Applied Statistical Methods

Principal Components Analysis

Xuemao Zhang
East Stroudsburg University

April 17, 2023

Outline

- Unsupervised Learning
- Principal Components Analysis

Unsupervised Learning

- Most of this course focuses on supervised learning methods such as regression and classification.
- In that setting we observe both a set of features $X_1, X_2, \ldots, X_p$ for each object, as well as a response or outcome variable Y. The goal is then to predict Y using $X_1, X_2, \ldots, X_p$.
- Here we instead focus on unsupervised learning, we where observe only the features $X_1, X_2, \ldots, X_p$. We are not interested in prediction, because we do not have an associated response variable Y.
- The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?

Unsupervised Learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.
- But techniques for unsupervised learning are of growing importance in a number of fields:
  - subgroups of breast cancer patients grouped by their gene expression measurements,
  - groups of shoppers characterized by their browsing and purchase histories,
  - movies grouped by the ratings assigned by movie viewers.

Why Dimension Reduction?

- Recall that PCA is a dimension reduction method

For Big-Data:

- Data visualization becomes very difficult! (Cannot draw 2D scatterplots between all pairs of features).
- Big-Data often has a high degrees of redundancy. (i.e. correlation among features).
- Many features may be uninformative for the particular problem under study (noise features).
- Dimension reduction ideally allows us retain information on most important features of the data, while reducing noise and simplifying visualization & analysis.

What is Dimension Reduction?

- Map the data into a new low-dimensional space where important characteristics of the data are preserved.
- The new space often gives a (linear or non-linear) transformation of the original data.
- Visualization and analysis (clustering/prediction/...) is then performed in the new space.
- In many cases, (especially for non-linear transformations) interpretation becomes difficult.

Principal Components Analysis

Set-up:

- Data matrix: $\mathbf{X}_{np}$, n observations and p features.

Idea:

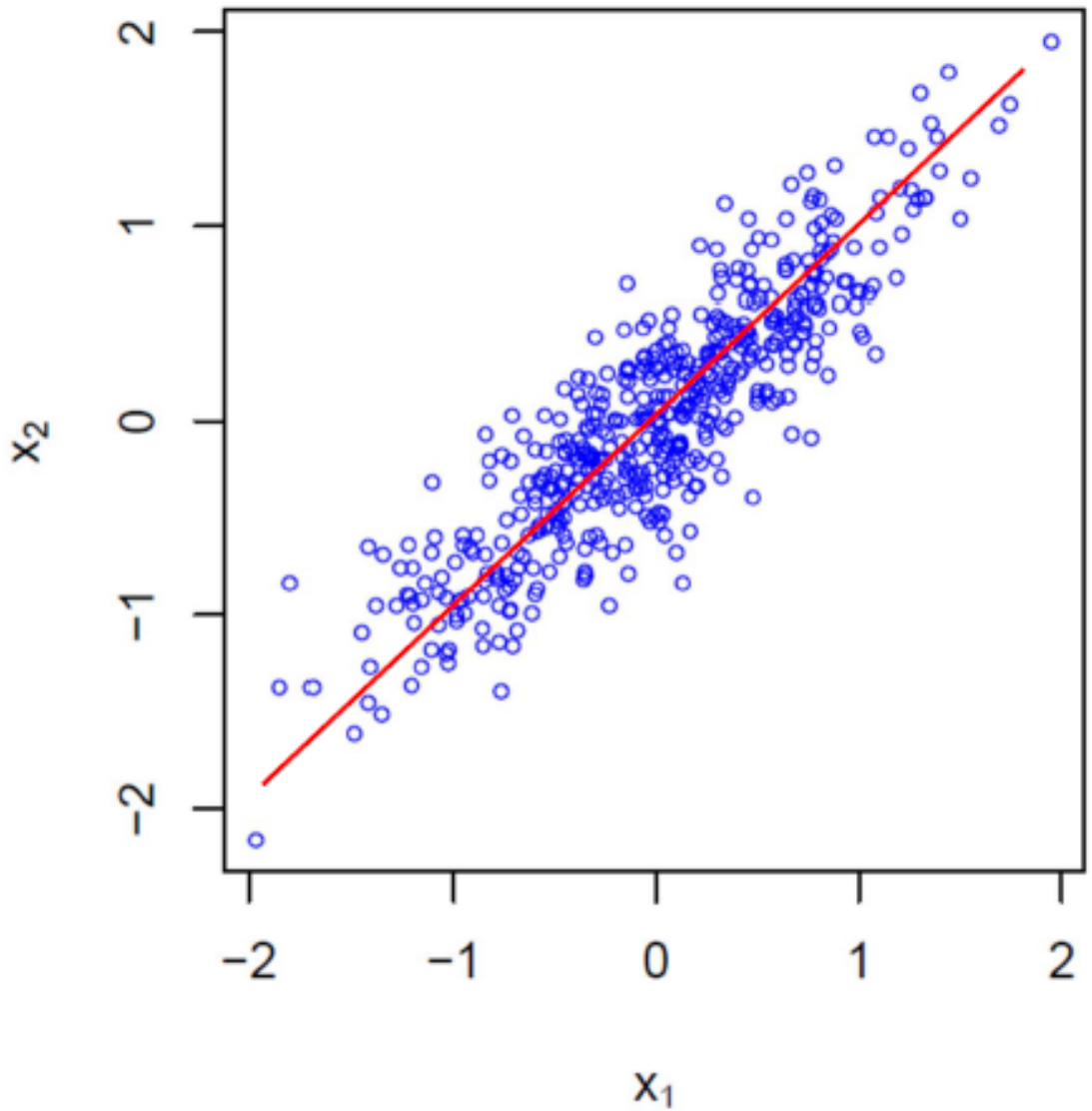
- Not all p features are needed (much redundant info).
- Find low-dimensional representations that capture most of the variation in the data.

