University of Pittsburgh

ECE 2195: Special Topics – Computers
Machine Learning

Dimensionality Reduction

Mai Abdelhakim, PhD

Assistant Professor of ECE

Swanson School of Engineering

University of Pittsburgh

maia@pitt.edu

Dimensionality Reduction

Finding smaller representation for input

- Why needed?
  - **Preprocessing for unsupervised learning**: reduce overfitting, less complex
  - Can also be used for data **visualization**
    - Observations with p feature, if we want to **visualize** the observations
      - We can see pair-plots: p(p-1)/2 plots! ➔ difficult to visualize
      - Instead, **find low dimensional representation that captures as much information as possible**
- Unsupervised Dimensionality Reduction approaches: No labels used
  - Ex. PCA, t-SNE (mainly visualization)
- Supervised Dimensionality Reduction Approaches
  - Ex. LDA

Principal Component Analysis (PCA)

unsupervised technique

- Principal Components: **Smaller number of representative features** that **explain most of the variability in the original data**

- Used as preprocessing for supervised learning and for visualization

  - After we get the principal components, we can use them instead of the original features for supervised learning.

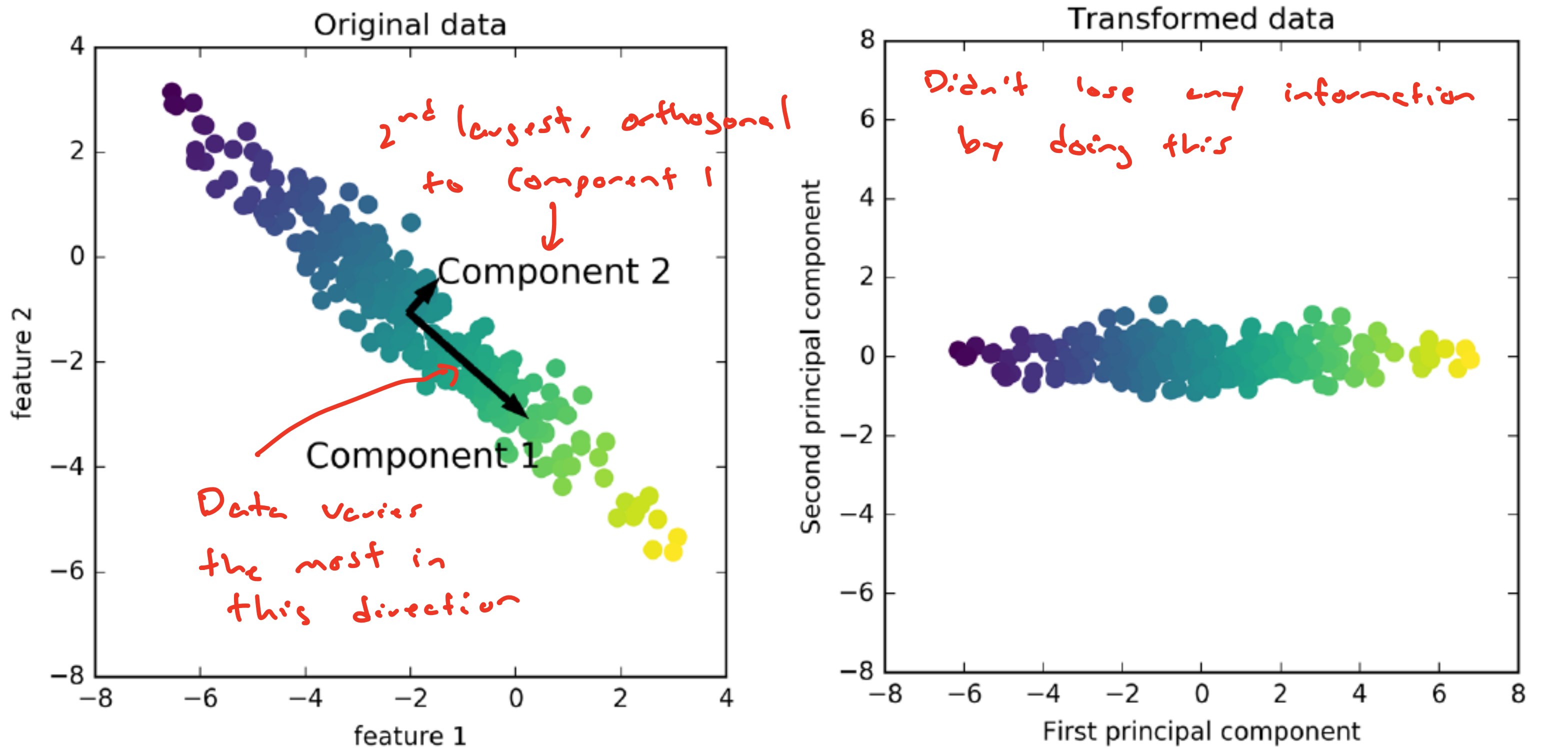
PCA: Characteristics of Principle Components

