

CHAPTER 1

INTRODUCTION TO PATTERN RECOGNITION SYSTEM

1.1 Overview

One of the most important capabilities of mankind is learning by experience, by our endeavors, by our faults. By the time we attain an age of five most of us are able to recognize digits, characters; whether it is big or small, uppercase or lowercase, rotated, tilted. We will be able to recognize, even if the character is on a mutilated paper, partially occluded or even on the clustered background. Looking at the history of the human search for knowledge, it is clear that humans are fascinated with recognizing patterns in nature, understand it, and attempt to relate patterns into a set of rules. But the question is how this experience can be used to make machines to learn. The most important challenge is how to generalize these experiences, how do we make decisions and how our experiences can be built into a machine? This has been one of the main fundamental principles behind the development of vast range of theories and concepts that are based on the natural world.

Looking at the history, pattern recognition system has come a long way. Earlier it was confined to theoretical research in the field of statistics for deriving various models out of the large amount of data. With the advent in computer technology, number of practical applications is increased in manifold which lead to further theoretical developments. At present, pattern recognition has become integral part of any machine intelligence system that exhibit decision making capabilities. Many different mathematical techniques are used for this purpose.

Pattern recognition is concerned with the design and development of systems that recognize patterns in data. The purpose of a pattern recognition program is to analyze a scene in the real world and to arrive at a description of the scene which is useful for the accomplishment of some task. The real world observations are gathered through sensors and pattern recognition system classifies or describes these observations. A feature extraction mechanism computes numeric or symbolic information from these observations. These extracted features are then classified or described using a classifier. The process used for pattern recognition consists of many procedures that ensure efficient description of the patterns.

1.2 Pattern Recognition

Pattern recognition can be defined as the categorization of input data into identifiable classes via the extraction of significant features or attributes of the data from a background of irrelevant detail. Duda and Hart defined it as a field concerned with machine recognition of meaningful regularities in noisy or complex environments. A more simple definition is search for structure in data. According to Jain et al. pattern recognition is a general term to describe a wide range of problems like recognition, description, classification, and grouping of patterns. Pattern recognition is about guessing or predicting the unknown nature of an observation, a discrete quantity such as black or white, one or zero, sick or healthy, real or fake. Watanabe defined a pattern as “opposite of a chaos; it is an entity, vaguely defined, that could be given a name.” For example, a pattern could be a fingerprint image, a handwritten word, a human face, or a speech signal. The pattern recognition problems are important in a variety of engineering and scientific disciplines such as biology, psychology, medicine, marketing, artificial intelligence, computer vision and remote sensing.

The field of pattern recognition is concerned mainly with the description and analysis of measurements taken from physical or mental processes. It consists of acquiring raw data and taking actions based on the “class” of the patterns recognized in the data. Earlier it was studied as a specialized subject due to higher cost of the hardware for acquiring the data and to compute the answers. The fast developments in computer technology and resources enhanced possible various practical applications of pattern recognition, which in turn contributed to the demands for further theoretical developments.

The design of a pattern recognition system essentially involves the following three aspects: data representation, Classification and finally, Prototyping. The problem domain dictates the choice of sensors, pre-processing techniques, representation scheme, and decision making model.

- i. **Representation** - It describes the patterns to be recognized;
- ii. **Classification** - It recognizes the “category” to which the patterns provided belong to;
- iii. **Prototyping** - It is the mechanism used for developing the prototypes or models. Prototypes are used for representing the different classes to be recognized.

A general pattern recognition system is shown in the Figure 1.1. In the first step data is acquired and preprocessed, this step is followed by feature extraction, feature reduction and grouping of features, and finally the features are classified. In the classification step, the trained classifier assigns the input pattern to one of the pattern classes based on the measured features. The training set used during construction of the classifier is different from the test set which is used for evaluation. This ensures different performance environment.

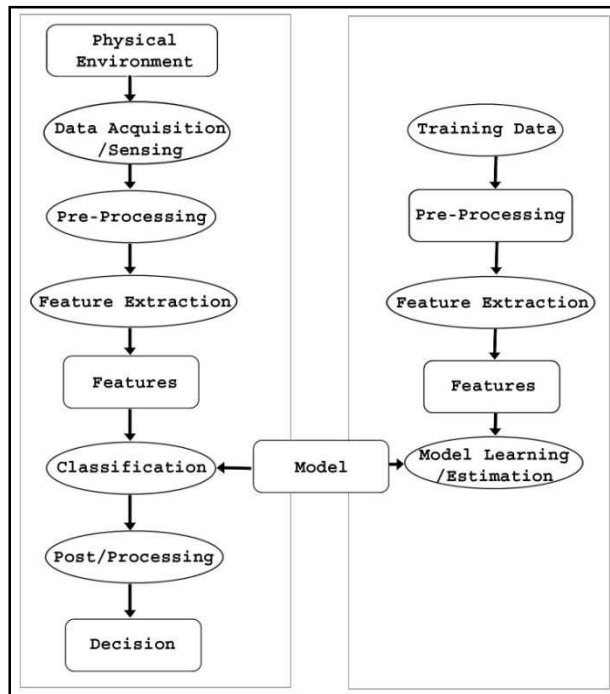


Figure 1.1: A general pattern recognition system

1.3 Pattern Recognition approaches

Patterns generated from the raw data depend on the nature of the data. Patterns may be generated based on the statistical feature of the data. In some situations, underlying structure of the data decides the type of the pattern generated. In some other instances, neither of the two situation exits. In such scenarios a system is developed and trained for desired responses. Thus, for a given problem one or more of these different approaches may be used to obtain the solution. Hence, to obtain the desired attributes for a pattern recognition system, there are many different mathematical techniques. The four best-known approaches for the pattern recognition are:

1. Template matching
2. Statistical classification
3. Syntactic matching
4. Neural networks

In template matching, the prototype of the pattern to be recognized is compared against the pattern to be recognized. In the statistical approach, the patterns are described as random variables, from which class densities can be inferred. Classification is done based on the statistical modeling of data. In the syntactic approach, a pattern is seen as being composed of simple sub-patterns which are themselves built from yet simpler sub-patterns, the simplest being the primitives. Inter relationships between these primitive patterns are used to represent a more complex pattern. The neural network approach to pattern recognition is strongly related to the statistical methods, since they can be regarded as parametric models with their own learning scheme.

The models proposed need not be independent and sometimes the same pattern recognition method exists with different interpretations. A hybrid system may be built involving multiple models. The comparison of different approaches is summarized in Table 1.1.

Table 1.1: Pattern Recognition Models

Approach	Representation	Recognition Function	Typical Criterion
Template Matching	Samples, pixels, Curves	Correlation, distance measure	Classification error
Statistical	Features	Discriminant Function	Classification error
Syntactic or Structural	Primitives	Rules, grammar	Acceptance error
Neural networks	Samples, pixels, features	Network function	Mean square error

1.3.1 Template matching

One of the simplest and earliest approaches to pattern recognition is based on template matching. Matching is carried out to determine the similarity between two entities such as points, curves, or shapes of the same type. In template matching, a template or a prototype of the pattern to be recognized is available. The pattern to be recognized is matched against the stored template while taking into account all allowable operations such as translation, rotation and scale changes. The similarity measure, often a correlation, may be optimized based on the available training set. Often, the template itself is learned from the training set. Template matching is computationally demanding. Present day computers with higher computation power, due to their faster processors, has made this approach more feasible. The rigid template matching even though effective in some application domains has a number of disadvantages. For example, it would fail if the patterns are distorted due to the imaging process, viewpoint change, or large intra-class variations among the patterns. When the deformation cannot be easily explained or modeled directly, deformable template models or rubber sheet deformations can be used to the match patterns.

1.3.2 Statistical Pattern Recognition

The statistical pattern recognition approach assumes statistical basis for classification of data. It generates random parameters that represent the properties of the pattern to be recognized. The main goal of statistical pattern classification is to find to which category or class a given sample belongs. Statistical methodologies such as statistical hypothesis testing, correlation and Bayes classification are used for implementing this method. The effectiveness of the representation is determined by how well pattern from different classes are well separated.

To measure the nearness of the given sample with one of the classes, statistical pattern recognition uses probability of error. Bayesian classifier is a natural choice in applying statistical methods to pattern recognition. However, its implementation is often difficult due to the complexity of the problems and especially when the dimensionality of the system is high. One can also consider simpler solution such as a parametric classifier based on assumed mathematical forms such as linear, quadratic or piecewise. Initially a parametric form of the decision boundary is specified; then the best decision boundary of the specified form is found based on the classification of training samples. Another important issue concerned with statistical pattern recognition is the estimation of the values of the parameters since they are not given in practice. In these systems it is always important to understand how the number of samples affects the classifier design and performance.

1.3.3 Syntactic Pattern Recognition

In many situations there exist interrelationship or interconnection between the features associated with a pattern. In such circumstances it is appropriate to assume a hierarchical relationship where a pattern is viewed as being consist of simple sub patterns which are themselves built with yet another sub pattern. This is the basis of Syntactic pattern recognition. In this method symbolic data structures such as arrays, strings, trees, or graphs are used for pattern representation. These data structures define the relations between fundamental pattern components and allow the representation of hierarchical models. Thus complex patterns can be represented from simpler ones. The recognition of an unknown pattern is accomplished by comparing its symbolic representation with a number of predefined objects. This comparison helps to compute the similarity measurement between the unknown input and with known patterns.

The symbolic data structures used for the representation of the patterns are represented by words of symbols or strings. The individual symbols in a string usually represent components of the atomic pattern. The strings are however one-dimensional in nature but many patterns are inherently two or more dimensional. One of the most used and powerful symbolic structure for higher dimensional data representation is a graph. A graph is composed of a set of nodes and a set of edges in which the nodes represent simpler sub-patterns and the edges the relations between those sub-patterns. These relations may be spatial, temporal or of any other type, depending on the problem. An important subclass of a graph is a tree. A tree has three different classes of nodes, which are root, interior and leaf. Trees are intermediate between strings and graphs. They are interesting for pattern recognition applications since they are more powerful than strings as a representation of the object and computationally less expensive than graphs. Another form of symbolic representation is the array which is a special type of graph which has the nodes and edges arranged in a regular form. This type of data structure is very useful for low level pattern representation.

Structural pattern recognition is found to be good because it provides a description of how the given pattern is constructed from the primitives in addition to classification. This method is useful in situations where the patterns have a definite structure which can be captured in terms of a set of rules. However, due to parsing difficulties the implementation of a syntactic approach is limited. It is very difficult to use this method for segmentation of noisy patterns and another problem is inference of the grammar from training data. Powerful pattern recognition capabilities can be achieved by combining the syntactic and statistical pattern recognition techniques [Fu 1986].

1.3.4 Neural Network

Neural computing is based on the way by which biological neural system store and manipulates information. It can be viewed as parallel computing environment consisting of interconnection of large number of simple processors. Neural network have been successfully applied in many tasks of pattern recognition and machine learning systems. The structure of neural system is drawn from analogies with biological neural systems. Many algorithms have been designed to work with neural network learning have been developed. In these algorithms, a set of rules defines the evolution process undertaken by the synaptic

connections of the networks, thus allowing them to learn how to perform specified tasks. Neural network models use a network of weighted directed graphs in which the nodes are artificial neurons and directed edges are connections between neuron outputs and neuron inputs. The neural networks have the ability to learn complex nonlinear input-output relationships, use sequential training procedures, and adapt themselves to the data.

Different types of neural networks are used for pattern classification. Among them Feed-forward network and Kohonen-Network is commonly used. The learning process involves updating network architecture and connection weights so that a network can efficiently perform a specific classification/clustering task. The neural network models are gaining popularity because of their ability to solve pattern recognition problems, seemingly low dependence on domain-specific knowledge, and due to the availability of efficient learning algorithms for practitioners to use. Neural networks are also useful for implementing nonlinear algorithms for feature extraction and classification. In addition, existing feature extraction and classification algorithms can also be mapped on neural network architectures for efficient implementation. In spite of the seemingly different underlying principles, most of the well-known neural network models are implicitly equivalent or similar to classical statistical pattern recognition methods.

1.4 Feature Extraction and Reduction

Feature selection is the process of choosing input to the pattern recognition system. Many methods can be used to extract the features. The feature selected is such that it is relevant to the task at hand. These features can be obtained from the mathematical tools or by applying feature extraction algorithm or operator to the input data. The level at which these features are extracted determines the amount of necessary preprocessing and may influence the amount of error introduced into the feature extracted. Features may be represented as continuous, discrete, or discrete binary variables. During the features extraction phase of the recognition process objects are measured. A measurement is the value of some quantifiable property of an object. A feature is a function of one or more measurements, computed so that it quantifies some significant characteristic of the object. This process produces a set of features that, taken together, forms the feature vector.

A number of transformations can be used to generate features. The basic idea is to transform a given set of measurements to a new set of features. Transformation of features

can lead to a strong reduction of information as compared with the original input data. In most of the situations relatively small number of features is sufficient for correct recognition. Obviously feature reduction is a sensitive procedure since if the reduction is done incorrectly the whole recognition system may fail or may not produce the expected results. Examples of such transformations are the Fourier transform, Empirical mode decomposition, and the Haar transform. Feature generation via linear transformation techniques is just one of the many possibilities. Feature extraction also depends on application in hand and may use different techniques such as moment-based features, chain codes, and parametric models to obtain required features.

1.5 Cluster Analysis

The main objective in clustering techniques is to partition a given data set into homogeneous clusters. The term homogeneous is used in the sense that all points in the same group are similar to each other and are not similar to points in other groups. The similarity of these points is defined according to some established criteria.

While the use of clustering in pattern recognition and image processing is relatively recent, cluster analysis is not a new field. It has been used in other disciplines, such as biology, psychology, geology and information retrieval. The majority of the clustering algorithms find clusters of a particular shape. Most of the real problems involve clustering in higher dimension. And the difficulties with the natural interpretation of data embedded in a high dimensional space are evident. Clustering method is a very active field in pattern recognition and data mining. Thus a large amount of clustering algorithms continues to appear in the literature. Most of these algorithms are based on proximity measures. Even though, there are a class of algorithm based on different combinations of a proximity measure and a clustering scheme. Clustering is a major tool used in a number of applications, which can be basically used in four different ways namely data reduction, hypothesis generation, hypothesis testing and prediction based on group.

1.6 Classifiers Design

Classifiers are designed to perform the classification stage of the pattern recognition system. A Classifier partitions the feature space into different regions. The border of each decision region is a decision boundary. The determination of region to which the feature vector belongs to is a challenging task. There are many approaches for the design of the

classifier in a pattern recognition system and they can be grouped in three classes: classifiers based on Bayes decision theory, linear and nonlinear classifiers.

The first approach builds upon probabilistic arguments stemming from the statistical nature of the generated features. This is due to the statistical variation of the patterns as well as to possible noise obtained in the signal acquisition phase. The objective of this type of design is to classify an unknown pattern in the most probable class as deduced from the estimated probability density functions. Even though linear classifiers are more restricted in their use, the major advantage is their simplicity and computational demand in solving problems which do not require more sophisticated nonlinear model. Examples of linear classifiers are the perceptron algorithm and least squares methods. For problems that are not linearly separable and for which the design of a linear classifier, even in an optimal way, does not lead to satisfactory performance, the use of nonlinear classifier are mandatory.

1.7 Importance and Applications

The progress of society from the era of industrial revolution to knowledge based era has created a need for faster and more reliable information handling and retrieval systems. Automation in industrial production and efficient management processes are gained much importance. With the advent in the Internet and information technology has made the manufacturing sector to reach any part of the globe. These tendencies have pushed pattern recognition to the high edge of computer and engineering research and applications. Today pattern recognition is an integral part in most machine intelligence systems design for decision making task which are used in a variety of applications such as artificial intelligent system and image understanding and analysis.

Nowadays the interest in the area of pattern recognition comes from applications such as data mining, document classification, biometrics, financial forecasting, and computer vision. Table 1.2 gives some more examples of applications in different domains. A common characteristic of a number of these applications is that the available features are usually not suggested by domain experts, but must be extracted and optimized by data-driven procedures. It is necessary to note that there is no simple approach for optimal solutions and that multiple methods and approaches need to be used. Accordingly, several classifiers are combined together to obtain better result in pattern recognition systems.

Table 1.2: Examples of different pattern recognition applications

Problem domain	Application	Input Pattern	Pattern Classes
Bio-informatics	Sequence analysis	DNA/Protein sequence	Known types of genes/patterns
Data Mining	Searching for required patterns	Points in multidimensional space	Compact and well separated clusters
Document classification	Internet search	Text document	Semantic categories (e.g. sports)
Document image analysis	Reading machine for blind	document image	Alphanumeric characters, words
Industrial automation	Printed circuit board inspection	Intensity or range image	Defective / non-defective nature of product
Multimedia database retrieval	Internet search	Video clip	Video genres (e.g. action, dialogue, etc.)
Biometric recognition	Personal identification	Face, iris, fingerprint	Authorized users for access control
Remote sensing	Forecasting weather, crop yield	Multispectral image	Land use categories, growth pattern of crops
Speech recognition	Speaker identification	Speech waveform	Spoken words
Medicine	Disease identification	Scanned image	Diseased areas in the body

Machine vision for example is an area in which pattern recognition is of clear importance. A machine vision system acquires images through a camera, these signals are analyzed so to produce a description and categorization of objects in the image. Typical

application of this type is desirable in the manufacturing industry for automated visual inspection or automation in the assembly line.

Character recognition is another important application in the area of pattern recognition, with major implications in automation and information handling. Optical character recognition (OCR) systems consist in a scanning device and pattern recognition software that translates the scanned imaged into computer coded characters. The advantage of storing the recognized document are clear since it is more efficient to store ASCII characters than a document image, also it turns possible further electronic processing. There is a great interest in systems that recognize handwritten characters besides the machine printed character recognition systems. A typical commercial application of such system is machine reading of bank checks. Another application lies in automatic mail sorting machines for postal code identification in post offices. On-line handwritten recognition systems are another area of great commercial interest. Such system would accompany pen computers and greatly improve human computer interface.

Recently, there has been a great amount of effort invested in speech recognition systems. Speech is the most natural means by which we communicate and exchange information. The potential application for such a system is numerous. One of the goal of this kind of system is to enter data into a computer via a microphone and a major effort has been done towards this direction with considerable success.

Computer-aided diagnosis is also an important and possible application of pattern recognition systems. The task of these systems would be assisting doctors in making diagnostic decisions. The need for a computer-aided diagnosis came from the fact that medical data are often not so easily interpretable. So an automatic pattern recognition system can assist a doctor with a second opinion.

In addition to the applications described above several other uses of pattern recognition system are of importance such as fingerprint identification, signature authentication, and text retrieval, and face and gesture recognition. The field of pattern recognition still poses some great challenges not just with applied and implementational problems, but also on the theoretical framework.

Parameter Estimation

Density estimation when the density is assumed to be in a specific parametric family. Special cases include maximum likelihood, maximum a posteriori, unbiased estimation, and predictive estimation. See the section on Parameter estimation techniques.

Parameter estimation techniques

Maximum likelihood

A parameter estimation heuristic that seeks parameter values that maximize the likelihood function for the parameter. This ignores the prior distribution and so is inconsistent with Bayesian probability theory, but it works reasonably well.

Maximum A Posteriori

A parameter estimation heuristic that seeks parameter values that maximize the posterior density of the parameter. This implies that you have a prior distribution over the parameter, i.e. that you are Bayesian. MAP estimation has the highest chance of getting the parameter exactly right. But for predicting future data, it can be worse than Maximum Likelihood; predictive estimation is a better Bayesian method for that purpose. MAP is also not invariant to reparameterization;

Unbiased estimation

A parameter estimation procedure based on finding an estimator function that minimizes average error. When the average error is zero then the estimator is "unbiased." The error of the function is averaged over possible data sets, including ones you never observed. The best function is then used to get parameter values. See "Pathologies of Orthodox Statistics".

Predictive estimation

Parameter estimation consistent with Bayesian probability theory. It seeks to minimize the expected "divergence" between the estimated distribution and the true distribution. The divergence is measured by Kullback and Leibler's formula. The distribution which achieves minimum divergence corresponds to integrating out the unknown parameter. Hence predictive estimation can be approximated by averaging over several different parameter choices. See "Inferring a Gaussian distribution", "A Comparison of Scientific and Engineering Criteria for Bayesian Model Selection", Geisser, and Bishop.

Minimum Message Length

A parameter estimation technique similar to predictive estimation but motivated by information theory. Consider compressing the data via a two-part code: the first part is a parameter setting, encoded with respect to the prior, and the second part is the data, encoded with respect to the model with that parameter. Parameters are continuous, and so cannot be encoded exactly---they must be quantized, which introduces error. So we can't choose the parameters which simply compress the data most; we have to choose parameters which compress the data well even if the parameters are slightly modified. The parameter setting which balances this tradeoff between accuracy and robustness is the MML estimate. See

- "Estimation and Inference by Compact Coding", Wallace and Freeman, Journal of the Royal Statistical Society B 49(3):240--265, 1987
- The Computer Journal special issue: MDL vs. MML
- "The Maximum Local Mass estimate"
- "Keeping Neural Networks Simple by Minimizing the Description Length of the Weights"
- Minimum Message Length model selection

Some related methods:

- "Flat Minima"
- "Bayesian backpropagation over I-O functions rather than weights"

Bootstrapping

A technique for simulating new data sets, to assess the robustness of a model or to produce a set of likely models. The new data sets are created by re-sampling with replacement from the original training set, so each datum may occur more than once. See "What are cross-validation and bootstrapping?" and "The Bootstrap is Inconsistent with Probability Theory".

Bagging

Bootstrap averaging. Generate a bunch of models via bootstrapping and then average their predictions. See "Bagging Predictors", "Why does bagging work?", and "Bayesian model averaging is not model combination".

Monte Carlo integration

A technique for approximating integrals in Bayesian inference. To approximate the integral of a function over a domain D , generate samples from a uniform distribution over D and average the value of the function at those samples. More generally, we can use a non-uniform proposal distribution, as long as we weight samples accordingly. This is known as **importance sampling** (which is an integration method, not a sampling method). For Bayesian estimation, a popular approach is to sample from the posterior distribution, even though it is usually not the most efficient proposal distribution. Gibbs sampling is typically used to generate the samples. **Gibbs sampling** employs a succession of univariate samples (a Markov Chain) to generate an approximate sample from a multivariate density. See "Introduction to Monte Carlo methods", "Probabilistic Inference using Markov Chain Monte Carlo Methods", and the Markov Chain Monte Carlo home page. Software includes BUGS and FBM.

Regularization

Any estimation technique designed to impose a prior assumption of "smoothness" on the fitted function. See "Regularization Theory and Neural Networks Architectures".

Expectation-Maximization (EM)

An optimization algorithm based on iteratively maximizing a lower bound. Commonly used for maximum likelihood or maximum a posteriori estimation, especially fitting a mixture of Gaussians. See

- "Expectation-Maximization as lower bound maximization"
- "A Gentle Tutorial on the EM Algorithm"

- "Convexity, Maximum Likelihood and All That"
- "Very Fast EM-based Mixture Model Clustering using Multiresolution kd-trees"

Variational bound optimization

A catch-all term for variations on the EM algorithm which use alternative lower bounds (usually simpler ones). The particular lower bound used by EM can lead to an intractable E-step. With a looser bound, the iterative update is more tractable, at the cost of increasing the number of iterations. Another approach, though less often used, is to use a tighter bound, for faster convergence but a more expensive update. See "An introduction to variational methods for graphical models", "Notes on variational learning", "Exploiting tractable substructures in intractable networks".

Variational bound integration

To approximate the integral of a function, lower bound the function and then integrate the lower bound. Not to be confused with Jensen bound integration. **Variational Bayes** applies this technique to the likelihood function for integrating out parameters. The EM bound can be used for this, or any of the simpler bounds used for variational bound optimization. See

- "Using lower bounds to approximate integrals"
- "Variational Bayes for 1-dimensional mixture models"
- "Ensemble Learning for Hidden Markov Models"
- "Inferring parameters and structure of latent variable models by variational Bayes"
- "Bayesian parameter estimation via variational methods"

Jensen bound integration

To approximate the integral of a function, apply Jensen's inequality to turn the integral into a product which lower-bounds the integral. The bound has free parameters which are chosen to make it as tight as possible. Unlike variational bound optimization, the integrand itself does not need to be bounded, and very different answers can result from the two methods. See

- "Ensemble Learning for Multi-Layer Networks"
- "Bayesian Model Selection for Support Vector Machines, Gaussian Processes and Other Kernel Classifiers"
- "Improving the mean field approximation via the use of mixture distributions"

Expectation Propagation

To approximate the integral of a function, approximate each factor by sequential moment-matching. For dynamic systems, it generalizes Iterative Extended Kalman filtering. For Markov nets, it generalizes belief propagation. See A roadmap to research on EP.

Newton-Raphson

A method for function optimization which iteratively maximizes a local quadratic approximation to the objective function (not necessarily a lower bound as in Expectation-

Maximization). If the local approximation is not quadratic, we have a **generalized Newton method**. See "Beyond Newton's method".

Iteratively Reweighted Least Squares

A method for maximum likelihood estimation of a generalized linear model. It is equivalent to Newton-Raphson optimization. See McCullagh&Nelder.

Back-propagation

A method for maximum likelihood estimation of a feed-forward neural network. It is equivalent to steepest-descent optimization. See Bishop.

Backfitting

A method for maximum likelihood estimation of a generalized additive regression. You iteratively optimize each f_i while holding the others fixed. It is equivalent to the Gauss-Seidel method in numerical linear algebra. See Hastie&Tibshirani and "Bayesian backfitting".

Kalman filtering

An algorithm for inferring the next state or next observation of a Linear Dynamical System. By making the state a constant, it can also be used for incrementally building up a maximum-likelihood estimate of a parameter. See "An Introduction to the Kalman Filter" (with links), "Dynamic Linear Models, Recursive Least Squares and Steepest Descent Learning", "From Hidden Markov Models to Linear Dynamical Systems", and Gelb (Ch.4).

Extended Kalman filtering

Kalman filtering applied to general dynamical systems with Gaussian noise. At each step, the dynamical system is approximated with a linear dynamical system, to which the Kalman filter is applied. The linear approximation can be iteratively refined to improve the accuracy of the Kalman filter output. Despite the name, extended Kalman filtering is not really different from Kalman filtering. See Gelb.

Relaxation labeling

An optimization algorithm for finding the most probable configuration of a Markov random field. It generalizes the Viterbi algorithm for Markov chains. Other approaches to this problem include Iterated Complete Modes, simulated annealing, network flow, and variational lower bounds. See "Foundations of Relaxation Labeling Processes" (Hummel and Zucker; appears in Readings in Computer Vision), "Self Annealing: Unifying deterministic annealing and relaxation labeling", "Probabilistic relaxation", and Li.

Deterministic annealing

An optimization technique where the true objective function is morphed into a convex function by a continuous convexity parameter. Start by solving the convex problem and gradually morph to the true objective while iteratively recomputing the optimum. It is called "graduated nonconvexity" in statistical physics, where the convexity parameter often corresponds to temperature. See

- "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems" (Rose, Proc. IEEE Nov 1998)
- "Deterministic Annealing Variant of the EM Algorithm" (Ueda and Nakano, NIPS 7)
- "Statistical Physics, Mixtures of Distributions and the EM Algorithm"
- "Self Annealing: Unifying deterministic annealing and relaxation labeling"

- "Distributional Clustering of English Words"
- "On the Generalization of Deterministic Annealing as Constrained Optimisation"

Boosting

A technique for combining models based on adaptive resampling: different data is given to different models. The idea is to successively omit the "easy" data points, which are well modeled, so that the later models focus on the "hard" data. See Schapire's page, "Additive Logistic Regression: a Statistical View of Boosting", "Prediction Games and Arcing Algorithms", and "Half&Half Bagging and Hard Boundary Points".

Empirical Risk Minimization

A parameter estimation heuristic that seeks parameter values that minimize the "risk" or "loss" that the model incurs on the training data. In classification, a "loss" usually means an error, so it corresponds to choosing the model with lowest training error. In regression, "loss" usually means squared error, so ERM corresponds to choosing the curve with lowest squared error on the training data. It is thus the most basic (and naive) estimation heuristic. This method only uses a loss function appropriate for the problem and does not utilize a probabilistic model for the data. See "Empirical Risk Minimization is an incomplete inductive principle".

Principal Component Analysis

Principle Component Analysis: A statistical technique used to examine the interrelations among a set of variables in order to identify the underlying structure of those variables. Also called *factor analysis*.

It is a non-parametric analysis and the answer is unique and independent of any hypothesis about data distribution.

These two properties can be regarded as weaknesses as well as strengths.

Since the technique is non-parametric, no prior knowledge can be incorporated.

PCA data reduction often incurs a loss of information.

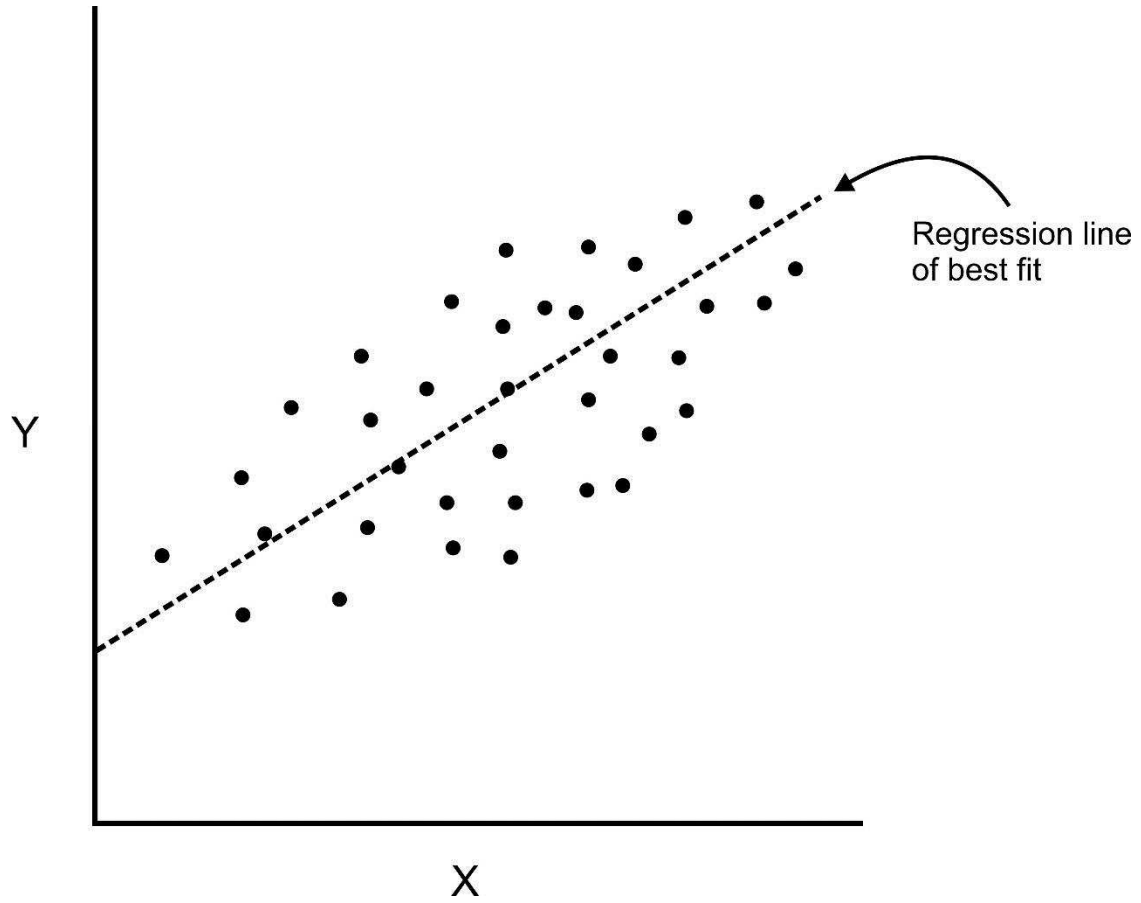
The assumptions of PCA:

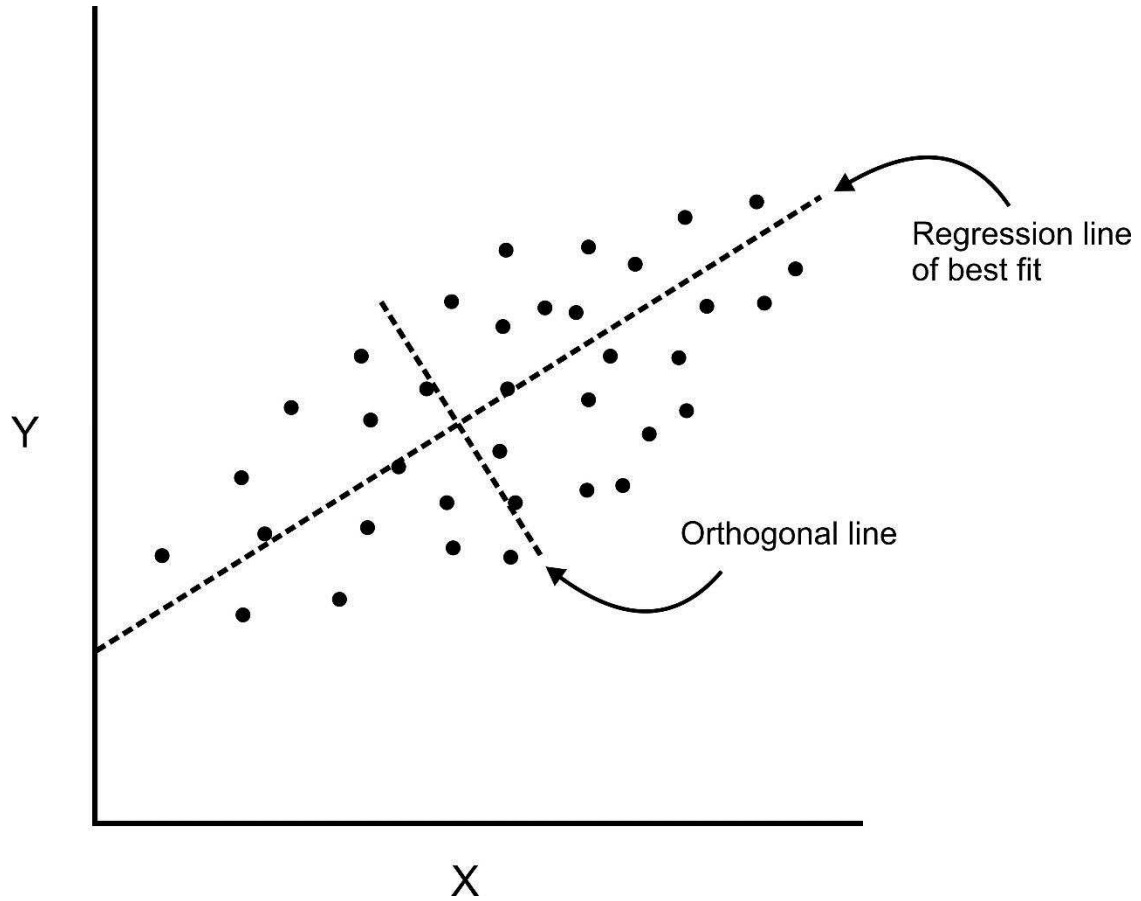
1. Linearity
 - Assumes the data set to be linear combinations of the variables.
2. The importance of mean and covariance
 - There is no guarantee that the directions of maximum variance will contain good features for discrimination.
3. That large variances have important dynamics
 - Assumes that components with larger variance correspond to interesting dynamics and lower ones correspond to noise.

Where regression determines a line of best fit to a data set, factor analysis determines several orthogonal lines of best fit to the data set.

Orthogonal: meaning “at right angles”. Actually the lines are perpendicular to each other in n -dimensional space.

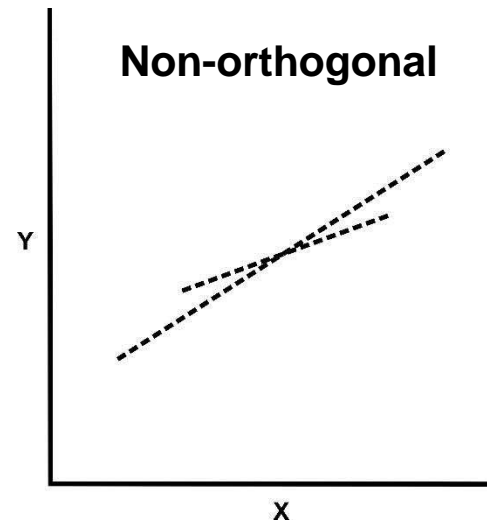
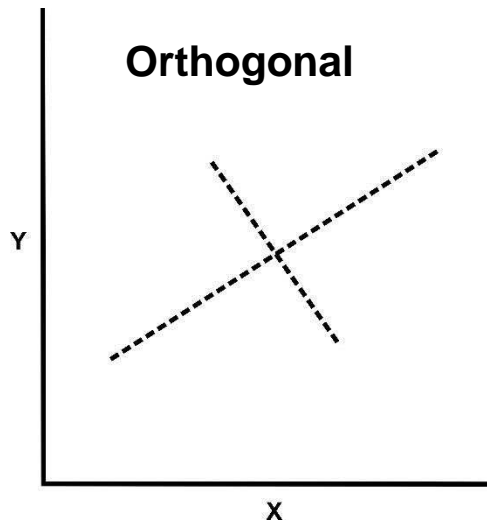
n-Dimensional Space: the variable sample space. There are as many dimensions as there are variables, so in a data set with 4 variables the sample space is 4-dimensional.

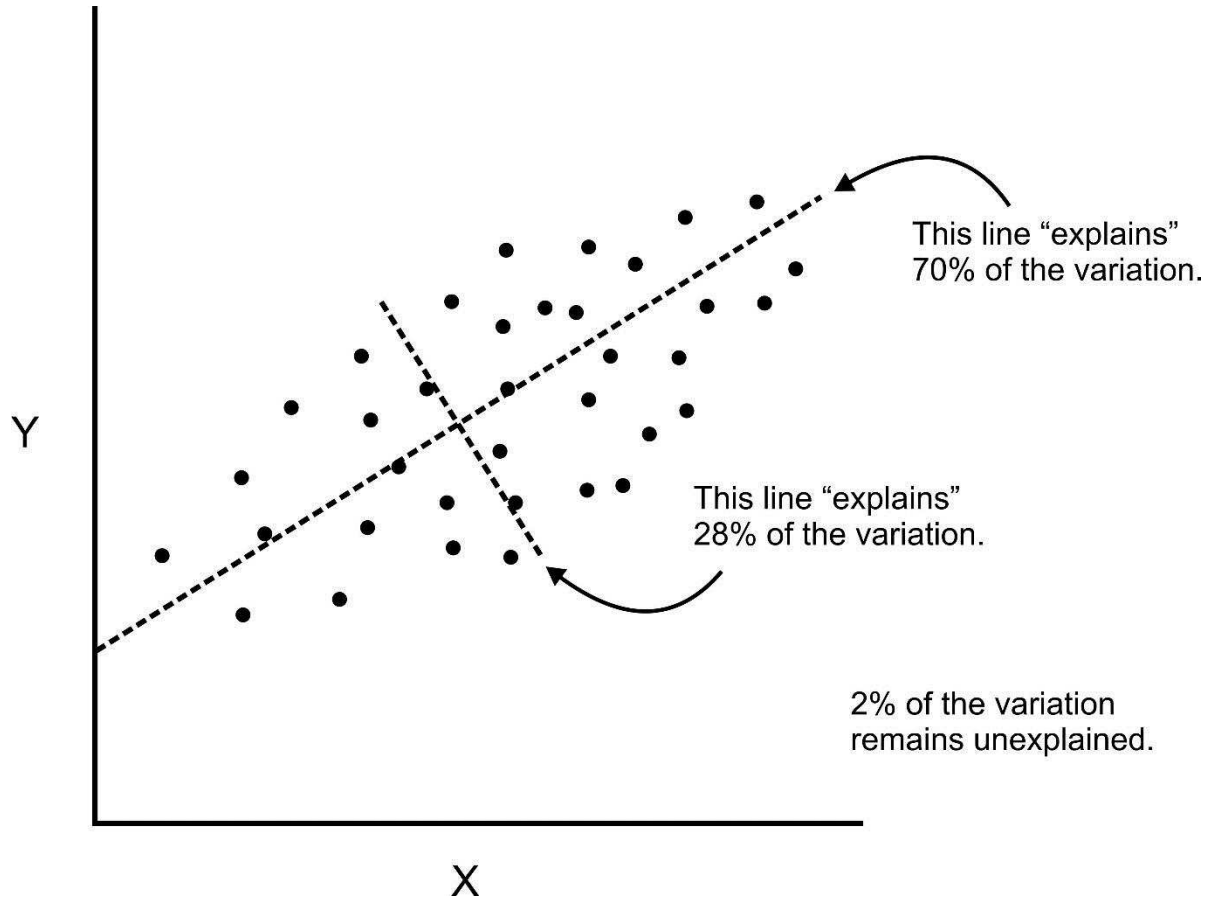




Components: a linear transformation that chooses a variable system for the data set such that the greatest variance of the data set comes to lie on the first axis (then called the *principal component*), the second greatest variance on the second axis, and so on ...

Note that components are uncorrelated, since in the sample space they are orthogonal to each other.

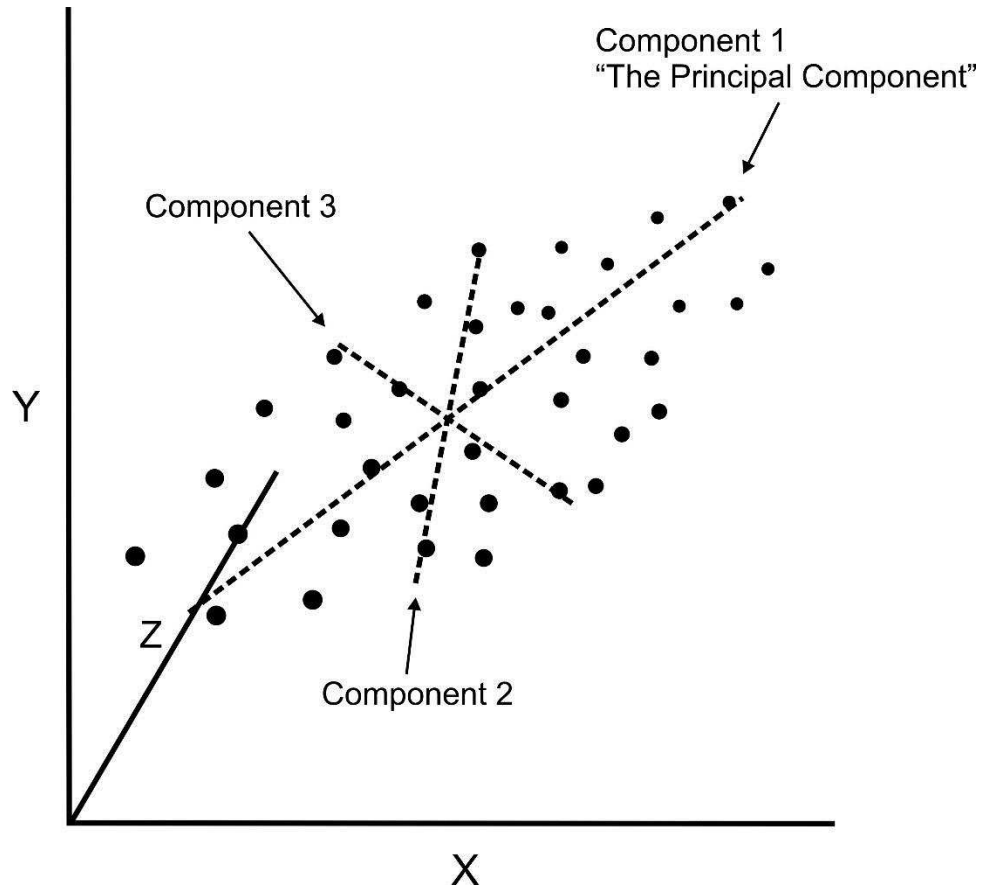




Locations along each component (or *eigenvector*) are then associated with values across all variables. This association between the components and the original variables is called the component's *eigenvalue*.

In multivariate (multiple variable) space, the correlation between the component and the original variables is called the *component loadings*.

Component loadings: analogous to correlation coefficients, squaring them give the amount of explained variation. Therefore the component loadings tell us how much of the variation in a variable is explained by the component.

