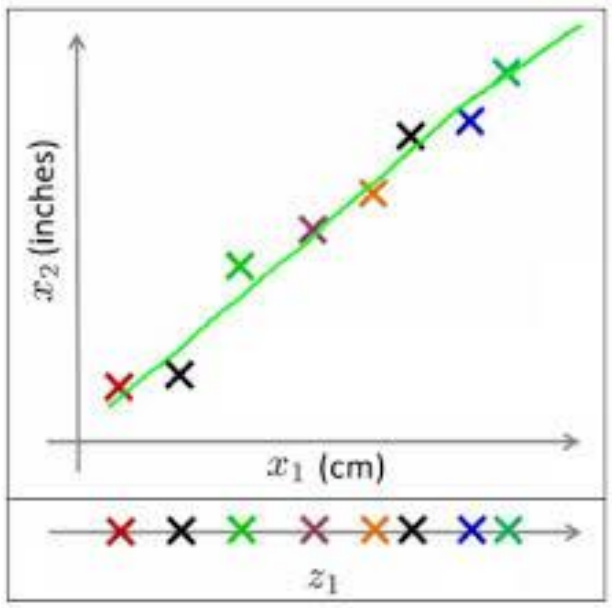
Machine Learning

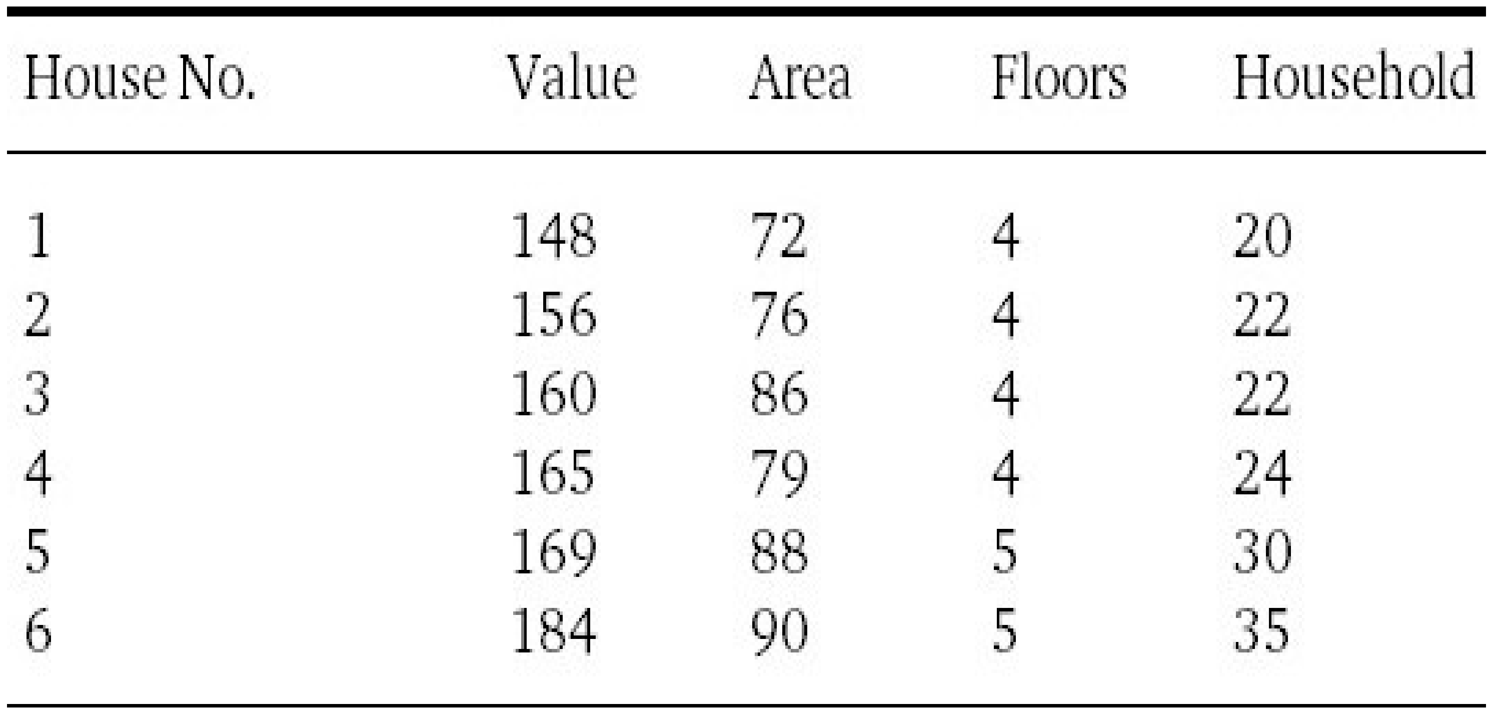
Dimensionality Reduction

Dimensionality Reduction

- **Finds a low-dimensional representation of the data that retains as much information as possible.**
- **Eg: two dimensions x1 and x2 converted into one dimension i.e z1.**