- PCA produces a low-dimensional representation of a dataset
  - It finds a sequence **of linear combinations of the original features** that have **maximal variance**, and are **mutually uncorrelated**

variability in feature space is information → how many features are actually necessary to explain most of the variability in the data

Example: p=2 -> lower to p=1

The principal component loading vectors are the direction in the feature space where the data varies the most

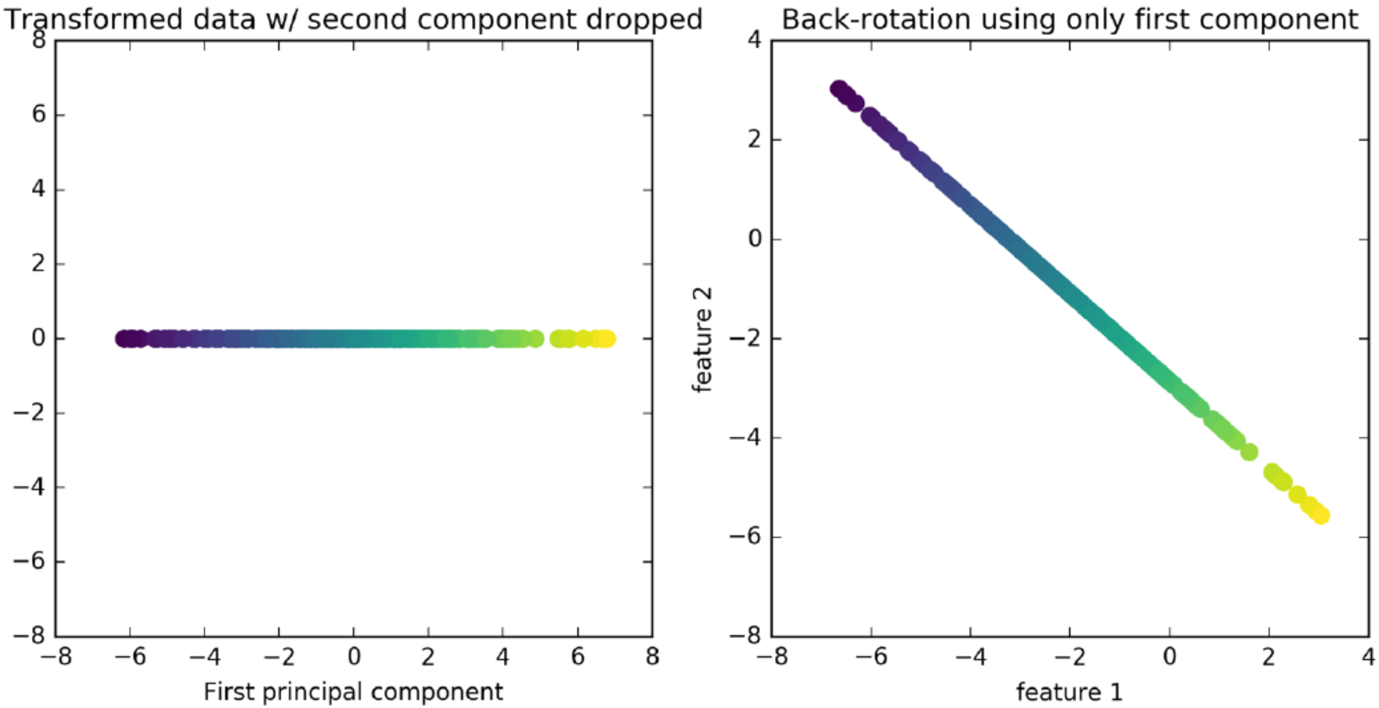


Muller et al., Introduction to Machine Learning with Python

Example: p=2 … cont..

2D → 1D Feature space

Most of variance in data remains



Muller et al., Introduction to Machine Learning with Python

PCA – first component

- If original features are X_1, X_2,…,Xp , then **the first principal component** (Z_1) is the normalized **linear combination of features** that has the **largest variance**:

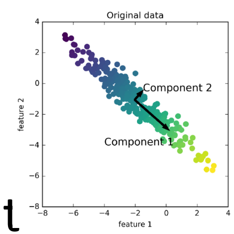
$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p$$

Coefficients $\phi_{11}, .. \phi_{p1}$ are called the loading of the principal component Z_1

- Normalized means: $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
  - Without this normalization the variance can be large due to $\phi's$ and not due to data (X)
- The **first principal component loading vector**

$$\phi_1 = (\phi_{11}, \phi_{21}, .. \phi_{p1})^T$$

  - A unit vector in the direction of the principal component



PCA – second component

- The second principal component (Z_2) has the second **maximal variance** out of **all linear combinations** of ($X_{1.}.X_p$) that are **uncorrelated** with Z_1