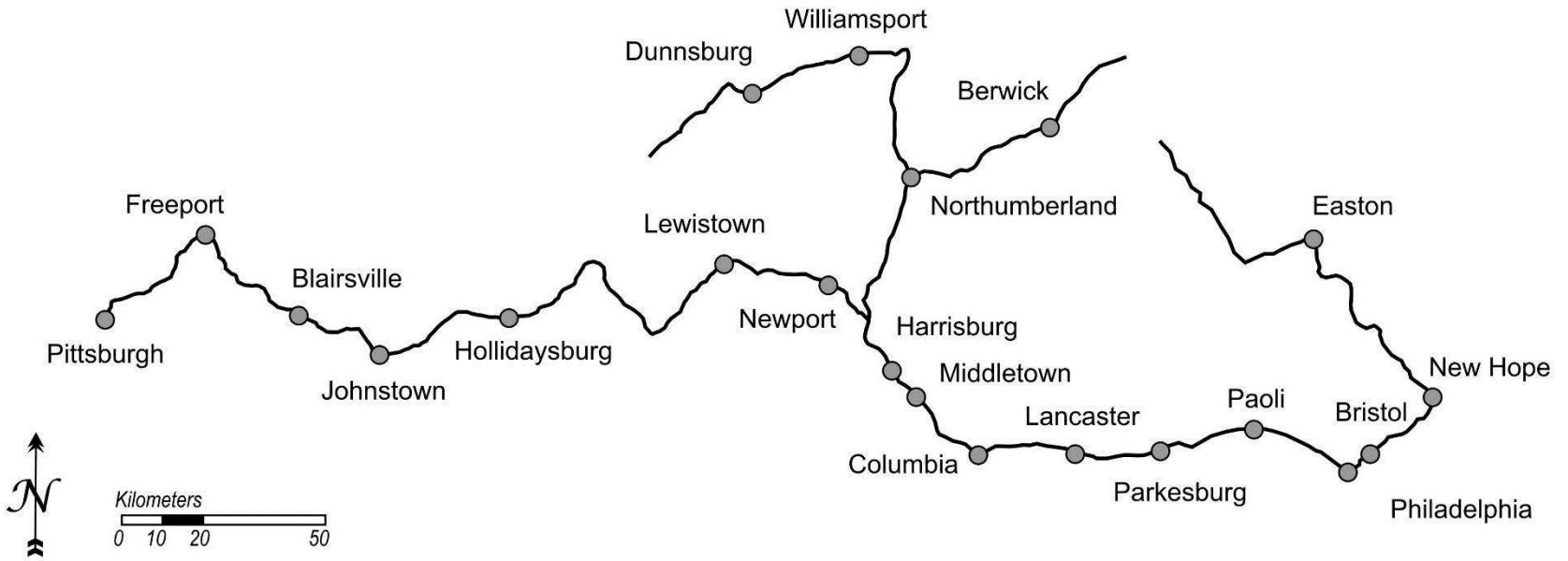


If we use this technique on a data set with a large number of variables, we can compress the amount of explained variation to just a few components.

What follows is an example of Principal Component Analysis using canal town commodity production figures (percentage of total production) for 1845.

The Pennsylvania Canal System

1845



Towns

Columbia
Middletown
Harrisburg
Newport
Lewistown
Hollidaysburg
Johnstown
Blairsville
Pittsburgh
Dunnsburg
Williamsport
Northumberland
Berwick
Easton
New Hope
Bristol
Philadelphia
Paoli
Parkesburg
Lancaster

Variables

Corn
Wheat
Flour
Whiskey
Groceries
Dry Goods

Total Variance Explained

Component	Initial Eigenvalues			Extraction Sums of Squared Loadings			Rotation Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	2.533	42.211	42.211	2.533	42.211	42.211	1.887	31.452	31.452
2	1.565	26.084	68.295	1.565	26.084	68.295	1.880	31.328	62.780
3	1.504	25.073	93.368	1.504	25.073	93.368	1.835	30.587	93.368
4	.174	2.901	96.269						
5	.119	1.988	98.257						
6	.105	1.743	100.000						

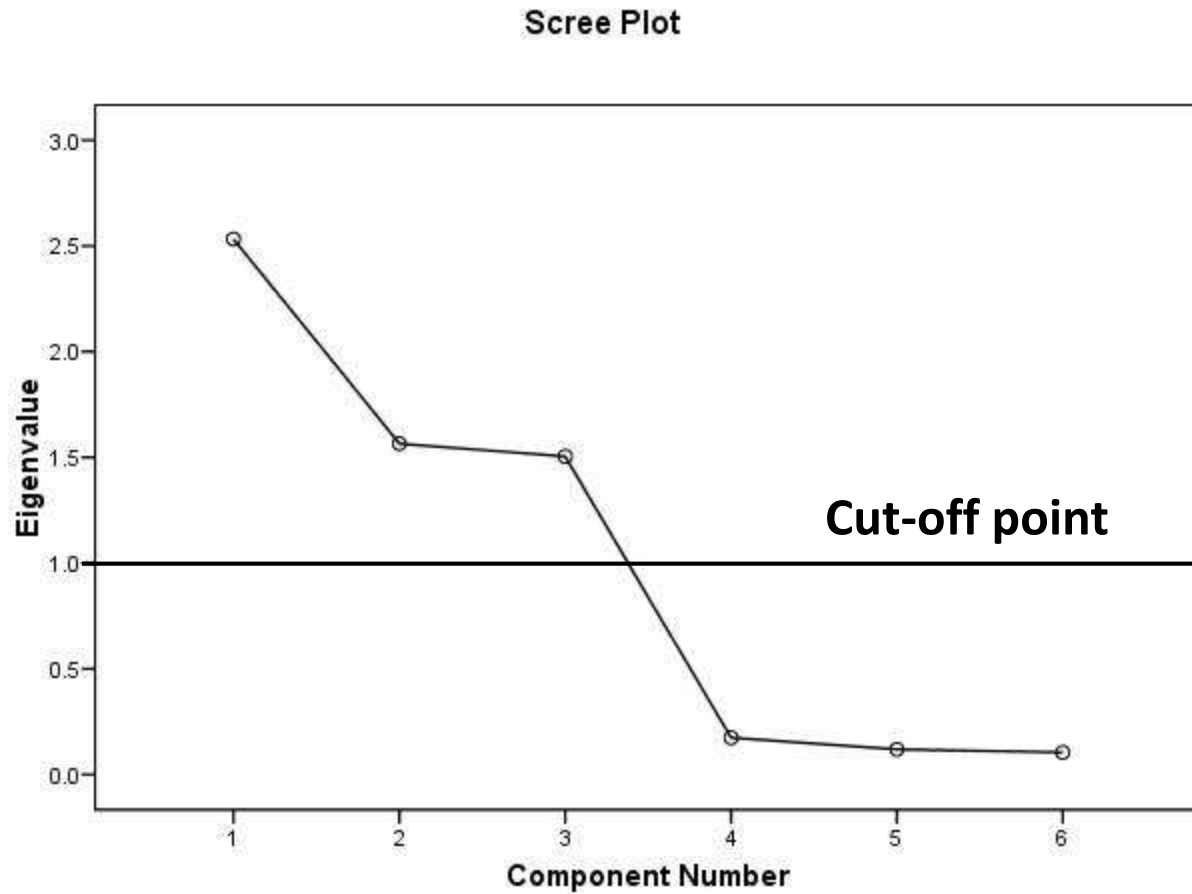
Extraction Method: Principal Component Analysis.

In this case, 3 components contain 93.368% of the variation of the 6 original variables. Note that there are as many components as original input variables.

Component 1 explains 42.211% of the variation, component 2 explains 26.084%, and component 3 explains 25.073%.

The remaining 3 components explain only 6.632%.

A scree plot graphs the amount of variation explained by each component.



Rotated Component Matrix (a)

	Component		
	1	2	3
Corn	-.065	.936	.214
Wheat	-.104	.952	-.057
Groceries	.962	-.092	-.086
Dry Goods	.963	-.074	-.092
Flour	-.126	-.097	.954
Whiskey	-.057	.275	.927

Extraction Method: Principal Component Analysis.
Rotation Method: Varimax with Kaiser Normalization.

a. Rotation converged in 4 iterations.

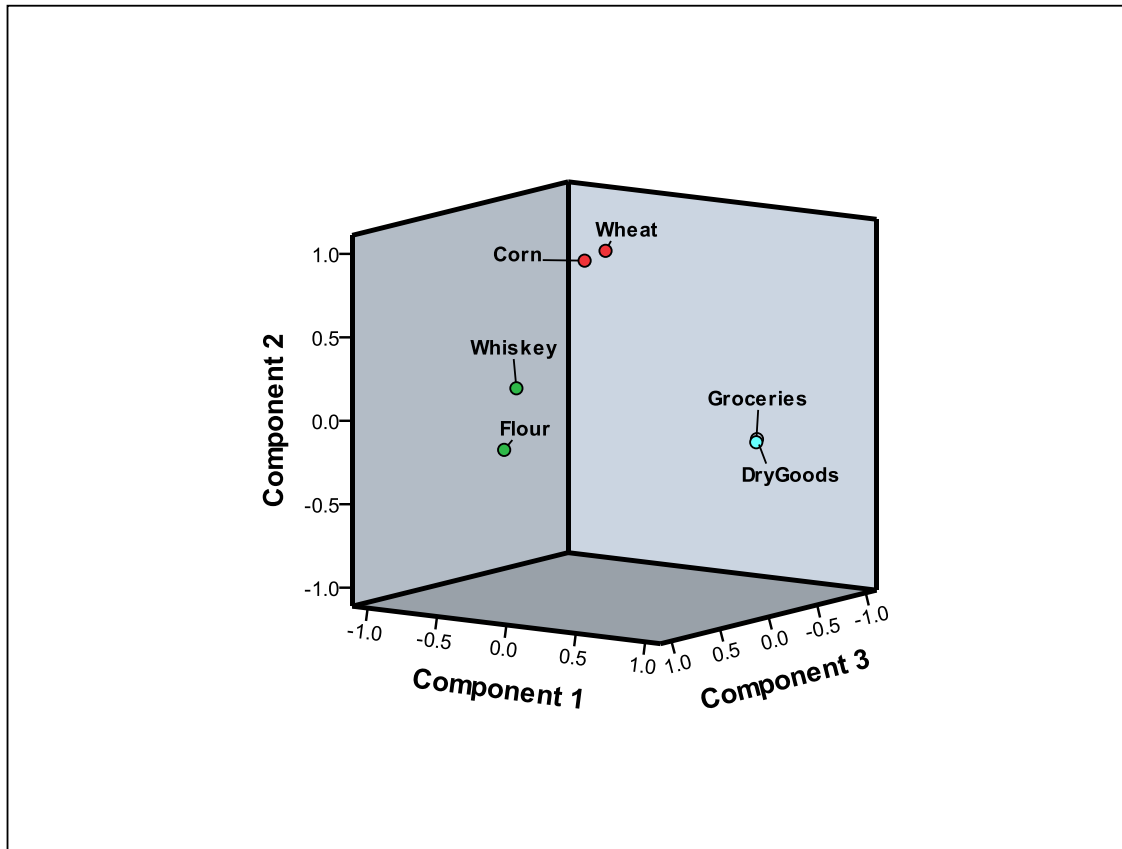
Highest Component Loading

Component 1: Groceries and dry goods.

Component 2: Corn and wheat.

Component 3: Flour and whiskey.

Component Plot in Rotated Space



Note how the variables that make up each component fall close to each other in the 3-dimensional sample space.

What do these components mean (how do we interpret them)?

- *Component 1 (groceries and dry goods)* – these two items are highly processed and value added.
- *Component 2 (corn and wheat)* – these two items are not processed (raw) and have no value added.
- *Component 3 (flour and whiskey)* – these two items are moderately processed and value added.

It appears that the components are indicators of either the amount of processing or value adding (or both).

The most challenging part of PCA is interpreting the components.

1. The higher the component loadings, the more important that variable is to the component.
2. Combinations of positive and negative loadings are interpreted as 'mixed'.
3. The specific sign of the is not important.
4. ALWAYS use the ROTATED component matrix!!

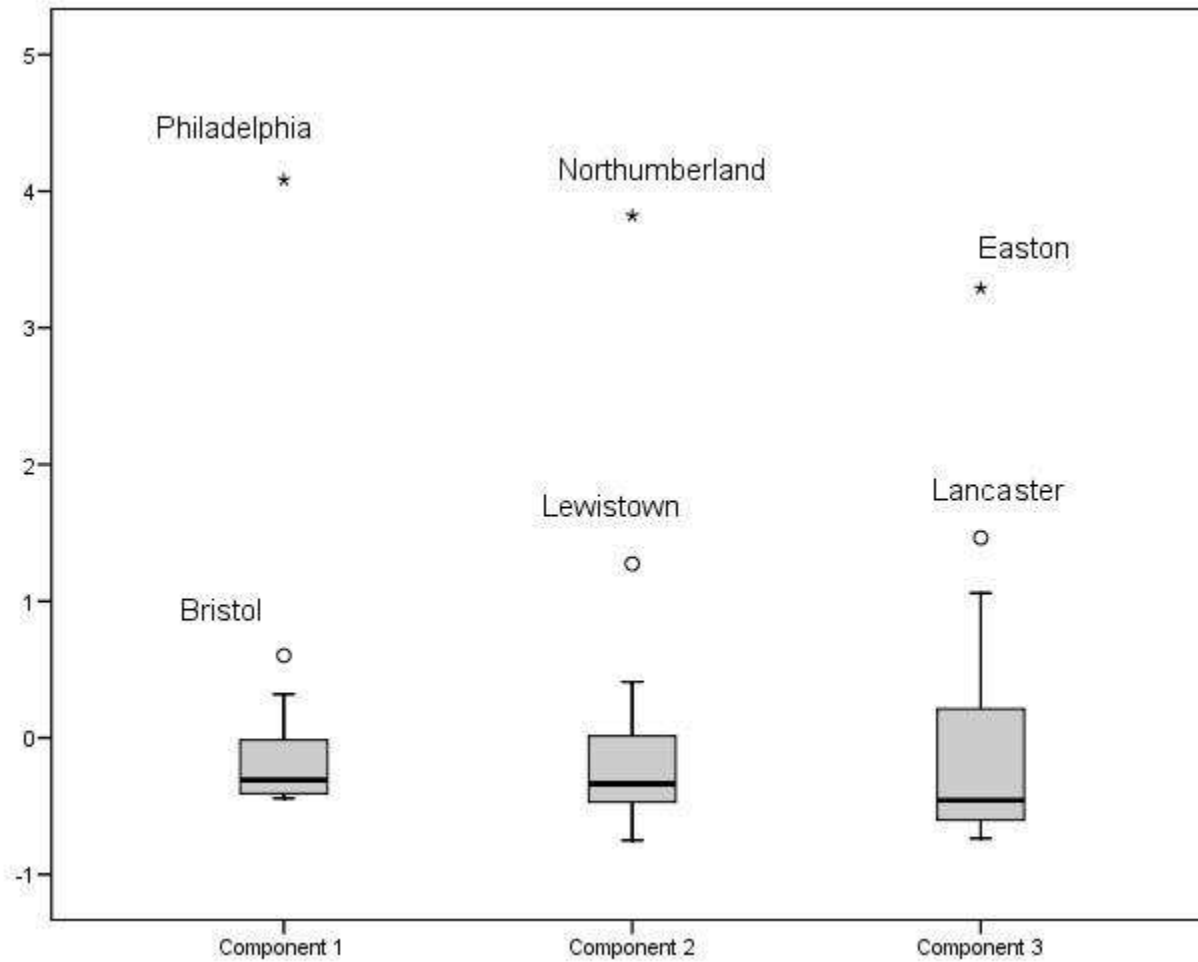
Component score: the new variable value based on the observation's component loading and the standardized value of the original variable, summed over all variables.

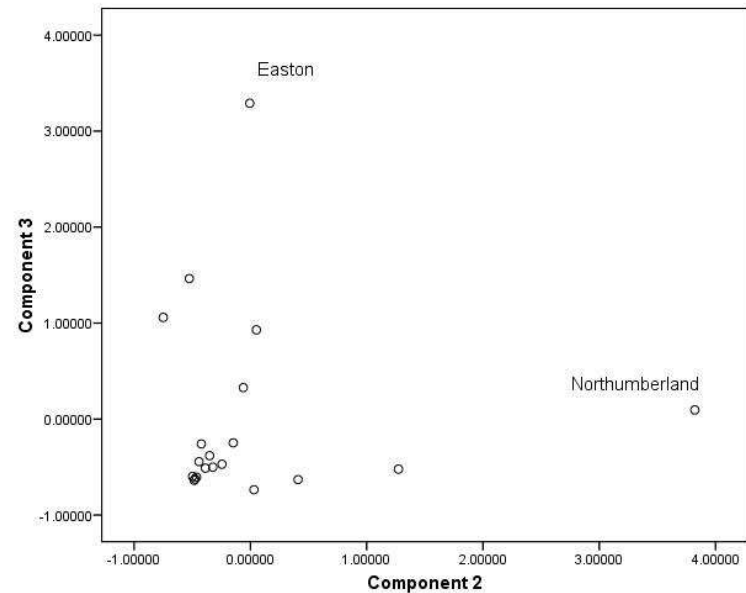
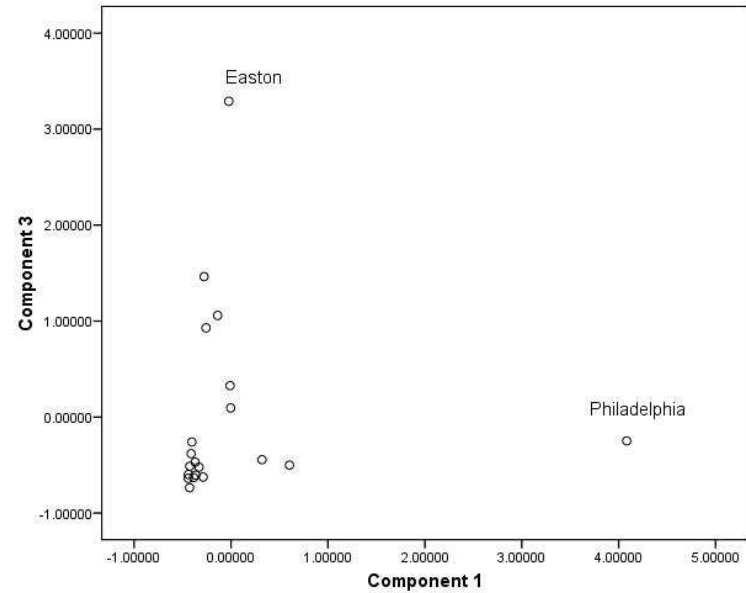
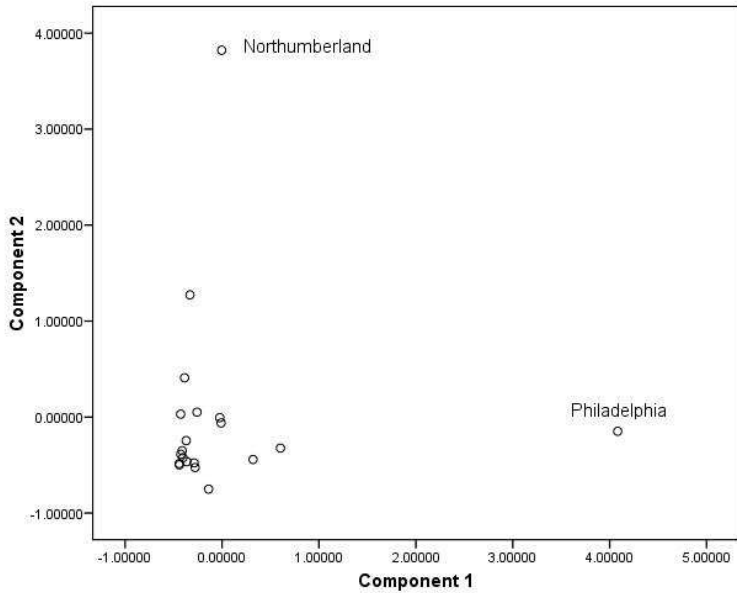
$$\mathbf{Score}_{ik} = \sum \mathbf{D}_{ij} \mathbf{L}_{jk}$$

where D_{ij} is the standardized value for observation i on variable j and L_{jk} is the loading of variable j on component k .

Examining the component scores for each town may give some clues as to the interpretation of the components.

Component Score Box Plot





Easton, Philadelphia, and Northumberland are the only towns that load highly on a single component.

Scoring highly on a single component simply means that the original variable values for these locations are overwhelmingly explained by a single component.

In this case, it means that the variation among ALL of the variables for Philadelphia (for example) is more completely explained by a single component composed of groceries and dry goods.

Rotated Component Matrix^a

	Component		
	1	2	3
Corn	-.065	.936	.214
Wheat	-.104	.952	-.057
Groceries	.962	-.092	-.086
Dry Goods	.963	-.074	-.092
Flour	-.126	-.097	.954
Whiskey	-.057	.275	.927

Extraction Method: Principal Component Analysis.
Rotation Method: Varimax with Kaiser Normalization.

a. Rotation converged in 4 iterations.

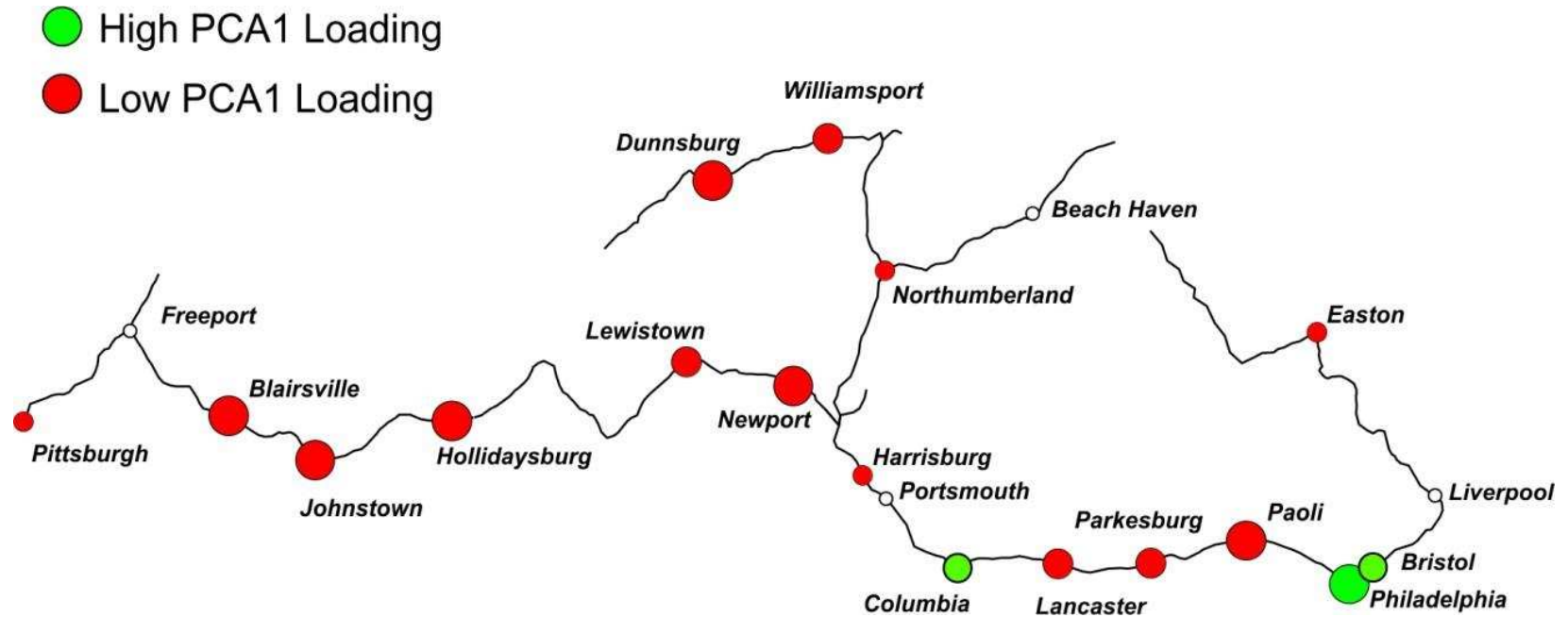
Town Component Scores

Town	Component 1	Component 2	Component 3
Columbia	0.31989	-0.44216	-0.44369
Middletown	-0.37101	-0.24531	-0.47020
Harrisburg	-0.00974	-0.06105	0.32792
Newport	-0.38678	0.40935	-0.62996
Lewistown	-0.33132	1.27318	-0.52170
Hollidaysburg	-0.44018	-0.49770	-0.59722
Johnstown	-0.44188	-0.48447	-0.63736
Blairsville	-0.42552	-0.38759	-0.51107
Pittsburgh	-0.13834	-0.75021	1.05942
Dunnsburg	-0.42728	0.03072	-0.73622
Williamsport	-0.28812	-0.47716	-0.62453
Northumberland	-0.00398	3.82169	0.09538
Berwick	-0.36503	-0.46398	-0.60501
Easton	-0.02349	-0.00587	3.28970
New Hope	-0.40354	-0.42291	-0.25891
Bristol	0.60267	-0.32311	-0.50086
Philadelphia	4.08309	-0.14799	-0.24733
Paoli	-0.41174	-0.35103	-0.38109
Parkesburg	-0.25890	0.05125	0.92910
Lancaster	-0.27880	-0.52566	1.46363

← Middletown is a 'mixed' town because it loads on all components equally.

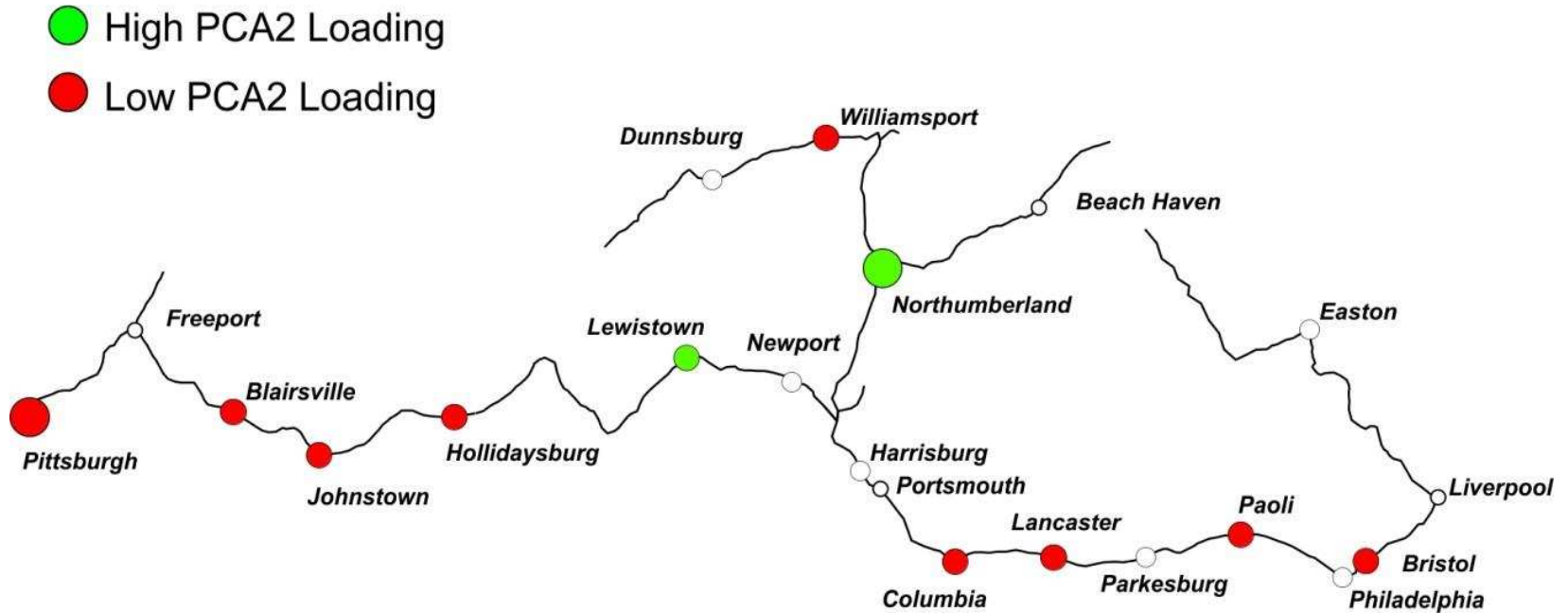
← Philly is a 'processed goods' town.

Component 1: Processed Goods



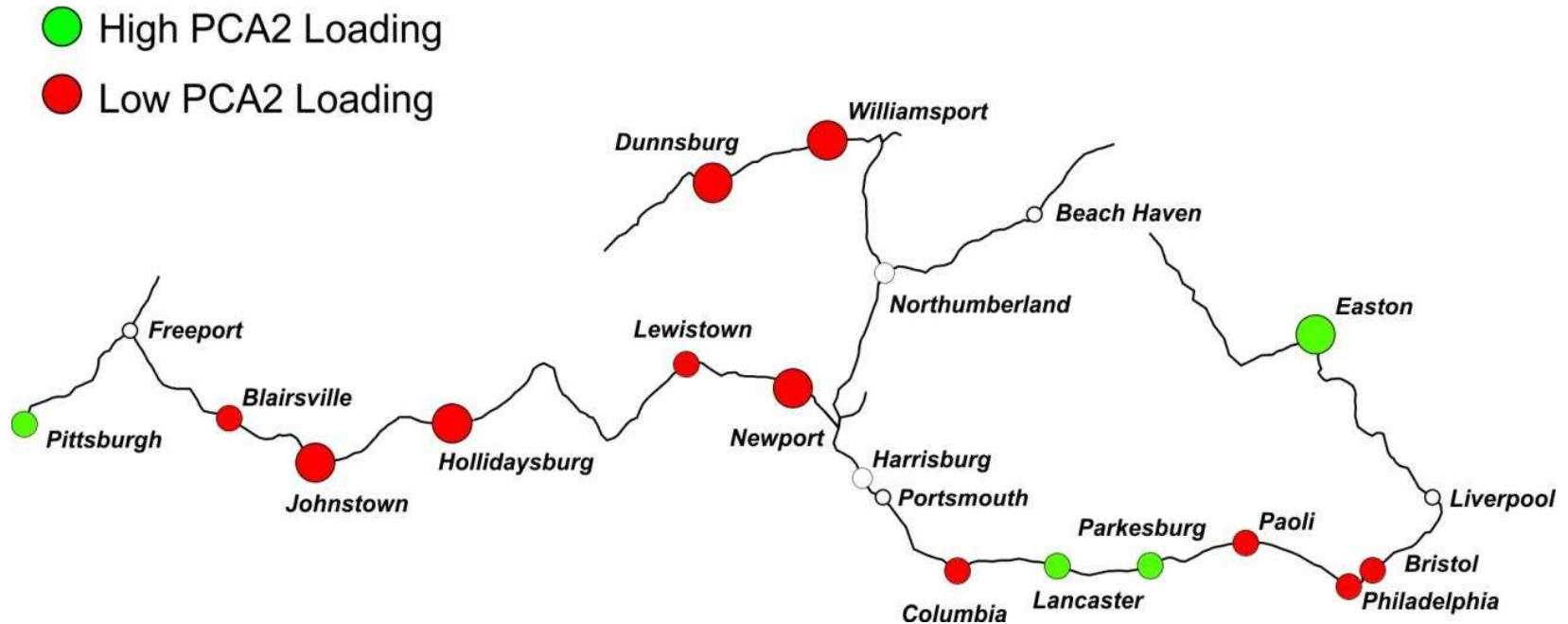
The *green* towns were producers of processed goods, while the *red* towns were consumers of those goods.

Component 2: Non-Processed Goods



The *green* towns were producers of non-processed goods, while the *red* towns were consumers of those goods.

Component 3: Partially Processed Goods



The *green* towns were producers of partially processed goods, while the *red* towns were consumers of those goods.

What information did PCA provide concerning the goods exported by the canal towns?

- The goods fell into recognizable categories (highly processed, moderately processed, not processed).
- A small number of towns were responsible for exporting most of these goods.
- The location of these towns relative to the goods they produced make sense.
 - Industrial towns on the Columbia railroad exported finished goods.
 - Small farming towns on the canal exported produce.
 - Midsize towns exported moderately processed goods.

Without the use of Principal Component Analyses these associations would be difficult to determine.

Principal Component Analyses is also used to remove correlation among independent variables that are to be used in multivariate regression analysis.

Correlation Matrix

	Corn	Wheat	Groceries	DryGoods	Flour	Whiskey
Corn	1.000	.812	-.163	-.160	.108	.450
Wheat	.812	1.000	-.183	-.157	-.096	.198
Groceries	-.163	-.183	1.000	.883	-.191	-.164
DryGoods	-.160	-.157	.883	1.000	-.198	-.163
Flour	.108	-.096	-.191	-.198	1.000	.806
Whiskey	.450	.198	-.164	-.163	.806	1.000

Correlation

	Dry Goods	Groceries	PCA 2	PCA 3
PCA 1	0.963	0.962	0.000	0.000

Note that PCA1 is highly correlated to dry goods and groceries, but uncorrelated to PCA2 and PCA3.

Principal Components Analysis

Covariance

- Variance and Covariance are a measure of the "spread" of a set of points around their center of mass (mean)
- Variance - measure of the deviation from the mean for points in one dimension e.g. heights
- Covariance as a measure of how much **each of the dimensions** vary from the mean **with respect to each other**.
- Covariance is measured between 2 dimensions to see if there is a relationship between the 2 dimensions e.g. number of hours studied & marks obtained.
- The covariance between one dimension and itself is the variance

Covariance

$$\text{covariance}(X, Y) = \frac{\sum_{i=1}^n (\bar{X}_i - \bar{X})(\bar{Y}_i - \bar{Y})}{(n - 1)}$$

- So, if you had a 3-dimensional data set (x, y, z) , then you could measure the covariance between the x and y dimensions, the y and z dimensions, and the x and z dimensions. Measuring the covariance between x and x , or y and y , or z and z would give you the variance of the x , y and z dimensions respectively.

Covariance Matrix

- Representing Covariance between dimensions as a matrix e.g. for 3 dimensions:

$$C = \begin{bmatrix} \text{cov}(x,x) & \text{cov}(x,y) & \text{cov}(x,z) \\ \text{cov}(y,x) & \text{cov}(y,y) & \text{cov}(y,z) \\ \text{cov}(z,x) & \text{cov}(z,y) & \text{cov}(z,z) \end{bmatrix}$$

Variances

- Diagonal is the **variances** of x, y and z
- $\text{cov}(x,y) = \text{cov}(y,x)$ hence matrix is **symmetrical** about the diagonal
- N-dimensional data will result in **NxN covariance matrix**

Covariance

- What is the interpretation of covariance calculations?

e.g.: 2 dimensional data set

x: number of hours studied for a subject

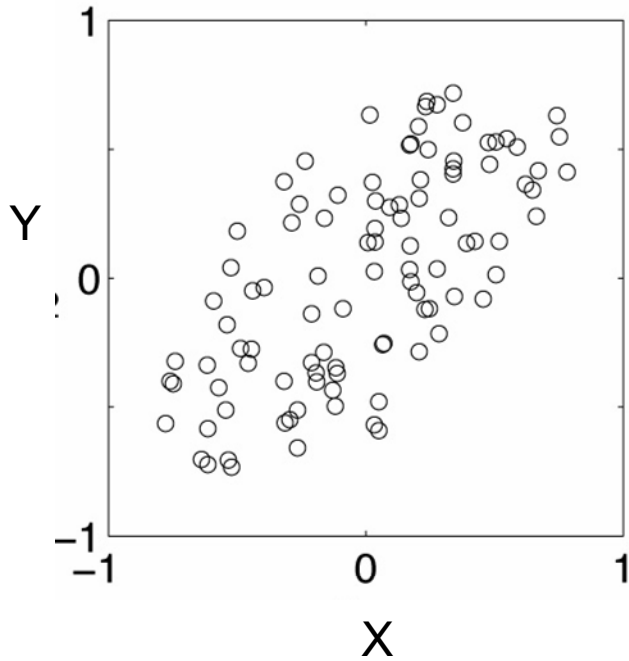
y: marks obtained in that subject

covariance value is say: 104.53

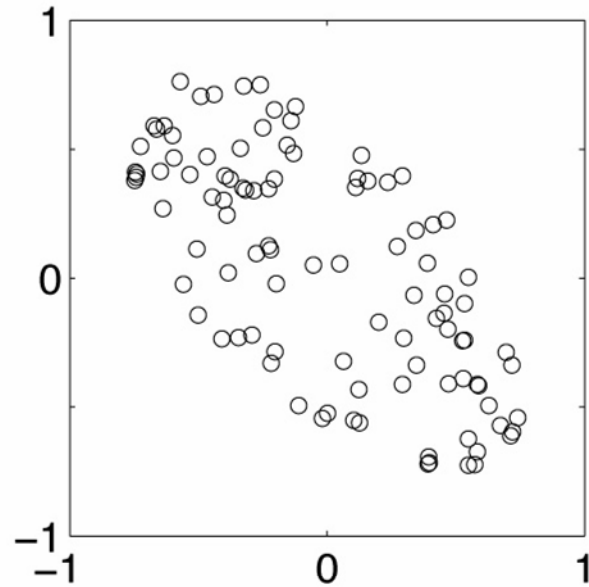
what does this value mean?

Covariance examples

positive covariance



negative covariance



Covariance

- Exact value is not as important as it's sign.
- A positive value of covariance indicates **both dimensions increase or decrease together** e.g. as the number of hours studied increases, the marks in that subject increase.
- A negative value indicates **while one increases the other decreases**, or vice-versa e.g. active social life at PSU vs performance in CS dept.
- If covariance is zero: the two dimensions are **independent** of each other e.g. heights of students vs the marks obtained in a subject

Covariance

- Why bother with calculating covariance when we could just plot the 2 values to see their relationship?

Covariance calculations are used to find relationships between dimensions in high dimensional data sets (usually greater than 3) where visualization is difficult.

PCA

- **principal components analysis (PCA)** is a technique that can be used to **simplify a dataset**
- It is a linear transformation that chooses a new coordinate system for the data set such that
 greatest variance by any projection of the data set comes to lie on the first axis (then called the **first principal component**),
 the second greatest variance on the second axis,
 and so on.
- PCA can be used for **reducing dimensionality** by eliminating the later principal components.

PCA Toy Example

Consider the following 3D points

1	2	4	3	5	6
2	4	8	6	10	12
3	6	12	9	15	18

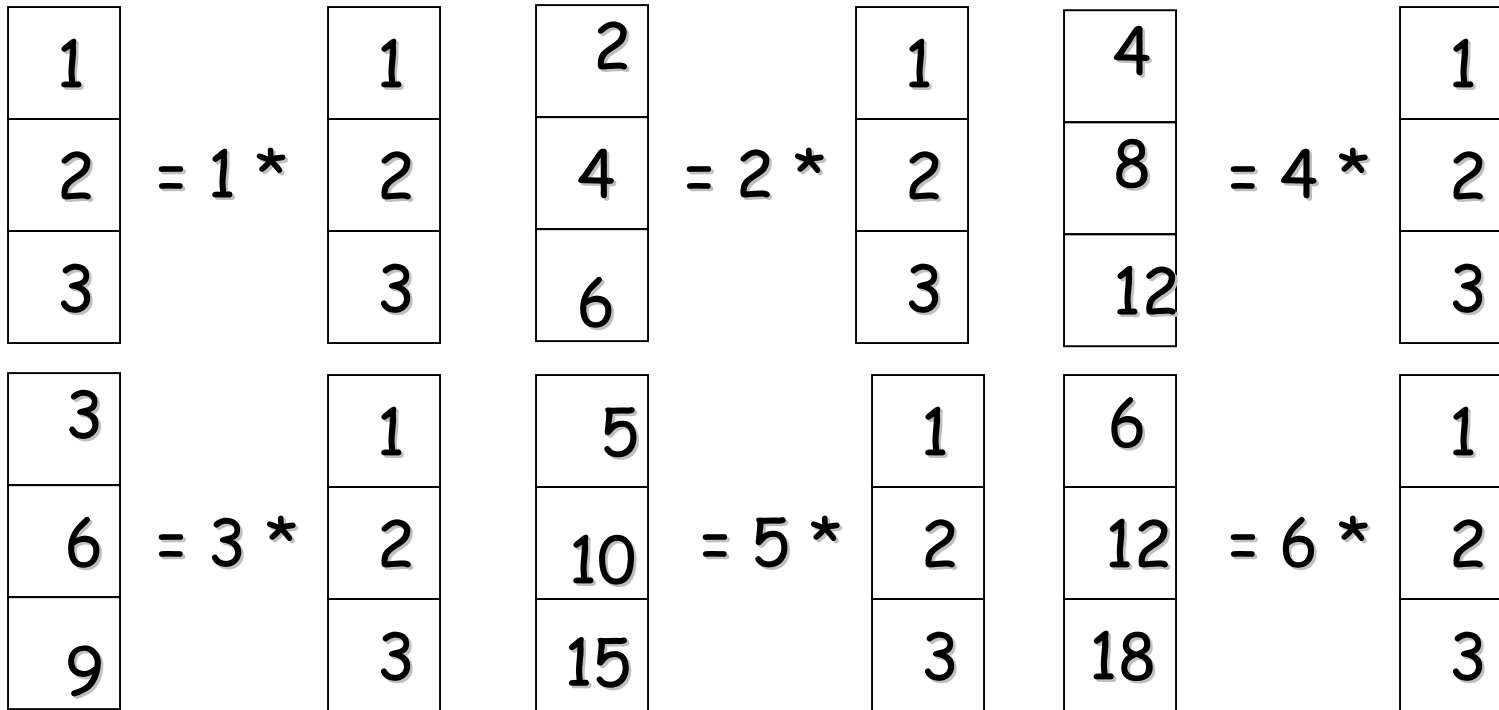
If each component is stored in a byte,
we need $18 = 3 \times 6$ bytes

PCA Toy Example

Looking closer, we can see that all the points are related geometrically: they are all the same point, scaled by a factor:

1		1		2		1		4		1
2	= 1 *	2		4	= 2 *	2		8	= 4 *	2
3		3		6		3		12		3
3		1		5		1		6		1
6	= 3 *	2		10	= 5 *	2		12	= 6 *	2
9		3		15		3		18		3

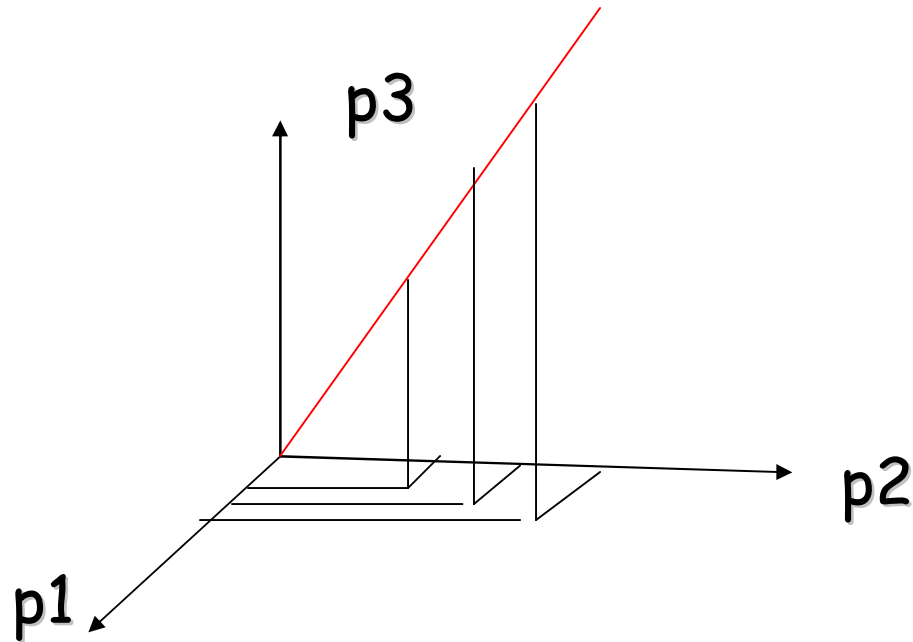
PCA Toy Example



They can be stored using only 9 bytes (50% savings!):
Store one point (3 bytes) + the multiplying constants (6 bytes)

Geometrical Interpretation:

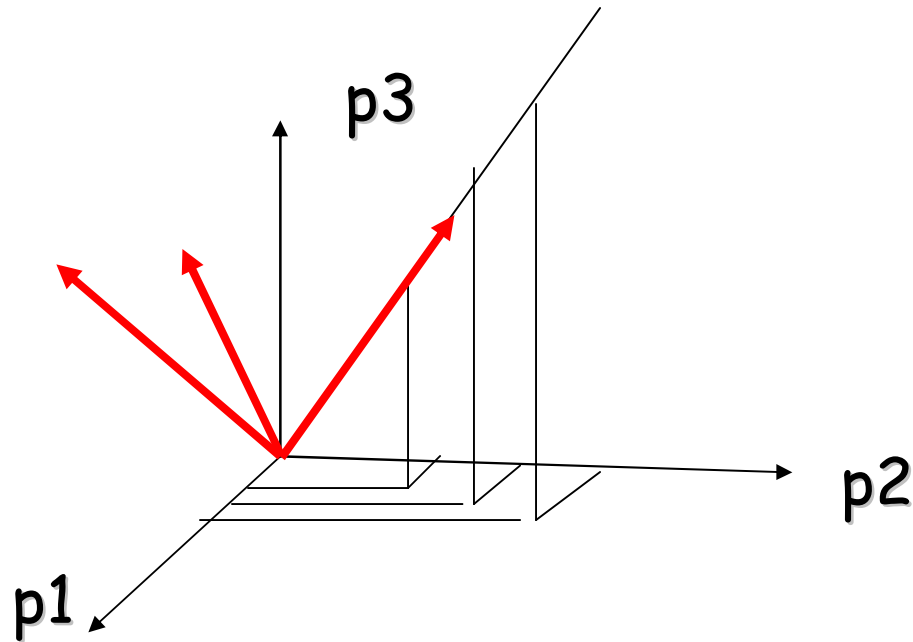
View each point in 3D space.



But in this example, all the points happen to belong to a line: a 1D subspace of the original 3D space.

Geometrical Interpretation:

Consider a new coordinate system where one of the axes is along the direction of the line:



In this coordinate system, every point has only one non-zero coordinate: we only need to store the direction of the line (a 3 bytes image) and the non-zero coordinate for each of the points (6 bytes).

