CHAPTER 2

UNIT II

Feature Selection

Syllabus

Scikit-learn Dataset, Creating training and test sets, managing categorical data, Managing missing features, Data scaling and normalization, Feature Selection and Filtering, Principle Component Analysis(PCA) - Non negative matrix factorization, Sparse PCA, Kernel PCA, Atom Extraction and Dictionary Learning.

2.1 Introduction to Feature Selection

Feature selection is an important task that goes in hand in hand with feature engineering, where the job of a data scientist is to select the best possible subset of features and attribute that would help in building the right model. Feature engineering and selection is not a one time task, it needs to be carried out multiple times each time you build a model to get the best and optimal model for your problem. Data preprocessing and Feature Engineering is one of the toughest task in building a model for machine learning by data scientist.

- Feature selection provides an effective way to solve a problem by removing redundant and irrelevant data.
- It is a process of isolating only those variables (or features) that are relevant to analysis.
- It keeps the best subset of predictors in the model.
- Feature selection is also called as variable selection or attributes selection.
- Selection of fewer attributes is advantageous as it reduces the complexity of the model and also reduces computational time.
- Feature selection is different from dimensionality reduction (Feature extraction). In dimensionality reduction, the number of attributes are reduced by creating a new combination of attributes present in the data. (e.g. of dimensionality reduction is PCA (Principal Component Analysis).

Uses of Feature Selection

1. The complexity of a model is reduced; this makes it easier to interpret.
2. Enables the Machine Learning algorithm to train faster.
3. If a right subset is chosen, accuracy of a model is improved.
4. Over fitting is reduced.

2.2 About Scikit-learn Library

- Scikit-learn is a library built upon SciPy (Scientific Python) which includes different packages like NumPy, SciPy, Matplotlib, IPython, SymPy and Pandas.
- It provides a range of Supervised and Unsupervised Learning algorithms.
- Simple and efficient tools for data mining and data analysis are provided.
- It is open source, commercially usable under BSD License.

2.2.1 Scikit - Learn Datasets [Ref URL : https://scikit-learn.org/stable/datasets/index.html]

- The Sklearn provides data that comes from the real world, which is used by the machine learning community. These data sets are used to benchmark algorithms.
- Based on the type of dataset, three different dataset interfaces are provided by the library, which can be used to get the data sets.

(i) The dataset Loaders

Used for loading small standard datasets. (Toy dataset section)

(ii) The Dataset Fetchers

Used for downloading and loading Larger datasets. (Real-world datasets section)

(iii) The Dataset generation functions

Used for generating controlled synthetic datasets. (Generated Datasets section)

2.2.2 The Toy Datasets

Scikit-learn comes with some small datasets and does not require any file download from some external source.

Following are the datasets available in Toy Datasets

1. Boston

Load and return the boston house-prices dataset (regression).

Samples total	506
Dimensionality	13
Features	real, positive
Targets	real 5 - 50.

Data Set Characteristics

Number of Instances
506
Number of Attributes
13 numeric/categorical predictive. Median Value (attribute 14) is usually the target.
Attribute Information (in order)
CRIM per capita crime rate by town
ZN proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS proportion of non-retail business acres per town
CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX nitric oxides concentration (parts per 10 million)
RM average number of rooms per dwelling

AGE proportion of owner-occupied units built prior to 1940
DIS weighted distances to five Boston employment centers
RAD index of accessibility to radial highways
TAX full-value property-tax rate per $10,000
PTRATIO pupil-teacher ratio by town
B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT % lower status of the population
MEDV Median value of owner-occupied homes in $1000's
Missing Attribute Values
None
Creator : Harrison, D. and Rubinfeld, D.L.

2. **Iris**

The iris dataset is a multi-class classification dataset of plants.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

Data Set Characteristics

Number of Instances
150 (50 in each of three classes)
Number of Attributes
4 numeric, predictive attributes and the class
Attribute Information
sepal length in cm sepal width in cm petal length in cm petal width in cm **class :** Iris-Setosa Iris-Versicolour Iris-Virginica

Summary Statistics					
sepal length :	4.3	7.9	5.84	0.83	0.7826
sepal width :	2.0	4.4	3.05	0.43	−0.4194
petal length :	1.0	6.9	3.76	1.76	0.9490 (high!)
petal width :	0.1	2.5	1.20	0.76	0.9565 (high!)

Missing Attribute Values
None
Class Distribution
33.3% for each of 3 classes.
Creator : R.A. Fisher
Donor : Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
Date : July, 1988

3. Diabetes

Load and return the diabetes dataset (regression).

Samples total	442
Dimensionality	10
Features	real, $-.2 < x < .2$
Targets	integer 25 - 346

Data Set Characteristics

Number of Instances
442
Number of Attributes
First 10 columns are numeric predictive values
Target Column 11 is a quantitative measure of disease progression one year after baseline
Attribute Information
Age Sex Body mass index Average blood pressure S1 S2 S3 S4 S5 S6

Note : Each of these 10 feature variables have been mean centered and scaled by the standard deviation times n_samples (i.e. the sum of squares of each column totals 1).

4. Digits

- Load and return the digits dataset (classification).
- Each data point is a 8x8 image of a digit.

Classes	10
Samples per class	~180
Samples total	1797
Dimensionality	64
Features	integers 0-16

Data Set Characteristics

Number of Instances
5620
Number of Attributes
64
Attribute Information
8x8 image of integer pixels in the range 0..16.
Missing Attribute Values
None
Creator : Alpaydin (alpaydin '@' boun.edu.tr)
Date : July ; 1998

5. Linnerud

Load and return the linnerud dataset (multivariate regression).

Samples total	20
Dimensionality	3 (for both data and target)
Features	Integer
Targets	Integer

Data Set Characteristics

Number of Instances
20
Number of Attributes
3
Missing Attribute Values
None

The Linnerud dataset constains two small dataset :

(a) Physiological - CSV containing 20 observations on 3 exercise variables Weight, Waist and Pulse.

(b) Exercise - CSV containing 20 observations on 3 physiological variables Chins, Situps and Jumps.

6. **Wine**

The wine dataset is a classic and very easy multi-class classification dataset.

Classes	3
Samples per class	[59,71,48]
Samples total	178
Dimensionality	13
Features	real, positive

Data Set Characteristics

Number of Instances
178 (50 in each of three classes)
Number of Attributes
13 numeric, predictive attributes and the class
Attribute Information
Alcohol
Malic acid
Ash
Alcalinity of ash
Magnesium
Total phenols
Flavanoids
Nonflavanoid phenols
Proanthocyanins
Color intensity
Hue
OD280/OD315 of diluted wines
Proline

Class

class_0

class_1

class_2

7. Breast cancer

- Load and return the breast cancer wisconsin dataset (classification).
- The breast cancer dataset is a classic and very easy binary classification dataset.

