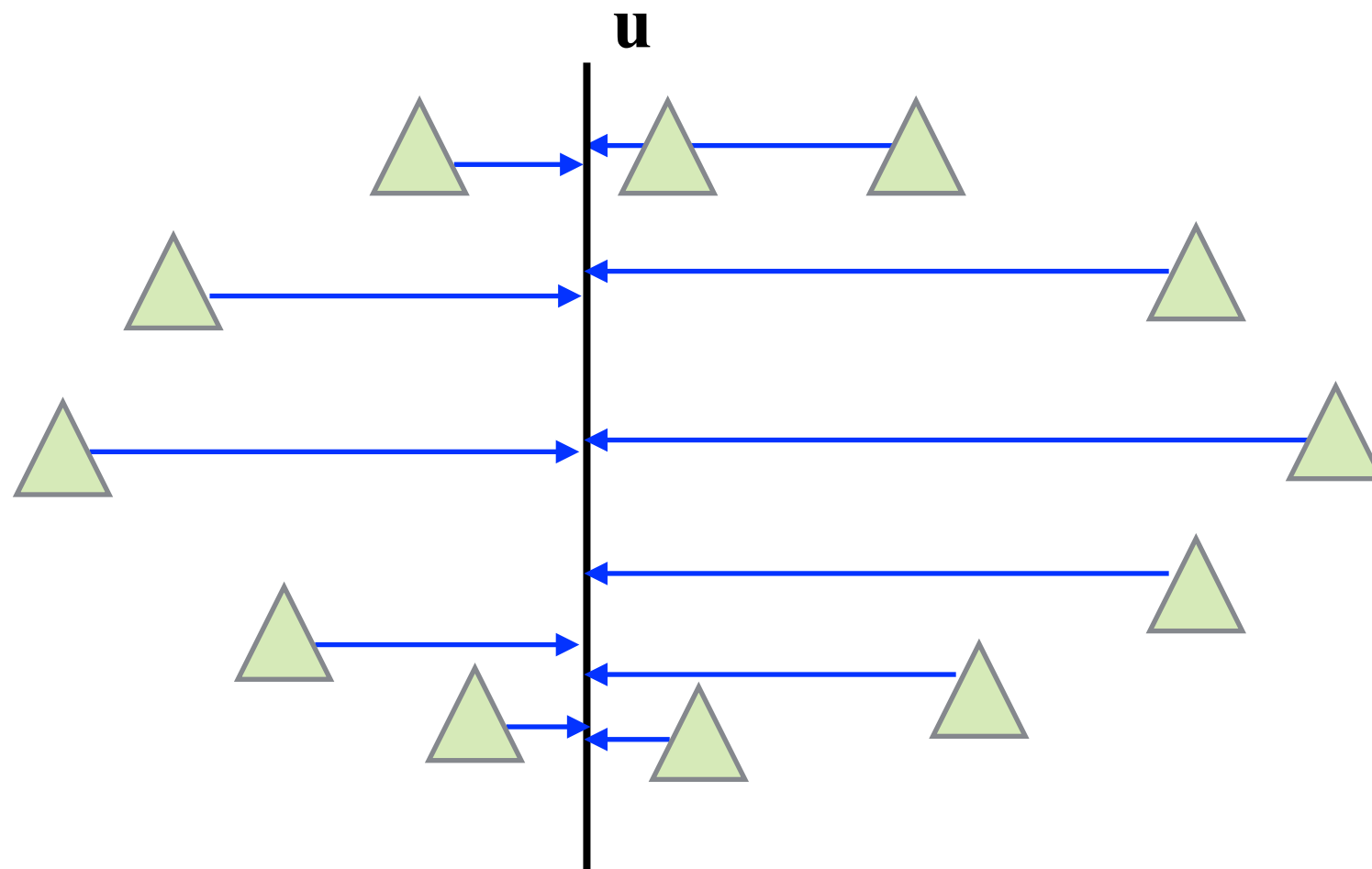
- In a **linear transformation**, we map data to new variables, which are linear functions of the original variables.
- **Example:** Bill owes Mary £5 and CHF15 after a holiday. Current exchange rates:
 $\text{€:£} \rightarrow 1.25:1$, $\text{€:CHF} \rightarrow 0.8:1$
- Based on rates, Bill owes $\text{€}(1.25 \times 5 + 0.8 \times 15) = \text{€}18.25$
- We can view this as a linear transformation...

£	CHF		€
5	15	\times	18.25
15	20		34.75
20	35		53.00



Projection Methods

- **Projection methods** map the original d -dimensional space to a new ($k < d$)-dimensional space, with the minimum loss of information.
- “Good” spaces for projections are characterised by preserving most of the useful information in the data - the **variation** in the data.