Principal Component Analysis (PCA)

- Given a set of points, how do we know if they can be compressed like in the previous example?
 - The answer is to look into the correlation between the points
 - The tool for doing this is called PCA

PCA

- By finding the **eigenvalues and eigenvectors** of the covariance matrix, we find that the eigenvectors with the largest eigenvalues correspond to the dimensions that have the strongest correlation in the dataset.
- This is the principal component.
- PCA is a useful statistical technique that has found application in:
 - fields such as face recognition and image compression
 - finding patterns in data of high dimension.

PCA Theorem

Let $x_1 x_2 \dots x_n$ be a set of n $N \times 1$ vectors and let \bar{x} be their average:

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix} \quad \bar{x} = \frac{1}{n} \sum_{i=1}^n \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iN} \end{bmatrix}$$

PCA Theorem

Let X be the $N \times n$ matrix with columns $x_1 - \bar{x}, x_2 - \bar{x}, \dots, x_n - \bar{x}$:

$$X = \begin{bmatrix} x_1 - \bar{x} & x_2 - \bar{x} & \cdots & x_n - \bar{x} \end{bmatrix}$$

Note: subtracting the mean is equivalent to translating the coordinate system to the location of the mean.

PCA Theorem

Let $Q = X X^T$ be the $N \times N$ matrix:

$$Q = X X^T = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix}$$

Notes:

1. Q is square
2. Q is symmetric
3. Q is the covariance matrix [aka scatter matrix]
4. Q can be very large (in vision, N is often the number of pixels in an image!)

PCA Theorem

Theorem:

Each x_j can be written as:
$$x_j = \bar{x} + \sum_{i=1}^{i=n} g_{ji} e_i$$

where e_i are the n eigenvectors of Q with non-zero eigenvalues.

Notes:

1. The eigenvectors $e_1 e_2 \dots e_n$ span an *eigenspace*
2. $e_1 e_2 \dots e_n$ are $N \times 1$ orthonormal vectors (directions in N -Dimensional space)
3. The scalars g_{ji} are the coordinates of x_j in the space.

$$g_{ji} = (x_j - \bar{x}) \cdot e_i$$

Using PCA to Compress Data

- Expressing x in terms of $e_1 \dots e_n$ has not changed the size of the data
- However, if the points are highly correlated many of the coordinates of x will be zero or closed to zero.

note: this means they lie in a lower-dimensional linear subspace

Using PCA to Compress Data

- Sort the eigenvectors e_i according to their eigenvalue:

$$\lambda_1 \geq \lambda_2 \geq \dots \lambda_n$$

- Assuming that $\lambda_i \approx 0$ if $i > k$

• Then

$$\mathbf{x}_j \approx \bar{\mathbf{x}} + \sum_{i=1}^{i=k} g_{ji} \mathbf{e}_i$$

PCA Example -STEP 1

<http://kybele.psych.cornell.edu/~edelman/Psych-465-Spring-2003/PCA-tutorial.pdf>

- DATA:

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

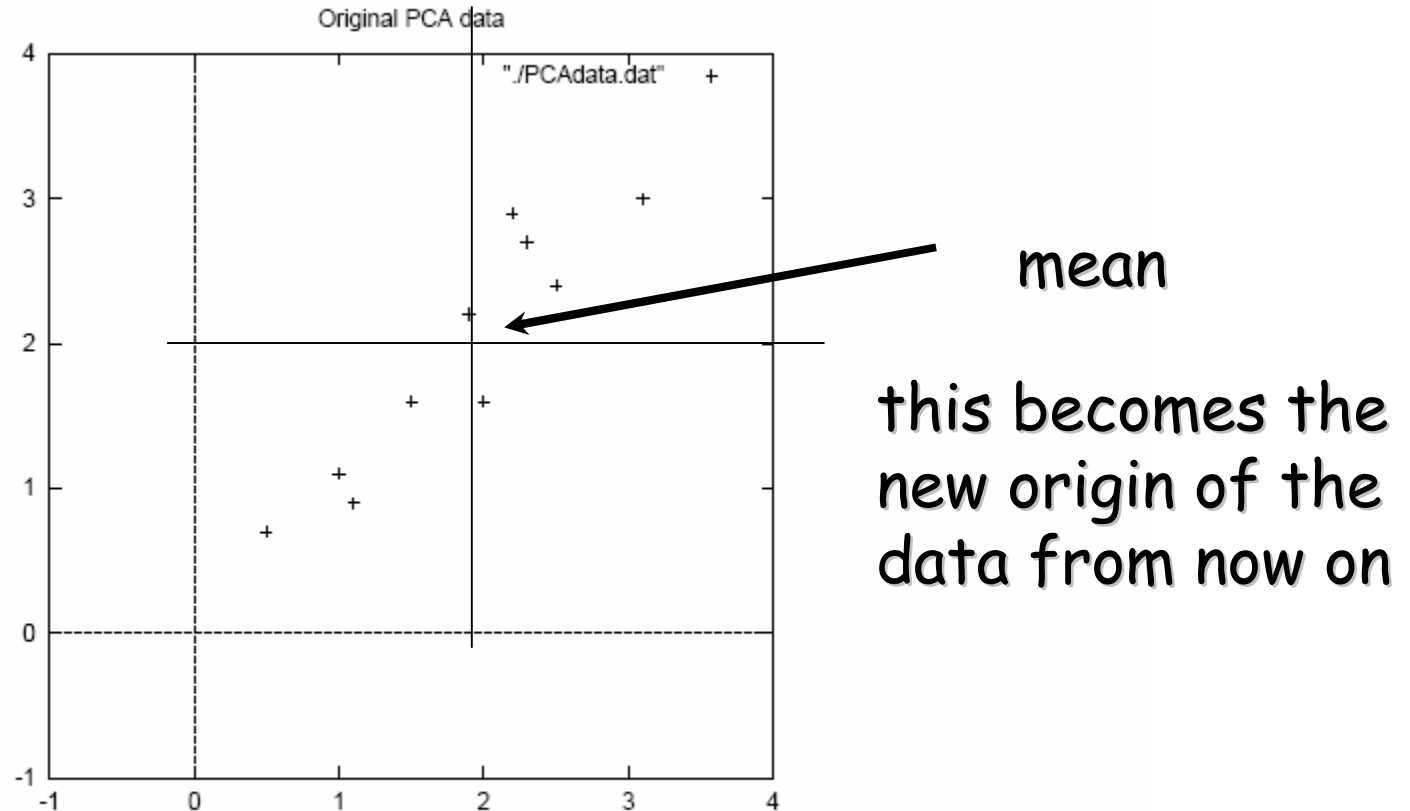


Figure 3.1: PCA example data, original data on the left, data with the means subtracted on the right, and a plot of the data

PCA Example -STEP 2

- Calculate the covariance matrix

$$\text{cov} = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

- since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y variable increase together.

PCA Example -STEP 3

- Calculate the eigenvectors and eigenvalues of the covariance matrix

$$\text{eigenvalues} = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$\text{eigenvectors} = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

PCA Example -STEP 3

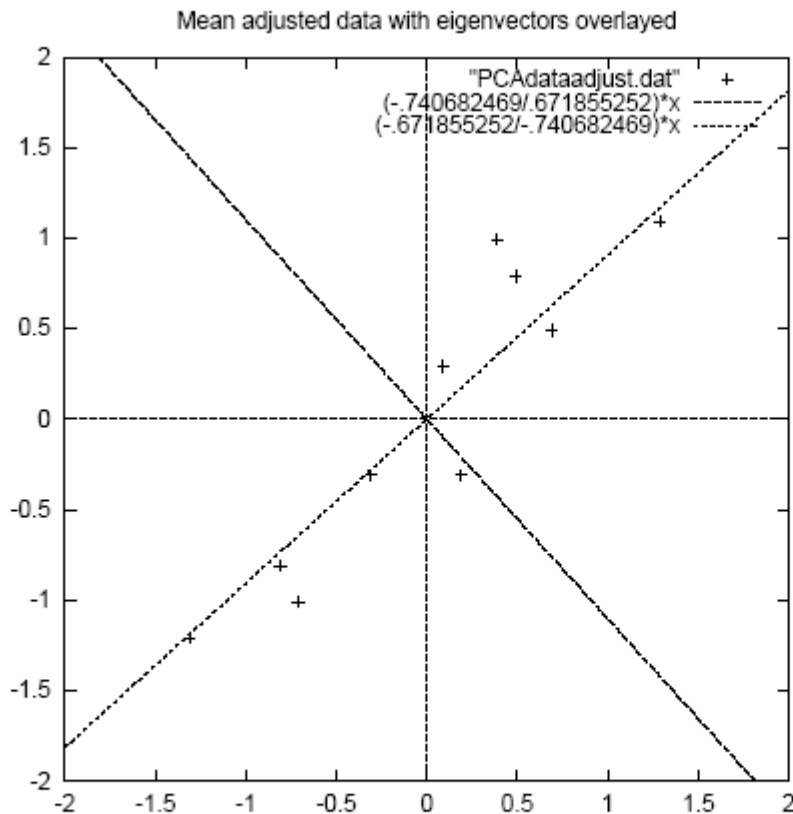


Figure 3.2: A plot of the normalised data (mean subtracted) with the eigenvectors of the covariance matrix overlaid on top.

- eigenvectors are plotted as diagonal dotted lines on the plot.
- Note they are perpendicular to each other.
- Note one of the eigenvectors goes through the middle of the points, like drawing a line of best fit.
- The second eigenvector gives us the other, less important, pattern in the data, that all the points follow the main line, but are off to the side of the main line by some amount.

PCA Example -STEP 4

- Feature Vector

$$\text{FeatureVector} = (\text{eig}_1 \text{ eig}_2 \text{ eig}_3 \dots \text{eig}_n)$$

We can either form a feature vector with both of the eigenvectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

or, we can choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

PCA Example -STEP 5

- Deriving new data coordinates

$\text{FinalData} = \text{RowFeatureVector} \times \text{RowZeroMeanData}$

RowFeatureVector is the matrix with the eigenvectors in the columns *transposed* so that the eigenvectors are now in the rows, with the most significant eigenvector at the top

RowZeroMeanData is the mean-adjusted data *transposed*, ie. the data items are in each column, with each row holding a separate dimension.

Note: this is essential Rotating the coordinate axes so higher-variance axes come first.

PCA Example -STEP 5

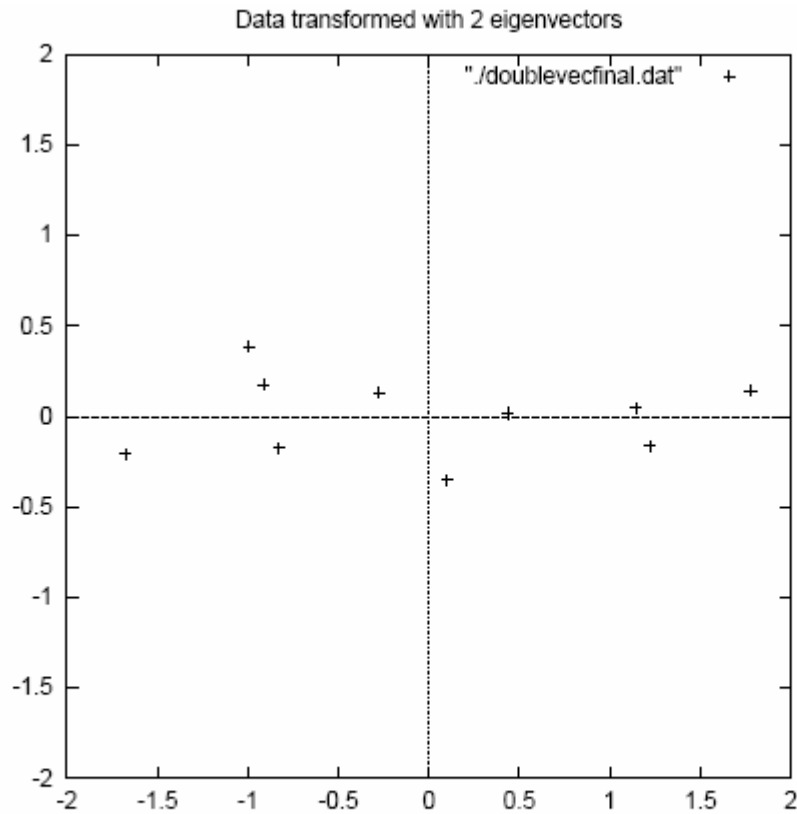
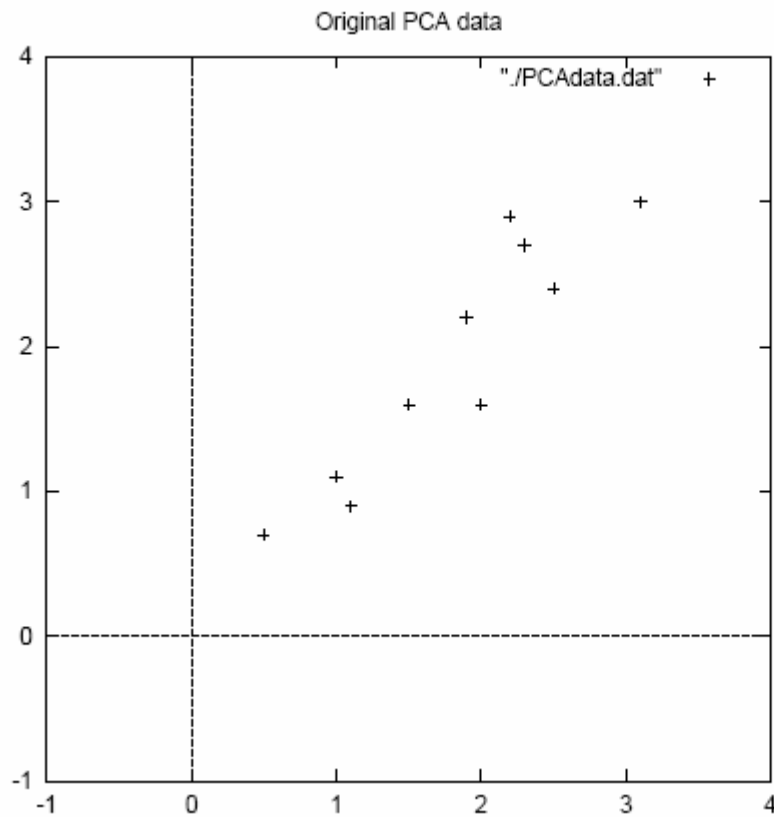


Figure 3.3: The table of data by applying the PCA analysis using both eigenvectors, and a plot of the new data points.

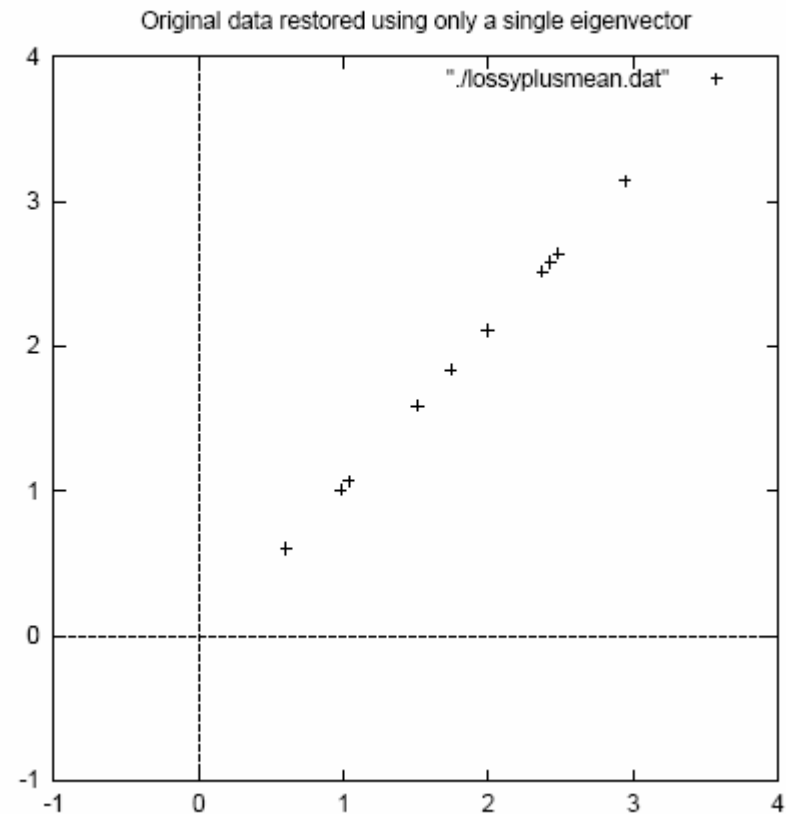
PCA Example : Approximation

- If we reduced the dimensionality, obviously, when reconstructing the data we would lose those dimensions we chose to discard. In our example let us assume that we considered only the x dimension...

PCA Example : Final Approximation



2D point cloud

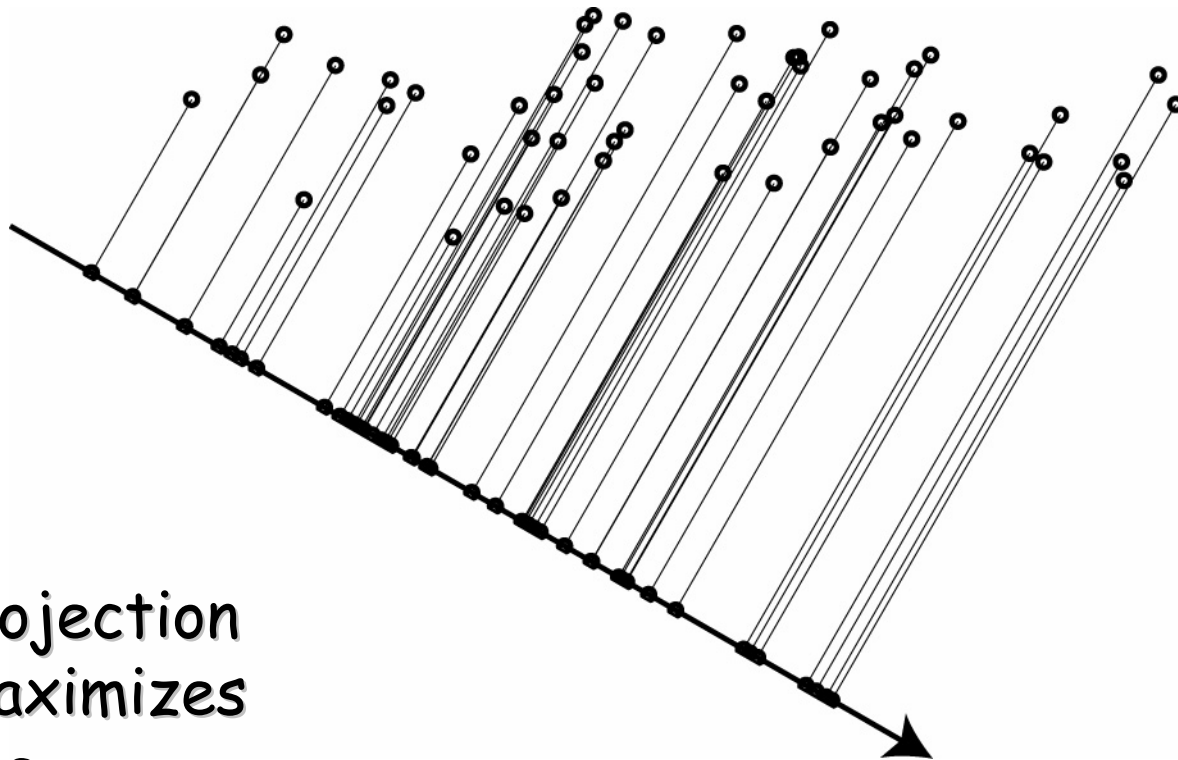


Approximation using
one eigenvector basis

Another way of thinking about Principal component

- direction of maximum variance in the input space
- happens to be same as the principal eigenvector of the covariance matrix

One-dimensional projection



find projection
that maximizes
variance

Covariance to variance

- From the covariance, the variance of any projection can be calculated.
- Let w be a unit vector

$$\begin{aligned}\langle (w^T x)^2 \rangle - \langle w^T x \rangle^2 &= w^T C w \\ &= \sum_{ij} w_i C_{ij} w_j\end{aligned}$$

Maximizing variance

- Principal eigenvector of C
 - the one with the largest eigenvalue.

$$w^* = \arg \max_{w: |w|=1} w^T C w$$

$$\begin{aligned} \lambda_{\max}(C) &= \max_{w: |w|=1} w^T C w \\ &= w^{*T} C w^* \end{aligned}$$

Implementing PCA

- Need to find "first" k eigenvectors of Q:

$$Q = XX^T = \begin{bmatrix} \mathbf{x}_1 - \bar{\mathbf{x}} & \mathbf{x}_2 - \bar{\mathbf{x}} & \cdots & \mathbf{x}_n - \bar{\mathbf{x}} \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \bar{\mathbf{x}})^T \\ (\mathbf{x}_2 - \bar{\mathbf{x}})^T \\ \vdots \\ (\mathbf{x}_n - \bar{\mathbf{x}})^T \end{bmatrix}$$

Q is N x N (Again, N could be the number of pixels in an image. For a 256 x 256 image, N = 65536 !!)
Don't want to explicitly compute Q!!!!

Singular Value Decomposition (SVD)

Any $m \times n$ matrix X can be written as the product of 3 matrices:

$$X = UDV^T$$

Where:

- U is $m \times m$ and its columns are orthonormal vectors
- V is $n \times n$ and its columns are orthonormal vectors
- D is $m \times n$ diagonal and its diagonal elements are called the singular values of X , and are such that:

$$\sigma_1, \sigma_2, \dots, \sigma_n, 0$$

SVD Properties

$$X = UDV^T$$

- The columns of U are the eigenvectors of XX^T
- The columns of V are the eigenvectors of X^TX
- The squares of the diagonal elements of D are the eigenvalues of XX^T and X^TX

Algebra of Principal Component Analysis

Data: $\mathbf{Y} = \begin{bmatrix} 2 & 1 \\ 3 & 4 \\ 5 & 0 \\ 7 & 6 \\ 9 & 2 \end{bmatrix}$ Centre each column on its mean: $\mathbf{Y}_c = [y - \bar{y}] = \begin{bmatrix} -3.2 & -1.6 \\ -2.2 & 1.4 \\ -0.2 & -2.6 \\ 1.8 & 3.4 \\ 3.8 & -0.6 \end{bmatrix}$

Covariance matrix (2 variables): $\mathbf{S} = \frac{1}{n-1} \mathbf{Y}_c' \mathbf{Y}_c = \begin{bmatrix} 8.2 & 1.6 \\ 1.6 & 5.8 \end{bmatrix}$

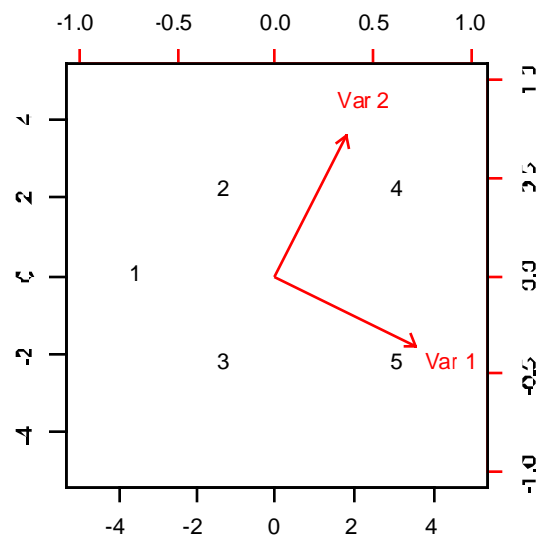
Equation for eigenvalues and eigenvectors of \mathbf{S} : $(\mathbf{S} - \lambda_k \mathbf{I}) \mathbf{u}_k = \mathbf{0}$

Eigenvalues: $\lambda_1 = 9, \lambda_2 = 5$ Matrix of eigenvalues: $\Lambda = \begin{bmatrix} 9 & 0 \\ 0 & 5 \end{bmatrix}$

Matrix of eigenvectors: $\mathbf{U} = \begin{bmatrix} 0.8944 & -0.4472 \\ 0.4472 & 0.8944 \end{bmatrix}$

Positions of the 5 objects in ordination space: $\mathbf{F} = [y - \bar{y}] \mathbf{U}$

$$\mathbf{F} = \begin{bmatrix} -3.2 & -1.6 \\ -2.2 & 1.4 \\ -0.2 & -2.6 \\ 1.8 & 3.4 \\ 3.8 & -0.6 \end{bmatrix} \begin{bmatrix} 0.8944 & -0.4472 \\ 0.4472 & 0.8944 \end{bmatrix} = \begin{bmatrix} -3.578 & 0 \\ -1.342 & 2.236 \\ -1.342 & -2.236 \\ 3.130 & 2.236 \\ 3.130 & -2.236 \end{bmatrix}$$



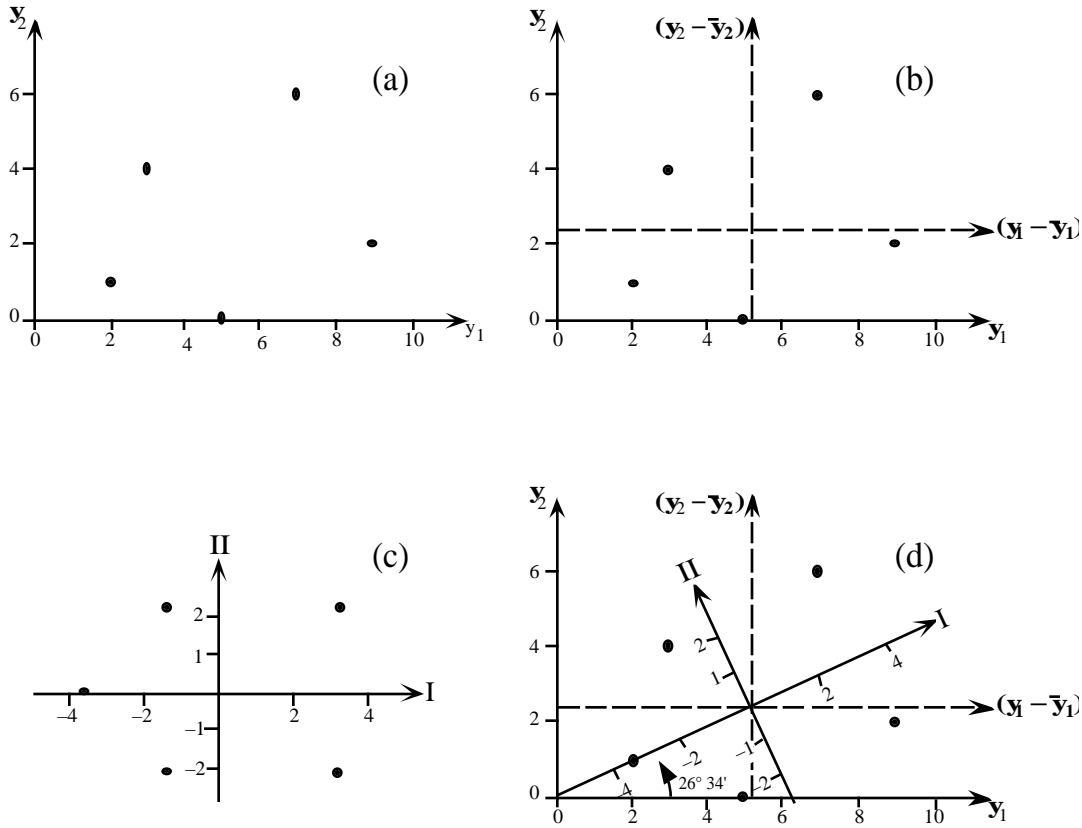


Figure 9.2 Numerical example of principal component analysis. (a) Five objects are plotted with respect to descriptors y_1 and y_2 . (b) After centring the data, the objects are now plotted with respect to $(y_1 - \bar{y}_1)$ and $(y_2 - \bar{y}_2)$, represented by dashed axes. (c) The objects are plotted with reference to principal axes I and II, which are centred with respect to the scatter of points. (d) The two systems of axes (b and c) can be superimposed after a rotation of $26^\circ 34'$.

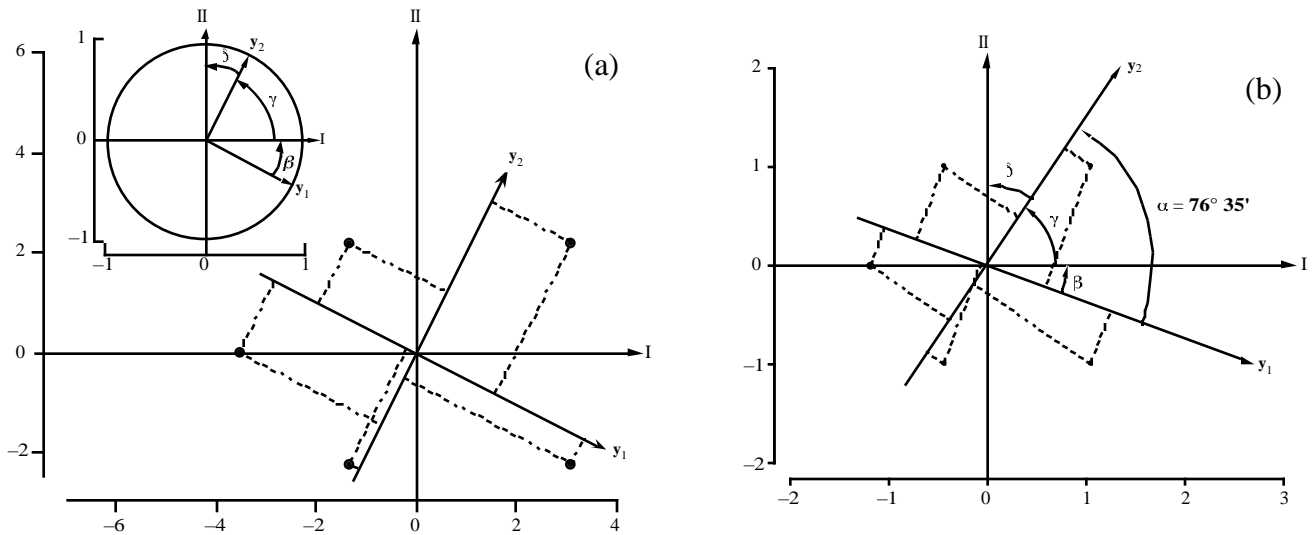


Fig. 9.3 Numerical example from Fig. 9.2. Distance and correlation biplots are discussed in Subsection 9.1.4. **(a) Distance biplot.** The eigenvectors are scaled to lengths 1. Inset: descriptors (matrix \mathbf{U}). Main graph: descriptors (matrix \mathbf{U} ; arrows) and objects (matrix \mathbf{F} ; dots). The interpretation of the object-descriptor relationships is not based on their proximity, but on orthogonal projections (dashed lines) of the objects on the descriptor-axes or their extensions. **(b) Correlation biplot.** Descriptors (matrix $\mathbf{U}\Lambda^{1/2}$; arrows) with a covariance angle of $76^\circ 35'$. Objects (matrix \mathbf{G} ; dots). Projecting the objects orthogonally on a descriptor (dashed lines) reconstructs the values of the objects along that descriptors, to within a multiplicative constant.

Use the following matrices to draw biplots

Distance biplot (scaling 1): objects = \mathbf{F} , variables = \mathbf{U}

Correlation biplot (scaling 2): objects = $\mathbf{G} = \mathbf{F}\Lambda^{-1/2}$, variables = $\mathbf{U}_{sc2} = \mathbf{U}\Lambda^{1/2}$

These two projections respect the biplot rule, that the product of the two projected matrices reconstruct the data \mathbf{Y} :

$$\text{Distance biplot: } \mathbf{F}\mathbf{U}' = \mathbf{Y}$$

$$\text{Correlation biplot: } \mathbf{G}(\mathbf{U}\Lambda^{1/2})' = \mathbf{Y}$$

Data transformation

Transform physical variables (*Ecology*) or characters (*Taxonomy*)

- Univariate distributions are not symmetrical
⇒ Apply skewness-reduction transformation
- Variables are not in the same physical units

⇒ Apply standardization $z_i = \frac{y_i - \bar{y}}{s_y}$ or ranging $y'_i = \frac{y_i - y_{min}}{y_{max} - y_{min}}$

- Multistate qualitative variables
⇒ In some cases, transform them to dummy (binary) variables

Transform community composition data (*Ecology*)

(species presence-absence or abundance)

- Reduce asymmetry of distributions
⇒ Apply $\log(y + c)$ transformation
- Make community composition data suitable for Euclidean-based ordination methods (PCA, RDA)
⇒ Use the chord, chi-square, or Hellinger transformations (Legendre & Gallagher 2001)

Some uses of principal component analysis (PCA)

- Two-dimensional ordination of the *objects*:
 - Sampling sites in ecology
 - Individuals or taxa in taxonomy

⇒ A 2-dimensional ordination diagram is an interesting graphical support for representing other properties of multivariate data, e.g., clusters.
- Detect outliers or erroneous data in data tables
- Find groups of *variables* that behave in the same way:
 - Species in ecology
 - Morphological/behavioural/molecular variables in taxonomy
- Simplify (collinear) data; remove noise
- Remove an identifiable component of variation
e.g., size factor in log-transformed morphological data

Algebra of Correspondence Analysis

$$\text{Frequency data table } \mathbf{Y} = \begin{bmatrix} \mathbf{f}_{ij} \end{bmatrix} = \begin{bmatrix} 10 & 10 & 20 \\ 10 & 15 & 10 \\ 15 & 5 & 5 \end{bmatrix} \begin{bmatrix} \mathbf{f}_{i+} \\ 40 \\ 35 \\ 25 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{f}_{+j} \end{bmatrix} = \begin{bmatrix} 35 & 30 & 35 \end{bmatrix} \quad 100 = \mathbf{f}_{++}$$

$$\begin{aligned} p_{ij} &= \mathbf{f}_{ij} / \mathbf{f}_{++} \\ p_{i+} &= \mathbf{f}_{i+} / \mathbf{f}_{++} \\ p_{+j} &= \mathbf{f}_{+j} / \mathbf{f}_{++} \end{aligned}$$

$$\text{Matrix } \mathbf{Q} = [\bar{q}_{ij}] = \left[\frac{p_{ij} - p_{i+}p_{+j}}{\sqrt{p_{i+}p_{+j}}} \right] = \frac{(O_{ij} - E_{ij}) / \sqrt{E_{ij}}}{\sqrt{\mathbf{f}_{++}}}$$

$$\text{Matrix } \mathbf{Q} = \begin{bmatrix} -0.10690 & -0.05774 & 0.16036 \\ -0.06429 & 0.13887 & -0.06429 \\ 0.21129 & -0.09129 & -0.12667 \end{bmatrix}$$

$$\text{Cross-product matrix: } \mathbf{Q}'\mathbf{Q} = \begin{bmatrix} 0.06020 & -0.02204 & -0.03980 \\ -0.02204 & 0.03095 & -0.00661 \\ -0.03980 & -0.00661 & 0.04592 \end{bmatrix}$$

$$\text{Compute eigenvalues and eigenvectors of } \mathbf{Q}'\mathbf{Q} : \quad (\mathbf{Q}'\mathbf{Q} - \lambda_k \mathbf{I}) \mathbf{u}_k = \mathbf{0}$$

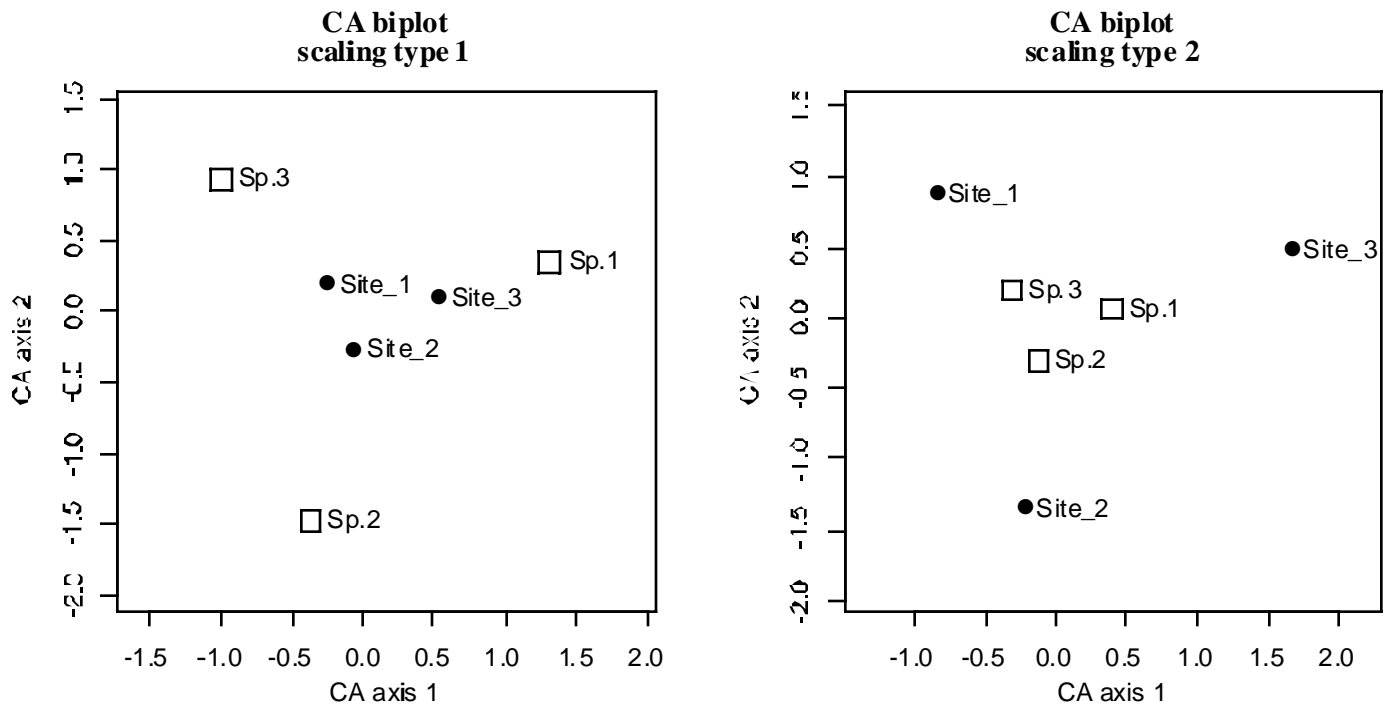
$$\text{Eigenvalues: } \lambda_1 = 0.096, \lambda_2 = 0.041 \quad \text{Matrix of eigenvalues: } \Lambda = \begin{bmatrix} 0.096 & 0 \\ 0 & 0.041 \end{bmatrix}$$

There are never more than $k = \min(r - 1, c - 1)$ eigenvalues > 0 in CA

$$\text{Matrix of eigenvectors of } \mathbf{Q}'\mathbf{Q}_{(c \times c)} : \quad \mathbf{U}_{(c \times k)} = \begin{bmatrix} 0.78016 & 0.20336 \\ -0.20383 & -0.81145 \\ -0.59144 & 0.54790 \end{bmatrix}$$

$$\text{Matrix of eigenvectors of } \mathbf{Q}\mathbf{Q}'_{(r \times r)} : \quad \hat{\mathbf{U}}_{(r \times k)} = \mathbf{Q}\mathbf{U}\Lambda^{-1/2} = \begin{bmatrix} -0.53693 & 0.55831 \\ -0.13043 & -0.79561 \\ 0.83349 & 0.23516 \end{bmatrix}$$

Compute matrices \mathbf{F} and \mathbf{V} for scaling 1 biplot, and $\hat{\mathbf{V}}$ and $\hat{\mathbf{F}}$ for scaling 2 biplot:



Calculation details

Compute matrices \mathbf{V} , $\hat{\mathbf{V}}$, \mathbf{F} , and $\hat{\mathbf{F}}$ used in the ordination biplots:

$$\mathbf{V}_{(c \times k)} = \mathbf{D}(p_{+j})^{-1/2} \mathbf{U} \quad \text{where } p_{+j} = f_{+j}/f_{++}$$

$$\hat{\mathbf{V}}_{(r \times k)} = \mathbf{D}(p_{i+})^{-1/2} \hat{\mathbf{U}} \quad \text{where } p_{i+} = f_{i+}/f_{++}$$

$$\mathbf{F}_{(r \times k)} = \hat{\mathbf{V}} \mathbf{A}^{1/2}$$

$$\hat{\mathbf{F}}_{(c \times k)} = \mathbf{V} \mathbf{A}^{1/2}$$

Biplot, scaling type 1: plot \mathbf{F} for sites, \mathbf{V} for species:

- This projection preserves the chi-square distance among the sites.
- The sites are at the centroids (barycentres) of the species.

Biplot, scaling type 2: plot $\hat{\mathbf{V}}$ for sites, $\hat{\mathbf{F}}$ for species:

- This projection preserves the chi-square distance among the species.
- The species are at the centroids (barycentres) of the sites.

Discriminate Analysis



Outline

- *Introduction*
- Linear Discriminant Analysis
- Examples

Introduction

- What is Discriminant Analysis?
 - Statistical technique to classify objects into mutually exclusive and exhaustive groups based on a set of measurable object's features

Introduction

- Purpose of Discriminant Analysis
 - To classify objects (people, customers, things, etc.) into one of two or more groups based on a set of features that describe the objects (e.g. gender, age, income, weight, preference score, etc.).
- Two things to check
 - Which *set of features* can best determine group membership of the object?  **Feature Selection**
 - What is the classification *rule* or *model* to best separate those groups?  **Classification**

Outline

- Introduction
- *Linear Discriminant Analysis*
- Examples

Linear Discriminant Analysis (LDA)

- Linear discriminant analysis (LDA),
 - Also called Fisher's linear discriminant
 - Methods used in statistics and machine learning to find the linear combination of features which
 - best separate two or more classes of object or event.
 - The resulting combination may be used as a linear classifier, or, more commonly, for **dimensionality reduction before later classification**.

Dimensionality Reduction

- Curse of dimensionality:
 - Problem caused by higher the dimension of the feature vectors
 - Data sparsity
 - Undertrained classifier



- Goal:
 - Reduce dimension of feature vectors without loss of information


Linear Discriminant Analysis (LDA)

- Goal:
 - Try to optimize class separability
 - Also known as Fisher's discriminant analysis

Linear Discriminant Analysis (LDA)

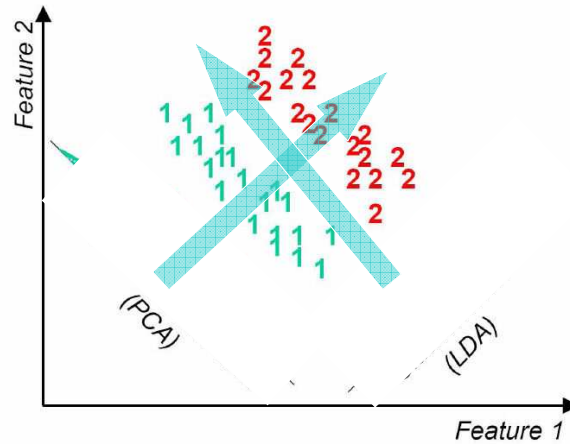
- Problem statement
 - Assign class category (or the group, class label): “good” and “bad” for each product
 - Class category is also called **dependent variable**.
 - Based on Features
 - Each measurement on the product is called features that describe the object
 - it is also called **independent variable**.
- **Dependent variable (Y)** is the group
 - The dependent variable is always category (nominal scale) variable
- **Independent variables (X)** are the object features that might describe the group
 - independent variables can be any measurement scale (i.e. nominal, ordinal, interval or ratio)

Linear Discriminant Analysis (LDA)

- Linear Discriminant Analysis (LDA)
 - Assume that the groups are linearly separable 
 - Use linear discriminant model (LDA)
- What is Linearly separable?
 - It suggests that the groups can be separated by a linear combination of features that describe the objects

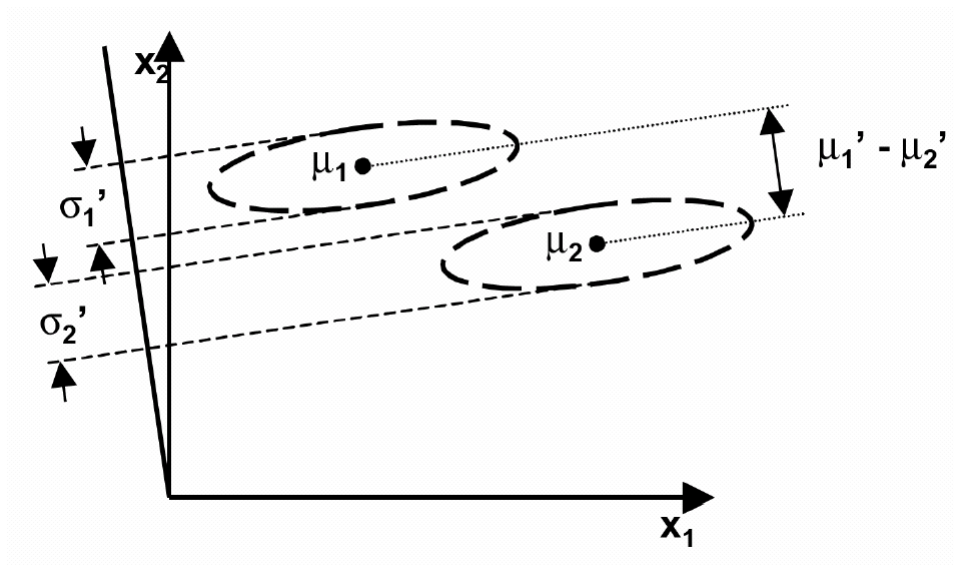
Linear Discriminant Analysis (LDA)

- PCA vs LDA
 - PCA is trying to find the strongest correlation in the dataset
 - LDA is trying to optimize class separability



Linear Discriminant Analysis (LDA)

- Goal of LDA: try to maximize class separability



Different Approaches to LDA

- Class-dependent transformation
 - Maximizing the ratio of between class variance to within class variance.
 - Involving using two optimizing criteria for transforming the data sets independently
- Class-independent transformation
 - Maximizing the ratio of overall variance to within class variance
 - Only using one optimizing criterion to transform the data sets and hence all data points irrespective of their class identity are transformed using this transform.

