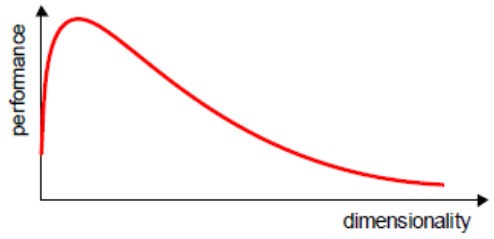
FEATURE EXTRACTION & SELECTION

CURSE OF DIMENSIONALITY

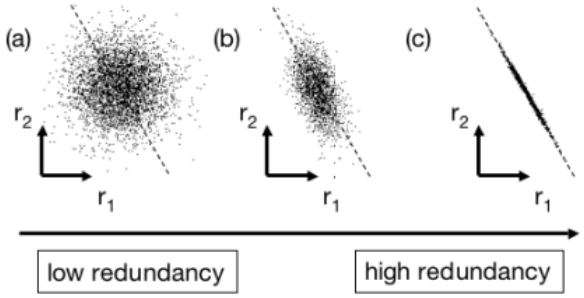
- *Curse of Dimensionality* refers to phenomena that arise when analyzing data in high-dimensional spaces that do not occur in low-dimensional settings.
- In the context of machine learning, challenge is that when the dimensionality increases (i.e., number of input variables), the volume of the space increases so fast that the available data become *sparse*.
- With a fixed number of training samples, the predictive power of the model reduces as the dimensionality increases (even if the variables have useful information), and this is known as *Hughes phenomenon* (named after Gordon F. Hughes).[1]



For a given sample size, there is a maximum number of features above which the performance of our classifier will degrade rather than improve. In most cases, the additional information that is lost by discarding some features is (more than) compensated by a more accurate mapping in the lower-dimensional space.[1]

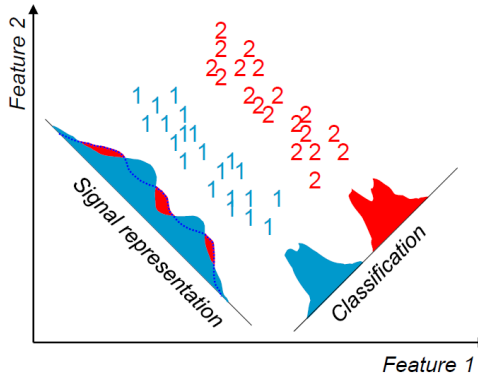
FEATURE EXTRACTION

- To overcome the curse of dimensionality, one has to rely on *feature extraction* or *feature selection* techniques to reduce dimensionality.
- Goal is extract features that eliminate noise and remove redundancy. Check the toy example by Jon Shlens.[2]



A spectrum of possible redundancies in data from the two separate recordings r_1 and r_2. The best-fit line $r_2 = kr_1$ is indicated by the dashed line.

- Feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.
- Best results are achieved when an expert constructs a set of application-dependent features, a process called *feature engineering* (important but difficult and expensive). Nevertheless, if no such expert knowledge is available, general dimensionality reduction techniques may help.
- *Signal Representation vs. Classification*: Depending on the criteria used by the objective function, feature extraction techniques are grouped into two categories:
  - *Signal Representation*: The goal of the feature extraction mapping is to represent the samples accurately in a lower-dimensional space. Popular feature extraction methods include: *Principal Component Analysis* (PCA), Kernel PCA, and Independent Component Analysis (ICA).
  - *Classification*: The goal of the feature extraction mapping is to enhance the class-discriminatory information in the lower-dimensional space. Example methods include *Linear discriminant analysis* (LDA).



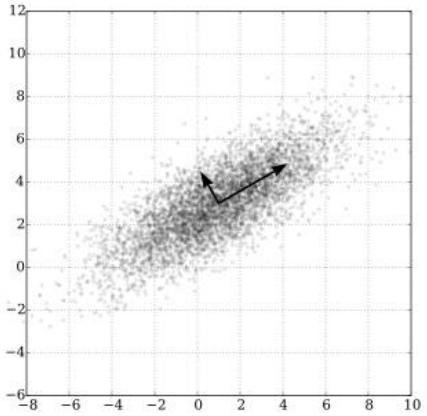
Signal representation versus classification.

