Dimensionality Reduction

Department of Computer Languages and Systems

General context

General objective of data reduction: to obtain a reduced representation of the data that is much smaller in size and make it feasible for analysis

We need to reduce the data because …

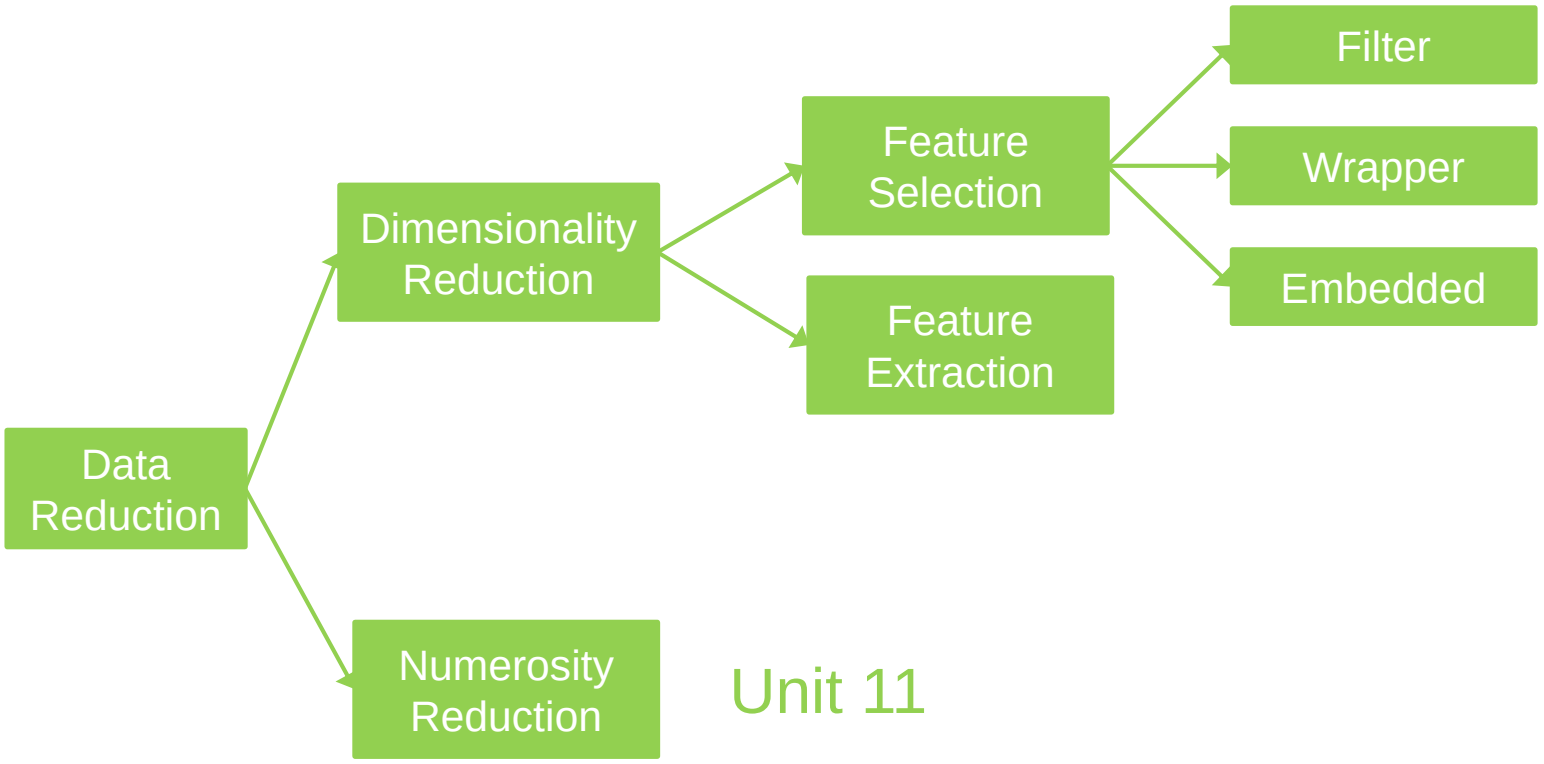
- data analysis could be too complex with huge data sets
- data analysis could be too computational expensive with huge data sets
- noisy instances can lead to misclassifications
- irrelevant and redundant attributes can hinder any machine learning process

General context (ii)

Data reduction is simply ..