Numerical Example

- Given a two-class problem.
- Input: two sets of 2-D data points

$$\begin{array}{ccc} \text{set1} = & \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \dots & \dots \\ \dots & \dots \\ a_{m1} & a_{m2} \end{bmatrix} & \text{set2} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ \dots & \dots \\ \dots & \dots \\ b_{m1} & b_{m2} \end{bmatrix} \\ \uparrow & & \uparrow \\ \text{Class 1} & & \text{Class 2} \end{array}$$

Numerical Example

- Step 1
 - Compute the mean of each data set and mean of entire data set.

Data Points in Class 1 \longrightarrow **Mean of Set 1** μ_1 **n*1 column vector**

Data Points in Class 2 \longrightarrow **Mean of Set 2** μ_2 **n*1 column vector**

Data Points in both Class 1 and Class 2 \longrightarrow **Mean of Entire Data** μ_3 **n*1 column vector**

$$\mu_3 = p_1 \times \mu_1 + p_2 \times \mu_2$$

Where n is the number of dimension. In our case, it is equal to 2

Numerical Example

- Step 2
 - Compute the Between Class Scatter Matrix S_b and Within Class Scatter Matrix S_w

Within Class Scatter Matrix $S_w = \sum_j p_j \times (cov_j)$

where p_j **is the prior probabilities of the jth class**

and $cov_j = (x_j - \mu_j)(x_j - \mu_j)^T$ **is covariance matrix of the jth class (set j)**

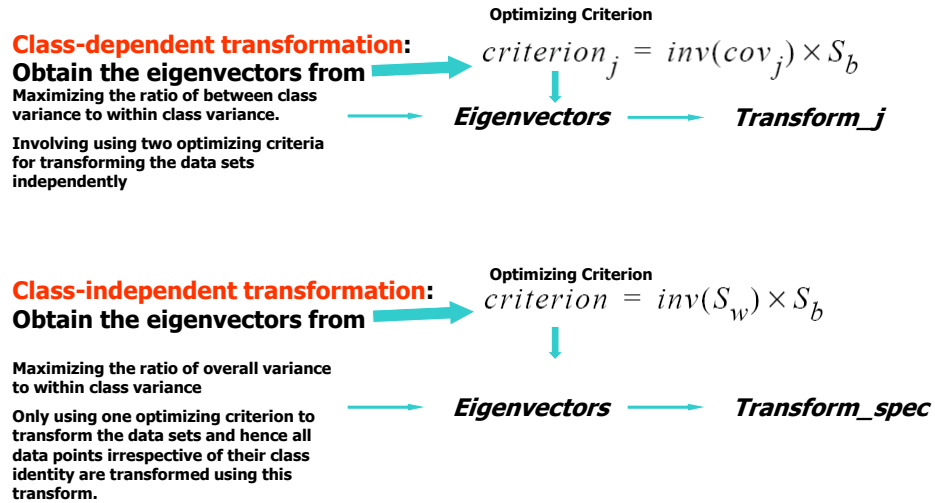
Between Class Scatter Matrix $S_b = \sum_j (\mu_j - \mu_3) \times (\mu_j - \mu_3)^T$

where μ_3 **is the mean of the entire data**

and μ_j **is the mean of the jth class (set j)**

Numerical Example

- Step 3
 - Eigenvectors computation



Numerical Example

- Step 4
 - Transformed matrix calculation

For the class dependent LDA,

$$transformed_set_j = transform_j^T \times set_j$$

Where “transform_j” is composed of eigenvectors from

$$criteria_j = inv(cov_j) \times S_b$$

For the class independent LDA,

$$transformed_set = transform_spec^T \times data_set^T$$

Where “transform_spec” is composed of eigenvectors from

$$criteria = inv(S_w) \times S_b$$

Numerical Example

- Step 5
 - Euclidean distance calculate

$$dist_n = (transform_n_spec)^T \times x - \mu_{ntrans}$$

where μ_{ntrans} is the mean of the transformed data set

n is the class index

x is the test vector

➡ For n classes, n Euclidean distances are obtained for each test point

Numerical Example

- Step 6
 - Classification result is based on the smallest Euclidean distance among the n distances classifiers the test vector a belonging to class n

Extension to Multiple Classes

- Between Class Scatter Matrix

$$S_b = \sum_{k=1}^K p_k (\vec{\mu}_k - \vec{\mu})(\vec{\mu}_k - \vec{\mu})^t$$

with : K : number of classes

$$p_k = \frac{N_k}{\sum_{l=1}^K N_l} \quad (\text{fraction of data belonging to class } k)$$

$$\vec{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \vec{x}_{i,k} \quad (\text{mean vector of class } k)$$

$\vec{\mu}$: mean of all vectors

Extension to Multiple Classes

- Within Class Scatter Matrix

$$S_w = \sum_{k=1}^K p_k \Sigma_k$$

with

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{N_k} (\vec{x}_{i,k} - \vec{\mu}_k)(\vec{x}_{i,k} - \vec{\mu}_k)^t \quad (\text{covariance matrix of class } k)$$

Extension to Multiple Classes

- Maximize class separability
 - Keep variance of all classes roughly constant
- ↳ optimization problem with constraint

Solution

$$S_b \vec{\phi}_i = \lambda_i S_w \vec{\phi}_i$$



$$S_w^{-1} S_b \vec{\phi}_i = \lambda_i \vec{\phi}_i$$

- Questions?

Introduction

Linear Discriminant Analysis (LDA) is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting (“curse of dimensionality”) and also reduce computational costs.

Ronald A. Fisher formulated the Linear Discriminant in 1936 (The Use of Multiple Measurements in Taxonomic Problems), and it also has some practical uses as classifier. The original Linear discriminant was described for a 2-class problem, and it was then later generalized as “multi-class Linear Discriminant Analysis” or “Multiple Discriminant Analysis” by C. R. Rao in 1948 (The utilization of multiple measurements in problems of biological classification)

The general LDA approach is very similar to a Principal Component Analysis (for more information about the PCA, see the previous article Implementing a Principal Component Analysis (PCA) in Python step by step), but in addition to finding the component axes that maximize the variance of our data (PCA), we are additionally interested in the axes that maximize the separation between multiple classes (LDA).

So, in a nutshell, often the goal of an LDA is to project a feature space (a dataset n-dimensional samples) onto a smaller subspace k

(where $k \leq n-1$) while maintaining the class-discriminatory information.

In general, dimensionality reduction does not only help reducing computational costs for a given classification task, but it can also be helpful to avoid overfitting by minimizing the error in parameter estimation (“curse of dimensionality”).

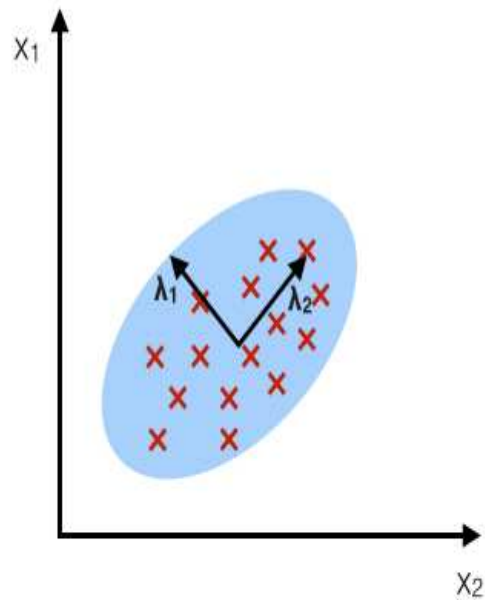
Principal Component Analysis vs. Linear Discriminant Analysis

Both Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) are linear transformation techniques that are commonly used for dimensionality reduction. PCA can be described as an “unsupervised” algorithm, since it “ignores” class labels and its goal is to find the directions (the so-called principal components) that maximize the variance in a dataset. In contrast to PCA, LDA is “supervised” and computes the directions (“linear discriminants”) that will represent the axes that maximize the separation between multiple classes.

Although it might sound intuitive that LDA is superior to PCA for a multi-class classification task where the class labels are known, this might not always be the case. For example, comparisons between classification accuracies for image recognition after using PCA or LDA show that PCA tends to outperform LDA if the number of samples per class is relatively small (PCA vs. LDA, A.M. Martinez et al., 2001). In practice, it is also not uncommon to use both LDA and PCA in combination: E.g., PCA for dimensionality reduction followed by an LDA.

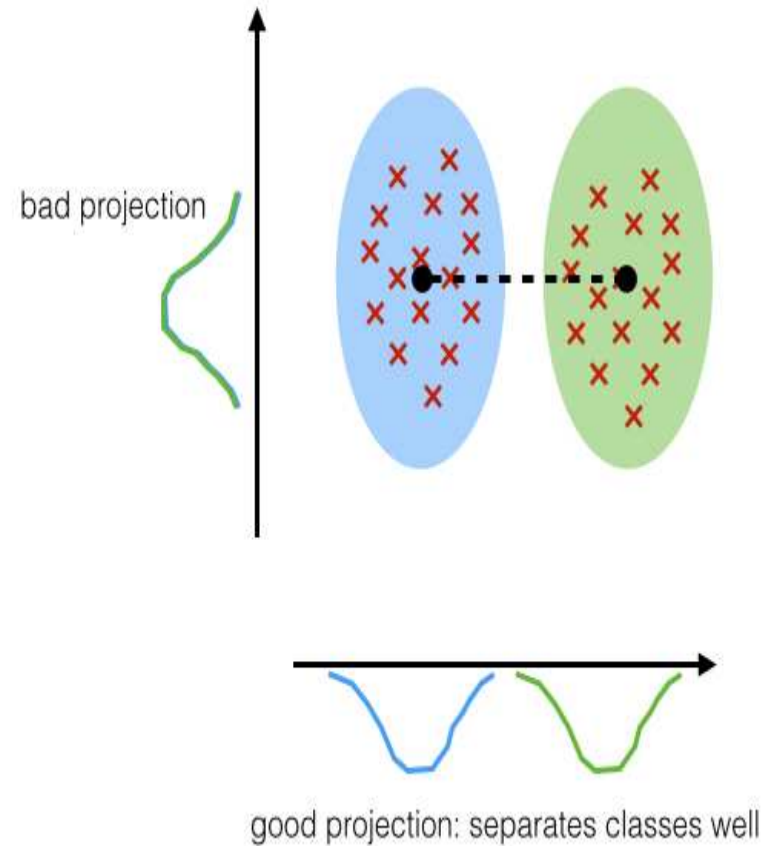
PCA:

component axes that maximize the variance



LDA:

maximizing the component axes for class-separation



What is a “good” feature subspace?

Let's assume that our goal is to reduce the dimensions of a d

-dimensional dataset by projecting it onto a (k) -dimensional subspace (where $k < d$). So, how do we know what size we should choose for k (k

= the number of dimensions of the new feature subspace), and how do we know if we have a feature space that represents our data “well”?

Later, we will compute eigenvectors (the components) from our data set and collect them in a so-called scatter-matrices (i.e., the in-between-class scatter matrix and within-class scatter matrix). Each of these eigenvectors is associated with an eigenvalue, which tells us about the “length” or “magnitude” of the eigenvectors.

If we would observe that all eigenvalues have a similar magnitude, then this may be a good indicator that our data is already projected on a “good” feature space.

And in the other scenario, if some of the eigenvalues are much much larger than others, we might be interested in keeping only those eigenvectors with the highest eigenvalues, since they contain more information about our data distribution. Vice versa, eigenvalues that are close to 0 are less informative and we might consider dropping those for constructing the new feature subspace.

Summarizing the LDA approach in 5 steps

Listed below are the 5 general steps for performing a linear discriminant analysis; we will explore them in more detail in the following sections.

1. Compute the d -dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors (e_1, e_2, \dots, e_d) and corresponding eigenvalues ($\lambda_1, \lambda_2, \dots, \lambda_d$) for the scatter matrices.
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $d \times k$ dimensional matrix W (where every column represents an eigenvector).
5. Use this $d \times k$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the matrix multiplication: $Y = X \times W$ (where X is a $n \times d$ -dimensional matrix representing the n samples, and yy are the transformed $n \times k$ -dimensional samples in the new subspace).

Principal Component Analysis and Linear Discriminant Analysis for Feature Reduction

Outline of lecture

- What is feature reduction?
- Why feature reduction?
- Feature reduction algorithms
 - Principal Component Analysis (PCA)
 - Linear Discriminant Analysis (LDA)

What is feature reduction?

- Feature reduction refers to the mapping of the original high-dimensional data onto a lower-dimensional space.
 - Criterion for feature reduction can be different based on different problem settings.
 - Unsupervised setting: minimize the information loss
 - Supervised setting: maximize the class discrimination
- Given a set of data points of p variables x_1, x_2, \dots, x_n
Compute the linear transformation (projection)

$$G \quad d \times p : x \quad d \quad y \quad G^T x \quad p \quad (p \quad d)$$

What is fe

Original data

reduced data

Linear transformation



$$G^T \quad p \quad d$$



$$X \quad d$$

$$G \quad d \quad p : X$$

$$Y \quad G^T X \quad p$$



Feature reduction versus feature selection

- Feature reduction
 - All original features are used
 - The transformed features are linear combinations of the original features.
- Feature selection
 - Only a subset of the original features are used.
- Continuous versus discrete

Outline of lecture

- What is feature reduction?
- **Why feature reduction?**
- Feature reduction algorithms
 - Principal Component Analysis (PCA)
 - Linear Discriminant Analysis (LDA)

Why feature reduction?

- Most machine learning and data mining techniques may not be effective for high-dimensional data
 - **Curse of Dimensionality**
 - Query accuracy and efficiency degrade rapidly as the dimension increases.
- The **intrinsic** dimension may be small.
 - For example, the number of genes responsible for a certain type of disease may be small.

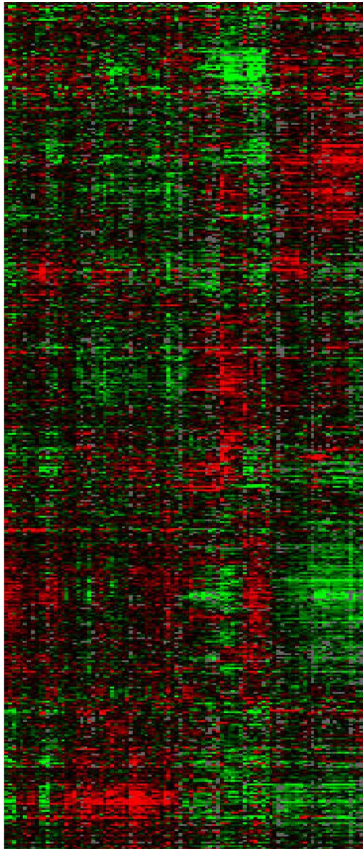
Why feature reduction?

- **Visualization**: projection of high-dimensional data onto 2D or 3D.
- **Data compression**: efficient storage and retrieval.
- **Noise removal**: positive effect on query accuracy.

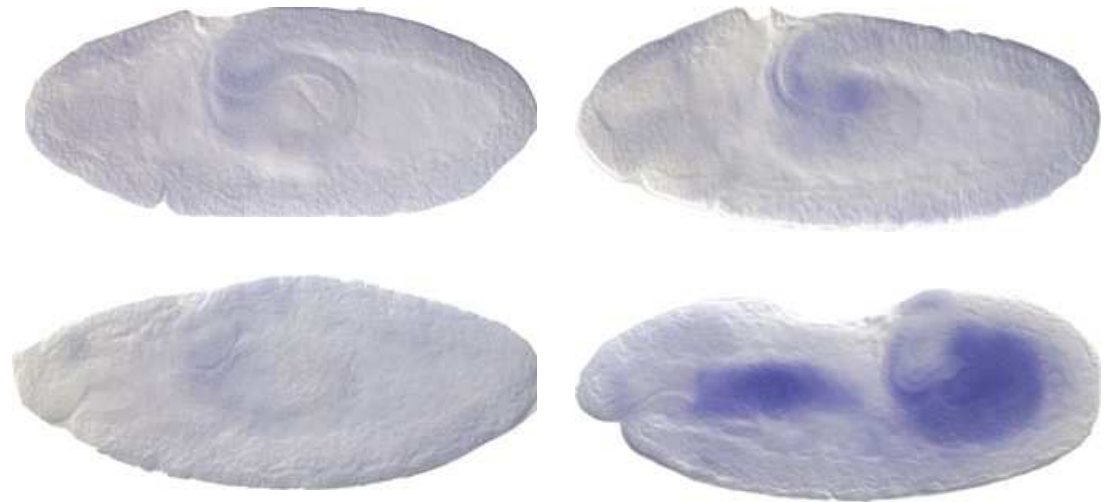
Applications of feature reduction

- Face recognition
- Handwritten digit recognition
- Text mining
- Image retrieval
- Microarray data analysis
- Protein classification

High-dimensional data in bioinformatics



Gene expression

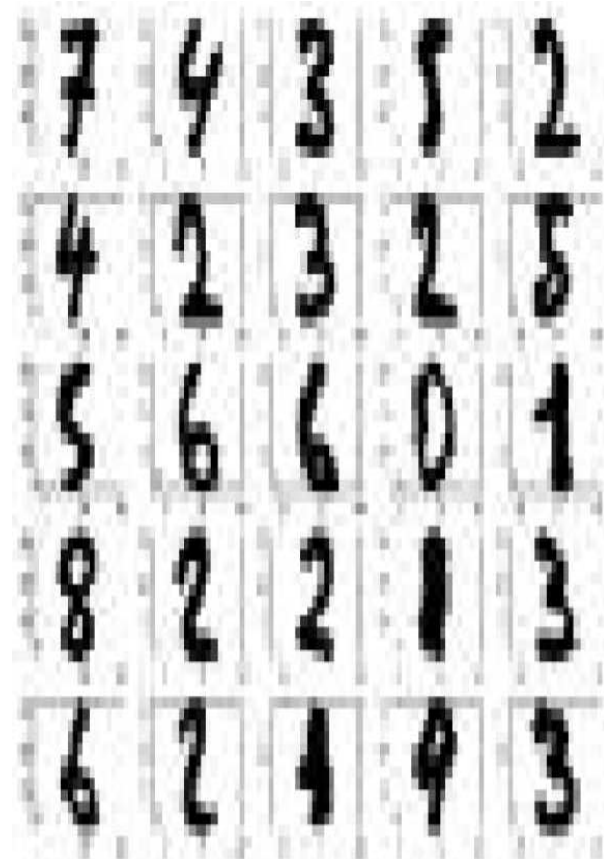


Gene expression pattern images

High-dimensional data in computer vision



Face images



Handwritten digits

Outline of lecture

- What is feature reduction?
- Why feature reduction?
- **Feature reduction algorithms**
 - **Principal Component Analysis (PCA)**
 - Linear Discriminant Analysis (LDA)

Feature reduction algorithms

- **Unsupervised**
 - Latent Semantic Indexing (LSI): truncated SVD
 - Independent Component Analysis (ICA)
 - Principal Component Analysis (PCA)
 - Canonical Correlation Analysis (CCA)
- **Supervised**
 - Linear Discriminant Analysis (LDA)
- **Semi-supervised**
 - Research topic

What is Principal Component Analysis?

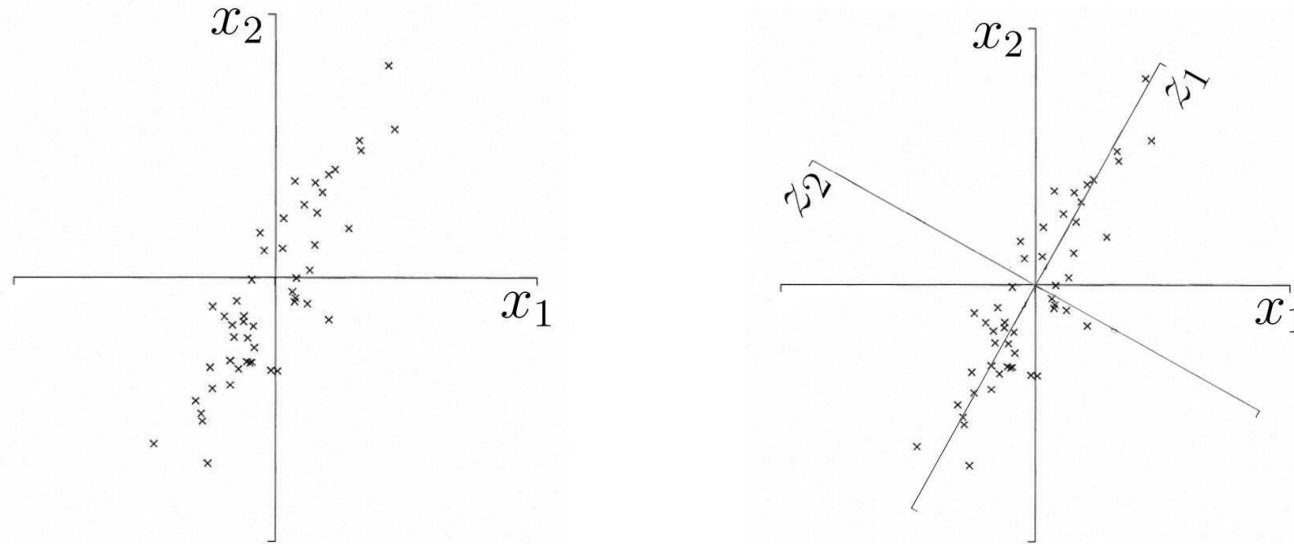
- Principal component analysis (PCA)

- Reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables
- Retains most of the sample's information.
- Useful for the compression and classification of data.

- By information we mean the variation present in the sample, given by the correlations between the original variables.

- The new variables, called principal components (PCs), are **uncorrelated**, and are ordered by the fraction of the total information each retains.

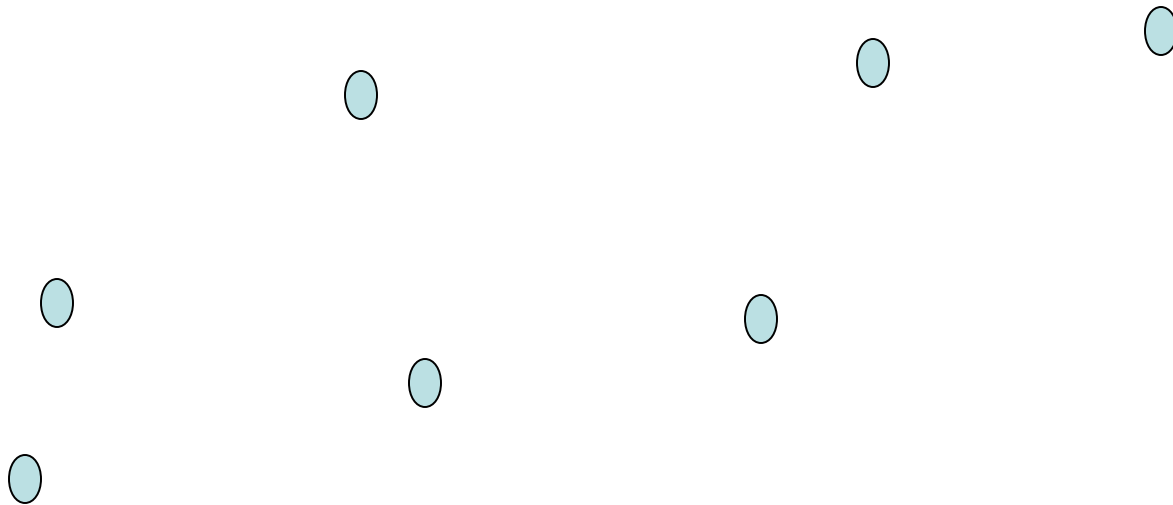
Geometric picture of principal components (PCs)



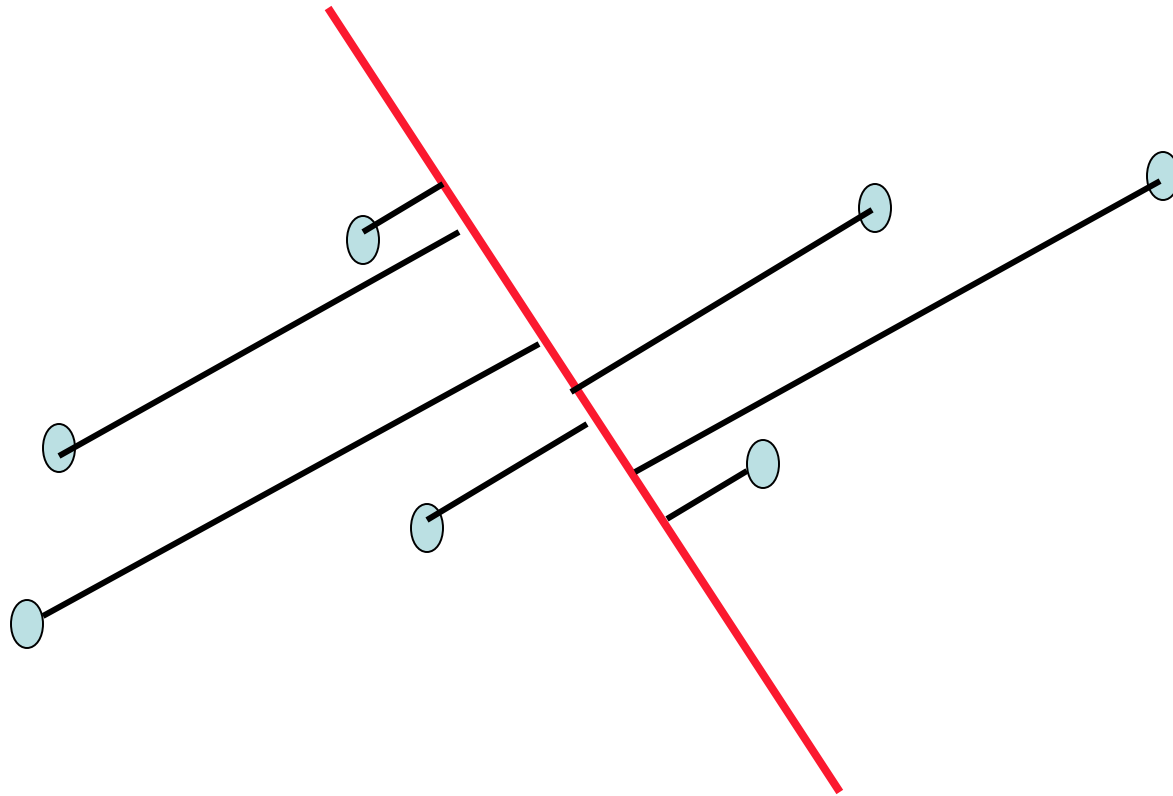
- the 1st PC Z_1 is a minimum distance fit to a line in X space
- the 2nd PC Z_2 is a minimum distance fit to a line in the plane perpendicular to the 1st PC

PCs are a series of linear least squares fits to a sample, each orthogonal to all the previous.

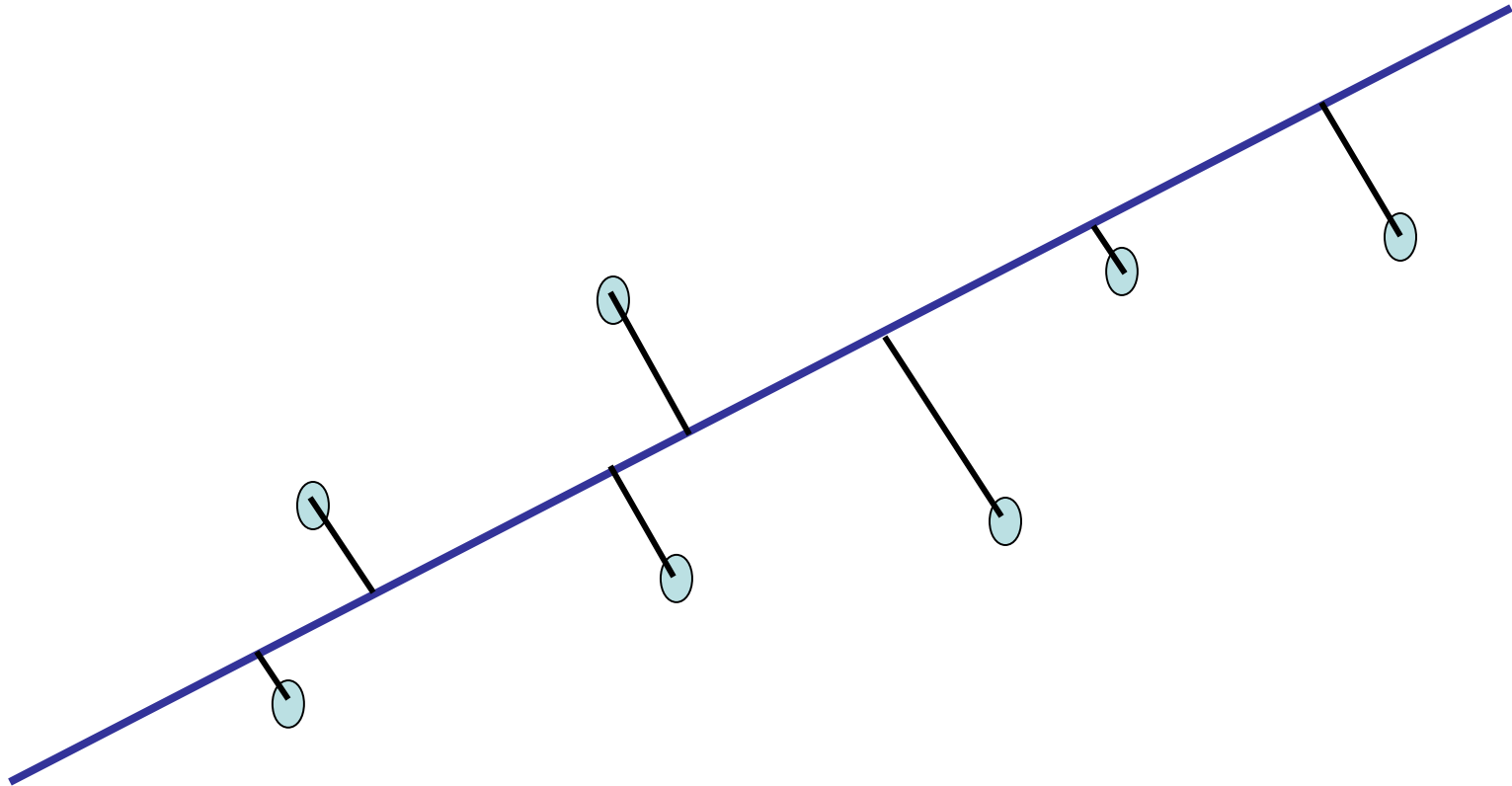
Geometric picture of principal components (PCs)



Geometric picture of principal components (PCs)



Geometric picture of principal components (PCs)



Algebraic definition of PCs

Given a sample of n observations on a vector of p variables

$$x_1, x_2, \dots, x_n \quad d$$

define the first principal component of the sample by the linear transformation

$$z_1 = a_1^T x_j = \sum_{i=1}^d a_{i1} x_{ij}, \quad j = 1, 2, \dots, n.$$

where the vector

$$a_1 = (a_{11}, a_{21}, \dots, a_{d1})$$

$$x_j = (x_{1j}, x_{2j}, \dots, x_{dj})$$

is chosen such that $\text{var}[z_1]$ is maximum.

Algebraic derivation of PCs

To find a_1 first note that

$$\text{var}[z_1] = E\left(\left(\frac{1}{n} \sum_{i=1}^n a_1^T x_i - a_1^T \bar{x}\right)^2\right)$$

$$= \frac{1}{n} \sum_{i=1}^n a_1^T x_i x_i^T a_1 - a_1^T S a_1$$

where $S = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T$

is the covariance matrix.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \text{ is the mean.}$$

Algebraic derivation of PCs

To find \mathbf{a}_1 that maximizes $\text{var}[z_1]$ subject to $\mathbf{a}_1^T \mathbf{a}_1 = 1$

Let λ be a Lagrange multiplier

$$L = \mathbf{a}_1^T \tilde{S} \mathbf{a}_1 - \lambda (\mathbf{a}_1^T \mathbf{a}_1 - 1)$$

$$\frac{\partial L}{\partial \mathbf{a}_1} = 2\tilde{S}\mathbf{a}_1 - 2\lambda\mathbf{a}_1 = 0$$

$$(\tilde{S} - \lambda I_p)\mathbf{a}_1 = 0$$

therefore \mathbf{a}_1 is an eigenvector of \tilde{S}

corresponding to the largest eigenvalue

Algebraic derivation of PCs

We find that a_2 is also an eigenvector of S whose eigenvalue λ_2 is the second largest.

In general

$$\text{var}[z_k] = a_k^T S a_k$$

- The k th largest eigenvalue of S is the variance of the k th PC.
- The k th PC z_k retains the k th greatest fraction of the variation in the sample.

Algebraic derivation of PCs

- Main steps for computing PCs
 - Form the covariance matrix S .
 - Compute its eigenvectors: a_i $i=1$ d
 - The first p eigenvectors a_i $i=1$ p form the p PCs.
 - The transformation G consists of the p PCs:

$$G = [a_1, a_2, \dots, a_p]$$

PCA for image compression



p=1



p=2



p=4



p=8

p=16



p=32



p=64



p=100



**Original
Image**



Outline of lecture

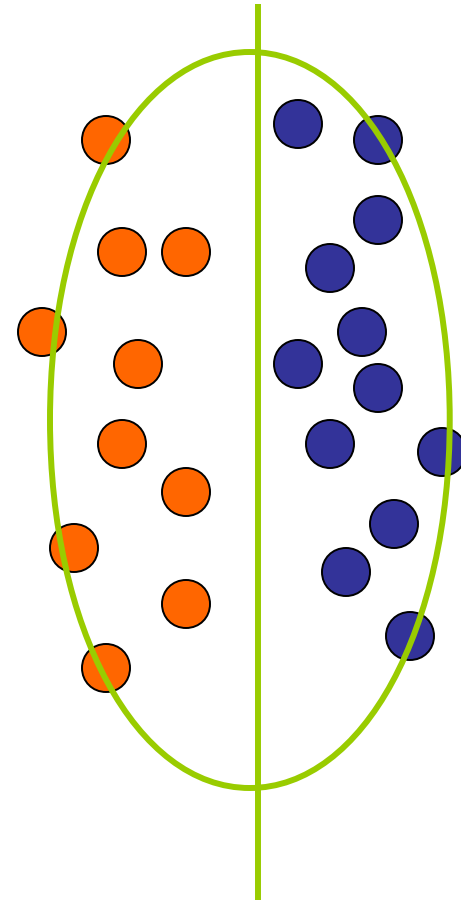
- What is feature reduction?
- Why feature reduction?
- Feature reduction algorithms
 - Principal Component Analysis (PCA)
 - **Linear Discriminant Analysis (LDA)**

Linear Discriminant Analysis

- First applied by M. Barnard at the suggestion of R. A. Fisher (1936), Fisher linear discriminant analysis (FLDA):
 - **Dimension reduction**
 - Finds linear combinations of the features $\mathbf{X} = X_1, \dots, X_d$ with large ratios of between-groups to within-groups sums of squares - discriminant variables;
 - **Classification**
 - Predicts the class of an observation \mathbf{X} by the class whose mean vector is closest to \mathbf{X} in terms of the discriminant variables

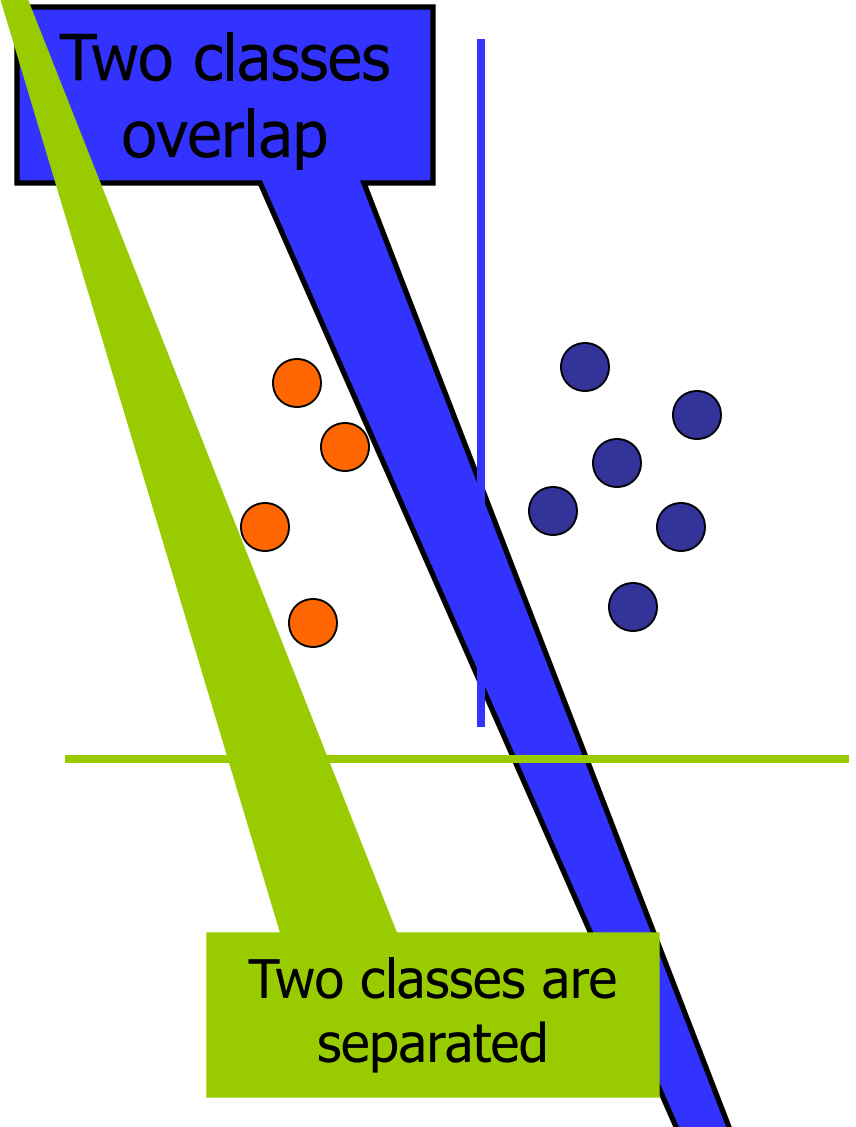
Is PCA a good criterion for classification?

- Data variation determines the projection direction
- What's missing?
 - Class information



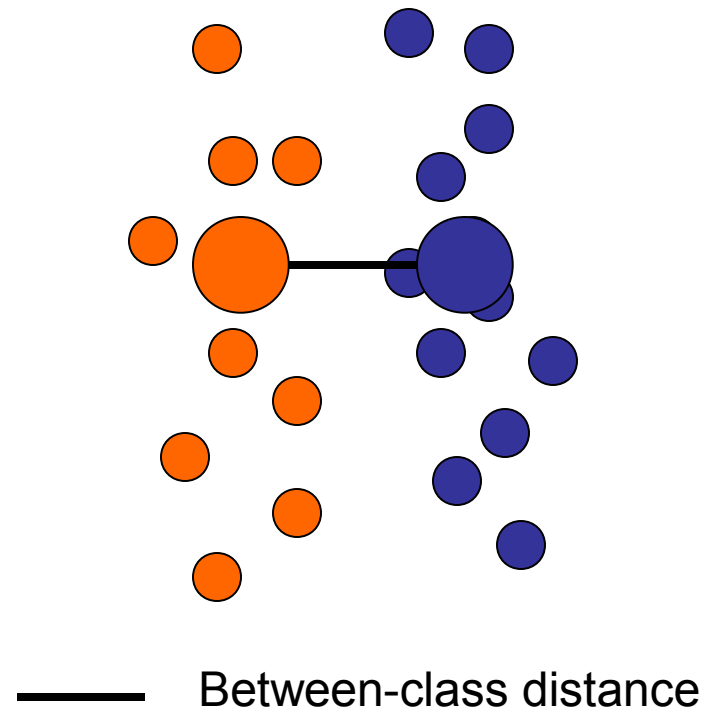
What is a good projection?

- Similarly, what is a good criterion?
 - Separating different classes



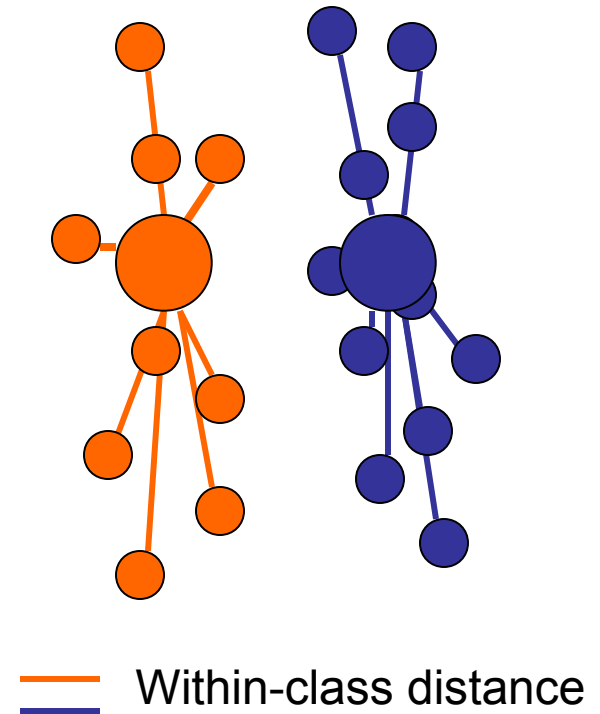
What class information may be useful?

- **Between-class distance**
 - Distance between the centroids of different classes



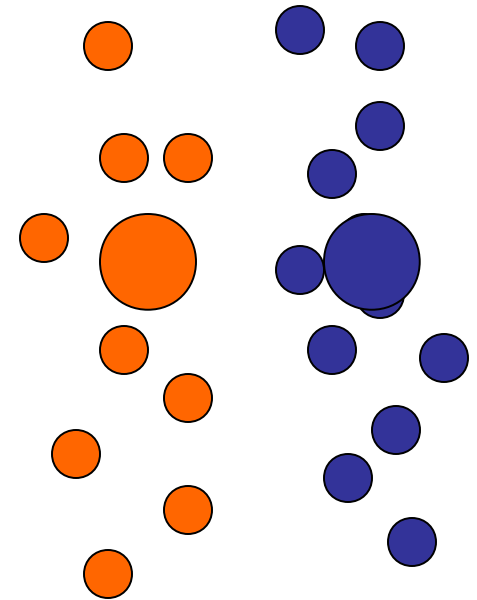
What class information may be useful?