  - The second principal component loading vector

$$\phi_2 = (\phi_{12}, \phi_{22}, .. \phi_{p2})^T$$

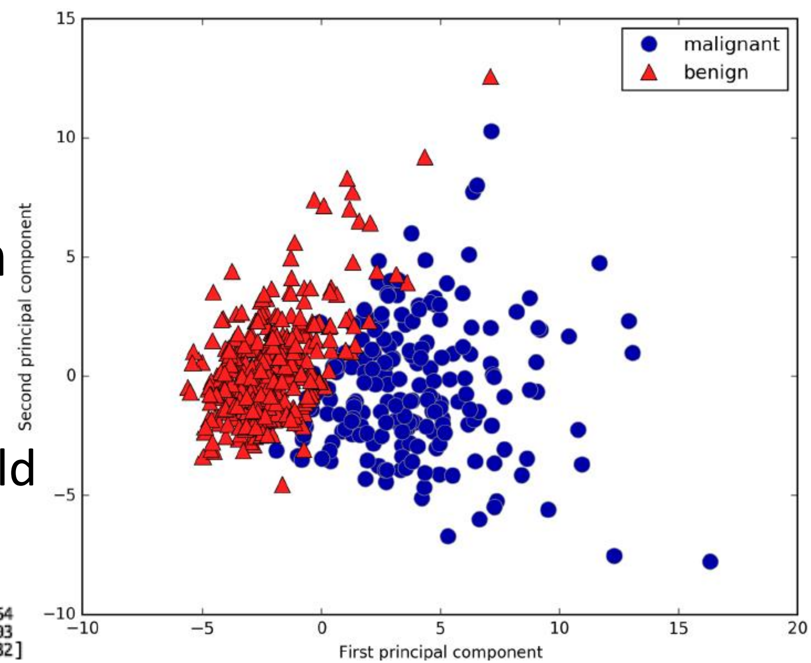
- And so on …
- we can have up to M principal components (vectors), $M \leq p$

Example: Breast Cancer Data Set

With a NN and 2 PCs, there are only 2 neurons in input layer

- Not very easy to visualize histogram of 30 features in cancer dataset
- We can use PCA & only 2 derived features
- Data is well separated with only 2 principal components!
  - Simple linear classifier would do well

Muller et al., Introduction to Machine Learning with Python



$\phi_{11}, .. \phi_{p1}$

```
PCA components:
[[ 0.219  0.104  0.228  0.221  0.143  0.239  0.258  0.261  0.138  0.064
   0.206  0.017  0.211  0.203  0.015  0.17   0.154  0.183  0.042  0.103
   0.228  0.104  0.237  0.225  0.128  0.21   0.229  0.251  0.123  0.132]
```

30 × 1

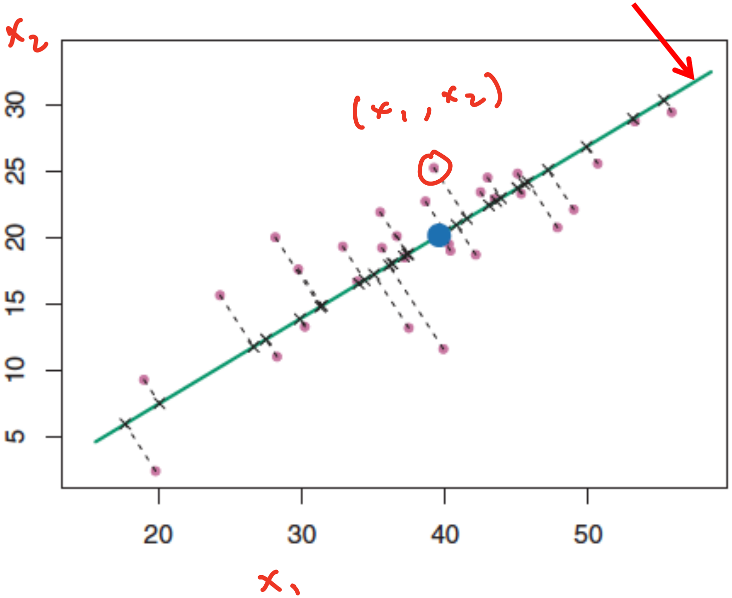
$\phi_{12}, .. \phi_{p2}$

```
 [-0.234 -0.06  -0.215 -0.231  0.186  0.152  0.06  -0.035  0.19   0.367
  -0.106  0.09  -0.089 -0.152  0.204  0.233  0.197  0.13   0.184  0.28
  -0.22  -0.045 -0.2   -0.219  0.172  0.144  0.098 -0.008  0.142  0.275]]
```

