**A comprehensive beginners guide to Linear Algebra for Data Scientists**

## Introduction

One of the most common questions we get on Analytics Vidhya is,

*How much maths do I need to learn to be a data scientist?*

Even though the question sounds simple, there is no simple answer to the the question. Usually, we say that you need to know basic descriptive and inferential statistics to start. That is good to start.

But, once you have covered the basic concepts in machine learning, you will need to learn some more math. You need it to understand how these algorithms work. What are their limitations and in case they make any underlying assumptions. Now, there could be a lot of areas to study including algebra, calculus, statistics, 3-D geometry etc.

If you get confused (like I did) and ask experts what should you learn at this stage, most of them would suggest / agree that you go ahead with **Linear Algebra.**

But, the problem does not stop there. The next challenge is to figure out how to learn Linear Algebra. You can get lost in the detailed mathematics and derivation and learning them would not help as much! I went through that journey myself and hence decided to write this comprehensive guide.

If you have faced this question about how to learn & what to learn in Linear Algebra – you are at the right place. Just follow this guide.

And if you’re looking to understand where linear algebra fits into the overall data science scheme, here’s the perfect article:

# 10 Powerful Applications of Linear Algebra in Data Science (with Multiple Resources)

## Overview

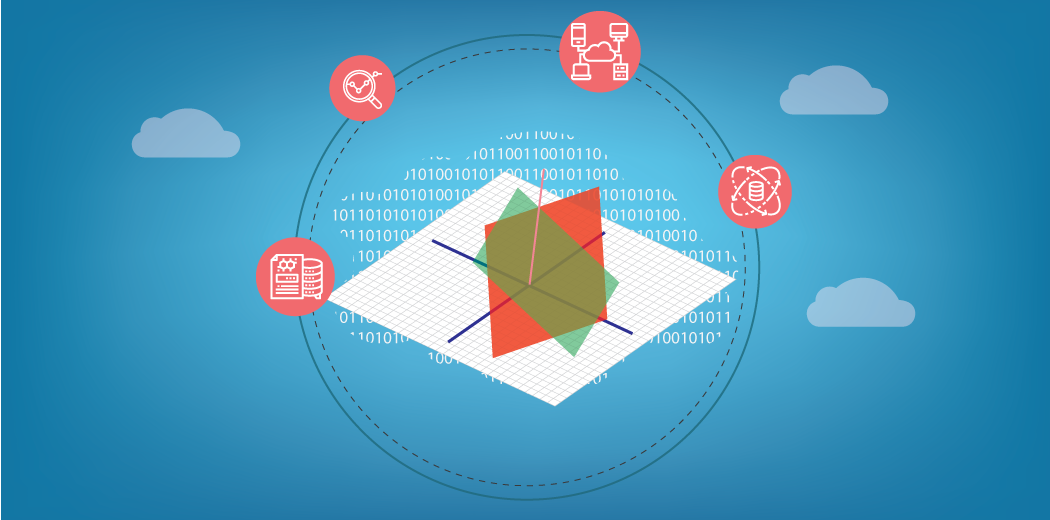
* Linear algebra powers various and diverse data science algorithms and applications
* Here, we present 10 such applications where linear algebra will help you become a better data scientist
* We have categorized these applications into various fields – Basic Machine Learning, Dimensionality Reduction, Natural Language Processing, and Computer Vision

## Introduction

If Data Science was Batman, Linear Algebra would be Robin. This faithful sidekick is often ignored. But in reality, it powers major areas of Data Science including the hot fields of Natural Language Processing and Computer Vision.

I have personally seen a LOT of data science enthusiasts skip this subject because they find the math too difficult to understand. When the programming languages for data science offer a plethora of packages for working with data, people don’t bother much with linear algebra.

That’s a mistake. Linear algebra is behind all the powerful machine learning algorithms we are so familiar with. It is a vital cog in a data scientists’ skillset. As we will soon see, you should consider linear algebra as a must-know subject in data science.



And trust me, Linear Algebra really is all-pervasive! It will open up possibilities of working and manipulating data you would not have imagined before.

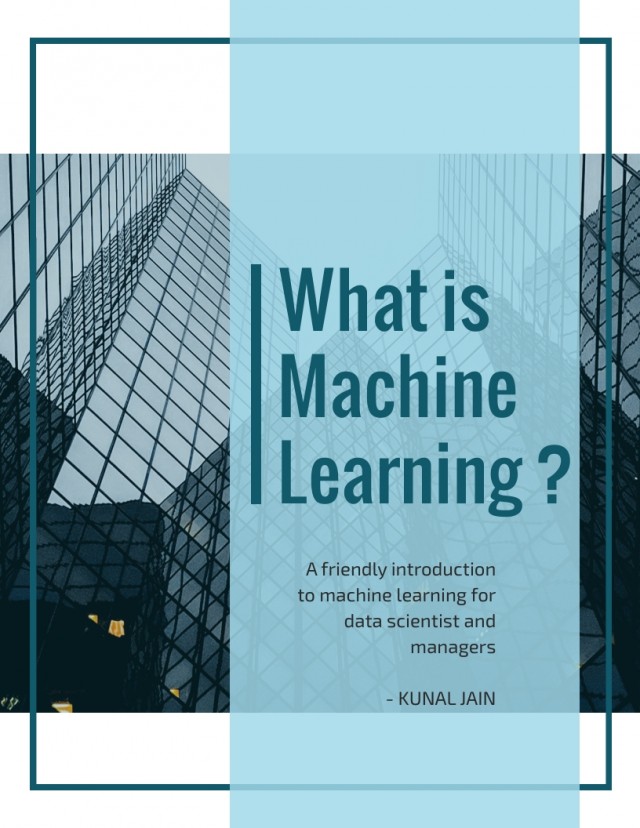
In this article, I have explained in detail ten awesome applications of Linear Algebra in Data Science. I have broadly categorized the applications into four fields for your reference:

* Machine Learning
* Dimensionality Reduction
* Natural Language Processing ( NLP )
* Computer Vision

Applied Machine Learning- Beginner to Professional

This Course provides you all the tools and techniques you need to apply machine learning to solve business problems. We will cover the basics of machine learning, how to build machine learning models, improve and deploy your machine learning models.

**The Mini-Book on Machine Learning**



### MAbout Applied Machine Learning - Beginner to Professional Course

Machine Learning is re-shaping and revolutionising the world and disrupting industries and job functions globally. It is no longer a buzzword - many different industries have already seen automation of business processes and disruptions from Machine Learning. In this age of machine learning, every aspiring data scientist is expected to upskill themselves in machine learning techniques & tools and apply them in real-world business problems.

**Key Takeaways from this course:**

* Understand how Machine Learning and Data Science are disrupting multiple industries today.
* Linear, Logistic Regression, Decision Tree and Random Forest algorithms for building machine learning models.
* Understand how to solve Classification and Regression problems in machine learning
* Ensemble Modeling and techniques like Bagging and Boosting
* Support Vector Machines (SVM) and Kernel Tricks
* Prior to building your machine learning model, learn how to reduce dimensions using techniques like Principal Component Analysis (PCA) and t-SNE
* How to evaluate your machine learning models and improve them through Feature Engineering
* Learn Unsupervised Machine Learning Techniques like k-means clustering and Hierarchical Clustering
* Learn how to work with different kinds of data for machine learning problems (tabular, text, unstructured)
* Improve and enhance your machine learning model’s accuracy through feature engineering

**Pre-requisites for the Applied Machine Learning course**

This course requires no prior knowledge about Data Science or any tool.

### Know more about Applied Machine Learning

### Course curriculum

##### **1.Welcome to the course**

* DataHack Summit 2019 - India’s largest Applied Artificial Intelligence and Machine Learning Conference

##### **2.Welcome to the Applied Machine Learning Course**

* Welcome to the Course

##### **3.Introduction to Data Science and Machine Learning**

* Overview of Machine learning and Data Science
* Common Terminology used in Data Science
* Applications of Data Science

##### **4.Introduction to the Course**

* Instructor Introduction
* [Overview of the](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial) **course**
* Course Handouts

##### **5.Setting up your system**

* Installation steps for Windows
* Installation steps for Linux
* Installation steps for Mac

##### **6.Python for Data Science**

* Brief Introduction to Python
* Quiz: Introduction to Python
* Theory of Operators
* Understanding Operators in Python
* Quiz: Theory of Operators
* Understanding variables and data types
* Variables and Data Types in Python
* Quiz: Understanding variables and data types
* Understanding Conditional Statements
* Implementing Conditional Statements in Python
* Quiz: Conditional Statements
* [Understanding Looping Constructs](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* Implementing Looping Constructs in Python
* Quiz: Looping Constructs
* Understanding Functions
* Implementing Functions in Python
* Quiz: Functions in Python
* A brief introduction to data structure
* Quiz: Data Structure
* Understanding the concept of Lists
* Implementing Lists in Python
* Quiz: Lists in Python
* Understanding the concept of Dictionaries
* Implementing Dictionaries in Python
* Quiz: Dictionaries in Python
* Understanding the concept of Standard Libraries
* Quiz: Standard Libraries
* Reading a CSV File in Python - Introduction to Pandas
* Reading a CSV file in Python - Implementation
* Quiz: Reading a csv file in Python
* Understanding dataframes and basic operations
* Reading dataframes and conduct basic operations in Python
* Quiz: DataFrames and basic operations
* Indexing a Dataframe
* Quiz: Indexing DataFrames
* Exercise
* Instructions
* Quiz
* Python Coding Challenge

##### **7.Statistics For Data Science**

* Introduction to statistics
* Mode of the data
* Understanding the various variable types
* Quiz: Understanding Variable Types
* Mean of the data
* Outliers in the datasets
* Quiz: Outlier in the datasets
* Median of the dataset
* Quiz: Mode , Mean and Median
* Spread of the data
* Quiz: Spread of the data
* Variance of the data
* Quiz: Variance of the data
* Standard Deviation of the data
* Quiz: Standard Deviation of the data
* Frequency Tables
* Quiz: Frequency Tables
* Histograms
* Quiz: Histograms
* Introduction to Probability
* Quiz: Introduction to probability
* Calculating Probabilities of events
* Quiz: Calculating Probabilities of events
* Bernoulli Trials and Probability Mass Function
* Quiz: Bernoulli Trials and PMF
* Probabilities for Continuous Random Variables
* Quiz: Probabilities for continuous random variable
* The Central Limit Theorem
* Quiz: Central Limit Theorem
* Properties of the Normal Distribution
* Quiz: Properties of Normal distribution
* Using the Normal Curve for Calculations
* Quiz: Normal Curve for calculations
* Z score Part 1
* Understanding the Z tables
* Quiz: Z scores
* Z score part 2
* Introduction to Inferential Statistics
* Quiz: Introduction to Inferential Statistics
* Short Review
* Quiz: Review
* Mean Estimation
* Confidence Interval and Margin of Error
* Quiz: CI and Margin of error
* Introduction to Hypothesis Testing
* Quiz: Hypothesis testing
* Steps to perform hypothesis testing
* Directional Non Directional hypothesis
* Quiz: Directional and Non Directional hypothesis
* Understanding Errors while Hypothesis Testing
* Quiz: Errors while Hypothesis testing
* Understanding T tests
* Quiz: Understanding T tests
* Degree of Freedom
* T-Critical Value
* Quiz: T-Critical Value
* Steps to perform T-Test
* Quiz: Steps to perform T-Test
* Conducting One sample T test
* Quiz: One sample T tests
* Paired T tests
* Quiz: Paired T tests
* 2 Sample T tests
* Quiz: 2 sample T tests
* Chi Squared Tests
* Quiz: Chi squared tests
* Correlation
* Quiz: Correlation
* Conclusion
* Module Test
* Instructions
* Quiz
* Statistics Coding Challenge

##### **8.Basics Steps of Machine Learning and EDA**

* Introduction to Predictive Modeling
* Quiz: Introduction to Predictive Modeling
* Types of Predictive Models
* Quiz: Types of Prediction Models
* [Stages of Predictive Modeling](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* Quiz: Stages of Predictive Modeling
* Understanding Hypothesis Generation
* Quiz: Hypothesis Generation
* Data Extraction
* Understanding Data Exploration
* Quiz: Data Extraction and Exploration
* Reading the data into Python
* Reading the data into Python : Implementation
* Quiz: Reading Data into Python
* Variable Identification
* Variable Identification : Implementation
* Quiz: Variable Identification
* Univariate analysis for Continuous Variables
* Univariate Analysis for Continuous Variables : Implementation
* Quiz: Univariate analysis for Continuous variables
* Understanding Univariate Analysis for Categorical Variables
* Univariate analysis for Categorical Variables : Implementation
* Quiz: Univariate Analysis for Categorical Variables
* Understanding Bivariate Analysis
* Quiz: Bivariate Analysis
* Bivariate Analysis : Implementation
* Quiz: Bivariate Analysis - Implementation
* Understanding and treating missing values
* Quiz: Treating missing values
* Treating missing values : Implementation
* Quiz: Treating missing values - Implementation
* Understanding Outlier Treatment
* Quiz: Outlier treatment
* Outlier Treatment in Python
* Quiz: Outlier Treatment in Python
* Understanding Variable Transformation
* Quiz: Transforming variables
* Variable Transformation in Python
* Quiz: Variable Transformation in Python
* Basics of Model Building
* Quiz: Basics of Model Building

##### **9.Data Manipulation and Visualization**

* Sorting Dataframes
* Merging Dataframes
* Quiz: Sorting and Merging dataframes
* Apply function
* Aggregating data
* Quiz: Aggregating data and Apply function
* Basics of Matplotlib
* Data Visualization using Matplotlib
* Quiz: Matplotlib
* Basics of Seaborn
* Data Visualization using Seaborn
* Quiz: Seaborn

##### **10.Project: EDA - Customer Churn Analysis**

* Understanding the Problem Statement
* Understanding the Data
* Understanding the NYC Taxi Trip Duration Problem
* Assignment: EDA

##### **11.Share your Learnings**

* Write for Analytics Vidhya's Medium Publication

##### **12.Build Your First Predictive Model**

* [Introduction and Overview](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* [Quiz: Introduction and Overview](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* [Preparing the Dataset](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* [Quiz: Preparing the dataset](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* [Build a Benchmark Model: Regression](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* Quiz: Build a Benchmark Model - Regression
* Benchmark Model: Regression Implementation
* Quiz: Benchmark Model - Regression Implementation
* Build a Benchmark Model: Classification
* Quiz: Build a Benchmark Model - Classification
* Benchmark Model: Classification Implementation
* Quiz: Benchmark - Classification Implementation

##### **13.Evaluation Metrics**

* Introduction to Evaluation Metrics
* Quiz: Introduction to Evaluation Metrics
* Confusion Matrix
* Quiz: Confusion Matrix
* Accuracy
* Quiz: Accuracy
* Alternatives of Accuracy
* Quiz: Alternatives of Accuracy
* Precision and Recall
* Quiz: Precision and Recall
* Thresholding
* Quiz: Thresholding
* AUC-ROC
* Quiz: AUC-ROC
* Log loss
* Quiz: Log loss
* Evaluation Metrics for Regression
* Quiz: Evaluation Metrics for Regression
* R2 and Adjusted R2
* Quiz: R2 and Adjusted R2

##### **14.Build Your First ML Model: k-NN**

* [Introduction to k-Nearest Neighbours](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* [Quiz: Introduction to k-Nearest Neighbours](https://courses.analyticsvidhya.com/enroll/453326?et=free_trial)
* Building a kNN model
* Quiz: Building a kNN model
* Determining right value of k
* Quiz: Determining right value of k
* How to calculate the distance
* Quiz: How to calculate the distance
* Issue with distance based algorithms
* Quiz: Issue with distance based algorithms
* Introduction to sklearn
* Dealing with Missing Values and Strings
* Implementing k-Nearest Neighbours algorithm
* Quiz: Implementing k-Nearest Neighbours algorithm

##### **15.Selecting the Right Model**

* Introduction to Overfitting and Underfitting Models
* Quiz: Introduction to Overfitting and Underfitting Models
* Visualizing overfitting and underfitting using knn
* Quiz: Visualizing overfitting and underfitting using knn
* Selecting the Right Model
* What is Validation?
* Quiz: What is Validation
* Understanding Hold-Out Validation
* Quiz: Understanding Hold-Out Validation
* Implementing Hold-Out Validation
* Quiz: Implementing Hold-Out Validation
* Understanding k-fold Cross Validation
* Quiz: Understanding k-fold Cross Validation
* Implementing k-fold Cross Validation
* Quiz: Implementing k-fold Cross Validation
* Bias Variance Tradeoff
* Quiz: Bias Variance Tradeoff

##### **16.Linear Models**

* Introduction to Linear Models
* Understanding Cost function
* Quiz: Understanding Cost function
* Understanding Gradient descent (Intuition)
* Maths behind gradient descent
* Convexity of cost function
* Quiz: Gradient Descent
* Assumptions of Linear Regression
* Implementing Linear Regression
* Generalized Linear Models
* Quiz: Generalized Linear Models
* Introduction to Logistic Regression
* Odds Ratio
* Implementing Logistic Regression
* Quiz: Logistic Regression
* Multiclass using Logistic Regression
* Quiz: Multi-Class Logistic Regression
* Challenges with Linear Regression
* Introduction to Regularisation
* Quiz: Introduction to Regularization
* Implementing Regularisation
* Coefficient estimate for ridge and lasso (Optional)

##### **17.Project: Customer Churn Prediction**

* Predicting whether a customer will churn or not
* Assignment: NYC taxi trip duration prediction

##### **18.Dimensionality Reduction (Part I)**

* Introduction to Dimensionality Reduction
* Quiz: Introduction to Dimensionality Reduction
* Common Dimensionality Reduction Techniques
* Quiz: Common Dimensionality Reduction Techniques
* Missing Value Ratio
* Missing Value Ratio Implementation
* Quiz: Missing Value Ratio
* Low Variance Filter
* Low Variance Filter Implementation
* Quiz: Low Variance Filter
* High Correlation Filter
* High Correlation Filter Implementation
* Quiz: High Correlation Filter
* Backward Feature Elimination
* Backward Feature Elimination Implementation
* Quiz: Backward Feature Elimination
* Forward Feature Selection
* Forward Feature Selection Implementation
* Quiz: Forward Feature Selection

##### **19.Decision Tree**

* Introduction to Decision Trees
* Quiz: Introduction to Decision Trees
* Purity in Decision Trees
* Quiz: Purity in Decision Trees
* Terminologies Related to Decision Trees
* Quiz: Terminologies Related to Decision Trees
* How to Select the Best Split Point in Decision Trees
* Quiz: How to Select the Best Split Point in Decision Trees
* Chi-Square
* Quiz: Chi-Square
* Information Gain
* Quiz: Information Gain
* Reduction in Variance
* Quiz: Reduction in Variance
* Optimizing Performance of Decision Trees
* Quiz: Optimizing Performance of Decision Trees
* Decision Tree Implementation

##### **20.Feature Engineering**

* Introduction to Feature Engineering
* Exercise on Feature Engineering
* Overview of the module
* Feature Transformation
* Quiz: Feature Transformation
* Feature Scaling
* Quiz: Feature Scaling
* Feature Encoding
* Quiz: Feature Encoding
* Combining Sparse classes
* Quiz: Combining Sparse classes
* Feature Generation: Binning
* Feature Interaction
* Quiz: Feature Interaction
* Generating Features: Missing Values
* Frequency Encoding
* Quiz: Frequency Encoding
* Feature Engineering: Date Time Features
* Implementing DateTime Features
* Quiz: Implementing DateTime Features
* Automated Feature Engineering : Feature Tools
* Implementing Feature tools

##### **21.Share your Learnings**

* Write for Analytics Vidhya's Medium Publication

##### **22.Project: NYC Taxi Trip Duration prediction**

* Exploring the NYC dataset
* Predicting the NYC taxi trip duration
* Predicting the NYC taxi trip duration

##### **23.Working with Text Data**

* Introduction to Text Feature Engineering
* Quiz: Introduction to Text Feature Engineering
* Create Basic Text Features
* Quiz: Create Basic Text Features
* Extract Information using Regular Expressions
* Quiz: Extract Information using Regular Expressions
* Learn to use Regular Expressions in Python
* Quiz: Learn to use Regular Expressions in Python
* Text Cleaning
* Quiz: Text Cleaning
* Create Linguistic Features
* Quiz: Create Linguistic Features
* Bag-of-Words
* Quiz: Bag-of-Words
* Text Pre-processing
* Quiz: Text Pre-processing
* TF-IDF Features
* Quiz: TF-IDF Features
* Word Embeddings
* Create word2vec Features
* Quiz: Word Embeddings

##### **24.Naïve Bayes**

* Introduction to Naive Bayes
* Quiz: Introduction to Naive Bayes
* Conditional Probability and Bayes Theorem
* Working of Naive Bayes
* Quiz: Conditional Probability and Naive Bayes
* Math Behind Naive Bayes
* Types of Naive Bayes
* Implementing Naive Bayes
* Quiz: Types of Naive Bayes

##### **25.Multiclass and Multilabel**

* Understanding how to solve Multiclass and Multilabel Classification Problem
* Quiz: Multiclass and Multilabel
* Evaluation Metrics: Multi Class Classification
* Quiz: Evaluation Metrics for Multi Class Classification

##### **26.Project: Web Page Classification**

* Understanding the Problem Statement
* Understanding the Data
* Building a Web Page Classifier

##### **27.Basics of Ensemble Techniques**

* Introduction to Ensemble
* Quiz: Introduction to Ensemble
* Basic Ensemble Techniques
* Quiz: Basic Ensemble Techniques
* Implementing Basic Ensemble Techniques
* Why Ensemble Models Work Well?

