RGPVNOTES.IN

Program : **B.E**

Subject Name: **Machine Learning**

Subject Code: **CS-8003**

Semester: **8th**

Introduction to Dimensionality Reduction

Machine Learning: As discussed in this article, machine learning is nothing but a field of study which allows computers to "learn" like humans without any need of explicit programming.

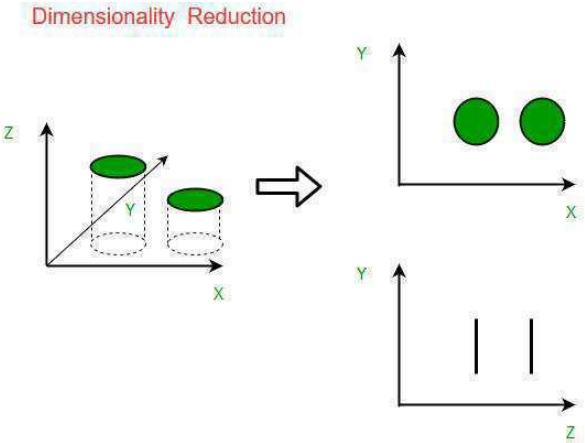
What is Predictive Modeling: Predictive modeling is a probabilistic process that allows us to forecast outcomes, on the basis of some predictors. These predictors are basically features that come into play when deciding the final result, i.e. the outcome of the model.

What is Dimensionality Reduction?

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

Why is Dimensionality Reduction important in Machine Learning and Predictive Modeling?

An intuitive example of dimensionality reduction can be discussed through a simple e-mail classification problem, where we need to classify whether the e-mail is spam or not. This can involve a large number of features, such as whether or not the e-mail has a generic title, the content of the e-mail, whether the e-mail uses a template, etc. However, some of these features may overlap. In another condition, a classification problem that relies on both humidity and rainfall can be collapsed into just one underlying feature, since both of the aforementioned are correlated to a high degree. Hence, we can reduce the number of features in such problems. A 3-D classification problem can be hard to visualize, whereas a 2-D one can be mapped to a simple 2 dimensional space, and a 1-D problem to a simple line. The below figure illustrates this concept, where a 3-D feature space is split into two 1-D feature spaces, and later, if found to be correlated, the number of features can be reduced even further.



Components of Dimensionality Reduction

There are two components of dimensionality reduction:

- **Feature selection:** In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:
  1. Filter
  2. Wrapper
  3. Embedded
- **Feature extraction:** This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

Methods of Dimensionality Reduction

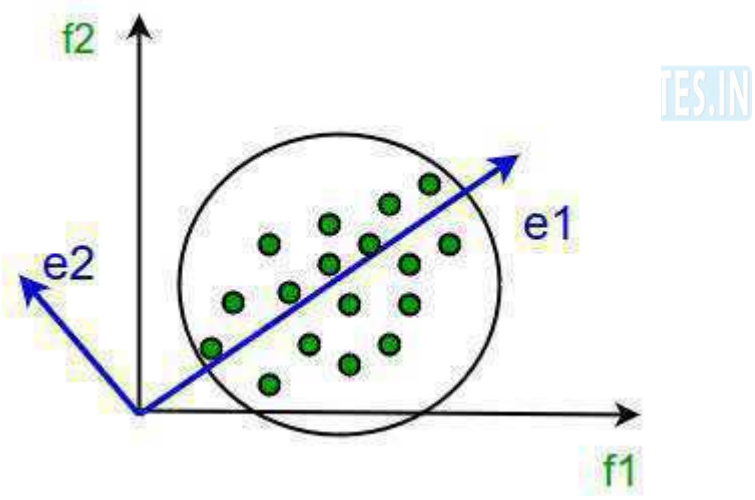
The various methods used for dimensionality reduction include:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

Dimensionality reduction may be both linear or non-linear, depending upon the method used. The prime linear method, called Principal Component Analysis, or PCA, is discussed below.

Principal Component Analysis

This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.