PRINCIPAL COMPONENT ANALYSIS (PCA)

- *Principal component analysis* (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called *principal components*.
  - Principal coponents are not independent but are linearly uncorrelated!
- The number of principal components is less than or equal to the number of original variables.
- This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.
- Resulting vectors are an uncorrelated orthogonal basis set.
- PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of data matrix, usually after mean centering (and normalizing) the data matrix for each attribute.

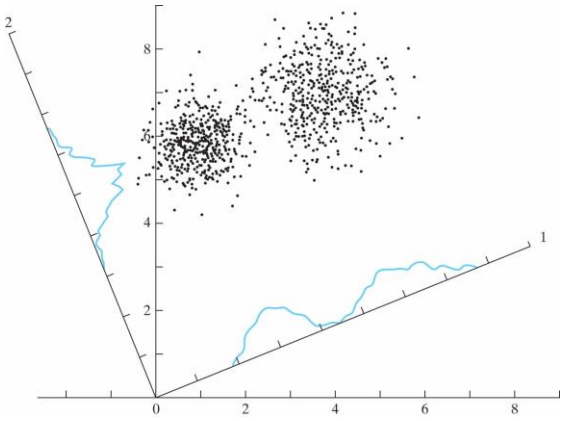
[1] Hughes, G.F. (January 1968). "On the mean accuracy of statistical pattern recognizers". *IEEE Transactions on Information Theory*. **14** (1): 55–63.doi:10.1109/TIT.1968.1054102

[2] Shlens, J. (March 2003). "A Tutorial on PCA". https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf



PCA of a multivariate Gaussian distribution. Vectors shown are the eigenvectors of covariance matrix scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean.

- The results of a PCA are usually discussed in terms of *component scores* (the transformed variable values corresponding to a particular data point) and *loadings* (the transformation 'weight' vector by which each standardized original variable should be multiplied to get component score).
- PCA is closely related to *factor analysis*, which typically incorporates more domain specific assumptions about underlying structure and solves eigenvectors of a slightly different matrix.



A cloud of data points is shown in two dimensions, and the density plots formed by projecting this cloud onto each of two axes, 1 and 2, are indicated. The projection onto axis 1 has maximum variance and clearly shows the bimodal, or clustered, character of the data.

PCA Intuition:

- PCA can be thought of as fitting an *n*-dimensional ellipsoid to the data, where each axis of the ellipsoid represents a principal component.
  - If some axis of the ellipse is small, then the variance along that axis is also small
  - By omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a commensurately small amount of information.
- To find the axes of the ellipse, we must first subtract the mean of each variable from the dataset to center the data around the origin.
- We then compute the covariance matrix of the data, and calculate the eigenvalues and corresponding eigenvectors of this covariance matrix.
  - We must also orthogonalize the eigenvector set and normalize each to become unit vectors.
  - Each of the mutually orthogonal unit eigenvectors can be interpreted as an axis of the ellipsoid fitted to the data.
- The proportion of the variance that each eigenvector represents can be calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues.
- It is important to note that this procedure is sensitive to the scaling of the data, and that there is no consensus as to how to best scale the data to obtain optimal results.

PCA Procedure Details:

- PCA can be derived several different ways. What follows is a particular approach.
- Consider a data matrix, $\boldsymbol{X}$, with column-wise zero empirical mean (the sample mean of each column has been shifted to zero), where each of the n rows represents a different observation and each of the p columns corresponds to a particular dataset variable or feature.
- Mathematically, the transformation is defined by a set of p-dimensional vectors of weights or *loadings* $\boldsymbol{w}_{(k)} = (w_1, \dots, w_p)_{(k)}$ that map each row vector of $\boldsymbol{x}_i$ of $\boldsymbol{X}$ to a new vector of principal component *scores* $t_{(i)} = (t_1, \dots, t_k)_{(i)}$, given by $t_{k\,(i)} = \boldsymbol{x}_i \cdot \boldsymbol{w}_{(k)}$ in such a way that the individual variables of $\boldsymbol{t}$ considered over the data set successively inherit the maximum possible variance from $\boldsymbol{x}$, with each loading vector $\boldsymbol{w}$ constrained to be a unit vector.