##### **28.Advance Ensemble Techniques**

* Introduction to Stacking
* Implementing Stacking
* Variants of Stacking
* Implementing Variants of Stacking
* Quiz: Variants of Stacking
* Introduction to Blending
* Implementation: Blending
* Quiz: Introduction to Blending
* Bootstrap Sampling
* Quiz: Bootstrap Sampling
* Introduction to Random Forest
* Quiz: Introduction to Random Forest
* Hyper-parameters of Random Forest
* Quiz: Hyper-parameters of Random Forest
* Implementing Random Forest
* Introduction to boosting
* Quiz: Introduction to Boosting
* Gradient Boosting Algorithm (GBM)
* Quiz: Gradient Boosting Algorithm
* Math Behind GBM
* Implementing GBM
* Extreme Gradient Boosting (XGBM)
* Implementing XGBM
* Quiz: Extreme Gradient Boosting
* Adaptive Boosting
* Implementing Adaptive Boosting
* Quiz: Adaptive Boosting

##### **29.Project: Ensemble Model on NYC Taxi Trip Duration Prediction**

* Predicting the NYC Taxi Trip Duration
* Prediction the NYC Taxi Trip Duration: Dataset

##### **30.Share your Learnings**

* Write for Analytics Vidhya's Medium Publication

##### **31.Hyperparameter Tuning**

* Introduction to Hyperparameter Tuning
* Different Hyperparameter Tuning methods
* Quiz: Hyperparameter Tuning
* Implementing different Hyperparameter Tuning methods

##### **32.Support Vector Machine**

* Understanding SVM Algorithm
* Quiz: Support Vector Machine
* SVM Kernel Tricks
* Kernels and Hyperparameters in SVM
* Implementing Support Vector Machine
* Quiz: Kernel Tricks

##### **33.Working with Image Data**

* Introduction to Images
* Understanding the Image data
* Quiz: Understanding the Image Data
* Understanding transformations on Images
* Understanding Edge Features
* Histogram of Oriented Features (HOG)
* Quiz: Image Features

##### **34.Project: Malaria Detection using Blood Cell Images**

* Understanding the Problem Statement
* Detecting Malaria using Blood Cell Images
* Dataset: Malaria Detection using Blood Cell Images

##### **35.Advance Dimensionality Reduction**

* Introduction to Principal Component Analysis
* Steps to perform Principal Component Analysis
* Quiz: Principal Component Analysis
* Computation of the Covariance Matrix
* Finding the Eigenvectors and Eigenvalues
* Understanding the MNIST dataset
* Implementing Principal Component Analysis
* Quiz: Steps to perform PCA
* Introduction to Factor Analysis
* Steps to perform Factor Analysis
* Quiz: Factor Analysis
* Implementing Factor Analysis
* Quiz: Implementing Factor Analysis

##### **36.Unsupervised Machine Learning Methods**

* Introduction to Clustering
* Quiz: Introduction to Clustering
* Applications of Clustering
* Evaluation Metrics for Clustering
* Quiz: Evaluation Metrics for Clustering
* Understanding K-Means
* K-Means from Scratch Implementation
* Quiz: Understanding K-Means
* Challenges with K-Means
* How to Choose Right k-Value
* K-Means Implementation
* Quiz: K-Means Implementation
* Hierarchical Clustering
* Implementation Hierarchical Clustering
* Quiz: Hierarchical Clustering
* How to Define Similarity between Clusters

##### **37.Working with Large Datasets: Dask**

* Introduction to Dask
* Understanding Dask Array and Dataframes
* Implementing Dask Array and Dataframes
* Machine Learning using Dask
* Implementing Linear Regression model using Dask

##### **38.Automated Machine Learning**

* Introduction to Automated Machine Learning
* Introduction to MLBox
* Implementing MLBox

##### **39.Introduction to Neural Network**

* Understanding the Problem
* Introduction to Neural Network
* Quiz: Introduction to Neural Network
* Understanding Forward Propagation
* Quiz: Forward Propogation
* Math Behind Forward Propagation
* Quiz: Math Behind Forward Propagation
* Error and Reason for Error
* Quiz: Error and Reason for Error
* Gradient Descent Intuition
* Understanding Maths Behind Gradient Descent
* Quiz: Gradient Descent
* Optimizers
* Quiz: Optimizer
* Back Propagation
* Quiz: Back Propagation
* Why Numpy?
* Quiz: Why Numpy?
* Understanding the Steps in Numpy
* Quiz: Understanding the Steps in Numpy
* Defining Parameters in Numpy
* Quiz: Defining Parameters in Numpy
* Implementing Forward Propagation
* Quiz: Implementing Forward Propagation
* Implementing Backward Propagation
* Quiz: Implementing Backward Propagation
* Notebook: Neural network from scratch
* Why Keras?
* Quiz: Why Keras?
* Neural Network in Keras
* Quiz: Neural Network in Keras
* Dataset: Emergency vs Non-Emergency Classification dataset
* How to handle image data?
* Quiz: How to handle Image data
* Exploring the Emergency Classification Dataset
* Quiz: Exploring the Emergency Classification Dataset
* Loading and Pre-Processing Dataset
* Quiz: Loading and Pre-Processing Dataset
* Solving the Challenge
* Quiz: Solving the challenge
* Hyperparameter Tuning
* Quiz: Hyperparameter Tuning
* Notebook: Simple Neural Network using keras
* Installation Steps

##### **40.Model Deployment**

* Introduction
* Applications of real time machine learning systems
* Offline Batch Processing vs Real Time Systems
* How to make real time systems
* Fundamentals of Memory and Storage
* Client Server Architecture
* Exposing our API to the world
* Assessing the scale of the problem
* Requirements and Implementation Strategy of our Article Recommender System
* Dataset: Data and Code for Recommendation System
* Simple Text Matching System
* Text Similarity Between Two Articles
* Creating Similarity Model
* Creating APIs for our application
* Performance Analysis of APIs
* Introduction to Git and Collaboration
* Resource: Model Deployment
* Module Test: Model Deployment

##### **41.Interpretability of Machine Learning Models**

* Introduction to Machine Learning Interpretability
* Framework and Interpretable Models
* Model Agnostic Methods for Interpretability
* Implementing Interpretable Model
* Implementing Global Surrogate and LIME

### Machine Learning Project 1

#### NYC Taxi Trip Duration Prediction

Uber, Lyft, Ola and many more online ride hailing services are trying hard to use their extensive data to create data products such as pricing engines, driver allotment etc. To improve the efficiency of taxi dispatching systems for such services, it is important to be able to predict how long a driver will have his taxi occupied or in other words the trip duration. This project will cover techniques to extract important features and accurately predict trip duration for taxi trips in New York using data from TLC commission New York.



### Machine Learning Project 2

#### Customer Churn Prediction

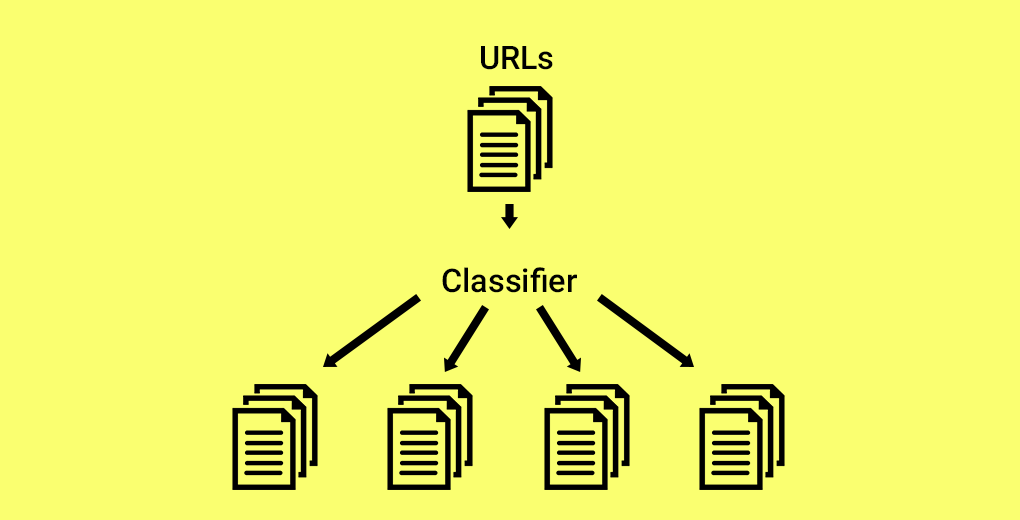
A Bank wants to take care of customer retention for their product; savings accounts. The bank wants you to identify customers likely to churn balances below the minimum balance in next quarter. You have the customers information such as age, gender, demographics along with their transactions with the bank. Your task as a data scientist would be to predict the propensity to churn for each customer.



### Machine Learning Project 3

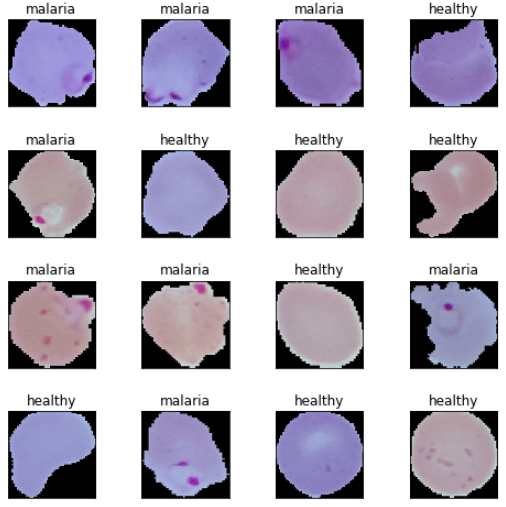
#### Web Page Classification

Classification of Web page content is vital to many tasks in Web information retrieval such as maintaining Web directories and focused crawling which is used to selectively seek out web pages that are relevant to a pre-defined set of topics. In this project, you will learn to build a web page classifier that can classify the web pages into their respective classes.



### Machine Learning Project 4

Malaria diagnosis involves close examination of the blood smear at 100x magnification. This is followed by a manual counting process wherein experts count the number of Red blood cells impacted by parasites. Automatic detection of Malaria from blood smear image is a scalable solution and can save a lot of hours for healthcare industry going a long way in our battle against this deadly disease. In this project, we try to identify from blood smears using deep learning to predict whether the sample is taken from an infected person.



FAQ

* 1. Who Should take the Applied Machine Learning course?

This course is meant for people looking to learn Machine Learning. We will start out to understand the pre-requisites, the underlying intuition behind several machine learning models and then go on to solve case studies using Machine learning concepts.

* 1. When will the classes be held in this course?

This is a self paced course, which you can take any time at your convenience over the 6 months after your purchase.

* 1. How many hours per week should I complete the course?

If you can put between 8 to 10 hours a week, you should be able to finish the course in 6 to 8 weeks.

* 1. Do I need to install any software before starting the course?

You will get information about all installation as part of the course.

* 1. What is the re-fund policy?

The for this course is not refundable.

* 1. Do I need to take the modules in a specific order?

We would highly recommend taking the course in the order in which it has been designed to gain the maximum knowledge from it.

* 1. Do I get a machine learning certificate upon completion of the course?

Yes, you will be given a certificate upon satisfactory completion of the Applied Machine learning course.

* 1. Which Machine Learning tools are we using in this course?
  2. How long I can access the course?

# The Ultimate Guide to 12 Dimensionality Reduction Techniques (with Python codes)

## Introduction

Have you ever worked on a dataset with more than a thousand features? How about over 50,000 features? I have, and let me tell you it’s a very challenging task, especially if you don’t know where to start! Having a high number of variables is both a boon and a curse. It’s great that we have loads of data for analysis, but it is challenging due to size.

It’s not feasible to analyze each and every variable at a microscopic level. It might take us days or months to perform any meaningful analysis and we’ll lose a ton of time and money for our business! Not to mention the amount of computational power this will take. We need a better way to deal with high dimensional data so that we can quickly extract patterns and insights from it. So how do we approach such a dataset?

Using dimensionality reduction techniques, of course. You can use this concept to reduce the number of features in your dataset without having to lose much information and keep (or improve) the model’s performance. It’s a really powerful way to deal with huge datasets, as you’ll see in this article.

This is a comprehensive guide to various dimensionality reduction techniques that can be used in practical scenarios. We will first understand what this concept is and why we should use it, before diving into the 12 different techniques I have covered. Each technique has it’s own implementation in Python to get you well acquainted with it.

## Table of Contents

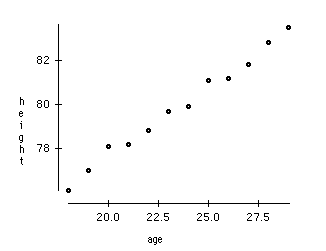
1. What is Dimensionality Reduction?
2. Why is Dimensionality Reduction required?
3. Common Dimensionality Reduction Techniques  
   3.1 Missing Value Ratio  
   3.2 Low Variance Filter  
   3.3 High Correlation Filter  
   3.4 Random Forest  
   3.5 Backward Feature Elimination  
   3.6 Forward Feature Selection  
   3.7 Factor Analysis  
   3.8 Principal Component Analysis  
   3.9 Independent Component Analysis  
   3.10 Methods Based on Projections  
   3.11 t-Distributed Stochastic Neighbor Embedding (t-SNE)  
   3.12 UMAP
4. Applications of Various Dimensionality Reduction Techniques

## 1. What is Dimensionality Reduction?

We are generating a tremendous amount of data daily. In fact, 90% of the data in the world has been generated in the last 3-4 years! The numbers are truly mind boggling. Below are just some of the examples of the kind of data being collected:

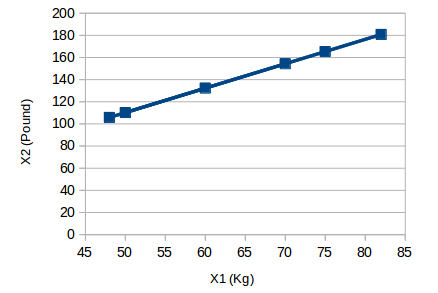
* Facebook collects data of what you like, share, post, places you visit, restaurants you like, etc.
* Your smartphone apps collect a lot of personal information about you
* Amazon collects data of what you buy, view, click, etc. on their site
* Casinos keep a track of every move each customer makes

As data generation and collection keeps increasing, visualizing it and drawing inferences becomes more and more challenging. One of the most common ways of doing visualization is through charts. Suppose we have 2 variables, Age and Height. We can use a scatter or line plot between Age and Height and visualize their relationship easily:



Now consider a case in which we have, say 100 variables (p=100). In this case, we can have 100(100-1)/2 = 5000 different plots. It does not make much sense to visualize each of them separately, right? In such cases where we have a large number of variables, it is better to select a subset of these variables (p<<100) which captures as much information as the original set of variables.

Let us understand this with a simple example. Consider the below image:



Here we have weights of similar objects in Kg (X1) and Pound (X2). If we use both of these variables, they will convey similar information. So, it would make sense to use only one variable. We can convert the data from 2D (X1 and X2) to 1D (Y1) as shown below:

https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2018/08/Screenshot-from-2018-07-26-13-51-52.png

Similarly, we can reduce p dimensions of the data into a subset of k dimensions (k<<n). This is called dimensionality reduction.

## 2. Why is Dimensionality Reduction required?

Here are some of the benefits of applying dimensionality reduction to a dataset:

* Space required to store the data is reduced as the number of dimensions comes down
* Less dimensions lead to less computation/training time
* Some algorithms do not perform well when we have a large dimensions. So reducing these dimensions needs to happen for the algorithm to be useful
* It takes care of multicollinearity by removing redundant features. For example, you have two variables – ‘time spent on treadmill in minutes’ and ‘calories burnt’. These variables are highly correlated as the more time you spend running on a treadmill, the more calories you will burn. Hence, there is no point in storing both as just one of them does what you require
* It helps in visualizing data. As discussed earlier, it is very difficult to visualize data in higher dimensions so reducing our space to 2D or 3D may allow us to plot and observe patterns more clearly

## 3. Common Dimensionality Reduction Techniques

Dimensionality reduction can be done in two different ways:

* By only keeping the most relevant variables from the original dataset (this technique is called feature selection)
* By finding a smaller set of new variables, each being a combination of the input variables, containing basically the same information as the input variables (this technique is called dimensionality reduction)

We will now look at various dimensionality reduction techniques and how to implement each of them in Python.

### 3.1 Missing Value Ratio

Suppose you’re given a dataset. What would be your first step? You would naturally want to explore the data first before building model. While exploring the data, you find that your dataset has some missing values. Now what? You will try to find out the reason for these missing values and then impute them or drop the variables entirely which have missing values (using appropriate methods).

What if we have too many missing values (say more than 50%)? Should we impute the missing values or drop the variable? I would prefer to drop the variable since it will not have much information. However, this isn’t set in stone. We can set a threshold value and if the percentage of missing values in any variable is more than that threshold, we will drop the variable.

Let’s implement this approach in Python.

# import required libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

First, let’s load the data:

# read the data

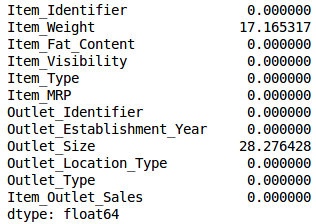
train=pd.read\_csv("Train\_UWu5bXk.csv")

Note: The path of the file should be added while reading the data.

Now, we will check the percentage of missing values in each variable. We can use .isnull().sum() to calculate this.

# checking the percentage of missing values in each variable

train.isnull().sum()/len(train)\*100



As you can see in the above table, there aren’t too many missing values (just 2 variables have them actually). We can impute the values using appropriate methods, or we can set a threshold of, say 20%, and remove the variable having more than 20% missing values. Let’s look at how this can be done in Python:

# saving missing values in a variable

a = train.isnull().sum()/len(train)\*100

# saving column names in a variable

variables = train.columns

variable = [ ]

for i in range(0,12):

    if a[i]<=20:   #setting the threshold as 20%

    variable.append(variables[i])

So the variables to be used are stored in “variable”, which contains only those features where the missing values are less than 20%.

### 3.2 Low Variance Filter

Consider a variable in our dataset where all the observations have the same value, say 1. If we use this variable, do you think it can improve the model we will build? The answer is no, because this variable will have zero variance.

So, we need to calculate the variance of each variable we are given. Then drop the variables having low variance as compared to other variables in our dataset. The reason for doing this, as I mentioned above, is that variables with a low variance will not affect the target variable.

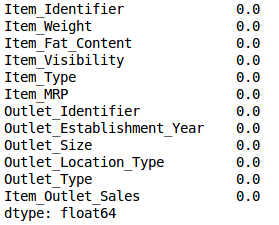
Let’s first impute the missing values in the Item\_Weight column using the median value of the known Item\_Weight observations. For the Outlet\_Size column, we will use the mode of the known Outlet\_Size values to impute the missing values:

train['Item\_Weight'].fillna(train['Item\_Weight'].median(), inplace=True)

train['Outlet\_Size'].fillna(train['Outlet\_Size'].mode()[0], inplace=True)

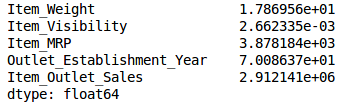
Let’s check whether all the missing values have been filled:

train.isnull().sum()/len(train)\*100



Voila! We are all set. Now let’s calculate the variance of all the numerical variables.

train.var()



As the above output shows, the variance of Item\_Visibility is very less as compared to the other variables. We can safely drop this column. This is how we apply low variance filter. Let’s implement this in Python:

numeric = train[['Item\_Weight', 'Item\_Visibility', 'Item\_MRP', 'Outlet\_Establishment\_Year']]

var = numeric.var()

numeric = numeric.columns

variable = [ ]

for i in range(0,len(var)):

    if var[i]>=10:   #setting the threshold as 10%

       variable.append(numeric[i+1])

The above code gives us the list of variables that have a variance greater than 10.

### 3.3 High Correlation filter

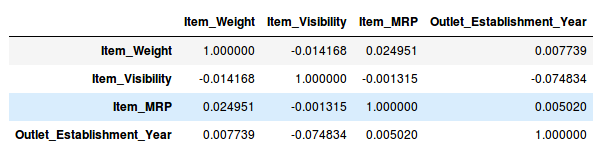
High correlation between two variables means they have similar trends and are likely to carry similar information. This can bring down the performance of some models drastically (linear and logistic regression models, for instance). We can calculate the correlation between independent numerical variables that are numerical in nature. If the correlation coefficient crosses a certain threshold value, we can drop one of the variables (dropping a variable is highly subjective and should always be done keeping the domain in mind).

**As a general guideline, we should keep those variables which show a decent or high correlation with the target variable.**

Let’s perform the correlation calculation in Python. We will drop the dependent variable (Item\_Outlet\_Sales) first and save the remaining variables in a new dataframe (df).

df=train.drop('Item\_Outlet\_Sales', 1)

df.corr()



Wonderful, we don’t have any variables with a high correlation in our dataset. Generally, if the correlation between a pair of variables is greater than 0.5-0.6, we should seriously consider dropping one of those variables.

### 3.4 Random Forest

Random Forest is one of the most widely used algorithms for feature selection. It comes packaged with in-built feature importance so you don’t need to program that separately. This helps us select a smaller subset of features.

We need to convert the data into numeric form by applying one hot encoding, as Random Forest (Scikit-Learn Implementation) takes only numeric inputs. Let’s also drop the ID variables (Item\_Identifier and Outlet\_Identifier) as these are just unique numbers and hold no significant importance for us currently.

from sklearn.ensemble import RandomForestRegressor

df=df.drop(['Item\_Identifier', 'Outlet\_Identifier'], axis=1)

model = RandomForestRegressor(random\_state=1, max\_depth=10)

df=pd.get\_dummies(df)

model.fit(df,train.Item\_Outlet\_Sales)

After fitting the model, plot the feature importance graph:

features = df.columns

importances = model.feature\_importances\_

indices = np.argsort(importances)[-9:]  # top 10 features

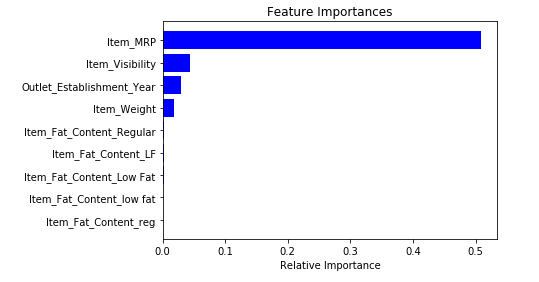
plt.title('Feature Importances')

plt.barh(range(len(indices)), importances[indices], color='b', align='center')

plt.yticks(range(len(indices)), [features[i] for i in indices])

plt.xlabel('Relative Importance')

plt.show()



Based on the above graph, we can hand pick the top-most features to reduce the dimensionality in our dataset. Alernatively, **we can use the SelectFromModel of sklearn to do so**. It selects the features based on the importance of their weights.

from sklearn.feature\_selection import SelectFromModel

feature = SelectFromModel(model)

Fit = feature.fit\_transform(df, train.Item\_Outlet\_Sales)

### 3.5 Backward Feature Elimination

Follow the below steps to understand and use the ‘Backward Feature Elimination’ technique:

* We first take all the n variables present in our dataset and train the model using them
* We then calculate the performance of the model
* Now, we compute the performance of the model after eliminating each variable (n times), i.e., we drop one variable every time and train the model on the remaining n-1 variables
* We identify the variable whose removal has produced the smallest (or no) change in the performance of the model, and then drop that variable
* Repeat this process until no variable can be dropped

**This method can be used when building Linear Regression or Logistic Regression models**. Let’s look at it’s Python implementation:

from sklearn.linear\_model import LinearRegression

from sklearn.feature\_selection import RFE

from sklearn import datasets

lreg = LinearRegression()

rfe = RFE(lreg, 10)

rfe = rfe.fit\_transform(df, train.Item\_Outlet\_Sales)

We need to specify the algorithm and number of features to select, and we get back the list of variables obtained from backward feature elimination. We can also check the ranking of the variables using the “rfe.ranking\_” command.

