KARPAGAM ACADEMY OF HIGHER EDUCATION

Coimbatore-641 021

(For the candidates admitted from 2016 onwards)

DEPARTMENT OF COMPUTER SCIENCE, CA & IT

SUBJECT NAME : MACHINE LEARNING
SEMESTER : V
SUBJECT CODE: 16CSU503A **CLASS: III B.SC CS**

COURSE OBJECTIVE:

The course aims to provide an introduction to the basic principles, techniques, and applications of Machine Learning.

COURSE OUTCOME:

The objectives to learn the fundamentals of

- Students will have a broad understanding of machine learning algorithms and their use in data-driven knowledge discovery and program synthesis.
- Students will have designed and implemented several machine learning algorithms in Java.
- Students will also be able to identify, formulate and solve machine learning problems that arise in practical applications.
- Students will have a knowledge of the strengths and weaknesses of different machine learning algorithms (relative to the characteristics of the application domain) and be able to adapt or combine some of the key elements of existing machine learning algorithms to design new algorithms as needed.

UNIT-I

Introduction: Concept of Machine Learning, Applications of Machine Learning, Key elements of Machine Learning, Supervised vs. Unsupervised Learning, Statistical Learning: Bayesian Method, The Naive Bayes Classifier

UNIT-II

Softwares for Machine Learning and Linear Algebra Overview : Plotting of Data, Vectorization, Matrices and Vectors: Addition, Multiplication, Transpose and Inverse using available tool such as MATLAB.

UNIT-III

Linear Regression: Prediction using Linear Regression, Gradient Descent, Linear Regression with one variable, Linear Regression with multiple variables, Polynomial Regression, Feature Scaling/Selection.

Logistic Regression: Classification using Logistic Regression, Logistic Regression vs. Linear Regression, Logistic Regression with one variable and with multiple variables.

UNIT-IV

Regularization: Regularization and its utility: The problem of Overfitting, Application of Regularization in Linear and Logistic Regression, Regularization and Bias/Variance.

UNIT-V

Neural Networks: Introduction, Model Representation, Gradient Descent vs. Perceptron Training, Stochastic Gradient Descent, Multilayer Perceptrons, Multiclass Representation, Backpropagation Algorithm.

Suggested Readings

1. Santanu Chattopadhyaya. (2011). Systems Programming. New Delhi: PHI.
2. Alfred, V. Aho., Monica, S. Lam., Ravi Sethi.,& Jeffrey, D. Ullman. (2006). Compilers: Principles, Techniques, and Tools (2nd ed.). New Delhi: Prentice Hall.
3. Dhamdhere, D. M. (2011). Systems Programming. New Delhi: Tata McGraw Hill.
4. Leland Beck., & Manjula, D. (2008). System Software: An Introduction to System Programming (3rd ed.). New Delhi: Pearson Education.
5. Grune, D., Van Reeuwijk, K., Bal, H. E., Jacobs, C. J. H., & Langendoen, K.(2012). Modern Compiler Design (2nd ed.). Springer.

ESE MARKS ALLOCATION

S.No	Category	Marks
1.	**Section A** 20 X1 = 20 (Online Examination)	20
2.	**Section B** 5 X 2 = 10 (Answer all the questions)	10
3.	**Section C** 5 X 6= 30 (Either 'A' or 'B' Choice)	30
4.	**Total**	**60**

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act 1956)
Pollachi Main Road, Eachanari Post, Coimbatore - 641021
(For the candidates admitted from 2016 onwards)
DEPARTMENT OF CS, CA & IT

SUBJECT : **MACHINE LEARNING**

SEMESTER : **V**

SUBJECT CODE: **16CSU503A** **CLASS** : **III B.SC (CS)**

LECTURE PLAN

S.NO	LECTURE DURATION (Hour)	TOPICS TO BE COVERED	SUPPORT MATERIALS
		UNIT I	
1	1	**Introduction**	S_1:1-2
		Concept of Machine Learning	S_4:1-5
2	1	Applications of Machine Learning	S_1:3-11
		Key elements of Machine Learning	W_1
3	1	Supervised vs. Unsupervised Learning	
		Supervised Learning	S_4:3-9, W_1
4	1	Unsupervised Learning	S_4:9-16
5	1	Statistical Learning: Introduction	S_4:27-28
6	1	Bayesian Method	S_3:21-24
7	1	The Naive Bayes Classifier	S_2:154-158
8	1	**Recapitulation of Unit I**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit - I	**8 Hrs**
		UNIT II	
1	1	Software's for Machine Learning and Linear Algebra Overview: Introduction	W_2
2	1	Plotting of Data	W_4
		Vectorization	W_3
3	1	➢ Matrices: Addition	W_4

		➢ Matrices : Multiplication	
4	1	➢ Vectors: Addition	W_4, W_3
		➢ Vectors: Multiplication	
5	1	Transpose and Inverse using available tool such as MATLAB.	W_3
6	1	➢ Transpose using MATLAB	
		➢ Inverse using MATLAB	W_4
7	1	**Recapitulation of Unit II**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit - II	**7 Hrs**
UNIT III			
1	1	Linear Regression: Introduction	S_3:137-147 S_4:217-220
		➢ Prediction using Linear Regression	
2	1	➢ Gradient Descent	
3	1	➢ Linear Regression with one variable	S_1:75,706-2082
		➢ Linear Regression with multiple variables	
4	1	➢ Polynomial Regression	
5	1	➢ Feature Scaling	S_1:106
		➢ Feature Selection	
6	1	Logistic Regression: Introduction	S_4:245-250
7	1	➢ Classification using Logistic Regression	
8	1	➢ Logistic Regression vs. Linear Regression	
9	1	➢ Logistic Regression with one variable	
10	1	➢ Logistic Regression with multiple variables	S_4:252-254
11	1	**Recapitulation of Unit III**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit -III	**11 Hrs**
UNIT IV			
1	1	Regularization: Introduction	S_4:429-430

2	1	➢ Regularization	S_1:77-79
3	1	➢ Regularization utility:	
4	1	➢ The problem of Overfitting,	W_5
5	1	➢ Application of Regularization in Linear Regression	
6	1	➢ Application of Regularization in Logistic Regression	W_6
7	1	➢ Bias	W_7
8	1	➢ Variance	
9	1	➢ Regularization and Bias	S_1:77
10	1	➢ Regularization and Variance	
11	1	**Recapitulation of Unit IV**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit - IV	**11 Hrs**
UNIT V			
1	1	Neural Networks: Introduction,	S_3:225-227
2	1	Model Representation,	S_3:239-241
3	1	➢ Gradient Descent	
4	1	➢ Perceptron Training	S_1:236-237, W_8, W_9
5	1	➢ Gradient Descent Vs Perceptron Training	
6	1	➢ Stochastic Gradient Descent,	
		➢ Multilayer Perceptrons	S_4:563-569
7	1	➢ Multiclass Representation	
8	1	➢ Backpropagation Algorithm.	S_3:241-247 S_4:569-572
9	1	**Recapitulation of Unit V**	
		Discussion of important Questions	
		Total no. of Hours Planned for Unit - V	**9 Hrs**
10	1	**Previous Year ESE Questions Discussion**	
		Previous Year ESE Questions Discussion	
11	1	**Previous Year ESE Questions Discussion**	
		Total no. of Hours Planned for Unit - V	**11 Hrs**

Total Planned Hours for the course	48 HRS

Suggested Reading:

S_1. Ethem Alpaydin. (2009). Introduction to Machine Learning, 2nd Edition, The MIT Press.

S_2. Tom M. Mitchell. (2013). Machine Learning, First Edition,Tata McGraw-Hill Education.

S_3. Christopher M. Bishop. (2007). Pattern Recognition and Machine Learning, Springer.

S_4. Mevin P. Murphy. (2012). Machine Learning: A Probabilistic Perspective, MIT Press.

S_5. SantanuChattopadhyaya. (2011). Systems Programming. New Delhi: PHI.

S_6. Alfred, V. Aho., Monica, S. Lam., RaviSethi.,& Jeffrey, D. Ullman. (2006). Compilers:Principles, Techniques, andTools (2nd ed.). New Delhi: Prentice Hall.

S_7. Dhamdhere, D. M. (2011). Systems Programming. New Delhi: Tata McGraw Hill.

S_8. Leland Beck., &Manjula, D. (2008). System Software: An Introduction to System Programming(3rded.). New Delhi: Pearson Education.

S_9. Grune, D., Van Reeuwijk, K., Bal, H. E., Jacobs, C. J. H.,&Langendoen, K.(2012). ModernCompiler Design (2nd ed.). Springer.

Websites

W_1: http://www.grresforgresks.org/ml-machinelearning/
W_2:https://machinelearningmastery.com/linera-algebra-machine-learning/
W_3:https://cedar.buffalo.edu/~srihar/CSE574/chap1/linearAlgebra.pdf
W_4:www.tutorialspoint.com/matlab/matlab-plotting.html
W_5:https://towardsdatascience.com
W_6: https://www.coursesa.org
W_7: www.cs.cmu.edu./~wcohen.bias-variance
W_8:https://medium.com/machinelearning..
W_9: https://mslab.csie.asia.edu.tw/~ktduc

UNIT-I

Introduction: Concept of Machine Learning, Applications of Machine Learning, Key elements of Machine Learning, Supervised vs. Unsupervised Learning, Statistical Learning: Bayesian Method, The Naive Bayes Classifier

Introduction to Machine Learning

DATA :

It can be any unprocessed fact, value, text, sound or picture that is not being interpreted and analyzed.

Data is the most important part of all Data Analytics, Machine Learning, and Artificial Intelligence.

Without data, we can't train any model and all modern research and automation will go unsuccessful.

Big Enterprises are spending loads of money just to gather as much certain data as possible.

Example: Why did Facebook acquire WhatsApp by paying a huge price of $19 billion?

The answer is very simple and logical – it is to have access to the users' information that Facebook may not have but WhatsApp will have. This information of their users is of paramount importance to Facebook as it will facilitate the task of improvement in their services.

INFORMATION:

Data that has been interpreted and manipulated and has now some meaningful inference for the users.

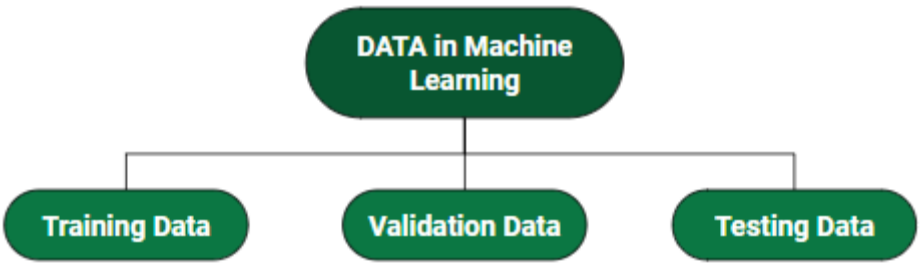
KNOWLEDGE:

Combination of inferred information, experiences, learning and insights.