Getting rid of Unnecessary



House No.	Value	Area	Floors	Household
1	148	72	4	20
2	156	76	4	22
3	160	86	4	22
4	165	79	4	24
5	169	88	5	30
6	184	90	5	35

Getting rid of Unnecessary

- **Can you characterize each house by its floor count?**
- **No, as they have low variance, σ^2=0.2**
- **Can you characterize each house by its households?**
- **May be, as its variance is σ^2=28**
- **Area has the variance 43.**
- **Value has the variance 127**

So, Area and Value are more representative than other two features.

Getting rid of Unnecessary

- What about the relation between features?
- Value is roughly double the Area.
- i.e. covariance is high in between Area and Value.
- Higher the covariance, the more correlated the two features are, which implies redundancy in the data.

Discussion

- **It's a *good thing* to have features with *high variance*, since they will be more informative and more important.**
- **It's a *bad thing* to have highly *correlated features* since they can be deduced from one another with little loss in information.**

Why Dimension Reduction is Important?

- **More variables, more trouble.**
- **Relevant/irrelevant variables.**
- **Is there any algorithm which can identify the most significant variable?**

Different Methods of Dimensionality Reduction

- Feature Selection (Data Preprocessing)
- Feature Extraction: summarize the information of a dataset by transforming it onto a new feature subspace of lower dimensionality.

Dimensionality Reduction via Feature Extraction

- Principal Component Analysis (PCA): unsupervised data compression
- Linear Discriminant Analysis (LDA): supervised technique for maximizing class separability.
- Kernal Principal Component Analysis: Nonlinear Dimensionality Reduction

What Is Principal Component Analysis?

- Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.
- Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

Principal Component Analysis

- Unsupervised linear transformation technique
- Identify patterns in data based on the correlation between features.
- Aims to find the directions of maximum variance in high dimensional data and projects it onto a new subspace with equal or fewer dimensions.

Steps of PCA

1. Standardize the d-dimensional dataset.
2. Construct the covariance matrix.
3. Decompose the covariance matrix into its eigenvectors and eigenvalues.
4. Select k eigenvectors that correspond to the k largest eigenvalues, where k is the dimensionality of the new feature space (k<=d).
5. Construct a projection matrix W from the “top” k eigenvectors.
6. Transform the d-dimensional input dataset X using the projection matrix W to obtain the new k-dimensional feature subspace.

STEP 1: STANDARDIZATION

- The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable.

$$z = \frac{value - mean}{standard\ deviation}$$

Once the standardization is done, all the variables will be transformed to the same scale.

STEP 2: COVARIANCE MATRIX COMPUTATION

- The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other, or in other words, to see if there is any relationship between them. Because sometimes, variables are highly correlated in such a way that they contain redundant information. So, in order to identify these correlations, we compute the covariance matrix.

The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 3-dimensional data set with 3 variables x, y, and z, the covariance matrix is a 3×3 matrix of this from:

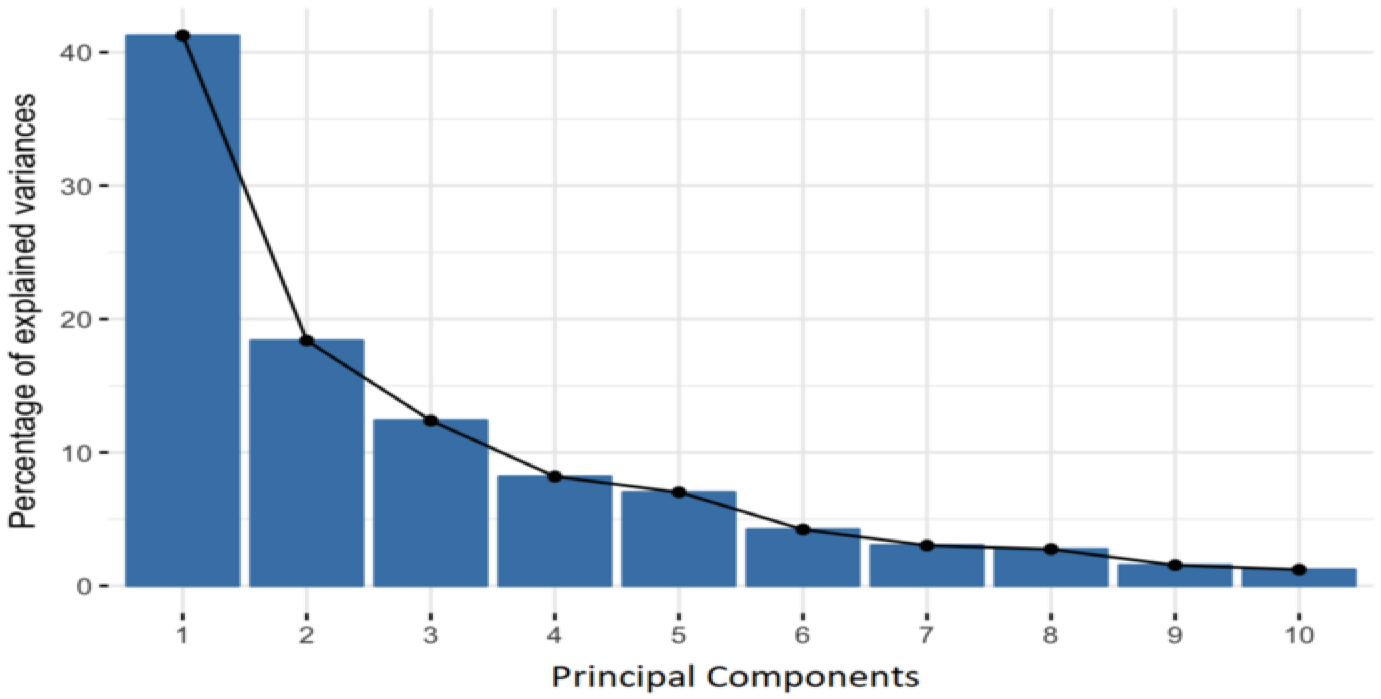
$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

- Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal (Top left to bottom right) we actually have the variances of each initial variable. And since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.

STEP 3: COMPUTE THE EIGENVECTORS AND EIGENVALUES OF THE COVARIANCE MATRIX TO IDENTIFY THE PRINCIPAL COMPONENTS

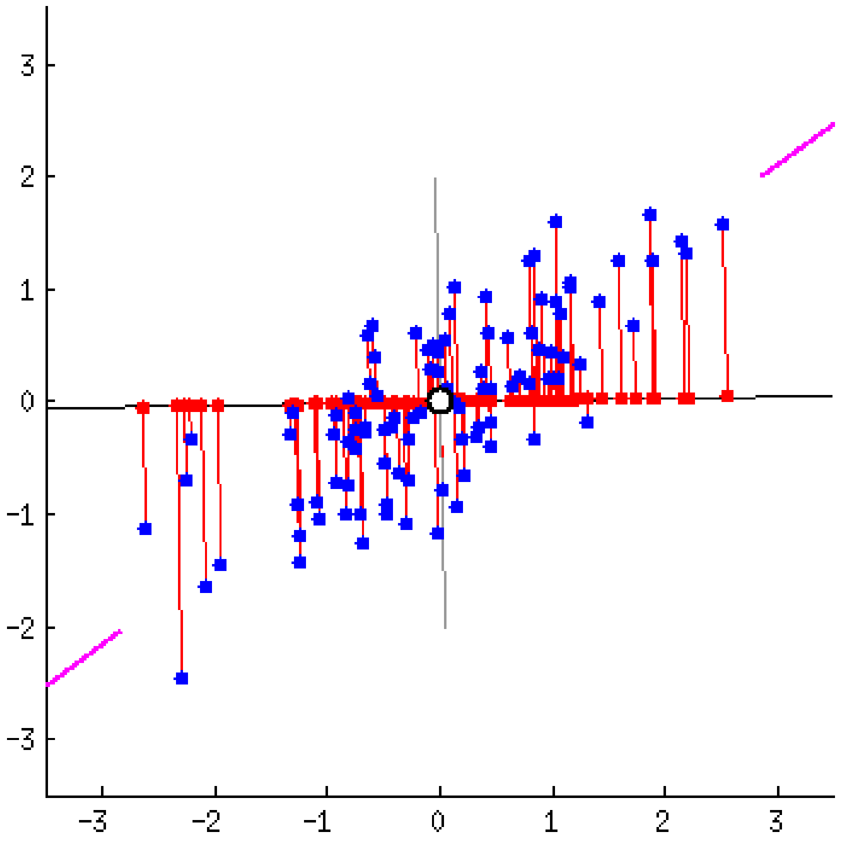
- Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine the ***principal components*** of the data. Before getting to the explanation of these concepts, let’s first understand what do we mean by principal components.

- Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables.



How PCA Constructs the Principal Components

- As there are as many principal components as there are variables in the data, principal components are constructed in such a manner that the first principal component accounts for the **largest possible variance** in the data set.