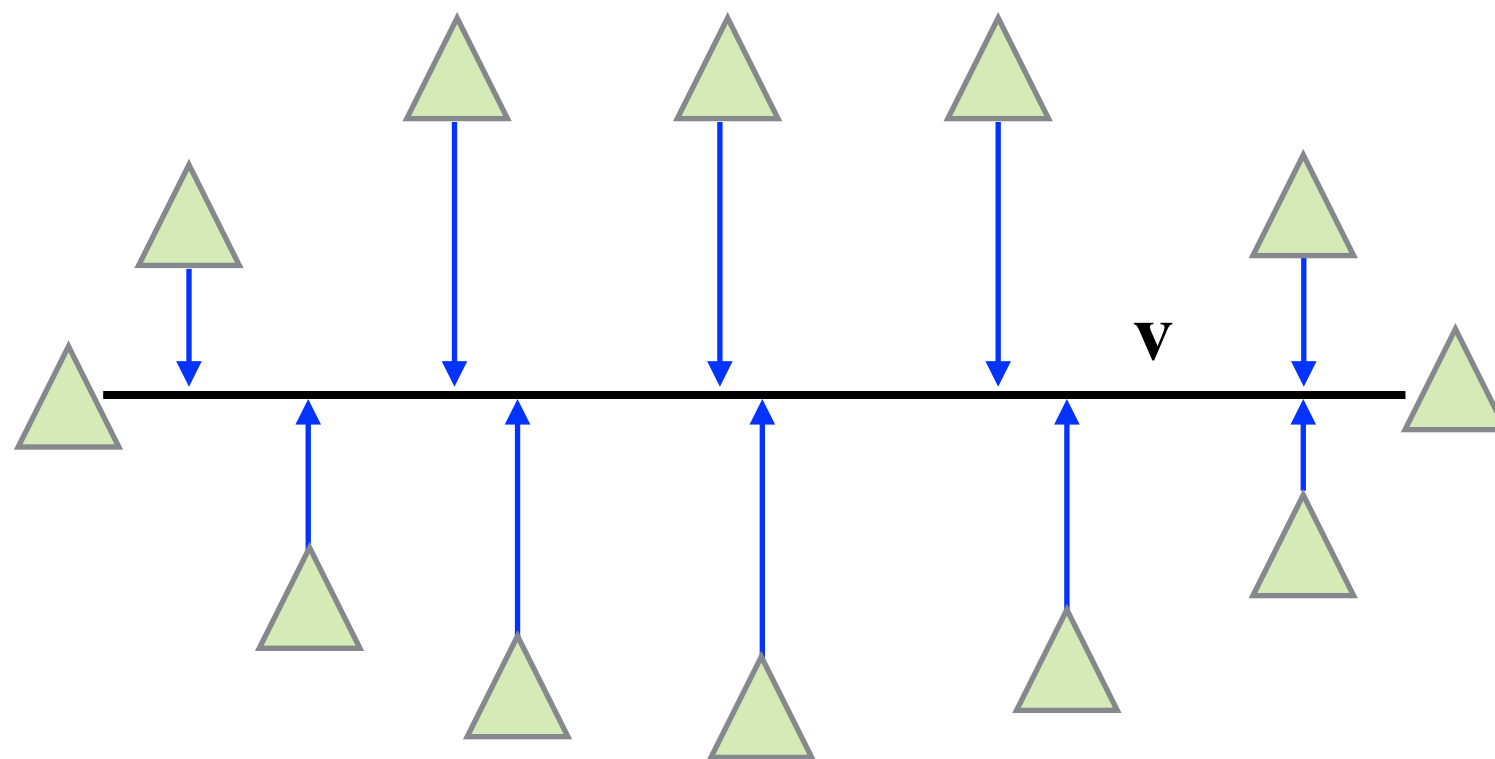


If we project the original data to this new dimension u , the data is not very spread out.

The dimension does not have high variance.

Projection Methods

- **Projection methods** map the original d -dimensional space to a new ($k < d$)-dimensional space, with the minimum loss of information.
- “Good” spaces for projections are characterised by preserving most of the useful information in the data - the **variation** in the data.

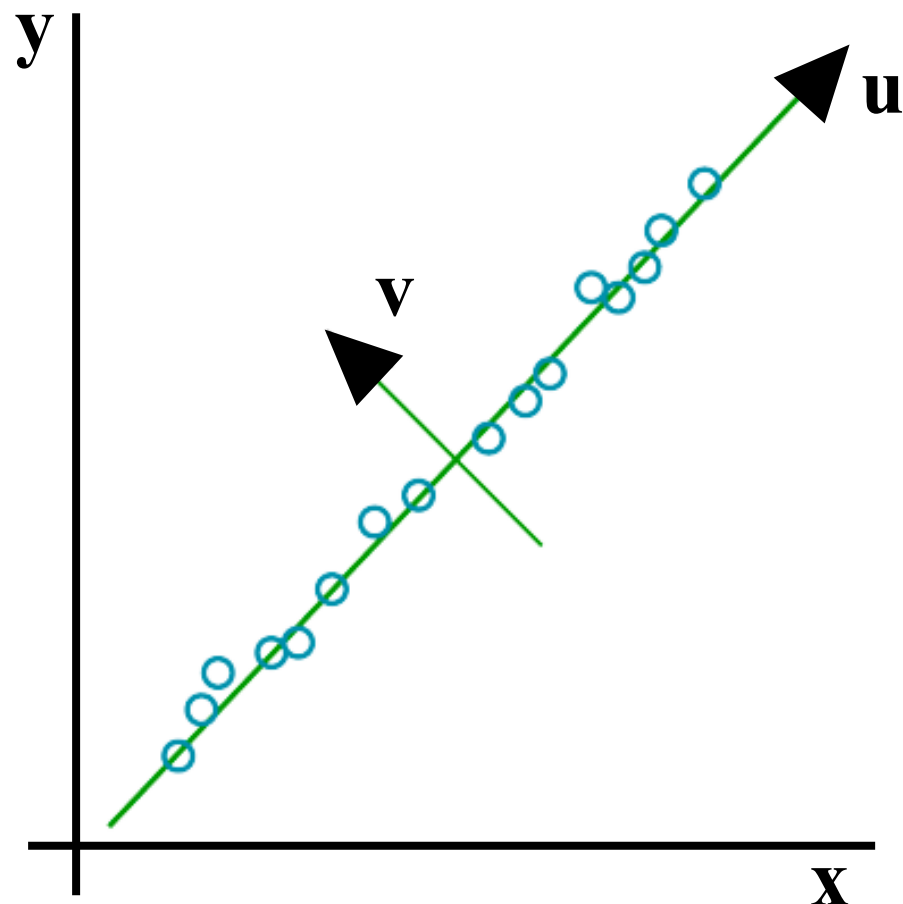


If we project to the alternative new dimension v , the data is far more spread out.

This new dimension has more variation - i.e. more information has been preserved.

Principal Component Analysis (PCA)

- To minimise the loss of information, we need to find new dimension(s) which maximise the variation.



Given data in terms of dimensions (x,y) , the principal direction in which the data varies is along the u axis.