It involves the following steps:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.

Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

Advantages of Dimensionality Reduction

- It helps in data compression, and hence reduced storage space.
- It reduces computation time.
- It also helps remove redundant features, if any.

Disadvantages of Dimensionality Reduction

- It may lead to some amount of data loss.
- PCA tends to find linear correlations between variables, which is sometimes undesirable.
- PCA fails in cases where mean and covariance are not enough to define datasets.
- We may not know how many principal components to keep- in practice, some thumb rules are applied.

2.3.1 Signal Feature Extractions

Feature extraction is the transformation of original data to a data set with a reduced number of variables, which contains the most discriminatory information. This will reduce the data dimensionality, remove redundant or irrelevant information, and transform it to a form more appropriate for subsequent classification [32]. Improved classification performance may be yielded through a more stable representation. By reducing the bandwidth of the input data, improved processing speed is achievable. Simple extraction methods are based on data reduction procedures in order to obtain scalar features, for example, maximum amplitude. An example of this is the conventional technique that exploits the peak height and arrival time. Advanced techniques lead to pattern or vectorial representations [33].

With regards to feature extraction, it has been shown that the simple, conventional approach using differential signal peak characteristics is generally not robust enough to be applied in the real field. In addition, this approach only offers a limited number of features, which might mean that the whole potential of PEC that is rich in frequency components has not been extracted optimally. A more effective feature extraction technique is required before PEC can be brought into applications in the NDT world.

Two advanced feature extraction techniques that utilize both temporal and spectral information of the signal will be discussed in the next section. The techniques are based on PCA and wavelet analysis.

Pattern recognition is a technique that studies a given pattern and determines the class membership of the pattern. A pattern is generally comprised of a vector of measurement results, $\boldsymbol{x} = (x_1, \ldots, x_n)^T$. A given vector will be associated with one of C classes, therefore it can be said that the main theme is *classification*.

A classifier is derived by training. Based on the training approach, two categories exist: supervised and unsupervised. In supervised classification, the possible classes are known and defined by the investigator, while in

unsupervised classification the classifier will create the possible classes from the given training data sets. The former is generally called *discrimination* and the latter is referred to as *classification* or *clustering* [32].
Part of classification is a stage where features, that are considered to be significant, are selected from the data vector ***x***. The dimension of the selected features is smaller than the dimension of the original data vector. This step is commonly called feature extraction, which is vital for a successful classification. By putting less relevant features aside, further processing can be performed faster and more effectively without losing information for classification of a pattern.

Feature ranking and subset selection

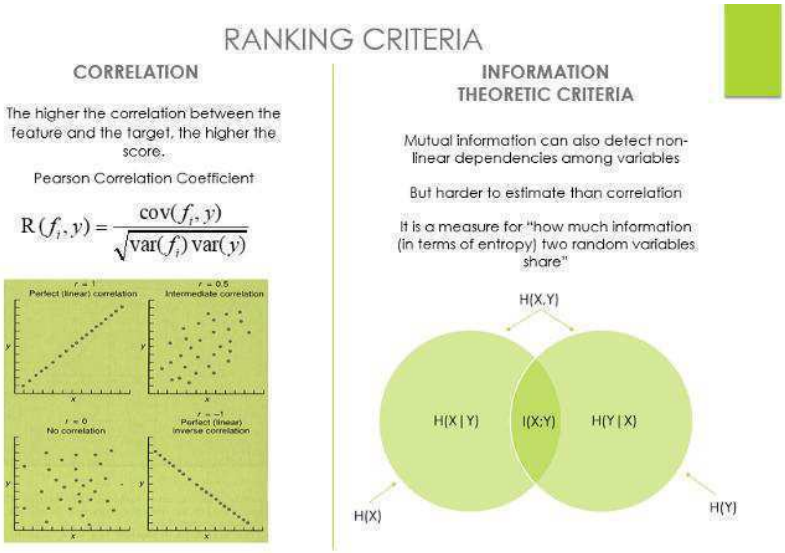
Topic 1: Variable Ranking

Variable Ranking is the process of ordering the features by the value of some scoring function, which usually measures feature-relevance.