First Component:

- The first loading vector $\boldsymbol{w}_{(1)}$ thus has to satisfy

$$\boldsymbol{w}_{(1)} = \underbrace{\arg\max \left\{ \sum_i (t_1)_{(i)}^2 \right\}}_{\|\boldsymbol{w}\|=1} = \underbrace{\arg\max \left\{ \sum_i \left(\boldsymbol{x}_{(i)} \cdot \boldsymbol{w}_{(1)} \right)^2 \right\}}_{\|\boldsymbol{w}\|=1}$$

- Equivalently, writing this in matrix form gives

$$\boldsymbol{w}_{(1)} = \underbrace{\arg\max \{ \|\boldsymbol{X}\boldsymbol{w}\|^2 \}}_{\|\boldsymbol{w}\|=1} = \underbrace{\arg\max \{ \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X} \boldsymbol{w} \}}_{\|\boldsymbol{w}\|=1}$$

- A standard result for a symmetric matrix such as $\boldsymbol{X}^T\boldsymbol{X}$ is that the maximum possible value is the largest eigenvalue of the matrix, which occurs when $\boldsymbol{w}$ is the corresponding eigenvector.
- With $\boldsymbol{w}_{(1)}$ found, the first component of a data vector $\boldsymbol{x}_{(i)}$ can then be given as a score $t_{1(i)} = \boldsymbol{x}_{(i)} \cdot \boldsymbol{w}_{(1)}$ in the transformed co-ordinates, or as the corresponding vector in the original variables, $\{\boldsymbol{x}_{(i)} \cdot \boldsymbol{w}_{(1)}\}\boldsymbol{w}_{(1)}$.

Further Components:

- The k^{th} component can be found by subtracting the first $k-1$ principal components from $\boldsymbol{X}$:

$$\widehat{\boldsymbol{X}}_k = \boldsymbol{X} - \sum_{s=1}^{k-1} \boldsymbol{X}\boldsymbol{w}_{(s)}\boldsymbol{w}_{(s)}^T$$

and then finding the loading vector which extracts the maximum variance from this new data matrix:

$$\boldsymbol{w}_{(k)} = \underbrace{\arg\max \left\{ \left\| \widehat{\boldsymbol{X}}_k \boldsymbol{w} \right\|^2 \right\}}_{\|\boldsymbol{w}\|=1}$$

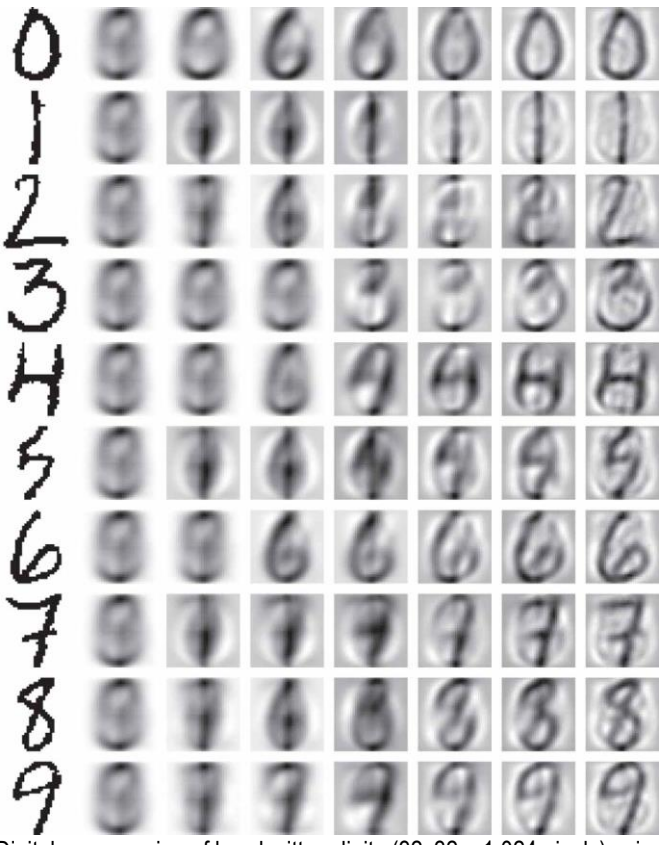
- It turns out that this gives the remaining eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$, with the maximum values for the quantity in brackets given by their corresponding eigenvalues. Thus the loading vectors are eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$.
- The k^{th} component of a data vector $\boldsymbol{x}_{(i)}$ can therefore be given as a score $t_{k(i)} = \mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}$ in the transformed co-ordinates, or as the corresponding vector in the space of the original variables, $\{\mathbf{x}_{(i)} \cdot \mathbf{w}_{(k)}\}\, \mathbf{w}_{(k)}$, where $\mathbf{w}_{(k)}$ is the k^{th} eigenvector of $\boldsymbol{X}^T\boldsymbol{X}$.
- The full principal components decomposition of $\boldsymbol{X}$ can therefore be given as $\boldsymbol{T} = \boldsymbol{X}\boldsymbol{W}$ where $\boldsymbol{W}$ is a p-by-p matrix whose columns are the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$.