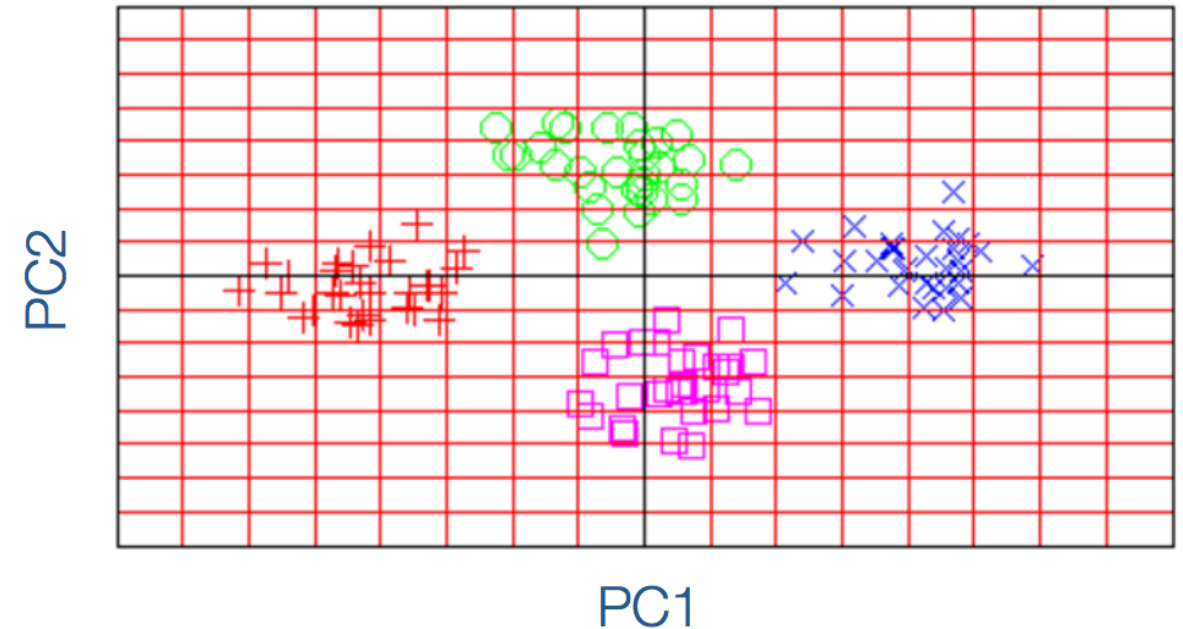
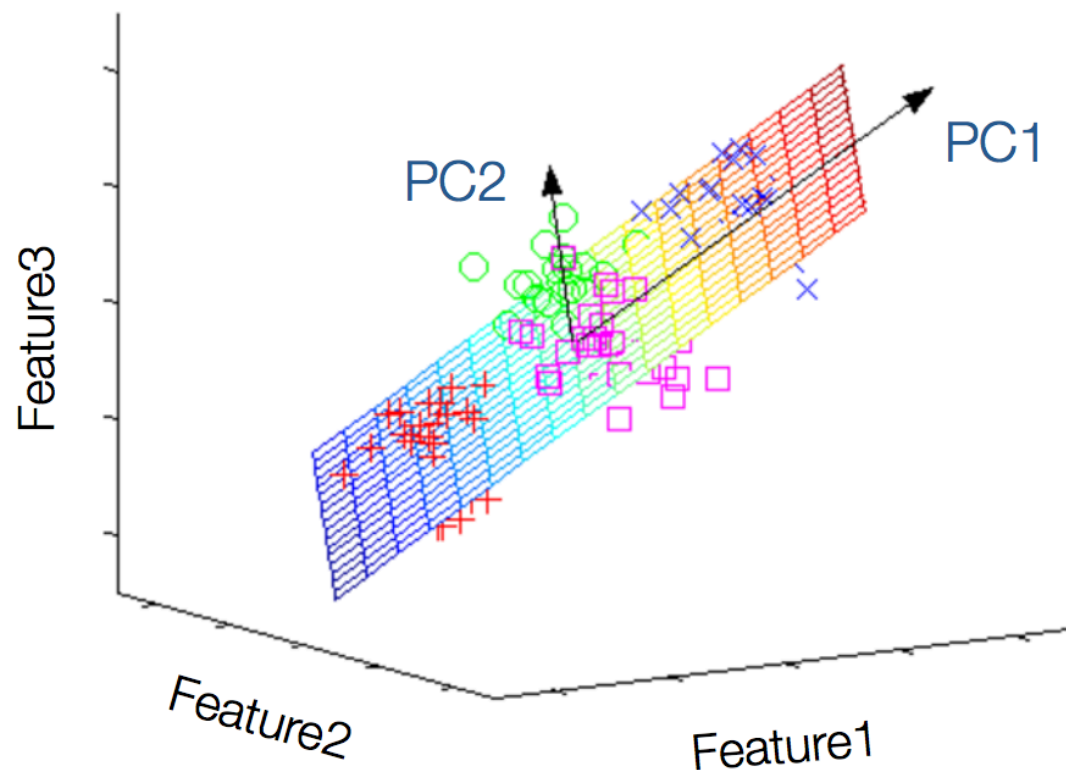
Very little variation in the data in the direction of v .

➔ Select the u dimension to represent the data.

- Use **Principal Component Analysis (PCA)** - an unsupervised projection method which performs dimensionality reduction, while trying to keep as much of the variance in the data as possible.

Principal Component Analysis (PCA)

- **Example:** Transformation of original data (3 dimensions) to a lower dimensional space (2 dimensions) via PCA.
- Using PCA, we can identify the two-dimensional plane that optimally describes the highest variance of the data. Each resultant dimension is a linear combination of the original 3 dimensions.



Eigenvectors and Eigenvalues

- Given an input matrix \mathbf{X} , an **eigenvector** of the matrix is a non-zero vector \mathbf{v} that satisfies the equation:

$$\mathbf{X}\mathbf{v} = \lambda\mathbf{v}$$

where the corresponding number λ is called an **eigenvalue**.