### 3.6 Forward Feature Selection

This is the opposite process of the Backward Feature Elimination we saw above. Instead of eliminating features, we try to find the best features which improve the performance of the model. This technique works as follows:

* We start with a single feature. Essentially, we train the model n number of times using each feature separately
* The variable giving the best performance is selected as the starting variable
* Then we repeat this process and add one variable at a time. The variable that produces the highest increase in performance is retained
* We repeat this process until no significant improvement is seen in the model’s performance

Let’s implement it in Python:

from sklearn.feature\_selection import f\_regression

ffs = f\_regression(df,train.Item\_Outlet\_Sales )

This returns an array containing the F-values of the variables and the p-values corresponding to each F value. Refer to [this link](http://www.statisticshowto.com/probability-and-statistics/f-statistic-value-test/)

<https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/f-statistic-value-test/>

to learn more about F-values. For our purpose, we will select the variables having F-value greater than 10:

variable = [ ]

for i in range(0,len(df.columns)-1):

    if ffs[0][i] >=10:

       variable.append(df.columns[i])

This gives us the top most variables based on the forward feature selection algorithm.

**NOTE** : **Both Backward Feature Elimination and Forward Feature Selection are time consuming and computationally expensive**.They are practically only used on datasets that have a small number of input variables.

The techniques we have seen so far are generally used when we do not have a very large number of variables in our dataset. These are more or less feature selection techniques. In the upcoming sections, we will be working with the Fashion MNIST dataset, which consists of images belonging to different types of apparel, e.g. T-shirt, trousers, bag, etc. **The dataset can be downloaded from the “**[**IDENTIFY THE APPAREL**](https://datahack.analyticsvidhya.com/contest/practice-problem-identify-the-apparels/)**” practice problem.**

The dataset has a total of 70,000 images, out of which 60,000 are in the training set and the remaining 10,000 are test images. For the scope of this article, we will be working only on the training images. The train file is in a zip format. Once you extract the zip file, you will get a .csv file and a train folder which includes these 60,000 images. The corresponding label of each image can be found in the ‘train.csv’ file.

### 3.7 Factor Analysis

Suppose we have two variables: Income and Education. These variables will potentially have a high correlation as people with a higher education level tend to have significantly higher income, and vice versa.

In the Factor Analysis technique, variables are grouped by their correlations, i.e., all variables in a particular group will have a high correlation among themselves, but a low correlation with variables of other group(s). Here, each group is known as a factor. These factors are small in number as compared to the original dimensions of the data. However, these factors are difficult to observe.

Let’s first read in all the images contained in the train folder:

import pandas as pd

import numpy as np

from glob import glob

import cv2

images = [cv2.imread(file) for file in glob('train/\*.png')]

NOTE: You must replace the path inside the glob function with the path of your train folder.

Now we will convert these images into a numpy array format so that we can perform mathematical operations and also plot the images.

images = np.array(images)

images.shape

(60000, 28, 28, 3)

As you can see above, it’s a 3-dimensional array. We must convert it to 1-dimension as all the upcoming techniques only take 1-dimensional input. To do this, we need to flatten the images:

image = []

for i in range(0,60000):

img = images[i].flatten()

image.append(img)

image = np.array(image)

Let us now create a dataframe containing the pixel values of every individual pixel present in each image, and also their corresponding labels (for labels, we will make use of the train.csv file).

train = pd.read\_csv("train.csv") # Give the complete path of your train.csv file

feat\_cols = [ 'pixel'+str(i) for i in range(image.shape[1]) ]

df = pd.DataFrame(image,columns=feat\_cols)

df['label'] = train['label']

Now we will decompose the dataset using Factor Analysis:

from sklearn.decomposition import FactorAnalysis

FA = FactorAnalysis(n\_components = 3).fit\_transform(df[feat\_cols].values)

Here, n\_components will decide the number of factors in the transformed data. After transforming the data, it’s time to visualize the results:

%matplotlib inline

import matplotlib.pyplot as plt

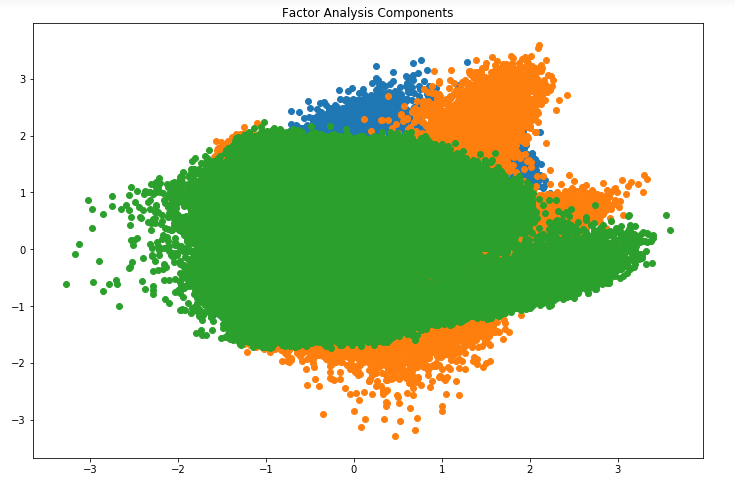
plt.figure(figsize=(12,8))

plt.title('Factor Analysis Components')

plt.scatter(FA[:,0], FA[:,1])

plt.scatter(FA[:,1], FA[:,2])

plt.scatter(FA[:,2],FA[:,0])



Looks amazing, doesn’t it? We can see all the different factors in the above graph. Here, the x-axis and y-axis represent the values of decomposed factors. As I mentioned earlier, it is hard to observe these factors individually but we have been able to reduce the dimensions of our data successfully.

### 3.8 Principal Component Analysis (PCA)

PCA is a technique which helps us in extracting a new set of variables from an existing large set of variables. These newly extracted variables are called Principal Components. You can refer to learn more about PCA. For your quick reference, below are some of the key points you should know about PCA before proceeding further:

* A principal component is a linear combination of the original variables
* Principal components are extracted in such a way that the first principal component explains maximum variance in the dataset
* Second principal component tries to explain the remaining variance in the dataset and is uncorrelated to the first principal component
* Third principal component tries to explain the variance which is not explained by the first two principal components and so on

Before moving further, we’ll randomly plot some of the images from our dataset:

rndperm = np.random.permutation(df.shape[0])

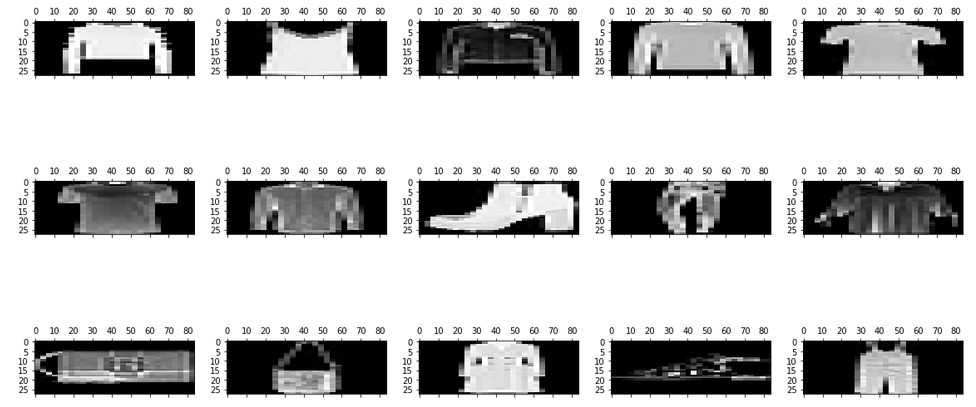
plt.gray()

fig = plt.figure(figsize=(20,10))

for i in range(0,15):

ax = fig.add\_subplot(3,5,i+1)

ax.matshow(df.loc[rndperm[i],feat\_cols].values.reshape((28,28\*3)).astype(float))



Let’s implement PCA using Python and transform the dataset:

from sklearn.decomposition import PCA

pca = PCA(n\_components=4)

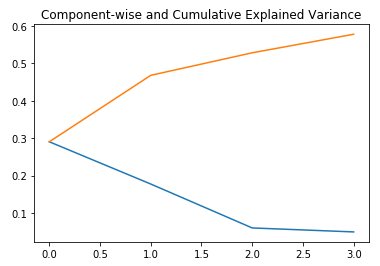
pca\_result = pca.fit\_transform(df[feat\_cols].values)

In this case, n\_components will decide the number of principal components in the transformed data. Let’s visualize how much variance has been explained using these 4 components. We will use explained\_variance\_ratio\_ to calculate the same.

plt.plot(range(4), pca.explained\_variance\_ratio\_)

plt.plot(range(4), np.cumsum(pca.explained\_variance\_ratio\_))

plt.title("Component-wise and Cumulative Explained Variance")



In the above graph, the blue line represents component-wise explained variance while the orange line represents the cumulative explained variance. **We are able to explain around 60% variance in the dataset using just four components.** Let us now try to visualize each of these decomposed components:

import seaborn as sns

plt.style.use('fivethirtyeight')

fig, axarr = plt.subplots(2, 2, figsize=(12, 8))

sns.heatmap(pca.components\_[0, :].reshape(28, 84), ax=axarr[0][0], cmap='gray\_r')

sns.heatmap(pca.components\_[1, :].reshape(28, 84), ax=axarr[0][1], cmap='gray\_r')

sns.heatmap(pca.components\_[2, :].reshape(28, 84), ax=axarr[1][0], cmap='gray\_r')

sns.heatmap(pca.components\_[3, :].reshape(28, 84), ax=axarr[1][1], cmap='gray\_r')

axarr[0][0].set\_title(

"{0:.2f}% Explained Variance".format(pca.explained\_variance\_ratio\_[0]\*100),

fontsize=12

)

axarr[0][1].set\_title(

"{0:.2f}% Explained Variance".format(pca.explained\_variance\_ratio\_[1]\*100),

fontsize=12

)

axarr[1][0].set\_title(

"{0:.2f}% Explained Variance".format(pca.explained\_variance\_ratio\_[2]\*100),

fontsize=12

)

axarr[1][1].set\_title(

"{0:.2f}% Explained Variance".format(pca.explained\_variance\_ratio\_[3]\*100),

fontsize=12

)

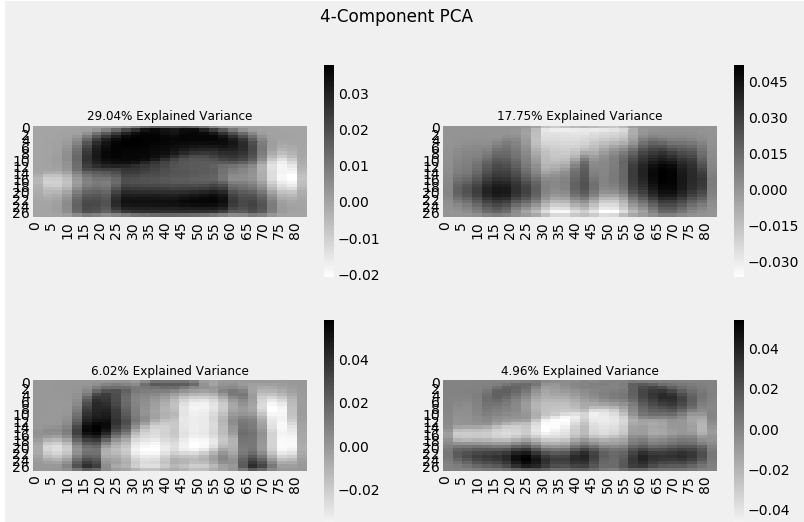
axarr[0][0].set\_aspect('equal')

axarr[0][1].set\_aspect('equal')

axarr[1][0].set\_aspect('equal')

axarr[1][1].set\_aspect('equal')

plt.suptitle('4-Component PCA')



Each additional dimension we add to the PCA technique captures less and less of the variance in the model. The first component is the most important one, followed by the second, then the third, and so on.

We can also use **Singular Value Decomposition** (SVD) to decompose our original dataset into its constituents, resulting in dimensionality reduction. To learn the mathematics behind SVD,.

VD decomposes the original variables into three constituent matrices. It is essentially used to remove redundant features from the dataset. It uses the concept of Eigenvalues and Eigenvectors to determine those three matrices. We will not go into the mathematics of it due to the scope of this article, but let’s stick to our plan, i.e. reducing the dimensions in our dataset.

Let’s implement SVD and decompose our original variables:

from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n\_components=3, random\_state=42).fit\_transform(df[feat\_cols].values)

Let us visualize the transformed variables by plotting the first two principal components:

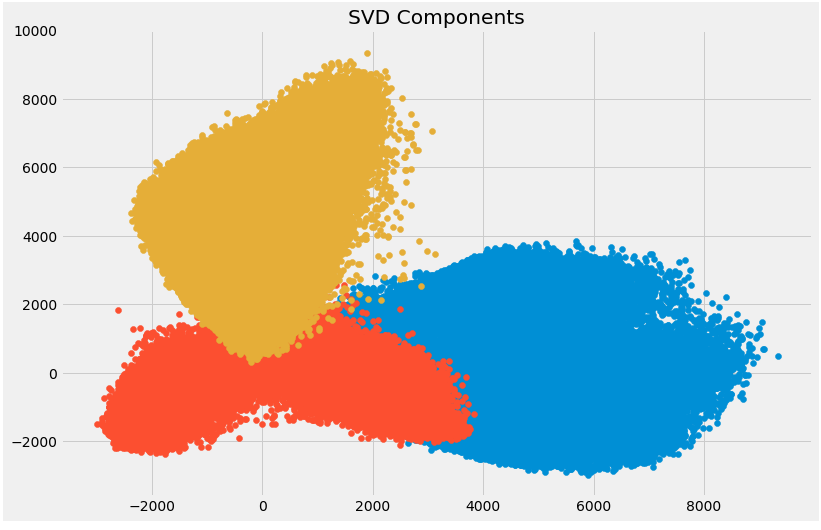
plt.figure(figsize=(12,8))

plt.title('SVD Components')

plt.scatter(svd[:,0], svd[:,1])

plt.scatter(svd[:,1], svd[:,2])

plt.scatter(svd[:,2],svd[:,0])



The above scatter plot shows us the decomposed components very neatly. As described earlier, there is not much correlation between these components.

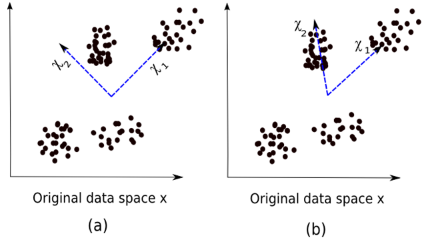
### 3.9 Independent Component Analysis

Independent Component Analysis (ICA) is based on information-theory and is also one of the most widely used dimensionality reduction techniques. The major difference between PCA and ICA is that PCA looks for uncorrelated factors while ICA looks for independent factors.

If two variables are uncorrelated, it means there is no linear relation between them. If they are independent, it means they are not dependent on other variables. For example, the age of a person is independent of what that person eats, or how much television he/she watches.

**This algorithm assumes that the given variables are linear mixtures of some unknown latent variables. It also assumes that these latent variables are mutually independent**, i.e., they are not dependent on other variables and hence they are called the independent components of the observed data.

Let’s compare PCA and ICA visually to get a better understanding of how they are different:



Here, image (a) represents the PCA results while image (b) represents the ICA results on the same dataset.

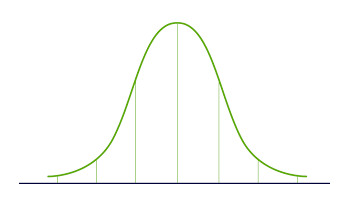
The equation of PCA is x = Wχ.

Here,

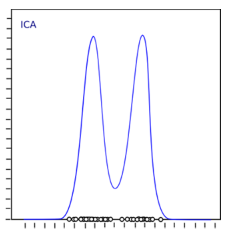
* x is the observations
* W is the mixing matrix
* χ is the source or the independent components

Now we have to find an un-mixing matrix such that the components become as independent as possible. Most common method to measure independence of components is Non-Gaussianity:

As per the central limit theorem, distribution of the sum of independent components tends to be normally distributed (Gaussian).



* So we can look for the transformations that maximize the kurtosis of each component of the independent components. Kurtosis is the third order moment of the distribution. To learn more about kurtosis, head over [here](https://brownmath.com/stat/shape.htm#Kurtosis).
* Maximizing the kurtosis will make the distribution non-gaussian and hence we will get independent components.



The above distribution is non-gaussian which in turn makes the components independent. Let’s try to implement ICA in Python:

from sklearn.decomposition import FastICA

ICA = FastICA(n\_components=3, random\_state=12)

X=ICA.fit\_transform(df[feat\_cols].values)

Here, n\_components will decide the number of components in the transformed data. We have transformed the data into 3 components using ICA. Let’s visualize how well it has transformed the data:

plt.figure(figsize=(12,8))

plt.title('ICA Components')

plt.scatter(X[:,0], X[:,1])

plt.scatter(X[:,1], X[:,2])

plt.scatter(X[:,2], X[:,0])

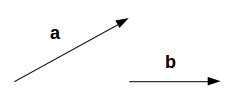


The data has been separated into different independent components which can be seen very clearly in the above image. X-axis and Y-axis represent the value of decomposed independent components.

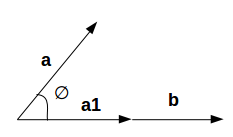
Now we shall look at some of the methods which reduce the dimensions of the data using projection techniques.

### 3.10 Methods Based on Projections

To start off, we need to understand what projection is. Suppose we have two vectors, vector **a** and vector **b**, as shown below:



We want to find the projection of **a** on **b**. Let the angle between a and b be ∅. The projection (**a1**) will look like:



**a1**is the vector parallel to **b.**So, we can get the projection of vector a on vector b using the below equation:

https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2018/08/Screenshot-from-2018-08-09-19-56-35.png

Here,

* a1 = projection of a onto b
* b̂ = unit vector in the direction of b

By projecting one vector onto the other, dimensionality can be reduced.

In projection techniques, multi-dimensional data is represented by projecting its points onto a lower-dimensional space. Now we will discuss different methods of projections:

* Projection onto interesting directions:
  + Interesting directions depend on specific problems but generally, directions in which the projected values are non-gaussian are considered to be interesting
  + Similar to ICA (Independent Component Analysis), projection looks for directions maximizing the kurtosis of the projected values as a measure of non-gaussianity

Projection onto Manifolds:

Once upon a time, it was assumed that the Earth was flat. No matter where you go on Earth, it keeps looking flat (let’s ignore the mountains for a while). But if you keep walking in one direction, you will end up where you started. That wouldn’t happen if the Earth was flat. The Earth only looks flat because we are minuscule as compared to the size of the Earth.

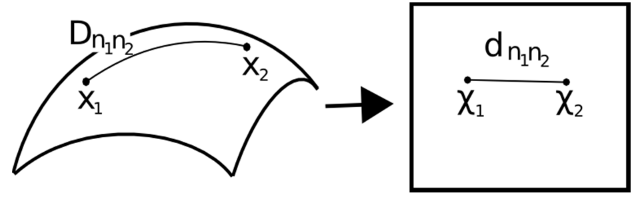
These small portions where the Earth looks flat are manifolds, and if we combine all these manifolds we get a large scale view of the Earth, i.e., original data. Similarly for an n-dimensional curve, small flat pieces are manifolds and a combination of these manifolds will give us the original n-dimensional curve. Let us look at the steps for projection onto manifolds:

* We first look for a manifold that is close to the data
* Then project the data onto that manifold
* Finally for representation, we unfold the manifold
* There are various techniques to get the manifold, and all of these techniques consist of a three-step approach:
  + Collecting information from each data point to construct a graph having data points as vertices
  + Transforming the above generated graph into suitable input for embedding steps
  + Computing an (nXn) eigen equation

Let us understand manifold projection technique with an example.

If a manifold is continuously differentiable to any order, it is known as smooth or differentiable manifold. ISOMAP is an algorithm which aims to recover full low-dimensional representation of a non-linear manifold. It assumes that the manifold is smooth.