- the reduction in the number of attributes or ←
- the reduction in the number of instances or
- the reduction in both the number of instances and the number of attributes

Categorization



Unit 11

Dimensionality reduction

Some reasons to go in for dimensionality reduction:

- reducing the complexity of the model (a simpler model is easier to understand and interpret)
- improving the predictive performance of models (less noise)
- providing faster and more cost-effective models (reducing time and storage requirements)

Dimensionality reduction (ii)

We can achieve these goals by removing …

- irrelevant attributes
- attributes with zero/low variance
- highly correlated attributes
- redundant attributes

The curse of dimensionality

- The previous challenges in dealing with high dimensional data are encapsulated in the term **Curse of Dimensionality**
  - as the number of dimensions increases, the sample size needs to increase exponentially in order to achieve an effective estimate of performance

The curse of dimensionality (ii)

- all the efforts to reduce dimensionality are geared towards the Principle of Parsimony or the Occam's Razor, which in Machine Learning means to develop the simplest and most explainable model
- dimensionality reduction is more important nowadays because of the Big Data phenomenon

The curse of dimensionality (iii)

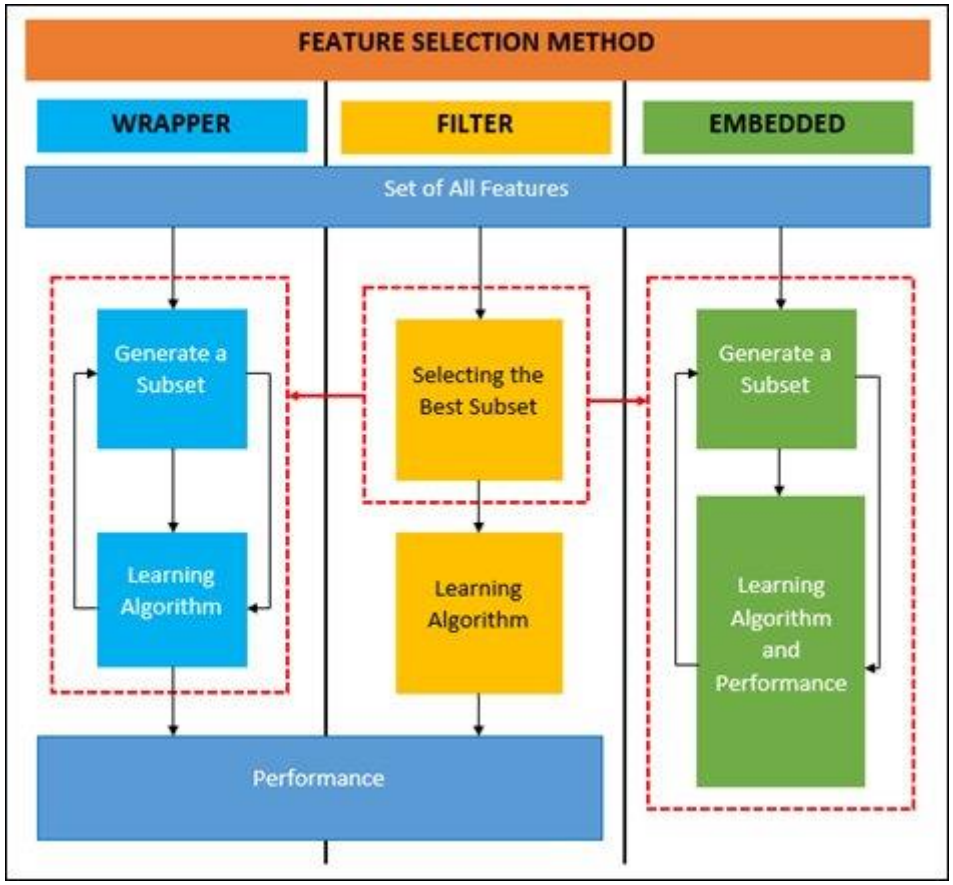
- some applications are characterized by a very large number of attributes and only a few number of instances
  - for instance, DNA microarrays to measure the expression levels of many thousands of genes (attributes) simultaneously while the number of samples is a few tens to hundreds
  - however, of the huge number of genes, only a few of them are related to the target phenotypes. For instance, in a two-class cancer subtype classification problem only a few genes are often sufficient
  - when a small number of genes are selected, their biological relationship with the target diseases is more easily identified

Feature selection

- The premise when using a feature selection algorithm is that the data contains many attributes that can be removed without incurring much loss of information
- A feature selection algorithm combines two main elements:
  - a search technique for proposing new feature subsets
  - an evaluation measure that scores the different feature subsets
- The simplest algorithm is based on an exhaustive search of the space: to test each possible subset finding the one that minimizes the error rate → it is computationally intractable

