Course : CM-072

Presentation 8

Dimensionality Reduction

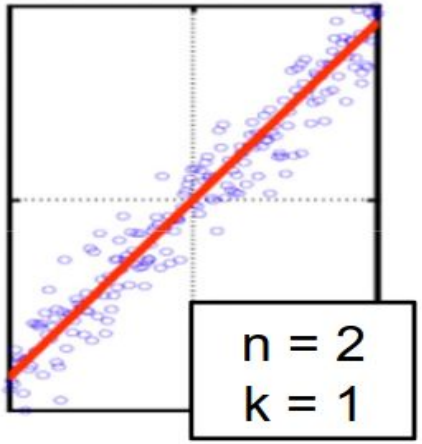
- **Unsupervised learning**
  - Goal: Find hidden patterns in the data
- **Used**
  - Visualization
  - Data compression
  - Data Preprocessing step for supervised algorithms
- **Reduce the number of considered data dimensions**
  - Feature selection: Keep a subset of the existing dimensions
  - Feature extraction: Identify combinations of the features that can be used to replace the existing features. Typically, they are more informative and better suited for analysis in high dimensional spaces.

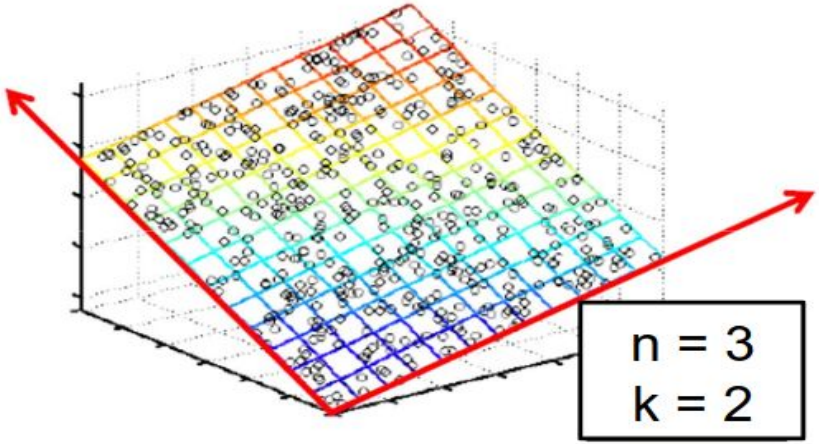
Dimensionality Reduction

- **Unsupervised learning**
  - Goal: Find hidden patterns in the data
- **Used**
  - Visualization
  - Data compression
  - Data Preprocessing step for supervised algorithms
- **Reduce the number of considered data dimensions**
  - **Feature selection:** Keep a subset of the existing dimensions
  - **Feature extraction:** Identify combinations of the features that can be used to replace the existing features. Typically, they are more informative and better suited for analysis in high dimensional spaces.
- **Feature extraction:** Principal Component Analysis algorithm.

Dimensionality Reduction

- **Assumption:**data (approximately) lies on a lower dimensional space.
- **Examples:**



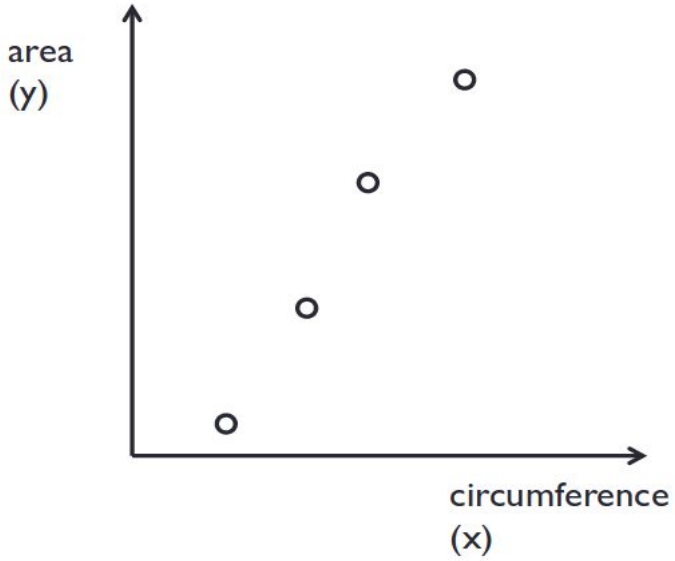


Slide from Yi Zhang

Principal Component Analysis

- **Question: Are our data features appropriate for analyzing our data? Is there a better set of features that we could use?**
- **Answer:**
  - Investigate how much correlated the original features are and how much the values of each feature are distributed in the value space.
    - Feature that are very correlated with each other contain redundancy.
    - Feature with a small variation contain low information.
  - Construct a new set of features based on the notions of **variance** and **covariance** of the original features.

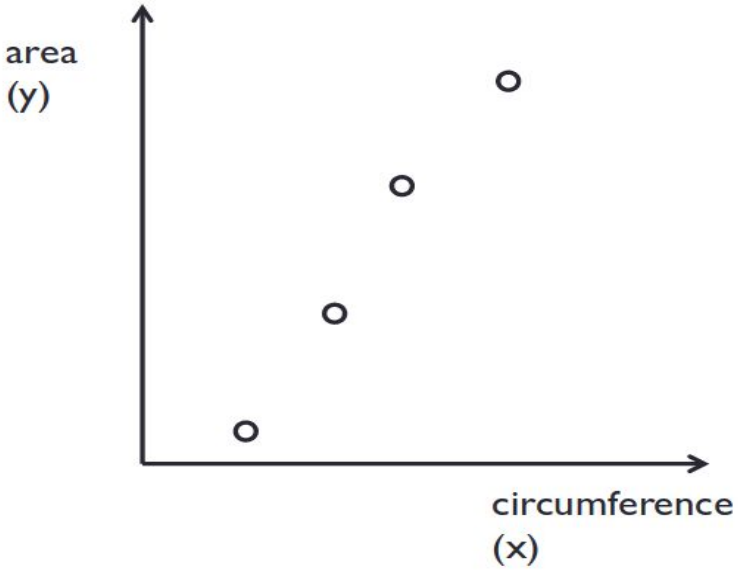
PCA: Motivating example



Circle
area= πr^2
circumference=$2 \pi r$

Here, our data are described by two features: area and circumference, which are obviously correlated!
What truly matters is the radius of the circle!

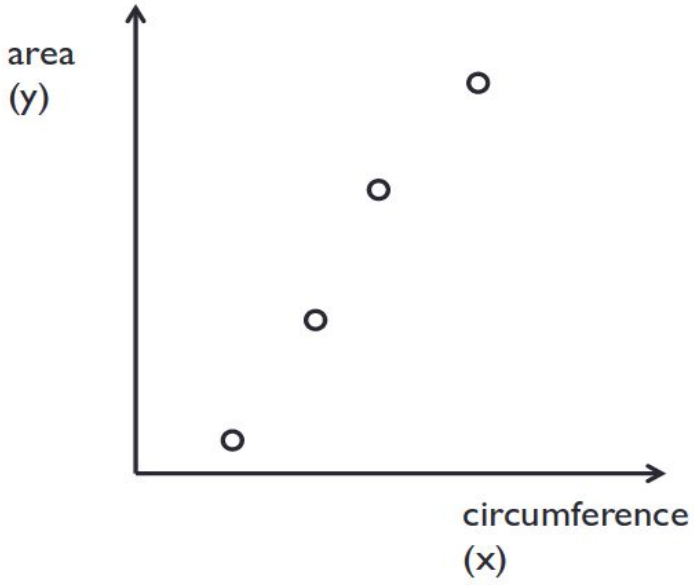
PCA: Motivating example



Circle
area= πr^2
circumference=$2 \pi r$

So, how can we obtain a new feature that is truly informative of our data?