It also assumes that for any pair of points on manifold, the geodesic distance (shortest distance between two points on a curved surface) between the two points is equal to the Euclidean distance (shortest distance between two points on a straight line). Let’s first visualize the geodesic and Euclidean distance between a pair of points:



Here,

* Dn1n2 = geodesic distance between X1 and X2
* dn1n2 = Euclidean distance between X1 and X2

ISOMAP assumes both of these distances to be equal. Let’s now look at a more detailed explanation of this technique. As mentioned earlier, all these techniques work on a three-step approach. We will look at each of these steps in detail:

Neighborhood Graph:

* First step is to calculate the distance between all pairs of data points:  
  dij = dχ(xi,xj) = || xi-xj || χ  
  Here,  
  dχ(xi,xj) = geodesic distance between xi and xj  
  || xi-xj || = Euclidean distance between xi and xj
* After calculating the distance, we determine which data points are neighbors of manifold
* Finally the neighborhood graph is generated: G=G(V,ℰ), where the set of vertices V = {x1, x2,…., xn} are input data points and set of edges ℰ = {eij} indicate neighborhood relationship between the points
* Compute Graph Distances:
  + Now we calculate the geodesic distance between pairs of points in manifold by graph distances
  + Graph distance is the shortest path distance between all pairs of points in graph G
* Embedding:
  + Once we have the distances, we form a symmetric (nXn) matrix of squared graph distance
  + Now we choose embedding vectors to minimize the difference between geodesic distance and graph distance
  + Finally, the graph G is embedded into Y by the (t Xn) matrix

Let’s implement it in Python and get a clearer picture of what I’m talking about. We will perform non-linear dimensionality reduction through Isometric Mapping. For visualization, we will only take a subset of our dataset as running it on the entire dataset will require a lot of time.

from sklearn import manifold

trans\_data = manifold.Isomap(n\_neighbors=5, n\_components=3, n\_jobs=-1).fit\_transform(df[feat\_cols][:6000].values)

Parameters used:

* *n\_neighbors* decides the number of neighbors for each point
* *n\_components* decides the number of coordinates for manifold
* *n\_jobs*= -1 will use all the CPU cores available

Visualizing the transformed data:

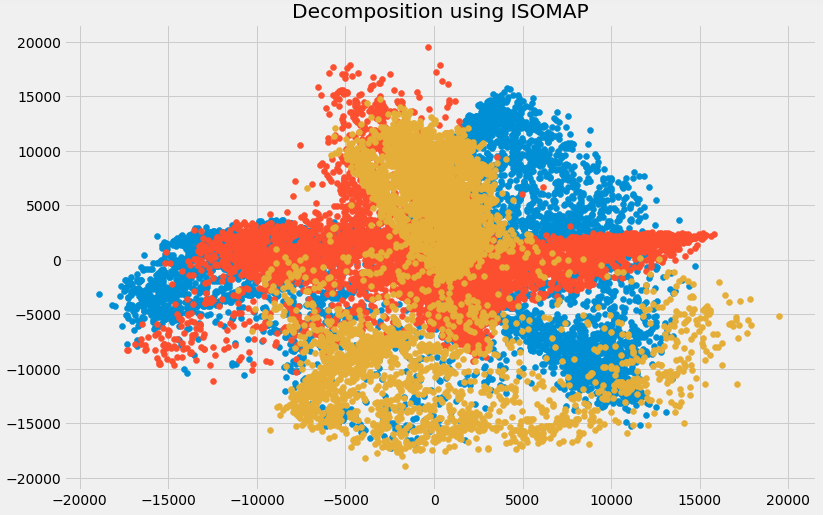
plt.figure(figsize=(12,8))

plt.title('Decomposition using ISOMAP')

plt.scatter(trans\_data[:,0], trans\_data[:,1])

plt.scatter(trans\_data[:,1], trans\_data[:,2])

plt.scatter(trans\_data[:,2], trans\_data[:,0])

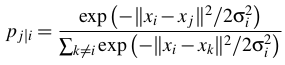


You can see above that the correlation between these components is very low. In fact, they are even less correlated as compared to the components we obtained using SVD earlier!

### 3.11 t- Distributed Stochastic Neighbor Embedding (t-SNE)

So far we have learned that PCA is a good choice for dimensionality reduction and visualization for datasets with a large number of variables. But what if we could use something more advanced? What if we can easily search for patterns in a non-linear way? t-SNE is one such technique. There are mainly two types of approaches we can use to map the data points:

* Local approaches :  They maps nearby points on the manifold to nearby points in the low dimensional representation.
* Global approaches : They attempt to preserve geometry at all scales, i.e. mapping nearby points on manifold to nearby points in low dimensional representation as well as far away points to far away points.
* t-SNE is one of the few algorithms which is capable of retaining both local and global structure of the data at the same time
* It calculates the probability similarity of points in high dimensional space as well as in low dimensional space
* High-dimensional Euclidean distances between data points are converted into conditional probabilities that represent similarities:



* xi and xj are data points, ||xi-xj|| represents the Euclidean distance between these data points, and 𝛔i is the variance of data points in high dimensional space
* For the low-dimensional data points yi and yj corresponding to the high-dimensional data points xi and xj, it is possible to compute a similar conditional probability using:

https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2018/08/Screenshot-from-2018-08-09-20-13-01.png

where ||yi-yj|| represents the Euclidean distance between yi and yj

* After calculating both the probabilities, it minimizes the difference between both the probabilities

You can learn about t-SNE in more detail.

We will now implement it in Python and visualize the outcomes:

from sklearn.manifold import TSNE

tsne = TSNE(n\_components=3, n\_iter=300).fit\_transform(df[feat\_cols][:6000].values)

n\_components will decide the number of components in the transformed data. Time to visualize the transformed data:

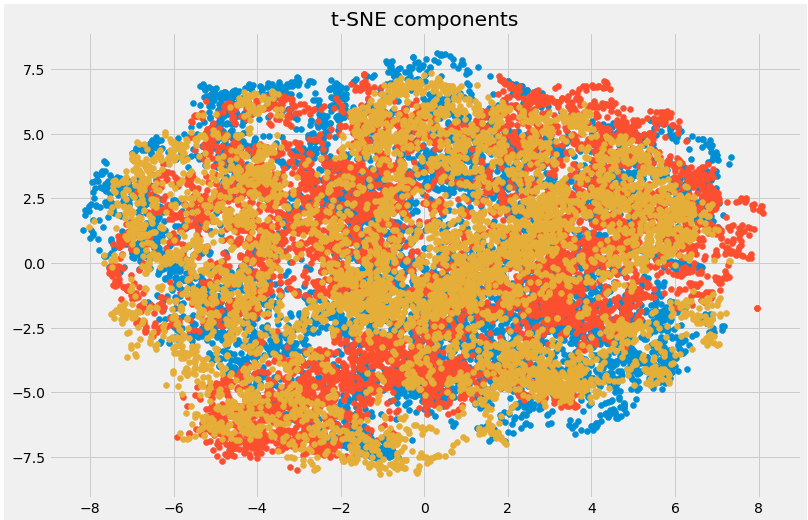
plt.figure(figsize=(12,8))

plt.title('t-SNE components')

plt.scatter(tsne[:,0], tsne[:,1])

plt.scatter(tsne[:,1], tsne[:,2])

plt.scatter(tsne[:,2], tsne[:,0])



Here you can clearly see the different components that have been transformed using the powerful t-SNE technique.

### 3.12 UMAP

t-SNE works very well on large datasets but it also has it’s limitations, such as loss of large-scale information, slow computation time, and inability to meaningfully represent very large datasets. Uniform Manifold Approximation and Projection (UMAP) is a dimension reduction technique that can preserve as much of the local, and more of the global data structure as compared to t-SNE, with a shorter runtime. Sounds intriguing, right?

Some of the key advantages of UMAP are:

* It can handle large datasets and high dimensional data without too much difficulty
* It combines the power of visualization with the ability to reduce the dimensions of the data
* Along with preserving the local structure, it also preserves the global structure of the data. UMAP maps nearby points on the manifold to nearby points in the low dimensional representation, and does the same for far away points

This method uses the concept of k-nearest neighbor and optimizes the results using stochastic gradient descent. It first calculates the distance between the points in high dimensional space, projects them onto the low dimensional space, and calculates the distance between points in this low dimensional space. It then uses Stochastic Gradient Descent to minimize the difference between these distances. To get a more in-depth understanding of how UMAP works, check out [this paper](https://arxiv.org/pdf/1802.03426.pdf).

Refer [here](http://umap-learn.readthedocs.io/en/latest/) to see the documentation and installation guide of UMAP. We will now implement it in Python:

import umap

umap\_data = umap.UMAP(n\_neighbors=5, min\_dist=0.3, n\_components=3).fit\_transform(df[feat\_cols][:6000].values)

Here,

* *n\_neighbors* determines the number of neighboring points used
* *min\_dist* controls how tightly embedding is allowed. Larger values ensure embedded points are more evenly distributed

Let us visualize the transformation:

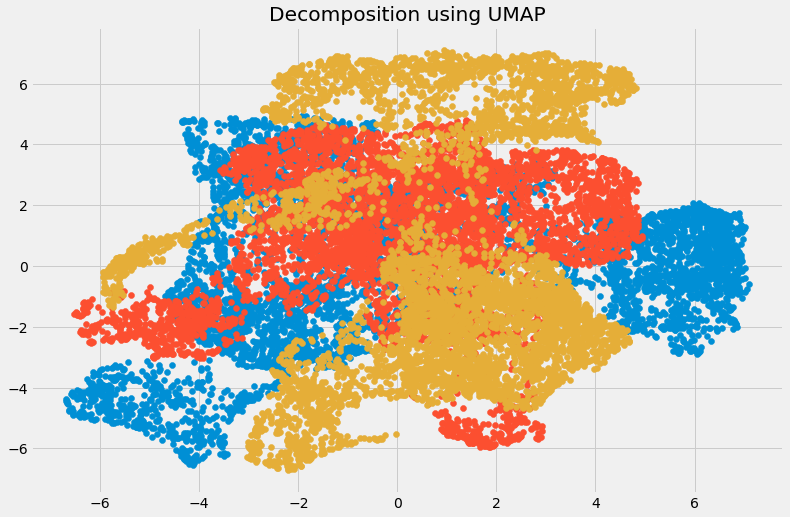
plt.figure(figsize=(12,8))

plt.title('Decomposition using UMAP')

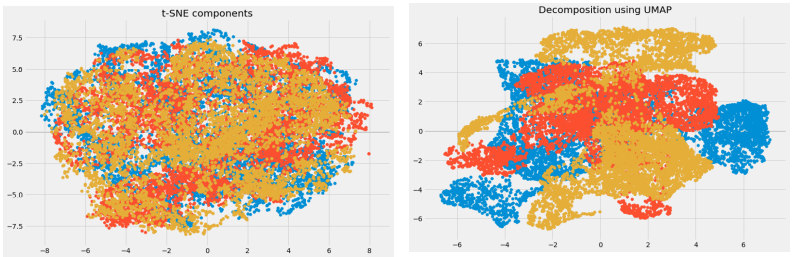
plt.scatter(umap\_data[:,0], umap\_data[:,1])

plt.scatter(umap\_data[:,1], umap\_data[:,2])

plt.scatter(umap\_data[:,2], umap\_data[:,0])



The dimensions have been reduced and we can visualize the different transformed components. There is very less correlation between the transformed variables. Let us compare the results from UMAP and t-SNE:



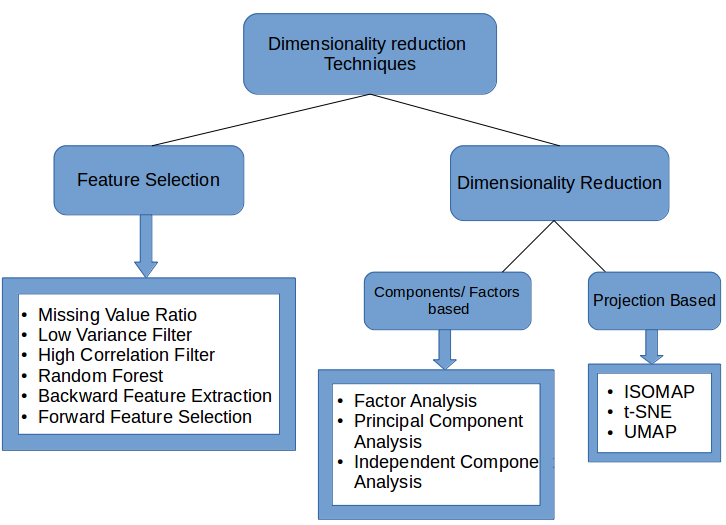
We can see that the correlation between the components obtained from UMAP is quite less as compared to the correlation between the components obtained from t-SNE. Hence, UMAP tends to give better results.

**As mentioned in UMAP’s GitHub repository, it often performs better at preserving aspects of the global structure of the data than t-SNE. This means that it can often provide a better “big picture” view of the data as well as preserving local neighbor relations.**

Take a deep breath. We have covered quite a lot of the dimensionality reduction techniques out there. Let’s briefly summarize where each of them can be used.

## 4. Brief Summary of when to use each Dimensionality Reduction Technique

In this section, we will briefly summarize the use cases of each dimensionality reduction technique that we covered. It’s important to understand where you can, and should, use a certain technique as it helps save time, effort and computational power.



* **Missing Value Ratio**: If the dataset has too many missing values, we use this approach to reduce the number of variables. We can drop the variables having a large number of missing values in them
* **Low Variance filter**: We apply this approach to identify and drop constant variables from the dataset. The target variable is not unduly affected by variables with low variance, and hence these variables can be safely dropped
* **High Correlation filter**: A pair of variables having high correlation increases multicollinearity in the dataset. So, we can use this technique to find highly correlated features and drop them accordingly
* **Random Forest**: This is one of the most commonly used techniques which tells us the importance of each feature present in the dataset. We can find the importance of each feature and keep the top most features, resulting in dimensionality reduction
* Both **Backward Feature Elimination** and **Forward Feature Selection** techniques take a lot of computational time and are thus generally used on smaller datasets
* **Factor Analysis**: This technique is best suited for situations where we have highly correlated set of variables. It divides the variables based on their correlation into different groups, and represents each group with a factor
* **Principal Component Analysis**: This is one of the most widely used techniques for dealing with linear data. It divides the data into a set of components which try to explain as much variance as possible
* **Independent Component Analysis**: We can use ICA to transform the data into independent components which describe the data using less number of components
* **ISOMAP**: We use this technique when the data is strongly non-linear
* **t-SNE**: This technique also works well when the data is strongly non-linear. It works extremely well for visualizations as well
* **UMAP**: This technique works well for high dimensional data. Its run-time is shorter as compared to t-SNE

## End Notes

This is as comprehensive an article on dimensionality reduction as you’ll find anywhere! I had a lot of fun writing it and found a few new ways of dealing with high number of variables I hadn’t used before (like UMAP).

Dealing with thousands and millions of features is a must-have skill for any data scientist. The amount of data we are generating each day is unprecedented and we need to find different ways to figure out how to use it. Dimensionality reduction is a very useful way to do this and has worked wonders for me, both in a professional setting as well as in machine learning hackathons.

I’m looking forward to hearing your feedback and ideas in the comments section below.

You can also read this article on Analytics Vidhya's Android APP

### NLP Assessment Test

NLP assessment test is specially designed to help you choose the right path in your journey of becoming a data scientist. Check if you are the right fit for the course.

Natural Language Processing(NLP) is the art of extracting information from unstructured text. This course teaches you basics of Python, Regular Expression, Topic Modeling, various techniques life TF-IDF, NLP using Neural Networks and Deep Learning.

### About Natural Language Processing (NLP) using Python Course

**Why pursue Natural Language Processing (NLP)?**

* More than 80% of the data in this world is unstructured in nature, which includes text. You need text mining and Natural Language processing  (NLP) to make sense out of this data.
* Natural Language Processing (NLP) helps you extract insights from emails of customers, their tweets, text messages.
* Natural Language Processing (NLP) can power many applications, such as language translation, question answering systems, chatbots and document summarizers.

**Key Takeways from the Natural Language Processing using Python course:**

* Understand the nature of text data and how to work with it.
* Learn about different text pre-processing techniques.
* Learn how to perform Parts-of-Speech Tagging and Named Entity Recognition
* Learn about the important techniques for feature extraction from text.
* Understand how deep learning can be used to solve complex tasks in NLP.
* Implement awesome NLP projects using Deep Learning.

#### Why Natural Language Processing?

### Highlights of Natural Language Processing (NLP) using Python

### Natural Language Processing (NLP) Using Python Course Curriculum

* 1. Course Handouts

##### Module 1 : Introduction to Natural Language Processing

* Getting Started
* Knowing each other
* Welcome to the Course
* [About the Course](https://courses.analyticsvidhya.com/enroll/426391?et=free_trial)
* Introduction to Natural Language Processing
* Exercise : Introduction to Natural Language Processing
* Podcast with NLP Researcher Sebastian Ruder

##### **Module 2 : A Refresher to Python**

* Installation steps for Linux
* Installation steps for Mac
* Installation steps for Windows
* Packages Installation
* Introduction to Python
* Variables and Operators
* Exercise : Variables and Operators
* Python Lists
* Exercise : Python Lists
* Dictionaries
* Exercise : Dictionaries
* Conditional Statements
* Exercise : Conditional Statements
* Loops
* Exercise : Loops
* Functions
* Python Functions Practice
* Exercise : Functions
* Packages
* Exercise : Packages
* Files
* Exercise : Files

4

##### **Module 3 : Learn to use Regular Expressions**

* Welcome to Module
* [Understanding Regular Expression](https://courses.analyticsvidhya.com/enroll/426391?et=free_trial)
* Implementing Regular Expression in Python
* Exercise : Implementing Regular Expression in Python
* Regular Expressions in Action

##### **Module 4 : First Step of NLP - Text Processing**

* Welcome to Module
* Tokenization and Text Normalization
* Exercise : Tokenization and Text Normalization
* Exploring Text Data
* Part of Speech Tagging and Grammar Parsing
* Exercise : Part of Speech Tagging and Grammar Parsing
* Implementing Text Pre-processing Using NLTK
* Exercise : Implementing Text Pre-processing Using NLTK
* Natural Language Processing Techniques using spaCy

##### **Module 5 : Extracting Named Entities from Text**

* Welcome to Module
* [Understanding Named Entity Recognition](https://courses.analyticsvidhya.com/enroll/426391?et=free_trial)
* Exercise : Understanding Named Entity Recognition
* Implementing Named Entity Recognition
* Exercise : Implementing Named Entity Recognition
* Named Entity Recognition and POS tagging using spaCy
* POS and NER in Action : Text Data Augmentation
* Assignment: Share your learning and build your profile

##### **Module 6 : Feature Engineering for Text**

* Introduction to Text Feature Engineering
* Count Vector, TFIDF Representations of Text
* Exercise : Introduction to Text Feature Engineering
* Understanding Vector Representation of Text
* Exercise : Understanding Vector Representation of Text
* Understanding Word Embeddings
* Word Embeddings in Action - Word2Vec
* Word Embeddings in Action - GloVe

##### **Module 7 : Mastering the Art of Text Cleaning**

* Introduction to Text Cleaning Techniques Part 1
* Exercise : Introduction to Text Cleaning Techniques Part 1
* Introduction to Text Cleaning Techniques Part 2
* Exercise : Introduction to Text Cleaning Techniques Part 2
* Text Cleaning Implementation
* Exercise : Text Cleaning Implementation
* NLP Techniques using spaCy

##### **Module 8 : Project I - Social Media Information Extraction**

* Project I - Social Media Information Extraction

##### **Module 9 : Interpreting Patterns from Text - Topic Modelling**

* Introduction to Topic Modelling
* Exercise : Introduction to Topic Modelling
* Understanding LDA
* Exercise : Understanding LDA
* Implementation of Topic Modelling
* Exercise : Implementation of Topic Modelling
* LSA for Topic Modelling

##### **Module 10: Project II - Categorization of Sports Articles**

* Understanding the Problem Statement
* Importing Dataset
* Text Cleaning and Pre-processing
* Categorizing Articles using Topic Modelling

##### **Module 11.1 : Machine Learning Algorithms**

* Note
* Types of Machine Learning Algorithms
* Logistic Regression
* Decision Tree
* Naive Bayes
* SVM (Support Vector Machine)
* Random Forest

##### **Module 11.2 : Understanding Text Classification**

* Overview of Text Classification
* Exercise : Overview of Text Classification
* Assignment: Share your learning and build your profile

##### **Module 12.1 : Introduction to Deep Learning (Optional)**

* Note
* Getting started with Neural Network
* Exercise : Getting started with Neural Network
* Understanding Forward Propogation
* Exercise : Forward Propogation
* Math Behind Forward Propagation
* Exercise : Math Behind Forward Propagation
* Error and Reason for Error
* Exercise : Error and Reason for Error
* Gradient Descent Intuition
* Understanding Math Behind Gradient Descent
* Exercise : Gradient Descent
* Optimizer
* Exercise : Optimizer
* Back Propagation
* Exercise : Back Propagation
* Why Keras?
* Exercise : Why Keras?
* Building a Neural Network for Text Classification
* Why CNN?
* Exercise : Why CNN?
* Understanding the working of CNN Filters
* Exercise : Understanding the working of CNN Filters
* Introduction to Padding
* Exercise : Introduction to Padding
* Padding Strategies
* Exercise : Padding Strategies
* Padding Strategies in Keras
* Exercise : Padding Strategies in Keras
* Introduction to Pooling
* Exercise : Introduction to Pooling
* CNN architecture and its working
* Exercise : CNN architecture and its working

##### **Module 12.2 : Deep Learning for NLP**

* Deep Learning for NLP Part 1
* Exercise : Deep Learning for NLP Part 1
* Deep Learning for NLP Part 2
* Exercise : Deep Learning for NLP Part 2
* Text Generation Using LSTM
* Exercise : Text Generation Using LSTM

##### **Module 13 : Project III – SMS Spam Classification**

* Dataset download
* Text Cleaning
* Feature Engineering
* Advanced Feature Engineering
* Combining Features
* ML Classifier
* Spam Classification using Deep Learning

##### **Module 14 : Project IV – Hate Speech Classification**

* Project III

##### **Module 15 : Project V – Building Auto-Tagging System**

* Overview of Auto-Tagging System
* Introduction to Dataset and Performance Metrics
* Auto-Tagging Implementation Using Machine Learning Part-1
* Auto-Tagging Implementation Using Machine Learning Part-2
* Auto-Tagging Implementation Using Deep Learning

##### **Module 16 : Recurrent Neural Networks**

* Why RNN
* Introduction to RNN: Shortcomings of an MLP
* Introduction to RNN: RNN Architecture
* Training an RNN: Forward propagation
* Training an RNN: Backpropagation through time
* Need for LSTM/GRU
* Long Short Term Memory (LSTM)
* Gated Recurrent Unit (GRU)
* Project: Categorisation of websites using LSTM and GRU I
* Dataset and Notebook
* Project: Categorisation of websites using LSTM and GRU II

##### **Module 17 : Introduction to Language Modeling in NLP**

* Overview : Language Modeling
* What is a Language Model in NLP?
* N-gram Language Model
* Implementing an N-gram Language Model - I
* Implementing an N-gram Language Model - II
* Neural Language Model
* Implementing a Neural Language Model

##### **Module 18 : Sequence-to-Sequence Modeling**

* Intuition Behind Sequence-to-Sequence Modeling
* Need for Sequence-to-Sequence Modeling
* Understanding the Architecture of Sequence-to-Sequence
* Understanding Functioning of Encoder and Decoder
* Case Study: Building an Spanish to English Machine Translation Model
* Preprocessing of Text Data
* Converting Text to Integer Sequences
* Model Building and Inference

##### **Module 19 : Project VI - Summarization of Customer Reviews**

* Introduction
* Preprocessing and Feature Creation
* Model Building and Summary Generation

##### **Module 20 : Project VII - Build your first Chatbot**

* Introduction
* About this module
* Overview of Conversational Agents
* Project - Foodbot
* Overview of Rasa Framework
* System Setup
* Rasa NLU: Understanding user intent from a message
* Rasa NLU: Extracting intents from a user's message
* Rasa Core: Making your chatbot conversational
* Working with Zomato API
* Create a Workspace in Slack
* Deploying to Slack
* Assignment: Share your learning and build your profile

##### **Module 21 : Bonus Section (Advance NLP tools)**

* Getting started with Bonus Section
* Text Classification & Word Representations using FastText (An NLP library by Facebook)
* Introduction to Flair for NLP: A Simple yet Powerful State-of-the-Art NLP Library
* Introduction to StanfordNLP: An Incredible State-of-the-Art NLP Library for 53 Languages (with Python code)
* A Step-by-Step NLP Guide to Learn ELMo for Extracting Features from Text
* Tutorial on Text Classification (NLP) using ULMFiT and fastai Library in Python
* 8 Excellent Pretrained Models to get you Started with Natural Language Processing (NLP)
* Geo-coding using NLP by Shantanu Bhattacharyya and Farhat Habib
* Demystifying the What, Why and How of Chatbot by Sonny Laskar
* Sentiment Analysis using NLP and Deep Learning by Jeeban Swain
* Identifying Location using Clustering and Language Model - By Divya Choudhary
* Building Intelligent Chatbots from Scratch

##### **Module 22 : Where to go from here?**

* Where to go from here?