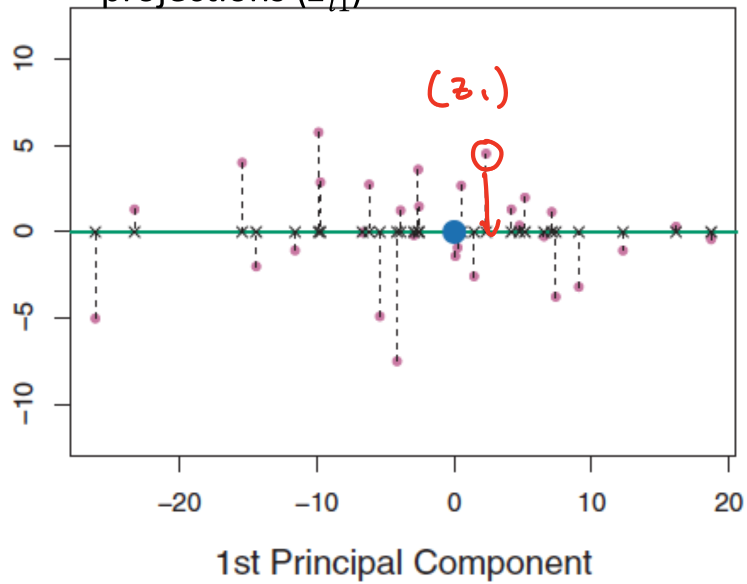
Elements in the 2 vectors are loadings of the 2 principal components on the 30 features

Data is Represented by Principal Components through projections

Direction of the first principal component, represented by: $\phi_1 = (\phi_{11}, \phi_{21}, .. \phi_{p1})^T$



Each sample is represented in the new low dimensional features space through projections (z_{i1})



Data is Represented by Principle Components

- Each **observation** is **represented by principal components** through **projections** on each component
  - Projection of training sample i on principal component (Z_j) is z_{ij}

- The **projection** (also called **score**) of observation i, $i = 1,2..n$, on first principal component is

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \ldots + \phi_{p1}x_{ip}$$

$$= x_i \cdot \phi_1$$

- The projection of observation i, $i = 1,2..n$, on second principal component is

$$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \ldots + \phi_{p2}x_{ip} \quad = x_i \cdot \phi_2$$

Coefficients $\phi_{1j}, .. \phi_{pj}$ are the loading of the principal component Z_j

Problem Formulation for First Principal Component

$Var(z_1) = \frac{1}{n} \sum_{i=1}^{n} (z_{i1})^2$

- In PCA, X is generally normalized to zero mean
  - PCA is sensitive to feature scaling
  - **Principle components also have zero mean**

- Finding the first principal component: **find loadings the maximizes the variance of Z_1**

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \frac{1}{n} \sum_{i=1}^{n} \Bigg(\overbrace{\sum_{j=1}^{p} \phi_{j1} x_{ij}}^{z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \ldots + \phi_{p1}x_{ip}} \Bigg)^2 \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

data points features

  - Similarly, problem can be formulated to get second principal component

Variance of Z_1

$$\mathrm{Var}(Z_1) = \frac{1}{n} \sum_{i=1}^{n} \left(Z_{i1}\right)^2$$

$$= \frac{1}{n} [X \phi_1]^T [X \phi_1]$$

$$= \frac{1}{n} \phi_1^T X^T X \phi_1$$

$$= \phi_1^T \frac{X^T X}{n} \phi_1$$

$$\frac{x^2}{n} = \frac{(x-0)^2}{n}$$

Since mean is 0, this is the covariance matrix for the features

$$= \phi_1^T \Sigma \phi_1$$

Covariance of features

max $\mathrm{var}(Z_1)$ such that $\phi_1^T \phi_1 = 1$

LG multiplier

$$J = \phi_1^T \Sigma \phi_1 - \lambda(\phi_1^T \phi_1 - 1)$$

$$\frac{\partial J}{\partial \phi_1} = 2\Sigma\phi_1 - 2\lambda\phi_1 = 0$$

scalar

$$\Sigma\phi_1 = \lambda\phi_1$$

$$[\phi_{11} \quad \phi_{21} \quad \cdots \quad \phi_{p1}]$$

There is a vector that is invariant to rotation when multiplied by covariance matrix Σ

ϕ_1 is an eigenvector

λ is an eigenvalue

Recall eigenvalue decomposition

- Matrix Σ (of dimension pxp) has p eigenvalues ($\lambda's$) and p eigenvectos (ϕ)
  symmetric
  - Eigenvectors are directions of variability
  - Eigenvalues is the magnitude of this variability

$$\Sigma = \phi^T D_\lambda \phi,$$

$$\begin{bmatrix} | & | & | \\ u_1 & u_2 & u_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} - & u_1 & - \\ - & u_2 & - \\ - & u_3 & - \end{bmatrix}$$

ϕ Matrix where every column is an eignvector

D_λ is a diagonal matrix wither diagonal elements are eigenvalues of the corresponding eigenvetors

Principal components are the eigenvectors of the covariance matrix

→ Eigenvectors are orthonormal

Eigenvalues are magnitude of this variability (ordered from largest to smallest)