- Eigendecomposition** is the factorisation of a matrix into its eigenvalues and eigenvectors.

$$\mathbf{X} = \begin{pmatrix} 1.0000 & 0.5000 & 0.3330 & 0.2500 \\ 0.5000 & 1.0000 & 0.6667 & 0.5000 \\ 0.3333 & 0.6667 & 1.0000 & 0.7500 \\ 0.2500 & 0.5000 & 0.7500 & 1.0000 \end{pmatrix}$$

4 x 4 symmetric matrix

$$\mathbf{\Lambda} = \begin{pmatrix} 2.5361 & 0 & 0 & 0 \\ 0 & 0.8483 & 0 & 0 \\ 0 & 0 & 0.4078 & 0 \\ 0 & 0 & 0 & 0.2077 \end{pmatrix}$$

4 eigenvalues

$$\mathbf{V} = \begin{pmatrix} -0.37775 & -0.81052 & -0.44217 & -0.06988 \\ -0.53223 & -0.18762 & 0.74199 & 0.36221 \\ -0.56139 & 0.30099 & 0.04872 & -0.76926 \\ -0.50881 & 0.46611 & -0.50155 & 0.52169 \end{pmatrix}$$

4 eigenvectors

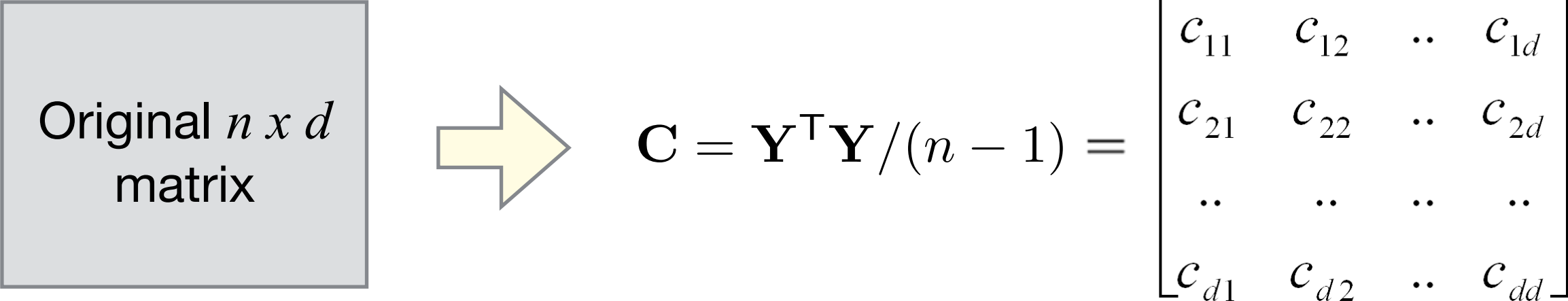
Eigenvectors and values exist in pairs:
every eigenvector has a corresponding eigenvalue.

Eigenvectors in PCA

- Each eigenvector has a direction - i.e. it is a dimension.
- Eigenvectors of symmetric matrices are **orthogonal** to each other - i.e. they point in completely different directions.
- Each eigenvalue is a number indicating how much variance there is in the data in that direction.
- The eigenvector with the largest eigenvalue has the most variance, making it a useful dimension for projection.
- **Principal Components (PCs)**: New dimensions constructed as linear combinations of the original features, which are uncorrelated with one another. Constructed from eigenvectors.
- The first PC accounts for the most variability in the data. The next PC has the highest variance possible under the constraint that it is uncorrelated with the first PC, and so on...

Covariance Matrix

- To assess variability in the data, we can measure the **covariance** between the original features - i.e. the tendency for two features x and y to vary in the same direction:
 - Do features x and y tend to increase together?
 - Or does feature y decrease as feature x increases?
- We estimate the covariances based on all n examples.
- By measuring the covariance between all pairs of features, we can create a symmetric **covariance matrix**.



The diagram illustrates the process of calculating the covariance matrix. On the left, a light gray box contains the text "Original $n \times d$ matrix". A large yellow arrow points from this box to the right. To the right of the arrow is the equation $\mathbf{C} = \mathbf{Y}^T \mathbf{Y} / (n - 1) =$ followed by a large square bracket containing a 4x4 matrix of elements: c_{11} , c_{12} , $..$, c_{1d} in the first row; c_{21} , c_{22} , $..$, c_{2d} in the second row; $..$, $..$, $..$, $..$ in the third row; and c_{d1} , c_{d2} , $..$, c_{dd} in the fourth row.

$$\mathbf{C} = \mathbf{Y}^T \mathbf{Y} / (n - 1) = \begin{bmatrix} c_{11} & c_{12} & .. & c_{1d} \\ c_{21} & c_{22} & .. & c_{2d} \\ .. & .. & .. & .. \\ c_{d1} & c_{d2} & .. & c_{dd} \end{bmatrix}$$