### Project-Social Media Information Extraction (In-class)

This project is designed to teach you how to extract relevant information such as entities, ngrams, keywords and sentiments from social media data using NLP techniques. The project highlights the importance of nlp techniques to extract business insights from the text data.



### Project-SMS Spam Classification (In-class)

This project is about the classification of SMS text messages as spam or nonspam. In this project, the students will learn to preprocess, feature engineering techniques, and text classification techniques using machine learning models and the CNN model.



### Project-Hate Speech Classification

Hate speech is an unfortunately common occurrence on the Internet. Often social media sites like Facebook and Twitter face the problem of identifying and censoring problematic posts while weighing the right to freedom of speech. The importance of detecting and moderating hate speech is evident from the strong connection between hate speech and actual hate crimes. Early identification of users promoting hate speech could enable outreach programs that attempt to prevent an escalation from speech to action. The objective of this task is to detect hate speech in tweets. For the sake of simplicity, we say a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets.



# achine Learning

### Project- Categorization of Sports Articles (In-class)

# Document categorization or segregation is an important NLP task which is used across a wide range of industries. In this project, we will learn to segregate sports articles using an unsupervised technique called Topic Modelling. We will categorize the articles based on the content of the articles, i.e., similar articles will be grouped together.



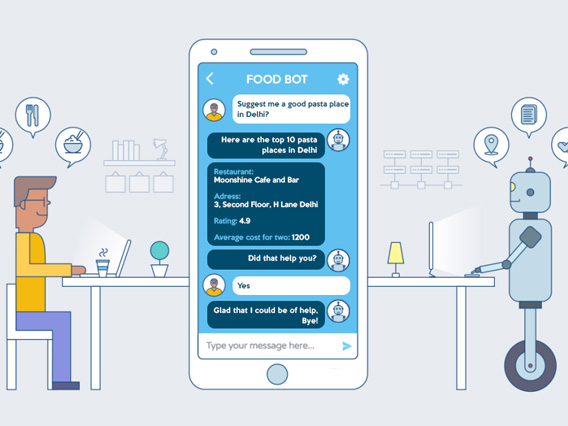
### Project - Building Auto Tagging System (In-Class)

Automatic tagging of questions on platforms like stackoverflow is quite vital to build a healthy user engagement at the platform. These tags help both, the users seeking solutions to their problems and the experts capable of solving those problems, find the relevant questions easily. In this project, we will build an automatic tagger for the Stack Overflow questions.



### Project- Build your first Chatbot (In-Class)

Chatbots are everywhere today, from booking your flight tickets to ordering food, chances are that you have already interacted with one. In this module, you will build your first chatbot to search for restaurants online and learn how to use it in a real-world application by deploying it on Slack.



### Project- Summarization of Customer Reviews (In-class)

Automatic Text Summarization is a process of generating a concise and meaningful summary of text from multiple text resources such as news articles, blog posts, research papers, customer reviews, emails, and tweets. In this project, we will create short summaries of customer reviews on the women's clothing dataset, using sequence-to-sequence modeling.



Computer Vision using Deep Learning 2.0

Accelerate your career with Analytics Vidhya’s computer vision course! Work on hands-on real world computer vision case studies, learn the fundamentals of deep learning and get familiar with tips and tricks to improve your models.

### About the course

There has been a tremendous boom in the applications of Computer Vision now a days.

The applications of Computer Vision range from understanding the environment in a Self - Driving Car to build Facial Recognition based Attention Systems for classrooms in Education Industry.

A question you might ask is: why would I even want to know about Computer Vision ? As a matter of fact, there is an undeniable demand for people who have knowledge in this domain, so that they can bring about disruptive solutions in any industry possible.

Computer Vision systems deal with high variety and volume of data, specifically images or videos.It is represented as bits and blobs which is hard to explain to a machine.As a result, these systems need intricate techniques to make sense of the data and then make data driven decisions.

This course is designed to give you a taste of how the underlying techniques work in current State - of -the - Art Computer Vision systems, and walks you through a few of the remarkable Computer Vision applications in a hands - on manner so that you can create such solutions on your own.

### Pre-requisites

This is a beginner friendly course, so it does not assume any familiarity with Computer Vision or Deep Learning algorithms. But, this course assumes that you are comfortable with Python programming.

##### **Course Handouts**

* Course Handouts

##### **Introduction to computer vision**

* Welcome to Computer Vision
* Knowing each other
* [Documentary on Computer Vision](https://courses.analyticsvidhya.com/enroll/447645?et=free_trial)
* Exercise-1
* Applications of Computer Vision
* Exercise-2
* Why Computer Vision is more in Demand?
* Exercise-3
* Understand your course content
* Exercise- 4

##### **Getting ready for the course**

* Getting ready for the course
* System Requirements
* Setting up the System on Cloud
* Setting up locally
* Accessing the course material
* Pre-reads for the next modules

##### **Introduction to Neural Network**

* Understanding the problem
* Getting started with Neural Network
* Exercise : Getting started with Neural Network
* Independent and dependent variables
* Understanding Forward Propogation
* Exercise : Forward Propogation
* Math Behind Forward Propagation
* Exercise : Math Behind Forward Propagation
* Error and Reason for Error
* Exercise : Error and Reason for Error
* Gradient Descent Intuition
* Understanding Math Behind Gradient Descent
* Exercise : Gradient Descent
* Optimizer
* Exercise : Optimizer
* Back Propagation
* Exercise : Back Propagation
* Why Numpy?
* Exercise : Why Numpy?
* [Understanding the Steps in Numpy](https://courses.analyticsvidhya.com/enroll/447645?et=free_trial)
* Exercise : Understanding the Steps in Numpy
* Defining Parameters in Numpy
* Exercise : Defining Parameters in Numpy
* Implementing Forward Propagation
* Exercise : Implementing Forward Propagation
* Implementing Backward Propagation
* Exercise : Implementing Backward Propagation
* Neural network from scratch (Notebook)
* Why Keras?
* Exercise : Why Keras?
* Neural Networks in Keras
* Exercise : Neural Network in Keras
* How to handle image data
* Exercise : How to handle Image data
* Exploring the Emergency Classification Dataset
* Exercise : Exploring the Emergency Classification Dataset
* Loading and Pre-Processing Dataset
* Exercise : Loading and Pre-Processing Dataset
* Solving the challenge
* Exercise : Solving the challenge
* Hyperparameter Tuning
* Exercise : Hyperparameter Tuning
* Simple Neural Network using keras (Notebook)
* Summary of the Module
* Assignment: Share your learning and build your profile

##### **Project I**

* Project I

##### **Introduction to Convolutional Neural Network**

* Why CNN?
* Exercise : Why CNN?
* Understanding the working of CNN Filters
* Exercise : Understanding the working of CNN Filters
* Introduction to Padding
* Exercise : Introduction to Padding
* Padding Strategies
* Exercise : Padding Strategies
* Padding Strategies in Keras
* Exercise : Padding Strategies in Keras
* Introduction to Pooling
* Exercise : Introduction to Pooling
* CNN architecture and its working
* Exercise : CNN architecture and its working
* Loading and Pre-Processing Dataset
* Solving the challenge
* Exercise : Solving the challenge
* Hyperparameter Tuning
* Exercise : Hyperparameter Tuning
* Convolutional Neural Network in keras (Notebook)
* Summary of the Module
* Exercise : Gender Classification using CNN
* Assignment: Share your learning and build your profile

##### **Introduction to Transfer Learning**

* [Introduction to Transfer Learning](https://courses.analyticsvidhya.com/enroll/447645?et=free_trial)
* How to select right pre-trained model?
* Exercise : How to select right pre-trained model?
* Steps to solve emergency vs non emergency vehicle classification challenge using transfer learning
* Exercise : Steps to build the model using transfer learning
* Solving the challenge using Transfer Learning (Part I)
* Exercise
* Solving the challenge using Transfer Learning (Part II)
* Exercise
* Solving the challenge using Transfer Learning (Part III)
* Exercise : Solving the challenge using transfer learning
* Solving the challenge using transfer learning (Notebook)
* Different fine tuning techniques
* Advantages of transfer learning
* Exercise : Gender Classification using transfer learning

##### **Tips and Tricks to Improve Model Performance**

* Understanding areas of improvement
* Exercise : Understanding areas of improvement
* Problem - Less Data to train the model
* Exercise : Less Data to train the model
* Solution - Less Data to train the model
* Exercise : Solution - Less Data to train the model
* Problem - High Variation in Data
* Exercise : High Variation in Data
* Solution - High Variation in Data
* Exercise : Solution - High Variation in Data
* Problem: Overfitting
* Exercise : Overfitting
* Solution - Overfitting
* Exercise : Solution - Overfitting
* Problem - Underfitting
* Exercise : Underfitting
* Problem - Too high training time
* Exercise : Too high training time
* Solution - Too high training time
* Problem - No Appropriate Architecture
* Exercise : No Appropriate Architecture
* Solution - No Appropriate Architecture
* Exercise : Solution - No Appropriate Architecture
* Problem - Wrong Evaluation Metric - Part I
* Exercise : Wrong Evaluation Metric - Part I
* Problem - Wrong Evaluation Metric - Part II
* Exercise : Wrong Evaluation Metric : Part II
* Combining Tips and Tricks
* Tips and tricks implementation (Notebook)
* Exercise : Combining the Trips and Tricks
* Exercise : Improving the model performance on Gender Classification
* Assignment: Share your learning and build your profile

##### **Horizon of Computer Vision and Case Studies**

* Understanding the Types of Computer Vision Problems
* Exercise : Understanding the type of problems
* Approach to solve different type of problems
* Exercise : Approach to solve different type of problems
* Understanding the Regression Problem
* Exploring Facial Keypoint Identification Problem
* Exercise : Exploring Facial Keypoint Identification Problem
* Loading and Pre-Processing Dataset
* Solving the Regression Challenge
* Exercise : Solving the Regression Challenge
* Implementation : Facial Keypoint Identification (Notebook)
* Project II
* Understanding Object Detection Problem
* Exercise : Understanding Object Detection Problem Statement
* Naïve Appoach to solve detection problem
* Exercise : Naïve Appoach to solve detection problem
* Exploring Blood Cell Detection Problem
* Loading and Pre-Processing Dataset
* Building the model
* Implementation : Blood cell detection - Part I (Notebook)
* Other Approaches to solve Detection Problem
* Exercise : Other Approaches to solve Detection Problem
* Implementing State-of-the-art Model on Blood Detection
* Understanding State-of-the-art Model
* Implementation : Blood cell detection - Part II (Notebook)
* Exercise : Understanding State-of-the-art Model
* Project III
* Assignment: Share your learning and build your profile

##### **Where to go from here?**

* What did we discuss?
* Where to go from here?

##### **Bonus Material**

* Visualization of Learning and Localization of Convolutional Neural Networks by Sunil Kumar Vuppala
* Generative Adversarial Networks by Keshav Dhandhania
* Attention-based Deep Learning Models to Extract Details from Images by Vijay Gabale
* Diagnosing your Model : Learning How To Debug Deep Learning Models Using Visualisation For Medical Images - By Rohit Ghosh
* Hack Session: Understanding the Building Blocks of Deep Learning using PyTorch - By Vishnu Subramanian
* Failing Fast with Deep Learning - By Jaidev Deshpande

### Project-1

#### Classify Emergency Vehicles from Non-Emergency Vehicles (In-class)

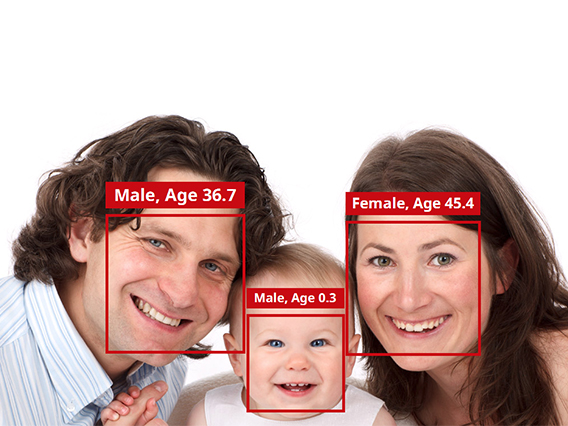
Fatalities due to traffic delays of emergency vehicles such as ambulance & fire brigade is a huge problem. In daily life, we often see that an emergency vehicles face difficulty in passing through traffic. So differentiating a vehicle into an emergency and non emergency category can be an important component in traffic monitoring as well as self drive car systems as reaching on time to their destination is critical for these services. In this project, you will get to design a computer vision system that can do just this.



### Project-2

#### Age Prediction of People from closeups of Facial Images

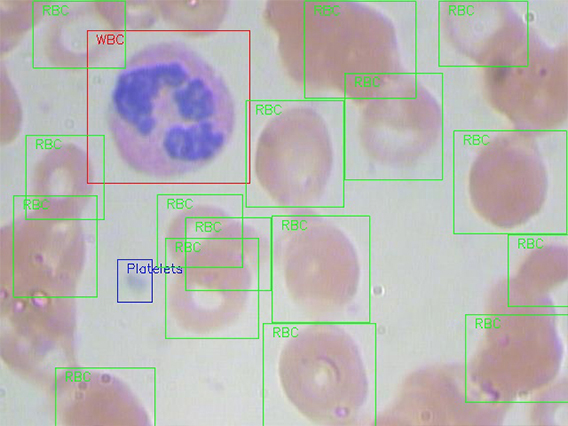
We now have systems that can correctly identify faces in the wild, but they fail to give us the the facial properties to build intelligent systems, like age of the person or their gender. This project will urge you to create algorithms that would power these intelligent systems, specifically by predicting the age of the person directly from an image clipping of his/her face.



### Project-3

#### Identify the Location of Red Blood Cells (In-class)

The analysis of blood cells allows the evaluation and diagnosis of a vast number of diseases. But this is generally done manually by skilled operators. In practice, we can automate a part of this process by identifying individual blood cell from a microscopic image. The task of this project will challenge you to find the locations of red blood cells through Deep Learning



I have also provided resources for each application so you can deep dive further into the one(s) which grabs your attention.

Note: Before you read on, I recommend going through this superb article. It’s not mandatory for understanding what we will cover here but it’s a valuable article for your budding skillset.

## Table of Contents

* Why Study Linear Algebra?
* Linear Algebra in Machine Learning
  + Loss functions
  + Regularization
  + Covariance Matrix
  + Support Vector Machine Classification
* Linear Algebra in Dimensionality Reduction
  + Principal Component Analysis (PCA)
  + Singular Value Decomposition (SVD)
* Linear Algebra in Natural Language Processing
  + Word Embeddings
  + Latent Semantic Analysis
* Linear Algebra in Computer Vision
  + Image Representation as Tensors
  + Convolution and Image Processing

## Why Study Linear Algebra?

I have come across this question way too many times. Why should you spend time learning Linear Algebra when you can simply import a package in Python and build your model? It’s a fair question. So, let me present my point of view regarding this.

I consider Linear Algebra as one of the foundational blocks of Data Science. You cannot build a skyscraper without a strong foundation, can you? Think of this scenario:

*You want to reduce the dimensions of your data using Principal Component Analysis (PCA). How would you decide how many Principal Components to preserve if you did not know how it would affect your data? Clearly, you need to know the mechanics of the algorithm to make this decision.*

With an understanding of Linear Algebra, you will be able to develop a better intuition for machine learning and deep learning algorithms and not treat them as black boxes. This would allow you to choose proper hyperparameters and develop a better model.

You would also be able to code algorithms from scratch and make your own variations to them as well. Isn’t this why we love data science in the first place? The ability to experiment and play around with our models? Consider linear algebra as the key to unlock a whole new world.

## Linear Algebra in Machine Learning

The big question – where does linear algebra fit in machine learning? Let’s look at four applications you will all be quite familiar with.

### 1. Loss Functions

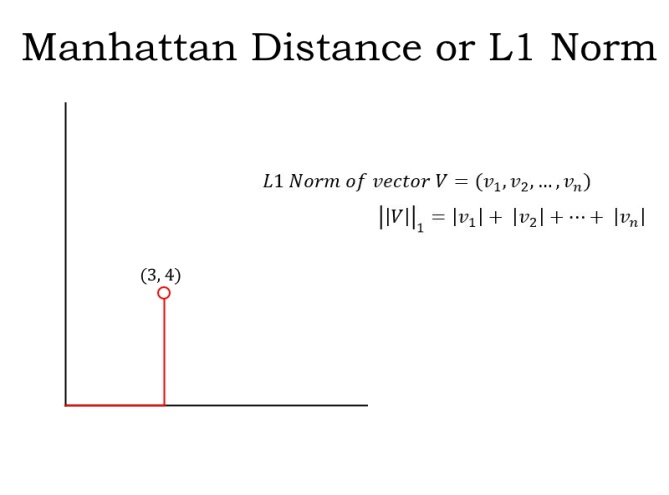
You must be quite familiar with how a model, say a Linear Regression model, fits a given data:

* You start with some arbitrary prediction function (a linear function for a Linear Regression Model)
* Use it on the independent features of the data to predict the output
* Calculate how far-off the predicted output is from the actual output
* Use these calculated values to optimize your prediction function using some strategy like Gradient Descent

**But wait – how can you calculate how different your prediction is from the expected output? Loss Functions, of course.**

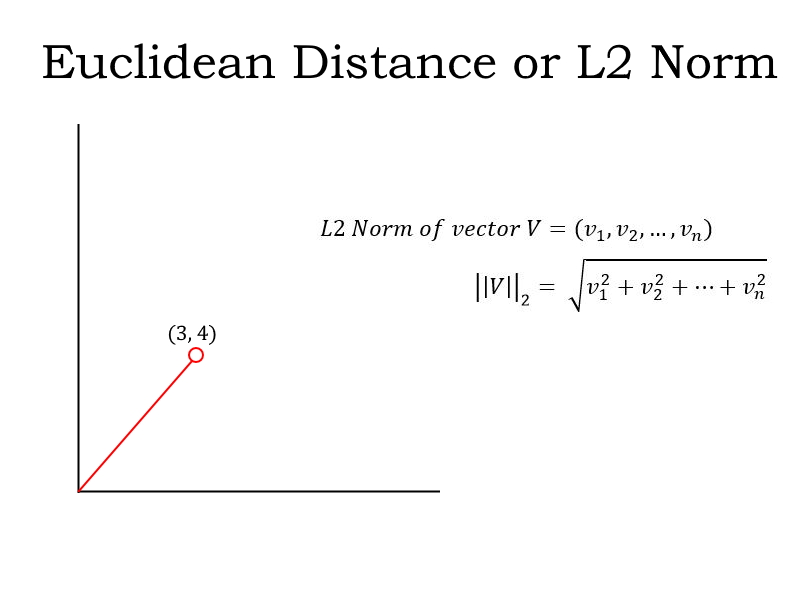
A loss function is an application of the **Vector Norm** in Linear Algebra. The norm of a vector can simply be its magnitude. There are many types of vector norms. I will quickly explain two of them:

* **L1 Norm**: Also known as the Manhattan Distance or Taxicab Norm. The L1 Norm is the distance you would travel if you went from the origin to the vector if the only permitted directions are parallel to the axes of the space.

**Coudsfesfldskf**

**Dfkldflkdsflkdsfkflkdsfpkdsfpigfporetw[ofw;lgrwaptjewfewfowefkdmf;lfwrlgfwrlglrsg’reg,r’g,rwg,r’;g**ginner to Pro In this 2D space, you could reach the vector (3, 4) by traveling 3 units along the x-axis and then 4 units parallel to the y-axis (as shown). Or you could travel 4 units along the y-axis first and then 3 units parallel to the x-axis. In either case, you will travel a total of 7 units.

* **L2 Norm**:  Also known as the Euclidean Distance. L2 Norm is the shortest distance of the vector from the origin as shown by the red path in the figure below:

fessional

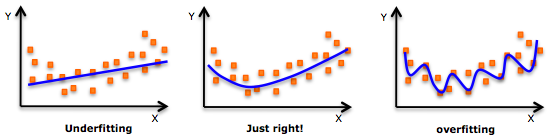
plied This distance is calculated using the Pythagoras Theorem (I can see the old math concepts flickering on in your mind!). It is the square root of (3^2 + 4^2), which is equal to 5.

But how is the norm used to find the difference between the predicted values and the expected values? Let’s say the predicted values are stored in a vector **P** and the expected values are stored in a vector ***E***. Then ***P-E*** is the difference vector. And the norm of ***P-E*** is the total loss for the prediction.

### 2. Regularization

Regularization is a very important concept in data science. It’s a technique we use to prevent models from overfitting. Regularization is actually another application of the Norm.