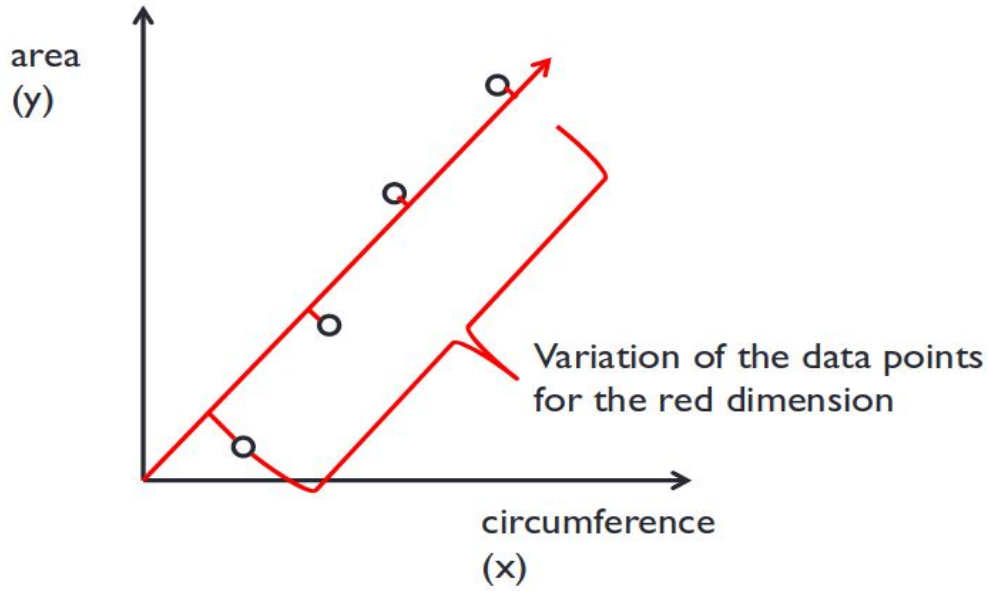
PCA: Motivating example



Circle
area= πr^2
circumference=$2 \pi r$

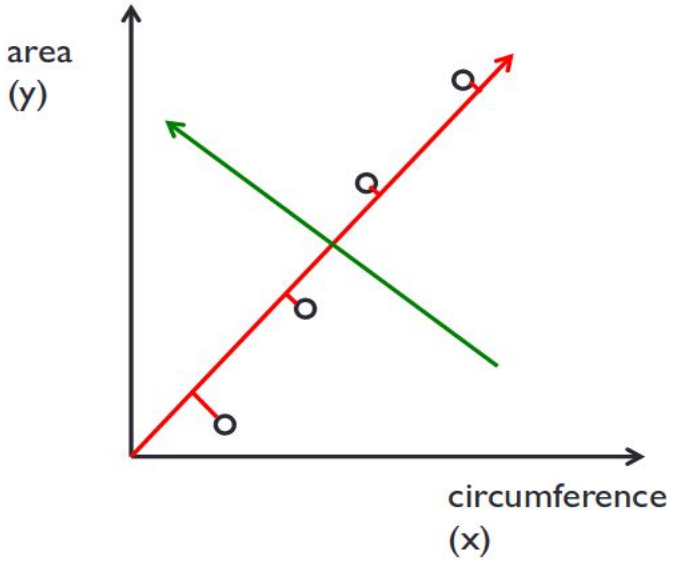
Find directions of maximal variance!

PCA: Motivating example



Circle
area= πr^2
circumference=$2 \pi r$

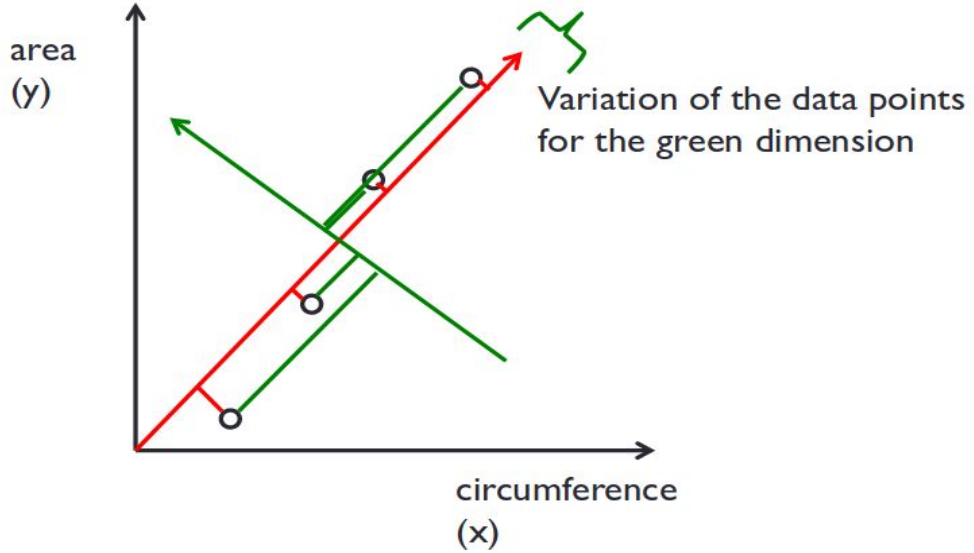
PCA: Motivating example



Circle
area= πr^2
circumference=$2 \pi r$

Find directions of maximal variance!
Find directions that are mutually orthogonal.

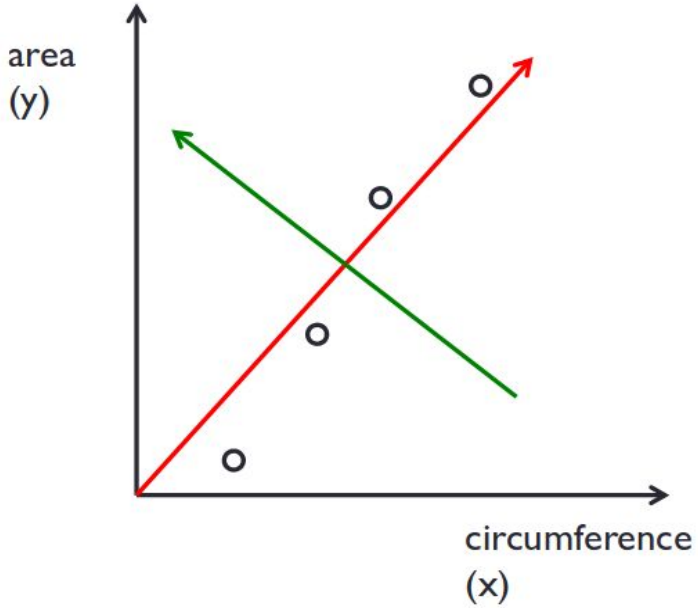
PCA: Motivating example



Circle
area= πr^2
circumference=$2 \pi r$

PCA: Motivating example

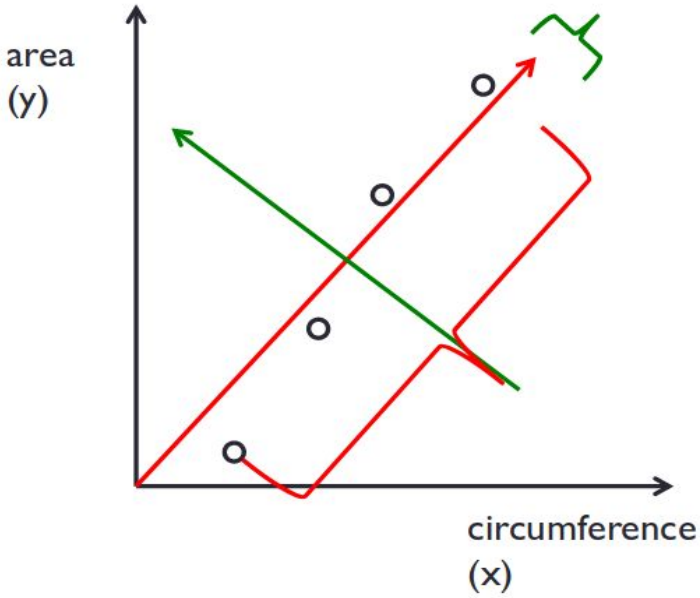
Circle
area= πr^2
circumference=$2 \pi r$



The red and the green directions are the principal components, the new dimensions for the data!