Feature selection (ii)

Three main categories:



Wrapper methods

- wrappers use a predictive model to score feature subsets
  - they use a search technique to search through the space of possible attributes
  - the usefulness of a feature subset is directly judged by the estimated accuracy of the learning model
- as wrapper methods train the model for each subset, they are very computationally intensive, but usually provide the best performing feature set for that particular type of model
- wrappers have a risk of overfitting to the model

Filter methods

- the selection of attributes is independent of any learning algorithm
- attributes are selected on the basis of their relationship to the class label
- filters use some proxy measure instead of the error rate to score a feature subset
- fast and computationally inexpensive, thus usually the best approach when the number of attributes is huge
- many filters provide an ordered ranking of attributes rather than an explicit best feature subset → **feature ranking algorithms**

Embedded methods

- these perform feature selection during the model construction process → they use algorithms that have built-in feature selection methods
- they tend to be between filters and wrappers in terms of computational complexity
- an example of this approach is the LASSO method for constructing a linear model:
  - LASSO penalizes the regression coefficients, reducing many of them to zero or almost zero
  - any features that have non-zero regression coefficients are 'selected' by the LASSO algorithm

Search techniques

- Sequential forward selection (SFS): an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature that best improves our model till an addition of a new variable does not improve the performance of the model, or until the desired number of features is reached
- Sequential backward elimination (SBE): we start with all the features and removes the least significant feature at each iteration that improves the performance of the model until no improvement is observed on removal of features, or until the desired number of features is reached
- Bidirectional selection: we start the search in both directions, performing SFS and SBE concurrently. It stops when one search finds the best subset with the desired number of features, or when both searches achieve the middle of the search space

Common proxy measures

- Information measures quantify the information that can be gained from each attribute
- Dependency measures evaluates the strength of the correlation between two attributes or between an attribute and the class
- Distance measures assess the separability between classes
- Consistency measures attempt to find the minimum number of attributes that ‘consistently’ discriminate classes as if using the full set of attributes
- Accuracy measures evaluates the performance of each subset of attributes

Information measures

- Mutual information (or information gain): this is a measure of the mutual dependence between two random variables

$$I(c, f) = H(c) - H(c|f)$$

where $I(c, f)$ is the mutual information between class *c* and attribute *f*, $H(c)$ is the initial entropy, and $H(c|f)$ is the conditional entropy

  - in brief, the mutual information measures the reduction in uncertainty (entropy) of the class due to knowledge of the attribute value

Information measures (ii)

- Entropy measures the uncertainty of a data set (and its class labels)

$$H(c) = -\sum_{c=1}^{N_c} p(c) \log_2 p(c)$$

where $p(c)$ is the prior probability (proportion) of class c in a data set with N_c classes

  - a low entropy indicates that the data labels are quite uniform (e.g., suppose a data set with 1 Positive and 99 Negative samples, then the entropy is very low. If all the 100 samples are Positive, then the entropy is zero)
  - a high entropy means the labels are in chaos, that is, lower information (e.g., a dataset with 45 Positive samples and 55 Negative samples has a very high entropy)

Information measures (iii)

- Conditional entropy is the uncertainty in a data set when the information of the attributes is already known

$$H(c|f) = -\sum_{f=1}^{N_f} p(f) \left(\sum_{c=1}^{N_c} p(c|f) \log_2 p(c|f) \right)$$

where $p(f)$ is the probability of attribute f, and $p(c|f)$ is the conditional probability for class c given the input attribute f

Dependency measures

- Pearson's correlation coefficient is a measure of the linear relationship of two or more variables

$$r_{x,y} = \frac{\sum_{i=1}^{d}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{d}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{d}(y_i - \bar{y})^2}}$$

  - the logic behind using correlation for feature selection is that the good variables are highly correlated with the class
  - furthermore, variables should be correlated with the class but should be uncorrelated among themselves
  - if two attributes are correlated, the model only really needs one of them, as the second one does not add additional information → redundant attributes

Dependency measures (ii)

- Spearman's correlation coefficient is equal to the Pearson's correlation coefficient, but
  - Pearson's correlation is the common choice for continuous data with linear relationships, while Spearman's correlation is for continuous data with non-linear relationships and for ordinal data

Consistency measures

- Consistency-based filter: it evaluates the worth of a subset of attributes by the level of consistency in the class values
  - inconsistency is defined as the case of two instances with the same inputs (same attribute values) but with different output values (classes)