Resulting set: The score *S(fi)* is computed from the training data, measuring some criteria of feature *fi*. By convention a high score is indicative for a valuable (relevant) feature.

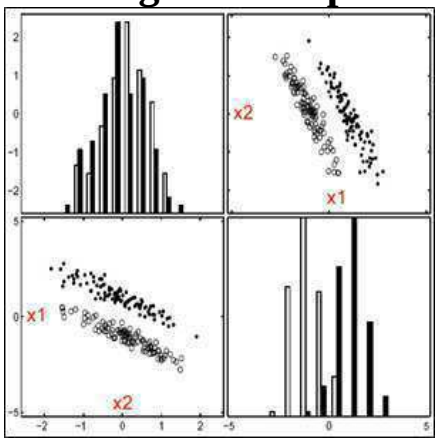
A simple method for *feature selection* using **variable ranking** is to select the *k*highest ranked features according to *S*. This is usually not optimal, but often preferable to other, more complicated methods. It is computationally efficient—only calculation and sorting of *n* scores.

Ranking Criteria: Correlation Criteria and Information Theoretic Criteria



Variable Ranking Criteria or Feature Ranking Criteria: Correlation Criteria and Information Theoretic Criteria

Ranking Criteria poses some questions:



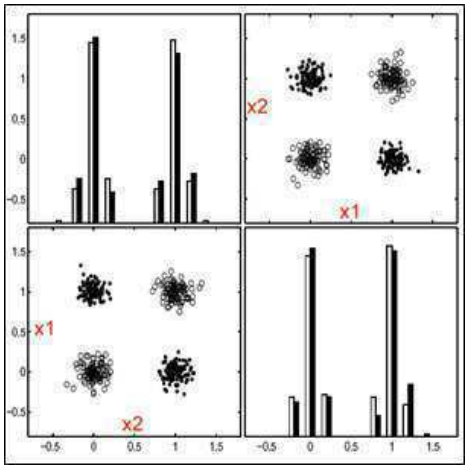
Can variables with small score be automatically discarded? NO!

1. ***Can variables with small score be automatically discarded?***

The answer is **NO!**

- Even variables with small score can improve class seperability
- Here, this depends on the correlation between *x1* and *x2*

Here, the class conditional distributions have a high co-variance in the direction orthogonal to the line between the two class centers.



Can a useless variable (i.e. one with a small score) be useful together with others? YES!

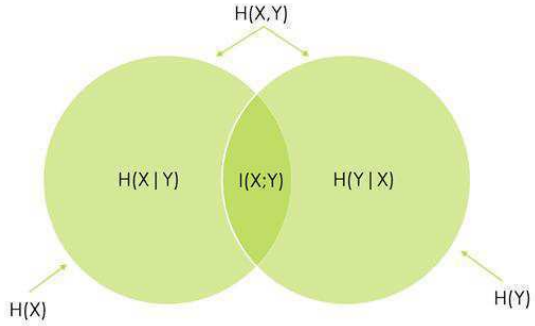
2. *Can a useless variable (i.e. one with a small score) be useful together with others?*

The answer is **YES!**

• The correlation between variables and target are not enough to assess relevance

• The correlation / co-variance between pairs of variables has to be considered too (potentially difficult)

- Also, the **diversity** of features needs to be considered.



Information Theoretic Criteria

3. *Can two variables that are useless by themselves can be useful together?*

The answer is **YES!**

This can be done using the Information Theoretic Criteria.

Information Theoretic Criteria

- Mutual information can also detect non-linear dependencies among variables
- But, it is harder to estimate than correlation
- It is a measure for "how much information (in terms of entropy) two random variables share".