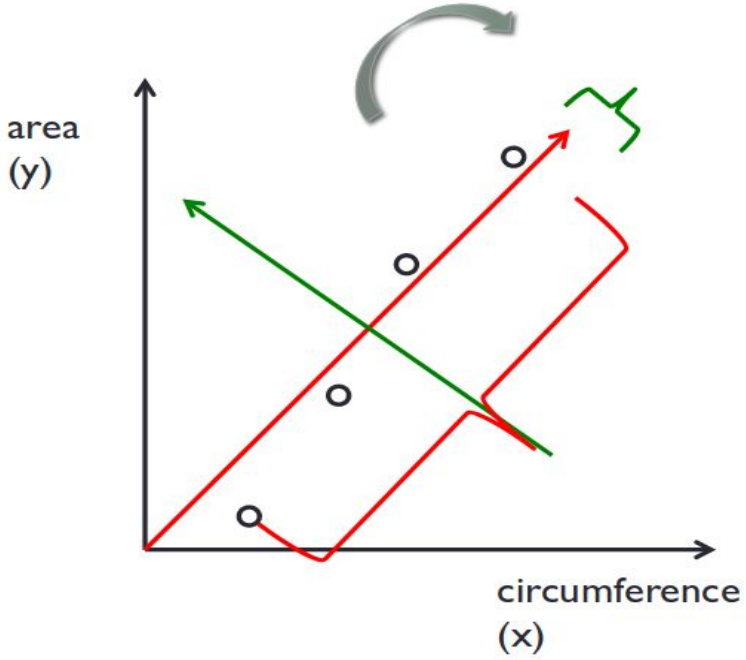
PCA: Motivating example

Circle
area= πr^2
circumference=$2 \pi r$



As the data vary (a lot) more on the red direction, this is the first principal component. The green one, is the second principal component.

PCA: Motivating example

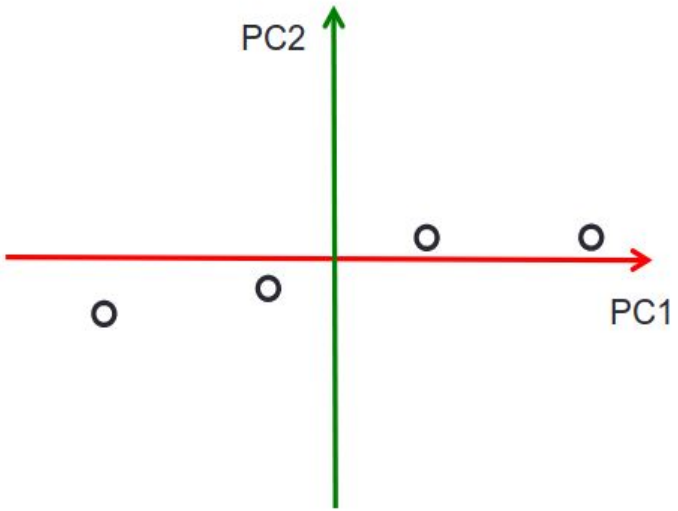


Circle
area= πr^2
circumference=$2 \pi r$

So, let's transform the data space by rotating the axes.

PCA: Motivating example

Circle
area= πr^2
circumference=$2 \pi r$



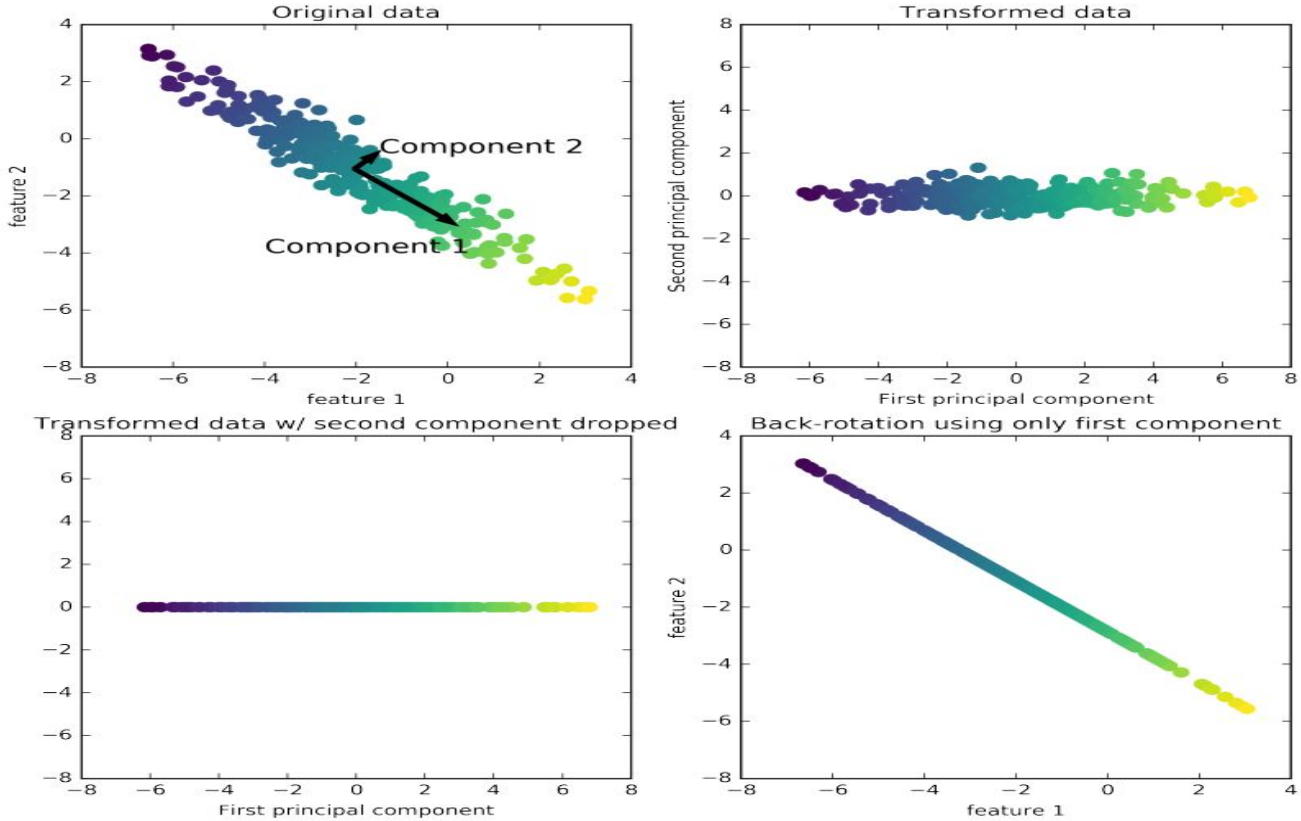
On the new dimensions, we can easily see that the most of the information is carried on PC1, and that PC2 can be safely dropped.