Results in awareness or concept building for an individual or organization.

How we split data in Machine Learning?

- **Training Data:** The part of data we use to train our model. This is the data which your model actually sees (both input and output) and learn from.
- **Validation Data:** The part of data which is used to do a frequent evaluation of model, fit on training dataset along with improving involved hyper parameters (initially set parameters before the model begins learning). This data plays its part when the model is actually training.
- **Testing Data:** Once our model is completely trained, testing data provides the unbiased evaluation. When we feed in the inputs of testing data, our model will predict some values (without seeing actual output). After prediction, we evaluate our model by comparing it with actual output present in the testing data. This is how we evaluate and see how much our model has learned from the experiences feed in as training data, set at the time of training.



Consider an example:

There's a Shopping Mart Owner who conducted a survey for which he has a long list of questions and answers that he had asked from the customers, this list of questions and answers is **DATA**. Now every time when he want to infer anything and can't just go through each and every question of thousands of customers to find something relevant as it would be time-consuming and not helpful. In order to reduce this overhead and time wastage and to make work easier, data is manipulated through software, calculations, graphs etc. as per own convenience, this inference from manipulated data is **Information**. So, Data is must for Information. Now **Knowledge** has its role in differentiating between two individuals having same information. Knowledge is actually not a technical content but is linked to human thought process.

Properties of Data –

1. **Volume:** Scale of Data. With growing world population and technology at exposure, huge data is being generated each and every millisecond.
2. **Variety:** Different forms of data – healthcare, images, videos, audio clippings.
3. **Velocity:** Rate of data streaming and generation.
4. **Value:** Meaningfulness of data in terms of information which researchers can infer from it.
5. **Veracity:** Certainty and correctness in data we are working on.

What is Machine Learning?

Machine Learning is getting computers to program themselves. If programming is automation, then machine learning is automating the process of automation.

Arthur Samuel, a pioneer in the field of artificial intelligence and computer gaming, coined the term **"Machine Learning"**. He defined machine learning as – **"Field of study that gives computers the capability to learn without being explicitly programmed.**

A breakthrough in machine learning would be worth ten Microsofts.

— Bill Gates, Former Chairman, Microsoft

- Traditional Programming: Data and program is run on the computer to produce the output.
- Machine Learning: Data and output is run on the computer to create a program. This program can be used in traditional programming.

Machine learning is like farming or gardening. Seeds is the algorithms, nutrients is the data, the gardner is you and plants is the programs.

"A computer program is said to learn from experience E with some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." -Tom M. Mitchell

Examples of Machine Learning

There are many examples of machine learning. Here are a few examples of classification problems where the goal is to categorize objects into a fixed set of categories.

Face detection: Identify faces in images (or indicate if a face is present).

Email filtering: Classify emails into spam and not-spam.

Medical diagnosis: Diagnose a patient as a sufferer or non-sufferer of some disease.

Weather prediction: Predict, for instance, whether or not it will rain tomorrow.

Applications of Machine Learning

Sample applications of machine learning:

- **Web search**: ranking page based on what you are most likely to click on.
- **Computational** biology: rational design drugs in the computer based on past experiments.
- **Finance**: decide who to send what credit card offers to. Evaluation of risk on credit offers. How to decide where to invest money.
- **E-commerce**: Predicting customer churn. Whether or not a transaction is fraudulent.
- **Space exploration**: space probes and radio astronomy.

- **Robotics**: how to handle uncertainty in new environments. Autonomous. Self-driving car.
- **Information extraction**: Ask questions over databases across the web.
- **Social networks**: Data on relationships and preferences. Machine learning to extract value from data.
- **Debugging**: Use in computer science problems like debugging. Labor intensive process. Could suggest where the bug could be.

Key elements of machine learning

There are tens of thousands of machine learning algorithms and hundreds of new algorithms are developed every year.

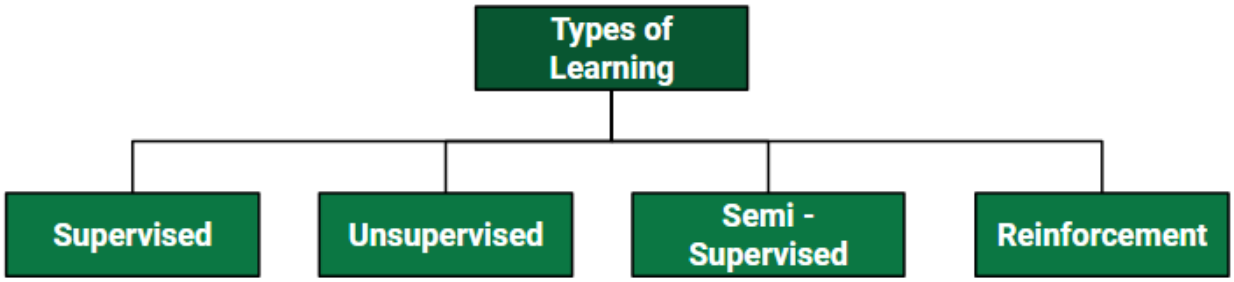
Every machine learning algorithm has three components:

- **Representation**: how to represent knowledge. Examples include decision trees, sets of rules, instances, graphical models, neural networks, support vector machines, model ensembles and others.
- **Evaluation**: the way to evaluate candidate programs (hypotheses). Examples include accuracy, prediction and recall, squared error, likelihood, posterior probability, cost, margin, entropy k-L divergence and others.
- **Optimization**: the way candidate programs are generated known as the search process. For example combinatorial optimization, convex optimization, constrained optimization.

All machine learning algorithms are combinations of these three components. A framework for understanding all algorithms.

Types of Learning

There are four types of machine learning:



- **Supervised learning**: (also called inductive learning) Training data includes desired outputs. This is spam this is not, learning is supervised.
- **Unsupervised learning**: Training data does not include desired outputs. Example is clustering. It is hard to tell what is good learning and what is not.
- **Semi-supervised learning**: Training data includes a few desired outputs.
- **Reinforcement learning**: Rewards from a sequence of actions. AI types like it, it is the most ambitious type of learning.

Supervised learning is the most mature, the most studied and the type of learning used by most machine learning algorithms. Learning with supervision is much easier than learning without supervision.

Inductive Learning is where we are given examples of a function in the form of data (x) and the output of the function ($f(x)$). The goal of inductive learning is to learn the function for new data (x).

- **Classification**: when the function being learned is discrete.

- **Regression**: when the function being learned is continuous.
- **Probability Estimation**: when the output of the function is a probability.

Supervised Learning :

Supervised learning is when the model is getting trained on a labelled dataset. **Labelled** dataset is one which have both input and output parameters. In this type of learning both training and validation datasets are labelled as shown in the figures below.

User ID	Gender	Age	Salary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	1
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	1
15728773	Male	27	58000	1
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	1
15727311	Female	35	65000	0
15570769	Female	26	80000	1
15606274	Female	26	52000	0
15746139	Male	20	86000	1
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1

Figure A: CLASSIFICATION

Temperature	Pressure	Relative Humidity	Wind Direction	Wind Speed
10.69261758	986.882019	54.19337313	195.7150879	3.278597116
13.59184184	987.8729248	48.0648859	189.2951202	2.909167767
17.70494885	988.1119385	39.11965597	192.9273834	2.973036289
20.95430404	987.8500366	30.66273218	202.0752869	2.965289593
22.9278274	987.2833862	26.06723423	210.6589203	2.798230886
24.04233986	986.2907104	23.46918024	221.1188507	2.627005816
24.41475295	985.2338867	22.25082295	233.7911987	2.448749781
23.93361956	984.8914795	22.35178837	244.3504333	2.454271793
22.68800023	984.8461304	23.7538641	253.0864716	2.418341875
20.56425726	984.8380737	27.07867944	264.5071106	2.318677425
17.76400389	985.4262085	33.54900114	280.7827454	2.343950987
11.25680746	988.9386597	53.74139903	68.15406036	1.650191426
14.37810685	989.6819458	40.70884681	72.62069702	1.553469896
18.45114201	990.2960205	30.85038484	71.70604706	1.005017161
22.54895853	989.9562988	22.81738811	44.66042709	0.264133632
24.23155922	988.796875	19.74790765	318.3214111	0.329656571

Figure B: REGRESSION

Both the above figures have labeled data set –

- **Figure A:** It is a dataset of a shopping store which is useful in predicting whether a customer will purchase a particular product under consideration or not based on his/ her gender, age and salary.
  Input : Gender, Age, Salary

  Ouput : Purchased i.e. 0 or 1 ; 1 means yes the customer will purchase and 0 means that customer won't purchase it.

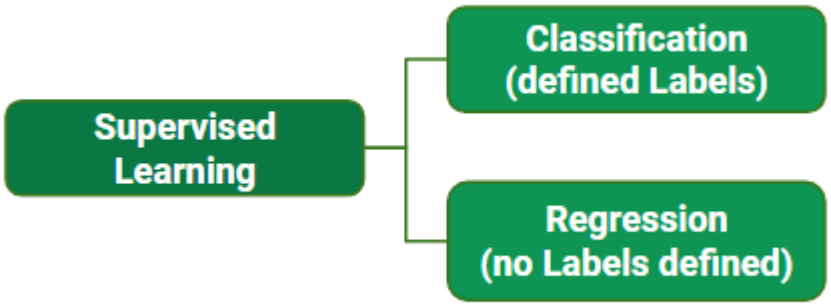
- **Figure B:** It is a Meteorological dataset which serves the purpose of predicting wind speed based on different parameters.

  Input : Dew Point, Temperature, Pressure, Relative Humidity, Wind Direction
  Output : Wind Speed

Training the system:

While training the model, data is usually split in the ratio of 80:20 i.e. 80% as training data and rest as testing data. In training data, we feed input as well as output for 80% data. The model learns from training data only. We use different machine learning algorithms (which we will discuss in detail in next articles) to build our model. By learning, it means that the model will build some logic of its own.

Once the model is ready then it is good to be tested. At the time of testing, input is fed from remaining 20% data which the model has never seen before, the model will predict some value and we will compare it with actual output and calculate the accuracy.



Types of Supervised Learning:

1. **Classification:** It is a Supervised Learning task where output is having defined labels (discrete value). For example in above Figure A, Output – Purchased has defined labels i.e. 0 or 1 ; 1 means the customer will purchase and 0 means that customer won't purchase. The goal here is to predict discrete values belonging to a particular class and evaluate on the basis of accuracy.

   It can be either binary or multi class classification. In **binary** classification, model predicts either 0 or 1 ; yes or no but in case of **multi class** classification, model predicts more than one class. **Example:** Gmail classifies mails in more than one classes like social, promotions, updates, forum.