Notes:

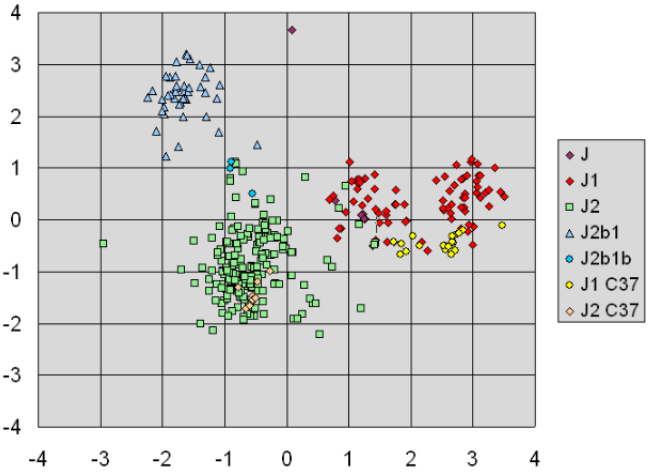
- $\boldsymbol{X}^T\boldsymbol{X}$ is proportional to the empirical sample covariance matrix of the dataset $\boldsymbol{X}$.
- Hence, PCA can be carried out by starting with covariance matrix or correlation matrix.
- In Matlab, the "pca" command carries out PCA [LINK]
- PCA also has similarities with another matrix factorization method known as the Singular Value Decomposition (SVD) and is employed by most software for carrying out PCA.

Dimensionality Reduction:

- The transformation $\boldsymbol{T} = \boldsymbol{X}\,\boldsymbol{W}$ maps a data vector $\boldsymbol{x}_{(i)}$ from an original space of p variables to a new space of p variables which are uncorrelated over the dataset.
- However, not all the principal components need to be kept. Keeping only the first L principal components, produced by using only the first L loading vectors, gives the truncated transformation $\boldsymbol{T}_L = \boldsymbol{X}\boldsymbol{W}_L$ where the matrix $\boldsymbol{T}_L$ now has n rows but only L columns.
- In other words, PCA learns a linear transformation where the columns of $p \times L$ matrix $\boldsymbol{W}$ form an orthogonal basis for the L features (the components of representation t) that are decorrelated.
- By construction of the transformed data matrix with only L columns, this score matrix maximizes the variance in the original data that has been preserved, while minimizing the total squared reconstruction error $\left\|\boldsymbol{T}\boldsymbol{W}^T - \boldsymbol{T}_L\boldsymbol{W}_L^T\right\|_2^2$ or $\|\boldsymbol{X} - \boldsymbol{X}_L\|_2^2$.
- Such dimensionality reduction can be a very useful step for visualizing and processing high-dimensional datasets, while still retaining as much of the variance in the dataset as possible.
- *Example*: In the scatterplot above, selecting $L = 2$ and keeping only the first two principal components (out of 37) finds the two-dimensional plane through the high-dimensional dataset in which the data is most spread out, so if the data contains clusters these too may be most spread out, and therefore most visible to be plotted out in a two-dimensional diagram.