PCA: Motivating example

Circle
area= πr^2
circumference=$2 \pi r$

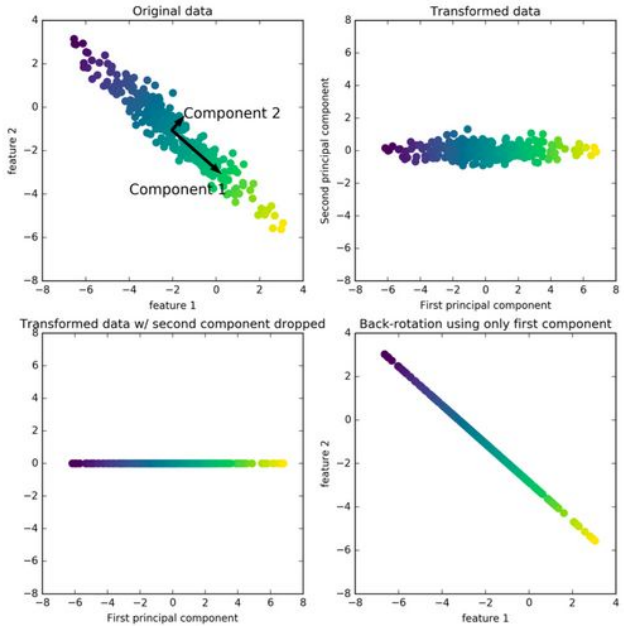
This means that the data points will be projected on PC1.
Finally, we got the one dimension that we were intuitively seeking in the beginnning.

Data transformation with PCA



PCA objectives

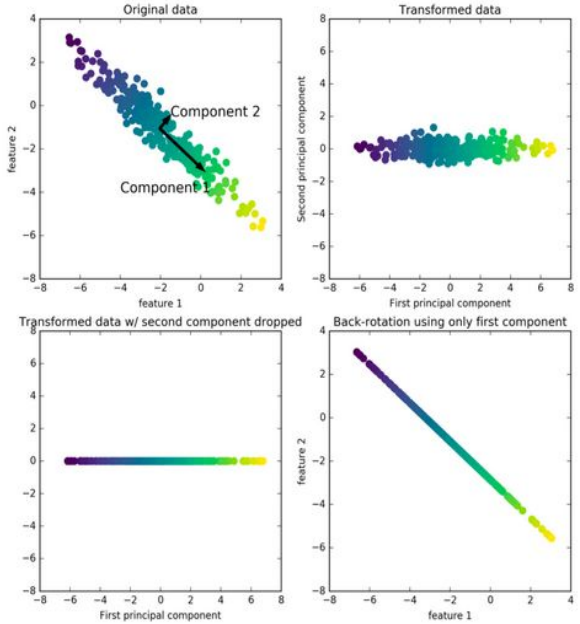
$$\min_{X', \text{rank}(X')=r} \|X - X'\|$$



PCA objectives

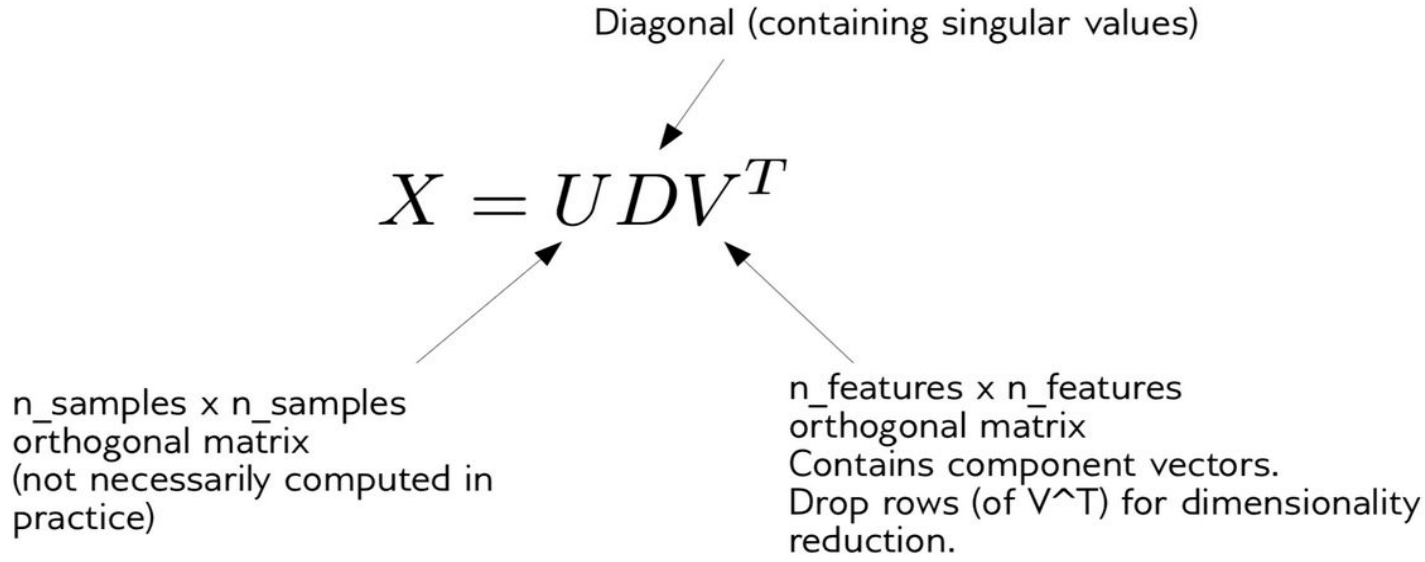
$$\max_{u_1 \in R^p, \|u_1\|=1} \operatorname{var}(Xu_1)$$

$$\max_{u_1 \in R^p, \|u_1\|=1} u_1^T \operatorname{cov}(X) u_1$$



PCA Computation

- Center X (subtract mean).
- In practice: Also scale to unit variance.
- Compute singular value decomposition:



Principal Component Analysis

- Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of **linearly uncorrelated** variables called **principal components** .
- The principal components analysis finds **directions of maximal variance**.
- All principle components are **orthogonal** to one another
- If there are N feature in the original data, there are N possible principal components. These are ordered based on their **eigenvalue** (capturing their variance).
- The principal components with the lowest variance (eigenvalue) are not informative and may be discarded to obtain M<N features (feature selection).

PCA: Algorithm

- Make the input data **zero-mean**, i.e., subtract the mean from each observation of each feature. (Sometimes we will have to also make them unit-variance by scaling).
- Compute the **covariance matrix** for the zero-mean data.
- Compute the **eigenvectors and eigenvalues** for the covariance matrix
  - The eigenvectors are the **principal components**; the first principal component has the highest eigenvalue, the second principal component the second highest, etc.
- Build the new feature vector and perform **feature selection** if necessary
  - If the eigenvalue of some principal component is zero, the feature selection is done automatically
- **Transform the data** to the new feature system.