Uses:

- Ubiquitously used - Dimension reduction, data visualization, pattern recognition, exploratory analysis, etc.
- "Best" linear dimension reduction possible.

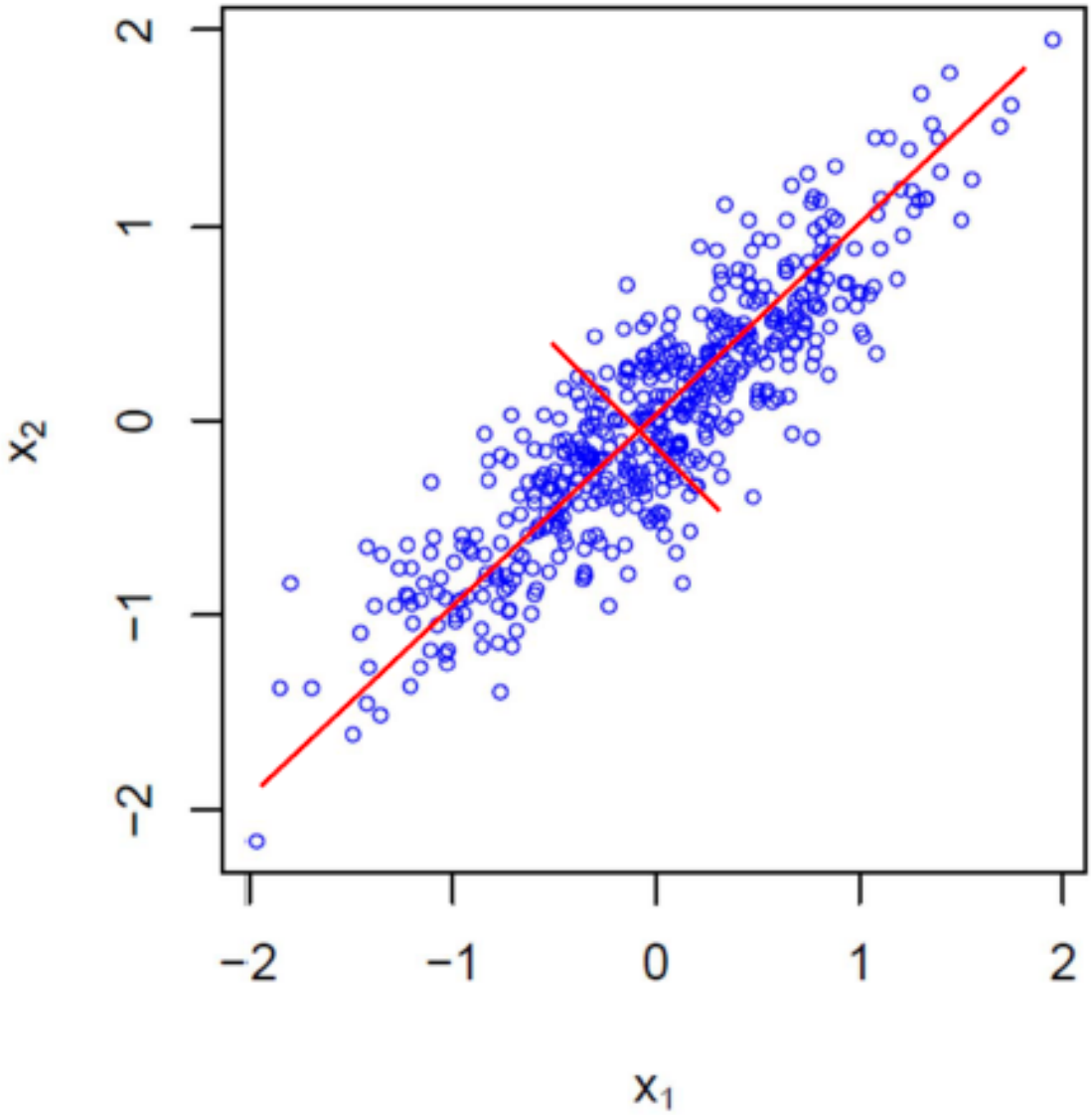
PCA - Main Idea