Recall eigenvalue decomposition

- Matrix Σ (of dimension pxp) has p eigenvalues ($\lambda's$) and p eigenvectos (ϕ)
  - Eigenvectors are directions of variability and magnitude of this variability is eigenvalues
- To solve eigenvalue problem
  - $\text{Det}(\Sigma - I.\lambda) = 0$
  - $\Sigma\, \phi_i = \lambda_i \phi_i$, for all $i = 1,2..p$

$$\det\left(\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$

$$\begin{vmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{vmatrix} = 0$$

$$(3-\lambda)^2 - 1 = 0$$

$$\lambda^2 - 6\lambda + 8 = 0$$

$$\lambda_1 = 4, \ \lambda_2 = 2$$

Example:

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

$$\begin{aligned} \det(A - \lambda I) &= \begin{vmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{vmatrix} \\ &= (3-\lambda)^2 - 1 \\ &= \lambda^2 - 6\lambda + 8. \end{aligned} \longrightarrow \lambda_1 = 4 \text{ and } \lambda_2 = 2.$$

$$\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{12} \end{bmatrix} = 4 \begin{bmatrix} \phi_{11} \\ \phi_{12} \end{bmatrix}$$

$$3\phi_{11} + \phi_{12} = 4\phi_{11}$$

$$\phi_{11} + 3\phi_{12} = 4\phi_{12} \rightarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\phi_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \phi_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

First principal component

Problem Formulation for First Principal Component

pxp covariance matrix: p eigenvectors and p eigenvalues

- In PCA, X is generally normalized to zero mean
  - PCA is sensitive to feature scaling
  - **Principle components also have zero mean**

- Finding the first principal component: **find loadings the maximizes the variance of Z_1**

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \; \frac{1}{n}\sum_{i=1}^{n}\left(\overbrace{\sum_{j=1}^{p}\phi_{j1}x_{ij}}^{Z_{i1}}\right)^2 \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1$$

  - Similarly, problem can be formulated to get second principal component

- The **principal components** are the **eigenvectors** corresponding to the **largest eigenvalues** of the pxp **covariance matrix** Σ **of features X**

Transform the data

- Then we transform each observation to M dimensional space
  - Projections discussed earlier

p features → M features

- Matrix form:

$$[z_{i1}, z_{i2} \dots z_{iM}] = [x_{i1}, x_{i2} \dots x_{ip}]\ \mathrm{W}$$

n × M　　n × p　　p × M

  - Column j of the W matric is loading vector $\phi_j = (\phi_{1j}, \phi_{2j}, .. \phi_{pj})^T$
    - X is n x p matrix
    - W is p x M matrix: M eigenvectors (principle components) in direction of largest variance
    - Z is n x M matrix

i.e., how many principal components do you want?

p-dimensional feature space ➔ M dimensional space

$$W = \begin{bmatrix} | & | & & | \\ \phi_1 & \phi_2 & \cdots & \phi_M \\ | & | & & | \end{bmatrix}$$

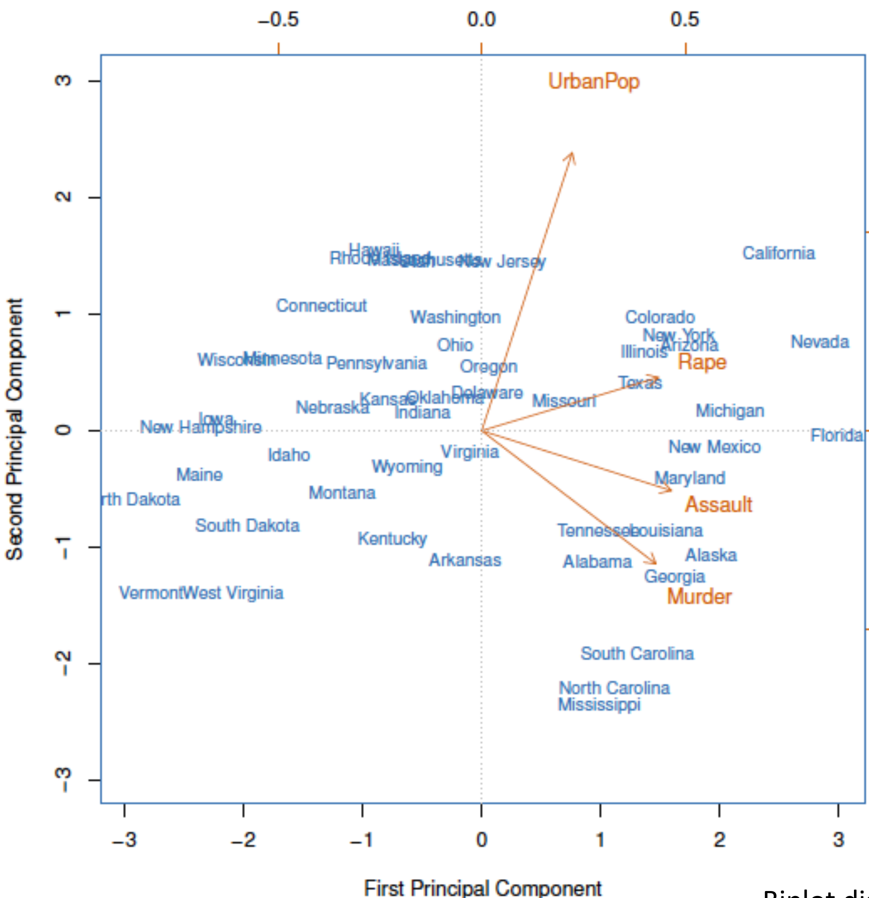
Example: USAarrests data

- **For 50 states** in the USA, the set contains:
  - The **number of arrests** per 100,000 residents for each of **three crimes**: Assault, Murder, Rape.
  - It also contains **UrbanPop** feature (the percent of the population in each state living in urban areas).