Other proxy measures

- Variance threshold: it removes all attributes whose variance does not reach some threshold
  - by default, it removes all zero-variance attributes, i.e., attributes that have the same value in all samples
  - we assume that attributes with higher variance may contain more useful information
- Mean absolute difference (MAD): it computes the absolute difference from the mean value
  - we assume that attributes whose values differ more from the mean value may have a higher discriminatory power

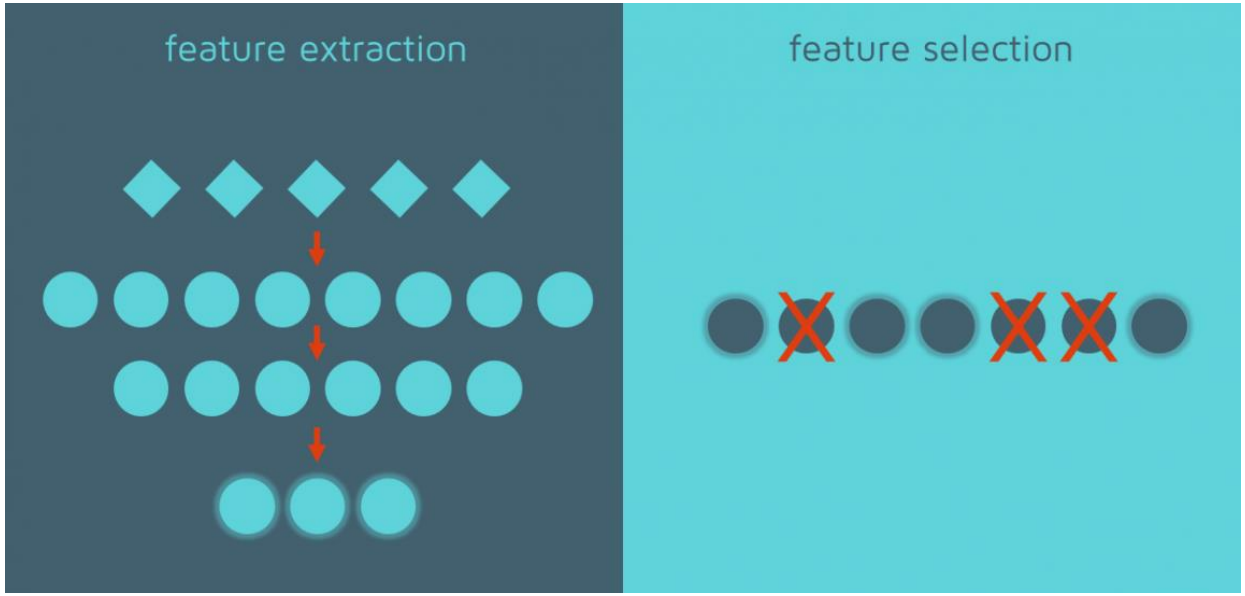
Filter vs. Wrapper vs. Embedded

	Filter method	Wrapper method	Embedded method
What is it?	Uses proxy measure	Uses predictive model	Feature selection is embedded in the model building phase
Speed	Computationally faster	Slower	Medium
Overfitting	Avoids overfitting	Prone to overfitting	Less prone to overfitting
Performance	Sometimes may fail to select best features	Better performance	Good performance

Feature extraction

Feature extraction transforms the data onto a new feature space containing basically the same information as the original attributes

(remember that feature selection keeps a subset of the original attributes)



Feature extraction (ii)

- in general, the feature extraction algorithms give new attributes that are a linear combination of the existing ones
- the objective is to capture the same information with fewer attributes
- feature extraction is not useful when model interpretability is a key requirement
- two of the most widely used techniques are PCA (Principal Component Analysis) and LDA (Linear Discriminant Analysis)