Applying PCA

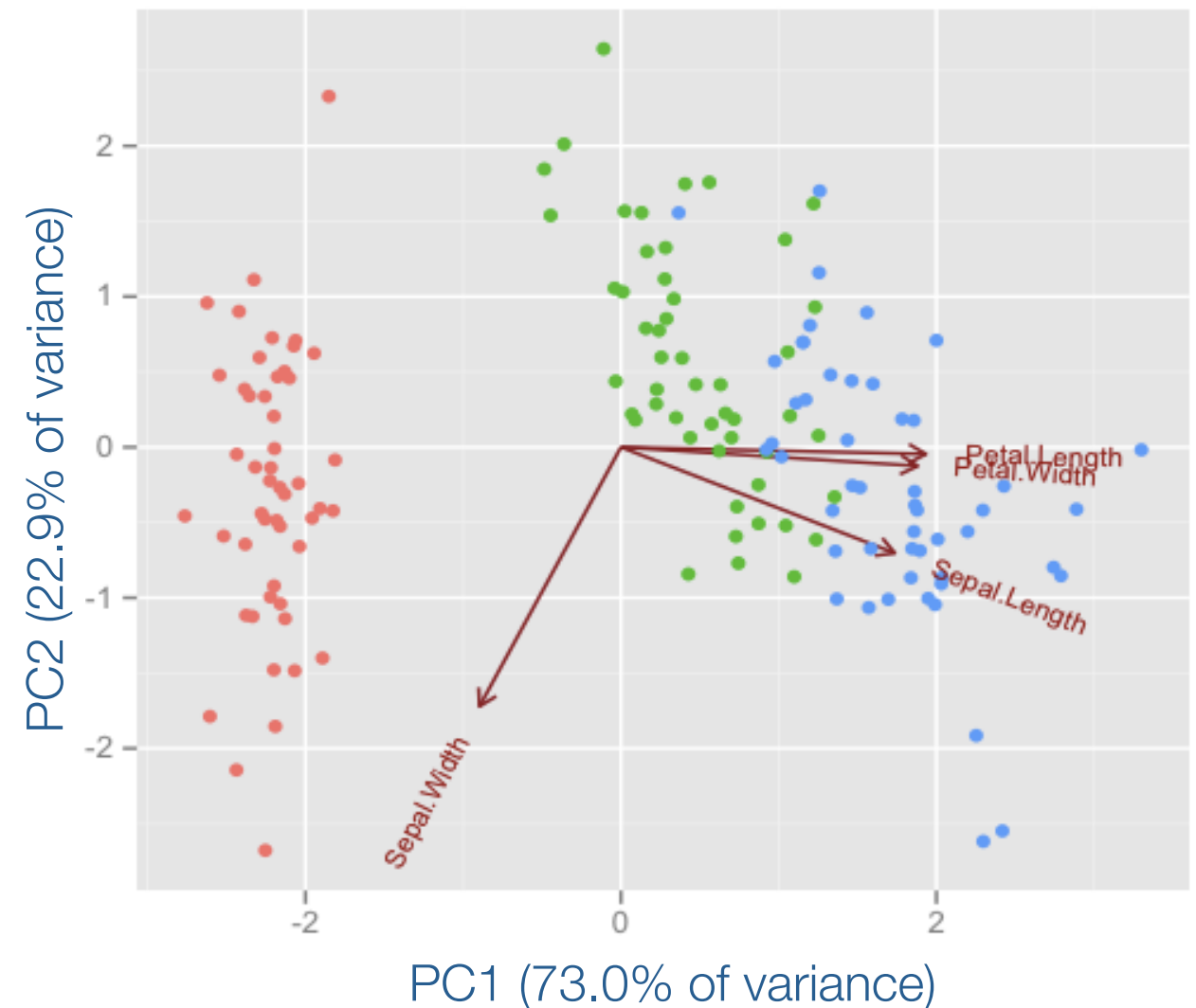
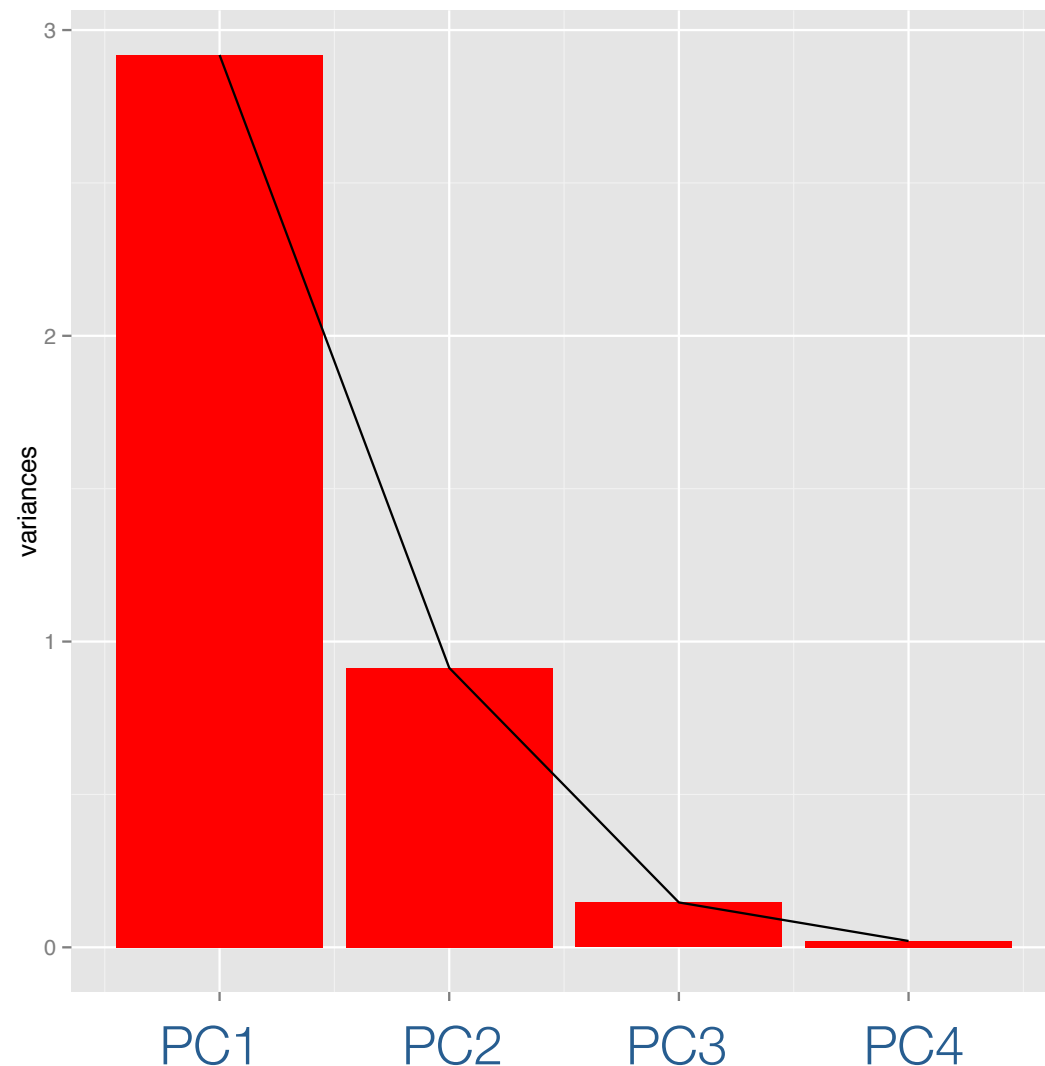
- **Input:** A dataset matrix \mathbf{X} with n examples (i.e. rows). The features of this matrix (i.e columns) might potentially be correlated.
- **PCA Process:**
 1. Calculate the mean of the columns of \mathbf{X} .
 2. Subtract the column means from each row of \mathbf{X} , to create the **centred matrix** \mathbf{Y} .
 3. Calculate the **covariance matrix** $\mathbf{C} = \mathbf{Y}^T \mathbf{Y} / (n - 1)$
 4. Calculate the eigenvectors of the covariance matrix \mathbf{C} .
 5. The Principal Components (PCs) are given by the eigenvectors of \mathbf{C} . The i -th PC is given by the eigenvector corresponding to the i -th largest eigenvalue of \mathbf{C} .
 6. Select an appropriate number of PCs k and use them as a new reduced $n \times k$ representation of the dataset.