Classes	2
Samples per class	212(M),357(B)
Samples total	569
Dimensionality	30
Features	real, positive

Data Set Characteristics

Number of Instances
569
Number of Attributes
30 numeric, predictive attributes and the class
Attribute Information
radius (mean of distances from center to points on the perimeter)
texture (standard deviation of gray-scale values)
Perimeter
Area
smoothness (local variation in radius lengths)
compactness (perimeter^2 / area - 1.0)
concavity (severity of concave portions of the contour)
concave points (number of concave portions of the contour)
symmetry
Fractal dimension ("coastline approximation" - 1)
The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.
Class
WDBC-Malignant
WDBC-Benign

Note : For Further information visit: http://scikitlearn.org/stable/datasets/index.html

2.2.3 Real World Datasets

Following are some of the large datasets available.

1. **California Housing dataset**

Data Set Characteristics

Number of Instances
20640
Number of Attributes
8 numeric, predictive attributes and the target
Attribute Information
MedInc median income in block
HouseAge median house age in block
AveRooms average number of rooms
AveBedrms average number of bedrooms
Population block population
AveOccup average house occupancy
Latitude house block latitude
Longitude house block longitude
Missing Attribute Values
None

2. **KDDCup 99 dataset**

- KDDCup dataset was created by MIT Lincoln Lab. The dataset is an Intrusion Detection System(IDS) evaluation dataset.
- The dataset was generated in a closed network. The attacks were hand injected to produce large number of different types of attack considering normal activity in the background.
- The dataset is transformed into two different data sets SA and SF
  1. **SA :** This dataset is a collection of all normal data with a small proportion of abnormal data, that gives an anomaly proportion of 1%
  2. **SF :** This dataset is obtained by data whose attribute logged in is positive, focusing on the intrusion attack, giving a proportion of 0.3% of attack.

General KDD structure

Samples total	4898431
Dimensionality	41
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

SA structure

Samples total	976158
Dimensionality	41
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

SF structure

Samples total	699691
Dimensionality	4
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

http structure

Samples total	619052
Dimensionality	3
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

smtp structure

Samples total	95373
Dimensionality	3
Features	discrete (int) or continuous (float)
Targets	str, 'normal.' or name of the anomaly type

3. The 20 newsgroups text dataset

This dataset consist of 18000 newsgroups posts on 20 topics, splitting into two subsets (training and testing).

Data Set Characteristics

Classes	20
Samples total	18846
Dimensionality	1
Features	text

Note : For More datasets visit : http://scikitlearn.org/stable/datasets/index.html

2.2.4 Generated Datasets

Scikit-learn incorporates various random sample generators that are used to build artificial datasets of controlled size and complexity.

The generators are classified into the following different types.

(i) Generators for Classification and Clustering

Produces a matrix of features and corresponding discrete target. This generator can generate datasets of single label, multi-label and Bi-clustering.

(ii) Generators for Regression

The regression targets are optionally sparse random linear combination of random features with noise.

(iii) Generators for Manifold learning.

(iv) Generators for decomposition.

2.3 Creating Training and Test Sets

For training and testing purpose of our Model we need to split the Dataset into three distinct datasets, they are training set, validation set and testing set.

1. **Training set**
   - A set of data used to train the model.
   - It is used to fit the model.
   - The model sees and learns from this data.
   - Later on the trained model can be deployed and used to accurately predict on new data that it has not seen before.
   - Labeled data is used.
2. **Validation set**
   - The validation set is the set of data separate from the training set.
   - It is used to validate our model during training.
   - It gives information, which is used for tuning model hyper parameters.
   - It ensures that our model is not overfitting to the data in the training set.
   - Labeled data is used.
3. **Test set**
   - A set of data used to test the model after the model has already been trained.
   - The test set is separate from both the train set and validation set.
   - Once the model is trained and validated using the training and validation sets then the model is used to predict the output for the data in the test set.
   - Unlabeled data is used.

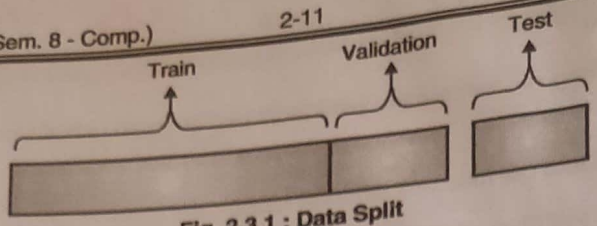


Fig. 2.3.1 : Data Split

Rules for performing data split Operation

- In order to avoid a correlation between the consequent elements, the original dataset must be randomly shuffled before applying the split phase.
- All the splits must represent the original distribution.
- The percentage of splitting is mostly 60% for training, 20% for Validation and 20% for testing. (Varies based on the amount of data and other factors).
- With Scikit-learn, this can be done using train_test_split() function.

2.4 Types of Data

- To understand data let us consider the following example of data.
- From the Fig. 2.4.1 we can see that data need not only be numeric representation, but it can also consists of names or symbols.
- Let us now bring the data into a table as shown in Fig. 2.4.2.
- Every data is given certain heading and the example data fits into heading (Name, age, Gender, score and experience).
- Some of the data may also require units as shown in the Fig. 2.4.2.
- In the Fig.2.4.2, columns are variables.
- Rows are cases or observations (n).

Data can be classified into two types

1. **Categorical Data**
   - Responses that belong to groups or categories.
   - From the data table, the first two columns are categorical data and the rest three are numerical data.
   - Example of categorical data is Yes/No, Male/Female etc.
   - Strongly agree to strongly disagree. Can be further classified as Nominal (no implied order), ordinal (order or rank).
2. **Numerical Data**
   - A numerical value as a response.
   - Discrete number or continuous.
   - For Example number of students in a class.
   - Height of people in locality.
   - Can be further classified as Interval (add/subtract), ratio (add, subtract, multiply and divide).