- Then, number of training examples is n=50, number of features is p=4 (Assault, Murder, Rape, UrbanPop)

- Apply PCA, get two principal components (PC1, PC2)

- Ref: Chapter 10, Ghareth et al.



Numbers in table represent loadings/weights (also in top and right axes)

	PC1 (z_1)	PC2 (z_2)
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

Equal weights on all crimes, less weigh on population

High weight on population

PC1: roughly measures overall rate of serious crimes, and PC2 roughly measures population (level of urbanization)

Biplot displays scores and the loading

Proportion of Variance Explained

- How much of the data is contained in the first few principle components?
  - We select just few principle components to represent the data
  - How much of the variance in the data in contained in the first few principle components

- We need to know the proportion of variance explained (PVE) by each principle component?

Proportion of Variance Explained

$(x_{ij} - \mu)^2$

- Total variance in the data (all p features) is

$$Var(X) = \sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

  - Note data is centered around zero mean so we took out the mean from the equation

- The **variance explained by a principle component Z_m** is:

$$\mathrm{Var}(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2$$

- You can show **that if we take all possible principle components (all eigenvectors, M=p) then the total variance is the same (all variance is explained)** Not losing any information

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{m=1}^{M} \mathrm{Var}(Z_m)$$

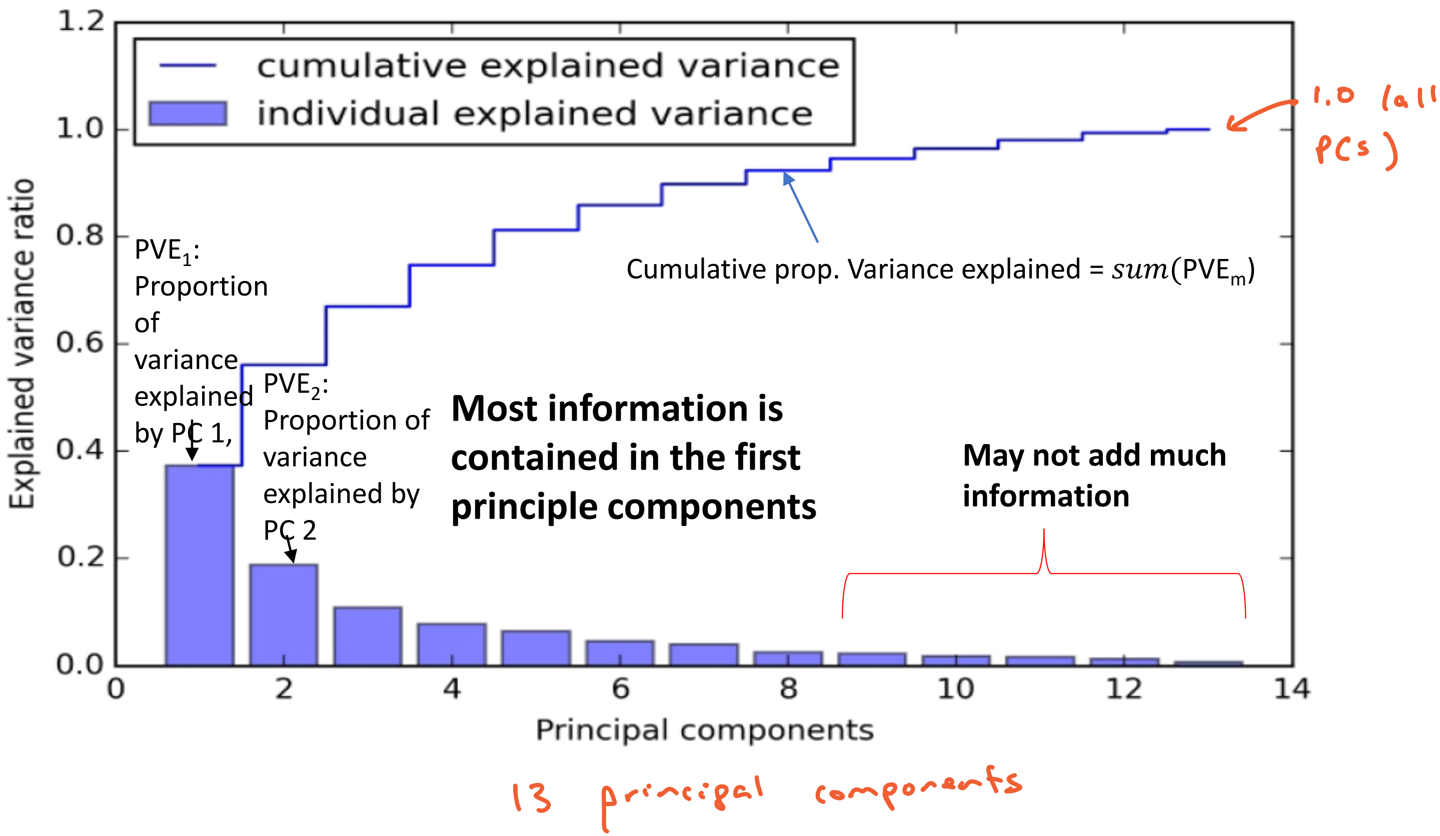
Proportion of Variance Explained

- The **proportion of variance explained (PVE)** of the mth principle component is **the ratio between variance of Z_m to the total variance of X**
  - Positive quantity between 0 and 1

$$\mathrm{PVE}_m = \mathrm{Var}(Z_m)/\mathrm{Var}(X) = \frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2}$$

  - Sum over all PVE's is equal to 1 if we take all eigenvectors (total variance)
  - However, we want to reduce the dimension so we only take few principle components (M<p)