Applying PCA

We generally select the k PCs with the highest variance.

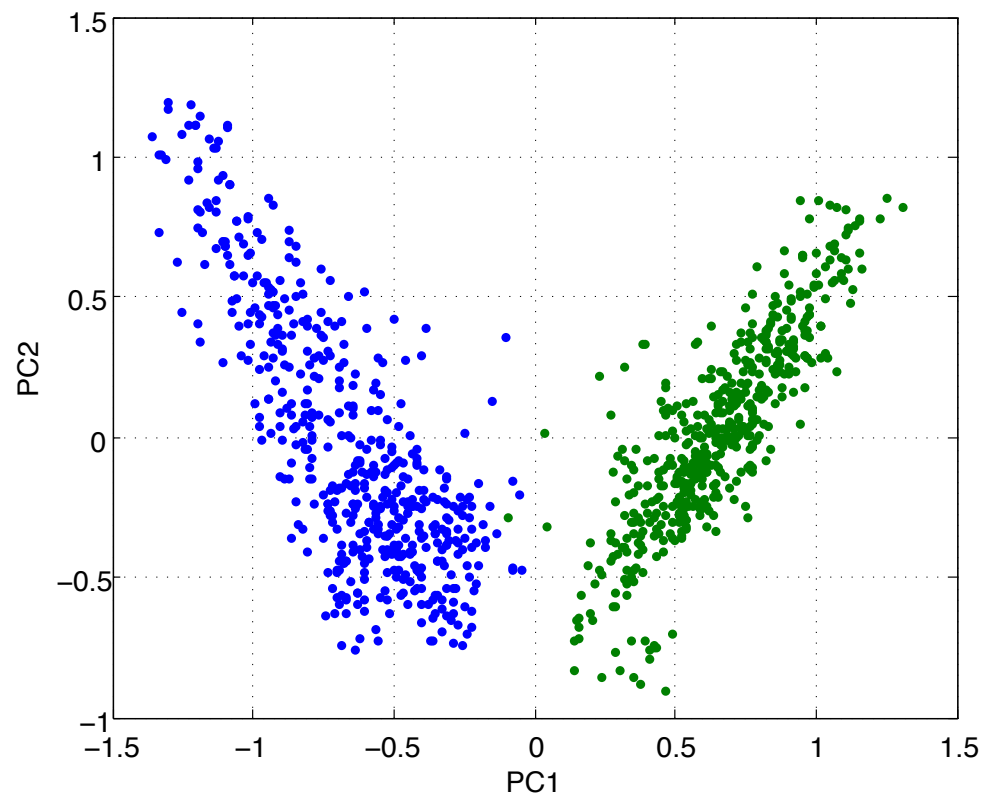
Example: Apply PCA to *Iris* data set, and examine amount of variance in each PC.

The first two PCs account for ~96% of the variance in the data.