- Between-class distance
 - Distance between the centroids of different classes
- Within-class distance
 - Accumulated distance of an instance to the centroid of its class



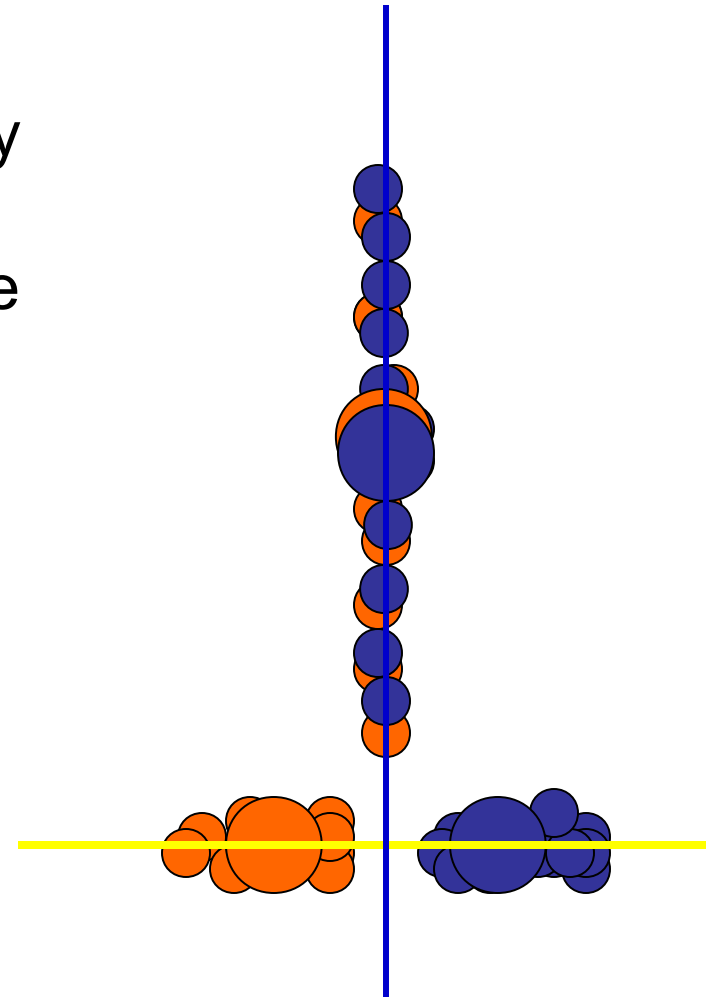
Linear discriminant analysis

- Linear discriminant analysis (LDA) finds most discriminant projection by maximizing between-class distance and minimizing within-class distance



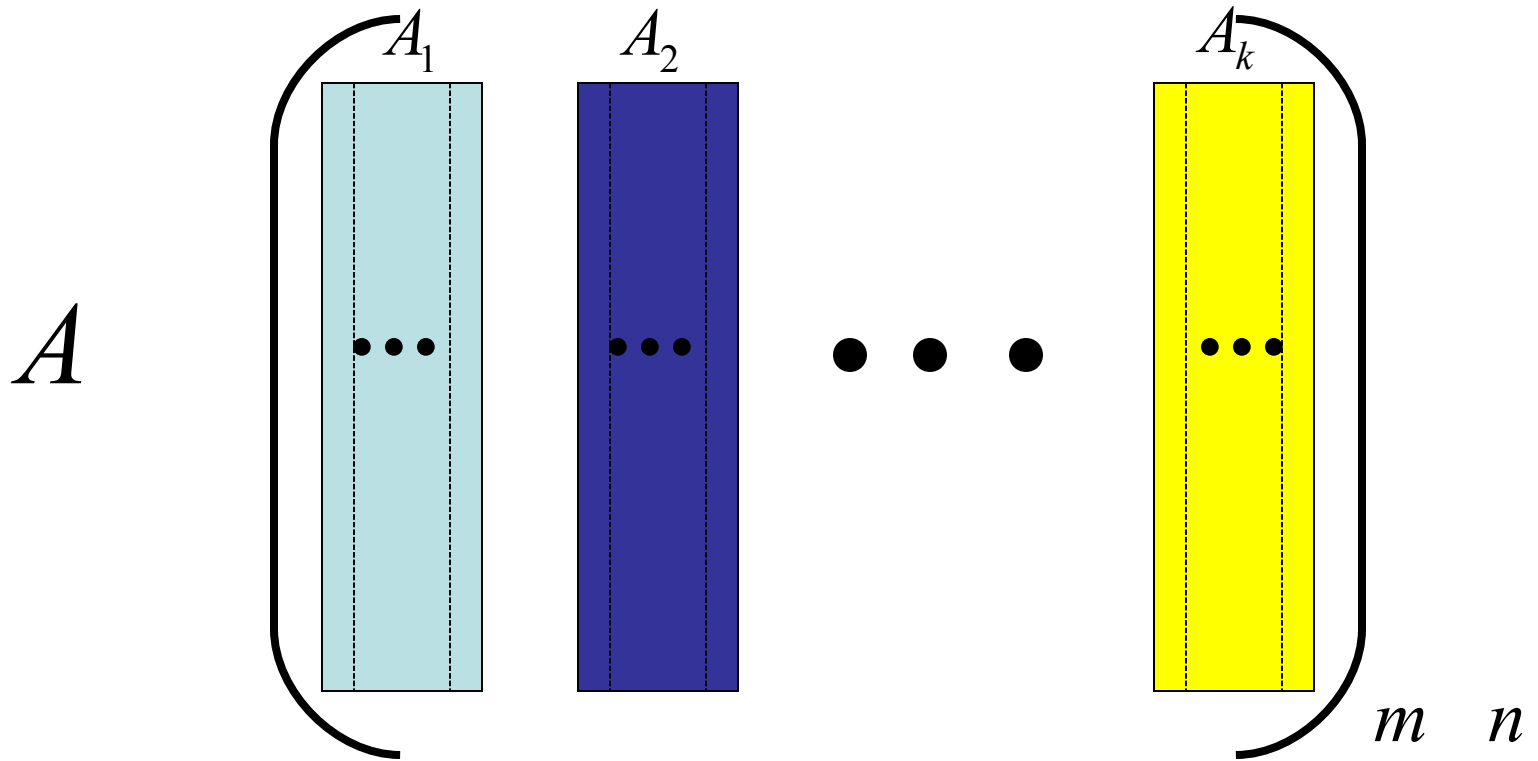
Linear discriminant analysis

- Linear discriminant analysis (LDA) finds most discriminant projection by maximizing between-class distance and minimizing within-class distance



Notations

Training data from different from 1, 2, ..., k

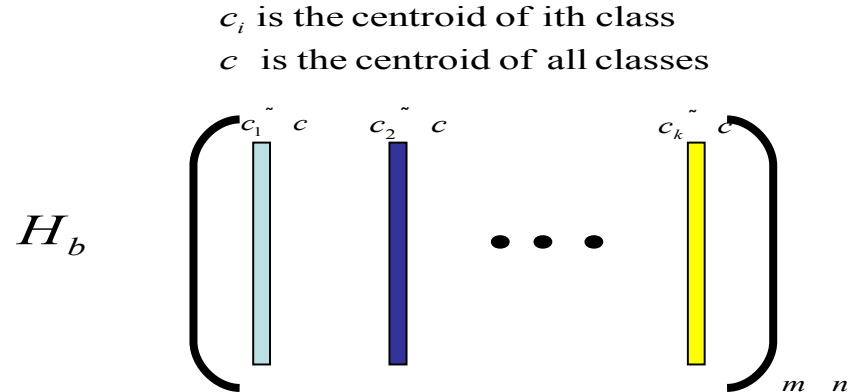


Data matrix

Notations

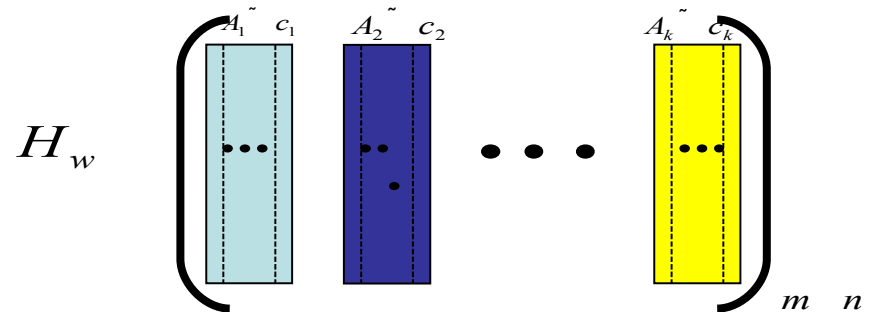
- Between-class scatter

$$S_b = H_b H_b^T$$



- Within-class scatter

$$S_w = H_w H_w^T$$



- Properties:

- Between-class distance = trace of between-class scatter (I.e., the summation of diagonal elements of the scatter)
- Within-class distance = trace of within-class scatter

Discriminant criterion

- Discriminant criterion in mathematical formulation

$$\arg \max_G \frac{\text{trace}(G^T S_b G)}{\text{trace}(G^T S_w G)}$$

$$S_b$$

- Between-class scatter matrix

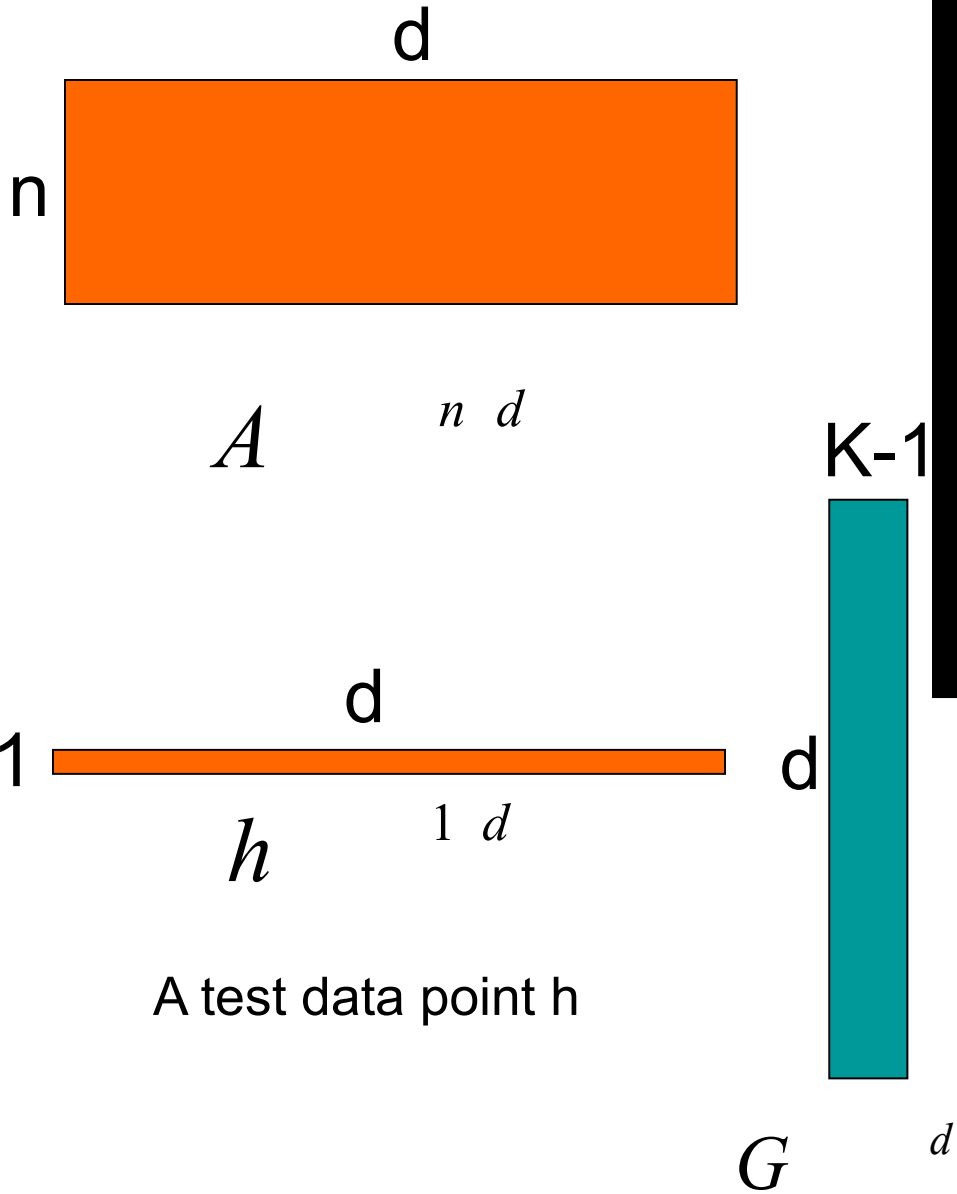
$$S_w$$

- Within-class scatter matrix

- The optimal transformation is given by solving a generalized eigenvalue problem

$$S_w^{-1} S_b$$

Graphical view of



Applications

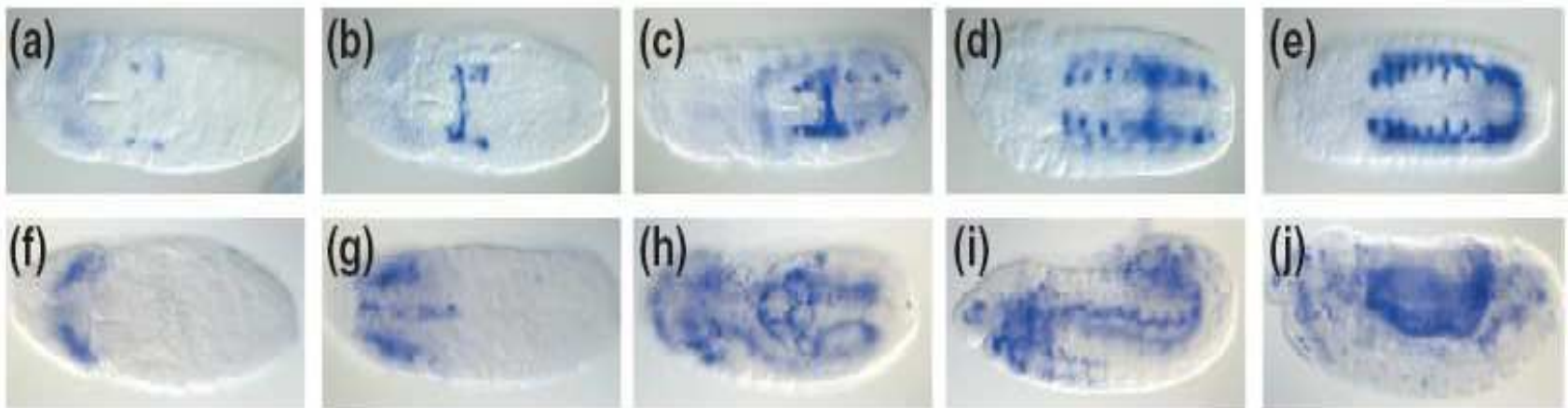
- Face recognition
 - Belhumeur *et al.*, PAMI'97
- Image retrieval
 - Swets and Weng, PAMI'96
- Gene expression data analysis
 - Dudoit *et al.*, JASA'02; Ye *et al.*, TCBB'04
- Protein expression data analysis
 - Lilien *et al.*, Comp. Bio.'03
- Text mining
 - Park *et al.*, SIMAX'03; Ye *et al.*, PAMI'04
- Medical image analysis
 - Dundar, SDM'05

Issues in LDA

- S_w is required to be nonsingular.
 - Singularity or undersampled problem (when $n < d$)
 - Example: gene expression data (d is around few thousands and n is around few hundreds), images, text documents
- Approaches
 - PCA+LDA (PCA: Principal Component Analysis)
 - Regularized LDA:
 - Uncorrelated LDA
 - Orthogonal LDA

Summary

- Feature reduction is an important pre-processing step in many applications.
- Unsupervised versus supervised
 - PCA and LDA
- Research problems:
 - Semi-supervised feature reduction
 - Nonlinear feature reduction
 - Determination of the reduced dimension in PCA



· Computational and theoretical issues in machine learning and data mining

- Dimensionality reduction
- Clustering and classification
- Semi-supervised learning
- Kernel methods

- Their applications to bioinformatics

- Expression pattern images
- Microarray gene expression data
- Protein sequences and structures

(a-e) Series of five embryos stained with a probe (*bgn*)

(f-j) Series of five embryos stained with a probe

(*CG4829*)

- *Are there any other expression patterns that are similar to the pattern I have observed?*
- *Which genes show extensive overlap in expression patterns?*
- *What is the extent and location of the overlap between gene expression patterns?*
- *Is there a change in the expression pattern of a gene when another gene's expression is altered?*

Project:

Machine learning approaches for biological image informatics

To answer the above questions, investigators generally rely on their own, a collaborator's, or senior mentor's knowledge, which has been gained by following the published literature over many years or even decades. It does not scale to enormous data.

We propose to develop computational approaches for answering these questions automatically.

*Nearest Neighbor
Algorithm*

Overview

- Instance-Based Learning

Comparison of Eager and Instance-Based Learning

- Instance Distances for Instance-Based Learning
- Nearest Neighbor (NN) Algorithm
- Advantages and Disadvantages of the NN algorithm
- Approaches to overcome the Disadvantages of the NN algorithm
- Combining Eager and Instance-Based Learning

Instance-Based Learning

- Learning = storing all training instances
- Classification = an instance gets a classification equal to the classification of the nearest instances to the instance.

Different Learning Methods

- **Eager Learning**

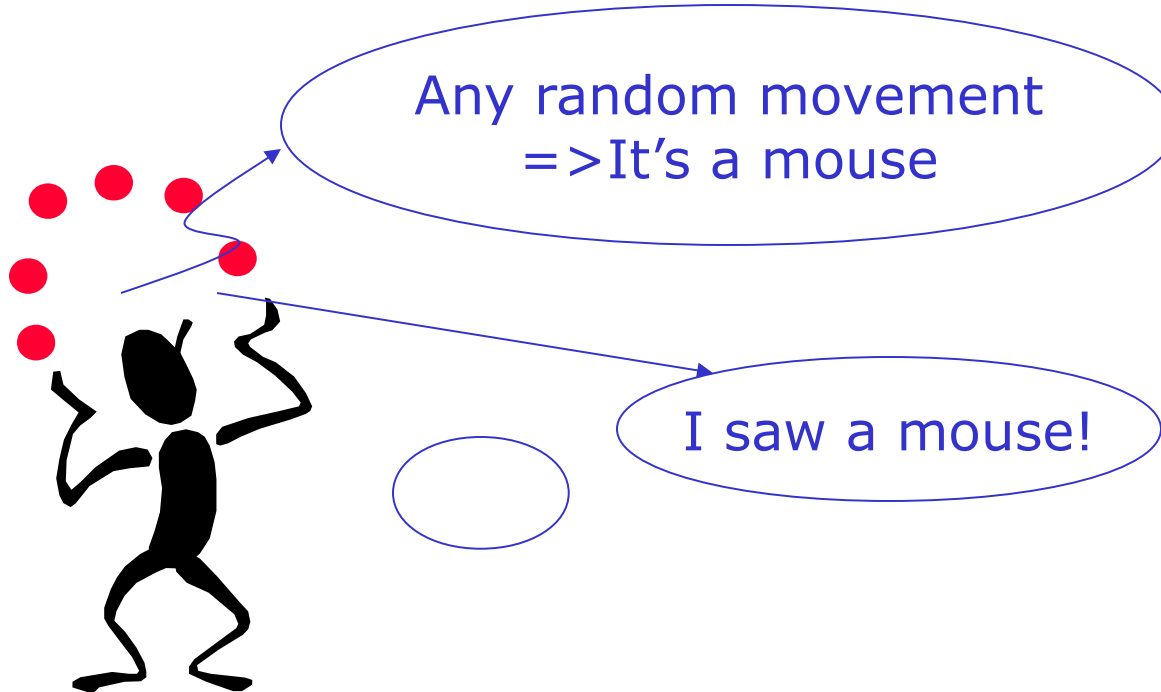
- Learning = acquiring explicit description of the target concepts on the whole training set;
- Classification = an instance gets a classification using the explicit description of the target concepts.

- **Instance-Based Learning (Lazy Learning)**

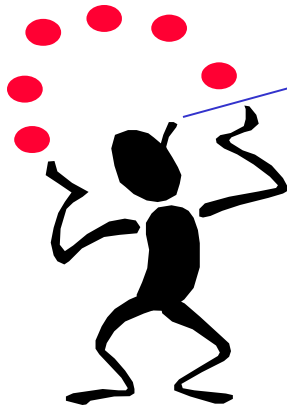
- Learning = storing all training instances
- Classification = an instance gets a classification equal to the classification of the nearest instances to the instance.

Different Learning Methods

- Eager Learning



Instance-Based Learning



Its very similar to a
Desktop!!



Nearest-Neighbor Algorithm (NN)

The Features of the Task of the NN Algorithm:

- the instance language L_i is a 1-CNF language with a set A with n attributes a_1, a_2, \dots, a_n . The domain of each attribute a_i , can be discrete or continuous.
- an instance x is represented as $\langle a_1(x), a_2(x), \dots, a_n(x) \rangle$, where $a_i(x)$ is the value of the attribute a_i for the instance x ;
- the concepts to be learned can be:
 - *discrete*. In this case we learn *discrete* function $f(x)$ and the co-domain C of the function consists of the concepts c to be learned.
 - *continuous*. In this case we learn *continuous* function $f(x)$ and the co-domain C of the function consists of the concepts c to be learned.

Distance Functions

The distance functions are composed from difference metrics d_a w.r.t. attributes a defined for each two instances x_i and x_j .

- If the attribute a is numerical, then :

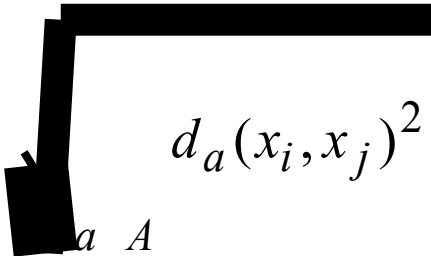
$$d_a(x_i, x_j) = \frac{|a(x_i) - a(x_j)|}{range_a}$$

- If the attribute a is discrete, then :

$$d_a(x_i, x_j) = \begin{cases} 0, & \text{if } a(x_i) = a(x_j) \\ 1, & \text{otherwise.} \end{cases}$$

Distance Functions

The main distance function for determining nearest neighbors is the Euclidean distance:



The diagram shows a right-angled triangle with a thick black outline. The right angle is at the bottom-left corner. The horizontal leg is on the right, and the vertical leg is on the left. The hypotenuse connects the top-right and bottom-left vertices. The label $d(x_i, x_j)$ is positioned to the left of the vertical leg. The label $d_a(x_i, x_j)^2$ is positioned to the right of the horizontal leg. The label a is positioned below the horizontal leg, and the label A is positioned below the vertical leg.

$$d(x_i, x_j) \quad d_a(x_i, x_j)^2$$

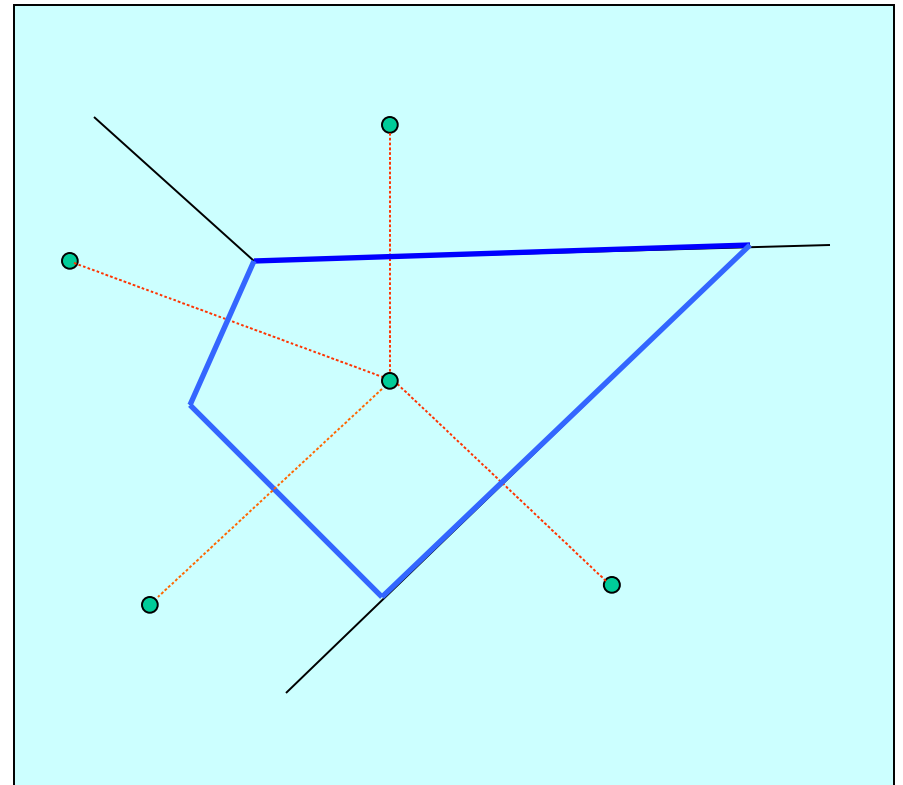
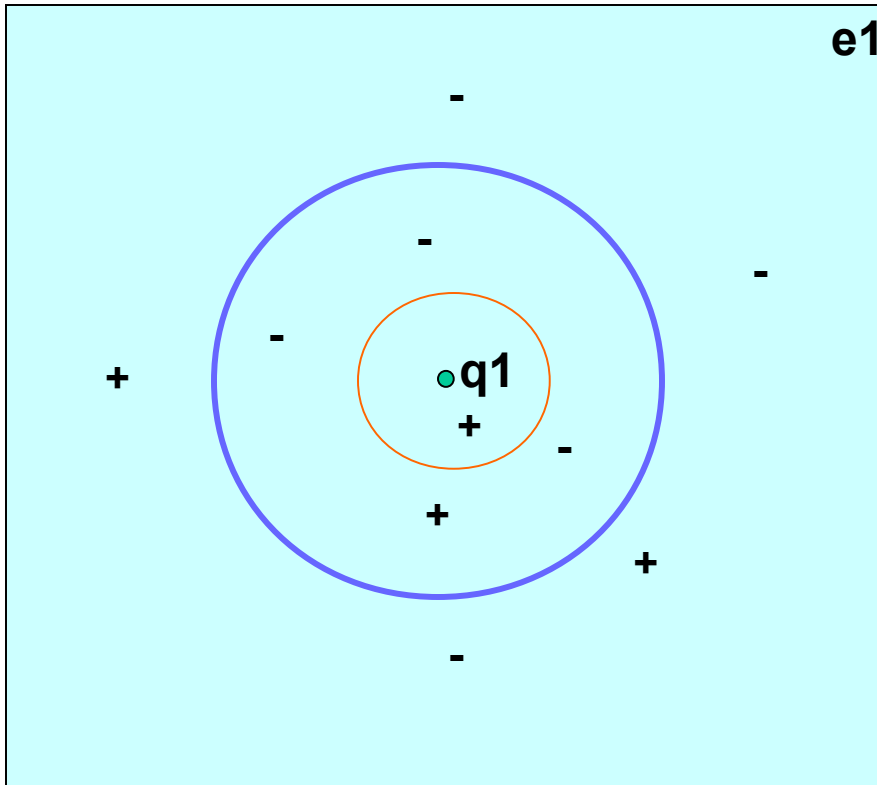
a A

k -Nearest-Neighbor Algorithm

The case of discrete set of classes.

1. Take the instance x to be classified
2. Find k nearest neighbors of x in the training data.
3. Determine the class c of the majority of the instances among the k nearest neighbors.
4. Return the class c as the classification of x .

Classification & Decision Boundaries



1-nn: q_1 is positive

5-nn: q_1 is classified as negative

1-nn:

Nearest Neighbor classification

Given:

Given a labeled sample of n feature vectors (call X)

A distance measure (say the Euclidian Distance)

To find:

The class label of a given feature vector x which is not in X

Nearest Neighbor classification (contd.)

The NN rule:

Find the point y in X which is nearest to x

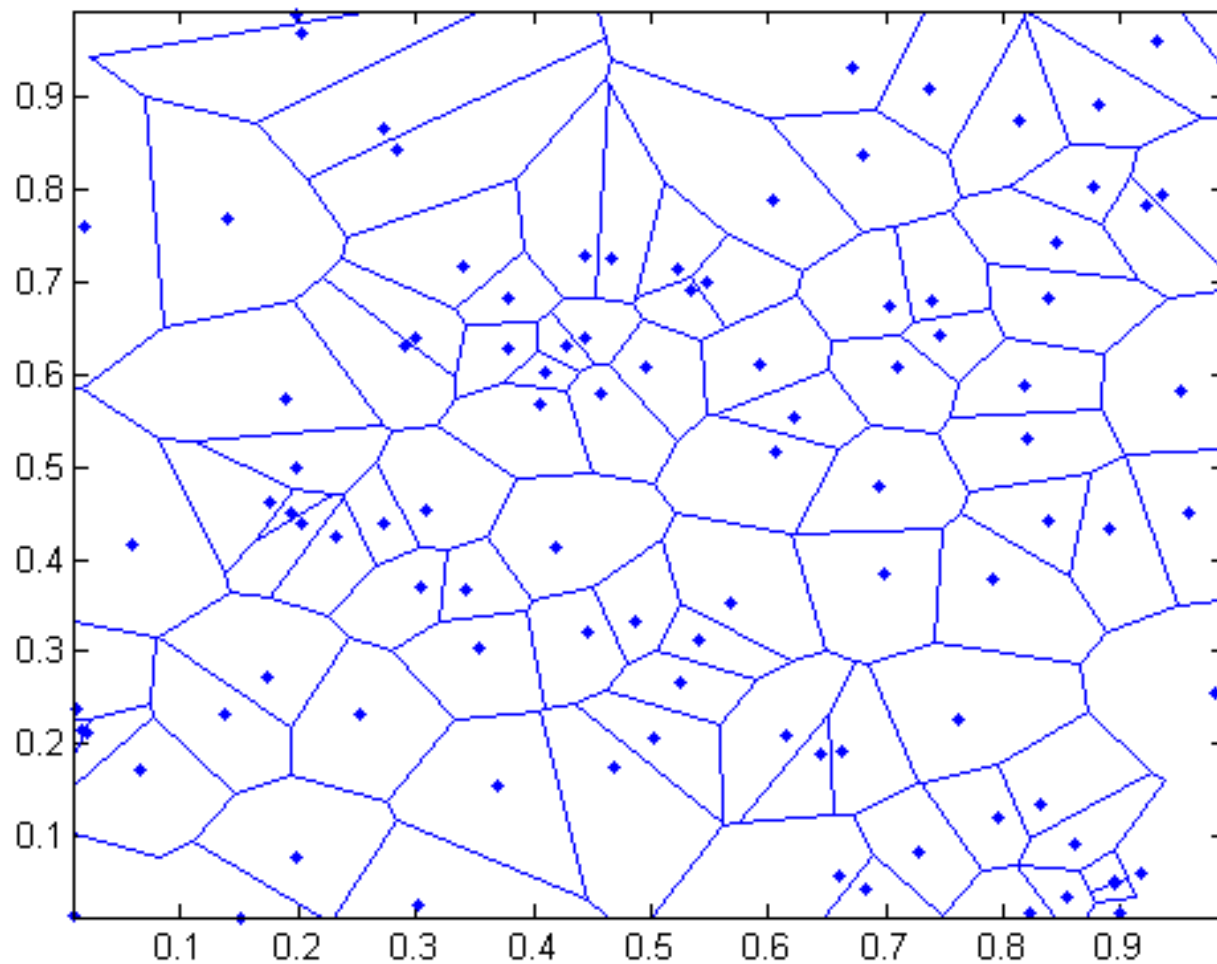
Assign the label of y to x

Nearest Neighbor classification (contd.)

This rule allows us to partition the feature space into cells consisting of all points closer to a given training point x

All points in such cells are labeled by the class of the training point. This partitioning is called a **Voronoi Tessellation**

Nearest Neighbor classification (contd.)



Voronoi Cells in 2d

k -Nearest-Neighbor Algorithm

The case of continuous set of classes.

1. Take the instance x to be classified
2. Find k nearest neighbors of x in the training data.
3. Return the average of the classes of the k nearest neighbors as the classification of x .

Distance Weighted Nearest-Neighbor Algorithm

The case of discrete set of classes.

1. Take the instance x to be classified
2. Determine for each class c the sum
3. Return the class c with the greater S_c .

$$D_c = \sum_{x_c \text{ belongs to } c} \frac{1}{d(x, x_c)^2}$$

Advantages of the NN Algorithm

- the NN algorithm can estimate complex target concepts locally and differently for each new instance to be classified;
- the NN algorithm provides good generalisation accuracy on many domains;
- the NN algorithm learns very quickly;
- the NN algorithm is robust to noisy training data;
- the NN algorithm is intuitive and easy to understand which facilitates implementation and modification.

Disadvantages of the NN Algorithm

- the NN algorithm has large storage requirements because it has to store all the data;
- the NN algorithm is slow during instance classification because all the training instances have to be visited;
- the accuracy of the NN algorithm degrades with increase of noise in the training data;
- the accuracy of the NN algorithm degrades with increase of irrelevant attributes.

Classification

k-nearest neighbor
classifier

Naïve Bayes

Logistic Regression

Support Vector Machines

NEAREST NEIGHBOR CLASSIFICATION

Instance-Based Classifiers

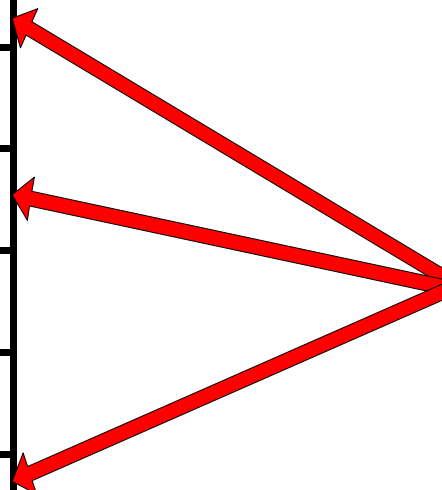
Set of Stored Cases

Atr1	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- Store the training records
- Use training records to predict the class label of unseen cases

Unseen Case

Atr1	AtrN

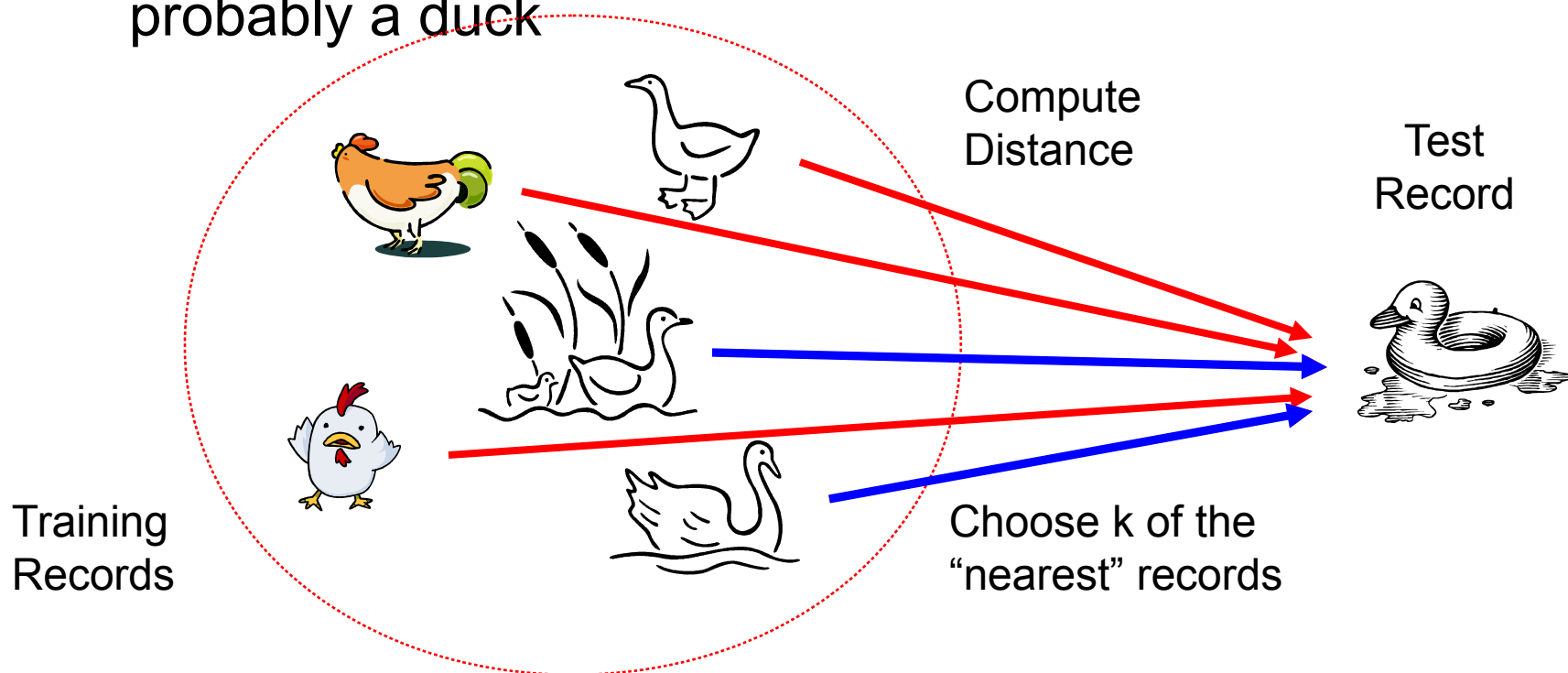


Instance Based Classifiers

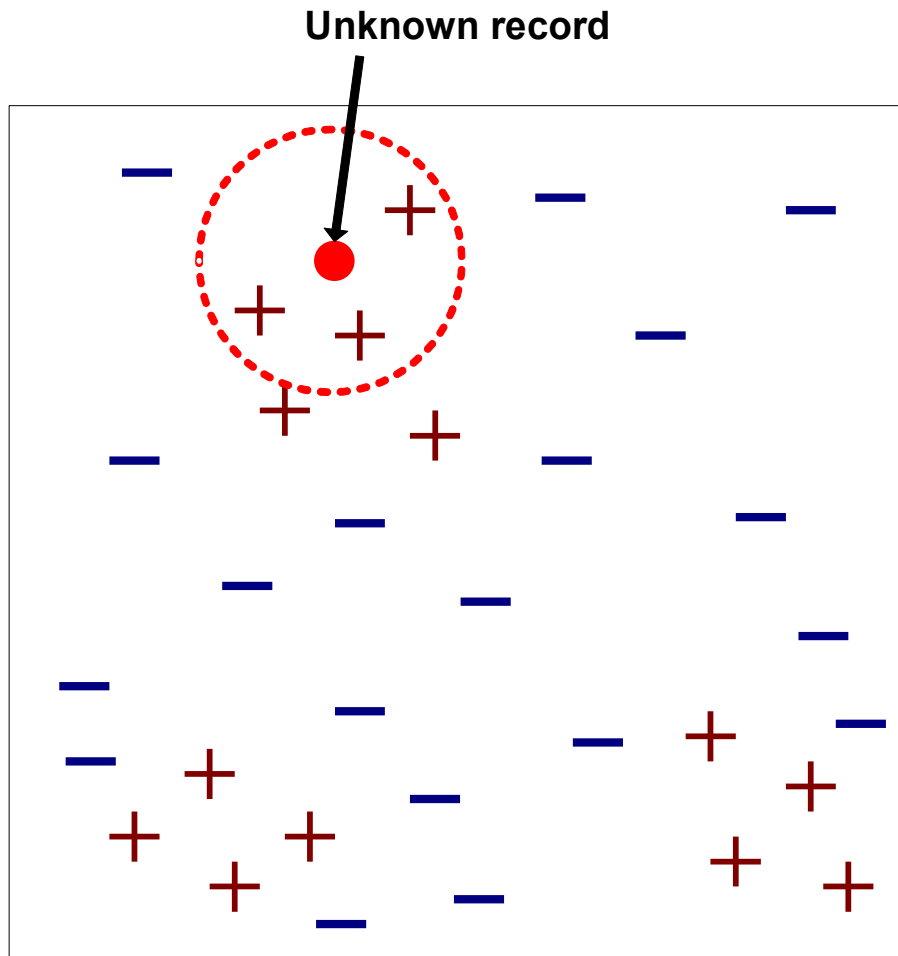
- Examples:
 - Rote-learner
 - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
 - Nearest neighbor
 - Uses k “closest” points (nearest neighbors) for performing classification

Nearest Neighbor Classifiers

- Basic idea:
 - If it walks like a duck, quacks like a duck, then it's probably a duck

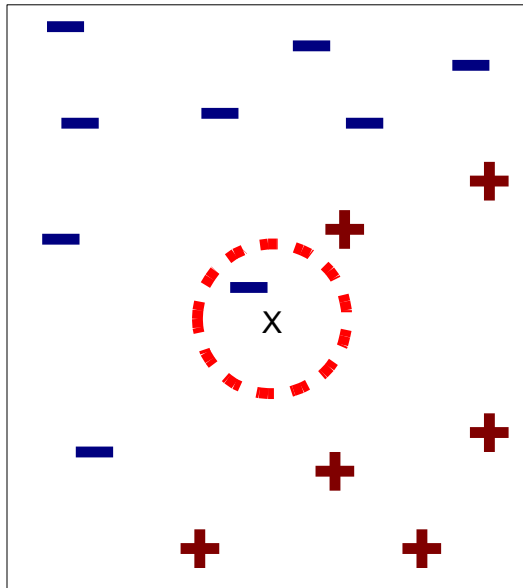


Nearest-Neighbor Classifiers

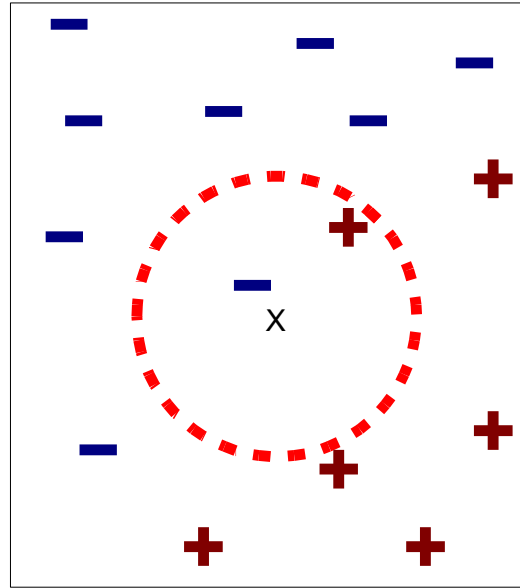


- Requires three things
 - The set of stored records
 - **Distance Metric** to compute distance between records
 - The value of **k , the number of nearest neighbors** to retrieve
- To classify an unknown record:
 - **Compute distance** to other training records
 - Identify **k** nearest neighbors
 - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

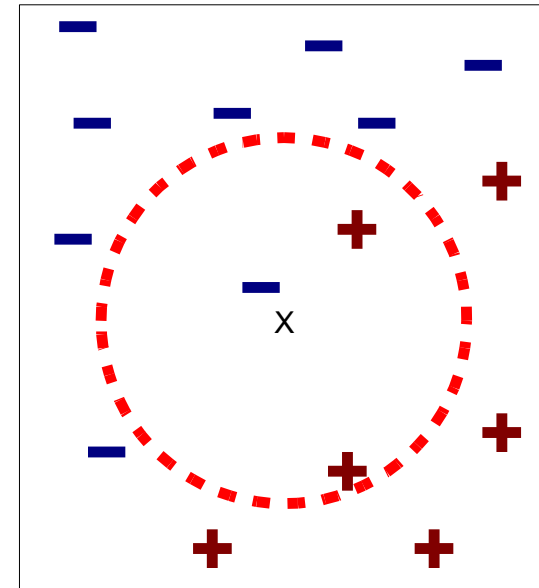
Definition of Nearest Neighbor



(a) 1-nearest neighbor



(b) 2-nearest neighbor

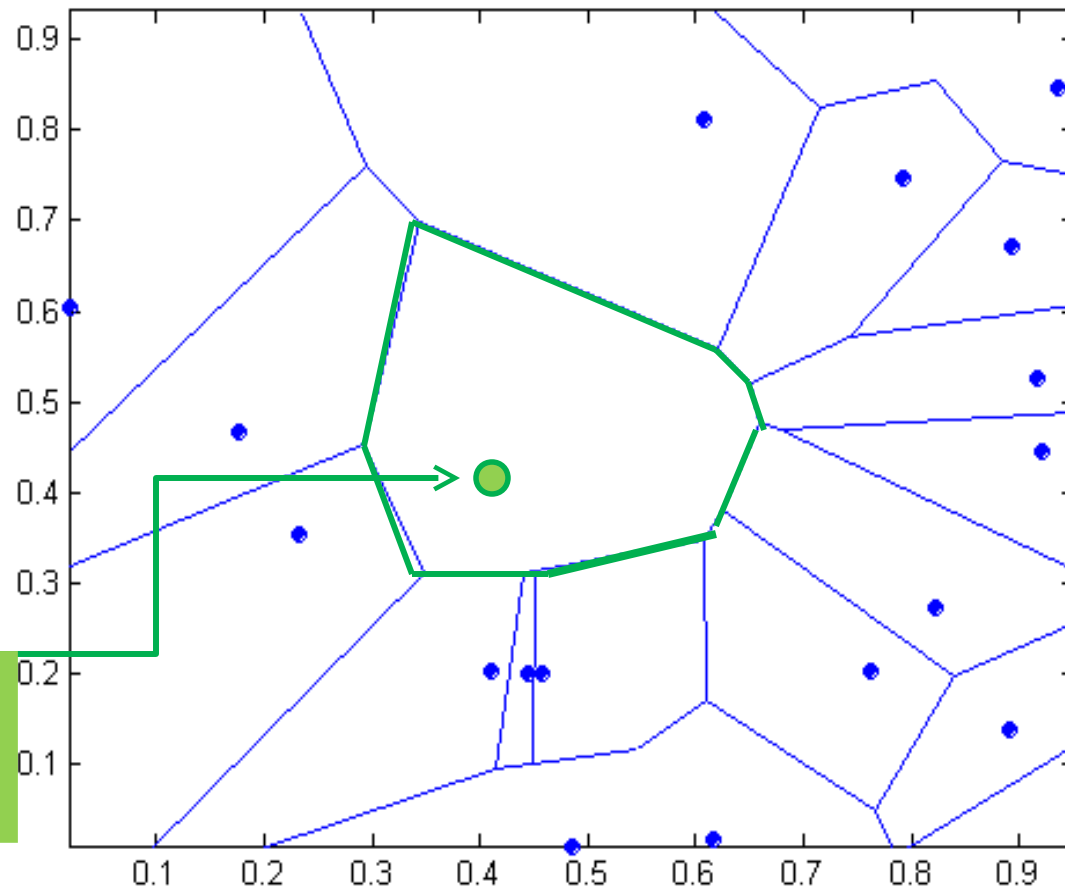


(c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

1 nearest-neighbor

Voronoi Diagram defines the classification boundary



The area takes the class of the green point

Nearest Neighbor Classification

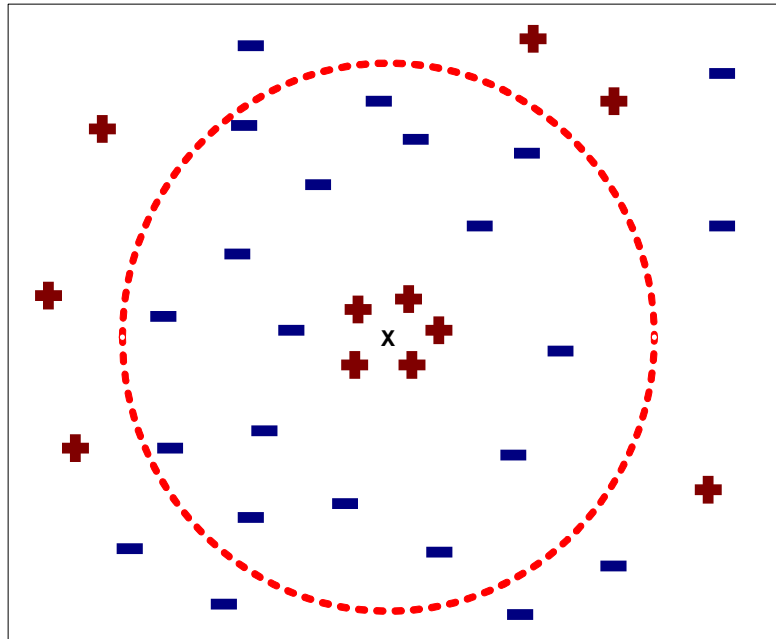
- Compute distance between two points:
 - Euclidean distance

$$d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
 - take the majority vote of class labels among the k-nearest neighbors
 - Weigh the vote according to distance
 - weight factor $w = 1/d^2$

Nearest Neighbor Classification...