- Question: What is a good 1D representation of the data?
  - Find line that maximizes the variance of the data projected onto the line:



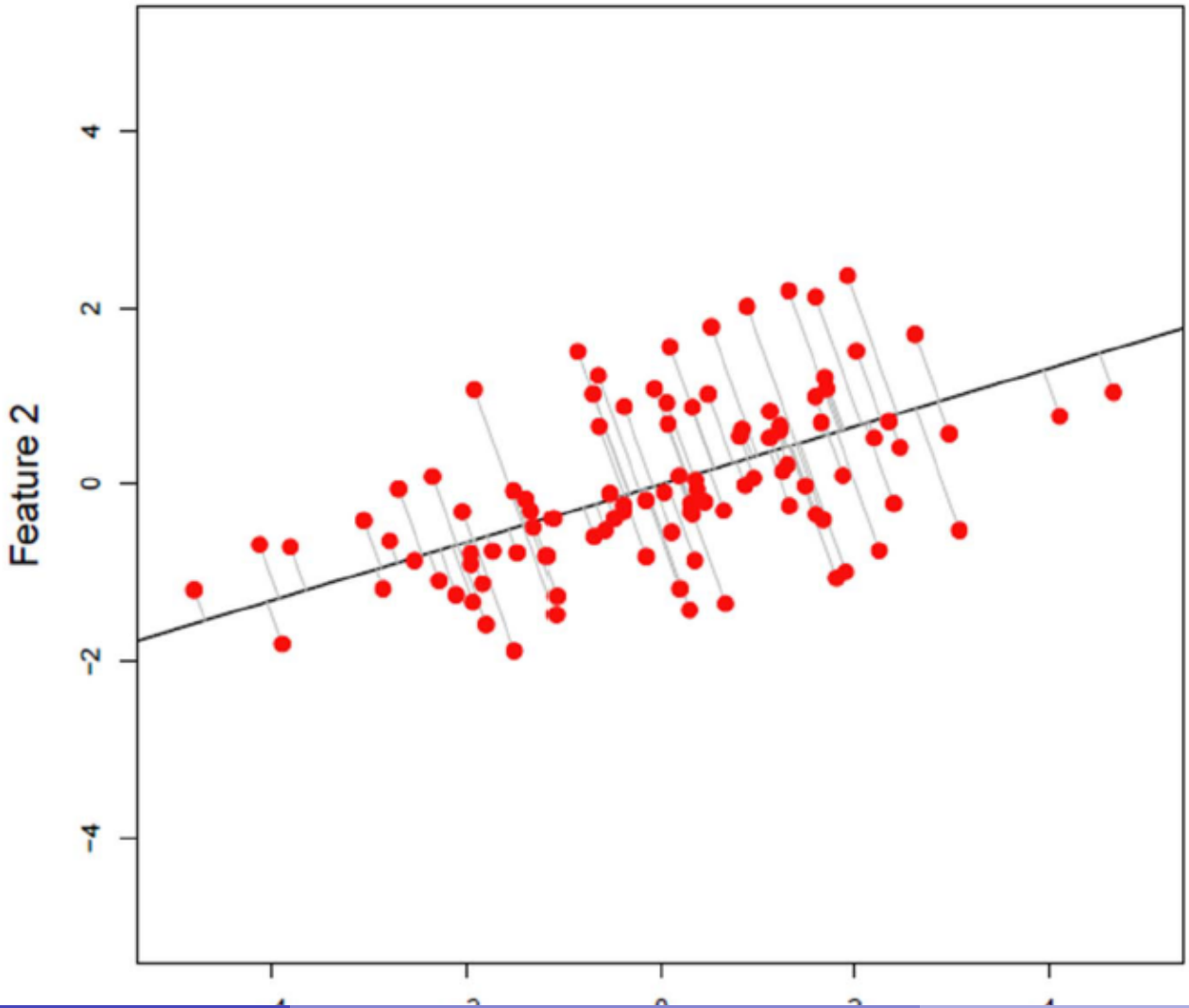
PCA - Main Idea

- Subsequent components orthogonal (perpendicular).