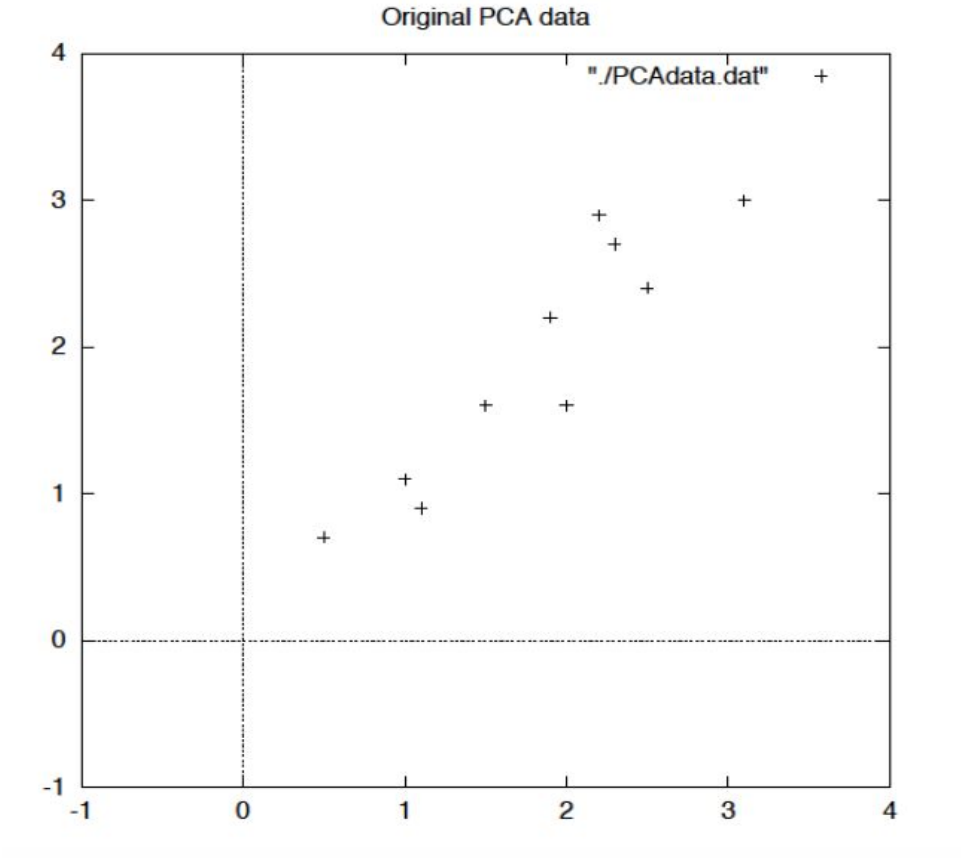
PCA Step 1: zero-mean

Original 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

Substract the mean

Zero-mean data	
x	y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

PCA Step 1: zero-mean



PCA Step 2: covariance matrix

The covariance matrix for two variables is:

	x	y
x	$\sigma^2(x)$	cov(x,y)
y	cov(y,x)	$\sigma^2(y)$

The (co)variance for zero-mean data is given by:

$$\text{cov}(\mathbf{x},\mathbf{y}) = \frac{1}{N-1}\mathbf{x}\mathbf{y}^T$$

and

$$\sigma^2(\mathbf{x}) = \frac{1}{N-1}\mathbf{x}\mathbf{x}^T$$

For our example we find (verify the result):

	x	y
x	0.616555556	0.615444444
y	0.615444444	0.716555556

Since the covariance is positive, we expect the variables to increase together.

PCA Step 3: eigenvectors- eigenvalues

For our running example, we can find:

eigenvalues
0.490833989
1.28402771

eigenvectors	
-0.735178656	-0.677873399
0.677873399	-0.735178656

Eigenvectors are the principal components.
The eigenvalues designate the first and second principal components. Here, we can see that the first principal component is far more informative than the second one, because its eigenvalue is much higher.

PCA Step 4: New feature vector

- New feature vector = (eigv1, eigv2...eigvn) a matrix of column vectors.

New Feature vector 1

$$\begin{bmatrix} -0.677873399 & -0.735178656 \\ -0.735178656 & 0.677873399 \end{bmatrix}$$

If we keep both principal components.

New Feature vector 2

$$\begin{bmatrix} -0.677873399 \\ -0.735178656 \end{bmatrix}$$

If we discard the second principal component.

PCA Step 5: New dataset

To transform the data to the new features system we apply the following formula:

TransformedData=RowNewFeatureVector x RowZeroMeanData

where

- RowNewFeatureVector: Each row represents an eigenvector. Dimensions: (#eigenvectors,#originalFeatures).
- RowZeroMeanData: Each row is a feature and each column is a (zero-mean) data item.

  Dimensions: (#originalFeatures,#dataItems).

So, TransformedData is a (#eigenvectors, #dataItems) matrix.

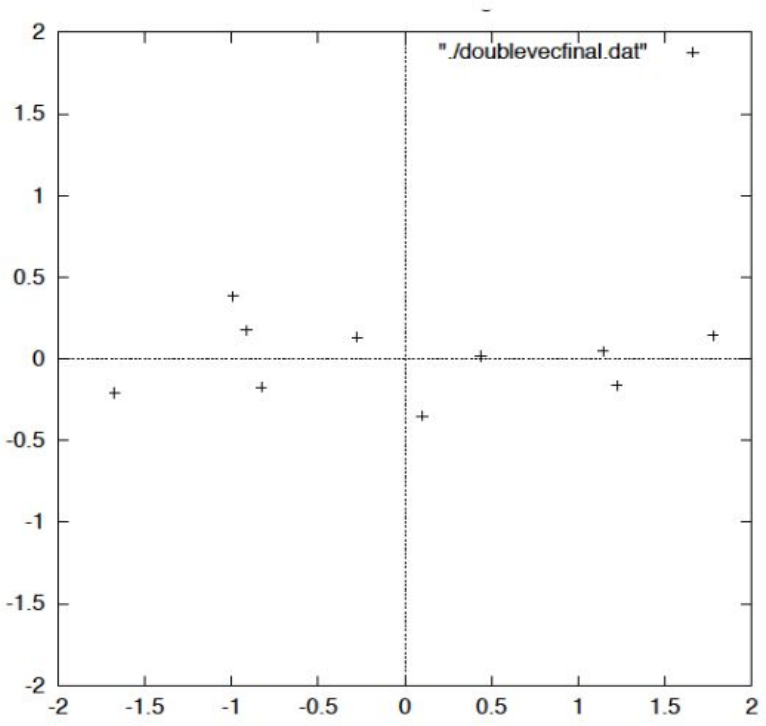
It is thus evident that the data have been transformed to the new feature system that is a linear combination of the original one.

PCA Step 5: New dataset

For our example

x	y
-.827970186	-.175115307
1.77758033	.142857227
-.992197494	.384374989
-.274210416	.130417207
-1.67580142	-.209498461
-.912949103	.175282444
.0991094375	-.349824698
1.14457216	.0464172582
.438046137	.0177646297
1.22382056	-.162675287

Data transformed with 2 eigenvectors



PCA Step 6: Next

- The new data and features are used as **input** to a subsequent supervised learning algorithm.

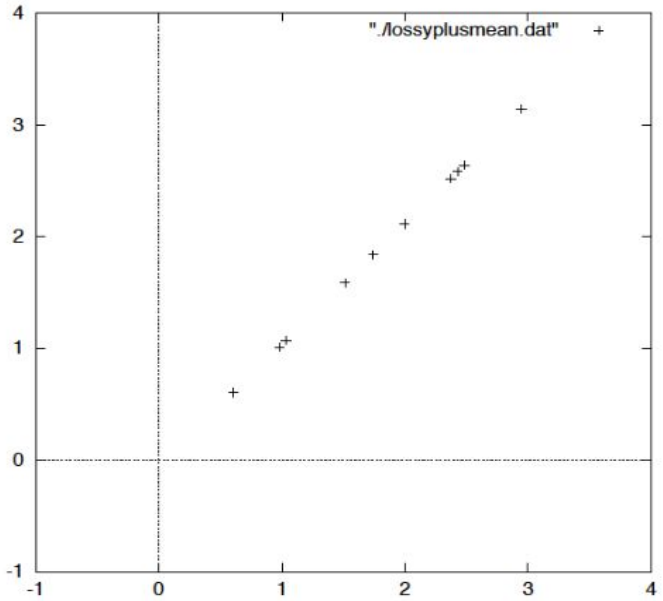
- If it used for compression we may want to get the original data back
  - Lossless: If we had kept all the principal components
  - Lossy If we kept only a number of the principal components.