- Choosing the value of k :
 - If k is too small, sensitive to noise points
 - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification...

- Scaling issues
 - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
 - Example:
 - height of a person may vary from 1.5m to 1.8m
 - weight of a person may vary from 90lb to 300lb
 - income of a person may vary from \$10K to \$1M

Nearest Neighbor Classification...

- Problem with Euclidean measure:
 - High dimensional data
 - **curse of dimensionality**
 - Can produce counter-intuitive results

1 1 1 1 1 1 1 1 1 1 1 0

vs

1 0 0 0 0 0 0 0 0 0 0 0

0 1 1 1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 1

$d = 1.4142$

$d = 1.4142$

- ◆ Solution: Normalize the vectors to unit length

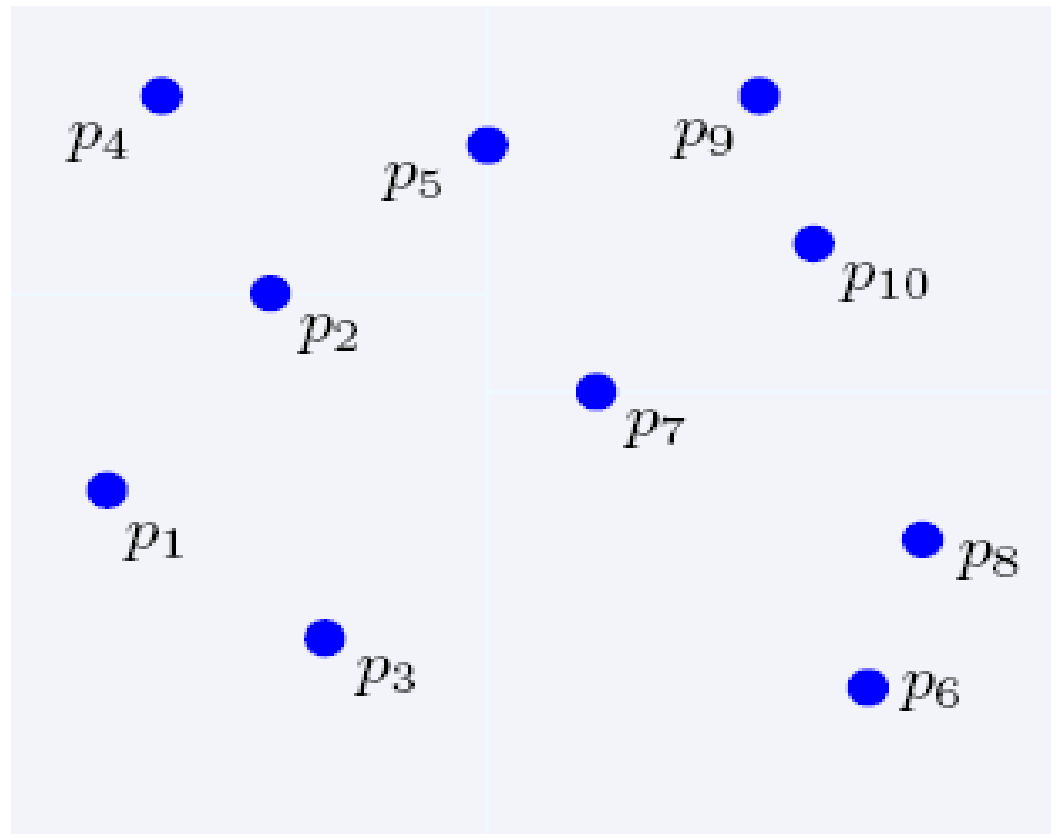
Nearest neighbor Classification...

- k-NN classifiers are **lazy learners**
 - It does not build models explicitly
 - Unlike **eager learners** such as decision tree induction and rule-based systems
- Classifying unknown records are relatively expensive
 - Naïve algorithm: $O(n)$
 - Need for structures to retrieve nearest neighbors fast.
 - The **Nearest Neighbor Search** problem.

Nearest Neighbor Search

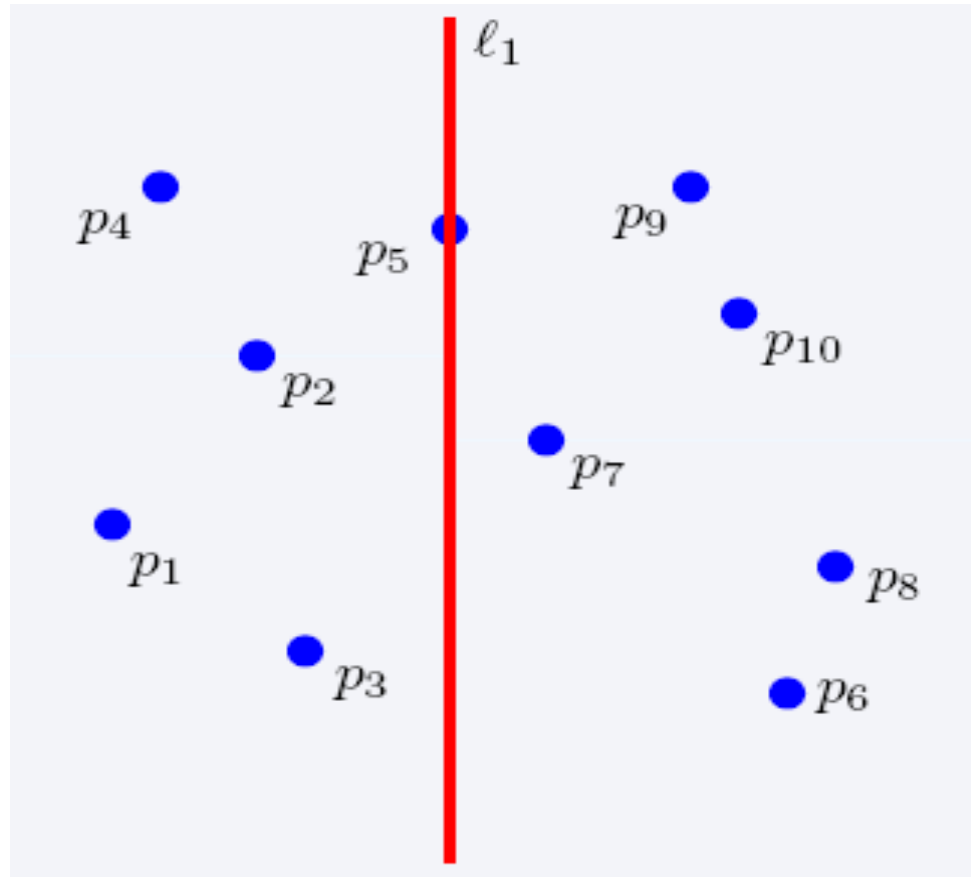
- Two-dimensional **kd-trees**
 - A data structure for answering nearest neighbor queries in R^2
- kd-tree construction algorithm
 - Select the x or y dimension (alternating between the two)
 - Partition the space into two with a line passing from the median point
 - Repeat recursively in the two partitions as long as there

Nearest Neighbor Search



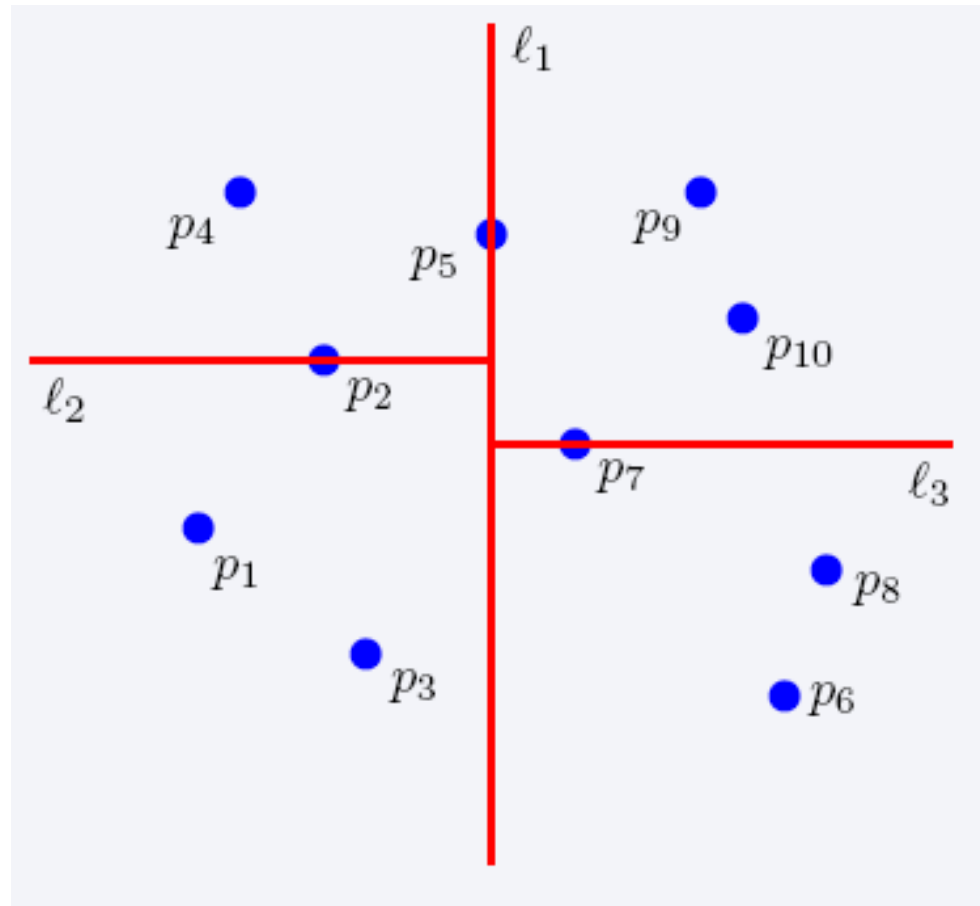
2-dimensional kd-trees

Nearest Neighbor Search



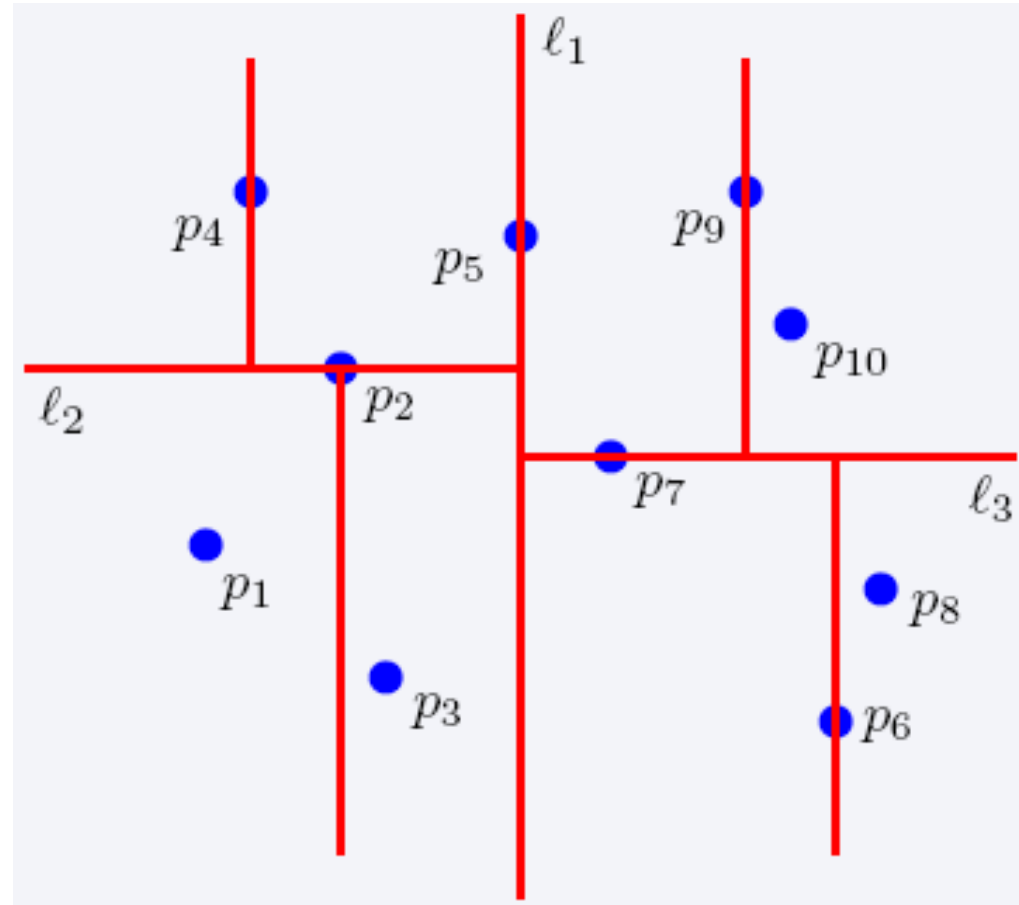
2-dimensional kd-trees

Nearest Neighbor Search



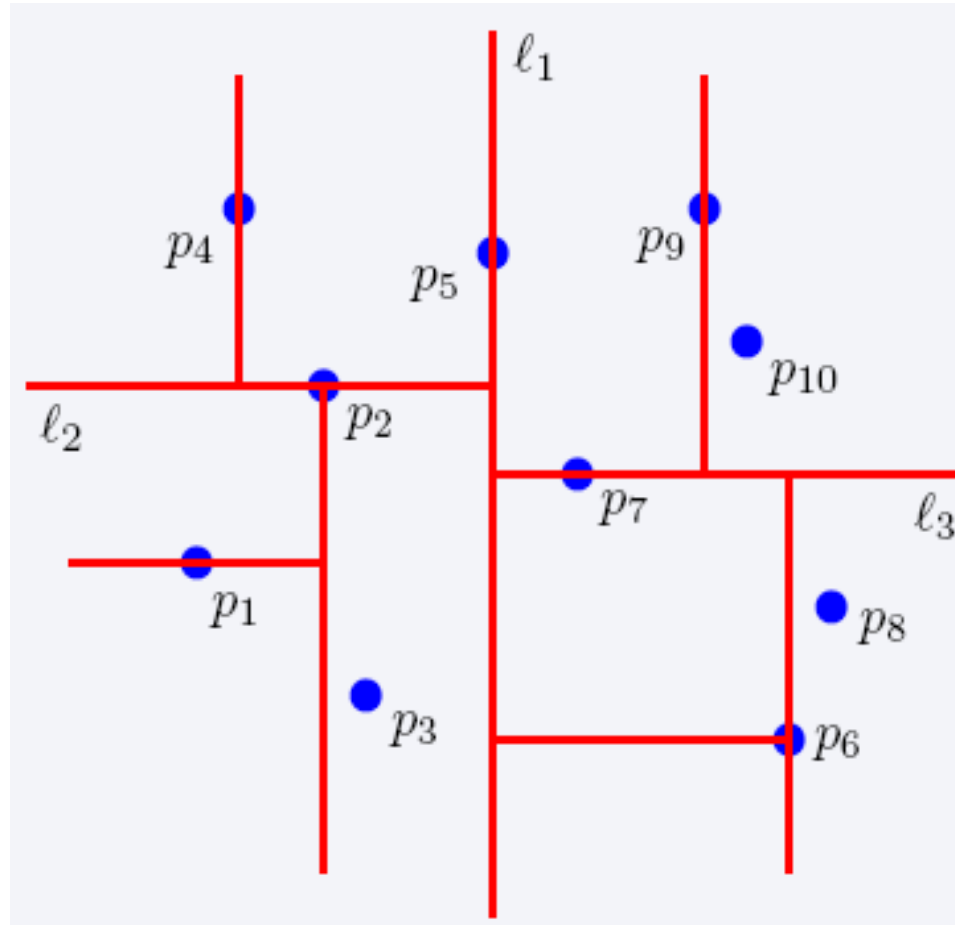
2-dimensional kd-trees

Nearest Neighbor Search



2-dimensional kd-trees

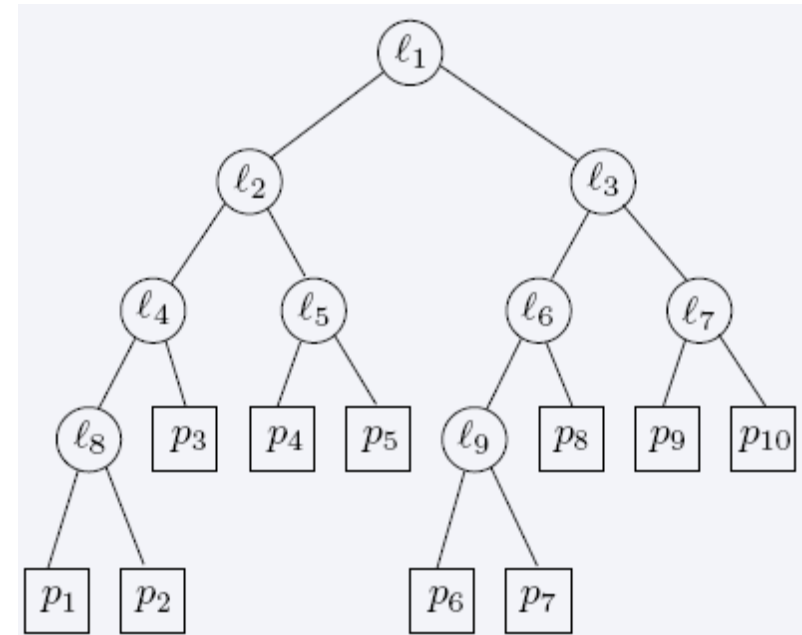
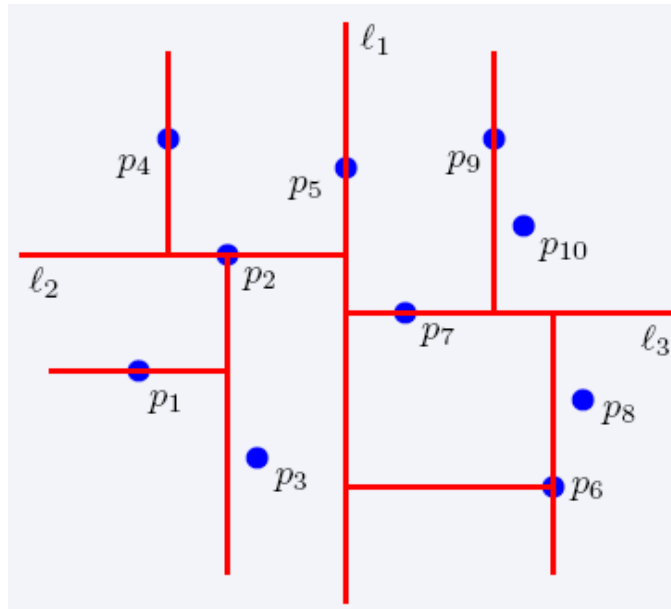
Nearest Neighbor Search



2-dimensional kd-trees

Nearest Neighbor Search

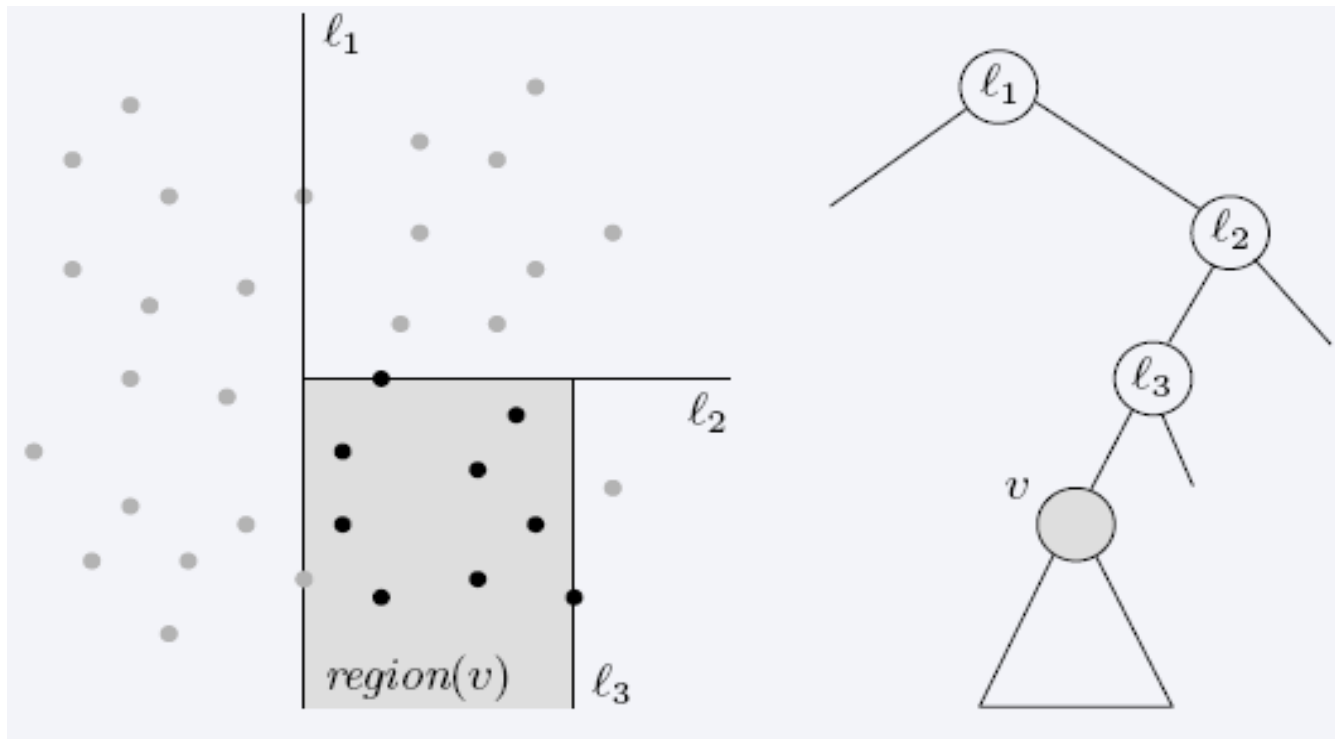
2-dimensional kd-trees



Nearest Neighbor Search

2-dimensional kd-trees

region(u) – all the black points in the subtree of u



Nearest Neighbor Search

2-dimensional kd-trees

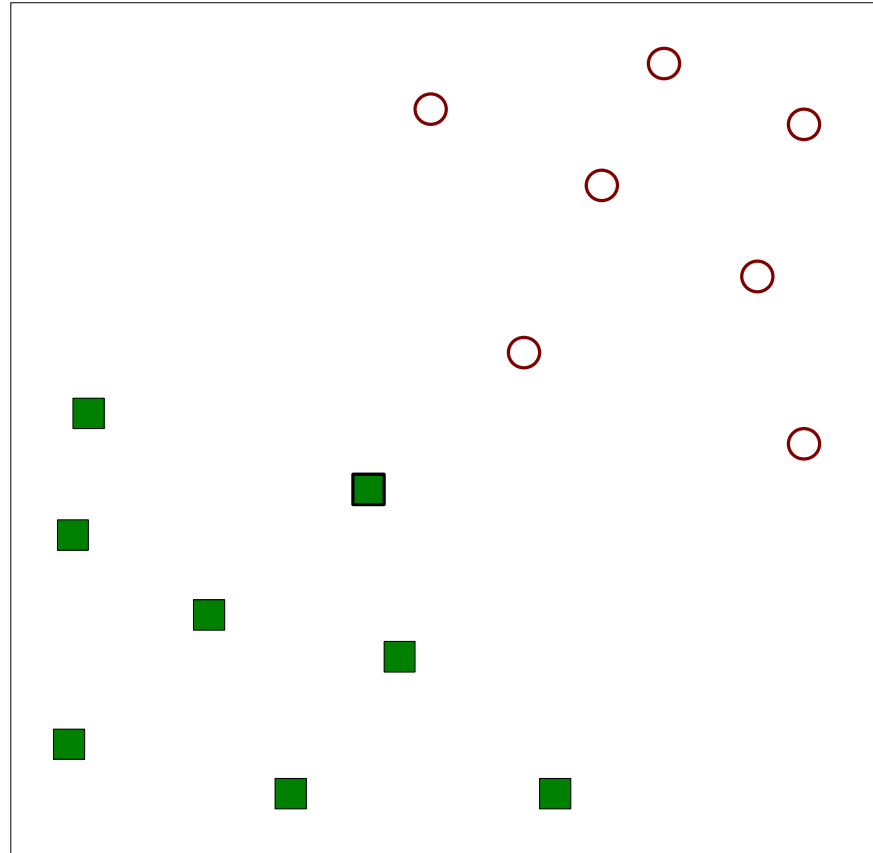
- A binary tree:
 - Size $O(n)$
 - Depth $O(\log n)$
 - Construction time $O(n \log n)$
 - Query time: worst case $O(n)$, but for many cases $O(\log n)$

Generalizes to d dimensions

- Example of Binary Space Partitioning

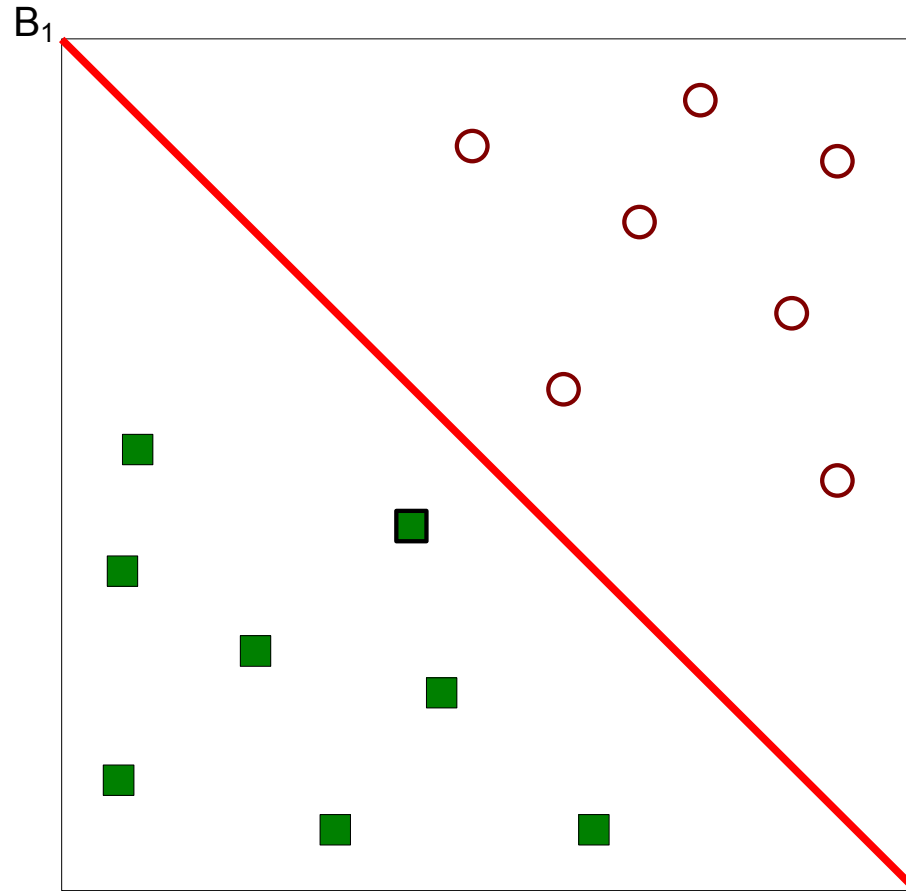
SUPPORT VECTOR MACHINES

Support Vector Machines



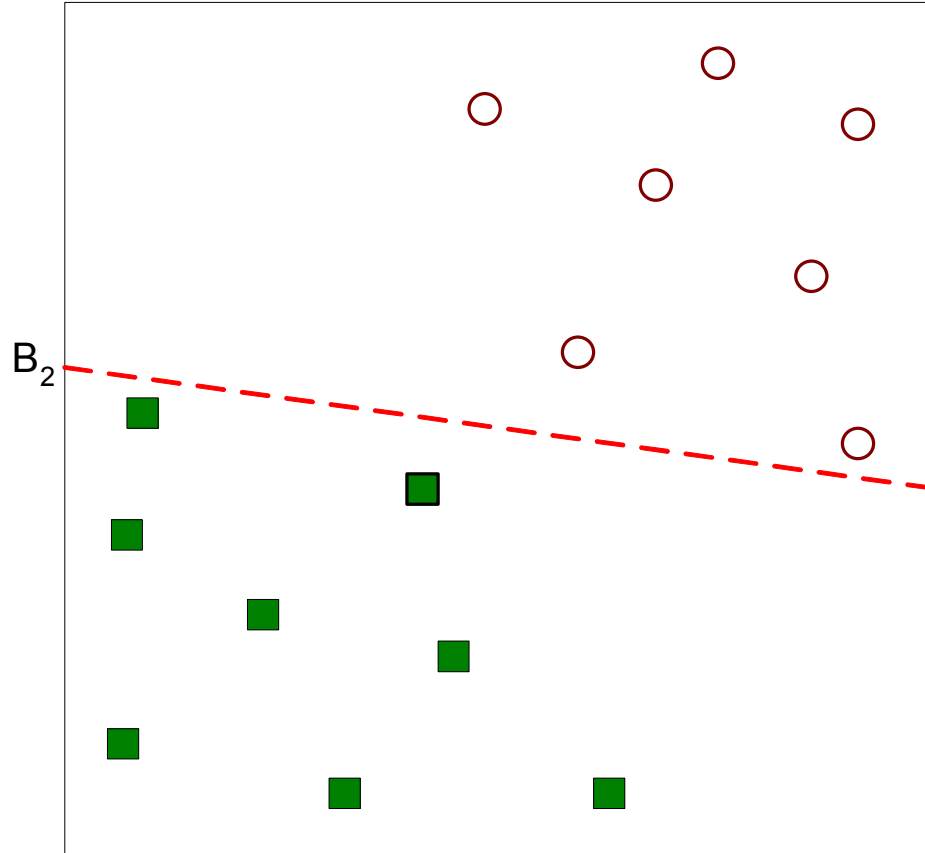
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



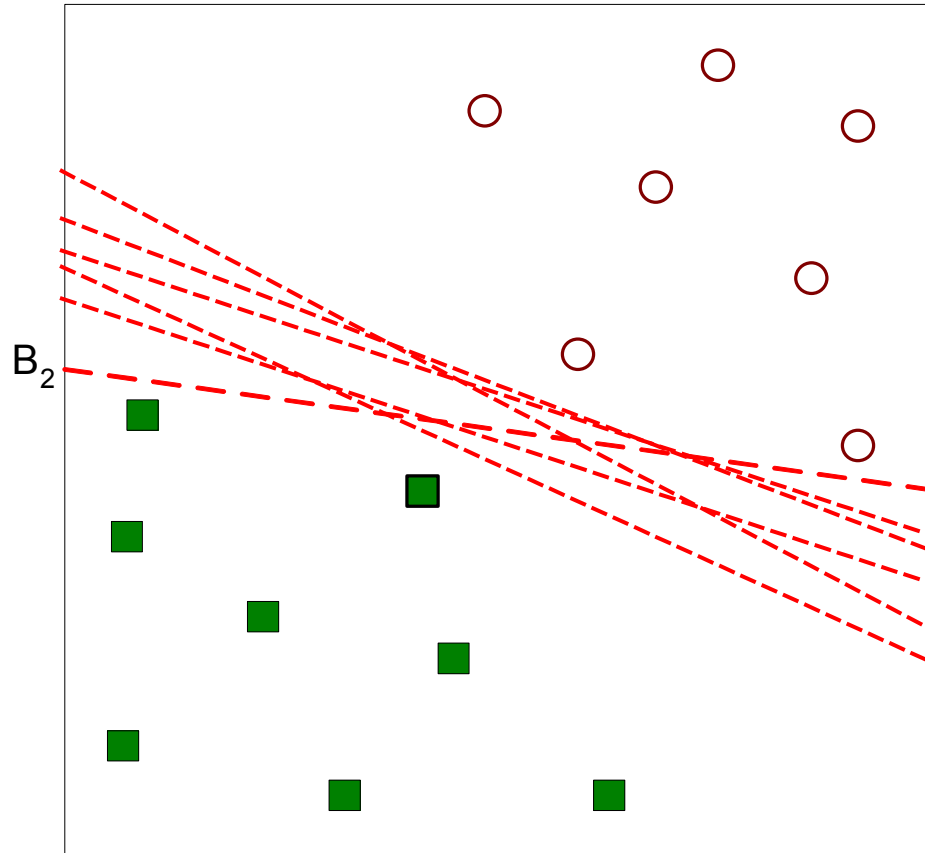
- One Possible Solution

Support Vector Machines



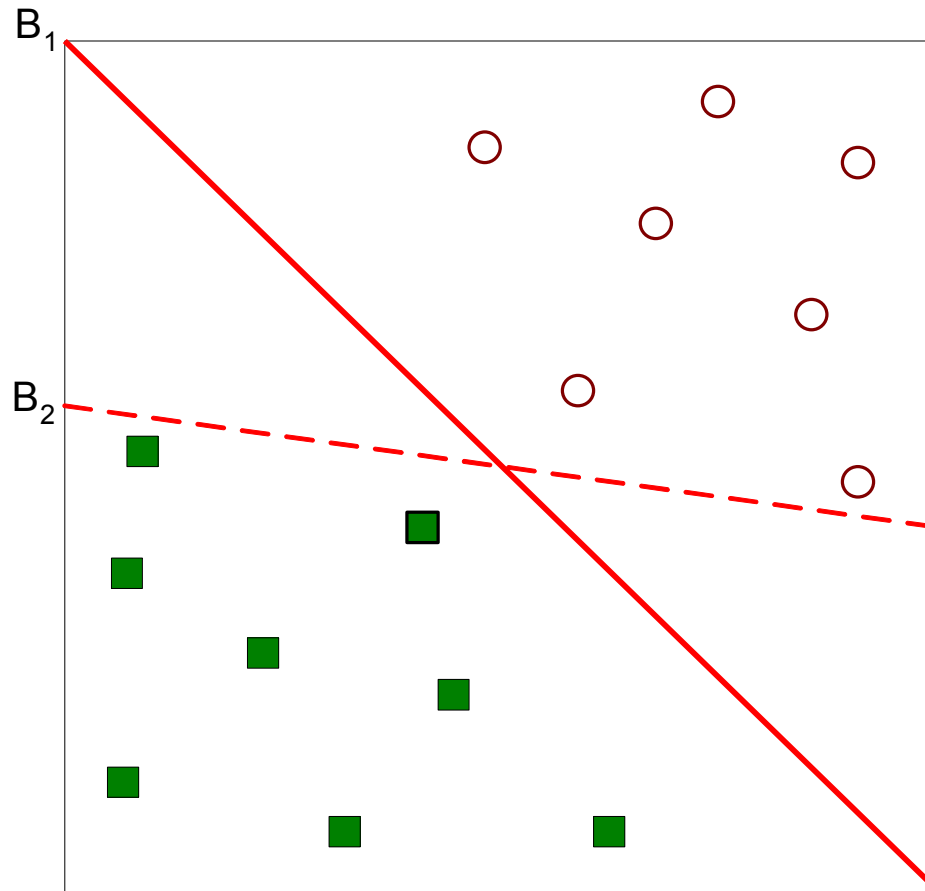
- Another possible solution

Support Vector Machines



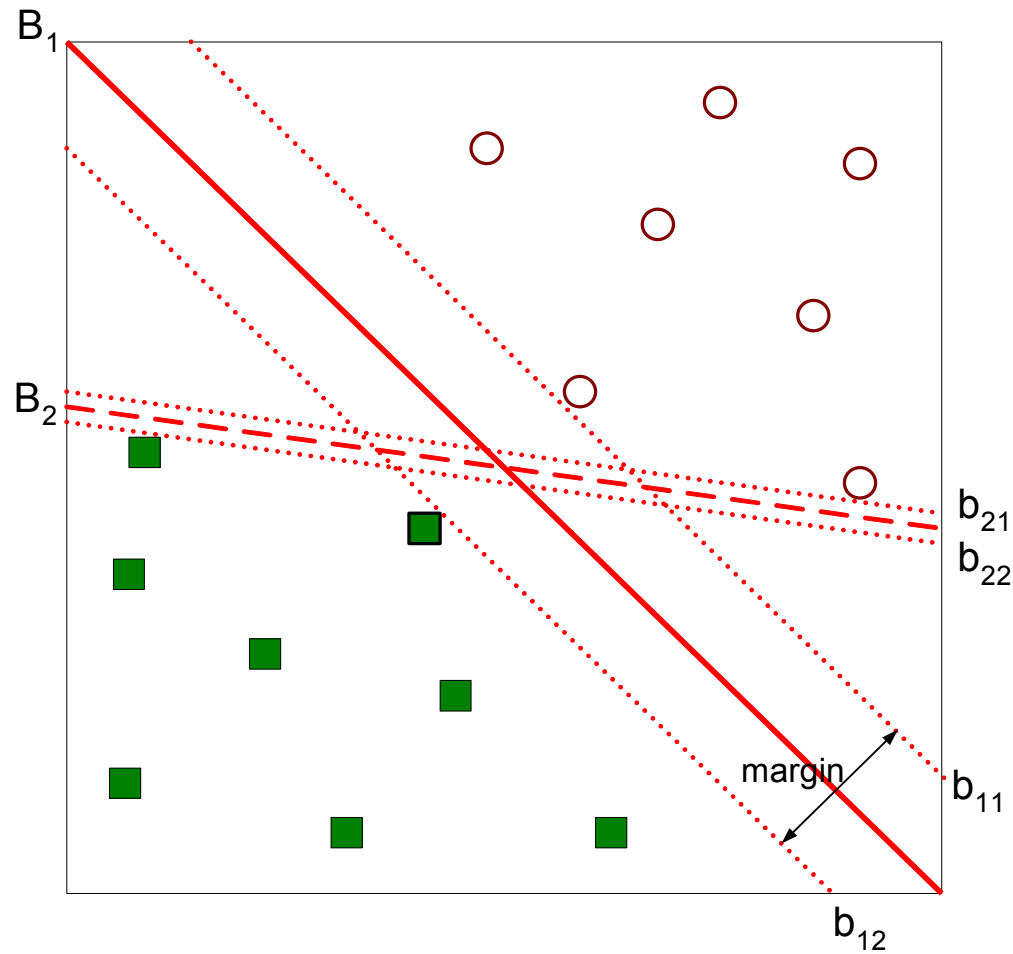
- Other possible solutions

Support Vector Machines



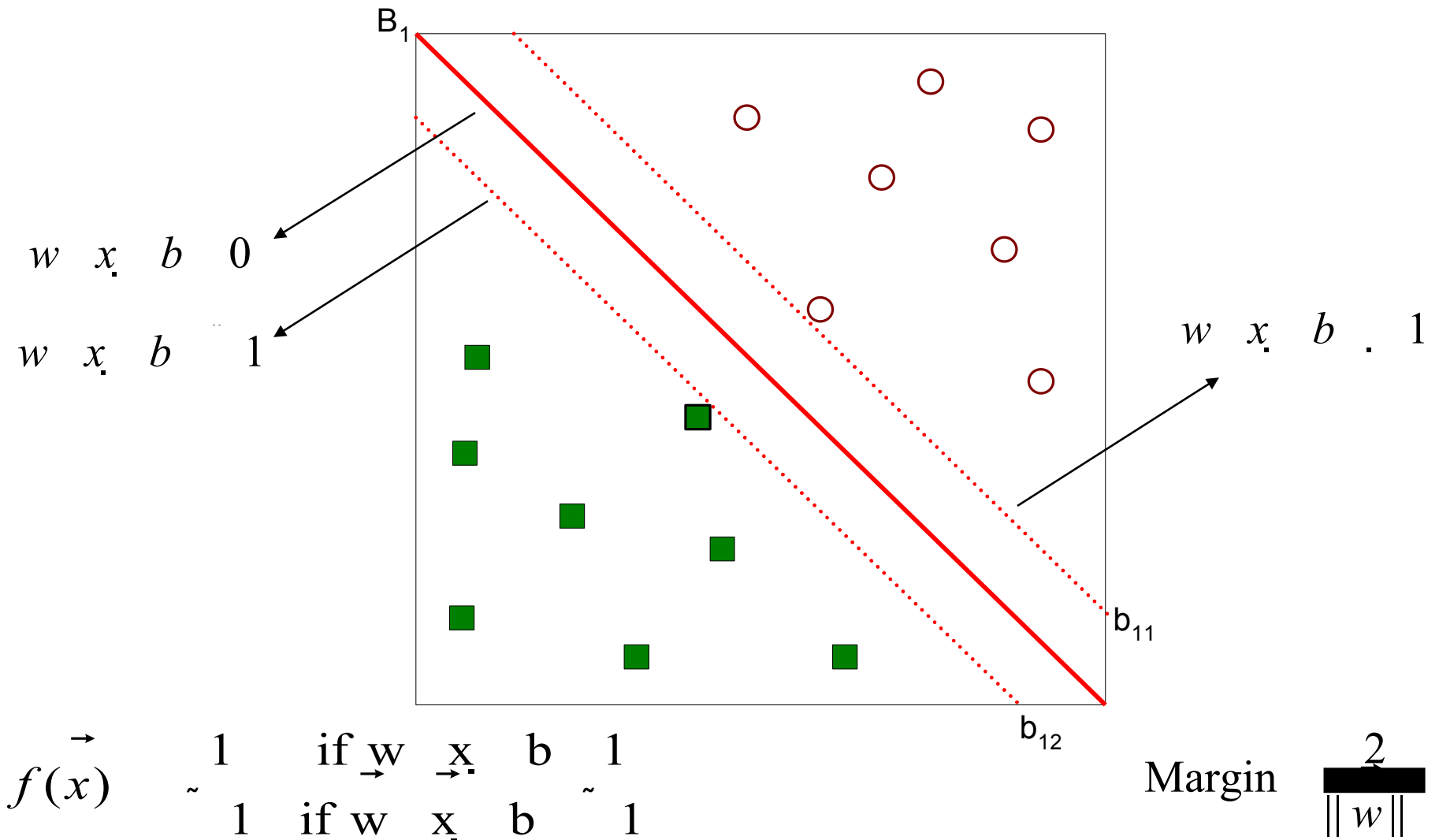
- Which one is better? B_1 or B_2 ?
- How do you define better?

Support Vector Machines



- Find hyperplane **maximizes** the margin => B_1 is better than B_2

Support Vector Machines

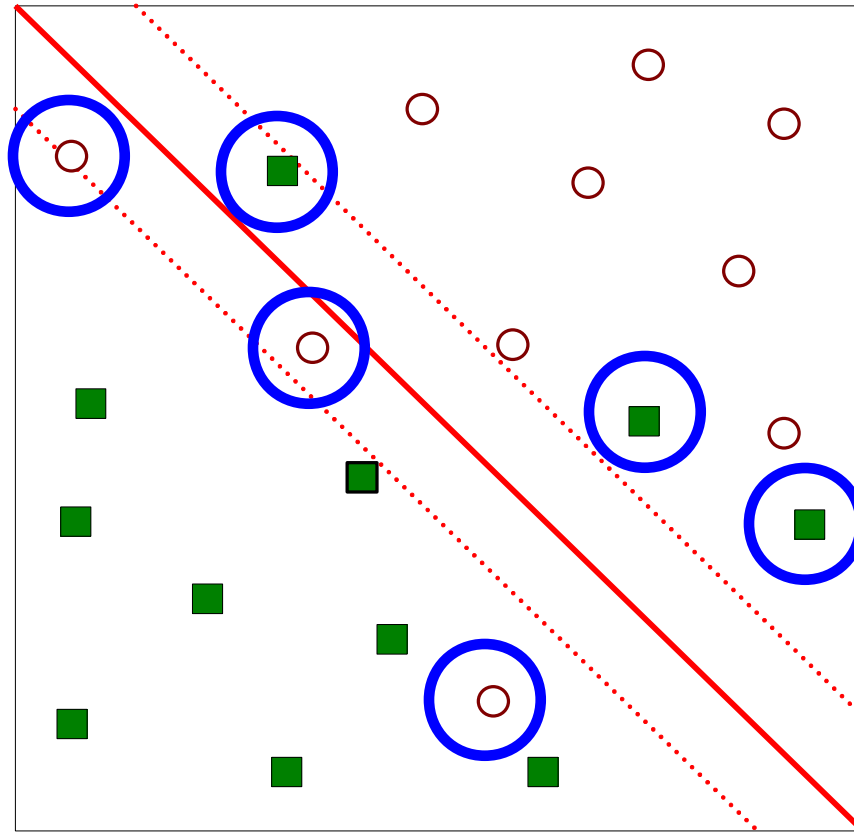


Support Vector Machines

- We want to maximize: Margin $\frac{2}{\|w\|^2}$
- Which is equivalent to minimizing: $L(w) = \frac{\|w\|^2}{2}$
- But subjected to the following constraints:
 - This is a **constrained optimization problem**

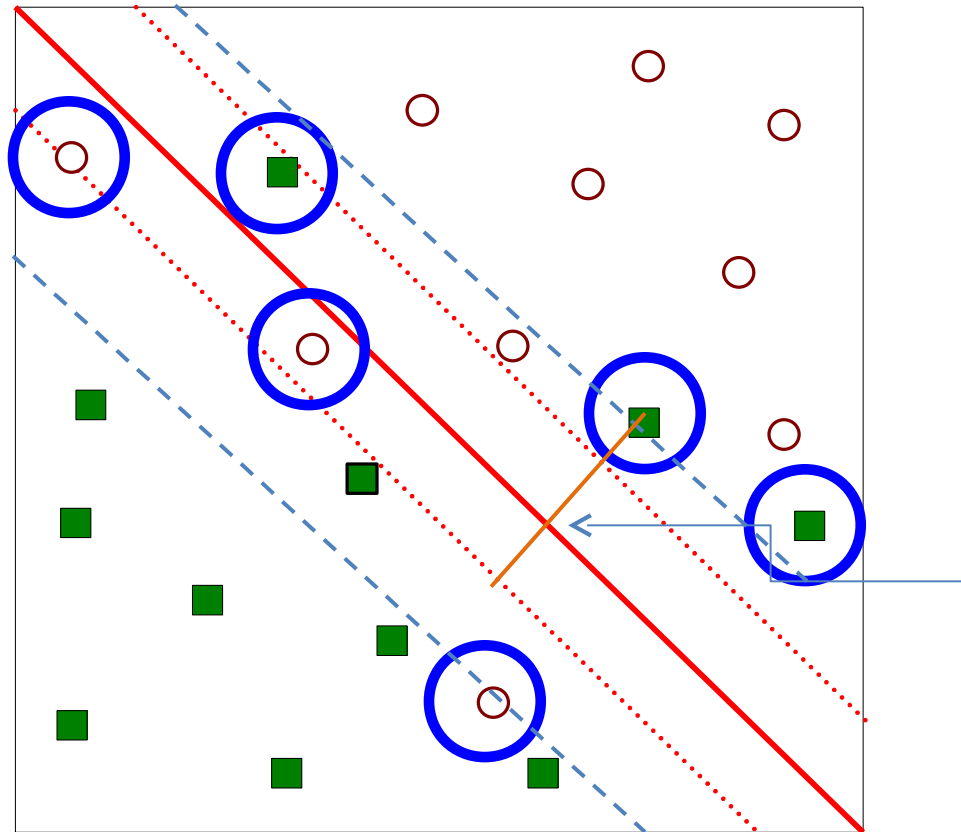
Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?