A model is said to overfit when it fits the training data too well. Such a model does not perform well with new data because it has learned even the noise in the training data. It will not be able to generalize on data that it has not seen before. The below illustration sums up this idea really well:

 Regularization **penalizes overly complex models by adding the norm of the weight vector to the cost function.** Since we want to minimize the cost function, we will need to minimize this norm. This causes unrequired components of the weight vector to reduce to zero and prevents the prediction function from being overly complex.

You can read the below article to learn about the complete mathematics behind regularization:

* How to avoid Over-Fitting using Regularization

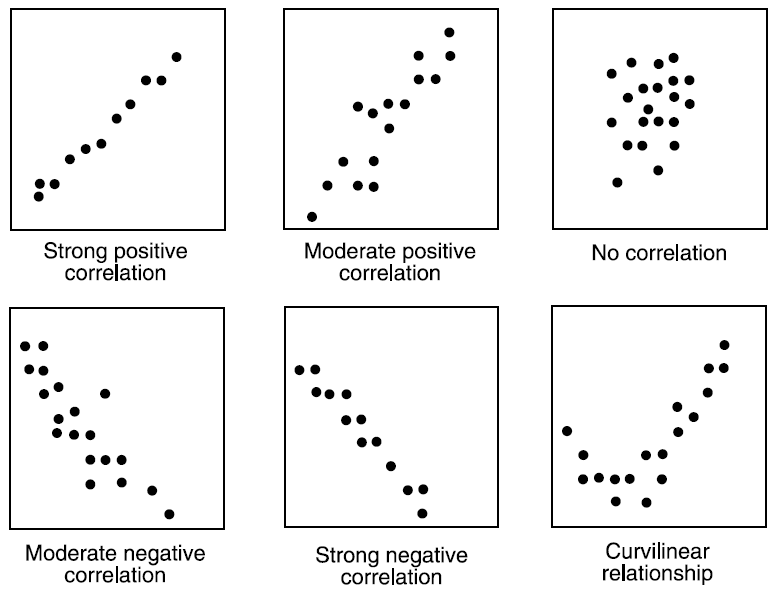
The L1 and L2 norms we discussed above are used in two types of regularization:

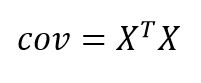
* L1 regularization used with **Lasso** **Regression**
* L2 regularization used with **Ridge Regression**

### 3. Covariance Matrix

Bivariate analysis is an important step in **data exploration**. We want to study the relationship between pairs of variables. Covariance or Correlation are measures used to study relationships between**two continuous variables**.

**Covariance indicates the direction of the linear relationship between the variables.** A positive covariance indicates that an increase or decrease in one variable is accompanied by the same in another. A negative covariance indicates that an increase or decrease in one is accompanied by the opposite in the other.

Mac**dtfesfewewrt**hine Learning - Beginner to On the other hand, **correlation is the standardized value of Covariance**. A correlation value tells us both the strength and direction of the linear relationship and has the range from -1 to 1.

Now, you might be thinking that this is a concept of Statistics and not Linear Algebra. Well, remember I told you Linear Algebra is all-pervasive? Using the concepts of **transpose and matrix multiplication** in Linear Algebra, we have a pretty neat expression for the covariance matrix:[](https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2019/07/covariance.jpg.jpg)

Here, X is the standardized data matrix containing all numerical features.

# I encourage you to read our Complete Tutorial on Data Exploration to know more about the Covariance Matrix, Bivariate Analysis and the other steps involved in Exploratory Data Analysis.

### 4. Support Vector Machine Classification

Ah yes, support vector machines. One of the most common classification algorithms that regularly produces impressive results. It is an application of the concept of **Vector Spaces** in Linear Algebra.

Support Vector Machine, or SVM, is a discriminative classifier that works by finding a decision surface. It is a supervised machine learning algorithm.

In this algorithm, we plot each data item as a point in an **n-dimensional space** (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the **hyperplane** that differentiates the two classes very well i.e. with the **maximum margin**, which is C is this case.

# linear_algebra_data_science

Pr A hyperplane is a **subspace** whose dimensions are one less than its corresponding vector space, so it would be a straight line for a 2D vector space, a 2D plane for a 3D vector space and so on. Again Vector Norm is used to calculate the margin.

But what if the data is not linearly separable like the case below?

# linear_algebra_data_scienceofessional

This Our intuition says that the decision surface has to be a circle or an ellipse, right? But how do you find it? Here, the concept of **Kernel Transformations** comes into play. The idea of transformation from one space to another is very common in Linear Algebra.

Let’s introduce a variable***z = x^2 + y^2***. This is how the data looks if we plot it along the z and x-axes:

## linear_algebra_data_sciencecourse provides you all the tools and techniques you need to apply machine learning to solve business problems. We will cover the basics of mach Now, this is clearly linearly separable by a line***z = a***, where a is some positive constant. On transforming back to the original space, we get x^2 + y^2 = a as the decision surface, which is a circle!

## linear_algebra_data_scienceine learning, how to build machine learning models, improve and deploy your machine learning models.

And the best part? We do not need to add additional features on our own. SVM has a technique called the **kernel trick**. Read this article on Support Vector Machines to learn about SVM, the kernel trick and how to implement it in Python.

## Dimensionality Reduction

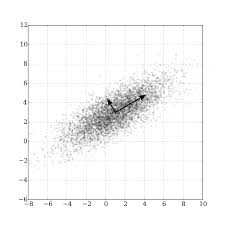
You will often work with datasets that have hundreds and even thousands of variables. That’s just how the industry functions. Is it practical to look at each variable and decide which one is more important?

That doesn’t really make sense. We need to bring down the number of variables to perform any sort of coherent analysis. This is what dimensionality reduction is. Now, let’s look at two commonly used dimensionality reduction methods here.

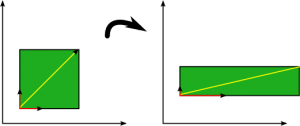
### 5. Principal Component Analysis (PCA)

Principal Component Analysis, or PCA, is an unsupervised dimensionality reduction technique. PCA finds the **directions of maximum variance** and projects the data along them to reduce the dimensions.

Without going into the math, these directions are the [**eigenvectors**](https://www.youtube.com/watch?v=PFDu9oVAE-g)**of the covariance matrix** of the data.



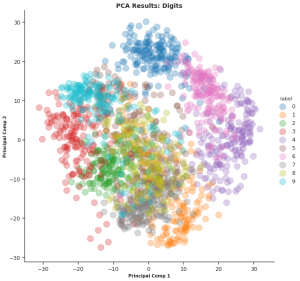
Eigenvectors for a square matrix are special non-zero vectors whose direction does not change even after applying linear transformation (which means multiplying) with the matrix. They are shown as the red-colored vectors in the figure below:



You can easily implement PCA in Python using the PCA class in the scikit-learn package:

|  |
| --- |
| from sklearn.decomposition import PCA |
|  |  |
|  | // say you want to reduce to 2 features |
|  | pca = PCA(n\_components = 2) |
|  |  |
|  | // obtain transformed data |
|  | data\_transformed = pca.fit\_transform(data) |

I applied PCA on the [Digits dataset](https://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html) from sklearn – a collection of 8×8 images of handwritten digits. The plot I obtained is rather impressive. The digits appear nicely clustered:



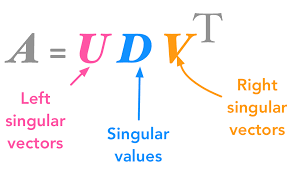
Head on to our Comprehensive Guide to 12 Dimensionality Reduction techniques with code in Python for a deeper insight into PCA and 11 other Dimensionality Reduction techniques. It is honestly one of the best articles on this topic you will find anywhere.

### 6. Singular Value Decomposition

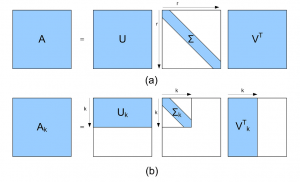
In my opinion, Singular Value Decomposition (SVD) is underrated and not discussed enough. It is an amazing technique of **matrix decomposition** with diverse applications. I will try and cover a few of them in a future article.

For now, let us talk about SVD in Dimensionality Reduction. Specifically, this is known as **Truncated SVD**.

* We start with the large m x n numerical data matrix A, where m is the number of rows and n is the number of features
* Decompose it into 3 matrices as shown here:



Choose k singular values based on the diagonal matrix and truncate (trim) the 3 matrices accordingly:

[](https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2019/07/svd1.png)

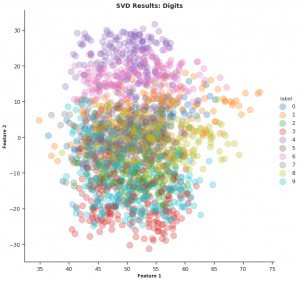
*Source: researchgate.net*

* Finally, multiply the truncated matrices to obtain the transformed matrix *A\_k*. It has the dimensions m x k. So, it has k features with k < n

Here is the code to implement truncated SVD in Python (it’s quite similar to PCA):

|  |
| --- |
| from sklearn.decomposition import TruncatedSVD |
|  |  |
|  | // say you want to reduce to 2 features |
|  | svd = TruncatedSVD(n\_features = 2) |
|  |  |
|  | //obtain the transformed data |
|  | data\_transformed = svd.fit\_transform(data) |

On applying truncated SVD to the Digits data, I got the below plot. You’ll notice that it’s not as well clustered as we obtained after PCA:



## Natural Language Processing (NLP)

Natural language Processing (NLP) is the hottest field in data science right now. This is primarily down to major breakthroughs in the last 18 months. If you were still undecided on which branch to opt for – you should strongly consider NLP.

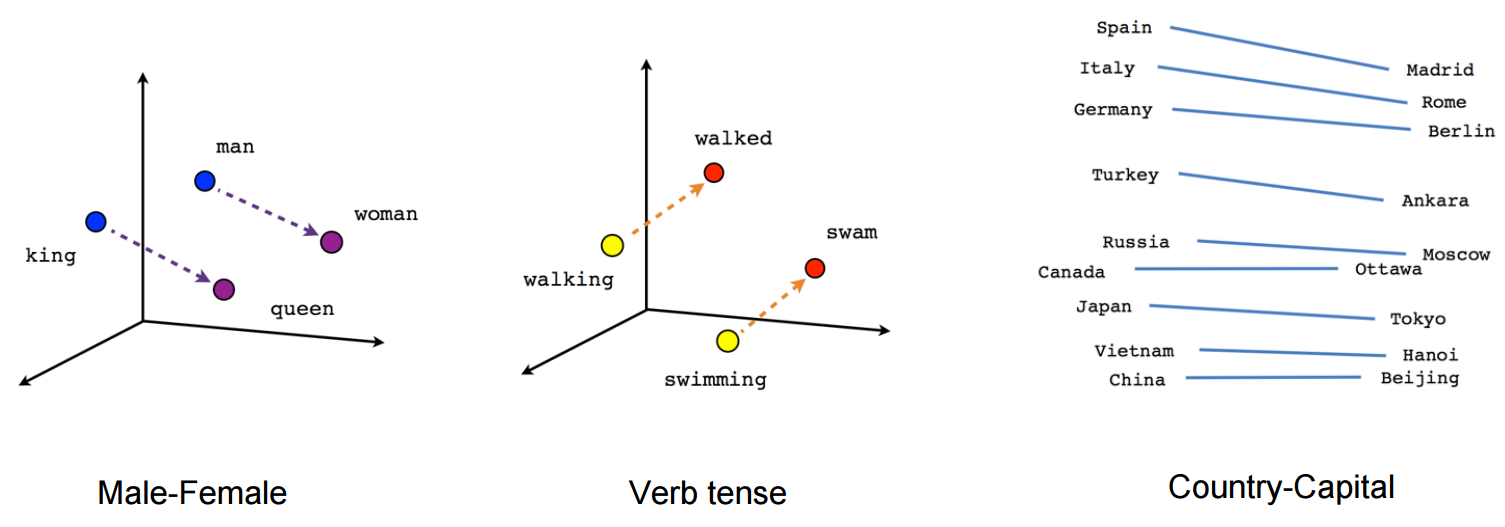
So let’s see a couple of interesting applications of linear algebra in NLP. This should help swing your decision!

### 7. Word Embeddings

Machine learning algorithms cannot work with raw textual data. We need to convert the text into some numerical and statistical features to create model inputs. There are many ways for engineering features from text data, such as:

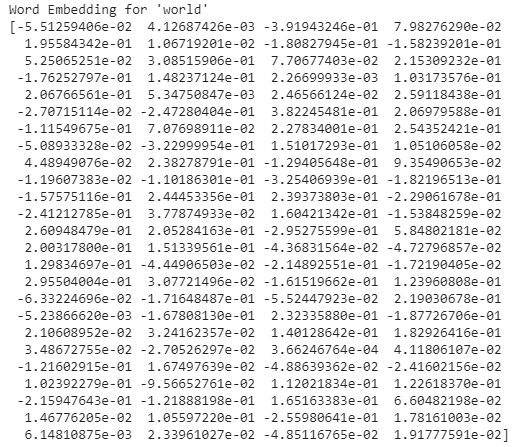
1. Meta attributes of a text, like word count, special character count, etc.
2. NLP attributes of text using Parts-of-Speech tags and Grammar Relations like the number of proper nouns
3. Word Vector Notations or Word Embeddings

Word Embeddings is a way of representing words as **low dimensional vectors** of numbers while preserving their context in the document. These representations are obtained by training different neural networks on a large amount of text which is called a **corpus**. They also help in analyzing syntactic similarity among words:



**Word2Vec** and **GloVe** are two popular models to create Word Embeddings.

I trained my model on the [Shakespeare corpus](https://norvig.com/ngrams/shakespeare.txt) after some light preprocessing using Word2Vec and obtained the word embedding for the word ‘world’:



Pretty cool! But what’s even more awesome is the below plot I obtained for the vocabulary. Observe that syntactically similar words are closer together. I have highlighted a few such clusters of words. The results are not perfect but they are still quite amazing:



There are several other methods to obtain Word Embeddings.

### 8. Latent Semantic Analysis (LSA)

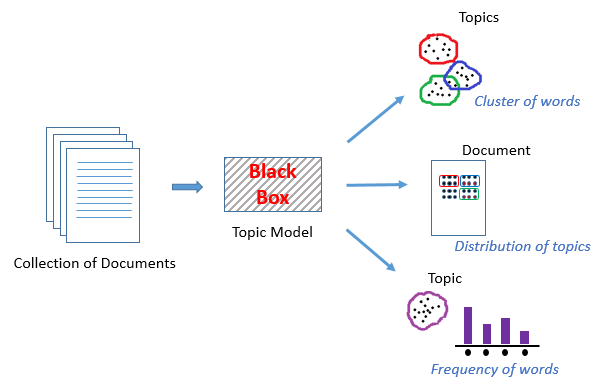
What is your first thought when you hear this group of words – “prince, royal, king, noble”? These very different words are **almost synonymous**.

Now, consider the following sentences:

* The pitcher of the Home team seemed out of form
* There is a pitcher of juice on the table for you to enjoy

The word ‘pitcher’ has **different meanings based on the other words in the two sentences**. It means a baseball player in the first sentence and a jug of juice in the second.

Both these sets of words are easy for us humans to interpret with years of experience with the language. But what about machines? Here, the NLP concept of Topic Modeling comes into play:



Topic Modeling **is an unsupervised technique to find topics across various text documents.** These topics are nothing but clusters of related words. Each document can have multiple topics. The topic model outputs the various topics, their distributions in each document, and the frequency of different words it contains.

Latent Semantic Analysis (LSA), or Latent Semantic Indexing, is one of the techniques of Topic Modeling. It is another application of **Singular Value Decomposition**.

Latent means ‘hidden’. True to its name, LSA attempts to capture the hidden themes or topics from the documents by leveraging the context around the words.

I will describe the steps in LSA in short so make sure you check out this Simple Introduction to Topic Modeling using Latent Semantic Analysis with code in Python for a proper and in-depth understanding.

* First, generate the **Document-Term** matrix for your data
* Use SVD to decompose the matrix into 3 matrices:
  + **Document-Topic** matrix
  + **Topic Importance** Diagonal Matrix
  + **Topic-term** matrix
* Truncate the matrices based on the importance of topics

|  |
| --- |
| # create document term matrix for your data |
|  | # you can use TfidfVectorizer instead of CountVectorizer as well |
|  | from sklearn.feature\_extraction.text import CountVectorizer |
|  | cvec = CountVectorizer() |
|  | docTermMat = cvec.fit\_transform(data['text'].values) |
|  |  |
|  | # truncated SVD to preserve 20 topics |
|  | from sklearn.decomposition import TruncatedSVD |
|  | lsa = TruncatedSVD(n\_components = 20, n\_iter = 500) |
|  | lsa.fit(docTermMat) |

For a hands-on experience with Natural Language Processing, you can check out our course on NLP using Python. The course is beginner-friendly and you get to build 5 real-life projects!

## Computer Vision

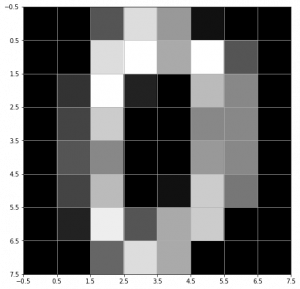
Another field of deep learning that is creating waves – Computer Vision. If you’re looking to expand your skillset beyond tabular data (and you should), then learn how to work with images.

This will broaden your current understanding of machine learning and also help you crack interviews quickly.

### 9. Image Representation as Tensors

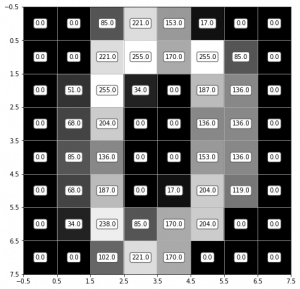
How do you account for the ‘vision’ in Computer Vision? Obviously, a computer does not process images as humans do. Like I mentioned earlier, machine learning algorithms need numerical features to work with.

A digital image is made up of small indivisible units called pixels. Consider the figure below:



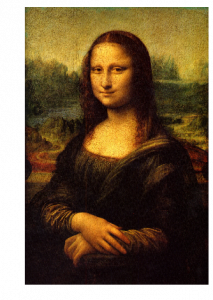
This **grayscale image** of the digit zero is made of 8 x 8 = 64 pixels. Each pixel has a value in the range 0 to 255. A value of 0 represents a black pixel and 255 represents a white pixel.

Conveniently, an m x n grayscale image can be represented as a **2D matrix** with m rows and n columns with the cells containing the respective pixel values:

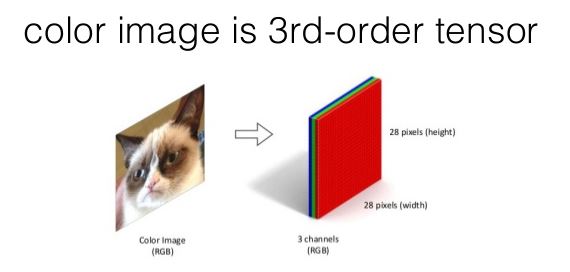


But what about a **colored image**? A colored image is generally stored in the RGB system. Each image can be thought of as being represented by three 2D matrices, one for each R, G and B channel. A pixel value of 0 in the R channel represents zero intensity of the Red color and of 255 represents the full intensity of the Red color.

Each pixel value is then a combination of the corresponding values in the three channels:

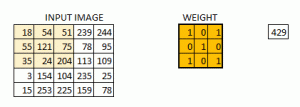
In reality, instead of using 3 matrices to represent an image, a **tensor** is used. A tensor is a **generalized n-dimensional matrix**. For an RGB image, a 3rd ordered tensor is used. Imagine it as three 2D matrices stacked one behind another:



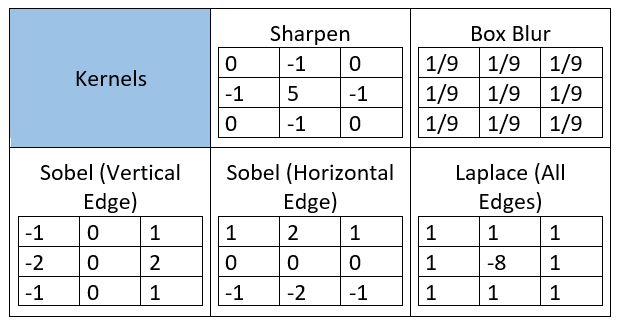
### 10. Convolution and Image Processing

**2D Convolution** is a very important operation in image processing. It consists of the below steps:

1. Start with a small matrix of weights, called a **kernel** or a filter
2. Slide this kernel on the 2D input data, performing element-wise multiplication
3. Add the obtained values and put the sum in a single output pixel

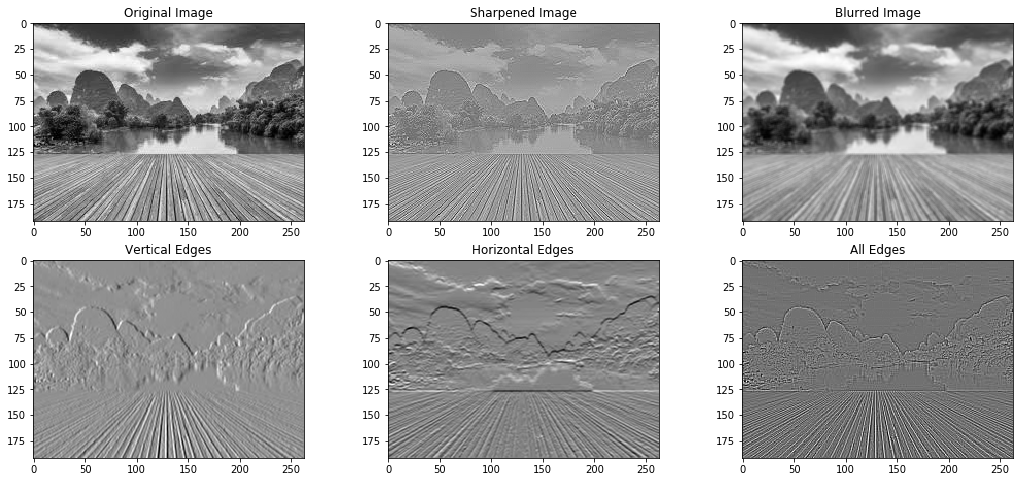


The function can seem a bit complex but it’s widely used for performing various image processing operations like **sharpening and blurring the images and edge detection**. We just need to know the right kernel for the task we are trying to accomplish. Here are a few kernels you can use:



|  |
| --- |
| # import required libraries |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  | import cv2 |
|  | from skimage.color import rgb2gray |
|  | from scipy import ndimage |
|  |  |
|  | # read the image |
|  | img = cv2.imread('1.jpeg') |
|  |  |
|  | # imread returns image in BRG format by default, convert it to RGB |
|  | img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) |
|  |  |
|  | # convert to grayscale for 2D convolution |
|  | gray = rgb2gray(img) |
|  |  |
|  | kernel = # define your kernel as a 2d numpy array |
|  |  |
|  | output = ndimage.convolve(gray, kernel, mode='reflect') |
|  | # The mode parameter determines how the input array is extended when the filter overlaps a border |
|  |  |
|  | # plot the output image |
|  | plt.imshow(output, cmap = 'gray') |

You can download [the image I used](https://drive.google.com/file/d/1aM4otWKSsDz1Rof3LZkY055YkYXeO-vf/view) and try these image processing operations for yourself using the code and the kernels above. Also, try this Computer Vision tutorial on image Segmentation techniques!