2. **Regression :** It is a Supervised Learning task where output is having continuous value.

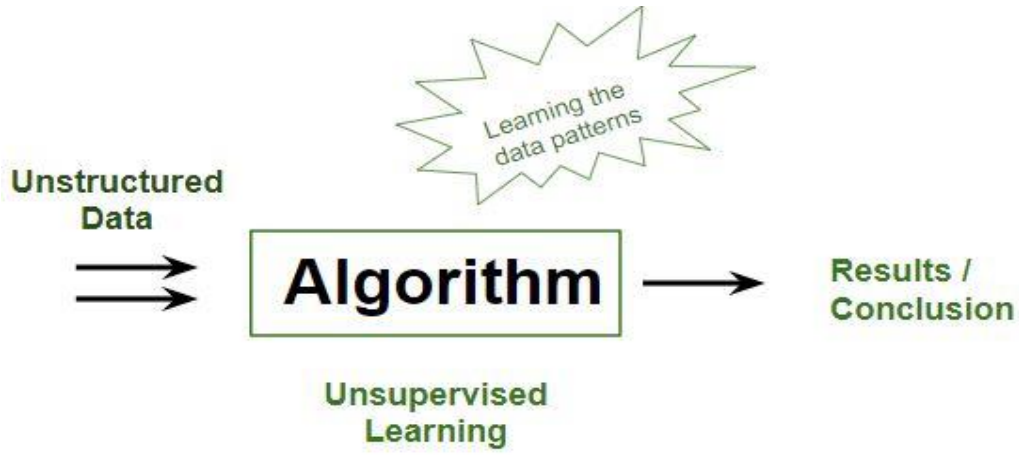
   Example in above Figure B, Output – Wind Speed is not having any discrete value but is continuous in the particular range. The goal here is to predict a value as much closer to actual output value as our model can and

then evaluation is done by calculating error value. The smaller the error the greater the accuracy of our regression model.

Example of Supervised Learning Algorithms:

- Linear Regression
- Nearest Neighbor
- Guassian Naive Bayes
- Decision Trees
- Support Vector Machine (SVM)
- Random Forest

Unsupervised Learning:



It's a type of learning where we don't give target to our model while training i.e. training model has only input parameter values. The model by itself has to find which way it can learn.

Data-set in Figure A is mall data that contains information of its clients that subscribe to them.

Once subscribed they are provided a membership card and so the mall has complete information about customer and his/her every purchase.

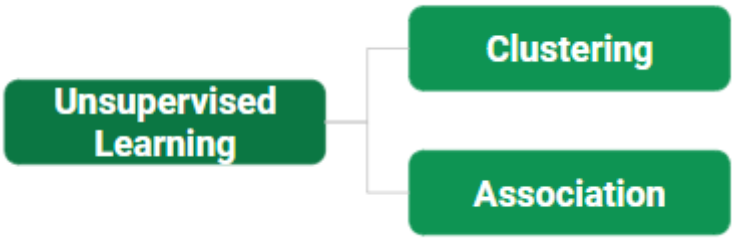
Now using this data and unsupervised learning techniques, mall can easily group clients based on the parameters we are feeding in.

CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
1	Male	19	15	39
2	Male	21	15	81
3	Female	20	16	6
4	Female	23	16	77
5	Female	31	17	40
6	Female	22	17	76
7	Female	35	18	6
8	Female	23	18	94
9	Male	64	19	3
10	Female	30	19	72
11	Male	67	19	14
12	Female	35	19	99
13	Female	58	20	15
14	Female	24	20	77
15	Male	37	20	13
16	Male	22	20	79
17	Female	35	21	35

Figure A

Training data we are feeding is –

- **Unstructured data:** May contain noisy(meaningless) data, missing values or unknown data
- **Unlabeled data:** Data only contains value for input parameters, there is no targeted value (output). It is easy to collect as compared to labeled one in supervised approach.



Types of Unsupervised Learning:

- **Clustering:** Broadly this technique is applied to group data based on different patterns, our machine model finds. For example in above figure we are not given output parameter value, so this technique will be used to group clients based on the input parameters provided by our data.
- **Association:** This technique is a rule based ML technique which finds out some very useful relations between parameters of a large data set. For e.g. shopping stores use algorithms based on this technique to find out relationship between sale of one product w.r.t to others sale based on customer behavior. Once trained well, such models can be used to increase their sales by planning different offers.

Some algorithms:

- K-Means Clustering
- DBSCAN – Density-Based Spatial Clustering of Applications with Noise
- BIRCH – Balanced Iterative Reducing and Clustering using Hierarchies
- Hierarchical Clustering

Semi-supervised Learning

Its working lies between Supervised and Unsupervised techniques. We use these techniques when we are dealing with a data which is a little bit labeled and rest large portion of it is unlabeled.

We can use unsupervised technique to predict labels and then feed these labels to supervised techniques.

This technique is mostly applicable in case of image data-sets where usually all images are not labeled.

Reinforcement Learning:

In this technique, model keeps on increasing its performance using a Reward Feedback to learn the behavior or pattern. These algorithms are specific to a particular problem e.g. Google Self Driving car, AlphaGo where a bot competes with human and even itself to getting better and better

performer of Go Game. Each time we feed in data, they learn and add the data to its knowledge that is training data. So, more it learns the better it get trained and hence experienced.

- Agents observe input.
- Agent performs an action by making some decisions.
- After its performance, agent receives reward and accordingly reinforce and the model stores in state-action pair of information.

Some algorithms:

- Temporal Difference (TD)
- Q-Learning
- Deep Adversarial Networks

Differences between Supervised Learning and Unsupervised Learning

(http://www.differencebetween.net/technology/differences-between-supervised-learning-and-unsupervised-learning/)

1. Input Data in Supervised Learning and Unsupervised Learning

The primary difference between supervised learning and unsupervised learning is the data used in either method of machine learning. It is worth noting that both methods of machine learning require data, which they will analyze to produce certain functions or data groups. However, the input

data used in supervised learning is well known and is labeled. This means that the machine is only tasked with the role of determining the hidden patterns from already labeled data. However, the data used in unsupervised learning is not known nor labeled. It is the work of the machine to categorize and label the raw data before determining the hidden patterns and functions of the input data.

2. Computational Complexity in Supervised Learning and Unsupervised Learning

Machine learning is a complex affair and any person involved must be prepared for the task ahead. One of the stand out differences between supervised learning and unsupervised learning is computational complexity. Supervised learning is said to be a complex method of learning while unsupervised method of learning is less complex. One of the reason that makes supervised learning affair is the fact that one has to understand and label the inputs while in unsupervised learning, one is not required to understand and label the inputs. This explains why many people have been preferring unsupervised learning as compared to the supervised method of machine learning.

3. Accuracy of the Results of Supervised Learning and Unsupervised Learning

The other prevailing difference between supervised learning and unsupervised learning is the accuracy of the results produced after every cycle of machine analysis. All the results generated from supervised method of machine learning are more accurate and reliable as compared to

the results generated from the unsupervised method of machine learning. One of the factor that explains why supervised method of machine learning produces accurate and reliable results is because the input data is well known and labeled which means that the machine will only analyze the hidden patterns. This is unlike unsupervised method of learning where the machine has to define and label the input data before determining the hidden patterns and functions.

4. Number of Classes in Supervised Learning and Unsupervised Learning

It is also worth noting that there is a significant difference when it comes to the number of classes. It is worth noting that all the classes used in supervised learning are known which means that also the answers in the analysis are likely to be known. The only goal of supervised learning is therefore to determine the unknown cluster. However, there is no prior knowledge in unsupervised method of machine learning. In addition, the numbers of classes are not known which clearly means that no information is known and the results generated after the analysis cannot be ascertained. Moreover, the people involved in unsupervised method of learning are not aware of any information concerning the raw data and the expected results.

5. Real Time Learning in Supervised Learning and Unsupervised Learning

Among other differences, there exist the time after which each method of learning takes place. It is important to highlight that supervised method of

learning takes place off-line while unsupervised method of learning takes place in real time. People involved in preparation and labeling of the input data do so off-line while the analysis of the hidden pattern is done online which denies the people involved in machine learning an opportunity to interact with the machine as it analyzes the discrete data. However, unsupervised method of machine learning takes place in real time such that all the input data is analyzed and labeled in the presence of learners which helps them to understand different methods of learning and classification of raw data. Real time data analysis remains to be the most significant merit of unsupervised method of learning.

TABLE SHOWING DIFFERENCES BETWEEN SUPERVISED LEARNING AND UNSUPERVISED LEARNING: COMPARISON CHART

	Supervised Learning	**Unsupervised Learning**
Input Data	Uses Known and Labeled Input Data	Uses Unknown Input Data
Computational Complexity	Very Complex in Computation	Less Computational Complexity
Real Time	Uses off-line analysis	Uses Real Time Analysis of Data
Number of Classes	Number of Classes is Known	Number of Classes is not Known

Accuracy of Results	Accurate and Reliable Results	Moderate Accurate and Reliable Results

Summary of Supervised Learning and Unsupervised Learning

- Data mining is becoming an essential aspect in the current business world due to increased raw data that organizations need to analyze and process so that they can make sound and reliable decisions.
- This explains why the need for machine learning is growing and thus requiring people with sufficient knowledge of both supervised machine learning and unsupervised machine learning.
- It is worth understanding that each method of learning offers its own advantages and disadvantages. This means that one has to be conversant with both methods of machine learning before determine which method one will use to analyze data.

Definition of Bayesian: Being, relating to, or involving statistical methods that assign probabilities or distributions to events (such as rain tomorrow) or parameters (such as a population mean) based on experience or best guesses before experimentation and data collection and that apply Bayes' theorem to revise the probabilities and distributions after obtaining experimental data

Bayes' Theorem

In machine learning we are often interested in selecting the best hypothesis (h) given data (d).

In a classification problem, our hypothesis (h) may be the class to assign for a new data instance (d).

One of the easiest ways of selecting the most probable hypothesis given the data that we have that we can use as our prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our prior knowledge.

Bayes' Theorem is stated as:

$P(h|d) = (P(d|h) * P(h)) / P(d)$

Where

- **P(h|d)** is the probability of hypothesis h given the data d. This is called the posterior probability.
- **P(d|h)** is the probability of data d given that the hypothesis h was true.
- **P(h)** is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.
- **P(d)** is the probability of the data (regardless of the hypothesis).

You can see that we are interested in calculating the posterior probability of P(h|d) from the prior probability p(h) with P(D) and P(d|h).

After calculating the posterior probability for a number of different hypotheses, you can select the hypothesis with the highest probability.

This is the maximum probable hypothesis and may formally be called the maximum a posteriori (MAP) hypothesis.