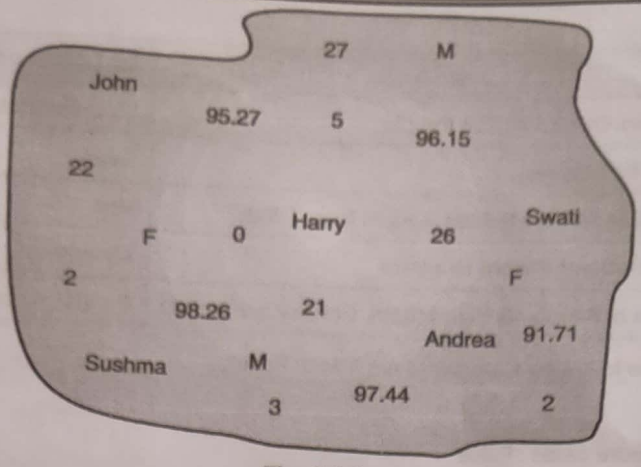


Fig. 2.4.1

Data table

Name	Gender	Age	Score	Experience
John	M	27	95.27	5
Sushma	F	25	96.15	2
Swati	F	26	91.71	3
Harry	M	21	97.44	0
Andrea	F	22	98.26	12
Nominal	**nominal**	**ratio**	**ordinal**	**interval**

Fig. 2.4.2 : Data Table

1. **Qualitative Data**
   - No measurable meaning to the difference of numbers.
   - E.g. Number in the shirt of a sports person (There is no significance to the meaning of numbers on their shirts e.g. a sports person having number 82 would not mean that he is a senior player when compared to a sports person wearing a shirt having number 45).
   - Qualitative data can be further classified into Nominal and Ordinal.
2. **Quantitative Data**
   - Meaning to the difference can be given.
   - E.g. 80 marks and 60 marks (we can say one person has scored more than the other person).

Example 1 : An example to give the type of variable (categorical, ordinal, numerical) for the following example.

1. Two Wheelers Owned by Ten People
2. Salary of 50 Employees
3. Size of Dress as Small, Medium, Large, Extra Large
4. Number of students absent in a class
5. Qualification of People as High school, Graduate, PG, PhD

Solution :

Sr. No.	Description	Variable Name	Variable Type
1.	Two Wheelers Owned by Ten People	Model(or Brand)	Categorical
2.	Salary of 50 Employees	Salary	Numerical
3.	Size of Dress as Small, Medium, Large , Extra Large	Size	Ordinal
4.	Number of students absent in a class	Absentees	ratio
5.	Qualification of People as High school, Graduate, PG, PhD	Qualification	Categorical

Example 2 : State whether the following statements are true or false.

Solution :

- Aggregation of Data adds more cases : **False**
  - It reduces the number of observations
- Pin Codes are examples of Numerical Data : **False**
  - Although they are numeric, since we can neither add, subtract, multiple or divide and make meaningful conclusions from data.
  - So it is categorical data.
- Cases represent columns in a data table : **False**
  - Cases are observations / rows in a data table.
- Frequency of time series is the time spacing between data : **True**
  - Time series data is data measured across time.
  - E.g. Students in a class for a particular year.
  - Sales in twelve months of the year.
  - Stock prices in last two weeks.
- Likert scale represents numerical data : **False**
  - Likert scale does not represent numerical data.
  - It represents ordinal data (categorical data).

Cross sectional and Times Series Data

- **Time series data :** Data measured across different points in time.
- **Cross sectional data :** Looking at the data in certain instance of time.

Example 3 : Find out whether the following examples are cross sectional or time series data.

1. Company has data on number of employees who are working on a Manufacturing project and the amount sanctioned to that project.
2. 2000 people were asked whether a particular political party would win the election in the upcoming year.
3. The number of people who bought their grocery items from a supermarket for more than Rs.5000 for five days of the week.
4. 100 customers give a movie review feedback, 50 ticked excellent, 30 ticked average and 20 ticked poor.
5. Number of cars parked in a parking lot for 7 days a week.

Solution :

1. Cross sectional : As it is taken at a certain point and not in different points for comparison.
2. Cross sectional.
3. Time series data : Data is measured according to some frequency.
4. Cross Sectional.
5. Time Series.

2.4.1 Managing Categorical Data

I. Transforming Categorical Features

As we have seen that categorical variables can be of two types : Nominal and Ordinal.

Transforming Nominal Features

- Nominal attributes are categorical variables having a distinct discrete values.
- The format of nominal variables are in text or string, these values cannot be understood by machine learning algorithms.
- Due to which they need to be transformed into numeric format.

Transforming Ordinal Features

- Ordinal features are analogous to nominal variables except they have ordering among their values.
- They can be represented in text form, so a mapping can be used to represent them into numeric form.

II. Encoding Categorical Features

Need for Encoding categorical Features

- If the transformed numeric representations of categorical features have no ordinal relationships, transforming categorical features are not enough.
- If these transformed categorical features are fed into any algorithm, the model will treat them as raw numeric features and the notion of magnitude will be wrongly introduced in the system.
- If model were built using these features then it would result in sub optimal and incorrect model.
- To encode the categorical features techniques like one hot encoding, dummy encoding, effect encoding and feature hashing schemes can be used.
- For example in our dataset we have a column for color and three different colors are present (Red, yellow and green).
- If we apply transformation then Red can have a numeric representation of 1, yellow = 2 and green = 3.

Color		Color
Red		1
Red	⇒	1
Yellow		2
Green		3
Yellow		2

Fig. 2.4.3 : Categorical Features having 3 labels

A. One Hot Encoding scheme

- Consider a numeric representation of any categorical features with m labels e.g. in the Fig. 2.4.3 (m = 3 labels).
- In this technique it encodes or transforms the features into m (m=3) binary features containing a value of 1 or 0.
- Each observation in the categorical feature is thus converted into a vector size of m (m=3) with only one of the value as 1, which indicates its active.
- The binary variables are often called as dummy variables.

Example :

Color		Red	Yellow	Green
Red		1	0	0
Red		1	0	0
Yellow	⇒	0	1	0
Green		0	0	1
Yellow		0	1	0

B. Dummy Coding scheme

- It's a technique similar to one hot encoding scheme, except we have m – 1 binary features as compared to m features in one hot encoding scheme.
- The categorical variable is converted into a vector of size m – 1.
- The extra feature is omitted, and if the category values ranges from {0, 1...., m–1} the 0^{th} or the $m–1^{th}$ feature is represented as a vector of all zeros(0).
- **Example :** (case i) 0^{th} feature is represented by a vector of all zeros(0)

Color		Yellow	Green
Red		0	0
Red		0	0
Yellow	⇒	1	0
Green		0	1
Yellow		1	0

(Case ii) $m–1^{th}$ *feature is removed and it is represented by a vector or all zeros (0)*

Color		Red	Yellow
Red		1	0
Red		1	0
Yellow	⇒	0	1
Green		0	0
Yellow		0	1

C. Effect coding scheme

- In most aspects, Effect coding scheme is similar to dummy coding scheme.
- In Effect coding scheme, those encoded features are replaced by -1 that are represented as all 0s in the dummy coding scheme.