Digital compression of handwritten digits (32x32 = 1,024 pixels) using principal-components analysis (1,700 samples). First column is a sample original image and 2nd column shows computed means across dataset. Remaining columns show reconstructed images with increasing number of PCs (1, 2, 5, 16, 32, and 64). (Source: Dr. Juha Karhunen through Haykin 2009)



Scatterplot (1st two PCs) of Y-STR haplotypes from 37 Y-chromosomal STR markers from 354 individuals. PCA nicely separates clusters corresponding to genetic descent. (Wikipedia)

KERNEL PRINCIPAL COMPONENT ANALYSIS (KERNEL-PCA)

- Given that PCA only explores linear transformations, for some applications, it might be necessary to explore non-linear feature extraction methods. Kernel-PCA is one such option.
- Kernel-PCA involves mapping the original data into the high dimensional feature space with the hope that nonlinearities that

are present in the original input space might become linear in the feature space. If true, PCA is carried out in feature space.

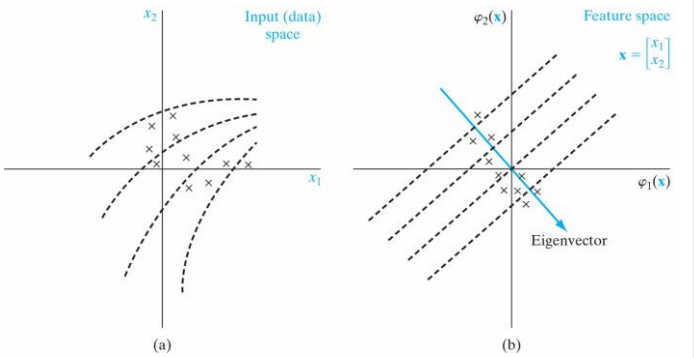
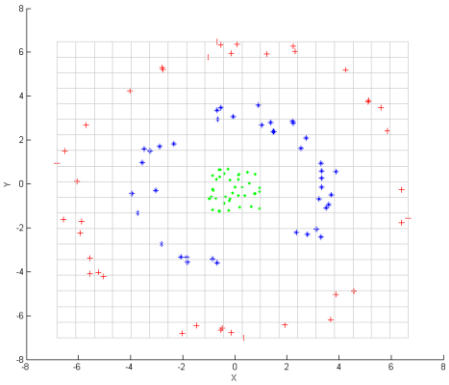
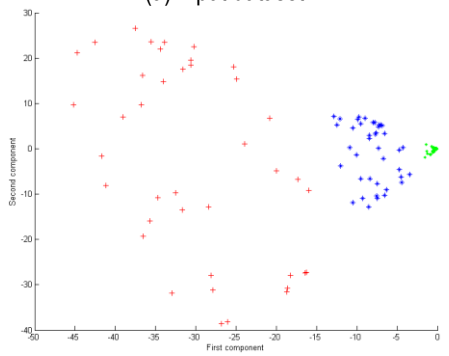


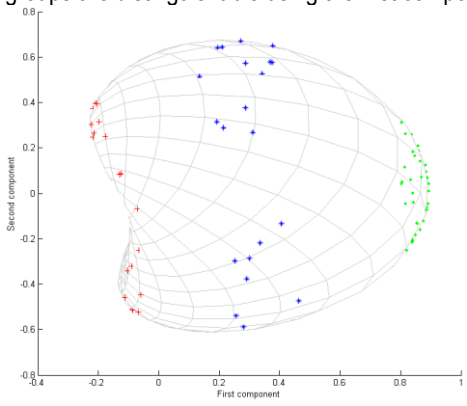
Illustration of kernel PCA. (a) 2-D input space displaying dataset. (b) 2-D feature space displaying induced images of the data points around a principal eigenvector. Uniformly spaced dashed lines in part (b) appear as nonlinear corresponding contours in input space.



(a) Input dataset



(b) Output with polynomial kernel of 2nd order with unit additive constant. The three groups are distinguishable using the first component only.



(c) Output with a Gaussian kernel.
Illustration of kernel PCA

- Using the kernel trick, we are able to avoid making calculations in the feature space.
- For Matlab, the Kernel Methods Toolbox from Matlab Central supports Kernel PCA and other kernel-based algorithms.