This can be written as:

MAP(h) = max(P(h|d))

or

MAP(h) = max((P(d|h) * P(h)) / P(d))

or

MAP(h) = max(P(d|h) * P(h))

The P(d) is a normalizing term which allows us to calculate the probability. We can drop it when we are interested in the most probable hypothesis as it is constant and only used to normalize.

Back to classification, if we have an even number of instances in each class in our training data, then the probability of each class (e.g. P(h)) will be equal. Again, this would be a constant term in our equation and we could drop it so that we end up with:

MAP(h) = max(P(d|h))

This is a useful exercise, because when reading up further on Naive Bayes you may see all of these forms of the theorem.

Naive Bayes Classifier

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values.

It is called *naive Bayes* or *idiot Bayes* because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P(d1, d2, d3|h), they are assumed to be conditionally independent given the target value and calculated as P(d1|h) * P(d2|H) and so on.

This is a very strong assumption that is most unlikely in real data, i.e. that the attributes do not interact. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold.

Representation Used By Naive Bayes Models

The representation for naive Bayes is probabilities.

A list of probabilities are stored to file for a learned naive Bayes model. This includes:

- **Class Probabilities**: The probabilities of each class in the training dataset.
- **Conditional Probabilities**: The conditional probabilities of each input value given each class value.

Learn a Naive Bayes Model From Data

Learning a naive Bayes model from your training data is fast.

Training is fast because only the probability of each class and the probability of each class given different input (x) values need to be calculated. No coefficients need to be fitted by optimization procedures.

Calculating Class Probabilities

The class probabilities are simply the frequency of instances that belong to each class divided by the total number of instances.

For example in a binary classification the probability of an instance belonging to class 1 would be calculated as:

P(class=1) = count(class=1) / (count(class=0) + count(class=1))

In the simplest case each class would have the probability of 0.5 or 50% for a binary classification problem with the same number of instances in each class.

Calculating Conditional Probabilities

The conditional probabilities are the frequency of each attribute value for a given class value divided by the frequency of instances with that class value.

For example, if a "*weather*" attribute had the values "*sunny*" and "*rainy*" and the class attribute had the class values "*go-out*" and "*stay-home*", then the conditional probabilities of each weather value for each class value could be calculated as:

- P(weather=sunny|class=go-out) = count(instances with weather=sunny and class=go-out) / count(instances with class=go-out)
- P(weather=sunny|class=stay-home) = count(instances with weather=sunny and class=stay-home) / count(instances with class=stay-home)
- P(weather=rainy|class=go-out) = count(instances with weather=rainy and class=go-out) / count(instances with class=go-out)
- P(weather=rainy|class=stay-home) = count(instances with weather=rainy and class=stay-home) / count(instances with class=stay-home)

Make Predictions With a Naive Bayes Model

Given a naive Bayes model, you can make predictions for new data using Bayes theorem.

MAP(h) = max(P(d|h) * P(h))

Using our example above, if we had a new instance with the *weather* of *sunny*, we can calculate:

go-out = P(weather=sunny|class=go-out) * P(class=go-out)
stay-home = P(weather=sunny|class=stay-home) * P(class=stay-home)

We can choose the class that has the largest calculated value. We can turn these values into probabilities by normalizing them as follows:

P(go-out|weather=sunny) = go-out / (go-out + stay-home)
P(stay-home|weather=sunny) = stay-home / (go-out + stay-home)

If we had more input variables we could extend the above example. For example, pretend we have a "*car*" attribute with the values "*working*" and "*broken*". We can multiply this probability into the equation.

For example below is the calculation for the "go-out" class label with the addition of the car input variable set to "working":

go-out = P(weather=sunny|class=go-out) * P(car=working|class=go-out) * P(class=go-out)

POSSIBLE QUESTIONS

Part B (2 Marks)

1. What is the difference between supervised and unsupervised machine learning?
2. What is inductive machine learning?
3. What is the difference between Machine learning and Data Mining?
4. What are the different types of algorithm techniques are available in machine learning?
5. What is meant by Reinforcement?
6. Define Clustering.
7. What are the three stages to build the model in machine learning?
8. List out few properties of Data.

Part C (6 Marks)

1. Explain about Types of Learning.
2. List out the following i) Data ii) Information iii) Knowledge iv) Volume
3. Explain about Bayes Theorem.
4. Describe about key elements of machine learning with example.
5. Illustrate on applications on Machine Learning.
6. Differentiate between supervised and unsupervised machine learning.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of CS,CA & IT

III B.Sc(CS) (BATCH 2016-2019)

Machine Learning

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

ONLINE EXAMINATIONS **ONE MARK QUESTIONS**

S.NO	QUESTIONS	OPTION1	OPTION2	OPTION3	OPTION4	ANSWER
1	_____is a field of computer science that deals with system programming to learn and improve with experience.	Artifical Intelligence	Machine learning	Data Mining	Model Selection.	Machine learning
2	The process of choosing models among diverse mathematical models, which are used to define the same data set is known as _____	Model Selection.	Data Mining	Artifical Intelligence	Machine learning	Model Selection.

3	Machine learning is_____	The autonomous acquisition of knowledge through the use of computer programs	The autonomous acquisition of knowledge through the use of manual programs	The selective acquisition of knowledge through the use of computer programs	The selective acquisition of knowledge through the use of manual programs	The autonomous acquisition of knowledge through the use of computer programs
4	Who provided a formal definition of machine learning?	Tom A. Mitchell	Tom M. Mitchell	Tom K. Mitchell	Tom J. Mitchell	Tom M. Mitchell
5	Machine learning is divided into how many categories?	2	3	4	5	2
6	Inputs are divided into how many in Classifications?	2	3	4	5	2
7	What happens in clustering?	An exponential of the inputs is found	Inputs are multiplied	Inputs are divided into groups	Sets are multiplied	Inputs are divided into groups

8	Computers are best at learning	facts	concepts	procedures	principles	facts
9	Data used to build a data mining model.	validation data	training data	test data	hidden data	training data
10	Supervised learning and unsupervised clustering both require at least one	hidden attribute.	output attribute	input attribute	categorical attribute.	hidden attribute.
11	Supervised learning differs from unsupervised clustering in that supervised learning requires	at least one input attribute.	input attributes to be categorical	at least one output attribute	ouput attriubutes to be categorical	input attributes to be categorical
12	Data used to optimize the parameter settings of a supervised learner model.	training	test	verification	validation	validation

13	The average squared difference between classifier predicted output and actual output.	mean squared error	root mean squared error	mean absolute error	mean relative error	mean squared error
14	The process of forming general concept definitions from examples of concepts to be learned.	deduction	abduction	induction	conjunction	induction
15	Data mining is best described as the process of	identifying patterns in data.	deducing relationships in data.	representing data.	simulating trends in data.	identifying patterns in data.
16	Computers are best at learning	facts.	concepts.	procedures.	principles.	concepts.
17	Like the probabilistic view, the _____ view allows us to associate a probability of membership with each classification.	exemplar	deductive	classical	inductive	exemplar

18	Data used to build a data mining model.	validation data	training data	test data	hidden data	training data
19	Supervised learning and unsupervised clustering both require at least one	hidden attribute.	output attribute.	input attribute.	categorical attribute.	input attribute.
20	Supervised learning differs from unsupervised clustering in that supervised learning requires	at least one input attribute.	input attributes to be categorical.	at least one output attribute.	ouput attriubutes to be categorical.	at least one output attribute.
21	Database query is used to uncover this type of knowledge.	deep	hidden	shallow	multidimensional	shallow
22	A statement to be tested.	theory	procedure	principle	hypothesis	hypothesis

23	A person trained to interact with a human expert in order to capture their knowledge.	knowledge programmer	knowledge developer	knowledge engineer	knowledge extractor	knowledge engineer
24	Which of the following is not a characteristic of a data warehouse?	contains historical data	designed for decision support	stores data in normalized tables	promotes data redundancy	stores data in normalized tables
25	A structure designed to store data for decision support.	operational database	flat file	decision tree	data warehouse	data warehouse
26	A nearest neighbor approach is best used	with large-sized datasets.	when irrelevant attributes have been removed from the data.	when a generalized model of the data is desireable.	when an explanation of what has been found is of primary importance.	when irrelevant attributes have been removed from the data.
27	If a customer is spending more than expected, the customer's intrinsic value is _____ their actual value.	greater than	less than	less than or equal to	equal to	less than

28	_____ can be any unprocessed fact, value, text, sound or picture that is not being interpreted and analyze	data	knowledge	information	machine	data
29	____ has been interpreted and manipulated and has now some meaningful inferences for the users.	data	knowledge	information	machine	data
30	_____ is the combination of inferred information and learning.	data	knowledge	information	machine	knowledge
31	_____ is the part of data we use to train our model.	training data	testing data	validation data	knowledge	training data
32	How we split data in Machine Learning?	3	5	6	8	3

33	_____ means scale of data	volume	value	velocity	veracity	volume
34	Which defined correctness in data?	volume	value	velocity	veracity	veracity
35	Supervised Learning is also called as _____.	Inductive Learning	semi-supervised	regression	labeled	Inductive Learning
36	Which dataset is one which has both input and output parameters.	labeled	unlabelled	function	model	labeled
37	What is another name for meaningless data?	unstructured data	structured data	labeled data	value	unstructured data

38	The data which contains only an input parameters.	unstructured data	structured data	unlabeled data	value	unlabeled data
39	_____ is a classification algorithm for binary and multi class classification problems.	naïve bayes	bayes stephen	bias	flemming bayes	naïve bayes
40	_____is a sub-field of mathematics concerned with vectors, matrices, and linear transforms.	linear algebra	linear graphs	linear arrays	linear matrix	linear algebra
41	_____ is a method of teaching and learning in a logical manner.	Machine learning	PAC Learning	Artifical Intelligence	Sequence learning	Sequence learning
42	_____is about identifying group membership while regression technique involves predicting a response	Classification	Association	Regression models	Clustering	Classification

43	Which training data includes a few desired outputs?	Inductive Learning	semi-supervised	regression	labeled	semi-supervised
44	The way candidate programs are generated known as the _____ process.	search	hypotheses	evaluation	knowledge	search
45	Which is called as idiot?	navie	navie bayes	bias	flemming bayes	navie bayes
46	The field of study that gives computers the capability to learn without being explicitly programmed is known as _____.	machine learning	data learning	testing learning	type learning.	machine learning
47	_____ defined labels.	classification	regression	supervised learning	data warehouse	classification

48	Predictive models having target attribute having discrete values can be termed as _____	Regression models	Classification models	supervised learning	data warehouse	Classification models
49	When was the name coined?	1987	1959	1978	1990	1959
50	_____is based on an assumption that all of the features in the data set are important, equal and independent.	navie	navie bayes	bias	flemming bayes	navie bayes
51	_____ is a process or a study whether it closely relates to design, development of the algorithms that provide an ability to the machines to capacity to learn.	Model Selection.	Data Mining	Artifical Intelligence	Machine learning	Machine learning
52	_____technique is a rule based ML technique which finds out some very useful relations between parameters of a large data set.	Classification models	Association	Regression models	data warehouse	Association