PCA - Main Idea

- PCA minimizes orthogonal projection onto line: $Z = v_1 x_1 + v_2 x_2$.
- Slope of line $= v_2/v_1$ (if features centered).
- Note: Not same as OLS which minimizes projection of y onto x!



PCA

- We have derived the PCA in the section of PCR. For the sample version, we simply replace $\mathbf{\Sigma}$ with its estimate $\mathbf{X}^{'}\mathbf{X}/(n-1)$. The kth PC is obtained by
  - maximize$_{\mathbf{v}_k}\mathbf{v}_k^{'}\mathbf{X}^{'}\mathbf{X}\mathbf{v}_k$ subject to $\|\mathbf{v}_k\|_2 = 1$ and $\mathbf{v}_k^{'}\mathbf{v}_j = 0\ \forall\ j < k$
- The first PC is $\boldsymbol{v}_1^{'}\mathbf{x} = v_{11}\mathbf{x}_1 + \cdots + v_{1p}\mathbf{x}_p = \sum_{j=1}^{p} v_{1j}\mathbf{x}_j$, where $\boldsymbol{v}_1$ is the eigenvector correspoding to the largest eigenvalue of $\mathbf{X}^{'}\mathbf{X}$..
- The second PC is $\boldsymbol{v}_2^{'}\mathbf{x} = v_{21}\mathbf{x}_1 + \cdots + v_{2p}\mathbf{x}_p = \sum_{j=1}^{p} v_{2j}\mathbf{x}_j$, where $\boldsymbol{v}_2$ is the eigenvector correspoding to the second largest eigenvalue of $\mathbf{X}^{'}\mathbf{X}$.
- and so on.
  - The first PC is the linear combination of features that maximizes the variance.
  - Subsequent linear combinations are orthogonal to previous combinations which maximizes the variance.

Solution:

- Eigenvalue decomposition of $\mathbf{X}^{'}\mathbf{X}$. (`eig()` in `numpy.linalg` or `eigen()` in `R`).

PCA

- Equivalent PCA Criterion:

maximize$_{\mathbf{u}_k, \mathbf{v}_k} \mathbf{u}_k^{'} \mathbf{X} \mathbf{v}_k$ subject to

$\|\mathbf{v}_k\|_2 = 1$ & $\mathbf{v}_k^{'} \mathbf{v}_j = 0 \; \forall \; j < k$ and $\|\mathbf{u}_k\|_2 = 1$ & $\mathbf{u}_k^{'} \mathbf{u}_j = 0 \; \forall \; j < k$.

- Finds left and right projection that maximize variance.
- Solution: Singular Value Decomposition (SVD) of $\mathbf{X}$. (`svd()` in `numpy.linalg` or `svd()` in `R`).

PCA

SVD: $\mathbf{X}_{np} = \mathbf{U}_{nn}\mathbf{D}_{np}\mathbf{V}'_{pp}$

- Singular vectors: (left) $\mathbf{U}$ and (right) $\mathbf{V}$.
  - Orthonormal - $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\mathbf{V}'\mathbf{V} = \mathbf{I}$.
- Singular values: Diagonals of $\mathbf{D}$
  - $d_1 \geq d_2 \geq \cdots \geq d_r$ where $r = \text{rank}(\mathbf{X})$

SVD Solution to PCA:

- PCs: $\mathbf{Z} = \mathbf{XV}$ or $\mathbf{Z} = \mathbf{UD}$
  - $\mathbf{z}_k = \mathbf{X}\mathbf{v}_k$ is the k^{th} PC
  - $\mathbf{z}_1, \ldots, \mathbf{z}_K$ gives best K-dimensional projection of the data.
- PC Loadings: $\mathbf{V}$
  - $\mathbf{v}_k$ is the kth PC loading (feature weights).

PCA

- The loading vector $\mathbf{v}_k, k = 1, 2, \ldots$ defines a direction in feature space along which the data vary the most.
- If we project the ith data point $(x_{i1}, \ldots, x_{ip})$ onto this direction, the projected value $z_{i1} = v_{i1}x_{i1} + \cdots + v_{ip}x_{ip}$ is referred as the **principal component score**.