Independent Component Analysis (ICA)

- *Independent component analysis* attempts to decompose the set of variables (or a multivariate signal) into independent non-Gaussian variables (or signals).
- In the model, the data variables are assumed to be linear mixtures of some unknown latent variables, and the mixing system is also unknown.
- The latent variables are assumed nongaussian and mutually independent, and they are called the independent components of the observed data. These independent components, also called sources or factors, can be found by ICA.
- In general, ICA is a very challenging exercise and not that intuitive w.r.t. PCA.
- *Example*: Sound is usually a signal that is composed of the numerical addition, at each time $\boldsymbol{t}$, of signals from several sources. The question then is whether it is possible to separate these contributing sources from the observed total signal. When the statistical independence assumption is correct, blind ICA separation of a mixed signal gives very good results.

Neural Networks for Implementing PCA and Kernel-PCA

Hebbian-Based Maximum Eigenfilter

- A single linear neuron with a Hebbian-type adaptation rule for its synaptic weights can evolve into a filter for the first principal component of the input distribution:

$$\boldsymbol{w(n+1) = w(n) + \eta y(n)[x(n) - y(n)w(n)]}$$

Generalized Hebbian-Based PCA

- The single linear neural model of the Hebbian-based maximum eigenfilter can be expanded into a single layer network for carrying out the full PCA
- The Generalized Hebbian Algorithm (GHA) update rule is:

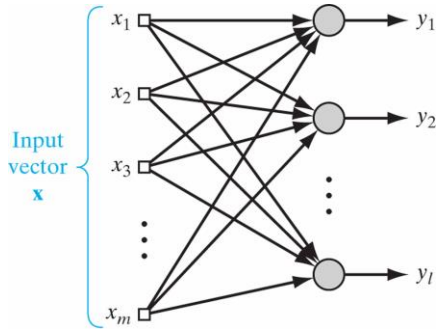
$$\boldsymbol{\Delta w_{ji}(n) = \eta\left(y_j(n)x_i(n) - y_j(n)\sum_{k=1}^{j} w_{ki}(n)y_k(n)\right)} \quad \begin{matrix} \boldsymbol{i = 1 \ldots, m} \\ \boldsymbol{j = 1 \ldots, l} \end{matrix}$$

- We can rewrite this as:

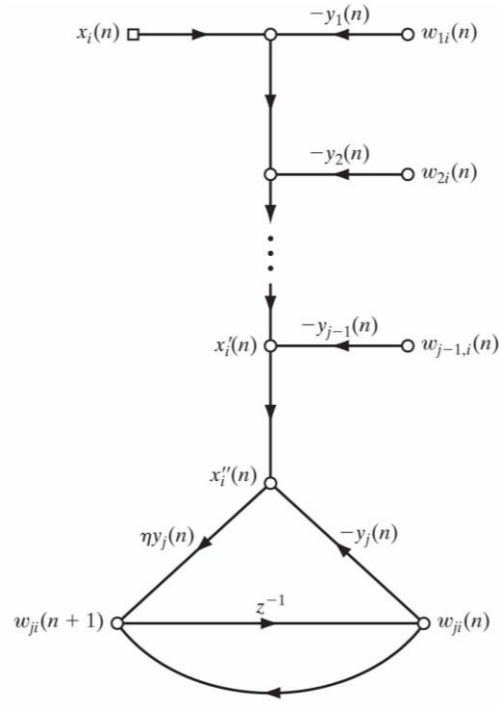
$$\boldsymbol{\Delta w_{ji}(n) = \eta y_j(n)x_i''(n)}$$
$$\boldsymbol{x_i''(n) = x_i'(n) - w_{ji}(n)y_j(n)}$$
$$\boldsymbol{x_i'(n) = x_i(n) - \sum_{k=1}^{j-1} w_{ki}(n)y_k(n)}$$

Kernel Principal Component Analysis (Kernel-PCA)

- For implementing Kernel PCA, we borrow the ideas from SVMs (kernel trick) to extend the GHA. It however becomes a batch learning algorithm.



(a) Feedforward network with a single layer.