PCA : Getting the original data back

Apply the formula

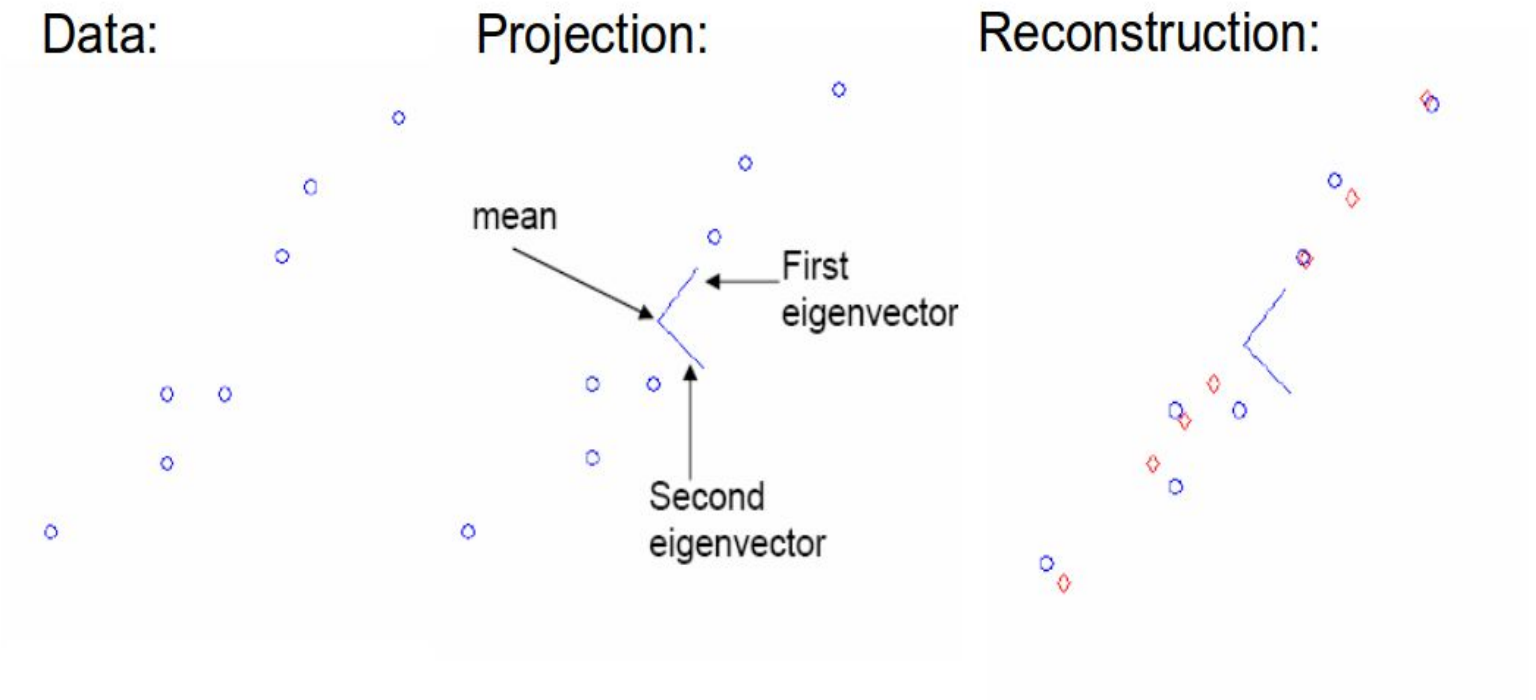
OriginalData=NewFeatureVector x TransformedData + OriginalMean*

For our example and for transformed data using one eigenvector we have the reconstructed data shown here:

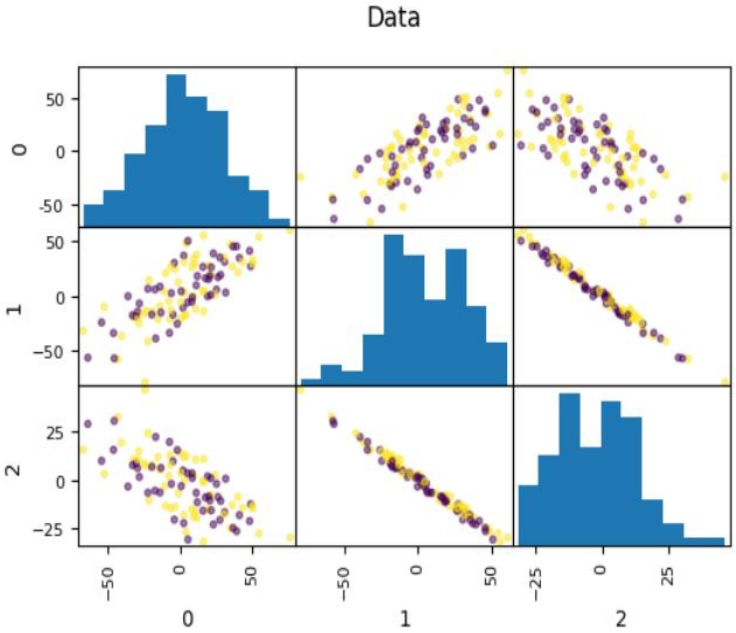


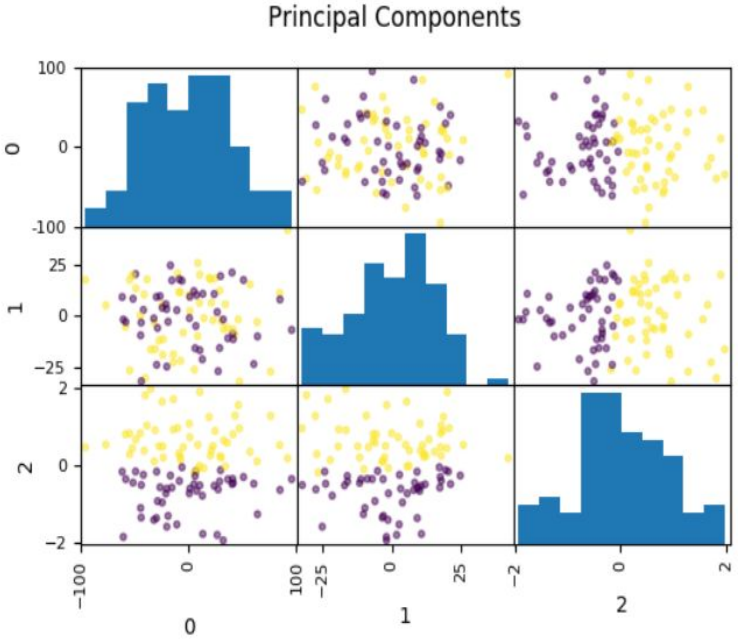
*Note that if we had also normalized to unit-variance we should revert also this normalization.