PCA

PCA Properties:

- Unique
  - **U** and **V** are unique up to a sign change.
  - **D** is unique.
- Global Solution
- Typically, one should center features (i.e. columns of **X**.
  - Maximizing variance interpretation (assumes multivariate Gaussian model).
- Scaling changes PCA solution.
  - Features with large scale contribute more to variance, have large PC loadings.
  - Don't scale if features measured in same way & scale has meaning

PCA - Dimension Reduction

How much variance is explained? (i.e. extent of dimension reduction)

- Variance explained by kth PC:

$$\mathbf{v}_k^{'}\mathbf{X}^{'}\mathbf{X}\mathbf{v}_k = d_k^2$$

.

- Total variance of data:

$$\sum_{k=1}^{p} d_k^2$$

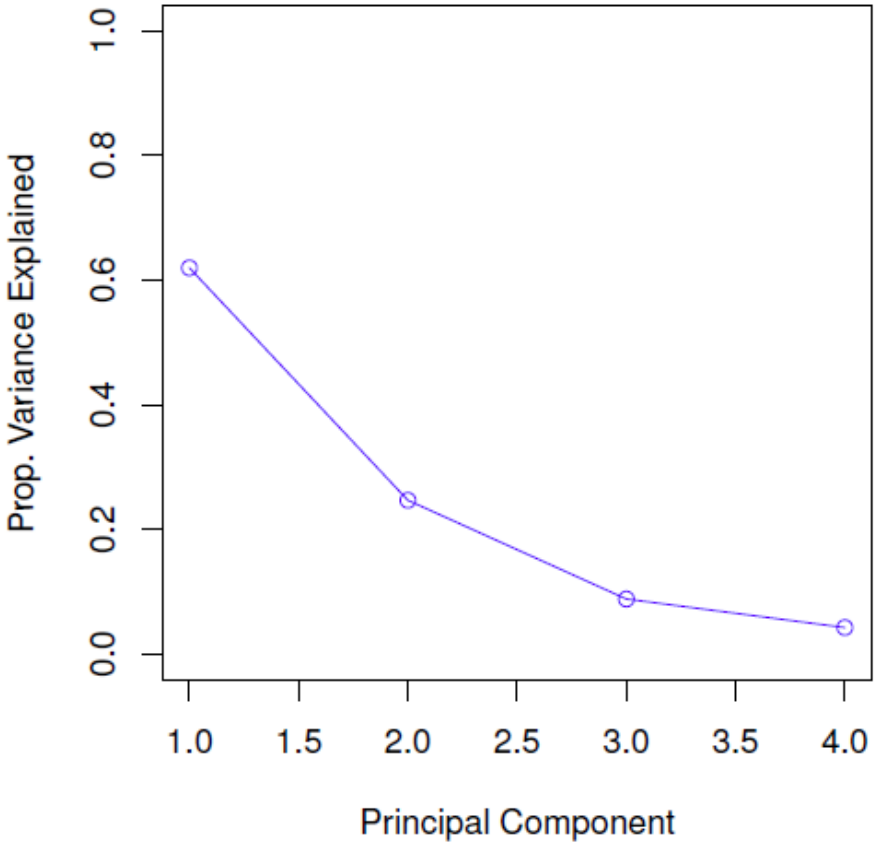
- Proportion of variance explained by kth PC:

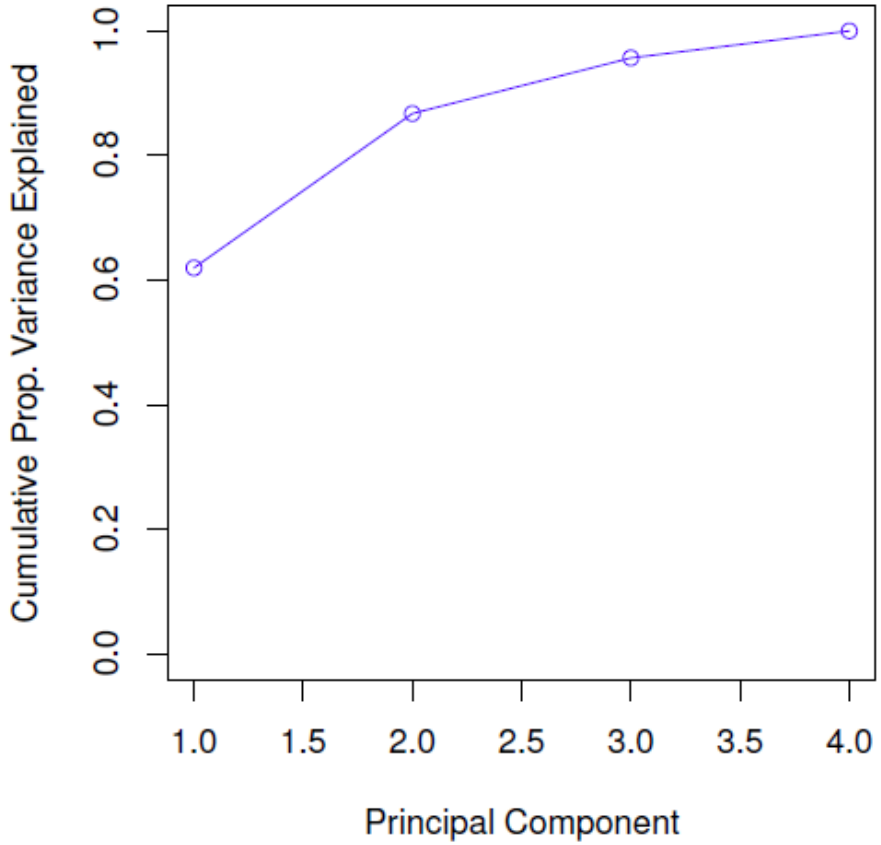
$$d_k^2 / \sum_{k=1}^{p} d_k^2$$

- Cumulative variance explained by the first r PCs (extent of dimension reduction achieve by the first r PC projections):

$$\sum_{k=1}^{r} d_k^2 / \sum_{k=1}^{p} d_k^2$$

PCA - Dimension Reduction





PCA - Data Visualization

PC Scatterplots:

- Problem: Can't visualize
- Solution: Plot $\mathbf{u}_1$ vs. $\mathbf{u}_2$ and so forth.
- Advantages:
  - Dramatically reduces number of 2D scatterplots to visualize.
  - Focuses on patterns with most variance.

PC Loadings Plots:

- Scatterplots of $\mathbf{v}_1$ vs. $\mathbf{v}_2$.
- Visualizations of $\mathbf{v}_k$.

Biplot:

- Scatterplot of PC 1 vs. PC 2 with loadings of $\mathbf{v}_1$ vs. $\mathbf{v}_2$ overlaid.

How many PCs should we use?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
- Take K PCs that explains at least 90% (95%, 99%, etc.) variance
- the "scree plot" on the previous slide can be used as a guide: we look for an "elbow".
  - This is done by eyeballing the scree plot, and looking for a point at which the proportion of variance explained by each subsequent principal component drops off. This is often referred to as an **elbow** in the scree plot.

Python: PCA

- We perform PCA on the `USArrests` data set, which is part of the base R package. The rows of the data set contain the 50 states, in alphabetical order.

```
import pandas as pd
USArrests = pd.read_csv('../data/USArrests.csv', index_col=0 )
USArrests.index
```

```
## Index(['Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California',
##        'Connecticut', 'Delaware', 'Florida', 'Georgia', 'Hawaii',
##        'Illinois', 'Indiana', 'Iowa', 'Kansas', 'Kentucky', 'Loui
##        'Maine', 'Maryland', 'Massachusetts', 'Michigan', 'Minnesc
##        'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada'
##        'New Hampshire', 'New Jersey', 'New Mexico', 'New York',
##        'North Carolina', 'North Dakota', 'Ohio', 'Oklahoma', 'Ore
##        'Pennsylvania', 'Rhode Island', 'South Carolina', 'South D
##        'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Wash
##        'West Virginia', 'Wisconsin', 'Wyoming'],
##       dtype='object')
```

Python: PCA

- The columns of the data set contain the four variables.

```
USArrests.columns
```

```
## Index(['Murder', 'Assault', 'UrbanPop', 'Rape'], dtype='object')
```

- We first briefly examine the data. We notice that the variables have vastly different means. We also examine the variances of the four variables
- pandas has a built-in function to get the mean and std of each column (see lecture 7).

```
USArrests.mean()
```

```
## Murder       7.788
## Assault    170.760
## UrbanPop    65.540
## Rape        21.232
## dtype: float64
```

```
USArrests.std()
```

```
## Murder       4.355510
## Assault     83.337661
## UrbanPop    14.474763
## Rape         9.366385
## dtype: float64
```

Python: PCA

- We now perform principal components analysis using the `PCA()` function, which is in `sklearn.decomposition`.
- *scale*: scale predictor variables: This tells Python that each of the predictor variables should be scaled to have a mean of 0 and a standard deviation of 1. This ensures that no predictor variable is overly influential in the model if it happens to be measured in different units.