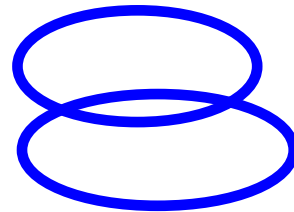
Support Vector Machines

- What if the problem is not linearly separable?
 - Introduce slack variables

- Need to minimize:

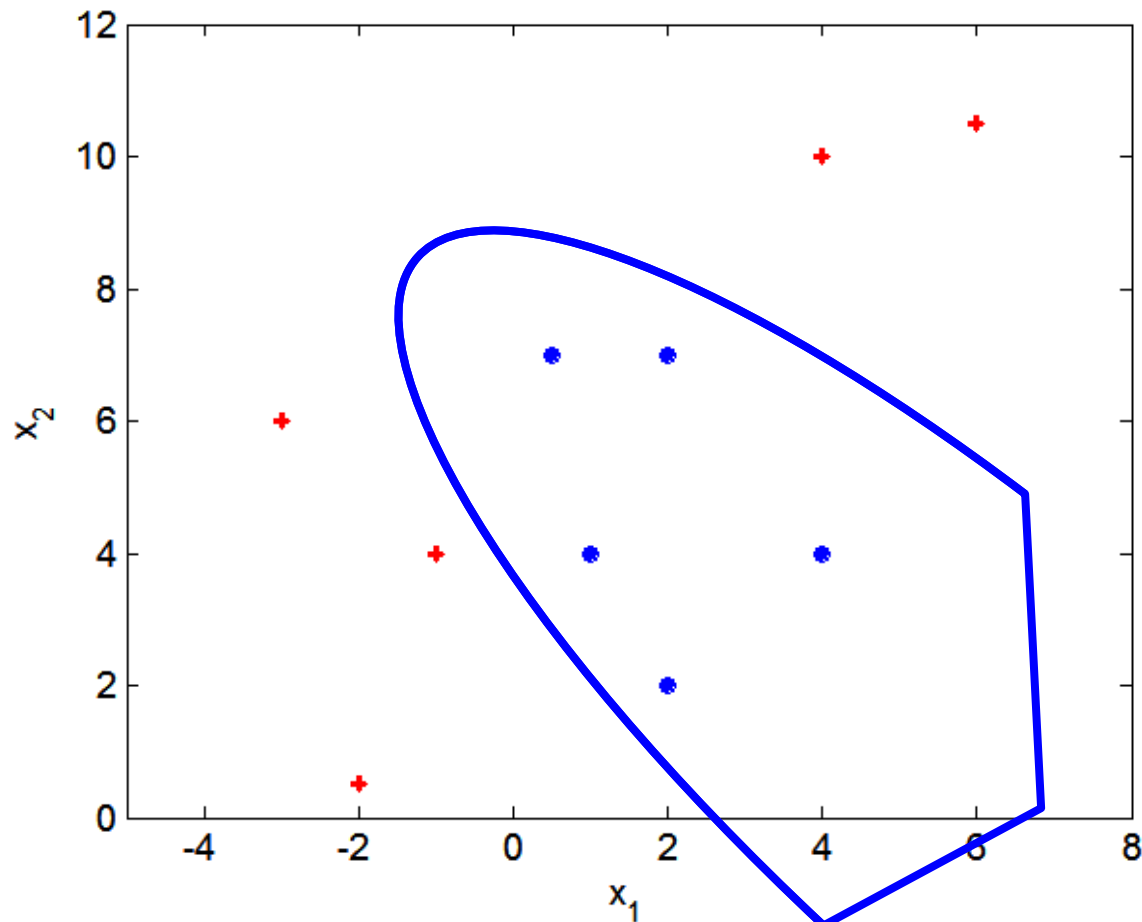
$$L(w) = \frac{\|w\|^2}{2} + C \sum_{i=1}^N \xi_i$$

- Subject to:



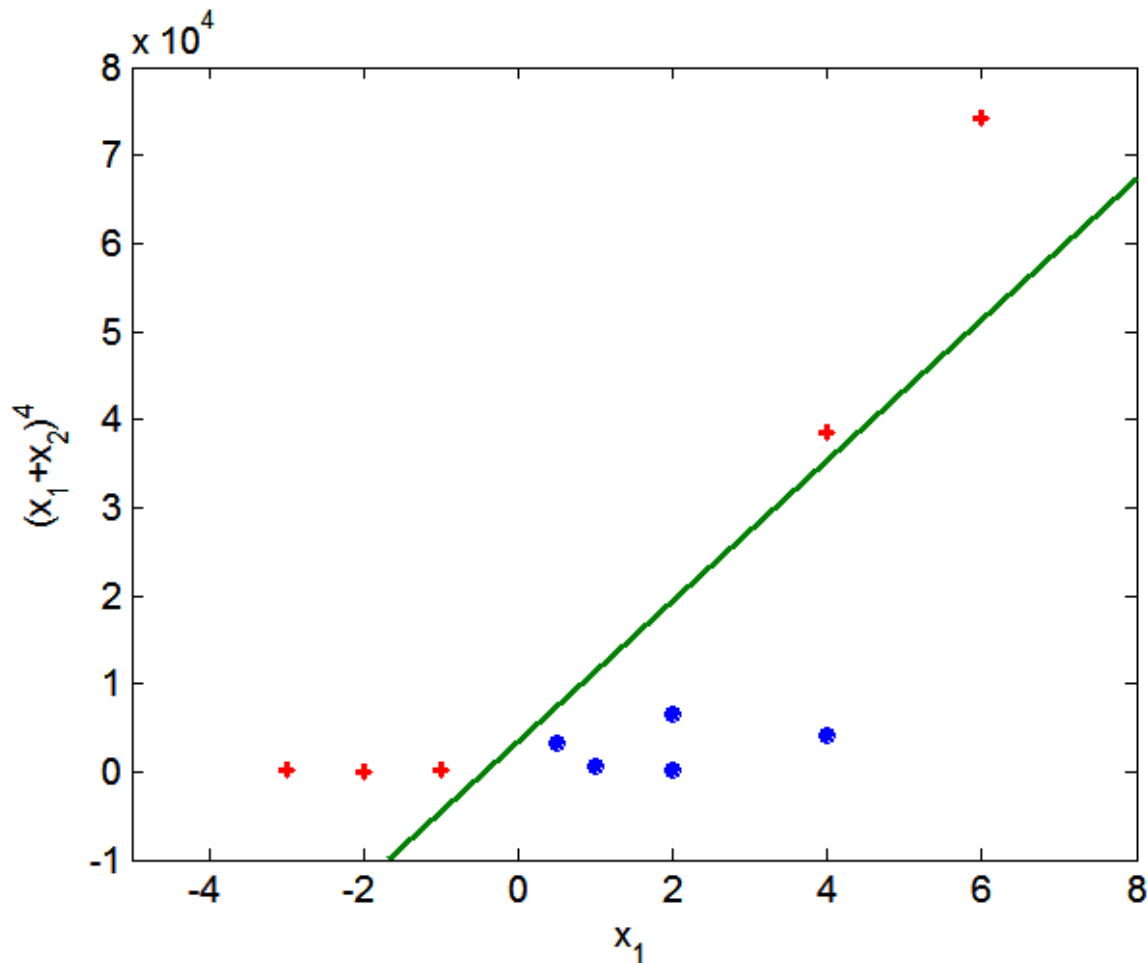
Nonlinear Support Vector Machines

- What if decision boundary is not linear?



Nonlinear Support Vector Machines

- Transform data into higher dimensional space

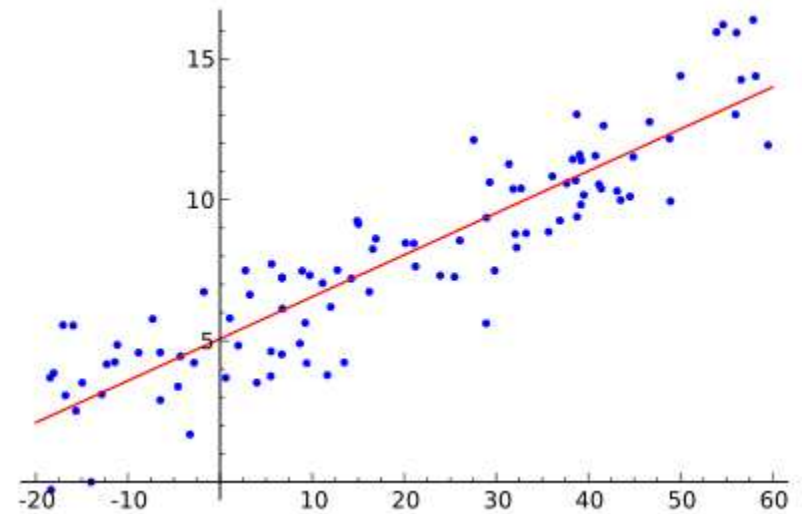


LOGISTIC REGRESSION

Classification via regression

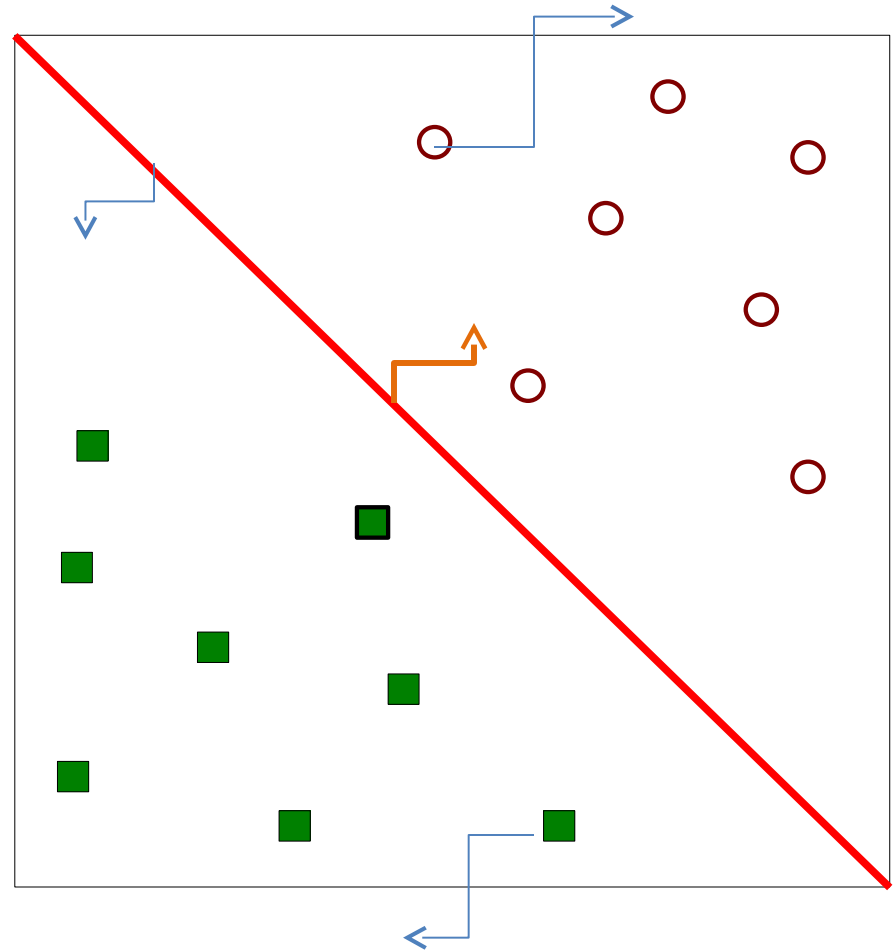
- Instead of predicting the class of an record we want to predict the probability of the class given the record
- The problem of predicting continuous values is called **regression** problem
- General approach: find a continuous function that models the continuous points.

Example: Linear regression



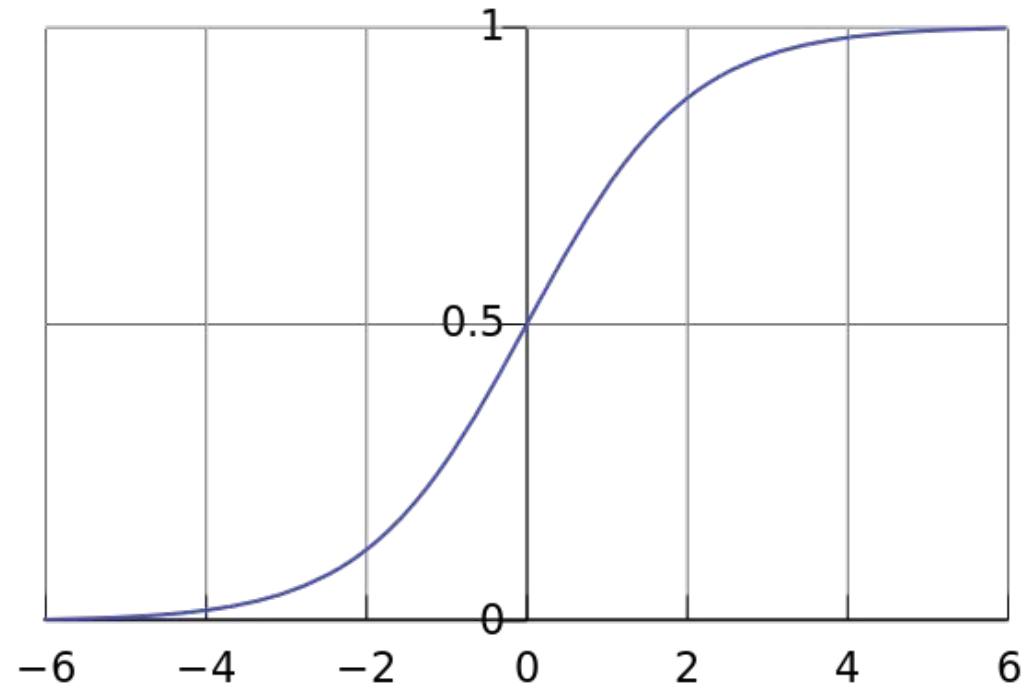
Classification via regression

- Assume a linear classification boundary



Logistic Regression

The **logistic function**



Logistic Regression

- Produces a probability estimate for the class membership which is often very useful.
- The weights can be useful for understanding the feature importance.
- Works for relatively large datasets
- Fast to apply.

NAÏVE BAYES CLASSIFIER

Bayes Classifier

- A probabilistic framework for solving classification problems
- **A, C** random variables
- Joint probability: **$\Pr(A=a, C=c)$**
- Conditional probability: **$\Pr(C=c | A=a)$**
- Relationship between joint and conditional probability distributions

$$P(C | A) = \frac{P(A | C)P(C)}{P(A)}$$

Example of Bayes Theorem

- Given:
 - A doctor knows that meningitis causes stiff neck 50% of the time
 - **Prior probability** of any patient having meningitis is 1/50,000
 - **Prior probability** of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M | S) = \frac{P(S | M)P(M)}{P(S)} = \frac{0.5 \cdot 1/50000}{1/20} = 0.0002$$

Bayesian Classifiers

- Consider each attribute and class label as random variables
- Given a record with attributes (A_1, A_2, \dots, A_n)
 - Goal is to predict class C
 - Specifically, we want to find the value of C that maximizes $P(C | A_1, A_2, \dots, A_n)$
- Can we estimate $P(C | A_1, A_2, \dots, A_n)$ directly from data?

Bayesian Classifiers

- Approach:
 - compute the posterior probability $P(C | A_1, A_2, \dots, A_n)$ for all values of C using the Bayes theorem

$$P(C | A_1 A_2 \dots A_n) = \frac{P(A_1 A_2 \dots A_n | C) P(C)}{P(A_1 A_2 \dots A_n)}$$

- Choose value of C that maximizes $P(C | A_1, A_2, \dots, A_n)$
- Equivalent to choosing value of C that maximizes $P(A_1, A_2, \dots, A_n | C) P(C)$

Naïve Bayes Classifier

How to Estimate Probabilities from Data?

categorical
categorical
continuous
class

<i>Tid</i>	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$

- e.g., $P(\text{No}) = 7/10$,
 $P(\text{Yes}) = 3/10$

- For discrete attributes:

$$P(A_i | C_k) = |A_{ik}| / N_c^k$$

- where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
- Examples:

$$P(\text{Status}=\text{Married} | \text{No}) = 4/7$$

How to Estimate Probabilities from Data?

- For continuous attributes:
 - **Discretize** the range into bins
 - one ordinal attribute per bin
 - violates independence assumption
 - **Two-way split:** $(A < v)$ or $(A > v)$
 - choose only one of the two splits as new attribute
 - **Probability density estimation:**
 - Assume attribute follows a normal distribution
 - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
 - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i | c_j) = \frac{1}{\sqrt{2\pi} \sigma_{ij}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

- One for each (Ai,ci) pair

- For (Income, Class=No):

- If Class=No

- sample mean = 110

- sample variance = 2975

$$P(\text{Income} = 120 | \text{No})$$

$$= \frac{1}{\sqrt{2\pi} (54.54)} e^{-\frac{(120 - 110)^2}{2 \cdot 2975}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

X (Refund No, Married, Income 120K)

naive Bayes Classifier:

$P(\text{Refund}=\text{Yes}|\text{No}) = 3/7$
 $P(\text{Refund}=\text{No}|\text{No}) = 4/7$
 $P(\text{Refund}=\text{Yes}|\text{Yes}) = 0$
 $P(\text{Refund}=\text{No}|\text{Yes}) = 1$
 $P(\text{Marital Status}=\text{Single}|\text{No}) = 2/7$
 $P(\text{Marital Status}=\text{Divorced}|\text{No}) = 1/7$
 $P(\text{Marital Status}=\text{Married}|\text{No}) = 4/7$
 $P(\text{Marital Status}=\text{Single}|\text{Yes}) = 2/7$
 $P(\text{Marital Status}=\text{Divorced}|\text{Yes}) = 1/7$
 $P(\text{Marital Status}=\text{Married}|\text{Yes}) = 0$

For taxable income:

If class=No: sample mean=110
 sample variance=2975
 If class=Yes: sample mean=90
 sample variance=25

- $P(X|\text{Class}=\text{No}) = P(\text{Refund}=\text{No}|\text{Class}=\text{No})$
 $P(\text{Married}|\text{Class}=\text{No})$
 $P(\text{Income}=120\text{K}|\text{Class}=\text{No})$
 $= 4/7 \cdot 4/7 \cdot 0.0072 = 0.0024$
- $P(X|\text{Class}=\text{Yes}) = P(\text{Refund}=\text{No}|\text{Class}=\text{Yes})$
 $P(\text{Married}|\text{Class}=\text{Yes})$
 $P(\text{Income}=120\text{K}|\text{Class}=\text{Yes})$
 $= 1 \cdot 0 \cdot 1.2 \cdot 10^{-9} = 0$

Since $P(X|\text{No})P(\text{No}) > P(X|\text{Yes})P(\text{Yes})$

Therefore $P(\text{No}|X) > P(\text{Yes}|X)$

$\Rightarrow \text{Class} = \text{No}$

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Probability estimation:

$$\text{Original: } P(A_i | C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace: } P(A_i | C) = \frac{N_{ic} + 1}{N_c + N_i}$$

$$\text{m - estimate: } P(A_i | C) = \frac{N_{ic} + mp}{N_c + m}$$

N_i : number of attribute values for attribute A_i

p : prior probability

m : parameter

Example of Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \frac{6}{7} \frac{2}{7} \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \frac{10}{13} \frac{3}{13} \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \frac{13}{20} = 0.0027$$

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

$$P(A|M)P(M) > P(A|N)P(N)$$

=> Mammals

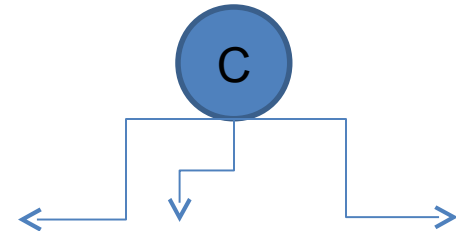
Implementation details

Naïve Bayes (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
 - Use other techniques such as Bayesian Belief Networks (BBN)

Generative vs Discriminative models

- Naïve Bayes is a type of a **generative model**
 - Generative process:
 - First pick the category of the record
 - Then given the category, generate the attribute values from the distribution of the category
 - Conditional independence given C



Generative vs Discriminative models

- Logistic Regression and SVM are **discriminative models**
 - The goal is to find the boundary that discriminates between the two classes from the training data
- In order to classify the language of a document, you can
 - Either learn the two languages and find which is more likely to have generated the words you see
 - Or learn what differentiates the two languages.



An Introduction to Support Vector Machines



Outline

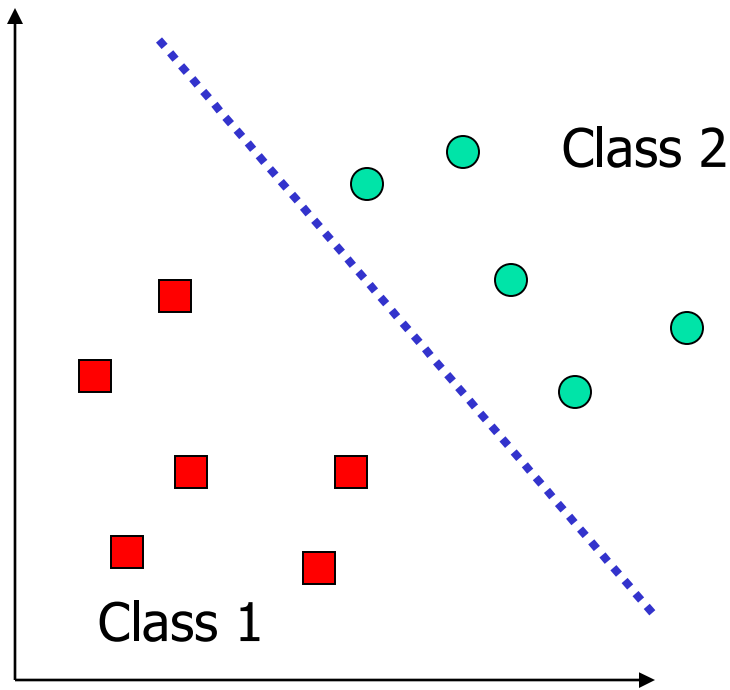
- History of support vector machines (SVM)
- Two classes, linearly separable
 - What is a good decision boundary?
- Two classes, not linearly separable
- How to make SVM non-linear: kernel trick
- Demo of SVM
- Epsilon support vector regression (ε -SVR)
- Conclusion



History of SVM

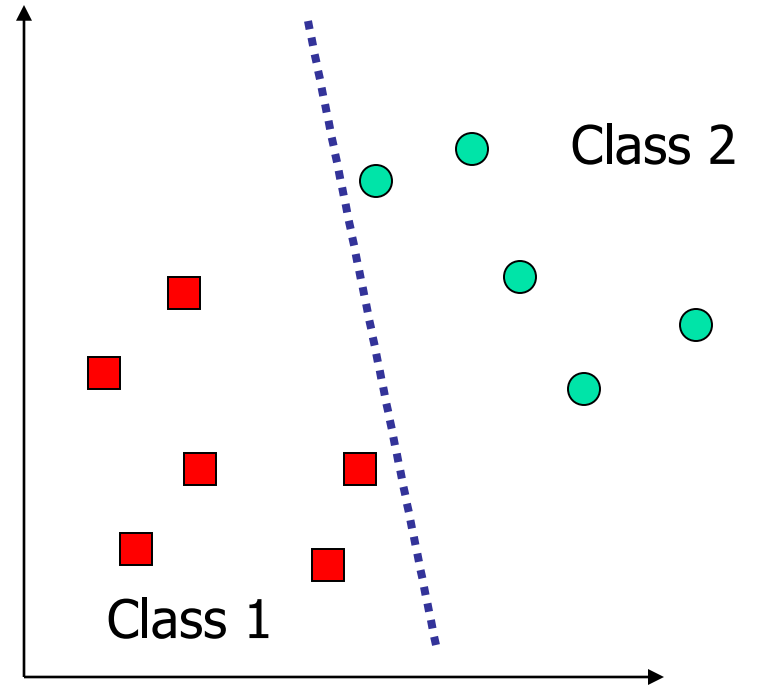
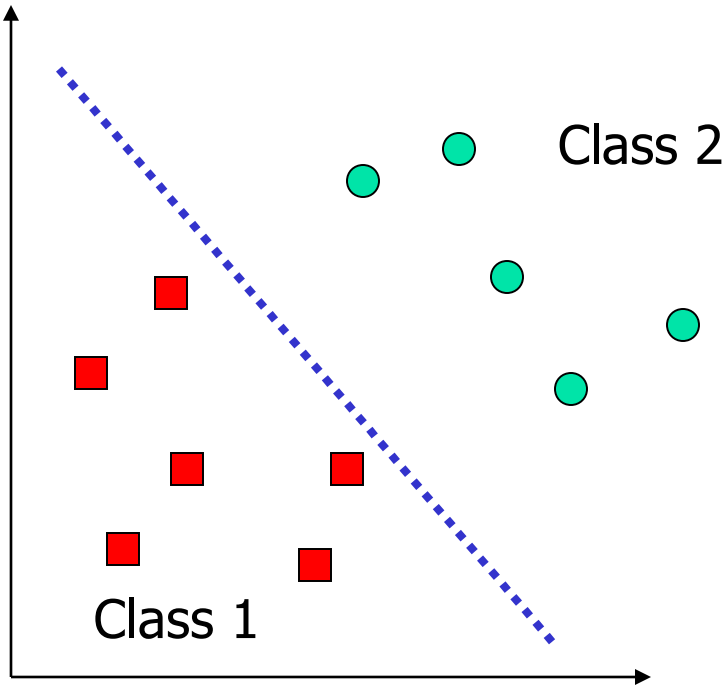
- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVM was first introduced in COLT-92
- SVM becomes famous when, using pixel maps as input, it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task
- Currently, SVM is closely related to:
 - Kernel methods, large margin classifiers, reproducing kernel Hilbert space, Gaussian process

Two Class Problem: Linear Separable Case



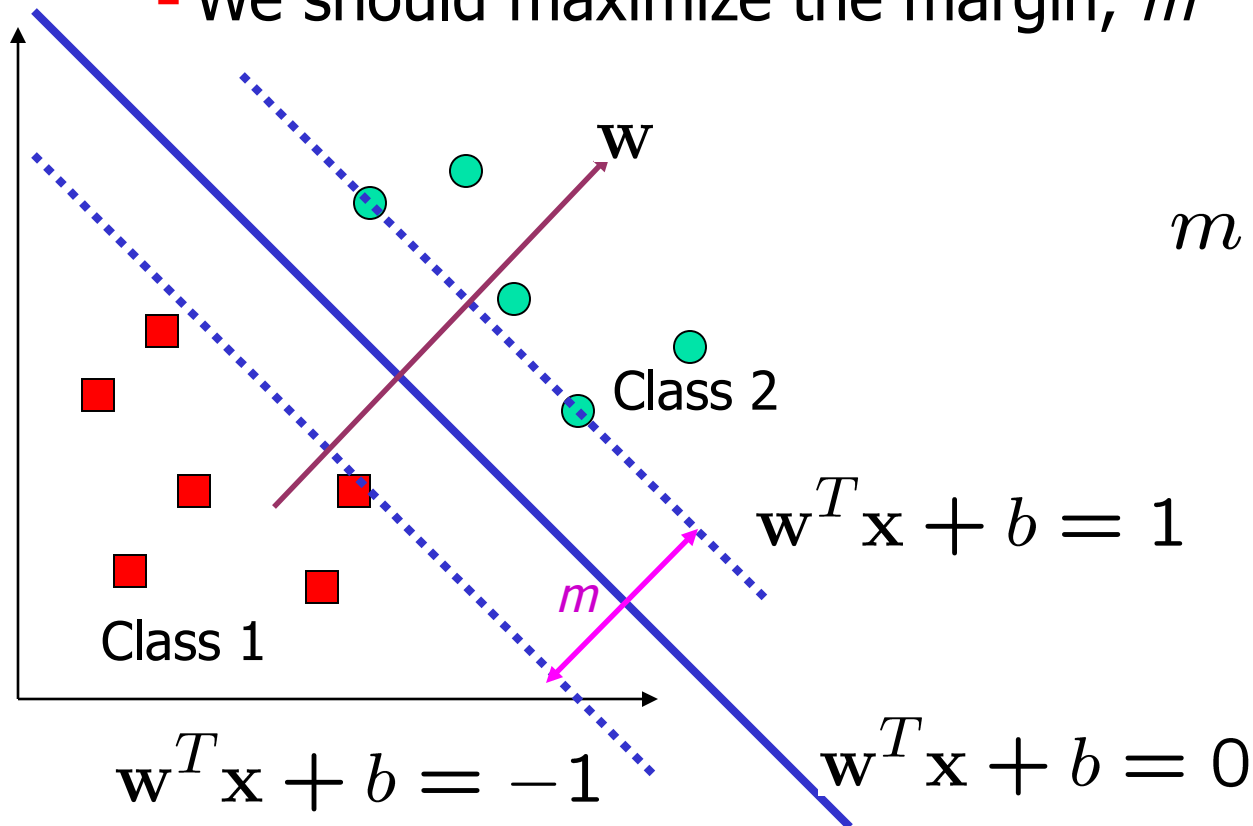
- Many decision boundaries can separate these two classes
- Which one should we choose?

Example of Bad Decision Boundaries



Good Decision Boundary: Margin Should Be Large

- The decision boundary should be as far away from the data of both classes as possible
 - We should maximize the margin, m



$$m = \frac{2}{\|w\|}$$

The Optimization Problem

- Let $\{x_1, \dots, x_n\}$ be our data set and let $y_i \in \{1, -1\}$ be the class label of x_i
- The decision boundary should classify all points correctly $\Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i$
- A constrained optimization problem

$$\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2$$

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad \forall i$$

The Optimization Problem

- We can transform the problem to its dual

$$\max. W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

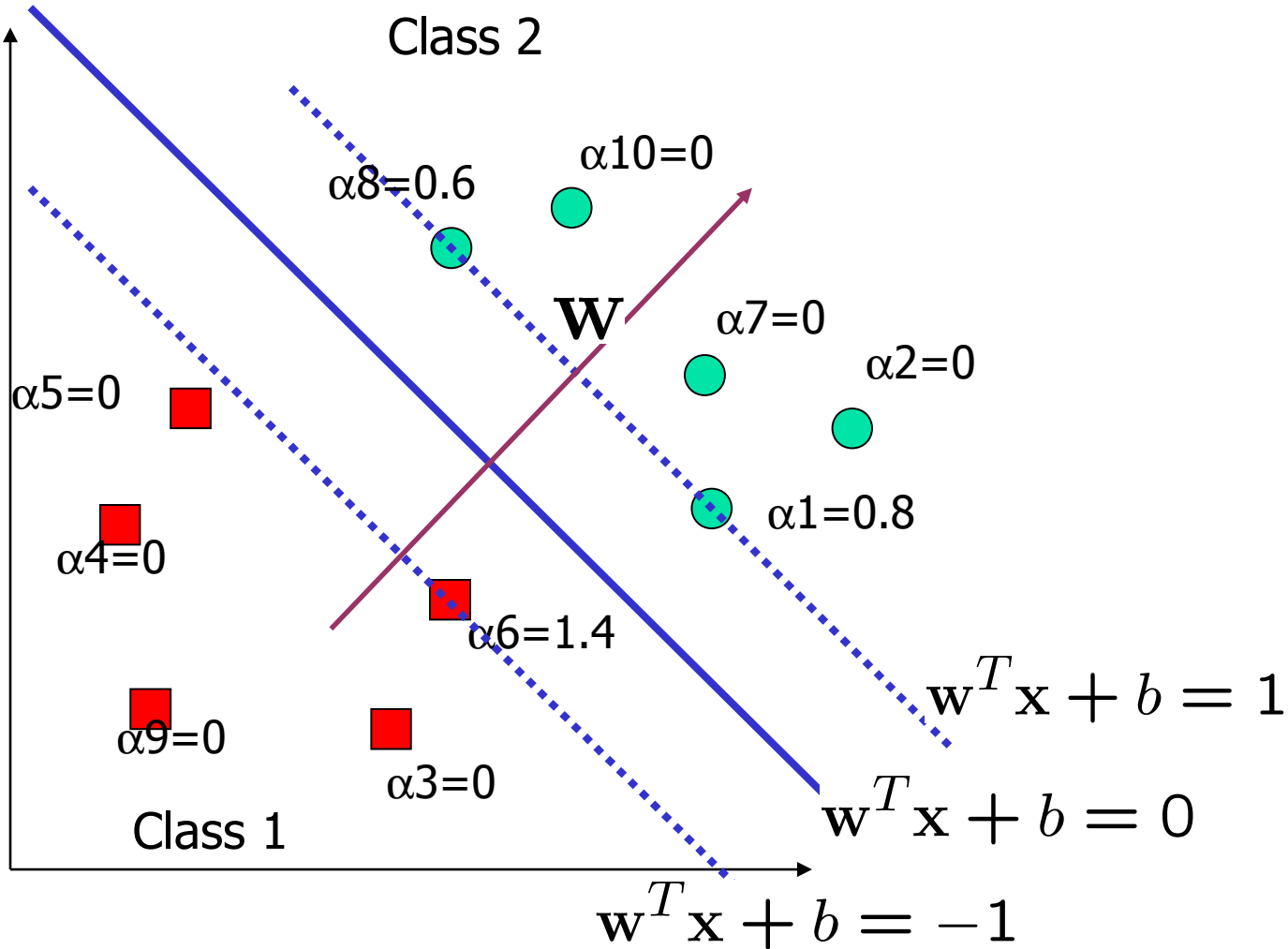
- This is a quadratic programming (QP) problem
 - Global maximum of α_i can always be found

- \mathbf{w} can be recovered by $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$

Characteristics of the Solution

- Many of the α_i are zero
 - \mathbf{w} is a linear combination of a small number of data
 - Sparse representation
- \mathbf{x}_i with non-zero α_i are called support vectors (SV)
 - The decision boundary is determined only by the SV
 - Let t_j ($j=1, \dots, s$) be the indices of the s support vectors. We can write $\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- For testing with a new data \mathbf{z}
 - Compute $\mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{z}) + b$ and classify \mathbf{z} as class 1 if the sum is positive, and class 2 otherwise

A Geometrical Interpretation



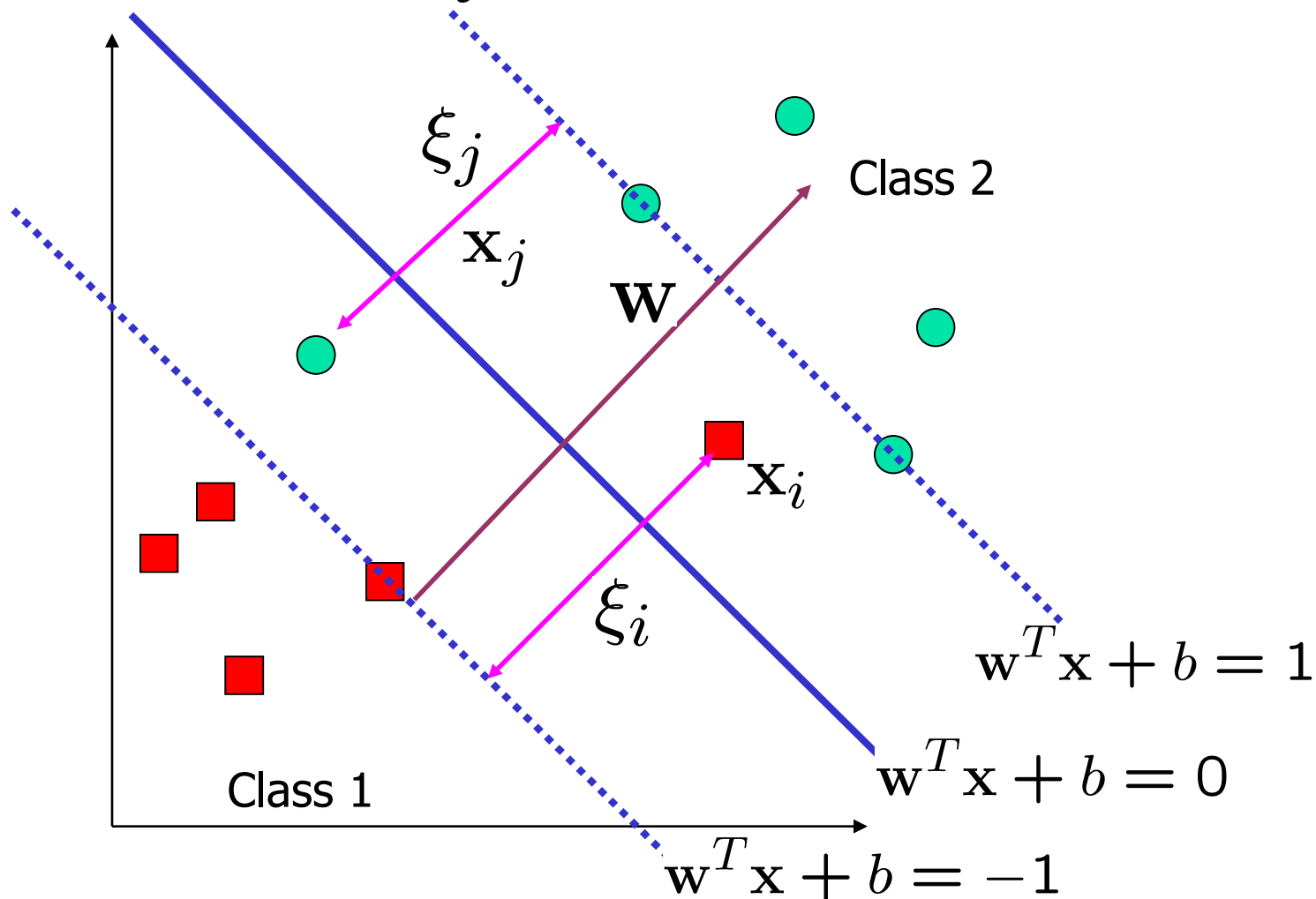


Some Notes

- There are theoretical upper bounds on the error on unseen data for SVM
 - The larger the margin, the smaller the bound
 - The smaller the number of SV, the smaller the bound
- Note that in both training and testing, the data are referenced only as inner product, $\mathbf{x}^T \mathbf{y}$
 - This is important for generalizing to the non-linear case

How About Not Linearly Separable

- We allow "error" ξ_i in classification



Soft Margin Hyperplane

- Define $\xi_i=0$ if there is no error for x_i
 - ξ_i are just "slack variables" in optimization theory

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 - \xi_i & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 + \xi_i & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

- We want to minimize $\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$
 - C : tradeoff parameter between error and margin

- The optimization problem becomes

$$\text{Minimize } \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i$$

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

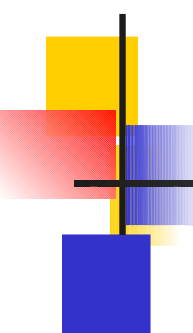
The Optimization Problem

- The dual of the problem is

$$\max. W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

- \mathbf{w} is also recovered as $\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$
- The only difference with the linear separable case is that there is an upper bound C on α_i
- Once again, a QP solver can be used to find α_i

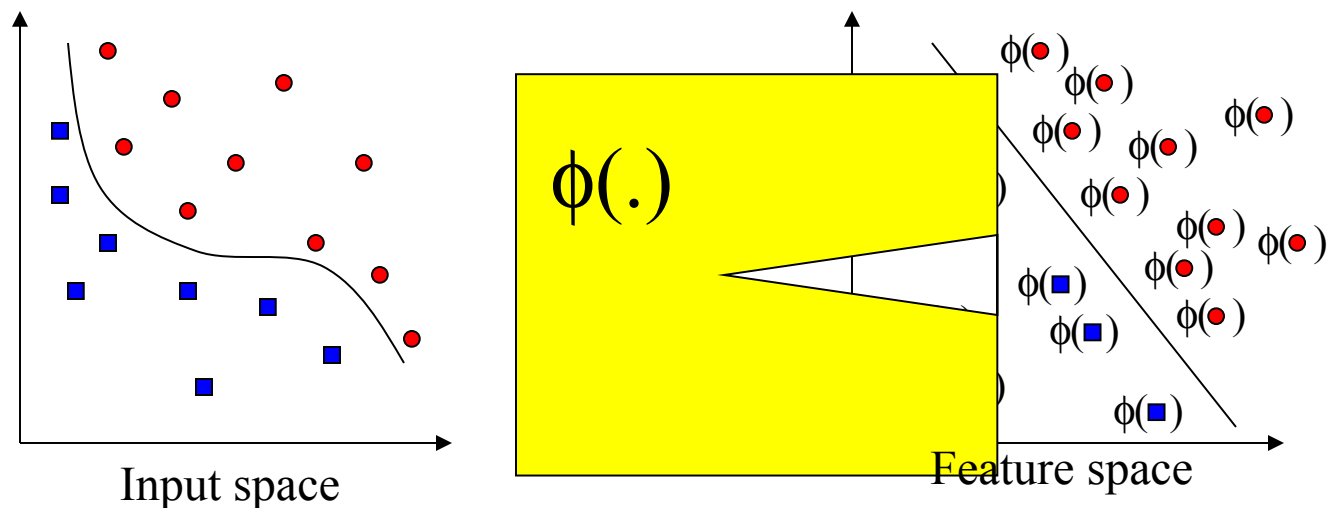


Extension to Non-linear Decision Boundary

- Key idea: transform \mathbf{x}_i to a higher dimensional space to “make life easier”
 - Input space: the space \mathbf{x}_i are in
 - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
 - Linear operation in the feature space is equivalent to non-linear operation in input space
 - The classification task can be “easier” with a proper transformation. Example: XOR

Extension to Non-linear Decision Boundary

- Possible problem of the transformation
 - High computation burden and hard to get a good estimate
- SVM solves these two issues simultaneously
 - Kernel tricks for efficient computation
 - Minimize $\|\mathbf{w}\|^2$ can lead to a “good” classifier



Example Transformation

- Define the kernel function $K(\mathbf{x}, \mathbf{y})$ as

$$K(\mathbf{x}, \mathbf{y}) = (1 + x_1y_1 + x_2y_2)^2$$

- Consider the following transformation

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

$$\phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) = (1, \sqrt{2}y_1, \sqrt{2}y_2, y_1^2, y_2^2, \sqrt{2}y_1y_2)$$

$$\begin{aligned} \langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}\right) \rangle &= (1 + x_1y_1 + x_2y_2)^2 \\ &= K(\mathbf{x}, \mathbf{y}) \end{aligned}$$

- The inner product can be computed by K without going through the map $\phi(\cdot)$



Kernel Trick

- The relationship between the kernel function K and the mapping $\phi(\cdot)$ is

$$K(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$$

- This is known as the kernel trick
- In practice, we specify K , thereby specifying $\phi(\cdot)$ indirectly, instead of choosing $\phi(\cdot)$
- Intuitively, $K(\mathbf{x}, \mathbf{y})$ represents our desired notion of similarity between data \mathbf{x} and \mathbf{y} and this is from our prior knowledge
- $K(\mathbf{x}, \mathbf{y})$ needs to satisfy a technical condition (Mercer condition) in order for $\phi(\cdot)$ to exist