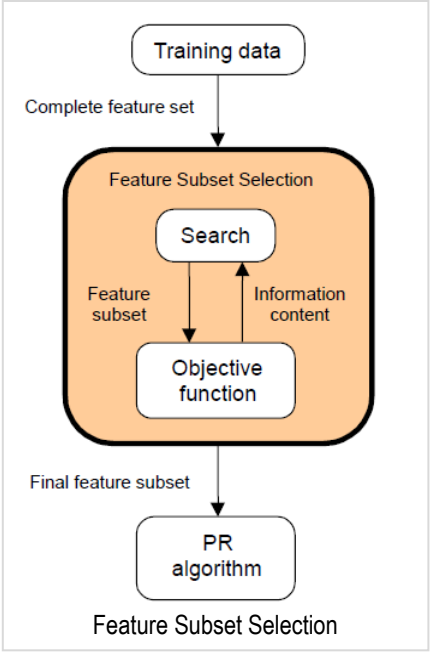


(b) Generalized Hebbian learning.

Signal-flow graph of generalized Hebbian algorithm.

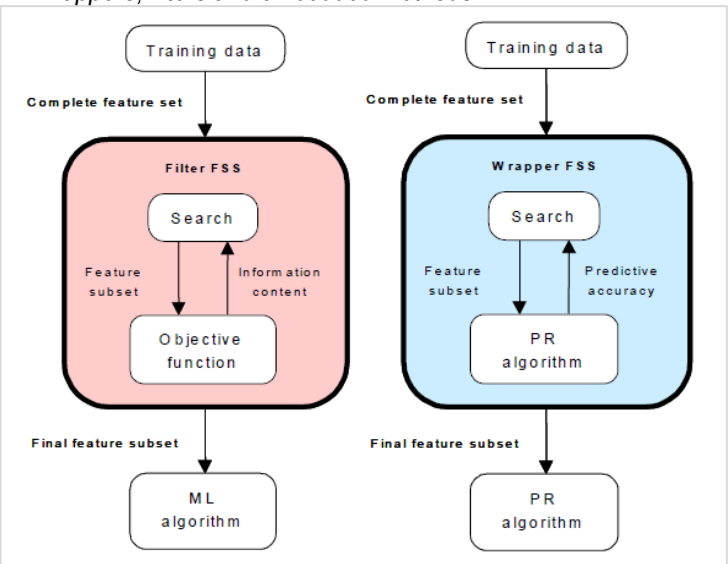
FEATURE SELECTION

- *Feature selection*, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are generally used for three reasons:
  - Simplification of models to make them easier to interpret by researchers/users
  - Shorter training times
  - Enhanced generalization by reducing overfitting
- The central premise when using a feature selection technique is that the data contains many features that are either *redundant* or *irrelevant*, and can thus be removed without incurring much loss of information.
  - Redundant or irrelevant features are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated.
- Feature selection algorithm can be seen as the combination of a *search technique* for proposing feature subsets along with an *evaluation measure* which scores the different feature subsets.



Feature Subset Selection

- Simplest algorithm is to test each possible subset of features for finding the one which minimizes the error rate.
  - This is an exhaustive search of the space and involves $\binom{N}{M}$ combinations for a fixed value of M, and 2^N combinations if M must be optimized as well
  - This number of combinations is unfeasible, even for moderate values of M and N, so a search procedure must be used in practice
  - For example, exhaustive evaluation of 10 out of 20 features involves $184{,}756$ feature subsets; exhaustive evaluation of 10 out of 100 involves more than 10^{13} feature subsets.
- Choice of evaluation metric heavily influences the algorithm, and it is these evaluation metrics which distinguish between the three main categories of feature selection algorithms: *wrappers*, *filters* and *embedded methods*.



Filters vs Wrappers for Feature Subset Selection

- *Wrapper Methods*: Use a predictive model to score feature subsets.

- Each new subset is used to train a model, which is tested on a hold-out set.
- Counting the number of mistakes made on that hold-out set (the error rate of the model) gives the score for that subset.

- As wrapper methods train a new model for each subset, they are computationally intensive, but usually provide the best performing feature set for that particular type of model.