Example :

Color	Yellow	Green
Red	-1	-1
Red	-1	-1
Yellow	1	0
Green	0	1
Yellow	1	0

D. Bin counting scheme

- One hot encoding, dummy coding and effect coding are techniques suitable for small number of distinct categories.
- If the number of distinct categories becomes large, one hot encoding, dummy coding and effect coding techniques start causing a problem.
- If the categorical feature has m distinct labels then m separate features are generated, this can lead to increase in the size of the feature set.
- This can result in model training problems with regard to time, space and memory and also storage issues.
- Another major drawback that arises is the curse of dimensionality in which there are enormous number of features and not enough representative samples, model performance starts getting affected.
- To deal with a large number of possible categories, Bin Counting scheme is useful.
- In this technique instead of using the actual label values for encoding, probability based statistical information about the value and the actual target or response value, which is aimed for prediction by the model is used.
- For example historical data for IP address and the ones that were used for DDOS attacks, a probability can be built for DDOS attack been caused by any one of the IP addresses.
- Using this information, input feature can be encoded which depicts whether the same IP address comes in the future, what is the probability of DDOS attack being caused.

E. Feature Hashing scheme

- This technique is another useful technique for handling large-scale categorical features.
- Feature Hashing scheme uses a Hash function, a vector of predefined length is used which represent the number of encoded features pre-set.
- The indices in this predefined vector are the hashed values of the features and they are updated accordingly

- The technique can result in problem of collision as the hash functions maps a large number of values into a finite set of values.
- To overcome the problem of collisions, a signed hash function can be used. The sign of the value obtained after applying the hash function is used as the sign of the value, this sign is stored in the final feature vector at the appropriate index.
- Hashing scheme is suitable on different data types like strings, numbers and structures like vectors.
- A hashed output represented as a set of h bins, whenever hash is calculated on the same values, they are assigned the same bin out of h bins based on the hash value obtained.
- The final size of the encoded feature vector for each categorical feature encoded using hashing scheme is h.
- For example if there 100 distinct categories in a feature and h is set to 20, the output will have 20 features as compared to 100 if one Hot encoding scheme is used.

2.5 Managing Missing Features

Sometimes the dataset may contain missing features, following are the different ways in which this can be handled :

(i) Removing the entire observation

This approach should be considered only when the data set is large enough.

(ii) Use a supervised model for predicting the missing values.

This approach is little difficult as it is important to decide upon the supervised strategy to train a model.

(iii) An automatic strategy can be used to input them according to other known values.

This option is the best choice. Scikit-learn offers the class Imputer that can be used to fill in the missing value using the mean, median or frequency. mode

2.6 Data Scaling and Normalization

2.6.1 Data Scaling

- When dealing with real time data having numeric features, some of the attributes may be unbounded in nature. For Example number of times a video is viewed on YouTube or the number of hits on a web page.
- Using such raw values as input features may make the model biased towards features having really high magnitude values.
- Models like Linear or Logistic Regression are highly sensitive to the magnitude or scale of features.
- Tree based models can work without feature scaling.
- So it is better to normalize and scale down the features using feature-scaling methods, if different machine learning algorithms are to be explored.
- For example currency like 1 USD = 100 Yen, but if we don't scale the two currencies, then a difference of 1 USD will be considered as same as the difference in 1 Yen.
- By scaling, different variables can be compared on a common basis.

Example of Scaling :

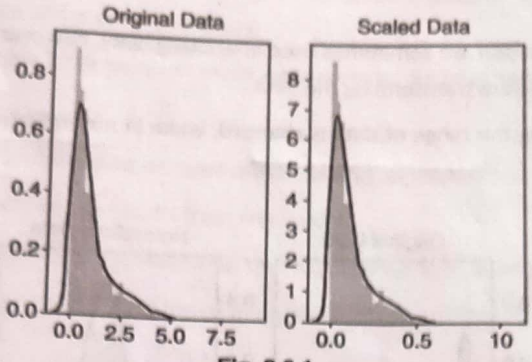


Fig. 2.6.1

A. Standardized Scaling

- In standardized scaling, the values in the feature column are standardized by removing mean and scaling the variance to be 1 from the values.
- This technique is also known as centering and scaling.
- It is popularly known as Z-score scaling.
- It can be mathematically denoted as,

$$SS(X_i) = \frac{X_i - \mu_x}{\sigma_x}$$

- Where, X_i is the feature, μ_x is the mean, σ_x is the standard deviation

B. Min-Max Scaling

- In Min-Max scaling, the features are transformed and scaled such that the each value is in the range of [0,1].
- The Min-Max scaler in scikit learn allows one to specify the upper and lower bound using the feature_range variable

$$MMS(X_i) = \frac{X_i - \min(X)}{\max(X) - \min(X)}$$

SME

X is the feature

min(X) is the minimum value in the feature X

max(X) is the maximum value in the feature X

C. Robust Scaling

- The drawback of Min Max scaling is that the presence of outliers affect the scaled values.
- Robust scaling uses statistical measure to scale the features without being affected by outliers.

$$RS(X_i) = \frac{X_i - \text{median}(X)}{IQR_{(1,3)}(X)}$$

Where,

X is the feature, Median(X) is the median value of X

$IQR_{(1,3)}(X)$ is the Inter quartile range difference between first quartile (25^{th}) and the third quartile (75^{th})

2.6.2 Data Normalization

- The term scaling and normalization are sometimes used interchangeably, however there is a difference between the two terms. In both the cases we are transforming the data.
- The difference is that in scaling the range of data is changed, while in normalization the shape of the distribution of data is changed.

Example :

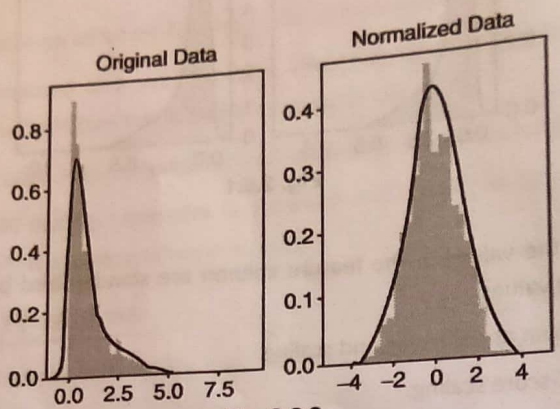


Fig. 2.6.2

- Scikit learn provides a class for per-sample normalization, Normalizer. It has max, L1 and L2 normalization.
- In Euclidean space the techniques may be given as,

$$\text{Max norm : } \|X\|_{max} = \frac{X}{|\max_i \{X\}|}$$

$$\text{L1 norm : } \|X\|_{L1} = \frac{X}{\sum_i |x_i|}$$

$$\text{L2 norm : } \|X\|_{L2} = \frac{X}{\sqrt{\sum_i |x_i|^2}}$$

X is the feature value whose norm is to be calculated. x_i are the values in the features.