53	_____ technique is mostly applicable in case of image data-sets where usually all images are not labeled.	Inductive Learning	semi-supervised	regression	labeled	semi-supervised
54	_____model keeps on increasing its performance using a Reward Feedback to learn the behavior or pattern.	supervised	semi-supervised	reinforcement	unsuervised	reinforcement
55	Which is example of supervised learning algorithm?	K-Means Clustering	Decision Trees	Temporal Difference (TD)	Q-Learning	Decision Trees
56	With Bayes classifier, missing data items are	treated as equal compares.	treated as unequal compares.	replaced with a default value.	ignored.	treated as unequal compares.
57	This unsupervised clustering algorithm terminates when mean values computed for the current iteration of the algorithm are identical to the computed mean values for the previous iteration.	agglomerative clustering	conceptual clustering	K-Means clustering	expectation maximization	K-Means clustering

58	Machine learning techniques differ from statistical techniques in that machine learning methods	typically assume an underlying distribution for the data.	are better able to deal with missing and noisy data.	are not able to explain their behavior.	have trouble with large-sized datasets.	are better able to deal with missing and noisy data.
59	The_____ involves the process of learning by examples, where a system, from a set of observed instances tries to induce a general rule.	semi-supervised	regression	Inductive machine learning	Artificial Intelligence	Inductive machine learning
60	_____ techniques allow learning a function or predictor from a set of observed data that can make predictions about unseen or future data.	Statistical learning techniques allow learning a function or predictor from a set of observed	Artificial Intelligence	inductive machine learning	machine learning	Statistical learning techniques allow learning a function or predictor from a set of observed data that can make predictions about unseen or

UNIT-II

Softwares for Machine Learning and Linear Algebra Overview: Plotting of Data, Vectorization, Matrices and Vectors: Addition, Multiplication, Transpose and Inverse using available tool such as MATLAB.

Software's for Machine learning and Linear Algebra Overview

Plotting of Data

To plot the graph of a function, you need to take the following steps −

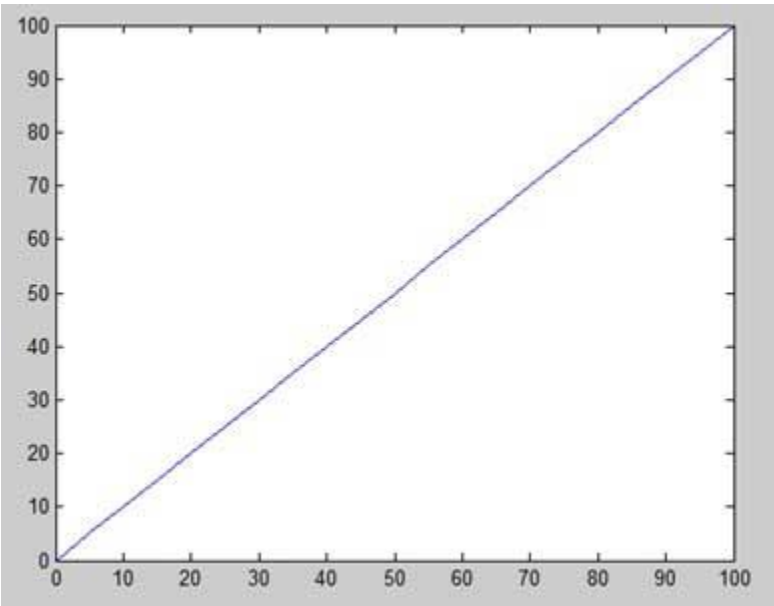
- Define x, by specifying the range of values for the variable x, for which the function is to be plotted
- Define the function, y = f(x)
- Call the plot command, as plot(x, y)

Following example would demonstrate the concept. Let us plot the simple function y = x for the range of values for x from 0 to 100, with an increment of 5.

Create a script file and type the following code −

```
x = [0:5:100];

y = x;

plot(x, y)
```

MATLAB displays the following plot −

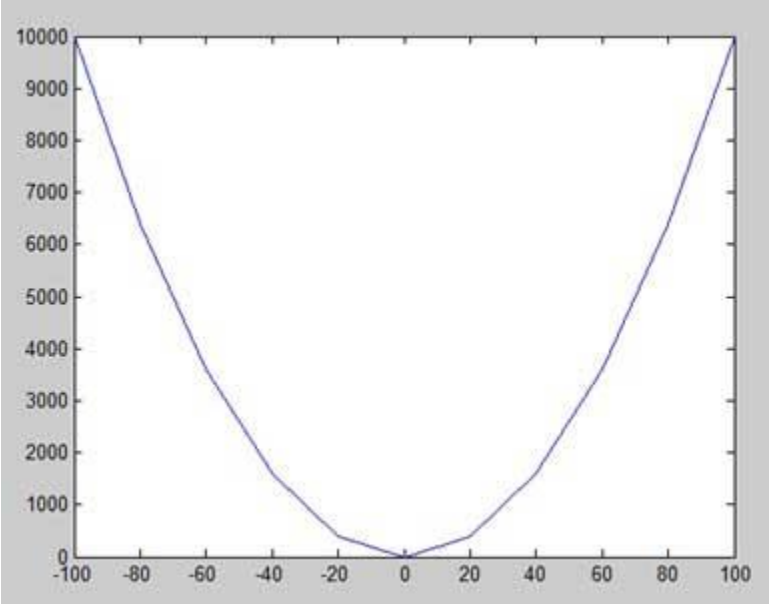


Let us take one more example to plot the function $y = x^2$. In this example, we will draw two graphs with the same function, but in second time, we will reduce the value of increment. Please note that as we decrease the increment, the graph becomes smoother.

Create a script file and type the following code −

```
x = [1 2 3 4 5 6 7 8 9 10];

x = [-100:20:100];

y = x.^2;

plot(x, y)
```

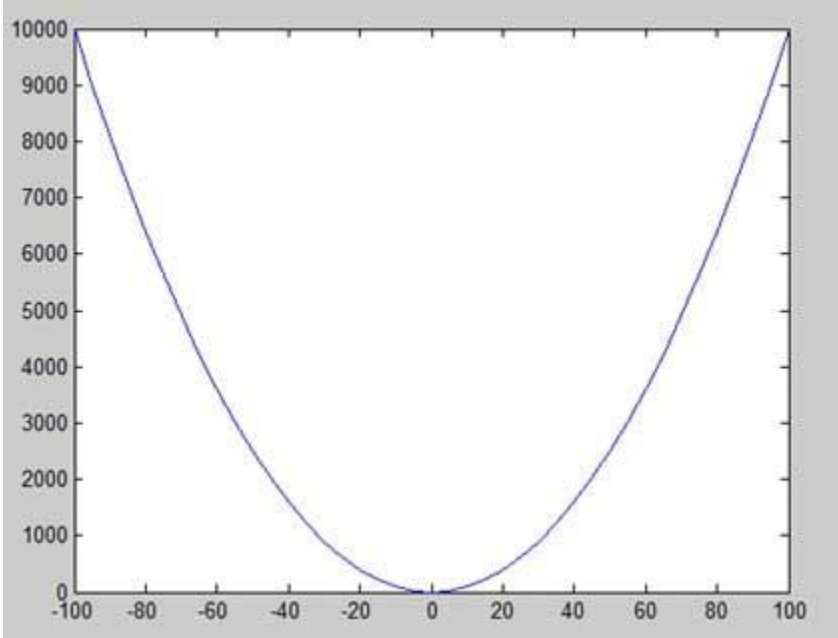
MATLAB displays the following plot −



Change the code file a little, reduce the increment to 5 −

```
x = [-100:5:100];

y = x.^2;

plot(x, y)
```

MATLAB draws a smoother graph −



Adding Title, Labels, Grid Lines and Scaling on the Graph

MATLAB allows you to add title, labels along the x-axis and y-axis, grid lines and also to adjust the axes to spruce up the graph.

- The xlabel and ylabel commands generate labels along x-axis and y-axis.
- The title command allows you to put a title on the graph.
- The grid on command allows you to put the grid lines on the graph.
- The axis equal command allows generating the plot with the same scale factors and the spaces on both axes.
- The axis square command generates a square plot.

Example

Create a script file and type the following code −

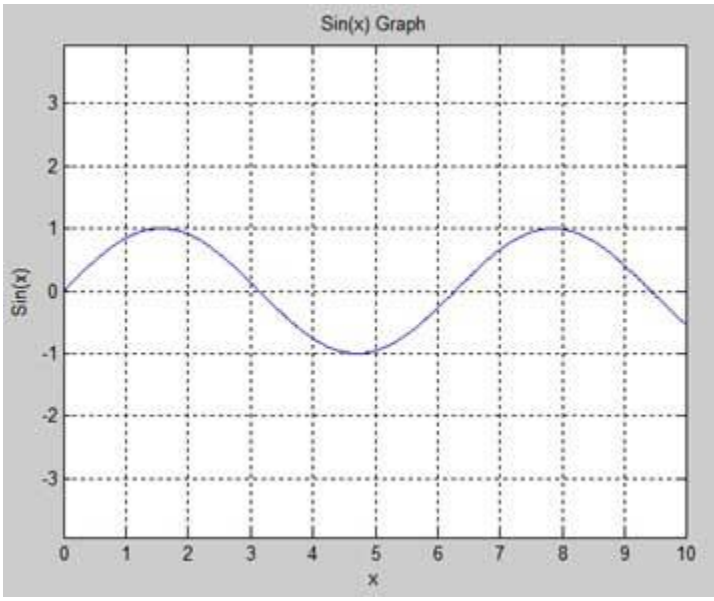
```
x = [0:0.01:10];
```

y = sin(x);

plot(x, y), xlabel('x'), ylabel('Sin(x)'), title('Sin(x) Graph'),

grid on, axis equal

MATLAB generates the following graph −



Drawing Multiple Functions on the Same Graph

To draw multiple graphs on the same plot.