- *Filter Methods*: Use a proxy measure instead of the error rate to score a feature subset. This measure is chosen to be fast to compute, while still capturing the usefulness of the feature set.
  - Common measures include the mutual information, Pearson product-moment correlation coefficient, intraclass correlation (a descriptive statistic that can be used when quantitative measurements are made on units that are organized into groups; describes how strongly units in the same group resemble each other), and rank correlation (studies relationships between rankings of different variables).
  - *Mutual Information:* Determines how similar the joint distribution $p(X,Y)$ is to the products of factored marginal distribution $p(X)p(Y)$; If two variables are independent, their MI is zero:

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right) \quad \text{Discrete variables}$$

$$I(X;Y) = \int_Y \int_X p(x,y) \log \left(\frac{p(x,y)}{p(x)\,p(y)} \right) dx\,dy \quad \text{Continuous variables}$$

  - *Pearson Product-Moment Correlation Coefficient*: Measures the strength and direction of the linear relationship between two variables that is defined as the (sample) covariance of the variables divided by the product of their (sample) standard deviations
  - Note: Unlike Pearson product-moment correlation coefficients, mutual information contains information about all dependence—linear and nonlinear. It is also a valid measure for cases where the variables are not real-valued (unlike correlation coefficient).
  - Filters are usually less computationally intensive than wrappers, but they produce a feature set which is not tuned to a specific type of predictive model.
  - Lack of tuning means a feature set from a filter is more general than the set from a wrapper, usually giving lower prediction performance than a wrapper.
  - However, the feature set doesn't contain the assumptions of a prediction model, and so is more useful for exposing the relationships between the features.
  - Many filters provide a feature ranking rather than an explicit best feature subset, and the cut off point in the ranking is chosen via cross-validation (see Matlab function 'rankfeatures').
  - Filter methods have also been used as a preprocessing step for wrapper methods, allowing a wrapper to be used on larger problems.
- *Embedded Methods*: Are a catch-all group of techniques which perform feature selection as part of the model construction process.
  - An exemplar of this approach is the LASSO method for constructing a linear model, which penalizes the regression coefficients with an L1 penalty, shrinking many of them to zero. Any features which have non-zero regression coefficients are 'selected' by the LASSO algorithm.
  - Another popular approach is the Recursive Feature Elimination algorithm, commonly used with Support Vector Machines to repeatedly construct a model and remove features with low weights. These approaches tend to be between filters and wrappers in terms of computational complexity.
- In traditional statistics, the most popular form of feature selection is stepwise regression, which is a wrapper technique. It is a greedy algorithm that adds the best feature (or deletes the worst feature) at each round. The main control issue is deciding when to stop the algorithm. In machine learning, this is typically done by cross-validation. In statistics, some criteria are optimized. This leads to the inherent problem of nesting. More robust methods have been explored, such as branch and bound methods.
- *Matlab*: The Statistics and Machine Learning Toolbox function 'sequentialfs' carries out sequential feature selection. Input arguments include predictor and response data and a function handle to a file implementing the criterion function. Optional inputs allow you to specify SFS (sequential forward selection) or SBS (sequential backward selection), required or excluded features, and the size of the feature subset.
- *Matlab Central*: Feature Selection Library: Library supports a number of feature selection methods, including:
  - Filters: Correlation based ranking of input features, Laplacian, and others.
  - Wrappers: Mutual Information, Minimum Redundancy Maximum Relevance (mRMR), Relief-F (for both classification and regression), and the very promising Infinite feature selection method (for classification).
- *Matlab Central*: Evolutional Feature Selection Toolbox: Allows application of Metaheuristics and Evolutionary Algorithms to Feature Selection in Various Modes:
  - Selection of variable number of features using Genetic Algorithm (GA): Fixed number of best features selection (e.g., the '5' most important features) or as a discrete combinatorial optimization problem using Simulated Annealing and/or Ant Colony Optimization (ACO)
  - As a real-valued optimization problem using Particle Swarm Optimization (PSO)
  - Multi-objective feature selection using Non-dominated Sorting Genetic Algorithm II (NSGA-II).

Table: Summary of Feature Subset Search Strategies

	Accuracy	Complexity	Advantages	Disadvantages	Approaches
Exhaustive	Always finds the optimal solution	Exponential	High accuracy	High complexity	Exhaustive; Branch & Bound
Sequential	Good if no backtracking needed	Quadratic	Simple and fast	Cannot backtrack	Forward Selection; Backward Selection; Bidirectional Search
Randomized	Good with proper control parameters	Generally low	Designed to escape local minima	Difficult to choose good parameters	Simulated Annealing; Genetic Algorithms