Variable Ranking—Single Variable Classifiers

- Idea: Select variables according to their **individual predictive power**
- Criterion: Performance of a classifier built with 1 variable e.g. the value of the variable itself
- The Predictive power is usually measured in terms of error rate (or criteria using False Positive Rate, False Negative Rate)
- Also, a combination of SVC's can be deployed using ensemble methods (boosting,…).

Topic 2: Feature Subset Selection

The Goal of **Feature Subset Selection** is to find the optimal feature subset. **Feature Subset Selection** Methods can be classified into three broad categories

- Filter Methods
- Wrapper Methods
- Embedded Methods

For **Feature Subset Selection** you'd need:

- A measure for assessing the goodness of a feature subset (scoring function)
- A strategy to search the space of possible feature subsets
- Finding a minimal optimal feature set for an arbitrary target concept is hard. It would need Good Heuristics.

Filter Methods

- In this method, select subsets of variables as a pre-processing step, independently of the used classifier
- It would be worthwhile to note that **Variable Ranking-*Feature Selection***is a Filter Method.

Filter Methods: Feature Subset Selection Method

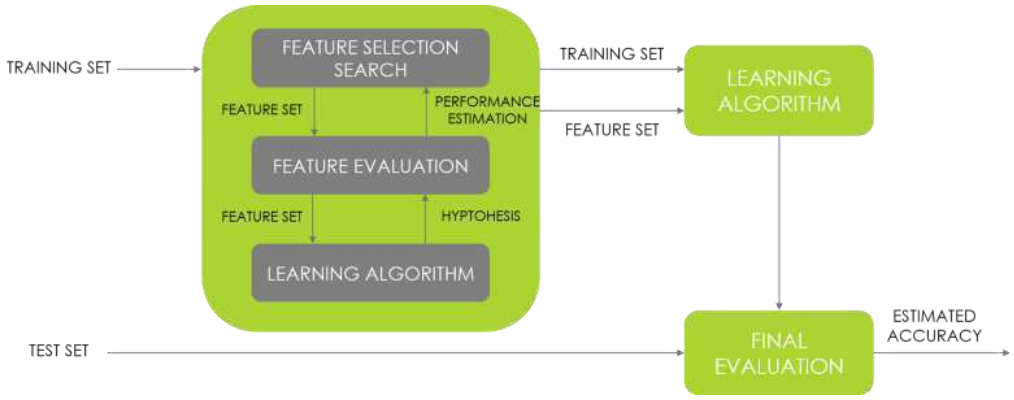
Key features of Filter Methods for **Feature Subset Selection:**

- Filter Methods are usually fast
- Filter Methods provide generic selection of features, not tuned by given learner (universal)
- Filter Methods are also often criticized (feature set not optimized for used classifier)
- Filter Methods are sometimes used as a pre-processing step for other methods.

Wrapper Methods

- In Wrapper Methods, the Learner is considered a black-box. Interface of the black-box is used to score subsets of variables according to the predictive power of the learner when using the subsets.

- Results vary for different learners
- One needs to define: – how to search the space of all possible variable subsets ?– how to assess the prediction performance of a learner ?



Wrapper Methods: Feature Subset Selection Method

Embedded Methods

- Embedded Methods are specific to a given learning machine
- Performs variable selection (implicitly) in the process of training
- E.g. WINNOW-algorithm (linear unit with multiplicative updates).

Filter – fearture selection me khi to h

Big data and map reduce – net

Intro to real world ml –

How to choose machine learning algorithm

This is a quick review on the important considerations when choosing machine learning algorithms:

Type of problem: It is obvious that algorithms have been designd to solve specific problems. So, it is important to know what type of problem we are dealing with and what kind of algorithm works best for each type of problem. I don't want to go into much detail but at high level, machine learning algorithms can be classified into Supervised, Unsupervised and Reinforcement learning. Supervised learning by itself can be categorized into Regression, Classification, and Anomoly Detection.