Example :

$$X = [1.0, 2.0]$$

$$\text{Max Norm} = [0.5, 1.0]$$

$$\text{L1 norm} = [0.3333, 0.6666]$$

$$\text{L2 norm} = [0.4472, 0.8944]$$

2.7 Feature Selection and Filtering

- Dealing with too many features results in "Curse of Dimensionality".
- More features make the model more complex and difficult to interpret.
- It can also lead to model overfitting on the training data.
- This in turn will result in a model that may give high performance only on the data, which is used for training but will end up in poor performance on previously unseen data.

- The objective is to select optimal number of features, these optimal features can be used for training and building models that generalize on data and prevent overfitting.
- Feature selection techniques can be divided into three main areas based on strategy and techniques employed.

2.7.1 Filter Methods

- In this approach, feature selection is based on metrics like correlation, mutual information etc.
- These techniques do not depend on the results from any model.
- In this approach the relationship of each feature with the response variable to be predicted is tested.
- Techniques under this category are threshold based methods and statistical tests.

2.7.2 Wrapper Methods

- In wrapper methods the interaction between multiple features is captured by using a recursive approach and builds multiple models using a subset of features.
- Then the best subset of features is selected thus giving the best performing model.
- Methods include backward selecting and forward elimination.

2.7.3 Embedded Methods

- In this approach the techniques try to combine the benefits of the other two methods.
- Machine learning models are leveraged themselves to rank and score feature variables based on their importance.
- Methods include tree based methods like decision tree and ensemble methods like random forests.

2.8 Principle Component Analysis (PCA)

- Principal Component Analysis also known as PCA, is a statistical technique.
- It uses a process of Linear, orthogonal transformation to transform higher dimensional features (may be highly correlated) to a lower dimensional set of linearly uncorrelated features.
- These uncorrelated features generated by PCA are called as Principal Components or PCs.
- The total number of Principal components are always less than or equal to the initial number of features.
- The maximum variance of the original features is captured by the first component of PCA.
- The remaining components capture more variance. Such that these PCs are orthogonal to the preceding components.
- This technique is sensitive to feature scaling.

2.8.1 Pre-requisites for PCA (Mathematics Background)

This section deals with some mathematics background that is needed to understand the concept of PCA.

Mean

- It is the average of the numbers.
- In order to calculate mean, add up all the numbers and divide by how many numbers are there.
- However mean does not tell us lot about the data, except its middle point.
- Example {2, 5, 8}, mean = (2 + 5 + 8) / 3 = 5

Formula

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

Standard Deviation

- It is a measure of how spread out the data is.
- It is the average distance from the mean of the data set to a point.

Formula

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

$\bar{X}$: mean ; n : Total number of elements

Example :

X	$(X - \bar{X})$	$(X - \bar{X})^2$
0	−10	100
8	−2	4
12	2	4
20	10	100
Total		208
Divided by (n − 1)		69.333
Square Root		8.3266

Variance

It is another measure that tells about the spread of the data.

Formula

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

Covariance

- The Standard deviation and Variance are 1D measures. However many datasets have more than 1 Dimensions.
- The aim of the statistical analysis is to see if there are any relationships between the dimensions.
- Example we could have dataset like the number of hours spent for studies and the marks obtained, statistical analysis could be performed to find out if the number of hours have any impact on the marks obtained.
- Covariance is a measure that tells how much the dimensions vary from the mean with respect to each other.
- It is always measured between two dimensions.
- If one calculates covariance between one dimension and itself , variance is obtained.
- cov(x, y) is equal to cov(y, x)

Formula

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

Covariance Matrix

- If the dataset has more than 2 dimensions, more than one covariance measure can be calculated.
- For example a three dimensional data set (dimensions x, y, z), cov(x, y), cov(y, z) and cov(x, z) may be calculated.

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

Example :

	Hours (H)	Mark (M)
Data	9	39
	15	56
	25	93
	14	61
	10	50
	18	75
	0	32
	16	85
	5	42
	19	70
	16	66
	20	80
Totals	167	749
Averages	13.92	62.42

Table 2.8.1 : 2-Dimensional data set and covariance calculation

H	M	$(H_i - \bar{H})$	$(M_i - \bar{M})$	$(H_i - \bar{H})(M_i - \bar{M})$
9	39	−4.92	−23.42	115.23
15	56	1.08	−6.42	−6.93
25	93	11.08	30.58	338.83
14	61	0.08	−1.42	−0.11
10	50	−3.92	−12.42	48.69
18	75	4.08	12.58	51.33
0	32	−13.92	−30.42	423.45
16	85	2.08	22.58	46.97
5	42	−8.92	−20.42	182.15
19	70	5.08	7.58	38.51
16	66	2.08	3.58	7.45
20	80	6.08	17.58	106.89
			Total	**1149.89**
			Average	**104.54**

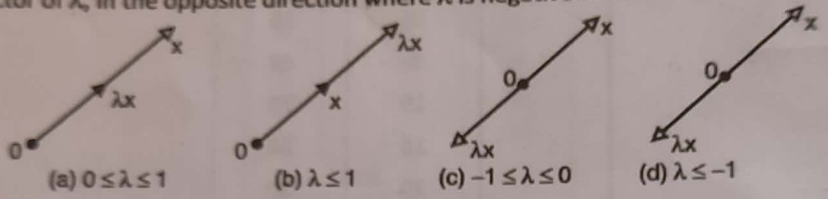
2.8.2 Eigen Values and Eigen Vectors

Definition of Eigenvalue and Eigenvector

- If A is an n × n matrix, then a nonzero vector x in R^n is called an eigenvector of A (or of the matrix operator T_A) if Ax is a scalar multiple of x ; that is,

$$Ax = \lambda x$$

- For some scalar λ the scalar λ is called an eigenvalue of A (or of T_A), and x is said to be an eigenvector corresponding to λ.
- When x is an Eigen vector of A, multiplication by A leaves the direction of the vector unchanged.
- For example in R^2 or R^3 multiplication by A maps each Eigen vector x of A (if any) along the same line through the origin as x.
- Based on the sign and magnitude of eigen value of λ, eigen vector x, the operation $Ax = \lambda x$ either compresses or stretches x by a factor of λ, in the opposite direction where λ is negative as shown in the Fig. 2.8.1.