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
X = pd.DataFrame(scale(USArrests), index=USArrests.index,
columns=USArrests.columns)
pca_loadings = pd.DataFrame(PCA().fit(X).components_.T,
index=USArrests.columns, columns=['V1', 'V2', 'V3', 'V4'])
print(pca_loadings)
```

```
##                 V1        V2        V3        V4
## Murder    0.535899  0.418181 -0.341233  0.649228
## Assault   0.583184  0.187986 -0.268148 -0.743407
## UrbanPop  0.278191 -0.872806 -0.378016  0.133878
## Rape      0.543432 -0.167319  0.817778  0.089024
```

- The `loadings` matrix provides the principal component loadings: a matrix whose columns contain the eigenvectors; each column contains the corresponding principal component loading vector.

Python: PCA

- fit the PCA model and transform X to get the principal components

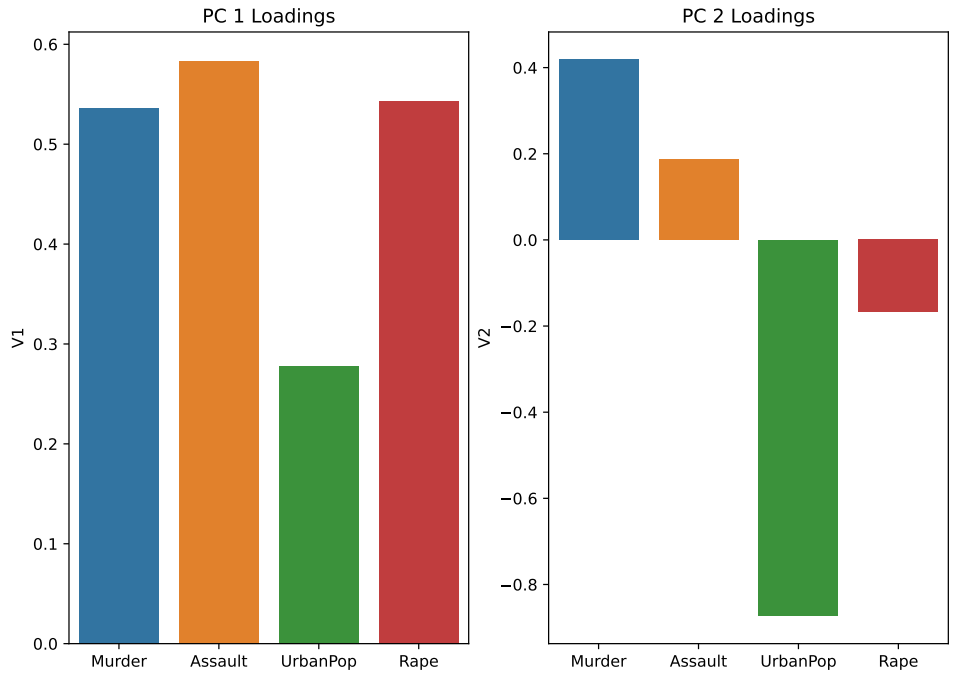
```
pca = PCA()
df_plot = pd.DataFrame(pca.fit_transform(X),
columns=['PC1', 'PC2', 'PC3', 'PC4'], index=X.index)
df_plot
```

```
##                    PC1       PC2       PC3       PC4
## Alabama       0.985566  1.133392 -0.444269  0.156267
## Alaska        1.950138  1.073213  2.040003 -0.438583
## Arizona       1.763164 -0.745957  0.054781 -0.834653
## Arkansas     -0.141420  1.119797  0.114574 -0.182811
## California    2.523980 -1.542934  0.598557 -0.341996
## Colorado      1.514563 -0.987555  1.095007  0.001465
## Connecticut  -1.358647 -1.088928 -0.643258 -0.118469
## Delaware      0.047709 -0.325359 -0.718633 -0.881978
## Florida       3.013042  0.039229 -0.576829 -0.096285
## Georgia       1.639283  1.278942 -0.342460  1.076797
## Hawaii       -0.912657 -1.570460  0.050782  0.902807
## Idaho        -1.639800  0.210973  0.259801 -0.499104
## Illinois      1.378911 -0.681841 -0.677496 -0.122021
```

Python: PCA

- Plot of the loadings lets us check how the variables contribute to the pattern

```
import matplotlib.pyplot as plt
import seaborn as sns
fig, (ax1, ax2) = plt.subplots(1, 2)
sns.barplot(x=pca_loadings.index,y=pca_loadings.V1, ax=ax1)
sns.barplot(x=pca_loadings.index,y=pca_loadings.V2, ax=ax2)
ax1.title.set_text('PC 1 Loadings')
ax2.title.set_text('PC 2 Loadings')
plt.show()
```



Python: PCA

- The variance explained by each principal component:

```
varex=pca.explained_variance_
print(varex)
```

```
## [2.53085875 1.00996444 0.36383998 0.17696948]
```

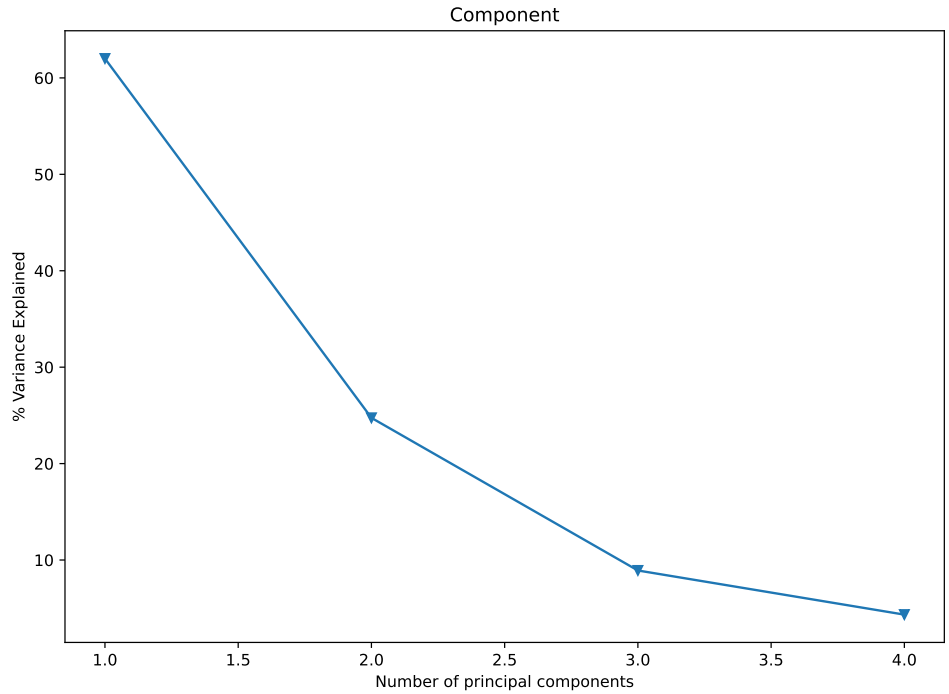
```
varex_ratio=pca.explained_variance_ratio_*100
print(varex_ratio)
```

```
## [62.00603948 24.74412881  8.91407951  4.33575219]
```

Python: PCA

- Plot of variance explained

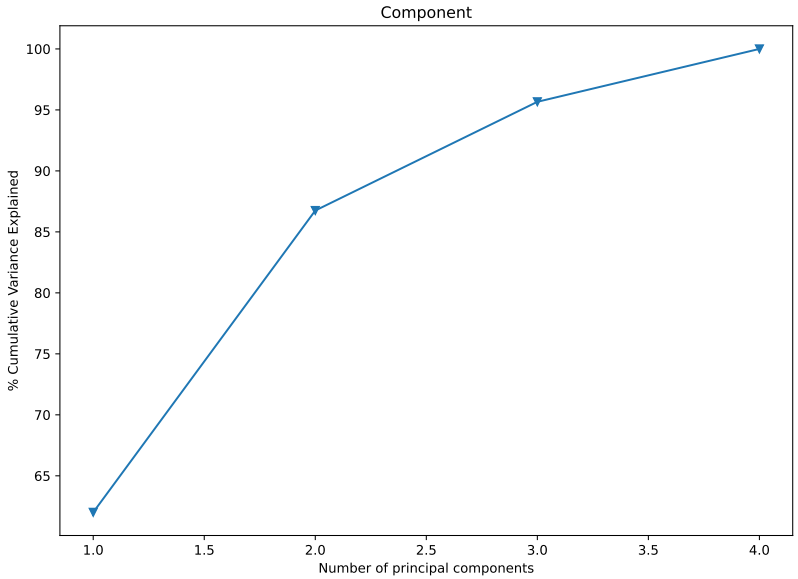
```
import matplotlib.pyplot as plt
plt.plot(range(1, 5), varex_ratio, '-v')
plt.xlabel('Number of principal components')
plt.ylabel('% Variance Explained')
plt.title('Component')
plt.show()
```



Python: PCA

- Plot of cumulative variance explained

```
import numpy as np
varex_ratio_cum=np.cumsum(varex_ratio)
import matplotlib.pyplot as plt
plt.plot(range(1, 5), varex_ratio_cum, '-v')
plt.xlabel('Number of principal components')
plt.ylabel('% Cumulative Variance Explained')
plt.title('Component')
plt.show()
```



License