Examples of Kernel Functions

- Polynomial kernel with degree d

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\sigma^2))$$

- Closely related to radial basis function neural networks

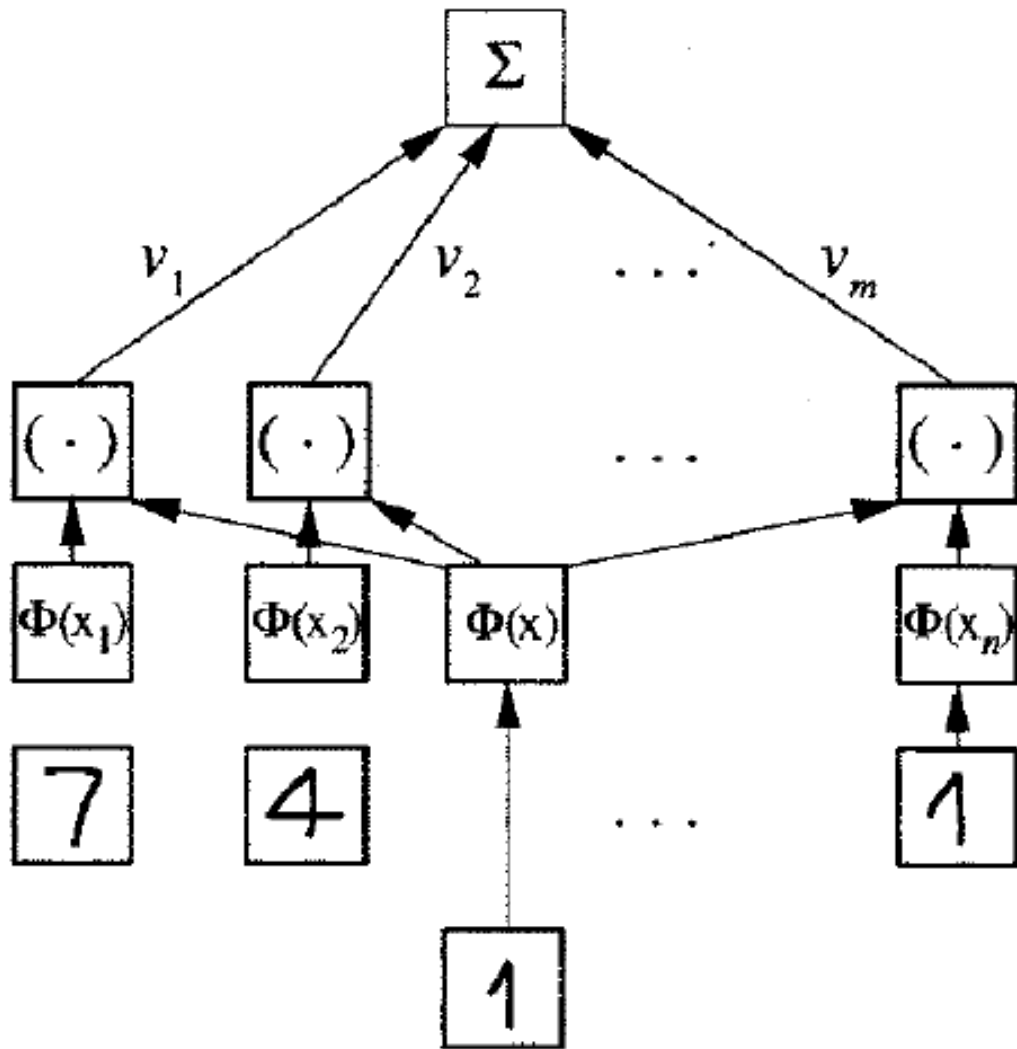
- Sigmoid with parameter κ and θ

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\kappa \mathbf{x}^T \mathbf{y} + \theta)$$

- It does not satisfy the Mercer condition on all κ and θ

- Research on different kernel functions in different applications is very active

Example of SVM Applications: Handwriting Recognition



output $\Sigma v_i k(x, x_i) + b$

weights

dot product $(\Phi(x) \cdot \Phi(x_i)) = k(x, x_i)$

mapped vectors $\Phi(x_i), \Phi(x)$

support vectors $x_1 \dots x_n$

test vector x

Modification Due to Kernel Function

- Change all inner products to kernel functions
- For training,

Original

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

With kernel function

$$\max. W(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to } C \geq \alpha_i \geq 0, \sum_{i=1}^n \alpha_i y_i = 0$$

Modification Due to Kernel Function

- For testing, the new data \mathbf{z} is classified as class 1 if $f \geq 0$, and as class 2 if $f < 0$

Original

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

$$f = \mathbf{w}^T \mathbf{z} + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}^T \mathbf{z} + b$$

With kernel function

$$\mathbf{w} = \sum_{j=1}^s \alpha_{t_j} y_{t_j} \phi(\mathbf{x}_{t_j})$$

$$f = \langle \mathbf{w}, \phi(\mathbf{z}) \rangle + b = \sum_{j=1}^s \alpha_{t_j} y_{t_j} K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$

Example

- Suppose we have 5 1D data points
 - $x_1=1, x_2=2, x_3=4, x_4=5, x_5=6$, with 1, 2, 6 as class 1 and 4, 5 as class 2
 - $y_1=1, y_2=1, y_3=-1, y_4=-1, y_5=1$
- We use the polynomial kernel of degree 2
 - $K(x,y) = (xy+1)^2$
 - C is set to 100

- We first find α_i ($i=1, \dots, 5$) by

$$\max. \sum_{i=1}^5 \alpha_i - \frac{1}{2} \sum_{i=1}^5 \sum_{j=1}^5 \alpha_i \alpha_j y_i y_j (x_i x_j + 1)^2$$

$$\text{subject to } 100 \geq \alpha_i \geq 0, \sum_{i=1}^5 \alpha_i y_i = 0$$

Example

- By using a QP solver, we get
 - $\alpha_1=0, \alpha_2=2.5, \alpha_3=0, \alpha_4=7.333, \alpha_5=4.833$
 - Note that the constraints are indeed satisfied
 - The support vectors are $\{x_2=2, x_4=5, x_5=6\}$
- The discriminant function is

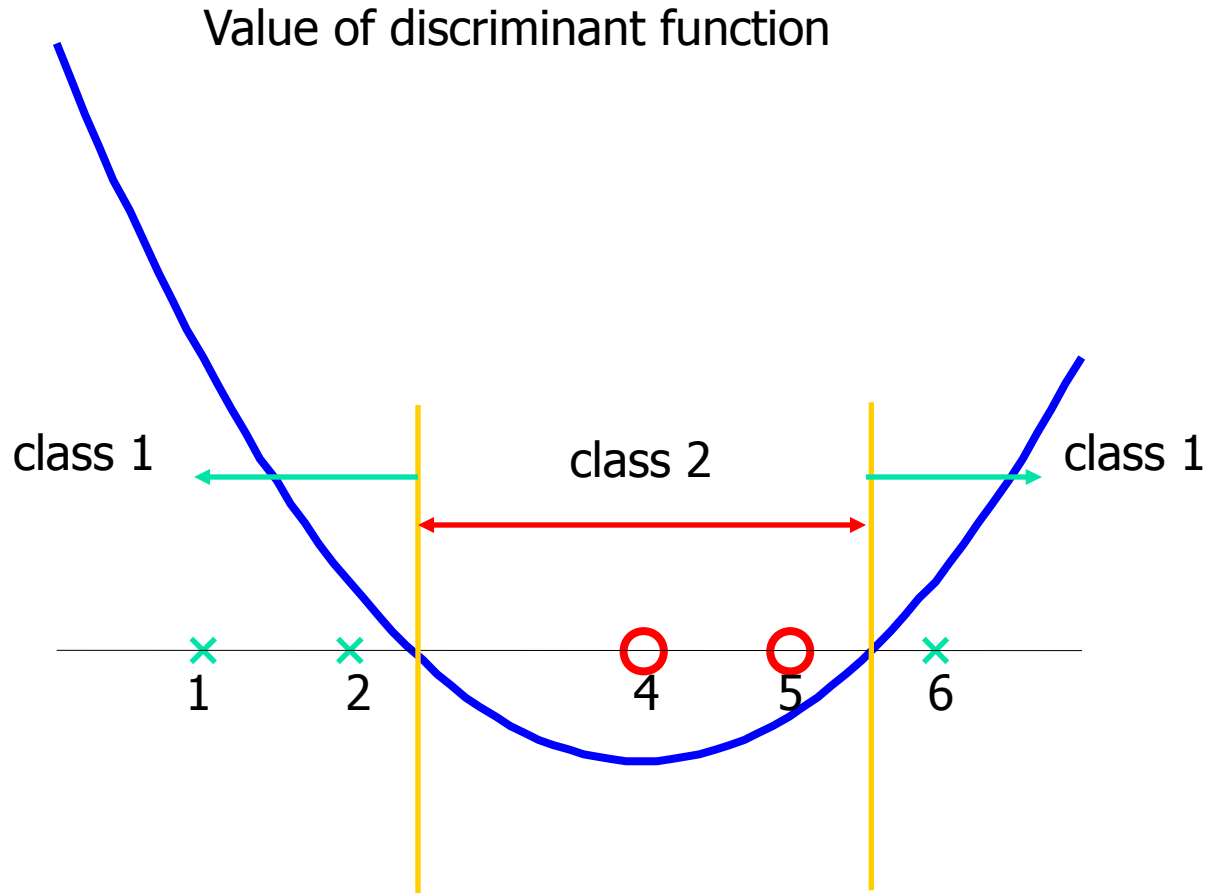
$$\begin{aligned} f(y) &= 2.5(1)(2y + 1)^2 + 7.333(-1)(5y + 1)^2 + 4.833(1)(6y + 1)^2 + b \\ &= 0.6667x^2 - 5.333x + b \end{aligned}$$

- b is recovered by solving $f(2)=1$ or by $f(5)=-1$ or by $f(6)=1$, as x_2, x_4, x_5 lie on $y_i(w^T \phi(z) + b) = 1$

and all give $b=9$


$$f(y) = 0.6667x^2 - 5.333x + 9$$

Example





Multi-class Classification

- SVM is basically a two-class classifier
- One can change the QP formulation to allow multi-class classification
- More commonly, the data set is divided into two parts “intelligently” in different ways and a separate SVM is trained for each way of division
- Multi-class classification is done by combining the output of all the SVM classifiers
 - Majority rule
 - Error correcting code
 - Directed acyclic graph



Software

- A list of SVM implementation can be found at <http://www.kernel-machines.org/software.html>
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM
- Several Matlab toolboxes for SVM are also available



Summary: Steps for Classification

- Prepare the pattern matrix
- Select the kernel function to use
- Select the parameter of the kernel function and the value of C
 - You can use the values suggested by the SVM software, or you can set apart a validation set to determine the values of the parameter
- Execute the training algorithm and obtain the α_i
- Unseen data can be classified using the α_i and the support vectors



Demonstration

- Iris data set

- Class 1 and class 3 are “merged” in this demo



Strengths and Weaknesses of SVM

■ Strengths

- Training is relatively easy
 - No local optimal, unlike in neural networks
- It scales relatively well to high dimensional data
- Tradeoff between classifier complexity and error can be controlled explicitly
- Non-traditional data like strings and trees can be used as input to SVM, instead of feature vectors

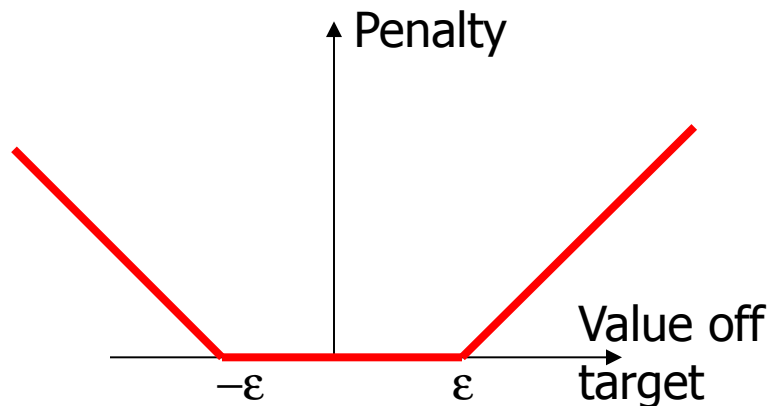
■ Weaknesses

- Need a “good” kernel function

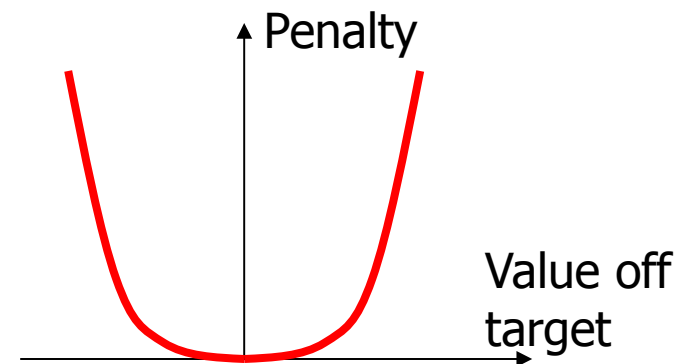
Epsilon Support Vector Regression (ϵ -SVR)

- Linear regression in feature space
- Unlike in least square regression, the error function is ϵ -insensitive loss function
 - Intuitively, mistake less than ϵ is ignored
 - This leads to sparsity similar to SVM

ϵ -insensitive loss function



Square loss function



Epsilon Support Vector Regression (ϵ -SVR)

- Given: a data set $\{x_1, \dots, x_n\}$ with target values $\{u_1, \dots, u_n\}$, we want to do ϵ -SVR
- The optimization problem is

$$\text{Min } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n (\xi_i + \xi_i^*)$$

$$\text{subject to } \begin{cases} u_i - w^T x_i - b \leq \epsilon + \xi_i \\ w^T x_i + b - u_i \leq \epsilon + \xi_i^* \\ \xi_i \geq 0, \xi_i^* \geq 0 \end{cases}$$

- Similar to SVM, this can be solved as a quadratic programming problem

Epsilon Support Vector Regression (ε -SVR)

- C is a parameter to control the amount of influence of the error
- The $\frac{1}{2}||w||^2$ term serves as controlling the complexity of the regression function
 - This is similar to ridge regression
- After training (solving the QP), we get values of α_i and α_i^* , which are both zero if \mathbf{x}_i does not contribute to the error function
- For a new data \mathbf{z} ,

$$f(\mathbf{z}) = \sum_{j=1}^s (\alpha_{t_j} - \alpha_{t_j}^*) K(\mathbf{x}_{t_j}, \mathbf{z}) + b$$



Other Types of Kernel Methods

- A lesson learnt in SVM: a linear algorithm in the feature space is equivalent to a non-linear algorithm in the input space
- Classic linear algorithms can be generalized to its non-linear version by going to the feature space
 - Kernel principal component analysis, kernel independent component analysis, kernel canonical correlation analysis, kernel k-means, 1-class SVM are some examples



Conclusion

- SVM is a useful alternative to neural networks
- Two key concepts of SVM: maximize the margin and the kernel trick
- Many active research is taking place on areas related to SVM
- Many SVM implementations are available on the web for you to try on your data set!



Resources

- <http://www.kernel-machines.org/>
- <http://www.support-vector.net/>
- <http://www.support-vector.net/icml-tutorial.pdf>
- <http://www.kernel-machines.org/papers/tutorial-ni>
- <http://www.clopinet.com/isabelle/Projects/SVM/ap>

K-MEANS CLUSTERING

What is clustering?

- **Clustering** is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

Types of clustering:

1. **Hierarchical algorithms**: these find successive clusters using previously established clusters.
 1. Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.
 2. Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
2. **Partitional clustering**: Partitional algorithms determine all clusters at once. They include:
 - **K-means and derivatives**
 - Fuzzy c-means clustering
 - QT clustering algorithm

Common Distance measures:

- *Distance measure* will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

They include:

1. The Euclidean distance (also called 2-norm distance) is given by:

$$d(x, y) = \sum_{i=1}^p |x_i - y_i|$$

2. The Manhattan distance (also called taxicab norm or 1-norm) is given by:

$$d(x, y) = \sqrt[2]{\sum_{i=1}^p |x_i - y_i|^2}$$

3. The maximum norm is given by:

$$d(x, y) = \max_{1 \leq i \leq p} |x_i - y_i|$$

4. The Mahalanobis distance corrects data for different scales and correlations in the variables.

5. Inner product space: The angle between two vectors can be used as a distance measure when clustering high dimensional data

6. Hamming distance (sometimes edit distance) measures the minimum number of substitutions required to change one member into another.

K-MEANS CLUSTERING

- The **k-means algorithm** is an algorithm to cluster n objects based on attributes into k partitions, where $k < n$.
- It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.
- It assumes that the object attributes form a vector space.

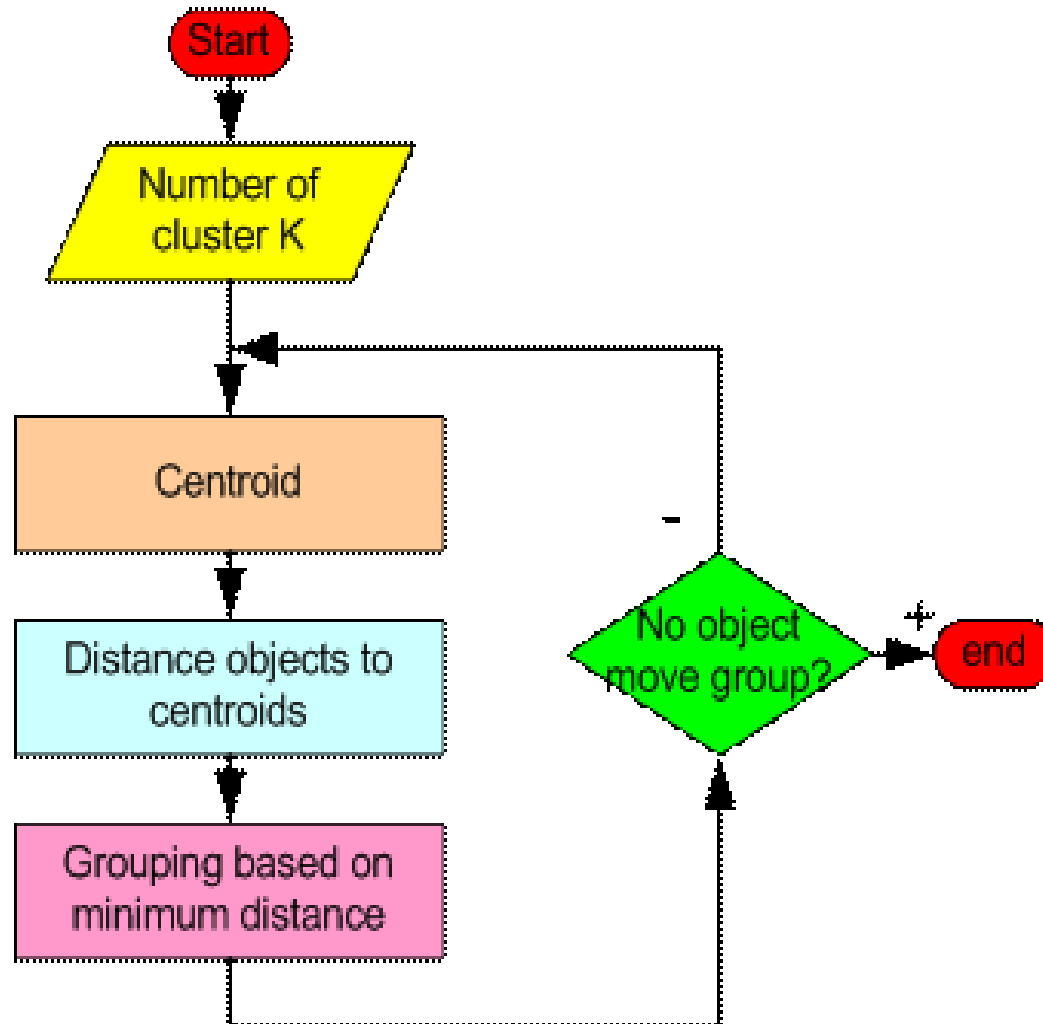
- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^K \sum_{n \in S_j} |x_n - \mu_j|^2,$$

where x_n is a vector representing the the n th data point and μ_j is the geometric centroid of the data points in S_j .

- Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group.
- K is positive integer number.
- The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

How the K-Mean Clustering algorithm works?



- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2:** Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as the following:
 1. Take the first k training sample as single-element clusters
 2. Assign each of the remaining $(N-k)$ training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

A Simple example showing the implementation of k-means algorithm

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 1:

Initialization: Randomly we choose following two centroids (k=2) for two clusters.

In this case the 2 centroid are: $m_1=(1.0,1.0)$ and $m_2=(5.0,7.0)$.

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 2:

- Thus, we obtain two clusters containing:
 {1,2,3} and {4,5,6,7}.
- Their new centroids are:

$$m_1 = \left(\frac{1}{3}(1.0 + 1.5 + 3.0), \frac{1}{3}(1.0 + 2.0 + 4.0) \right) = (1.83, 2.33)$$

$$m_2 = \left(\frac{1}{4}(5.0 + 3.5 + 4.5 + 3.5), \frac{1}{4}(7.0 + 5.0 + 5.0 + 4.5) \right) \\ = (4.12, 5.38)$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$d(m_1, 2) = \sqrt{|1.0 - 1.5|^2 + |1.0 - 2.0|^2} = 1.12$$

$$d(m_2, 2) = \sqrt{|5.0 - 1.5|^2 + |7.0 - 2.0|^2} = 6.10$$

Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.
- Therefore, the new clusters are:
 $\{1,2\}$ and $\{3,4,5,6,7\}$
- Next centroids are:
 $m_1=(1.25,1.5)$ and $m_2 = (3.9,5.1)$

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.84	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

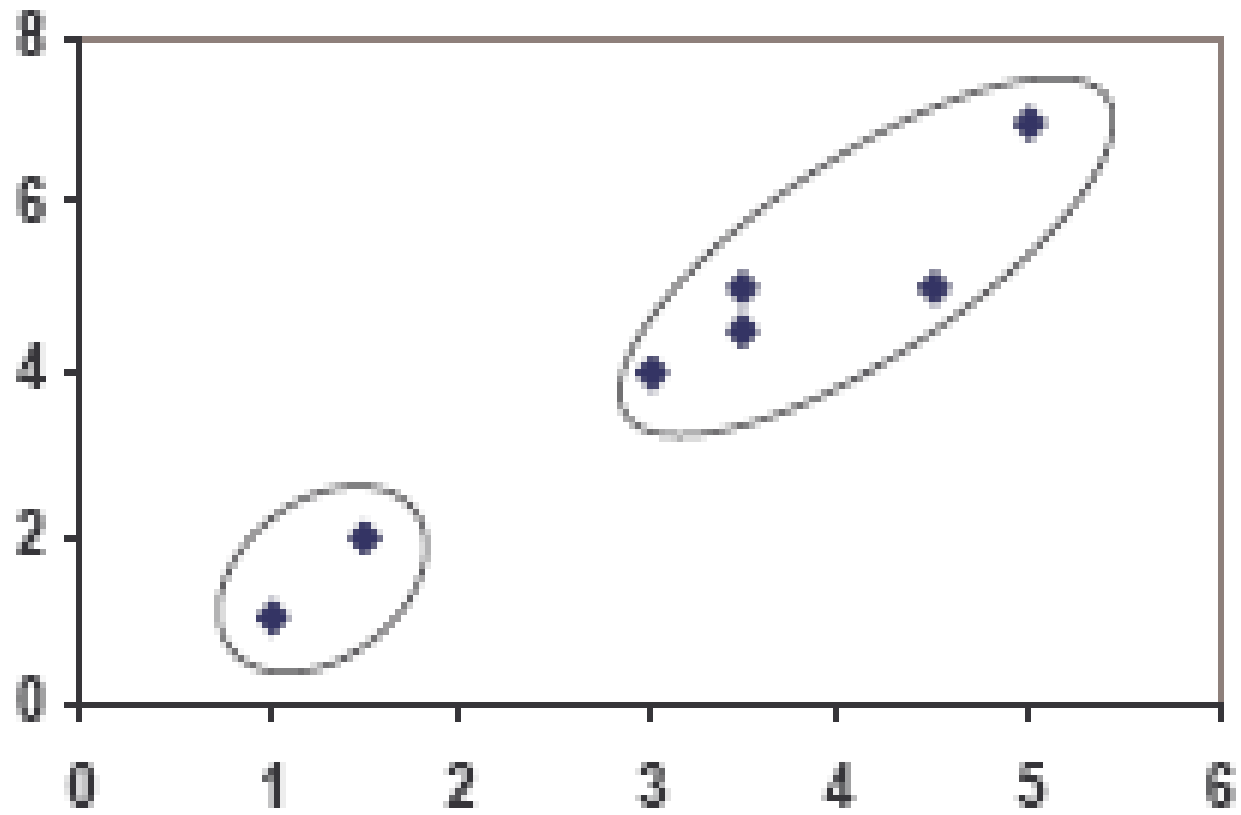
- Step 4 :

The clusters obtained are:
 $\{1,2\}$ and $\{3,4,5,6,7\}$

- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters $\{1,2\}$ and $\{3,4,5,6,7\}$.

Individual	Centroid 1	Centroid 2
1	0.58	5.02
2	0.58	3.92
3	3.05	1.42
4	6.88	2.20
5	4.18	0.41
6	4.78	0.81
7	3.75	0.72

PLOT



(with $K=3$)

Individual	$m_1 = 1$	$m_2 = 2$	$m_3 = 3$	cluster
1	0	1.11	3.61	1
2	1.12	0	2.5	2
3	3.61	2.5	0	3
4	7.21	6.10	3.61	3
5	4.72	3.61	1.12	3
6	5.31	4.24	1.80	3
7	4.30	3.20	0.71	3

} C_3

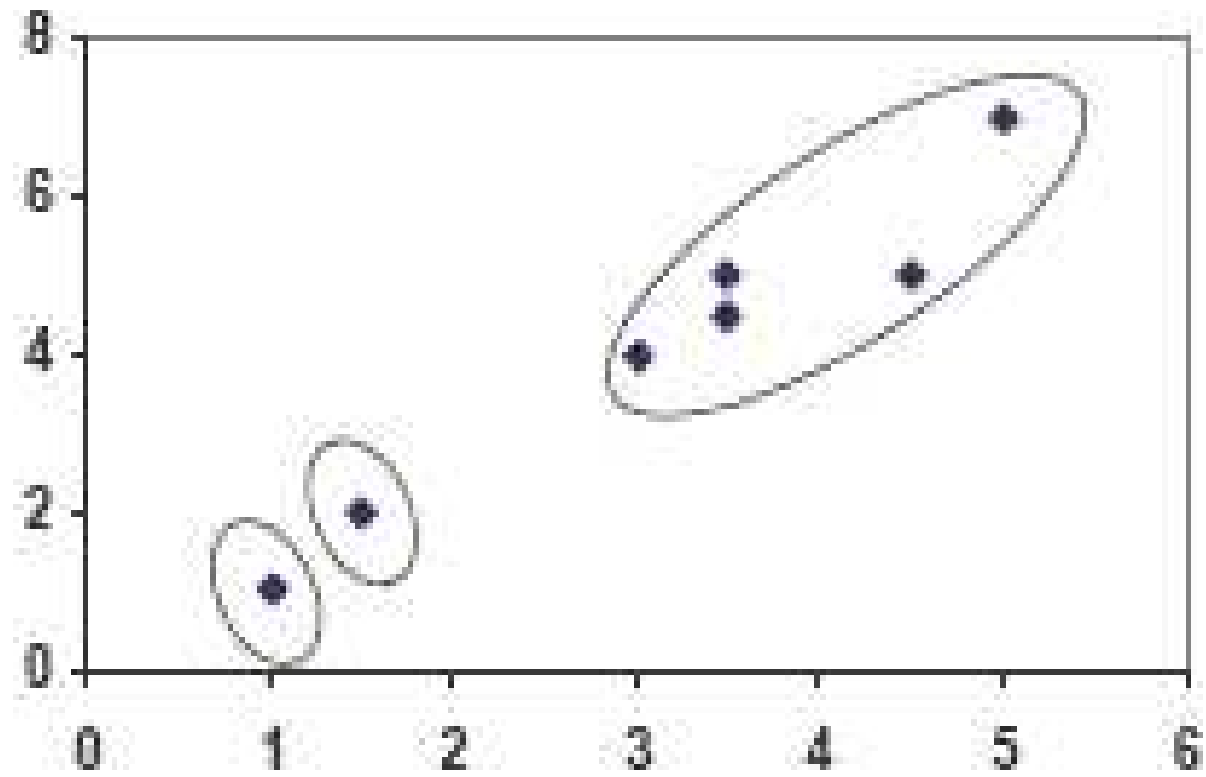
clustering with initial centroids (1, 2, 3)

Step 1

Individual	m_1 (1.0, 1.0)	m_2 (1.5, 2.0)	m_3 (3.9, 5.1)	cluster
1	0	1.11	5.02	1
2	1.12	0	3.92	2
3	3.61	2.5	1.42	3
4	7.21	6.10	2.20	3
5	4.72	3.61	0.41	3
6	5.31	4.24	0.61	3
7	4.30	3.20	0.72	3

Step 2

PLOT



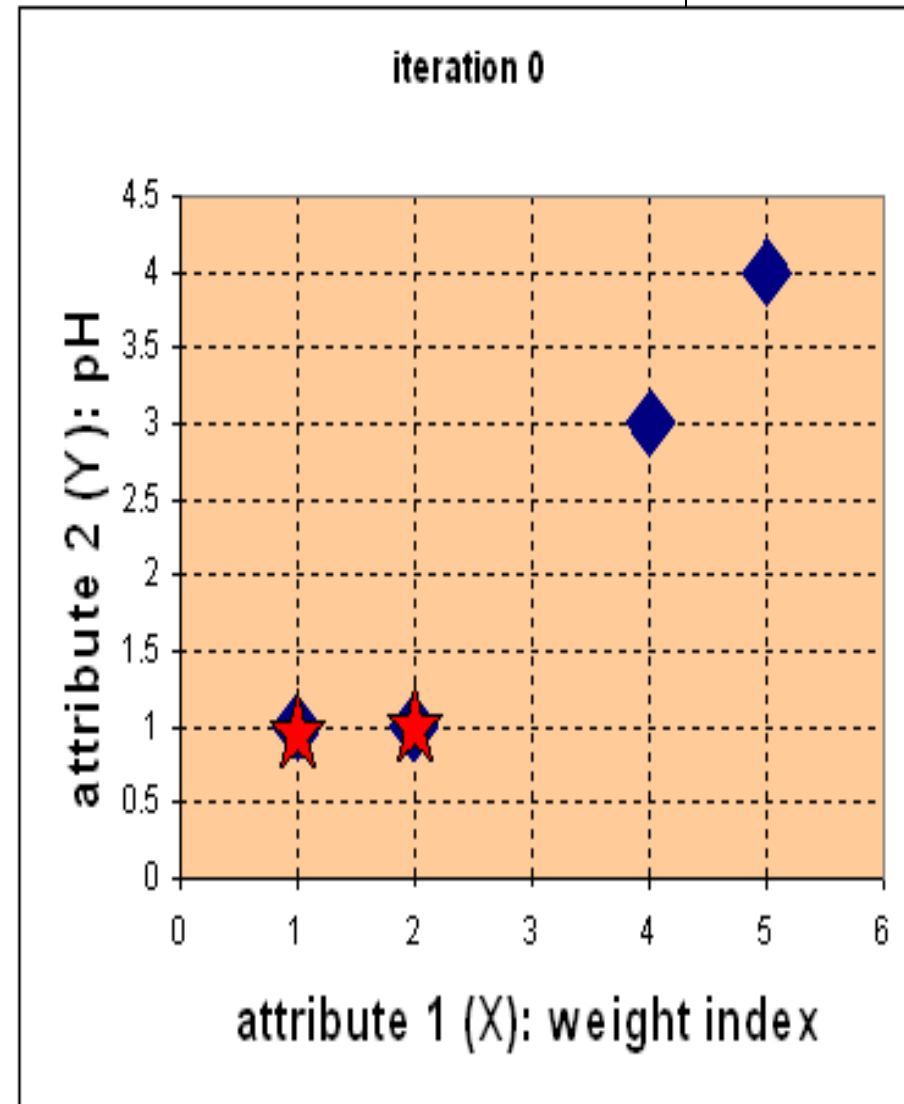
Real-Life Numerical Example of K-Means Clustering

We have 4 medicines as our training data points object and each medicine has 2 attributes. Each attribute represents coordinate of the object. We have to determine which medicines belong to cluster 1 and which medicines belong to the other cluster.

Object	Attribute1 weight index (X):	Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

Step 1:

- **Initial value of centroids** : Suppose we use medicine A and medicine B as the first centroids.
- Let c_1 and c_2 denote the coordinate of the centroids, then $c_1=(1,1)$ and $c_2=(2,1)$



- **Objects-Centroids distance** : we calculate the distance between cluster centroid to each object. Let us use Euclidean distance, then we have distance matrix at iteration 0 is

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1,1) \text{ group-1} \\ \mathbf{c}_2 = (2,1) \text{ group-2} \end{array}$$

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	
$\left[\begin{array}{c} 1 \\ 1 \end{array} \right]$	$\left[\begin{array}{c} 2 \\ 1 \end{array} \right]$	$\left[\begin{array}{c} 4 \\ 3 \end{array} \right]$	$\left[\begin{array}{c} 5 \\ 4 \end{array} \right]$	$\left[\begin{array}{c} X \\ Y \end{array} \right]$

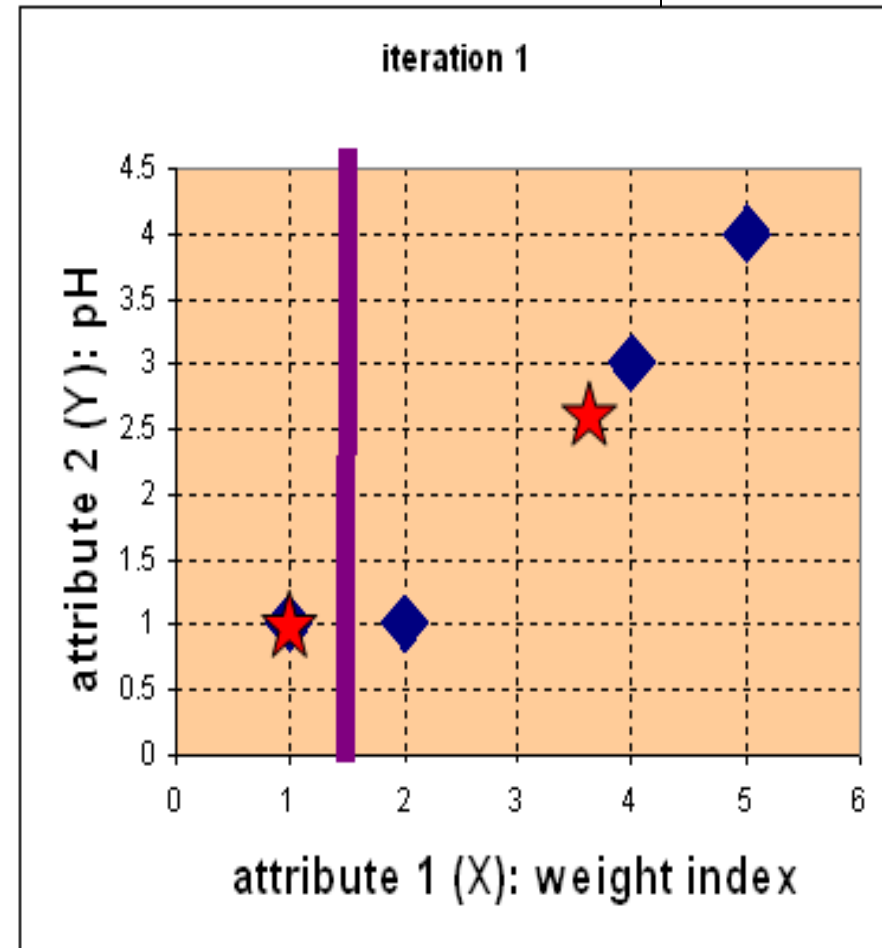
- Each column in the distance matrix symbolizes the object.
- The first row of the distance matrix corresponds to the distance of each object to the first centroid and the second row is the distance of each object to the second centroid.
- For example, distance from medicine C = (4, 3) to the first centroid $\mathbf{c}_1 = (1,1)$ is , $\sqrt{(4-1)^2 + (3-1)^2} = 3.61$ and its distance to the second centroid is , $\mathbf{c}_2 = (2,1)$ is $\sqrt{(4-2)^2 + (3-1)^2} = 2.83$ etc.

Step 2:

- **Objects clustering** : We assign each object based on the minimum distance.
- Medicine A is assigned to group 1, medicine B to group 2, medicine C to group 2 and medicine D to group 2.
- The elements of Group matrix below is 1 if and only if the object is assigned to that group.

$$\mathbf{G}^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{array}{l} \textit{group} - 1 \\ \textit{group} - 2 \end{array}$$

A *B* *C* *D*



The
all

- **Iteration-1, Objects-Centroids distances** : next step is to compute the distance of objects to the new centroids.
- Similar to step 2, we have distance matrix at iteration 1 is

$$\mathbf{D}^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1,1) \text{ group - 1} \\ \mathbf{c}_2 = (\frac{11}{3}, \frac{8}{3}) \text{ group - 2} \end{array}$$

A B C D

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{array}{l} X \\ Y \end{array}$$

Iteration-1, Objects

clustering: Based on the new distance matrix, we move the medicine B to Group 1 while all the other objects remain. The Group matrix is shown below

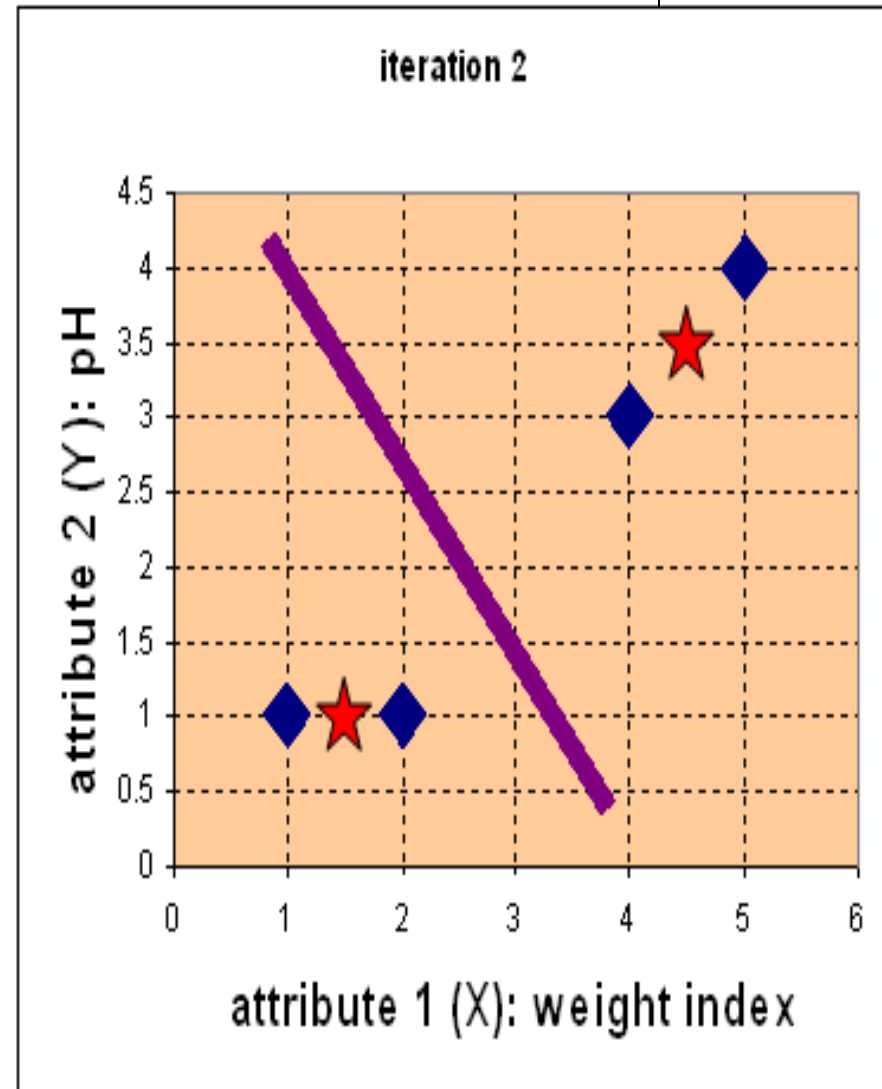
$$\mathbf{G}^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{array}{l} \textit{group - 1} \\ \textit{group - 2} \end{array}$$

A B C D

Iteration 2, determine

centroids: Now we repeat step 4 to calculate the new centroids coordinate based on the clustering of previous iteration. Group 1 and group 2 both has two members, thus the new centroids are $c_1 = (\frac{1+2}{2}, \frac{1+1}{2}) = (1\frac{1}{2}, 1)$

and $c_2 = (\frac{4+5}{2}, \frac{3+4}{2}) = (4\frac{1}{2}, 3\frac{1}{2})$



- **Iteration-2, Objects-Centroids distances :**
Repeat step 2 again, we have new distance matrix at iteration 2 as

$$\mathbf{D}^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1\frac{1}{2}, 1) \text{ group - 1} \\ \mathbf{c}_2 = (4\frac{1}{2}, 3\frac{1}{2}) \text{ group - 2} \end{array}$$

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	
[1	2	4	5] <i>X</i>
[1	1	3	4] <i>Y</i>

- **Iteration-2, Objects clustering:** Again, we assign each object based on the minimum distance.

$$\mathbf{G}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{array}{l} \text{group - 1} \\ \text{group - 2} \end{array}$$

A B C D

- We obtain result that $\mathbf{G}^2 = \mathbf{G}^1$. Comparing the grouping of last iteration and this iteration reveals that the objects does not move group anymore.
- Thus, the computation of the k-mean clustering has reached its stability and no more iteration is needed..

We get the final grouping as the results as:

<u>Object</u>	<u>Feature1(X): weight index</u>	<u>Feature2 (Y): pH</u>	<u>Group (result)</u>
Medicine A	1	1	1
Medicine B	2	1	1
Medicine C	4	3	2
Medicine D	5	4	2

K-Means Clustering Visual Basic Code

```
Sub kMeanCluster (Data() As Variant, numCluster As Integer)
' main function to cluster data into k number of Clusters
' input:
' + Data matrix (0 to 2, 1 to TotalData);
' Row 0 = cluster, 1 =X, 2= Y; data in columns
' + numCluster: number of cluster user want the data to be clustered
' + private variables: Centroid, TotalData
' ouput:
' o) update centroid
' o) assign cluster number to the Data (= row 0 of Data)
```

```
Dim i As Integer
Dim j As Integer
Dim X As Single
Dim Y As Single
Dim min As Single
Dim cluster As Integer
Dim d As Single
Dim sumXY()
```

```
Dim isStillMoving As Boolean
isStillMoving = True
if totalData <= numCluster Then
'only the last data is put here because it designed to be interactive
Data(0, totalData) = totalData ' cluster No = total data
Centroid(1, totalData) = Data(1, totalData) ' X
Centroid(2, totalData) = Data(2, totalData) ' Y
Else
'calculate minimum distance to assign the new data
min = 10 ^ 10 'big number
X = Data(1, totalData)
Y = Data(2, totalData)
For i = 1 To numCluster
```

```
Do While isStillMoving
' this loop will surely convergent
'calculate new centroids
' 1 =X, 2=Y, 3=count number of data
ReDim sumXY(1 To 3, 1 To numCluster)
For i = 1 To totalData
sumXY(1, Data(0, i)) = Data(1, i) + sumXY(1, Data(0, i))
sumXY(2, Data(0, i)) = Data(2, i) + sumXY(2, Data(0, i))
Data(0, i))
sumXY(3, Data(0, i)) = 1 + sumXY(3, Data(0, i))
Next i
For i = 1 To numCluster
Centroid(1, i) = sumXY(1, i) / sumXY(3, i)
Centroid(2, i) = sumXY(2, i) / sumXY(3, i)
Next i
'assign all data to the new centroids
isStillMoving = False

For i = 1 To totalData
min = 10 ^ 10 'big number
X = Data(1, i)
Y = Data(2, i)
For j = 1 To numCluster
d = dist(X, Y, Centroid(1, j), Centroid(2, j))
If d < min Then
min = d
cluster = j
End If
Next j
If Data(0, i) <> cluster Then
Data(0, i) = cluster
isStillMoving = True
End If
Next i
Loop
End If
End Sub
```

Weaknesses of K-Mean Clustering

1. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
2. The number of cluster, K , must be determined before hand. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.
3. We never know the real cluster, using the same data, because if it is inputted in a different order it may produce different cluster if the number of data is few.
4. It is sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the local optimum.

Applications of K-Mean Clustering

- It is relatively *efficient and fast*. It computes result at $O(tkn)$, where n is number of objects or points, k is number of clusters and t is number of iterations.
- k-means clustering can be applied to *machine learning or data mining*
- *Used on acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization or Image Segmentation).*
- *Also used for choosing color palettes on old fashioned graphical display devices and Image Quantization.*

CONCLUSION

- *K-means algorithm* is useful for undirected knowledge discovery and is relatively simple. K-means has found wide spread usage in lot of fields, ranging from unsupervised learning of neural network, Pattern recognitions, Classification analysis, Artificial intelligence, image processing, machine vision, and many others.