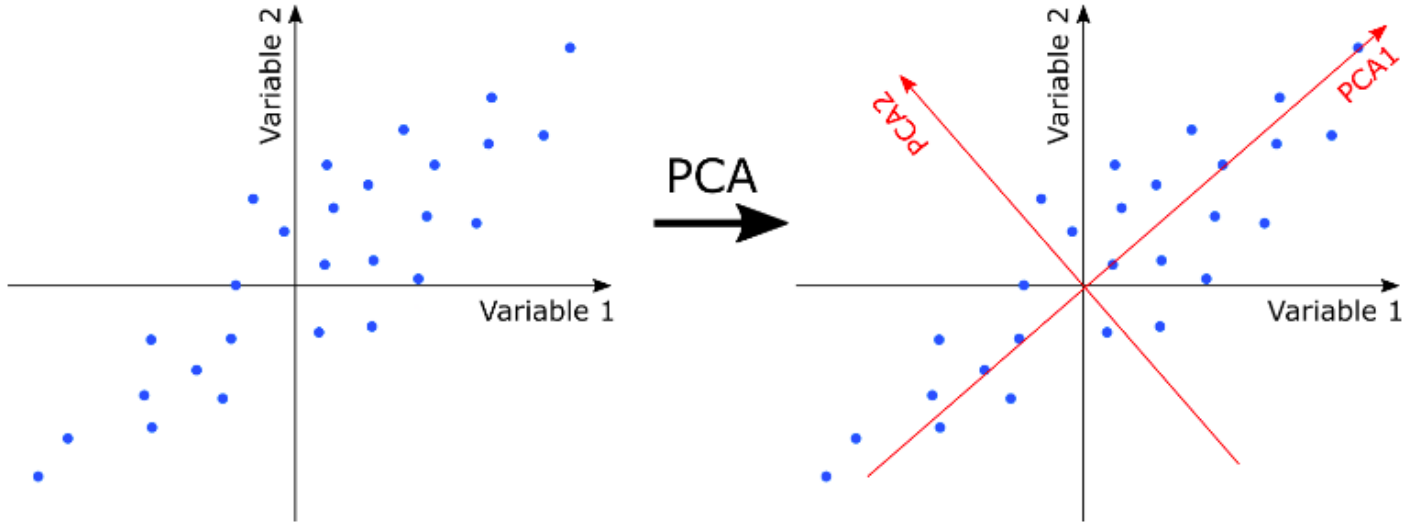
PCA

- in simple words, PCA is an unsupervised learning technique for obtaining important variables (in form of components) by finding a small combination of variables that best summarizes the initial attributes
- it tends to find the most significant attributes (those with the direction of maximum variance) in data and project the data to a set of orthogonal axes

PCA (ii)

- a principal component is a normalized linear combination of the original attributes that captures most of the variance of the data
  - the first principal component (PCA1) will always be in the direction of maximum variance
  - all the principal components will be perpendicular to each other. The main intention behind this is that no information present in PCA1 will be present in PCA2 when they are perpendicular to each other
  - principal components are orthogonal projections (perpendicular) of data onto lower-dimensional space

PCA (iii)

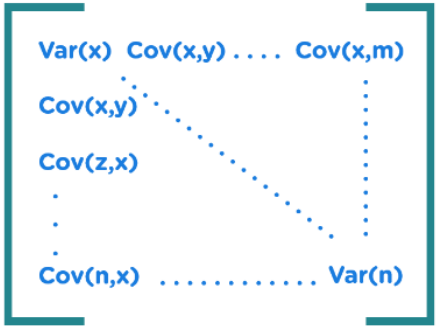


In the above figure, we have several points plotted on a 2-D plane. There are two principal components: PCA1 is the primary principal component that explains the maximum variance in the data; PCA2 is another principal component that is orthogonal to PCA1

PCA (iv)

How does PCA work?

1. Normalize the data to ensure that each feature has a mean = 0 and variance = 1
2. Find the covariance matrix A of the scaled data



The off-diagonal elements of A represent the covariance among each pair of variables and the diagonal elements represent the variances of each variable/dimension

3. Find the eigen values of matrix A: $|\mathrm{A} - \lambda \mathrm{I}| = 0$
4. Find the eigen vector v of matrix A corresponding to the eigen value λ
5. Project the scaled data into one dimension using eigen vector v

PCA (v)

Remember that ...