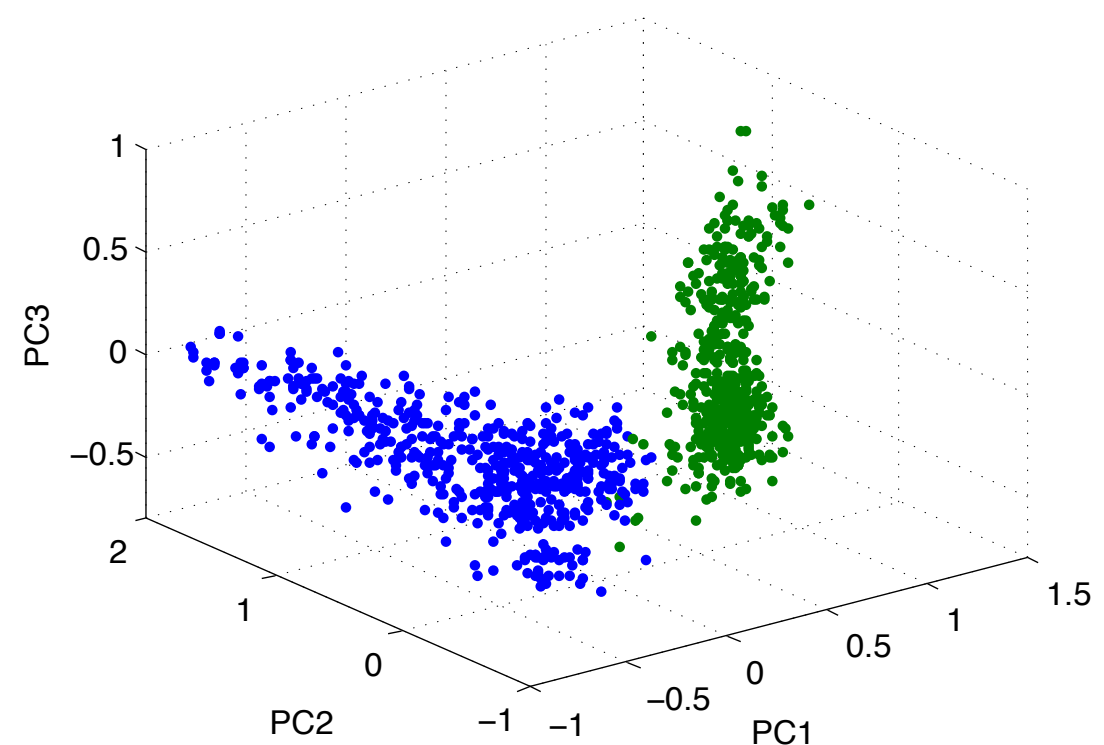


Example: PCA

- Collection of 1,021 BBC news articles on business + sport, represented by high-dimensional space with 5,570 features (words).
- Applying PCA allows us to visualise the data in a low dimensional space using a small number of PCs.



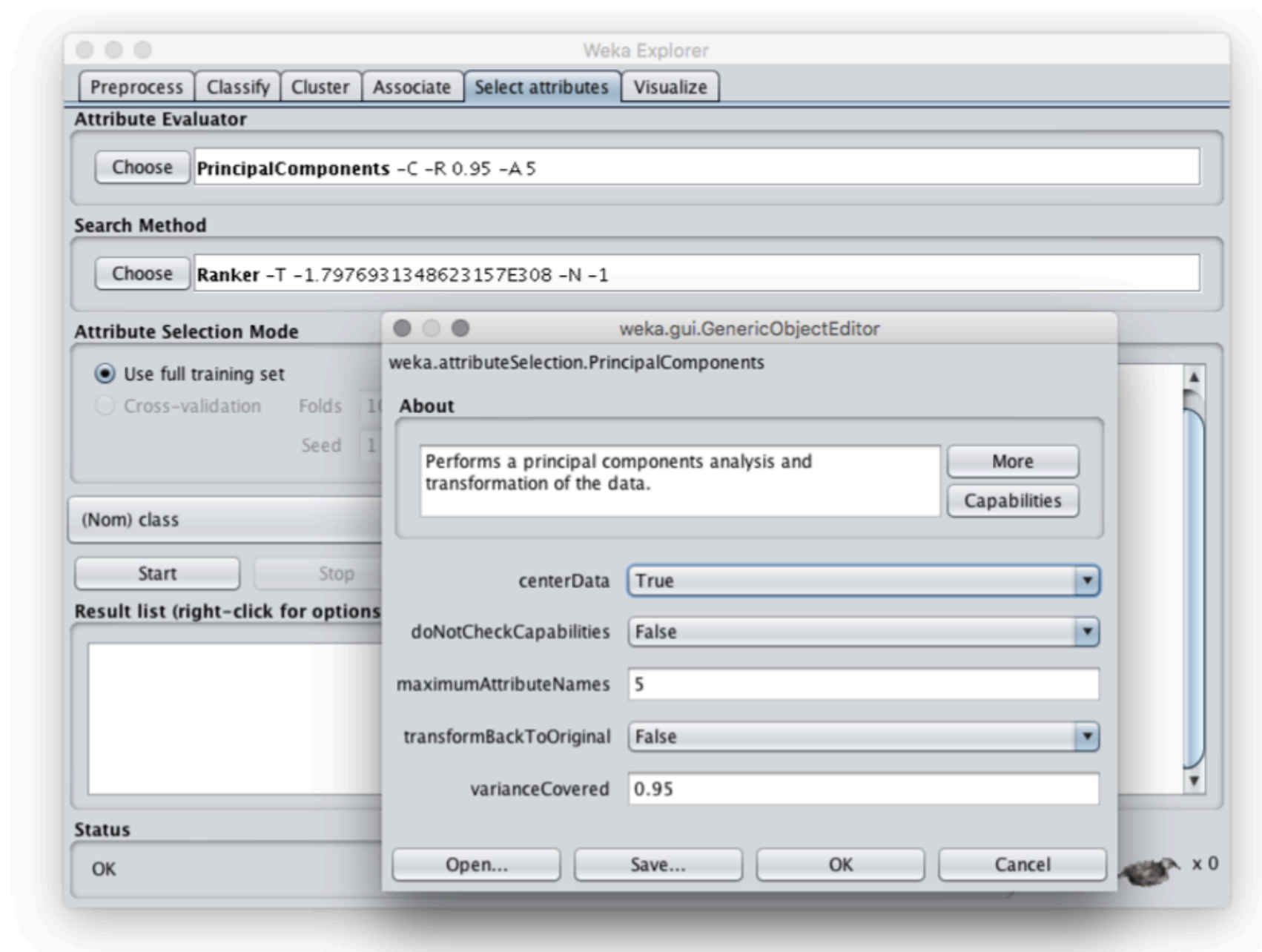
2 leading principal components (PCs)



3 leading principal components (PCs)