Example

Create a script file and type the following code −

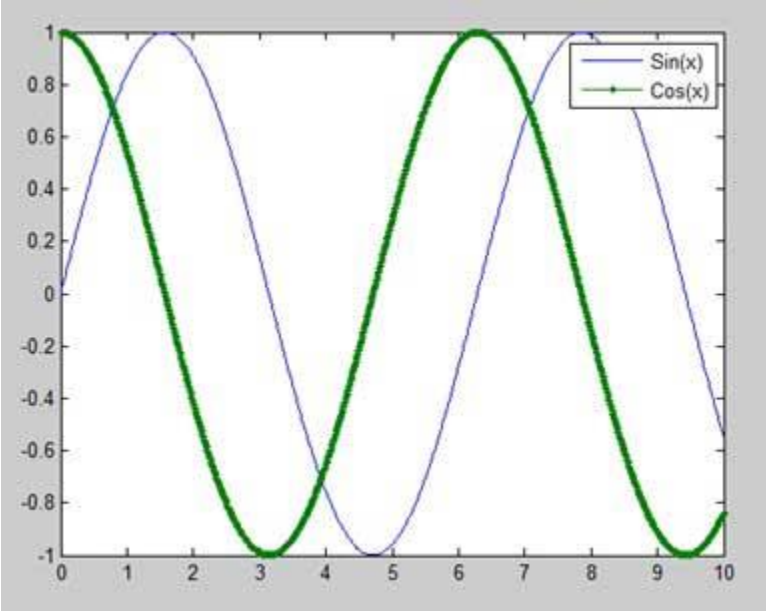
x = [0 : 0.01: 10];

y = sin(x);

g = cos(x);

plot(x, y, x, g, '.-'), legend('Sin(x)', 'Cos(x)')

MATLAB generates the following graph −



Setting Colors on Graph

MATLAB provides eight basic color options for drawing graphs. The following table shows the colors and their codes −

Code	Color
w	White
k	Black
b	Blue
r	Red
c	Cyan
g	Green
m	Magenta

y	Yellow

Example

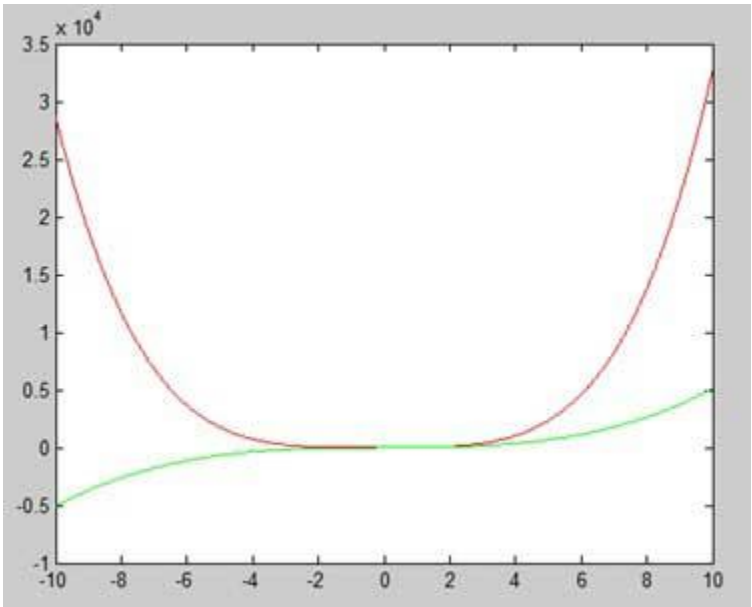
Let us draw the graph of two polynomials

- $f(x) = 3x^4 + 2x^3 + 7x^2 + 2x + 9$ and
- $g(x) = 5x^3 + 9x + 2$

Create a script file and type the following code −

```
x = [-10 : 0.01: 10];

y = 3*x.^4 + 2 * x.^3 + 7 * x.^2 + 2 * x + 9;

g = 5 * x.^3 + 9 * x + 2;

plot(x, y, 'r', x, g, 'g')
```

When you run the file, MATLAB generates the following graph −



Setting Axis Scales

The axis command allows you to set the axis scales. You can provide minimum and maximum values for x and y axes using the axis command in the following way −

axis ([xmin xmax ymin ymax])

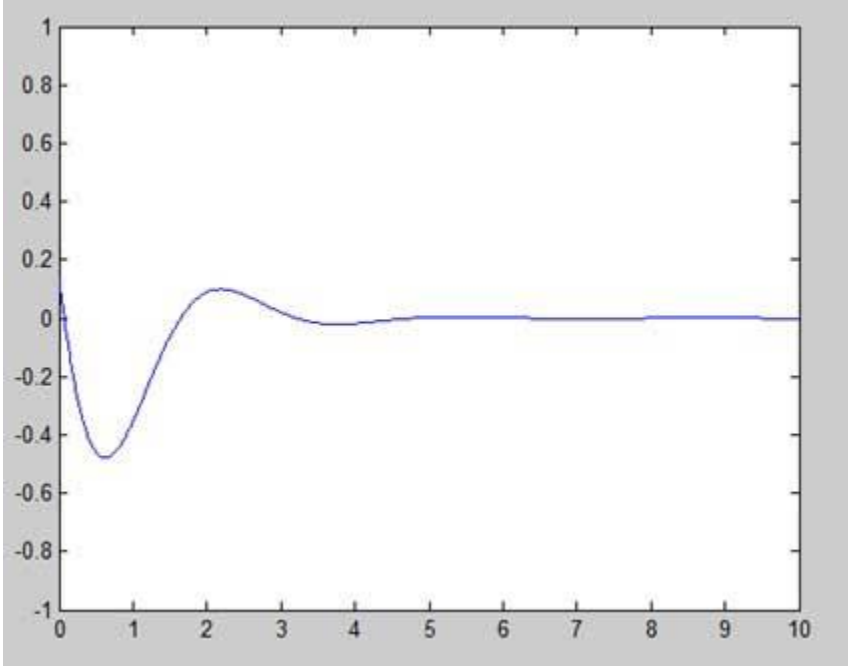
The following example shows this −

Example

Create a script file and type the following code −

```
x = [0 : 0.01: 10];

y = exp(-x).* sin(2*x + 3);

plot(x, y), axis([0 10 -1 1])
```

When you run the file, MATLAB generates the following graph −



Generating Sub-Plots

To create an array of plots in the same figure, each of these plots is called a subplot. The subplot command is used for creating subplots.

Syntax for the command is –

subplot(m, n, p)

where, *m* and *n* are the number of rows and columns of the plot array and *p*specifies where to put a particular plot.

Each plot created with the subplot command can have its own characteristics. Following example demonstrates the concept –

Example

Let us generate two plots –

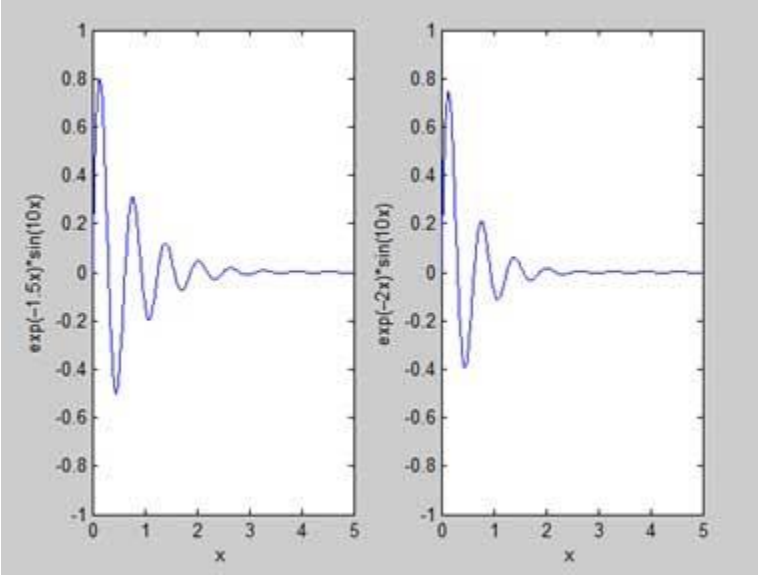
$y = e^{-1.5x}\sin(10x)$

$y = e^{-2x}\sin(10x)$

Create a script file and type the following code –

```
x = [0:0.01:5];

y = exp(-1.5*x).*sin(10*x);

subplot(1,2,1)

plot(x,y), xlabel('x'),ylabel('exp(–1.5x)*sin(10x)'),axis([0 5 -1 1])

y = exp(-2*x).*sin(10*x);

subplot(1,2,2)

plot(x,y),xlabel('x'),ylabel('exp(–2x)*sin(10x)'),axis([0 5 -1 1])
```

MATLAB generates the following graph –



Vectorization

A vector is a one-dimensional array of numbers. MATLAB allows creating two types of vectors −

- Row vectors
- Column vectors

Row Vectors

Row vectors are created by enclosing the set of elements in square brackets, using space or comma to delimit the elements.

r = [7 8 9 10 11]

MATLAB will execute the above statement and return the following result

r =

7 8 9 10 11

Column Vectors

Column vectors are created by enclosing the set of elements in square brackets, using semicolon to delimit the elements.

```
c = [7;  8;  9;  10; 11]
```

MATLAB will execute the above statement and return the following result

```
c =

    7

    8

    9

    10

    11
```

Referencing the Elements of a Vector

To reference one or more of the elements of a vector in several ways. The i^{th} component of a vector v is referred as v(i). For example −

```
v = [ 1; 2; 3; 4; 5; 6];    % creating a column vector of 6 elements

v(3)
```

MATLAB will execute the above statement and return the following result

```
ans =  3
```

When we reference a vector with a colon, such as v(:), all the components of the vector are listed.

```
v = [ 1; 2; 3; 4; 5; 6];    % creating a column vector of 6 elements
```

v(:)

MATLAB will execute the above statement and return the following result

ans =

1

2

3

4

5

6

MATLAB allows selecting a range of elements from a vector.

For example, let us create a row vector *rv* of 9 elements, then we will reference the elements 3 to 7 by writing *rv(3:7)* and create a new vector named *sub_rv*.

rv = [1 2 3 4 5 6 7 8 9];

sub_rv = rv(3:7)

MATLAB will execute the above statement and return the following result

sub_rv =

3 4 5 6 7

Vector Operations

- Addition and Subtraction of Vectors
- Scalar Multiplication of Vectors

- Transpose of a Vector
- Appending Vectors
- Magnitude of a Vector
- Vector Dot Product
- Vectors with Uniformly Spaced Elements

Addition and subtraction

To add or subtract two vectors. Both the operand vectors must be of same type and have same number of elements.

Example

Create a script file with the following code −

```
A = [7, 11, 15, 23, 9];

B = [2, 5, 13, 16, 20];

C = A + B;

D = A - B;

disp(C);

disp(D);
```

it displays the following result −

```
9    16    28    39    29

5     6     2     7   -11
```

Multiplication

To multiply a vector by a number, this is called the scalar multiplication. Scalar multiplication produces a new vector of same type with each element of the original vector multiplied by the number.

Example

Create a script file with the following code −

```
v = [ 12 34 10 8];

m = 5 * v
```

it displays the following result −

```
m =

   60   170    50    40
```

Transpose

The transpose operation changes a column vector into a row vector and vice versa. The transpose operation is represented by a single quote (').