variance is the square of the standard deviation

$$\mathrm{var}(X) = s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}$$

covariance depicts spread of data related to two variables

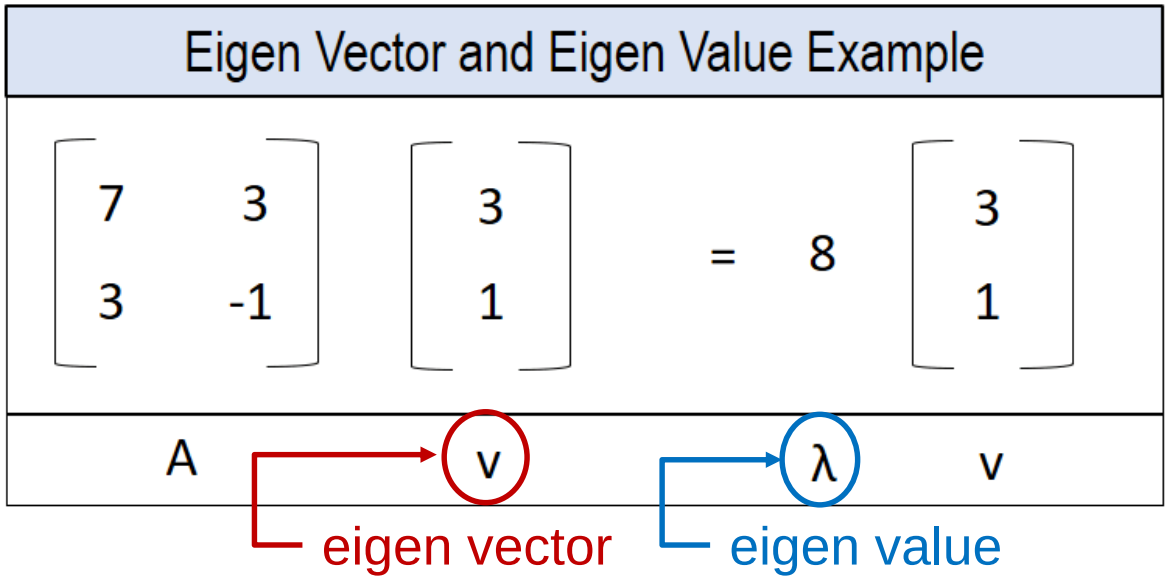
$$\mathrm{cov}(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

PCA (vi)

Remember that ...

v is an eigen vector of a square matrix A if multiplying v and A yields another matrix (or a vector) that is an scalar multiple of v (it does not change direction)

$$Av = \lambda v$$



PCA (vii)

1. Normalize (z-score) the data

Original data

X	Y
2	3
4	5
6	5
6	7
7	8
5	8

Z-Score calculation

X	Y
(2-5)/1.633	(3-6)/1.826
(4-5)/1.633	(5-6)/ 1.826
(6-5)/1.633	(5-6)/ 1.826
(6-5)/1.633	(7-6)/ 1.826
(7-5)/1.633	(8-6)/ 1.826
(5-5)/1.633	(8-6)/ 1.826

Scaled data

X	Y
-1.837	-1.643
-0.612	-0.548
0.612	-0.548
0.612	0.548
1.225	1.095
0	1.095

PCA (viii)

2. Find the covariance matrix A of the scaled data

$$\mathrm{var}(\mathrm{X}) = \frac{\sum_{i=1}^{n}(\mathrm{x}_i - \bar{\mathrm{x}})^2}{n-1} = 1.2$$

$$\mathrm{var}(\mathrm{Y}) = \frac{\sum_{i=1}^{n}(\mathrm{y}_i - \bar{\mathrm{y}})^2}{n-1} = 1.2$$

$$\mathrm{cov}(\mathrm{X},\mathrm{Y}) = \frac{\sum_{i=1}^{n}(\mathrm{x}_i - \bar{\mathrm{x}})(\mathrm{y}_i - \bar{\mathrm{y}})}{n-1} = 0.939$$

$$A = \begin{bmatrix} 1.2 & 0.939 \\ 0.939 & 1.2 \end{bmatrix}$$

PCA (ix)

3. Find the eigen values of matrix A: $|A - \lambda \mathrm{I}| = 0$

$$A - \lambda \mathrm{I} = \begin{bmatrix} 1.2 & 0.939 \\ 0.939 & 1.2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1.2 - \lambda & 0.939 \\ 0.939 & 1.2 - \lambda \end{bmatrix}$$

$$\begin{vmatrix} 1.2 - \lambda & 0.939 \\ 0.939 & 1.2 - \lambda \end{vmatrix} = 0 \rightarrow \lambda = 2.139, \lambda = 0.261$$

As we will consider one PCA component only, we will go with the highest eigen value $\boldsymbol{\lambda = 2.139}$

PCA (x)

4. Find the eigen vector v of matrix A corresponding to λ

$$A\mathrm{v} = \lambda\mathrm{v} \rightarrow \begin{bmatrix} 1.2 & 0.939 \\ 0.939 & 1.2 \end{bmatrix} \begin{bmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \end{bmatrix} = 2.139 \begin{bmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \end{bmatrix}$$

$$\mathrm{v} = \begin{vmatrix} 0.7071 \\ 0.7071 \end{vmatrix}$$

PCA (xi)