Amazing, right? This is by far my most favorite application of Linear Algebra in Data Science.

Now that you are acquainted with the basics of Computer Vision, it is time to start your Computer Vision journey with 16 awesome OpenCV functions. We also have a comprehensive course on Computer Vision using Deep Learning in which you can work on real-life Computer Vision case studies!

## End Notes

My aim here was to make Linear Algebra a bit more interesting than you might have imagined previously. Personally for me, learning about applications of a subject motivates me to learn more about it.

I am sure you are as impressed with these applications as I am. Or perhaps you know of some other applications that I could add to the list? Let me know in the comments section below.

## Table of contents

1. Motivation – Why learn Linear Algebra?
2. Representation of problems in Linear Algebra  
   2.1. Visualising the problem: Line  
   2.2. Complicate the problem  
   2.3. Planes
3. Matrix  
   3.1 Terms related to Matrix  
   3.2 Basic operations on Matrix  
   3.3 Representing in Matrix form
4. Solving the problem  
   4.1. Row Echelon form  
   4.2. Inverse of a Matrix  
   4.2.1  Finding Inverse  
   4.2.2 The power of Matrices: solving the equations in one go  
   4.2.3 Use of Inverse in Data Science
5. Eigenvalues and Eigenvectors  
   5.1 Finding Eigenvectors  
   5.2 Use of Eigenvectors in Data Science: PCA algorithm
6. Singular Value Decomposition of a Matrix
7. End Notes

## 1. Motivation – Why learn Linear Algebra?

I would like to present 4 scenarios to showcase why learning Linear Algebra is important, if you are learning Data Science and Machine Learning.

### Scenario 1:



What do you see when you look at the image above? You most likely said flower, leaves -not too difficult. But, if I ask you to write that logic so that a computer can do the same for you – it will be a very difficult task (to say the least).

You were able to identify the flower because the human brain has gone through million years of evolution. We do not understand what goes in the background to be able to tell whether the colour in the picture is red or black. We have somehow trained our brains to automatically perform this task.

But making a computer do the same task is not an easy task, and is an active area of research in Machine Learning and Computer Science in general. But before we work on identifying attributes in an image, let us ponder over a particular question- How does a machine stores this image?

You probably know that computers of today are designed to process only 0 and 1. So how can an image such as above with multiple attributes like colour be stored in a computer? This is achieved by storing the pixel intensities in a construct called **Matrix.**Then, this matrix can be processed to identify colours etc.

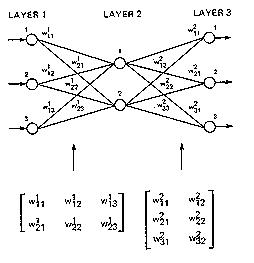
So any operation which you want to perform on this image would likely use Linear Algebra and matrices at the back end.

### Scenario 2:

If you are somewhat familiar with the Data Science domain, you might have heard about the world “XGBOOST” – an algorithm employed most frequently by winners of Data Science Competitions. It stores the numeric data in the form of **Matrix** to give predictions. It enables XGBOOST to process data faster and provide more accurate results. Moreover, not just XGBOOST but various other algorithms use Matrices to store and process data.

### Scenario 3:

Deep Learning- the new buzz word in town employs Matrices to store inputs such as image or speech or text to give a state-of-the-art solution to these problems. Weights learned by a Neural Network are also stored in Matrices. Below is a graphical representation of weights stored in a Matrix.



### Scenario 4:

Another active area of research in Machine Learning is dealing with text and the most common techniques employed are Bag of Words, Term Document Matrix etc. All these techniques in a very similar manner store counts(or something similar) of words in documents and store this frequency count in a Matrix form to perform tasks like Semantic analysis, Language translation, Language generation etc.

So, now you would understand the importance of Linear Algebra in machine learning. We have seen image, text or any data, in general, employing matrices to store and process data. This should be motivation enough to go through the material below to get you started on Linear Algebra. This is a relatively long guide, but it builds Linear Algebra from the ground up.

## 2. Representation of problems in Linear Algebra

Let’s start with a simple problem. Suppose that price of 1 ball & 2 bat or 2 ball and 1 bat is 100 units. We need to find price of a ball and a bat.

Suppose the price of a bat is Rs ‘x’ and the price of a ball is Rs ‘y’. Values of ‘x’ and ‘y’ can be anything depending on the situation i.e. ‘x’ and ‘y’ are variables.

Let’s translate this in mathematical form –

2x + y = 100 ...........(1)

Similarly, for the second condition-

x + 2y  =  100 ..............(2)

Now, to find the prices of bat and ball, we need the values of ‘x’ and ‘y’ such that it satisfies both the equations. The basic problem of linear algebra is to find these values of ‘x’ and ‘y’ i.e. the solution of a set of linear equations.

Broadly speaking, in linear algebra data is represented in the form of linear equations. These linear equations are in turn represented in the form of matrices and vectors.

The number of variables as well as the number of equations may vary depending upon the condition, but the representation is in form of matrices and vectors.

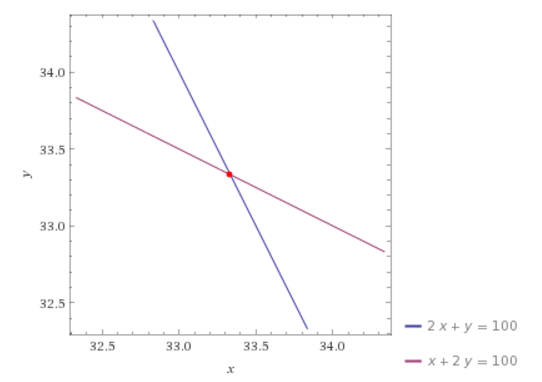
### 2.1 Visualise the problem

It is usually helpful to visualize data problems. Let us see if that helps in this case.

Linear equations represent flat objects. We will start with the simplest one to understand i.e. line. A line corresponding to an equation is the set of all the points which satisfy the given equation. For example,

Points (50,0) , (0,100), (100/3,100/3) and (30,40) satisfy our  equation (1) . So these points should lie on the line corresponding to our equation (1). Similarly, (0,50),(100,0),(100/3,100/3) are some of the points that satisfy equation (2).

Now in this situation, we want both of the conditions to be satisfied i.e. the point which lies on both the lines.  Intuitively, we want to find the intersection point of both the lines as shown in the figure below.



Let’s solve the problem by elementary algebraic operations like addition, subtraction and substitution.

2x + y = 100 .............(1)

x + 2y = 100 ..........(2)

from equation (1)-

y = (100- x)/2

put value of y in equation (2)-

x + 2\*(100-x)/2 = 100......(3)

Now, since the equation (3) is an equation in single variable x, it can be solved for **x** and subsequently **y.**

That looks simple – let’s go one step further and explore.

### 2.2 Let’s complicate the problem

Now, suppose you are given a set of three conditions with three variables each as given below and asked to find the values of all the variables. Let’s solve the problem and see what happens.

x+y+z=1.......(4)

2x+y=1......(5)

5x+3y+2z=4.......(6)

From equation (4) we get,

z=1-x-y....(7)

Substituting value of z in equation (6), we get –

5x+3y+2(1-x-y)=4

3x+y=2.....(8)

Now, we can solve equations (8) and (5) as a case of two variables to find the values of ‘**x**’ and ‘**y**’ in the problem of bat and ball above. Once we know‘**x**’ and ‘**y**’, we can use (7)  to find the value of ‘**z**’.

As you might see, adding an extra variable has tremendously increased our efforts for finding the solution of the problem. Now imagine having 10 variables and 10 equations. Solving 10 equations simultaneously can prove to be tedious and time consuming. Now dive into data science. We have millions of data points. How do you solve those problems?

We have millions of data points in a real data set. It is going to be a nightmare to reach to solutions using the approach mentioned above. And imagine if we have to do it again and again and again. It’s going to take ages before we can solve this problem. And now if I tell you that it’s just one part of the battle, what would you think? So, what should we do? Should we quit and let it go? Definitely NO. Then?

Matrix is used to solve a large set of linear equations. But before we go further and take a look at matrices, let’s visualise the physical meaning of our problem. Give a little bit of thought to the next topic. It directly relates to the usage of Matrices.

### 2.3 Planes

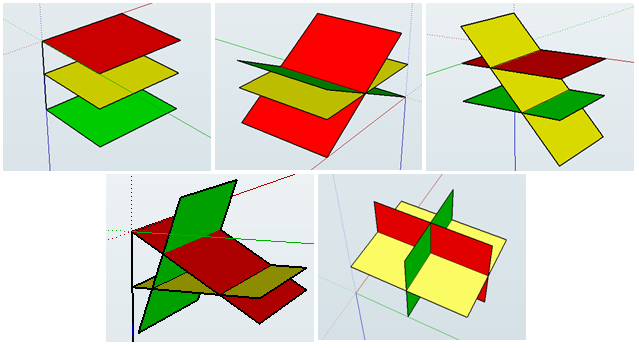
A linear equation in 3 variables represents the set of all points whose coordinates satisfy the equations. Can you figure out the physical object represented by such an equation? Try to think of 2 variables at a time in any equation and then add the third one. You should figure out that it represents a three-dimensional analogue of line.

Basically, a linear equation in three variables represents a plane. More technically, a plane is a flat geometric object which extends up to infinity.

As in the case of a line, finding solutions to 3 variables linear equation means we want to find the intersection of those planes. Now can you imagine, in how many ways a set of three planes can intersect? Let me help you out. There are 4 possible cases –

1. No intersection at all.
2. Planes intersect in a line.
3. They can intersect in a plane.
4. All the three planes intersect at a point.

Can you imagine the number of solutions in each case? Try doing this. Here is an aid picked from Wikipedia to help you visualise.



So, what was the point of having you to visualise all graphs above?

Normal humans like us and most of the super mathematicians can only visualise things in 3-Dimensions, and having to visualise things in 4 (or 10000) dimensions is  impossible for mortals. So, how do mathematicians deal with higher dimensional data so efficiently? They have tricks up their sleeves and Matrices is one such trick employed by mathematicians to deal with higher dimensional data.

Now let’s proceed with our main focus i.e. Matrix.

## 3. Matrix

Matrix is a way of writing similar things together to handle and manipulate them as per our requirements easily. In Data Science, it is generally used to store information like weights in an Artificial Neural Network while training various algorithms. You will be able to understand my point by the end of this article.

Technically, a matrix is a 2-D array of numbers (as far as Data Science is concerned). For example look at the matrix A below.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Generally, rows are denoted by ‘i’ and column are denoted by ‘j’.  The elements are indexed by ‘**i**’th row and ‘**j**’th column.We denote the matrix by some alphabet e.g.  A and its elements by A(ij).

In above matrix

A12 =  2

To reach to the result, go along first row and reach to second column.

### 3.1 Terms related to Matrix

**Order of matrix** – If a matrix has 3 rows and 4 columns, order of the matrix is 3\*4 i.e. row\*column.

**Square matrix** – The matrix in which the number of rows is equal to the number of columns.

**Diagonal matrix** – A matrix with all the non-diagonal elements equal to 0 is called a diagonal matrix.

**Upper triangular matrix** – Square matrix with all the elements below diagonal equal to 0.

**Lower triangular matrix** – Square matrix with all the elements above the diagonal equal to 0.

**Scalar matrix** – Square matrix with all the diagonal elements equal to some constant k.

**Identity matrix** – Square matrix with all the diagonal elements equal to 1 and all the non-diagonal elements equal to 0.

**Column matrix** –  The matrix which consists of only 1 column. Sometimes, it is used to represent a vector.

**Row matrix** –  A matrix consisting only of row.

**Trace** – It is the sum of all the diagonal elements of a square matrix.

### 3.2 Basic operations on matrix

Let’s play with matrices and realise the capabilities of matrix operations.

**Addition** – Addition of matrices is almost similar to basic arithmetic addition. All you need is the order of all the matrices being added should be same. This point will become obvious once you will do matrix addition by yourself.

Suppose we have 2 matrices ‘A’ and ‘B’ and the resultant matrix after the addition is ‘C’. Then

Cij  =   Aij + Bij

For example, let’s take two matrices and solve them.

A      =

|  |  |
| --- | --- |
| 1 | 0 |
| 2 | 3 |

B    =

|  |  |
| --- | --- |
| 4 | -1 |
| 0 | 5 |

Then,

C        =

|  |  |
| --- | --- |
| 5 | -1 |
| 2 | 8 |

Observe that to get the elements of C matrix, I have added A and B element-wise i.e. 1 to 4, 3 to 5 and so on.

**Scalar Multiplication** –  Multiplication of a matrix with a scalar constant is called scalar multiplication. All we have to do in a scalar multiplication is to multiply each element of the matrix with the given constant.  Suppose we have a constant scalar ‘c’ and a matrix ‘A’.  Then multiplying ‘c’ with ‘A’  gives-

c[Aij] =  [c\*Aij]

**Transposition** – Transposition simply means interchanging the row and column index. For example-

AijT= Aji

Transpose is used in vectorized implementation of linear and logistic regression.

**Code in python**

**Main.py**

**Login/signup to view & run code in the browser**

**Code in R**

|  |
| --- |
| A<-matrix(c(11,12,13,14,15,16,17,18,19),nrow = 3,byrow = T) |
|  | A |

**Output**

[,1] [,2] [,3]

[1,] 11 12 13

[2,] 14 15 16

[3,] 17 18 19

|  |
| --- |
| #Transpose of a matrix |
|  | t(A) |

t(A)

[,1] [,2] [,3]

[1,] 11 14 17

[2,] 12 15 18

[3,] 13 16 19

**Matrix multiplication**

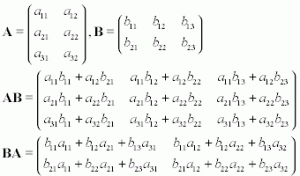
Matrix multiplication is one of the most frequently used operations in linear algebra. We will learn to multiply two matrices as well as go through its important properties.

Before landing to algorithms, there are a few points to be kept in mind.

1. The multiplication of two matrices of orders i\*j and j\*k results into a matrix of order i\*k.  Just keep the outer indices in order to get the indices of the final matrix.
2. Two matrices will be compatible for multiplication only if the number of columns of the first matrix and the number of rows of the second one are same.
3. The third point is that order of multiplication matters.

Don’t worry if you can’t get these points. You will be able to understand by the end of this section.

Suppose, we are given two matrices A and B to multiply. I will write the final expression first and then will explain the steps.



I have picked this image from Wikipedia for your better understanding.

In the first illustration, we know that the order of the resulting matrix should be 3\*3. So first of all, create a matrix of order 3\*3. To determine (AB)ij , multiply each element of ‘i’th row of A with ‘j’th column of B one at a time and add all the terms. To help you understand element-wise multiplication, take a look at the code below.

import numpy as np

A=np.arange(21,30).reshape(3,3)  
B=np.arange(31,40).reshape(3,3)

A.dot(B)

AB= array([[2250, 2316, 2382],

[2556, 2631, 2706],

[2862, 2946, 3030]]) B.dot(A)

BA= array([[2310, 2406, 2502],

[2526, 2631, 2736],

[2742, 2856, 2970]])

So, how did we get 2250 as first element of AB matrix?  2250=21\*31+22\*34+23\*37. Similarly, for other elements.

**Code in R**

|  |
| --- |
| #Multiplication of matrix |
|  |  |
|  | A<-matrix(c(11,12,13,14,15,16,17,18,19),nrow = 3,byrow = T) |
|  | B<-matrix(c(20,21,22,23,24,25,26,27,28),nrow = 3,byrow = T) |
|  |  |
|  | A\*B |

A\*B

[,1] [,2] [,3]

[1,] 220 252 286

[2,] 322 360 400

[3,] 442 486 532

Notice the difference between AB and BA.

**Properties of matrix multiplication**

1. Matrix multiplication is associative provided the given matrices are compatible for multiplication i.e.

ABC =  (AB)C = A(BC)

import numpy as np  
A=np.arange(21,30).reshape(3,3)  
B=np.arange(31,40).reshape(3,3)  
C=np.arange(41,50).reshape(3,3)

temp1=(A.dot(B)).dot(C)

array([[306108, 313056, 320004],

[347742, 355635, 363528],

[389376, 398214, 407052]])

temp2=A.dot((B.dot(C)))

array([[306108, 313056, 320004],

[347742, 355635, 363528],

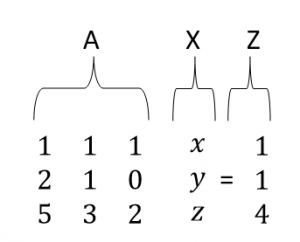
[389376, 398214, 407052]])

2. Matrix multiplication is not commutative i.e. AB and  BA are not equal. We have verified this result above.

Matrix multiplication is used in linear and logistic regression when we calculate the value of output variable by parameterized vector method. As we have learned the basics of matrices, it’s time to apply them.

### 3.3 Representing equations in matrix form

Let me do something exciting for you.  Take help of pen and paper and try to find the value of the matrix multiplication shown below



It can be verified very easily that the expression contains our three equations. We will name our matrices as ‘A’, ‘X’ and ‘Z’.

It explicitly verifies that we can write our equations together in one place as

AX   = Z

Next step has to be solution methods.We will go through two methods to find the solution.

## 4. Solving the Problem

Now, we will look in detail the two methods to solve matrix equations.

1. Row Echelon Form
2. Inverse of a Matrix

### 4.1 Row Echelon form

Now you have visualised what an equation in 3 variables represents and had a warm up on matrix operations. Let’s find the solution of the set of equations given to us to understand our first method of interest and explore it later in detail.

I have already illustrated that solving the equations by substitution method can prove to be tedious and time taking. Our first method introduces you with a neater and more systematic method to accomplish the job in which, we manipulate our original equations systematically to find the solution.  But what are those valid manipulations? Are there any qualifying criteria they have to fulfil? Well, yes. There are two conditions which have to be fulfilled by any manipulation to be valid.

1. Manipulation should preserve the solution i.e. solution should not be altered on imposing the manipulation.
2. Manipulation should be reversible.

So, what are those manipulations?

1. We can swap the order of equations.
2. We can multiply both sides of equations by any non-zero constant ‘c’.
3. We can multiply an equation by any non-zero constant and then add to other equation.

These points will become more clear once you go through the algorithm and practice it. The basic idea is to clear variables in successive equations and form an upper triangular matrix. Equipped with prerequisites, let’s get started. But before that, it is strongly recommended to go through this [link](https://www.youtube.com/watch?v=0-GaihnICmo&index=17&list=PLAwxTw4SYaPlH16rY8KgDwciMZPxCnCX_) for better understanding.

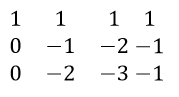
I will solve our original problem as an illustration. Let’s do it in steps.

1. Make an augmented matrix from the matrix ‘A’ and ‘Z’.

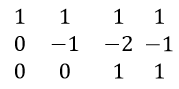


What I have done is I have just concatenated the two matrices. The augmented matrix simply tells that the elements in a row are coefficients of ‘x’, ‘y’ and ‘z’ and last element in the row is right-hand side of the equation.

1. Multiply row (1) with 2 and subtract from row (2). Similarly, multiply equation 1 with 5 and subtract from row (3).



In order to make an upper triangular matrix, multiply row (2) by 2 and then subtract from row (3).



Remember to make each leading coefficient, also called pivot equal to 1, by suitable manipulations; in this case multiplying row 2 with -1. Also, if a row consists of 0 only, it should be below each row which consists of a non-zero entry. The resulting form of Matrix is called Row Echelon form. Notice that the planes corresponding to new equations formed by manipulation are not equivalent. Doing these operations, we are just conserving the solution of equations and trying to reach to it.

1. Now we have simplified our job, let’s retrieve the modified equations. We will start from the simplest i.e. the one with the minimum number of remaining variables. If you follow the illustrated procedure, you will find that last equation comes to be the simplest one.

0\*x+0\*y+1\*z=1  
z=1

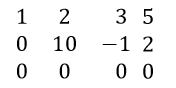
Now retrieve equation (2) and put the value of ‘z’ in it to find ‘y’. Do the same for equation (1).

Isn’t it pretty simple and clean?

Let’s ponder over another point. Will we always be able to make an upper triangular matrix which gives a unique solution? Are there different cases possible? Recall that planes can intersect in multiple ways. Take your time to figure it out and then proceed further.

Different possible cases-

1. It’s possible that we get a unique solution as illustrated in above example. It indicates that all the three planes intersect in a point.
2. We can get a case like shown below



Note that in last equation, 0=0 which is always true but it seems like we have got only 2 equations. One of the equations is redundant. In many cases, it’s also possible that the number of redundant equations is more than one. In this case, the number of solutions is infinite.

1. There is another case where Echelon matrix looks as shown below



Let’s retrieve the last equation.

0\*x+0\*y+0\*z=4

0=4

Is it possible? Very clear cut intuition is NO. But, does this signify something? It’s analogous to saying that it is impossible to find a solution and indeed, it is true. We can’t find a solution for such a set of equations. Can you think what is happening actually in terms of planes? Go back to the section where we saw planes intersecting and find it out.

Note that this method is efficient for a set of 5-6 equations. Although the method is quite simple, if equation set gets larger, the number of times you have to manipulate the equations becomes enormously high and the method becomes inefficient.