(a) $0 \le \lambda \le 1$ (b) $\lambda \le 1$ (c) $-1 \le \lambda \le 0$ (d) $\lambda \le -1$

Fig. 2.8.1

Example :

The vector $x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ is an eigenvector of

$$A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$$

Corresponding to the eigenvalue $\lambda = 3$, since

$$Ax = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix} = 3x$$

Geometrically, multiplication by A has stretched the vector x by a factor of 3 (Fig. 2.8.2)

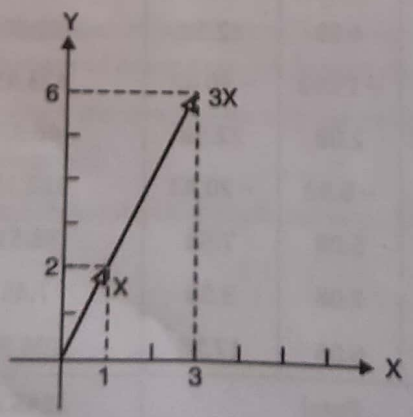


Fig. 2.8.2

Computing Eigen values and Eigen Vectors

A 2×2 matrix A has the following form :

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$ Where a, b, c and d are the elements of the matrix.

Calculating the trace and determinant

1. **Trace :** The trace of a matrix is defined as the sum of elements on the main diagonal (from upper left to lower right). It is also equal to the sum of the eigenvalues. In the case of our 2×2 matrix,

$$T = a + d$$

2. **Determinant :** The determinant of a matrix is useful in multiple further operations - for example while finding the inverse of a matrix. For a 2×2 matrix,

$$D = ad - bc$$

Determining the eigenvalues and Eigen vectors

Each 2×2 matrix A has two eigenvalues : λ_1 and λ_2. These are defined as real numbers that fulfill the following condition for a nonzero column vector $v = (v_1, v_2)$, called an eigenvector :

$$A * V = \lambda * V$$

You can also find another, equivalent version of the equation above :

$$(A - \lambda I)\, v = 0$$

where I is a 2×2 identity matrix.

Knowing the trace and determinant of the matrix A, you have to do is, input these values into the following equations :

$$\lambda_1 = \frac{T}{2} + \sqrt{\left(\frac{T^2}{4} - D\right)}$$

$$\lambda_2 = \frac{T}{2} - \sqrt{\left(\frac{T^2}{4} - D\right)}$$

Example : Finding Eigen Values

$$A = \begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix}$$

Applying the above procedure we get,

Trace

$$T = a + d$$

$$T = 3 - 1$$

$$T = 2$$

Determinant D

$$D = ad - bc$$

$$D = (3*-1) - (0*8)$$

$$D = -3$$

Finding Eigen Values

$$\lambda_1 = \frac{T}{2} + \sqrt{\left(\frac{T^2}{4} - D\right)}$$

$$\lambda_1 = \left(\frac{2}{2}\right) + \sqrt{\frac{2^2}{4} - (-3)}$$

$$\lambda_1 = 3$$

$$\lambda_2 = \frac{T}{2} - \sqrt{\left(\frac{T^2}{4} - D\right)}$$

$$\lambda_2 = \left(\frac{2}{2}\right) - \sqrt{\frac{2^2}{4} - (-3)}$$

$$\lambda_2 = -1$$

Now that we have found Eigen values we can find the Eigen vectors by the following equation

$$(A - \lambda I)\, v = 0$$

Let us use $\lambda_1 = 3$, and find the Eigen vector

Let $v = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$\begin{bmatrix} 3 & 0 \\ 8 & -1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 3-\lambda & 0 \\ 8 & -1-\lambda \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & 0 \\ 8 & -4 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

$$8x_1 - 4x_2 = 0$$

$$2x_1 - x_2 = 0$$

$$2x_1 = x_2$$

$$x_1 = \frac{1}{2} x_2$$

Let $x_2 = t$

$$v = \begin{bmatrix} \frac{1}{2}t \\ t \end{bmatrix}$$

$$v = t \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$$

$$v = \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$$

Similarly, $\lambda_2 = -1$

$$v = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

2.8.3 Steps for Computing PCA

Step 1 : Given to you a dataset for Example.

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

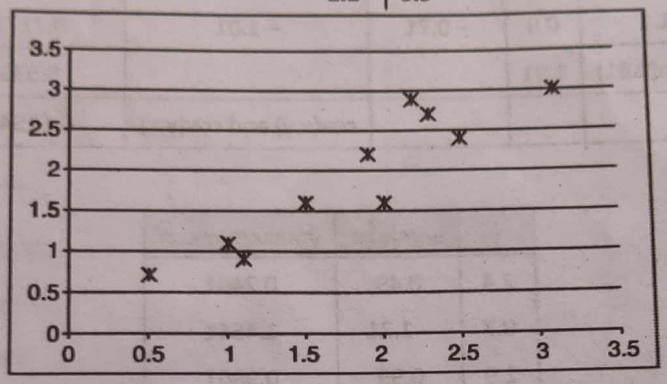


Fig. 2.8.3 : Data Plot

Step 2 : Calculate the Covariance Matrix.

The Covariance Matrix for a data set having 2 dimensions can be represented as

$$C = \begin{bmatrix} \text{cov}(x, x) & \text{cov}(x, y) \\ \text{cov}(y, x) & \text{cov}(y, y) \end{bmatrix}$$

x	X – xmean	xmean*xmean
2.5	0.69	0.4761
0.5	–1.31	1.7161
2.2	0.39	0.1521
1.9	0.09	0.0081
3.1	1.29	1.6641
2.3	0.49	0.2401
2	0.19	0.0361
1	–0.81	0.6561
1.5	–0.31	0.0961
1.1	–0.71	0.5041
1.81	Sum	5.549
	Cov (x,x)	0.61655

x	y	x – xmean	y – ymean	xmean*ymean
2.5	2.4	0.69	0.49	0.3381
0.5	0.7	– 1.31	– 1.21	1.5851
2.2	2.9	0.39	0.99	0.3861
1.9	2.2	0.09	0.29	0.0261
3.1	3	1.29	1.09	1.4061
2.3	2.7	0.49	0.79	0.3871
2	1.6	0.19	– 0.31	– 0.0589
1	1.1	– 0.81	– 0.81	0.6561
1.5	1.6	– 0.31	– 0.31	0.0961
1.1	0.9	– 0.71	– 1.01	0.7171
Mean(1.81)	1.91			5.539
			cov(x,y) and cov(y,x)	0.61544

y	y-ymean	ymean*ymean
2.4	0.49	0.2401
0.7	– 1.21	1.4641
2.9	0.99	0.9801
2.2	0.29	0.0841
3	1.09	1.1881
2.7	0.79	0.6241
1.6	– 0.31	0.0961
1.1	– 0.81	0.6561
1.6	– 0.31	0.0961
0.9	– 1.01	1.0201
1.91	average	6.449
	cov (y, y)	0.71655

From the above tables the covariance matrix is calculated,

Therefore

$$C = \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

Step 3 : Calculate the Eigen Vectors and Eigen Values from the covariance matrix

$$C = \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

$$\det(C - \lambda I) = 0$$

$$\begin{vmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{vmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = 0$$

Determinant(D) = ad – bc

Trace(T) = (sum of diagonal elements)