PCA

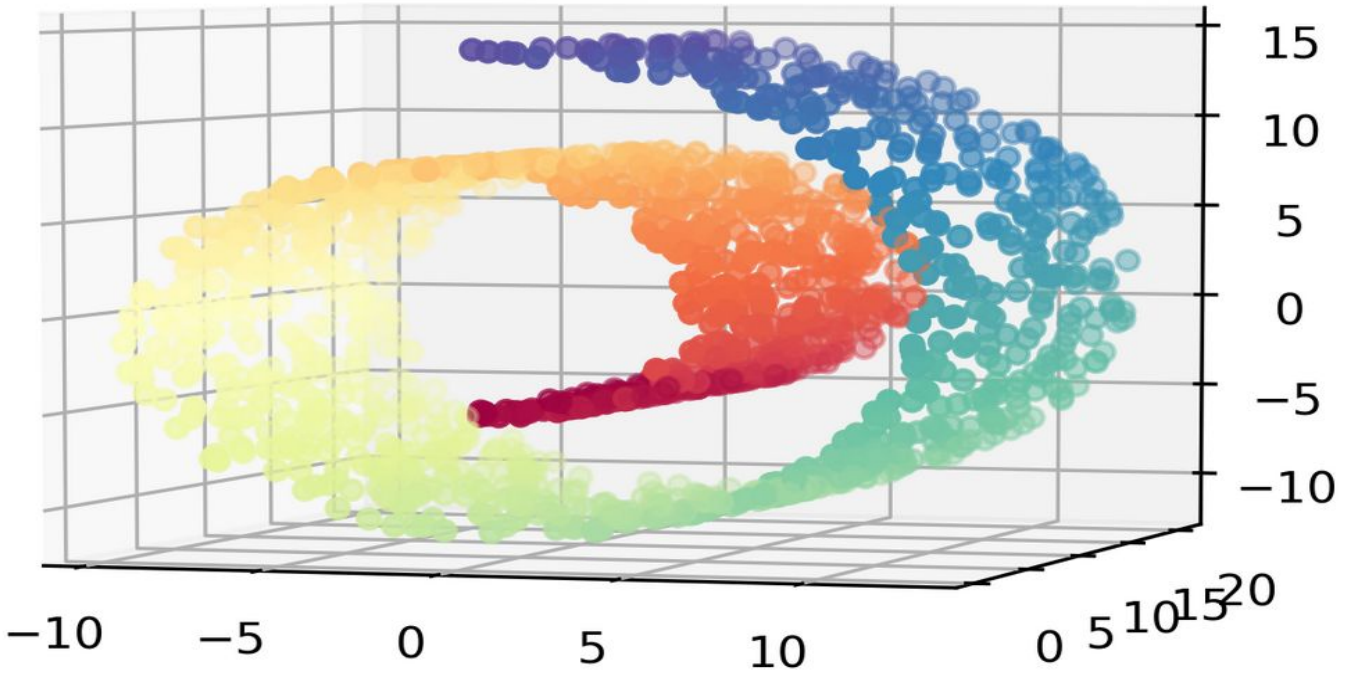

Data:
Projection:
Reconstruction:
mean
First
eigenvector
Second
eigenvector

PCA is Unsupervised





Manifold Learning



Manifold Learning

- When you think of a manifold, imagining a sheet of paper. In the parlance of manifold learning, we can think of this sheet as a two-dimensional manifold embedded in three-dimensional space.
- Manifold learning algorithms would seek to learn about the fundamental two-dimensional nature of the paper, even as it is contorted to fill the three-dimensional space.
- Axes don’t correspond to anything in the input space.
- Often can’t transform new data.

LDA and QDA classifiers

- Both LDA and QDA can be derived from simple probabilistic models which model the class conditional distribution of the data for each class k.
- In the case of LDA, the Gaussians for each class are assumed to share the same covariance matrix. In the case of QDA, there are no assumptions on the covariance matrices of the Gaussians, leading to quadratic decision surfaces.
- If in the QDA model one assumes that the covariance matrices are diagonal, then the inputs are assumed to be conditionally independent in each class, and the resulting classifier is equivalent to the **Gaussian Naive Bayes classifier.**

Linear Discriminant Analysis

$$P(y=k|X) = \frac{P(X|y=k)P(y=k)}{P(X)} = \frac{P(X|y=k)P(y=k)}{\sum_l P(X|y=l) \cdot P(y=l)}$$

$$p(X|y=k) = \frac{1}{(2\pi)^n |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t \Sigma^{-1}(X-\mu_k)\right)$$

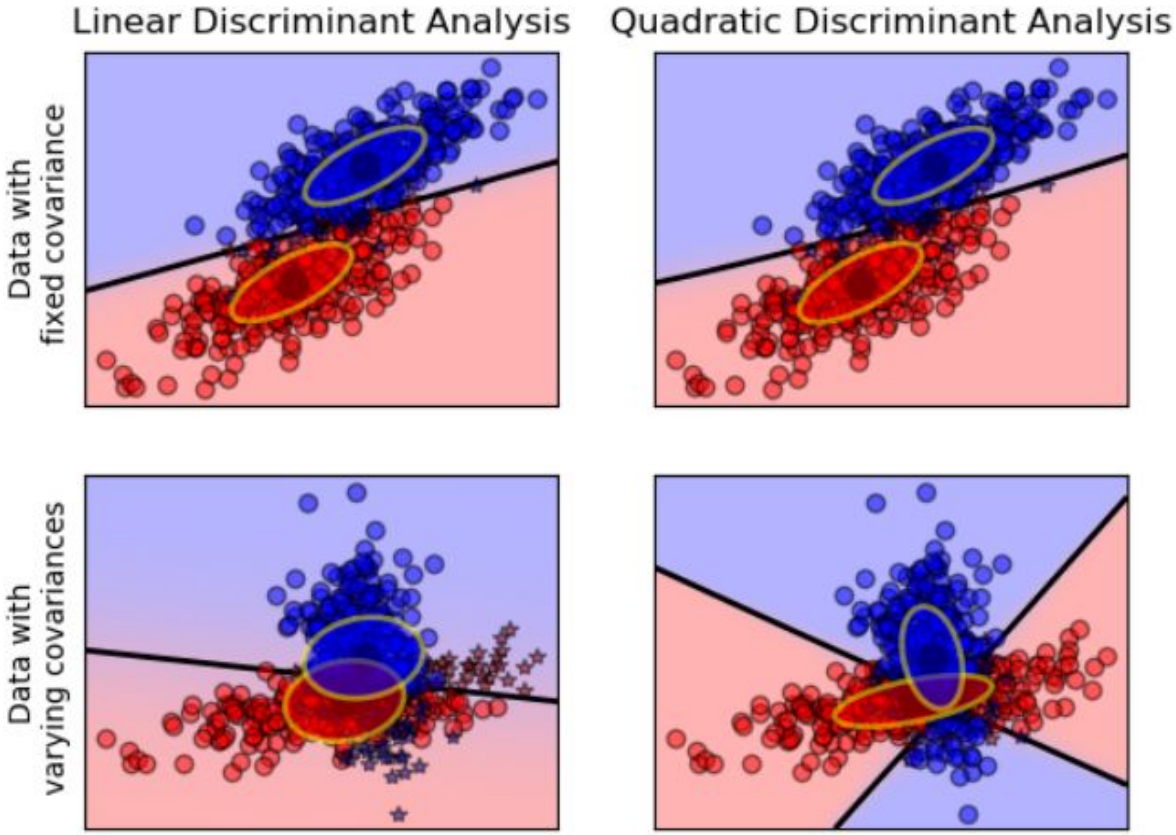
$$\log\left(\frac{P(y=k|X)}{P(y=l|X)}\right) = 0 \Leftrightarrow (\mu_k - \mu_l)\Sigma^{-1}X = \frac{1}{2}(\mu_k^t \Sigma^{-1}\mu_k - \mu_l^t \Sigma^{-1}\mu_l)$$