5. Project the scaled data into one dimension using eigen vector v

Scaled data

X	Y
-1.837	-1.643
-0.612	-0.548
0.612	-0.548
0.612	0.548
1.225	1.095
0	1.095

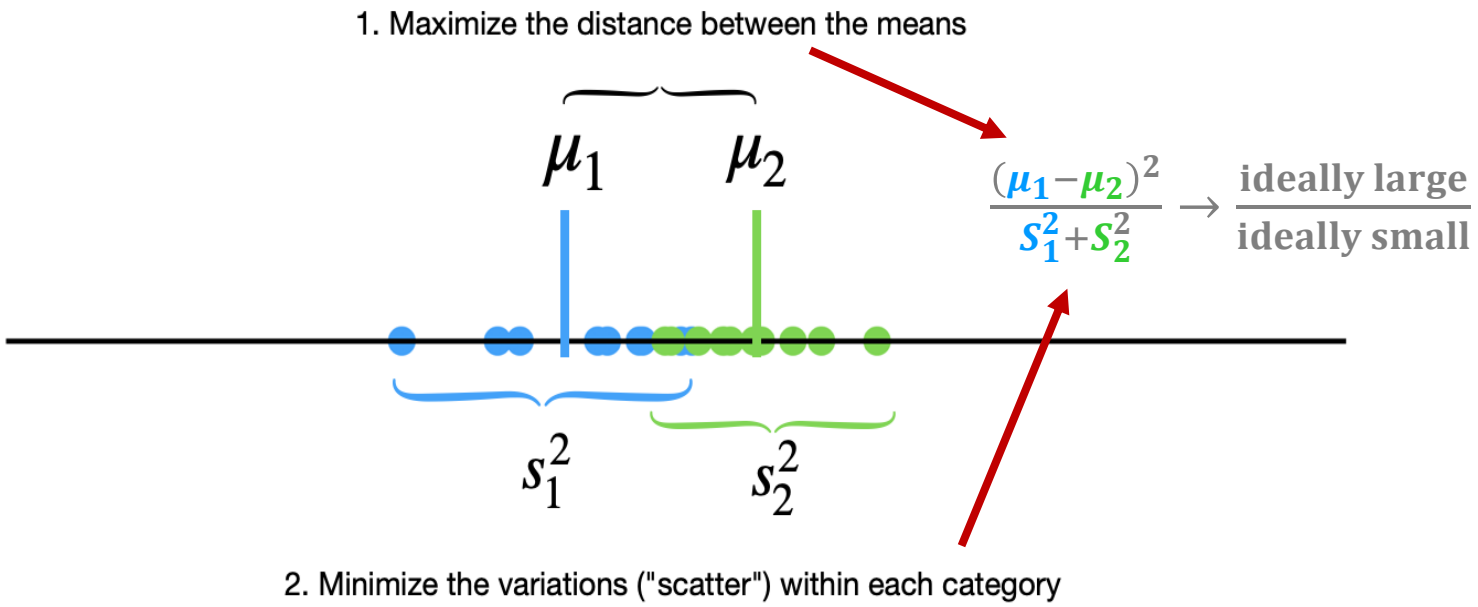
$$\begin{bmatrix}0.7071\\0.7071\end{bmatrix}^T \times \begin{bmatrix}-1.837\\-1.643\end{bmatrix} = -2.461$$

Similarly, if we project all the feature vectors, we will end up with the following projected data:

Projected Data
-2.461
-0.820
0.046
0.820
1.640
0.774

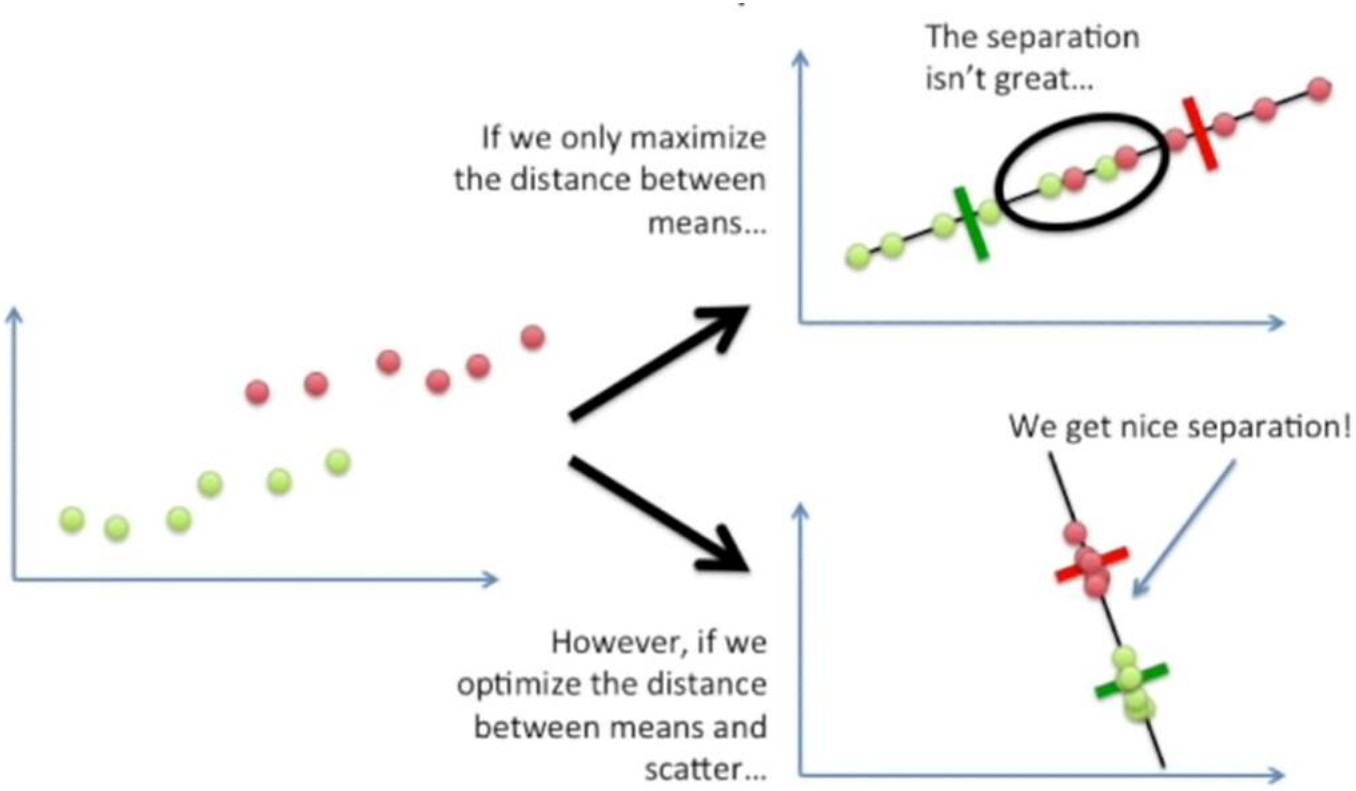
LDA

- LDA projects the data onto a new new axis to maximize the separation of the classes. The new axis is created according to two criteria considered simultaneously



LDA (ii)

An example showing why both criteria are important



LDA (iii)

How does LDA work?

1. Normalize the data to ensure that each feature has a mean = 0 and variance = 1
2. Compute the *d*-dimensional mean and variance vectors for each class
3. Compute the within-class scatter matrix S_W (measures the spread around means of each class) and the between-class scatter matrix S_B (measures the distance between class means)

$$S_W = \sum_{i=1}^{C} S_i \quad \text{where} \quad S_i = \sum_{x \in C_i}^{n_i} (x - \mu_i)(x - \mu_i)^T$$

and

$$S_B = \sum_{i=1}^{C} n_i (\mu_i - \mu)(\mu_i - \mu)^T$$

where μ is the overall mean, and μ_i and n_i are the mean and size of class i

LDA (iv)

4. Use the matrix $S_W^{-1}S_B$ to calculate the eigen vectors $(\mathrm{v}_1, \mathrm{v}_2, \cdots, \mathrm{v}_d)$ and eigen values $(\lambda_1, \lambda_2, \cdots, \lambda_d)$
5. Sort the eigen vectors by decreasing eigen values (note that the eigen vectors with the lowest eigen values are the least informative/discriminant)
6. Choose the desired number (k) of eigen vectors with the highest eigen values to form a $(k \times d)$-dimensional transformation matrix W (where each column corresponds to an eigen vector)
7. Use the matrix W to transform the data onto the new subspace

$$\mathrm{Y} = \mathrm{X} \times \mathrm{W}$$

where X is a $(n \times d)$-dimensional matrix representing the n samples, and Y are the transformed $(n \times k)$-dimensional samples in the new subspace

LDA (v)

You can find a very complete example of LDA at …

Linear Discriminant Analysis – Bit by Bit

PCA vs LDA

The main differences between PCA and LDA are:

1. LDA is supervised PCA is unsupervised
2. LDA describes the direction of maximum separability in data, while PCA describes the direction of maximum variance in data
3. Unlike PCA, LDA requires class label information