Example

Create a script file with the following code −

```
r = [ 1 2 3 4 ];

tr = r';

v = [1;2;3;4];

tv = v';

disp(tr); disp(tv);
```

When you run the file, it displays the following result −

```
   1
```

2

3

4

1 2 3 4

Matrix Using Matlab

A matrix is a two-dimensional array of numbers.

In MATLAB, you create a matrix by entering elements in each row as comma or space delimited numbers and using semicolons to mark the end of each row.

For example, let us create a 4-by-5 matrix *a* −

a = [1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8]

MATLAB will execute the above statement and return the following result −

a =

1 2 3 4 5

2 3 4 5 6

3 4 5 6 7

4 5 6 7 8

Referencing the Elements of a Matrix

To reference an element in the m^{th} row and n^{th} column, of a matrix *mx*, we can write as

mx(m, n);

For example, to refer to the element in the 2nd row and 5th column, of the matrix *a*, as created in the last section, we type −

```
a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
a(2,5)
```

MATLAB will execute the above statement and return the following result

```
ans =  6
```

To reference all the elements in the mth column we type A(:,m).

Let us create a column vector v, from the elements of the 4th row of the matrix a

```
a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
v = a(:,4)
```

MATLAB will execute the above statement and return the following result

```
v =
    4
    5
    6
    7
```

To select the elements in the mth through nth columns, for this we can write as

```
a (:,m:n)
```

To create a smaller matrix taking the elements from the second and third columns

```
a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
a(:, 2:3)
```

MATLAB will execute the above statement and return the following result −

```
ans =
    2    3
    3    4
    4    5
    5    6
```

In the same way, you can create a sub-matrix taking a sub-part of a matrix.

```
a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
a(:, 2:3)
```

MATLAB will execute the above statement and return the following result −

```
ans =
    2    3
    3    4
    4    5
    5    6
```

In the same way, we can create a sub-matrix taking a sub-part of a matrix.

For example, let us create a sub-matrix *sa* taking the inner subpart of a −

```
3    4    5
4    5    6
```

To do this, write −

```
a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
sa = a(2:3,2:4)
```

MATLAB will execute the above statement and return the following result −

```
sa =

    3    4    5

    4    5    6
```

Deleting a Row or a Column in a Matrix

Suppose we want to delete an entire row or column of a matrix by assigning an empty set of square braces [] to that row or column. Basically, [] denotes an empty array.

For example, let us delete the fourth row of a −

```
a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
a( 4 , : ) = []
```

MATLAB will execute the above statement and return the following result −

```
a =
    1    2    3    4    5
    2    3    4    5    6
    3    4    5    6    7
```

Next, let us delete the fifth column of a −

```
a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8];
a(: , 5)=[]
```

MATLAB will execute the above statement and return the following result −

```
a =
    1    2    3    4
    2    3    4    5
    3    4    5    6
    4    5    6    7
```

Example

In this example, let us create a 3-by-3 matrix m, then we will copy the second and third rows of this matrix twice to create a 4-by-3 matrix.

Create a script file with the following code −

```
a = [ 1 2 3 ; 4 5 6; 7 8 9];
new_mat = a([2,3,2,3],:)
```

When you run the file, it displays the following result −

```
new_mat =
    4    5    6
    7    8    9
    4    5    6
    7    8    9
```

Matrix Operations

Basic and commonly used matrix operations −

- Addition and Subtraction of Matrices
- Division of Matrices
- Scalar Operations of Matrices
- Transpose of a Matrix
- Concatenating Matrices
- Matrix Multiplication
- Determinant of a Matrix
- Inverse of a Matrix

Addition and Subtraction of Matrix

We can add or subtract matrices. Both the operand matrices must have the same number of rows and columns.

Example

Create a script file with the following code −

```
a = [ 1 2 3 ; 4 5 6; 7 8 9];

b = [ 7 5 6 ; 2 0 8; 5 7 1];

c = a + b

d = a - b
```

To run the file, it displays the following result −

```
c =

     8     7     9

     6     5    14

    12    15    10

d =

    -6    -3    -3

     2     5    -2

     2     1     8
```

Multiplication Operation

Consider two matrices A and B. If A is an m x n matrix and B is an n x p matrix, they could be multiplied together to produce an m x n matrix C.

Matrix multiplication is possible only if the number of columns n in A is equal to the number of rows n in B.

In matrix multiplication, the elements of the rows in the first matrix are multiplied with corresponding columns in the second matrix.

Each element in the (i, j)th position, in the resulting matrix C, is the summation of the products of elements in ith row of first matrix with the corresponding element in the jth column of the second matrix.

Matrix multiplication in MATLAB is performed by using the * operator.

Example

Create a script file with the following code −

```
a = [ 1 2 3; 2 3 4; 1 2 5]

b = [ 2 1 3 ; 5 0 -2; 2 3 -1]

prod = a * b
```

To run the file, it displays the following result −

```
a =

     1     2     3

     2     3     4

     1     2     5

b =

     2     1     3

     5     0    -2

     2     3    -1
```

```
prod =

    18    10    -4

    27    14    -4

    22    16    -6
```

Transpose Operation

The transpose operation switches the rows and columns in a matrix. It is represented by a single quote(').

Example

Create a script file with the following code −

```
a = [ 10 12 23 ; 14 8 6; 27 8 9]

b = a'
```

To run the file, it displays the following result −

```
a =

    10    12    23

    14     8     6

    27     8     9

b =

    10    14    27

    12     8     8

    23     6     9
```

Inverse Operations

The transpose operation switches the rows and columns in a matrix. It is represented by a single quote(').

Example

Create a script file with the following code −

```
a = [10 12 23; 14 8 6; 27 8 9]
b = a'
```

it displays the following result −

```
a =
    10    12    23
    14     8     6
    27     8     9

b =
    10    14    27
    12     8     8
    23     6     9
```

Part B (2 Marks)

1. Explain what is MATLAB? Where MATLAB can be applicable?
2. List out the operators that MATLAB allows?
3. What does MATLAB consist of?
4. What is a variable in MATLAB?
5. What is an Expression? Give one example.
6. What is an Array**?**
7. What is meant by Vectorization?
8. Define Matrix

Part C (6 Marks)

1. Describe about Matrices addition and multiplication with example.
2. Explain about plotting of data.
3. Describe about Vector operation with example.
4. Differentiate between matrix and vector.
5. Illustrate on Matrix Operations.
6. Discuss in detail about transpose and Inverse operation in matrix.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of CS,CA & IT

III B.Sc(CS) (BATCH 2016-2019)

Machine Learning

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

ONLINE EXAMINATIONS ONE MARK QUESTIONS

QUESTIONS	OPT1	OPT2	OPT3	OPT4	ANSWER
MATLAB stands for _____	Maths Laboratory	Matrix Laboratory	Mathematical Lab	Maths Lab	Matrix Laboratory
MATLAB was developed by _____	MathsWorks	Intel	Microsoft	IBM	MathsWorks
In MATLAB the matrix is defined as an _____	vector	scalar	array	integer	array
_____ acts as an outstanding tool for visulaizing technical data	C	C++	Java	MATLAB	MATLAB
In command window the _____ are entered	datas	values	commands	fiels	commands

_____ window displays plots and graphs	command	Edit	Figure	Command history	Figure
A variable can be deleted from the workspace with the _____ command	delete	remove	clear	omit	clear
The _____ command will display a list of possible help topics in the command window	help	helper	lookfor	order	help
The _____ operator swaps the row and columns of any array that is given	transpose	concatenates	colon	semicolon	transpose
The _____ function can be used ti create an all zero array	ones	zero	eye	randn	zero
The _____ function can be used to generate arrays containing all ones	ones	zero	eye	randn	ones

The _____ function accepts an array arugment and displayus the value of athe array in the command window	disp	format	special	fprintf	disp
The MATLAB command to make a plot is	figure	fit	plot	pplot	plot
The command to add text to the x axis of a plot is	xtitle	label,x	xlabel	xtext	xlabel
The basic building block in MATLAB is _____	matrix	vector	scalar	functions	matrix
The _____ command clears the screen	clc	clr	cls	cle	clc
When the _____ function is executed, MATLAB opens the Figure window and displays the plot in that window	edit	figure	plotting	plot	plot

_____ function is used to find the minimum of given numbers	min	max	medium	poor	min
_____ function is used to find the maximum of given numbers	poor	min	max	medium	max
The _____ command can be used to save a plot as a graphical image by specifying appropriate options and a filename	plot	print	draw	multiple	print
The _____ gives the transpose of x	x'	x''	x'''	x	x'
_____ are operations performed between arrays on an element by element basis	matrix operations	array operations	vector operations	arthimetic operations	array operations
In _____ the number of rows and columns in both arrays must be the same	matrix operations	array operations	vector operations	arthimetic operations	array operations

The term _____ is used to describe an array with only one dimension	array	vector	matrix	scale	vector
The term _____ is used to describe an array with two or more dimensions	array	vector	matrix	scale	matrix
_____ window displays plots and graphs	command	Edit	Figure	Command history	Figure
In MATLAB, the process of replacing loops by vectorized statements is known as _____	scalarization	vectorization	looping	branching	vectorization

UNIT-III

Linear Regression: Prediction using Linear Regression, Gradient Descent, Linear Regression with one variable, Linear Regression with multiple variables, Polynomial Regression, Feature Scaling/Selection.

Logistic Regression: Classification using Logistic Regression, Logistic Regression vs. Linear Regression, Logistic Regression with one variable and with multiple variables.

Regression

- Technique used for the modeling and analysis of numerical data.
- Exploits the relationship between two or more variables so that we can gain information about one of them through knowing values of the other.
- Regression can be used for prediction, estimation, hypothesis testing, and modeling causal relationships.

Y = X1 + X2 + X3

Dependent Variable	Independent Variable
Outcome Variable	Predictor Variable
Response Variable	Explanatory Variable

Linear Regression

Regression is a parametric technique used to predict continuous (dependent) variable given a set of independent variables.

It is parametric in nature because it makes certain assumptions based on the data set.

If the data set follows those assumptions, regression gives incredible results. Otherwise, it struggles to provide convincing accuracy.

Mathematically, regression uses a linear function to approximate (predict) the dependent variable given as:

$$Y = \beta o + \beta 1X + \in$$

where, Y - Dependent variable

X - Independent variable

βo – Intercept

β1 – Slope

∈ - Error

βo and β1 are known as coefficients. This is the equation of simple linear regression. It's called 'linear' because there is just one independent variable (X) involved. In multiple regression, we have many independent variables (Xs). If you recall, the equation above is nothing but a line equation (y = mx + c) we studied in schools.

Parameters are:

Y - This is the variable we predict

X - This is the variable we use to make a prediction

βo - This is the intercept term. It is the prediction value you get when X = 0

β1 - This is the slope term. It explains the change in Y when X changes by 1 unit.

∈ - This represents the residual value, i.e. the difference between actual and predicted values.