Typical Pattern for PVE



Choice of the Number of Principle Components

- One way to choose the number of principle components is to find number of components after which only slight information is gained
  - from previous graph: 6 components explained more than 80% of the variance, which can be sufficient for an application
- **PCA** can be used to reduce the dimension for **supervised learning** methods
  - in this case **cross-validation** can be used to choose the number of principle components
- The number of PC to use depends on the application and dataset

Python

from sklearn.decomposition import PCA

- Define PCA components using scaled training data:
  N_components=2 # Define the number of principle components
  Data_pca = **PCA**(n_components=N_components)**.fit(X_train_scaled)**

- Get variance explained by each of the PCA components
  print('Explained variance,' , **Data_pca**.**explained_variance_ratio_**)

- Transform data into the defined principal components
  X_train_pca = **Data_pca** **.transform**(X_train_scaled)
  X_test_pca = **Data_pca** **.transform**(X_test_scaled)

- More: http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

covariance matrix of data

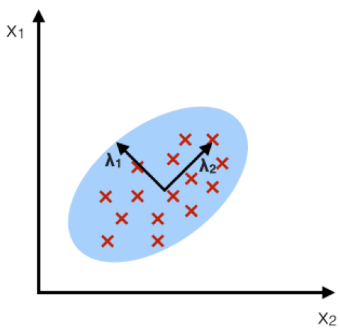
PCA through eigenvalue decomposition:

eig_values_cov, eig_vectors_cov = numpy.linalg.eig(cov_mat)

LDA for Dimensionality Reduction

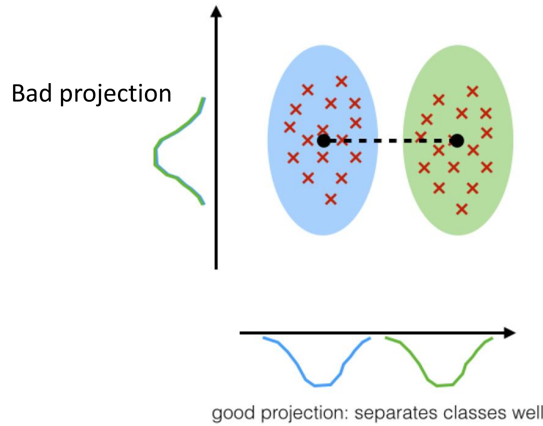
have not used labels
at all

PCA:
component axes that
maximize the variance



have to know the labels
(supervised)

LDA:
maximizing the component
axes for class-separation



2D

1D

Ref: Sebastian Raschka

LDA for Dimensionality Reduction –With Known Class Labels

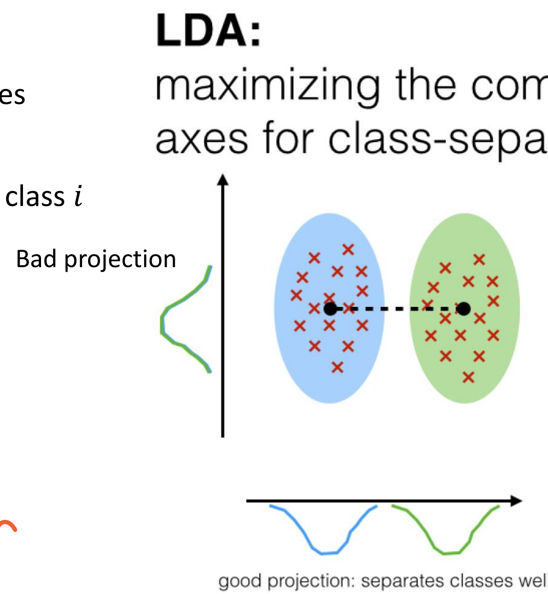
- Within-class scatter matrix
  - Assume c classes $i = \{1,2,..,c\}$

$$S_W = \sum_{i=1}^{c} S_i$$ Sum all scatter matrices over all classes

$$S_i = \sum_{\boldsymbol{x} \in D_i}^{n} (\boldsymbol{x} - \boldsymbol{m}_i)\,(\boldsymbol{x} - \boldsymbol{m}_i)^T$$ Scatter matrix for every class i

$$\boldsymbol{m}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in D_i}^{n} \boldsymbol{x}_k$$ mean of samples in class i