Size of training set: This factor is a big player in our choice of algorithm. For a small training set, high bias/low variance classifiers (e.g., Naive Bayes) have an advantage over low bias/high variance classifiers (e.g., kNN), since the latter will overfit. But low bias/high

variance classifiers start to win out as training set grows (they have lower asymptotic error), since high bias classifiers aren't powerful enough to provide accurate models [1].

Accuracy: Depending on the application, the required accuracy will be different. Sometimes an approximation is adequate, which may lead to huge reduction in processing time. In addition, approximate methods are very robust to overfitting.

Training time: Various algorithms have different running time. Training time is normally function of size of dataset and the target accuracy.

Linearity: Lots of machine learning algorithms such as linear regression, logistic regression, and support vector machines make use of linearity. These assumptions aren't bad for some problems, but on others they bring accuracy down. Despite their dangers, linear algorithms are very popular as a first line of attack. They tend to be algorithmically simple and fast to train.

Number of parameters: Parameters affect the algorithm's behavior, such as error tolerance or number of iterations. Typically, algorithms with large numbers parameters require the most trial and error to find a good combination. Even though having many parameters typically provides greater flexibility, training time and accuracy of the algorithm can sometimes be quite sensitive to getting just the right settings.

Number of features: The number of features in some datasets can be very large compared to the number of data points. This is often the case with genetics or textual data. The large number of features can bog down some learning algorithms, making training time unfeasibly long. Some algorithms such as Support Vector Machines are particularly well suited to this case [2,3].

Below is an algorithm cheatsheet provided by scikit-learn (works as rule of thumb), which I believe it has implicitely considered all the above factors in making recommendation for choosing the right algorithm. But it doesn't work for all situations and we need to have a deeper understanding of these algorithms to employ the best one for a unique problem.

design and analysis of machine learning experiments – nhi mila

common software for ml

1. AMAZON WEB SERVICES

Amazon Web Services (AWS) comes with several AI toolkits for developers. For example, AWS Rekognition utilizes AI to build image interpretation and facial recognition into apps with common biometric security features.

Furthermore, AWS Lex is the open source tool behind Amazon's personal assistant Alexa. This technology enables developers to integrate chatbots into mobile and web applications. AWS Polly, on the other hand, utilizes AI to automate voice to written text in 24 languages and 47 voices.

2. AI-ONE

This is a tool that enables developers to build intelligent assistants within almost all software applications. Often referred to as biologically inspired intelligence, ai-one's Analyst Toolbox is equipped with the following:

- APIs
- building agents
- document library

The primary benefit of this tool is the ability to turn data into generalized sets of rules that enable in-depth ML and AI structures.

3. DEEPLEARNING4J

Deeplearning4j or Deep Learning for Java is a leading open source deep learning (DL) library written for Java and Java Virtual Machine (JVM). It's specifically designed to run on enterprise applications such as Apache Spark and Hadoop.
It also includes the following:

- Boltzmann machine
- Deep autoencoder
- Deep belief net
- Doc2vec
- Recursive neural tensor network
- Stacked denoising autoencoder
- Word2vec

4. APACHE MAHOUT

This is a library of scalable ML algorithms that can be implemented on top of Apache Hadoopby utilizing the MapReduce paradigm. As a result, once all the big data is stored on Hadoop Distributed File System (HDFS), you can use the data science tools provided by Apache Mahoutto identify valuable patterns in those big data sets.
The primary advantage of the Apache Mahout project is that it makes it much easier and faster to derive real value from big data.

5. OPEN NEURAL NETWORKS LIBRARY (OPENNN)

This is another open-source tool that's essentially a class library written in the programming language C++ for SL that is utilized to stimulate neural networks. With this OpenNN tool, you can implement neural networks that are characterized by high performance and deep architecture.

Some other open source AI and ML tools to consider are as follows:

- Distributed Machine Learning Toolkit (Microsoft)
- NuPIC
- Oryx 2

You can expect more AI and ML tools to hit the market in the near future to keep up with rapid development within this space. As Canada continues to grow as an innovative hub for AI, you can also expect more cutting-edge intelligent technology to come out of North America.

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in