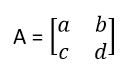
**Rank of a matrix** – Rank of a matrix is equal to the maximum number of linearly independent row vectors in a matrix.

A set of vectors is linearly dependent if we can express at least one of the vectors as a linear combination of remaining vectors in the set.

### 4.2 Inverse of a Matrix

For solving a large number of equations in one go, the inverse is used. Don’t panic if you are not familiar with the inverse. We will do a good amount of work on all the required concepts. Let’s start with a few terms and operations.

**Determinant of a Matrix** – The concept of determinant is applicable to square matrices only. I will lead you to the generalised expression of determinant in steps. To start with, let’s take a 2\*2 matrix  A.

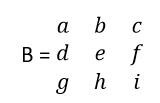


For now, just focus on 2\*2 matrix. The expression of determinant of the matrix A will be:

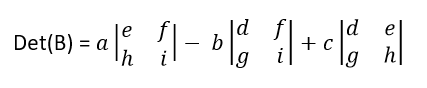
det(A) =a\*d-b\*c

Note that det(A) is a standard notation for determinant. Notice that all you have to do to find determinant in this case is to multiply diagonal elements together and put a positive or negative sign before them. For determining the sign, sum the indices of a particular element. If the sum is an even number, put a positive sign before the multiplication and if the sum is odd, put a negative sign.  For example, the sum of indices of element ‘a11’ is 2. Similarly the sum of indices of element ‘d’ is 4. So we put a positive sign before the first term in the expression.  Do the same thing for the second term yourself.

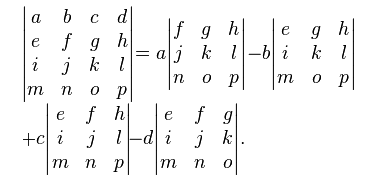
Now take a 3\*3 matrix ‘B’ and find its determinant.



I am writing the expression first and then will explain the procedure step by step.



Each term consists of two parts basically i.e. a submatrix and a coefficient. First of all, pick a constant. Observe that coefficients are picked from the first row only. To start with, I have picked the first element of the first row. You can start wherever you want. Once you have picked the coefficient, just delete all the elements in the row and column corresponding to the chosen coefficient. Next, make a matrix of the remaining elements; each one in its original position after deleting the row and column and find the determinant of this submatrix . Repeat the same procedure for each element in the first row. Now, for determining the sign of the terms, just add the indices of the coefficient element. If it is even, put a positive sign and if odd, put a negative sign. Finally, add all the terms to find the determinant. Now, let’s take a higher order matrix ‘C’ and generalise the concept.



Try to relate the expression to what we have done already and figure out the final expression.

**Code in python**

import numpy as np  
#create a 4\*4 matrix  
arr = np.arange(100,116).reshape(4,4)

array([[100, 101, 102, 103],

[104, 105, 106, 107],

[108, 109, 110, 111],

[112, 113, 114, 115]])

#find the determinant  
np.linalg.det(arr)

-2.9582283945788078e-31

**Code in R**

|  |
| --- |
| #Inverse of matrix |
|  |  |
|  | B<-matrix(c(30,31,40,41,50,51,60,61,70),nrow = 3,byrow = T) |
|  |  |
|  | A<-solve(B) |
|  | A |
|  |  |
|  | #Determinant of A |
|  | det(A) |

[,1] [,2] [,3]

[1,] -0.16208333 -0.1125 0.17458333

[2,] -0.07916667 0.1250 -0.04583333

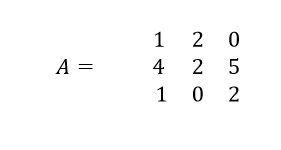
[3,] 0.20791667 -0.0125 -0.09541667

#Determinant

-0.0004166667

**Minor of a matrix**

Let’s take a square matrix A. then minor corresponding to an element A(ij)  is the determinant of the submatrix formed by deleting the ‘i’th  row and ‘j’th column of the matrix. Hope you can relate with what I have explained already in the determinant section. Let’s take an example.



To find the minor corresponding to element A11, delete first row and first column to find the submatrix.



Now find the determinant of this matrix as explained already. If you calculate the determinant of this matrix, you should get 4. If we denote minor by M11, then

M11 = 4

Similarly, you can do for other elements.

**Cofactor of a matrix**

In the above discussion of minors, if we consider signs of minor terms, the resultant we get is called cofactor of a matrix. To assign the sign, just sum the indices of the corresponding element. If it turns out to be even, assign positive sign. Else assign negative. Let’s take above illustration as an example. If we add the indices i.e. 1+1=2, so we should put a positive sign. Let’s say it C11. Then

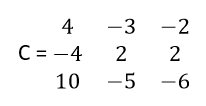
C11 = 4

You should find cofactors corresponding to other elements by yourself for a good amount of practice.

**Cofactor matrix**

Find the cofactor corresponding to each element. Now in the original matrix, replace the original element by the corresponding cofactor. The matrix thus found is called the cofactor matrix corresponding to the original matrix.

For example, let’s take our matrix A. if you have found out the cofactors corresponding to each element, just put them in a matrix according to rule stated above. If you have done it right, you should get cofactor matrix



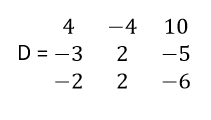
**Adjoint of a matrix** – In our journey to find inverse, we are almost at the end. Just keep hold of the article for a couple of minutes and we will be there. So, next we will find the adjoint of a matrix.

Suppose we have to find the adjoint of a matrix A. we will do it in two steps.

In step 1, find the cofactor matrix of A.

In step 2, just transpose the cofactor matrix.

The resulting matrix is the adjoint of the original matrix. For illustration, lets find the adjoint of our matrix A. we already have cofactor matrix C. Transpose of cofactor matrix should be



Finally, in the next section, we will find the inverse.

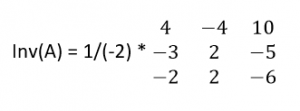
#### 4.2.1 Finding Inverse of a matrix

Do you remember the concept of the inverse of a number in elementary algebra? Well, if there exist two numbers such that upon their multiplication gives 1 then those two numbers are called inverse of each other. Similarly in linear algebra, if there exist two matrices such that their multiplication yields an identity matrix then the matrices are called inverse of each other. If you can not get what I explained, just go with the article. It will come intuitively to you. The best way to learning is learning by doing. So, let’s jump straight to the algorithm for finding the inverse of a matrix A. Again, we will do it in two steps.

**Step 1**: Find out the adjoint of the matrix A by the procedure explained in previous sections.

**Step2:** Multiply the adjoint matrix by the inverse of determinant of the matrix A. The resulting matrix is the inverse of A.

For example, let’s take our matrix A and find it’s inverse. We already have the adjoint matrix. Determinant of matrix A comes to be -2. So, its inverse will be



Now suppose that the determinant comes out to be 0. What happens when we invert the determinant i.e. 0?  Does it make any sense?  It indicates clearly that we can’t find the inverse of such a matrix. Hence, this matrix is non-invertible. More technically, this type of matrix is called a singular matrix.

Keep in mind that the resultant of multiplication of a matrix and its inverse is an identity matrix. This property is going to be used extensively in equation solving.

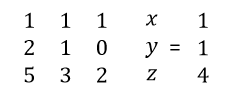
Inverse is used in finding parameter vector corresponding to minimum cost function in linear regression.

#### 4.2.2 Power of matrices

What happens when we multiply a number by 1? Obviously it remains the same. The same is applicable for an identity matrix i.e. if we multiply a matrix with an identity matrix of the same order, it remains same.

Lets solve our original problem with the help of matrices. Our original problem represented in matrix was as shown below

AX = Z i.e.



What happens when we pre multiply both the sides with inverse of coefficient matrix i.e. A. Lets find out by doing.

A-1A X =A-1 Z

We can manipulate it as,

(A-1 A) X = A -1Z

But we know multiply a matrix with its inverse gives an Identity Matrix. So,

IX =  A -1Z

Where I is the identity matrix of the corresponding order.

If you observe keenly, we have already reached to the solution. Multiplying identity matrix to X does not change it. So the equation becomes

X = A -1Z

For solving the equation, we have to just find the inverse. It can be very easily done by executing a few lines of codes. Isn’t it a really powerful method?

**Code for inverse in python**

import numpy as np  
#create an array arr1  
arr1 = np.arange(5,21).reshape(4,4)

#find the inverse  
np.linalg.inv(arr1)

#### 4.2.3 Application of inverse in Data Science

Inverse is used to calculate parameter vector by normal equation in linear equation. Here is an illustration. Suppose we are given a data set as shown below-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Team** | **League** | **Year** | **RS** | **RA** | **W** | **OBP** | **SLG** | **BA** | **G** | **OOBP** | **OSLG** | | ARI | NL | 2012 | 734 | 688 | 81 | 0.328 | 0.418 | 0.259 | 162 | 0.317 | 0.415 | | ATL | NL | 2012 | 700 | 600 | 94 | 0.32 | 0.389 | 0.247 | 162 | 0.306 | 0.378 | | BAL | AL | 2012 | 712 | 705 | 93 | 0.311 | 0.417 | 0.247 | 162 | 0.315 | 0.403 | | BOS | AL | 2012 | 734 | 806 | 69 | 0.315 | 0.415 | 0.26 | 162 | 0.331 | 0.428 | | CHC | NL | 2012 | 613 | 759 | 61 | 0.302 | 0.378 | 0.24 | 162 | 0.335 | 0.424 | | CHW | AL | 2012 | 748 | 676 | 85 | 0.318 | 0.422 | 0.255 | 162 | 0.319 | 0.405 | | CIN | NL | 2012 | 669 | 588 | 97 | 0.315 | 0.411 | 0.251 | 162 | 0.305 | 0.39 | | CLE | AL | 2012 | 667 | 845 | 68 | 0.324 | 0.381 | 0.251 | 162 | 0.336 | 0.43 | | COL | NL | 2012 | 758 | 890 | 64 | 0.33 | 0.436 | 0.274 | 162 | 0.357 | 0.47 | | DET | AL | 2012 | 726 | 670 | 88 | 0.335 | 0.422 | 0.268 | 162 | 0.314 | 0.402 | | HOU | NL | 2012 | 583 | 794 | 55 | 0.302 | 0.371 | 0.236 | 162 | 0.337 | 0.427 | | KCR | AL | 2012 | 676 | 746 | 72 | 0.317 | 0.4 | 0.265 | 162 | 0.339 | 0.423 | | LAA | AL | 2012 | 767 | 699 | 89 | 0.332 | 0.433 | 0.274 | 162 | 0.31 | 0.403 | | LAD | NL | 2012 | 637 | 597 | 86 | 0.317 | 0.374 | 0.252 | 162 | 0.31 | 0.364 | |

It describes the different variables of different baseball teams to predict whether it makes to playoffs or not. But for right now to make it a regression problem, suppose we are interested in predicting OOBP from the rest of the variables. So, ‘OOBP’ is our target variable. To solve this problem using linear regression, we have to find parameter vector. If you are familiar with Normal equation method, you should have the idea that to do it, we need to make use of Matrices. Lets proceed further and denote our Independent variables below as matrix ‘X’.This data is a part of a data set taken from analytics edge. Here is the [link](https://d37djvu3ytnwxt.cloudfront.net/assets/courseware/v1/dfb1bb5463c388fb167745888e3a6dd9/asset-v1:MITx+15.071x_3+1T2016+type@asset+block/baseball.csv) for the data set.

so,  X=

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | 734 | 688 | 81 | 0.328 | 0.418 | 0.259 | | 700 | 600 | 94 | 0.32 | 0.389 | 0.247 | | 712 | 705 | 93 | 0.311 | 0.417 | 0.247 | | 734 | 806 | 69 | 0.315 | 0.415 | 0.26 | | 613 | 759 | 61 | 0.302 | 0.378 | 0.24 | | 748 | 676 | 85 | 0.318 | 0.422 | 0.255 | | 669 | 588 | 97 | 0.315 | 0.411 | 0.251 | | 667 | 845 | 68 | 0.324 | 0.381 | 0.251 | | 758 | 890 | 64 | 0.33 | 0.436 | 0.274 | | 726 | 670 | 88 | 0.335 | 0.422 | 0.268 | | 583 | 794 | 55 | 0.302 | 0.371 | 0.236 | | 676 | 746 | 72 | 0.317 | 0.4 | 0.265 | | 767 | 699 | 89 | 0.332 | 0.433 | 0.274 | | 637 | 597 | 86 | 0.317 | 0.374 | 0.252 | |

|  |
| --- |
| 0.259  0.247  0.247  0.26  0.24  0.255  0.251  0.251  0.274  0.268  0.236  0.265  0.274  0.252 |

To find the final parameter vector(θ) assuming our initial function is parameterised by θ and X , all you have to do is to find the inverse of (XTX) which can be accomplished very easily by using code as shown below.

First of all, let me make the Linear Regression formulation easier for you to comprehend.

f θ (X)= θTX, where θ is the parameter we wish to calculate and X is the column vector of features or independent variables.

import pandas as pd  
import numpy as np

#you don’t need to bother about the following. It just #transforms the data from original source into matrix

Df = pd.read\_csv( "../baseball.csv”)  
Df1 = df.head(14)

# We are just taking 6 features to calculate θ.  
X = Df1[[‘RS’, ‘RA’, ‘W’, ‘OBP’,'SLG','BA']]  
Y=Df1['OOBP']

#Converting X to matrix  
X = np.asmatrix(X)

#taking transpose of X and assigning it to x  
x= np.transpose(X)

#finding multiplication  
T= x.dot(X)

#inverse of T - provided it is invertible otherwise we use pseudoinverse  
inv=np.linalg.inv(T)

#calculating θ  
theta=(inv.dot(X.T)).dot(Y)

Imagine if you had to solve this set of equations without using linear algebra. Let me remind you that this data set is less than even 1% of original date set. Now imagine if you had to find parameter vector without using linear algebra. It would have taken a lots of time and effort and could be even impossible to solve sometimes.

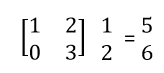
One major drawback of normal equation method when the number of features is large is that it is computationally very costly. The reason is that if there are ‘n’ features, the matrix (XTX) comes to be the order n\*n and its solution costs time of order O( n\*n\*n). Generally, normal equation method is applied when a number of features is of the order of 1000 or 10,000. Data sets with a larger number of features are handled with the help another method called Gradient Descent.

Next, we will go through another advanced concept of linear algebra called Eigenvectors.

## 5. Eigenvalues and Eigenvectors

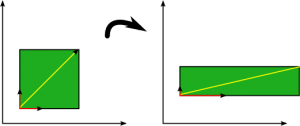
Eigenvectors find a lot of applications in different domains like computer vision, physics and machine learning. If you have studied machine learning and are familiar with Principal component analysis algorithm, you must know how important the algorithm is when handling a large data set. Have you ever wondered what is going on behind that algorithm? Actually, the concept of Eigenvectors is the backbone of this algorithm. Let us explore Eigen vectors and Eigen values for a better understanding of it.

Let’s multiply a 2-dimensional vector with a 2\*2 matrix and see what happens.



This operation on a vector is called linear transformation.  Notice that the directions of input and output vectors are different. Note that the column matrix denotes a vector here.

I will illustrate my point with the help of a picture as shown below.



In the above picture, there are two types of vectors coloured in red and yellow and the picture is showing the change in vectors after a linear transformation. Note that on applying a linear transformation to yellow coloured vector, its direction changes but the direction of the red coloured vector doesn’t change even after applying the linear transformation. The vector coloured in red is an example of Eigenvector.

Precisely, for a particular matrix; vectors whose direction remains unchanged even after applying linear transformation with the matrix are called Eigenvectors for that particular matrix. Remember that the concept of Eigen values and vectors is applicable to square matrices only. Another thing to know is that I have taken a case of two-dimensional vectors but the concept of Eigenvectors is applicable to a space of any number of dimensions.

### 5.1 How to find Eigenvectors of a matrix?

Suppose we have a matrix A and an Eigenvector ‘x’ corresponding to the matrix. As explained already, after multiplication with matrix the direction of ‘x’ doesn’t change. Only change in magnitude is permitted. Let us write it as an equation-

Ax = cx

(A-c)x = 0  …….(1)

Please note that in the term (A-c), ‘c’ denotes an identity matrix of the order equal to ‘A’ multiplied by a scalar ‘c’

We have two unknowns ‘c’ and ‘x’ and only one equation. Can you think of a trick to solve this equation?

In equation (1), if we put the vector ‘x’ as zero vector, it makes no sense. Hence, the only choice is that (A-c) is a singular matrix. And singular matrix has a property that its determinant equals to 0. We will use this property to find the value of ‘c’.

Det(A-c) = 0

Once you find the determinant of the matrix (A-c) and equate to 0, you will get an equation in ‘c’ of the order depending upon the given matrix A. all you have to do is to find the solution of the equation. Suppose that we find solutions as ‘c1’ , ‘c2’ and so on. Put ‘c1’ in equation (1) and find the vector ‘x1’ corresponding to ‘c1’. The vector ‘x1’ that you just found is an Eigenvector of A. Now, repeat the same procedure with ‘c2’, ‘c3’ and so on.

**Code for finding EigenVectors in python**

import  numpy as np

#create an array  
arr = np.arange(1,10).reshape(3,3)

#finding the Eigenvalue and Eigenvectors of arr  
np.linalg.eig(arr)

**Code in R for finding Eigenvalues and Eigenvectors:**

|  |
| --- |
| #Calculating eigenvalues and eigenvectors |
|  | A<-matrix(c(30,31,40,41,50,51,60,61,70),nrow = 3,byrow = T) |
|  | e <- eigen(A) |
|  | e$values |
|  | e$vectors |

**Output**

147.737576 5.317459 -3.055035

[,1] [,2] [,3]

[1,] -0.3948374 0.4437557 -0.74478185

[2,] -0.5497457 -0.8199420 -0.06303763

[3,] -0.7361271 0.3616296 0.66432391

### 5.2 Use of Eigenvectors in Data Science

The concept of Eigenvectors is applied in a machine learning algorithm Principal Component Analysis. Suppose you have a data with a large number of features i.e. it has a very high dimensionality. It is possible that there are redundant features in that data. Apart from this, a large number of features will cause reduced efficiency and more disk space. What PCA does is that it craps some of lesser important features. But how to determine those features? Here, Eigenvectors come to our rescue.Let’s go through the algorithm of PCA. Suppose we have an ‘n’ dimensional data and we want to reduce it to ‘k’ dimensions. We will do it in steps.

**Step 1**: Data is mean normalised and feature scaled.

**Step 2:** We find out the covariance matrix of our data set.

Now we want to reduce the number of features i.e. dimensions. But cutting off features means loss of information. We want to minimise the loss of information i.e. we want to keep the maximum variance. So, we want to find out the directions in which variance is maximum. We will find these directions in the next step.

**Step 3:** We find out the Eigenvectors of the covariance matrix. You don’t need to bother much about covariance matrix. It’s an advanced concept of statistics.  As we have data in ‘n’ dimensions, we will find ‘n’ Eigenvectors corresponding to ‘n’ Eigenvalues.

**Step 4**: We will select ‘k’ Eigenvectors corresponding to the ‘k’ largest Eigenvalues and will form a matrix in which each Eigenvector will constitute a column. We will call this matrix as U.

Now it’s the time to find the reduced data points. Suppose you want to reduce a data point ‘a’ in the data set to ‘k’ dimensions.  To do so, you have to just transpose the matrix U and multiply it with the vector ‘a’. You will get the required vector in ‘k’ dimensions.

Once we are done with Eigenvectors, let’s talk about another advanced and highly useful concept in Linear algebra called Singular value decomposition, popularly called as SVD. Its complete understanding needs  a rigorous study of linear algebra.  In fact, SVD is a complete blog in itself. We will come up with another blog completely devoted to SVD. Stay tuned for a better experience. For now, I will just give you a glimpse of how SVD helps in data science.

## 6. Singular Value Decomposition

Suppose you are given a feature matrix A. As suggested by name, what we do is we decompose our matrix A in three constituent matrices for a special purpose.  Sometimes, it is also said that svd is some sort of generalisation of Eigen value decomposition.  I will not go into its mathematics for the reason already explained and will stick to our plan i.e. use of svd in data science.

Svd is used to remove the redundant features in a data set. Suppose you have a data set which comprises of 1000 features. Definitely, any real data set with such a large number of features is bound to contain redundant features. if you have run ML, you should be familiar with the fact that Redundant features cause a lots of problems in running machine learning algorithms. Also, running an algorithm on the original data set will be time inefficient and will require a lot of memory. So, what should you to do handle such a problem? Do we have a choice?  Can we omit some features? Will it lead to significant amount of information loss? Will we be able to get an efficient enough algorithm even after omitting the rows? I will answer these questions with the help of an illustration.

Look at the pictures shown below taken from this [link](http://andrew.gibiansky.com/blog/mathematics/cool-linear-algebra-singular-value-decomposition/images/tiger.jpg)



We can convert this tiger into black and white and can think of it as a matrix whose elements represent the pixel intensity as relevant location. In simpler words, the matrix contains information about the intensity of pixels of the image in the form of rows and columns. But, is it necessary to have all the columns in the intensity matrix? Will we be able to represent the tiger with a lesser amount of information? The next picture will clarify my point. In this picture, different images are shown corresponding to different ranks with different resolution. For now, just assume that higher rank implies the larger amount of information about pixel intensity. The image is taken from this [link](http://andrew.gibiansky.com/blog/mathematics/cool-linear-algebra-singular-value-decomposition/images/tigers.png)

It is clear that we can reach to a pretty well image with 20 or 30 ranks instead of 100 or 200 ranks and that’s what we want to do in a case of highly redundant data. What I want to convey is that to get a reasonable hypothesis, we don’t have to retain all the information present in the original dataset. Even, some of the features cause a problem in reaching a solution to the best algorithm. For the example, presence of redundant features causes multi co-linearity in linear regression. Also, some features are not significant for our model. Omitting these features helps to find a better fit of algorithm along with time efficiency and lesser disk space. Singular value decomposition is used to get rid of the redundant features present in our data.

## 7. End notes

If you have made this far – give yourself a pat at the back. We have covered different aspects of Linear algebra in this article. I have tried to give sufficient amount of information as well as keep the flow such that everybody can understand the concepts and be able to do necessary calculations.