Det = 0.6154*0.7165 – 0.6154*0.6154)

D = 0.4409341 – 0.3787

D = 0.0622

Trace = (0.6154+0.7165) =1.3319

Knowing the trace and determinant of the matrix C, finding the eigenvalues is to input these values into the following equations :

$$\lambda_1 = \frac{T}{2} + \sqrt{\frac{T^2}{4} - D}$$

$$\lambda_2 = \frac{T}{2} - \sqrt{\frac{T^2}{4} - D}$$

$\lambda_1 = 1.28$

$\lambda_2 = 0.0485$

To find the i^{th} Eigen vector

$$Ce_i = \lambda_1 e_i$$

$$\begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix} \begin{bmatrix} e_{11} \\ e_{12} \end{bmatrix} = 1.28 \begin{bmatrix} e_{11} \\ e_{12} \end{bmatrix}$$

$0.6165\, e_{11} + 0.6154\, e_{12} = 1.28\, e_{11}$ ... (1)

$0.6154 e_{11} + 0.7165\, e_{12} = 1.28\, e_{12}$ ... (2)

From (1)

$1.28\, e_{11} - 0.6165\, e_{11} = 0.6154\, e_{12}$

$0.6635\, e_{11} = 0.6154\, e_{12}$

$$e_{11} = \frac{0.6154}{0.6635} e_{12}$$

$e_{11} = 0.9275\, e_{12}$

$$e_1 = \begin{bmatrix} 0.9275 \\ 1 \end{bmatrix}$$

Find the Euclidean length of above e_1 vector and divide the same to obtain the first Eigen vector (e_1)

Euclidean length $e_1 = \sqrt{x_1^2 + x_2^2 \ldots + x_n^2}$

Euclidean length = 1.363

Dividing e_1 elements with Euclidean length we get

Therefore e_1 = [0.6803 0.733]

Similarly applying above procedure we can obtain ,

$$e_2 = [-0.734 \quad 0.677]$$

$$\text{Eigen vectors} = \begin{bmatrix} 0.6803 & 0.733 \\ -0.734 & 0.677 \end{bmatrix}$$

Step 4 : Choosing components and forming feature vector.

The Eigen vector with the highest Eigen value is the principle component of dataset.

$$\text{Eigen vectors} = [0.6803 \quad 0.733]$$

Reduce dimensionality and form feature vector. The eigenvector with the highest eigenvalue is the principal component of the data set.

2.9 Non-negative Matrix Factorization

- NMF is a collection of algorithms in the field of linear algebra and multivariate data analysis.
- Consider a matrix V. This matrix V can be factored into two matrices W and H such that

$$WH = V$$

- The dimensions of the matrix are V is $m \times n$, W is $m \times k$ and H is $k \times n$.
- The matrix W and H are known as Factor matrices. These factor matrices are non negative, i.e. all the elements of factor matrices are equal or greater than zero.
- One of the early work was done by Paatero and Tapper, 1994 (called positive matrix factorization).
- The name Non Negative Matrix Factorization (NMF) was coined by Lee and Seung in 1999.
- There is very little theoretical work on NMF : Donoho 2004 studied when factorization is unique, etc. Otherwise, only fixed point convergence has been shown for some algorithms.

NMF and low rank modelling

- The statistical analysis of multivariate data using NMF is as follows :
- Consider a multivariate n-dimensional data vectors, place these vectors in the columns of a n x m matrix V. m represents the number of examples in the data.
- The matrix V can be factorized in to two matrices W and H. W having dimensions n x k and H having dimensions k x m
- The size of W and H matrix have to smaller than the original matrix V, so k is usually chosen to be smaller than n or m.
- The result of NMF is a compressed version of the original data matrix.
- Low rank approximation $V_k = W_k H_k$ where all matrices are non-negative.
- The factor matrices may have low rank than the matrix V or it can be approximated by V_k (rank k approximation of V).
- A linear combination of column vectors in W and the coefficients given by elements of H are used to compute each column of V,

$$V_i = \sum_{j=1}^{N} H_{ji} W_j$$

- The column vectors of W can be considered as basis vectors of vector space defined by V.
- Columns of V are build from k columns of W

$$W_k H_{*1} = \begin{bmatrix} \vdots \\ w_1 \\ \vdots \end{bmatrix} h_{11} + \begin{bmatrix} \vdots \\ w_2 \\ \vdots \end{bmatrix} h_{21} + \ldots + \begin{bmatrix} \vdots \\ w_k \\ \vdots \end{bmatrix} h_{k1}$$

- The representation is additive since W_k and H_k are non-negative.
- A few basis vectors represent many data vectors, good approximation can be achieved only if these basis vectors discover structure latent in the data.

NMF factor interpretation

1. **Non-negative factors give benefits in interpretation**
   - Text processing requiring factorization of term by document matrix V_{mxn}, k can be considered the number of topics present in the document collection.
   - Element i, j of V tells how many times i^{th} word in vocabulary appears in document j.
   - W is a term by topic matrix whose columns are the basis vectors.
   - The nonzero elements of column 1 of W correspond to particular terms. Considering the highest weighted terms in this vector, one can assign label or topic to the basis vector.
2. **NMF interpretation**
   - Consider the field of image processing, the blocks in W and H matrix are sparse, where matrix H represents weight.
   - For example, in image database of face, each face image (obtained by stacking image into column vector) is a column in V.
   - W contains basis images and H their weights.
   - Basis images correspond to parts of a face such as eyes, nose, lips, hence **additive representation.**

NMF Applications

NMF can be used in a number of fields like

(i) Image Processing

(ii) Text processing and mining

(iii) Music transcription

(iv) Video analysis

(v) Bioinformatics

(vi) Chemistry

2.10 Sparse PCA

- Sparse PCA is a variant of Classical Principal Component Analysis technique.
- The classical PCA dimensionality reduction tool makes use of near orthogonal vectors to find linear combination of a small number of features that maximizes the variance across data.

- Need for Sparse PCA :
  - The features obtained using PCA are not sparse, thus they are difficult to interpret.
  - Sparse PCA improves the interpretability and relevance of the components; it also shows the structure of data. As the features in real time applications have a concrete physical meaning (e.g. sensors, people, genes).
  - Sparse components can be computed faster.
  - Sparse components provides better statistical regularization.

2.10.1 Sparse Principal Component Analysis using the Lasso (Elastic Net)

- In this technique Standard PCA is formulated as a regression-type optimization problem.
- The sparse loadings can be obtained by imposing the lasso (elastic net) constraint on the regression coefficients.