D_i = data from class i



- Between class-scatter matrix

class mean

overall mean

$$S_B = \sum_{i=1}^{c} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$$

  - m is the overall mean, m_i is the mean of class i

Maximize S_B

Minimize S_W

- In LDA, we solve the eigenvalue problem of $S_W^{-1} S_B$

Ref: Sebastian Raschka

Manifold Learning with t-SNE

unsupervised

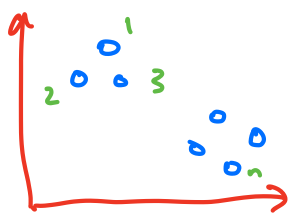
- t-Distributed Stochastic Neighbor Embedding (**t-SNE**): **non-linear** dimensionality reduction technique

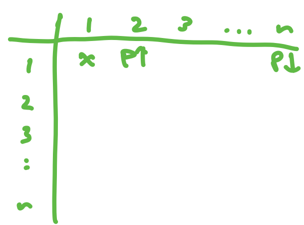
- Used for data **visualization**
    - Typically used to generate 2 new features (visualized in 2D plots)
    - Rarely used for supervised learning

Find 1D space that maintains similarity in higher space

- **The transformation depends on how points are in the original feature space**
    - Tries to make points that are close in the original feature space closer in new space, and
    - points that are far apart in the original feature space farther apart in the new space.
        - (Uses joint probabilities and Kullback-Leibler divergence)

Data in high dim space → 2D/3D space for visualization





Example: t-SNE Applied to Handwritten Digit Dataset

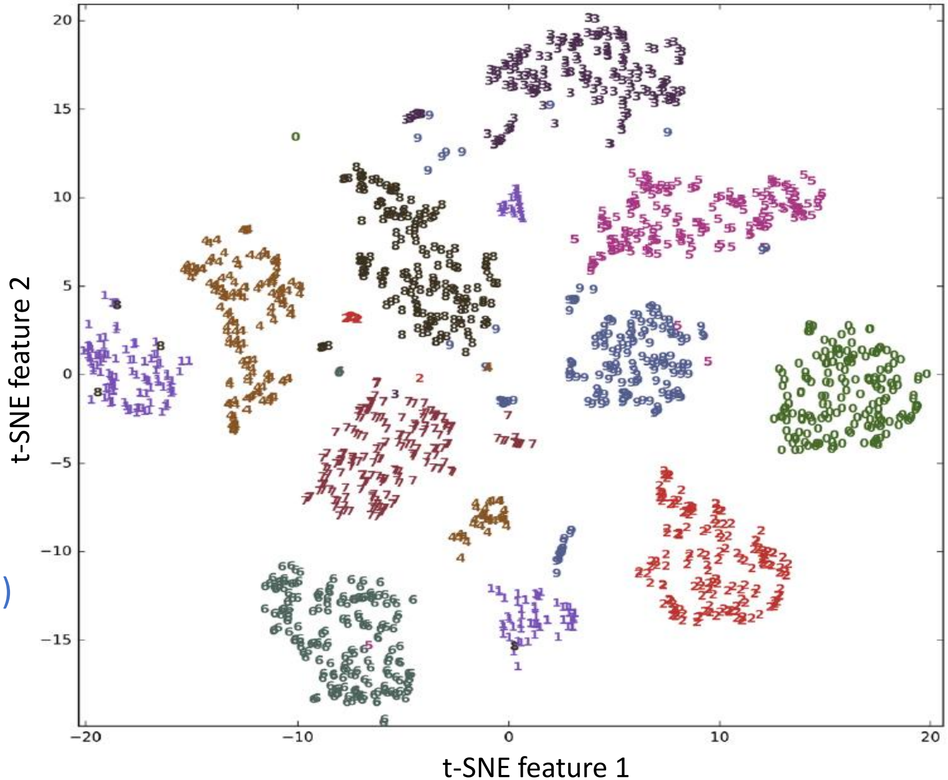
- Each observation is colored by its class (0-9)
- # Original feature is 64 (8x8), gray scale values of pixels
- All classes are clearly separated using 2 derived features of t-SNE

64 features →

2 features

From sklearn.manifold import TSNE
Tsne_data=TSNE().fit_transform(digits.data)

http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html



Example: PCA Applied to Handwritten Digit Dataset

Using two principle components on digits data, classes are not well-separated

PCA - linear

tSNE - nonlinear