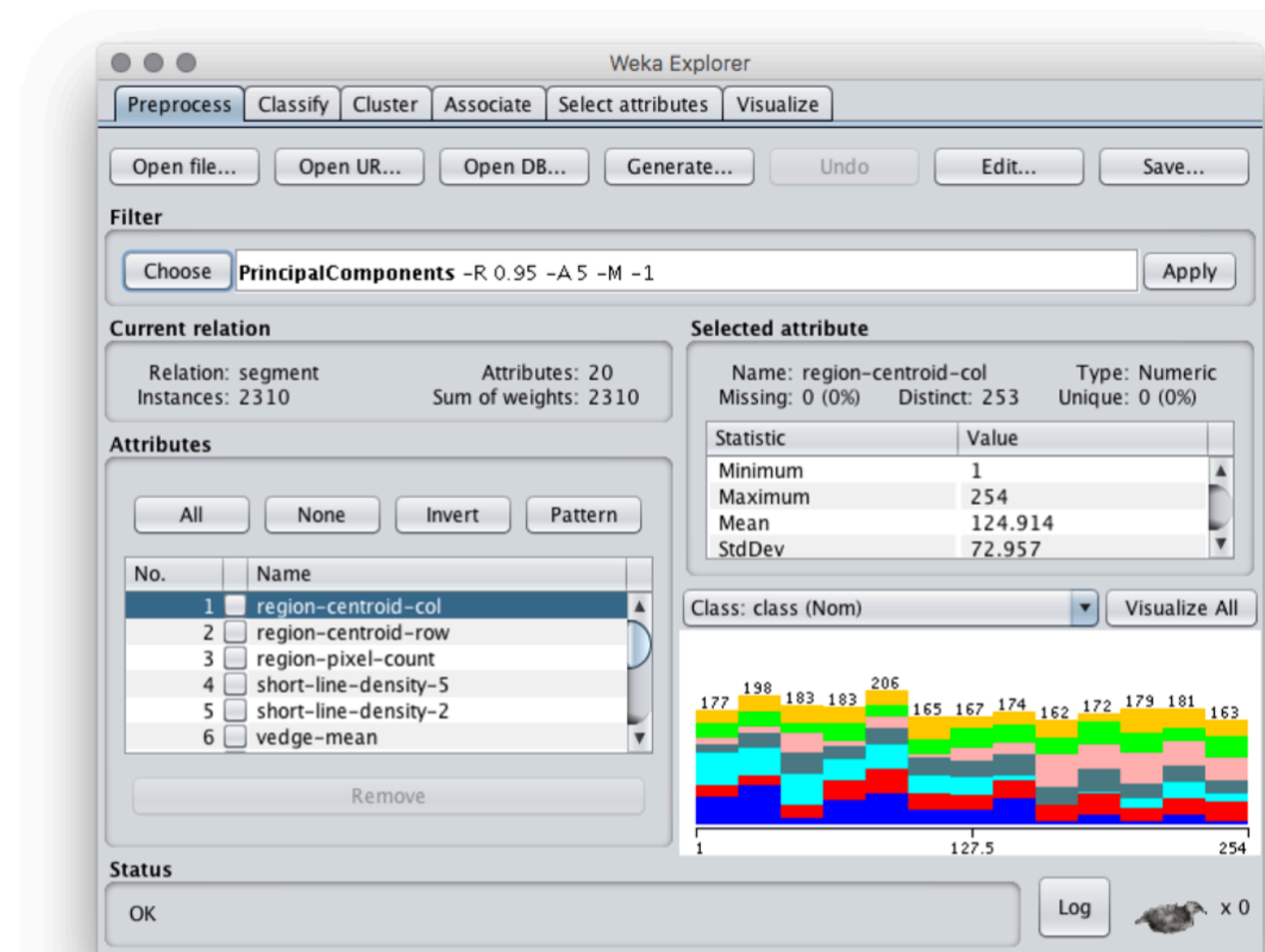
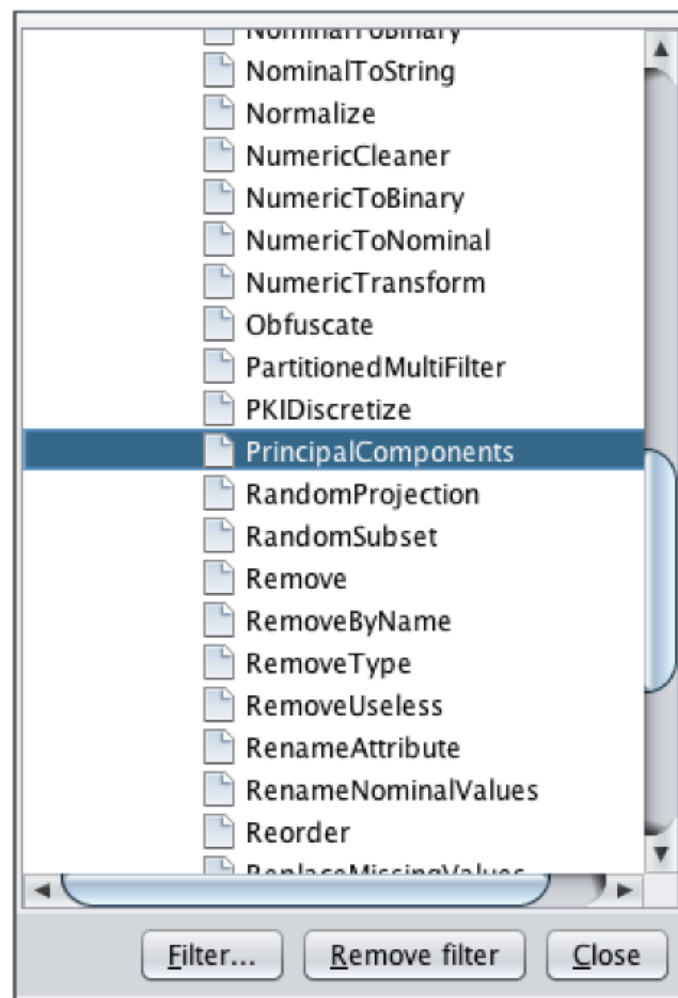
PCA in Weka - Method 1

In Weka Explorer, go to the *Select attributes* tab, choose *PrincipalComponents* as the evaluator, and *Ranker* as the search method. Change options for PCA, set *centerData* to True to use the standard covariance matrix approach.



PCA in Weka - Method 2

In Weka Explorer, go to the *Preprocess* tab. From *Filter* choose *PrincipalComponents* as the filter. Change to set *centerData* to True to use the standard covariance matrix approach. Hit *Apply* to replace the original features with PCA-based features.



References

- R. Bellman. “Adaptive control processes: a guided tour”. Princeton University Press, 1961.
- 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 international workshop on Machine Learning, 1992.
- T. Mitchell. “Machine Learning”. McGraw-Hill, 1997.
- P. Cunningham. "Dimension reduction", UCD CS Technical report UCD-COI-2007-5, 2007.