General SPCA Algorithm

1. Let α start at V[, 1:k] , the loading of first k ordinary principal components.
2. Given fixed α solve the following naïve elastic net problem for j = 1 ,2,...k

$$\beta_j = \arg\min_{\beta^*} \beta^{*T}(X^T X + \lambda)\beta^* - 2\alpha_j^T X^T X\beta^* + \lambda_{1,j}|\beta^*|_1$$

3. For each β do the SVD of $X^T X\beta = UDV^T$, then update $\alpha = UV^T$
4. Repeat steps 2-3 until β converges.
5. Normalization of

$$V_i = \frac{\beta_i}{|\beta_i|}, \; j = 1, 2, \ldots, k$$

2.11 Kernel PCA

- Standard PCA is used for Linear dimensionality reduction.
- Incase if the data has more complicated structures and cannot be represented in a linear sub space, standard PCA will not be suitable. In such cases, Kernel PCA may used.
- Kernel PCA generalizes standard PCA to non linear dimensionality reduction.

Basic Idea of Kernel method

- A kernel method is used for mapping data to higher dimensions (often) where the data is linear.
- In this transformation data becomes linearly separable in the new feature space.
- Feature mapping $\phi = R^2 \rightarrow R^3$

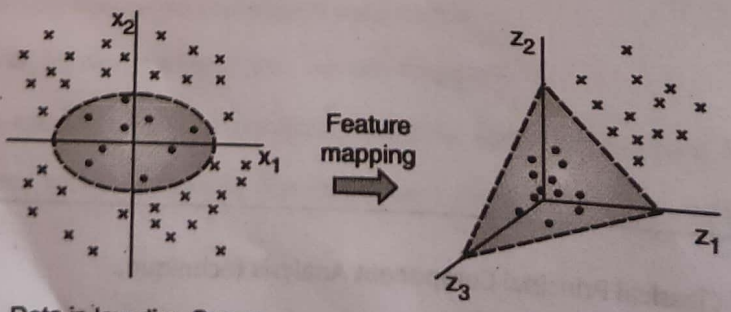


Fig. 2.11.1

Some Popular Kernel functions

1. **Polynomial Kernel function**

$$k(x, y) = (x^T y + c)^d , \; d \in Z^+$$

2. **Gaussian Kernel function**

$$k(x, y) = \exp\left(\frac{-|x-y|^2}{2\sigma^2}\right)$$

Deriving Kernel PCA

- To project the data to a high-dimension feature space k

$$\phi = x \rightarrow H , \; x \in R^d , \quad H \in R^D \text{ and } d << D$$

- Compute the covariance matrix of data in feature space

$$C_F = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) \phi(x_i)^T$$

- Compute the principal components by solving eigenvalue problem

$$C_F v = \lambda v$$

- Eigen vector can be expressed as linear combination of features

$$v = \sum_{i=1}^{N} \alpha_i \phi(x_i)$$

Proof :

$$C_F v = \lambda v = \sum_{i=1}^{N} \phi(x_i) \phi(x_i)^T v$$

$$v = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) \phi(x_i)^T v = \sum_{i=1}^{N} \frac{\phi(x_i)^T}{N\lambda} \phi(x_i) = \sum_{i=1}^{N} \alpha_i \phi(x_i)$$

Multiply $\phi(x_i)$ to both the sides of $\lambda v = C_F v$

$$\lambda [\phi(x_k) v] = [\phi(x_k) C_F v]$$

$$\lambda \sum_{i=1}^{N} \alpha_i \phi(x_k) \phi(x_i) = \frac{1}{N} \sum_{i=1}^{N} \alpha_i (\phi(x_k) \sum_{j=1}^{N} \phi(x_j)) (\phi(x_j) \phi(x_i))$$

We define matrix K as $K_{ij} := (\phi(x_i) \cdot \phi(x_j))$

Then we will get $N\lambda K\alpha = K^2 \alpha$

Which can be solved via the eigen - decomposition.

$$N\lambda\alpha = K\alpha$$

Normalizing the feature space

Generally, $\phi(x_i)$ may not be zero mean. Thus, we want to centre feature.

$$\phi(\hat{x}_k) = \phi(x_i)\frac{1}{N}\sum_{k=1}^{N}\phi(x_k)$$

The corresponding kernel is :

$$\hat{K}(x_i, x_j) = \hat{\phi}(x_i)^T\hat{\phi}(x_j) = (\phi(x_i) - \frac{1}{N}\phi(x_k))^T(\phi(x_j) - \frac{1}{N}\phi(x_k))$$

$$= K(x_i, x_j) - \frac{1}{N}\sum_{i=1}^{N}K(x_i, x_k) - \frac{1}{N}\sum_{i=1}^{N}K(x_j, x_k) + \frac{1}{N^2}\sum_{l,k}^{N}K(x_l, x_k)$$

Thus K is the centered kernel function

Steps in Kernel PCA Construction

1. Pick up one kernel function.
2. Construct the normalized kernel matrix of the data.

$$\hat{K} = K - 2I_{\frac{1}{N}}K + I_{\frac{1}{N}}KI_{\frac{1}{N}}$$

3. To solve an eigen value problem.

$$K\alpha_i = \lambda_i\alpha_i$$

4. At last, we can represent the outputs as :

$$y_j = \sum_{i=1}^{N}\alpha_{ji}K(x, x_i), \quad j = 1, \ldots, d$$

2.12 Atom Extraction and Dictionary Learning

- Dictionary learning is a technique which allows rebuilding a sample starting from a sparse dictionary of atoms (similar to principal components).

$$X = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\} \text{ where } \bar{x}_i \in R^m$$

- Is an input dataset and the target is to find both a dictionary D and a set of weights for each sample :

$$D \in R^{m \times k} \text{ and } A = \{\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_m\} \text{ where } \bar{\alpha}_i \in R^k$$

- After the training process, an input vector can be computed as :

$$\bar{x}_i = D\bar{\alpha}_i$$

- The optimization problem (which involves both D and alpha vectors) can be expressed as the minimization of the following loss function :

$$L(D, A) = \frac{1}{2}\sum_i ||x_i - D\bar{\alpha}_i||_2^2 + c||\bar{\alpha}_i||_1$$

- Here the parameter c controls the level of sparsity (which is proportional to the strength of L1 normalization). This problem can be solved by alternating the least square variable until a stable point is reached.

Review Questions

Q. 1 Explain the concept of Feature selection in machine learning

Q. 2 Explain the different types of datasets with scikit learn library

Q. 3 With the help of an example explain the different types of data

Q. 4 Explain the different methods of managing categorical data

Q. 5 Explain data scaling and normalization

Q. 6 Explain feature selection and filtering

Q. 7 Explain Principal component analysis. Support your answer with a suitable example

Q. 8 Write short notes on

(i) Non negative Matrix factorization

(ii) Sparse PCA

(iii) Kernel PCA

□□□