Quadratic Discriminant Analysis

$$p(X|y=k) = \frac{1}{(2\pi)^n |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t \Sigma_k^{-1}(X-\mu_k)\right)$$

https://scikit-learn.org/stable/modules/lda_qda.html

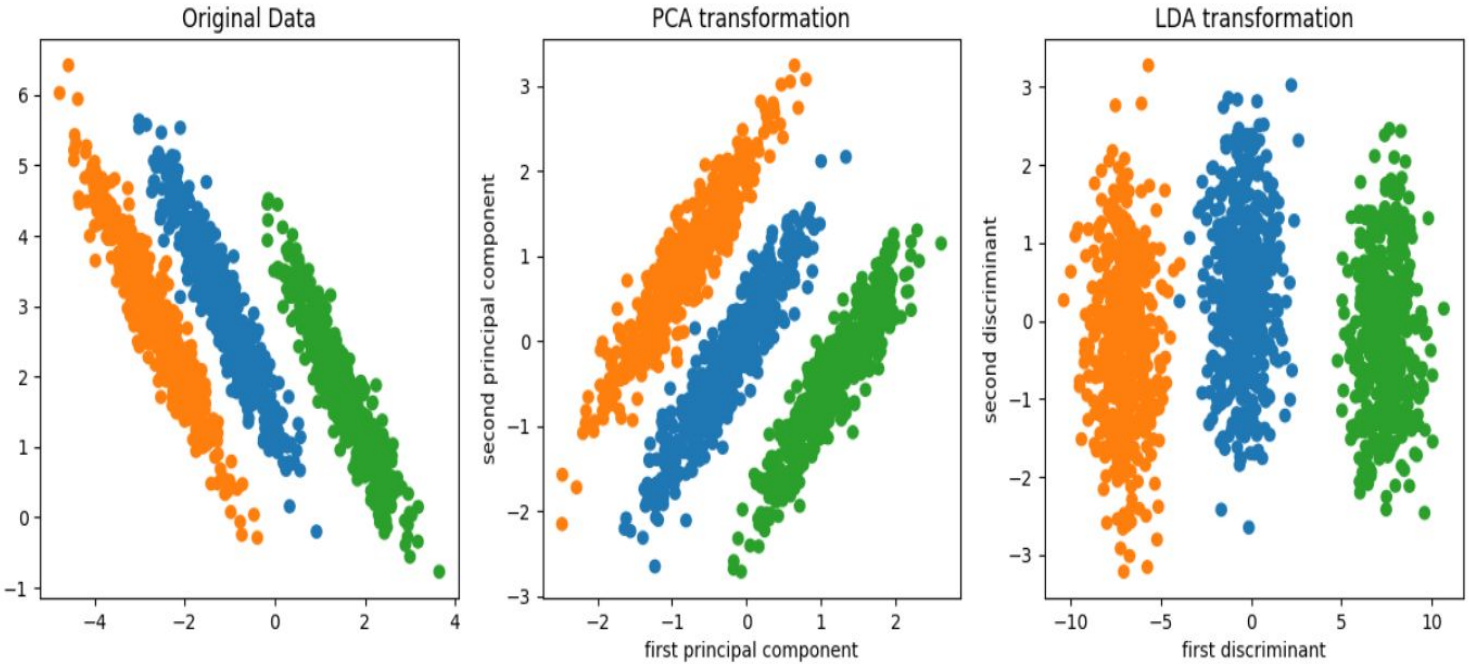
LDA VS QDA


Linear Discriminant Analysis
Quadratic Discriminant Analysis
Data with fixed covariance
Data with varying covariances

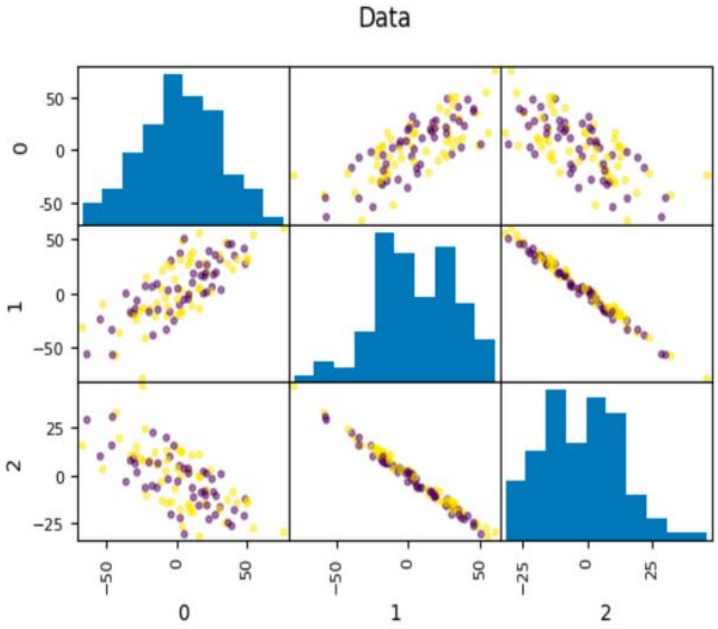
Discriminants and PCA

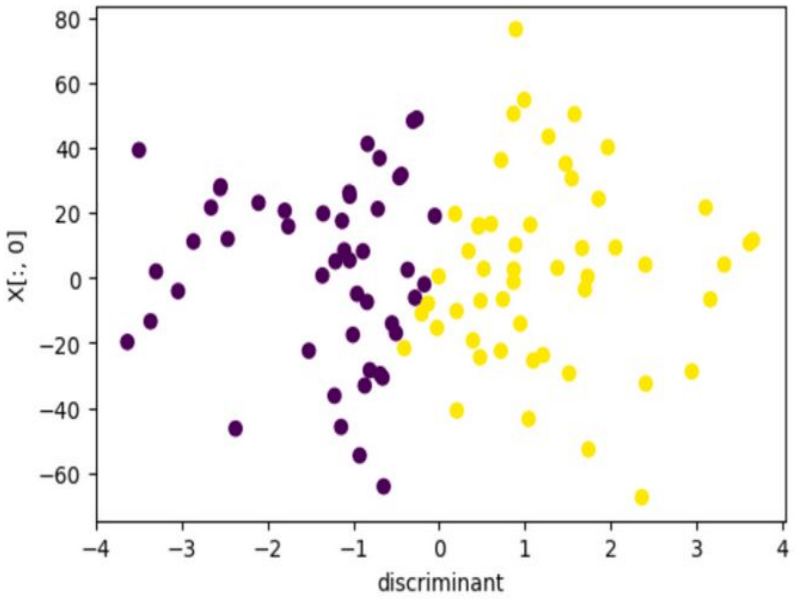
- Both fit Gaussian model
- PCA for the whole data
- LDA multiple Gaussians with shared covariance
- Can use LDA to transform space!
- At most as many components as there are classes (needs between class variance).

PCA vs Linear Discriminants



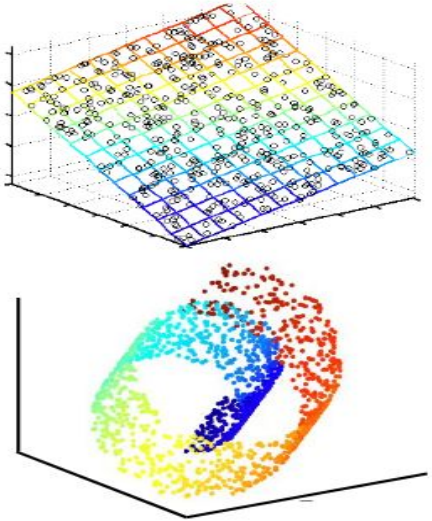
Data where PCA failed





Non-linear methods

- Linear
  - **Principal Component Analysis (PCA)**
  - Factor Analysis
  - Independent Component Analysis (ICA)
- Nonlinear
  - **Laplacian Eigenmaps**
  - ISOMAP
  - Local Linear Embedding (LLE)



Slide from Aarti Singh

Algorithms in sklearn

- Spectral embedding (Laplacian Eigenmaps). Uses eigenvalues of graph Laplacian→ Spectral Embedding
- Locally Linear Embedding→ Locally Linear Embedding
- Isomap. Kernel PCA on manifold→ Isomap
- t-SNE (t-distributed stochastic neighbor embedding) → t-SNE
- KernelPCA. Eigenvalues of kernel matrix→ Kernel PCA