3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

STEP 4: FEATURE VECTOR

- As we saw in the previous step, computing the eigenvectors and ordering them by their eigenvalues in descending order, allow us to find the principal components in order of significance. In this step, what we do is, to choose whether to keep all these components or discard those of lesser significance (of low eigenvalues), and form with the remaining ones a matrix of vectors that we call *Feature vector*.

- So, the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep. This makes it the first step towards dimensionality reduction, because if we choose to keep only ***p*** eigenvectors (components) out of ***n***, the final data set will have only ***p*** dimensions.

Example:

Continuing with the example from the previous step, we can either form a feature vector with both of the eigenvectors *v*1 and *v*2:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix}$$

Or discard the eigenvector *v*2, which is the one of lesser significance, and form a feature vector with *v*1 only:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix}$$

Discarding the eigenvector *v2* will reduce dimensionality by 1, and will consequently cause a loss of information in the final data set. But given that *v*2 was carrying only 4% of the information, the loss will be therefore not important and we will still have 96% of the information that is carried by *v*1.

LAST STEP: RECAST THE DATA ALONG THE PRINCIPAL COMPONENTS AXES

- In the previous steps, apart from standardization, you do not make any changes on the data, you just select the principal components and form the feature vector, but the input data set remains always in terms of the original axes (i.e, in terms of the initial variables).

In this step, which is the last one, the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components (hence the name Principal Components Analysis). This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

PCA

After transformation:

- **The first principal component will have the largest variance.**
- **All principal components will have largest possible variance given that they are uncorrelated to the other principal components.**
- **PCA directions are highly sensitive to data scaling, so *features* must be *standardized* prior to PCA.**

Program to compute total and explained variance

```
from sklearn.datasets import load_wine
dataset=load_wine()
data=dataset.data
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test=train_test_split(data, dataset.target,test_size=0.3)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_sc_train=sc.fit_transform(x_train)
x_sc_test=sc.fit_transform(x_test)
import numpy as np
cov_mat=np.cov(x_sc_train.T)
eigen_vals, eigen_vecs=np.linalg.eig(cov_mat)
print("Eigenvalues: ",eigen_vals)
tot=sum(eigen_vals)
var_exp=[(i/tot) for i in sorted(eigen_vals, reverse=True)]
cum_var_exp=np.cumsum(var_exp)
import matplotlib.pyplot as plt
plt.bar(range(1,14),var_exp, alpha=0.5, align='center', label='individual explained variance')
plt.step(range(1,14), cum_var_exp, where='mid', label='cumulative explained variance',c='red')
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal Components')
plt.legend()
plt.show()
```

Program to implement PCA using Scikit-Learn

```
from sklearn.datasets import load_wine
dataset=load_wine()
data=dataset.data
target=dataset.target
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(data, target)
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train_sc=sc.fit_transform(x_train)
x_test_sc=sc.fit_transform(x_test)
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
y=lr.predict(x_test)
from sklearn.metrics import accuracy_score
accuracy_score(y,y_test)
from sklearn.decomposition import PCA
pca=PCA()
x_train_pca=pca.fit_transform(x_train_sc)
x_test_pca=pca.transform(x_test_sc)
lr1=LogisticRegression()
lr1.fit(x_train_pca,y_train)
y1=lr1.predict(x_test_pca)
accuracy_score(y1,y_test)
```

LDA: Linear Discriminant Analysis

- Linear Discriminant Analysis or LDA is a dimensionality reduction technique. It is used as a pre-processing step in Machine Learning and applications of pattern classification. The goal of LDA is to project the features in higher dimensional space onto a lower-dimensional space in order to avoid the curse of dimensionality and also reduce resources and dimensional costs.

LDA is a supervised classification technique that is considered a part of crafting competitive machine learning models. This category of dimensionality reduction is used in areas like image recognition and predictive analysis in marketing.

How to have a practical approach to an LDA model?

- Consider a situation where you have plotted the relationship between two variables where each color represents a different class. One is shown with a red color and the other with blue.

- If you are willing to reduce the number of dimensions to 1, you can just project everything to the x-axis as shown below:

- This approach neglects any helpful information provided by the second feature. However, you can use LDA to plot it. The advantage of LDA is that it uses information from both the features to create a new axis which in turn minimizes the variance and maximizes the class distance of the two variables.

Linear Discriminant Analysis vs PCA

- PCA ignores class labels and focuses on finding the principal components that maximizes the variance in a given data. Thus it is an unsupervised algorithm. On the other hand, LDA is a supervised algorithm that intends to find the linear discriminants that represents those axes which maximize separation between different classes.

- LDA performs better multi-class classification tasks than PCA. However, PCA performs better when the sample size is comparatively small. An example would be comparisons between classification accuracies that are used in image classification.
- Both LDA and PCA are used in case of dimensionality reduction. PCA is first followed by LDA.

COMING UP