Error is an inevitable part of the prediction-making process. No matter how powerful the algorithm we choose, there will always remain an (∈) irreducible error which reminds us that the "future is uncertain."

Linear Regression

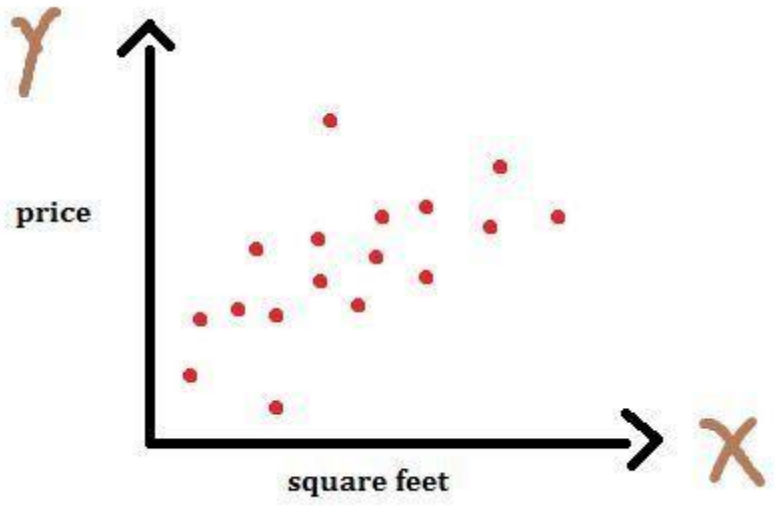
Linear Regression is a simple machine learning model for regression problems, i.e., when the target variable is a real value.

Example

Let's start with an example — suppose we have a dataset with information about the area of a house (in square feet) and its price (in thousands of dollars) and our task is to build a machine learning model which can predict the price given the area. Here is what our dataset looks like

area (sq.ft)	price (1k$s)
3456	600
2089	395
1416	232

If we plot our data, we might get something similar to the following:



Making Predictions with Linear Regression

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs.

Let's make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

y = B0 + B1 * x1

or

weight =B0 +B1 * height

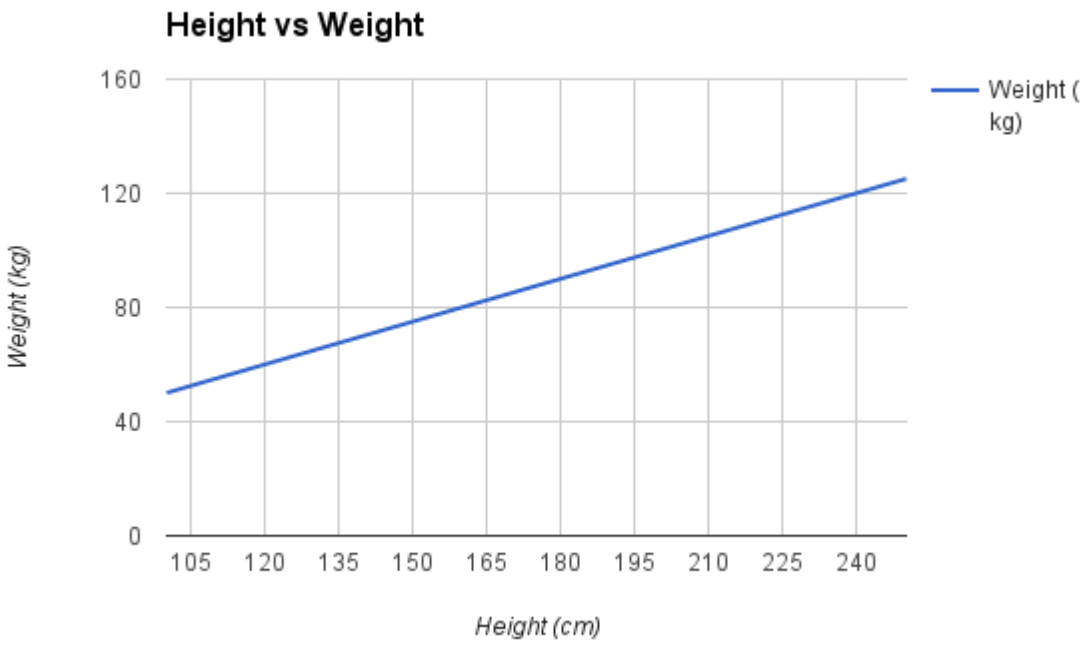
Where B0 is the bias coefficient and B1 is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.

For example, lets use B0 = 0.1 and B1 = 0.5. Let's plug them in and calculate the weight (in kilograms) for a person with the height of 182 centimeters.

weight = 0.1 + 0.05 * 182

weight = 91.1

You can see that the above equation could be plotted as a line in two-dimensions. The B0 is our starting point regardless of what height we have. We can run through a bunch of heights from 100 to 250 centimeters and plug them to the equation and get weight values, creating our line.



Sample Height vs Weight Linear Regression

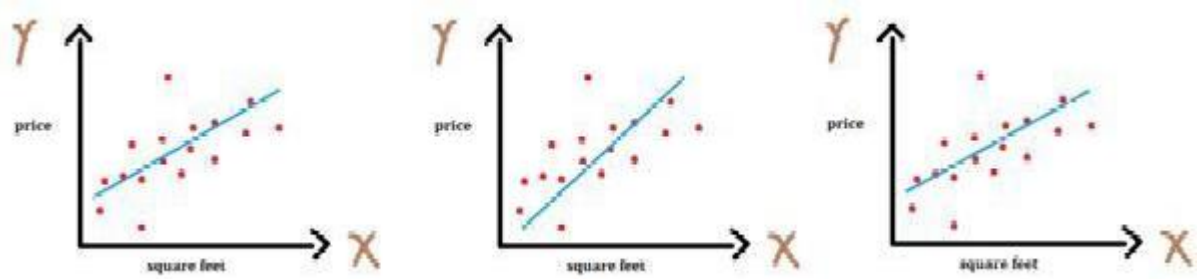
Now that we know how to make predictions given a learned linear regression model, let's look at some rules of thumb for preparing our data to make the most of this type of model.

Simple linear regression

In simple linear regression, we establish a relationship between target variable and input variables by fitting a line, known as the regression line.

In general, a line can be represented by linear equation y = m * X + b. Where, y is the dependent variable, X is the independent variable, m is the slope, b is the intercept.

In machine learning, we rewrite our equation as y(x) = w0 + w1 * x where w's are the parameters of the model, x is the input, and y is the target variable. Different values of w0 and w1 will give us different lines, as shown below

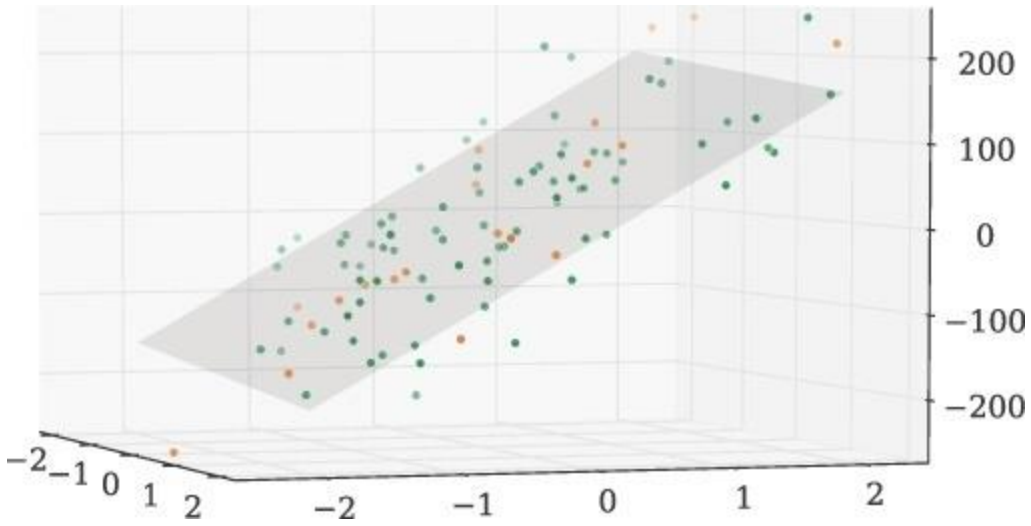


Multiple linear regression

The above equation can be used when we have one input variable (also called feature). However, in general, we usually deal with datasets which have multiple input variables. The case when we have more than one feature is known as multiple linear regression, or simply, linear regression. We can generalize our previous equation for simple linear regression to multiple linear regression:

$y(x)=w_0+w_1x_1+w_2x_2+\ldots+w_nx_n$

In the case of multiple linear regression, instead of our prediction being a line in 2-dimensional space, it is a hyperplane in n-dimensional space. For example, in 3D, our plot would look as follows



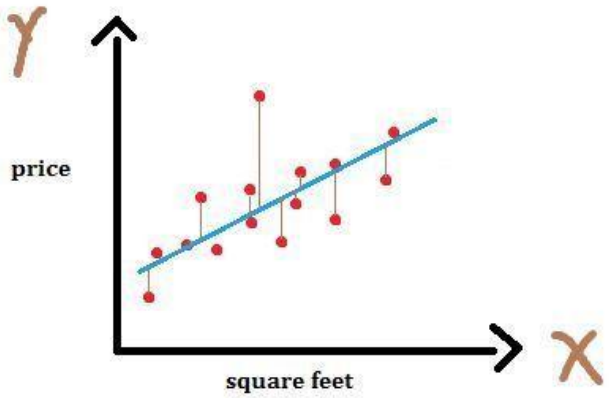
Cost functions

Different values of the weights (w0, w1, w2, ... wn) gives us different lines, and our task is to find weights for which we get best fit. One question you may have is, how can we determine how well a particular line fits our data? Or, given two lines, how do we determine which one is better? For this, we introduce a cost function which measures, given a particular value for the w's, how close the y's are to corresponding ytrue's. That is, how well do a particular set of weights predict the target value.

For linear regression, we use the mean squared error cost function. It is the average over the various data points (xi, yi) of the squared error between the predicted value y(x) and the target value ytrue.

$$\mathbf{J}(w) = \frac{1}{n}\sum_{i=1}^{n}[(\mathbf{y}(\square]x^i) - y_{true}^i)^2$$

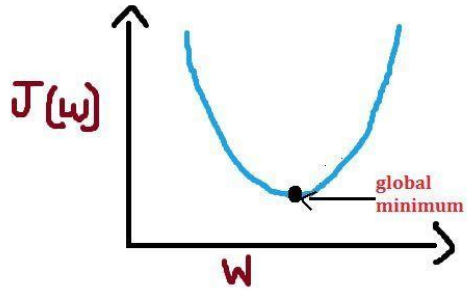
The cost function defines a cost based on the distance between true target and predicted target (shown in the graph as lines between sample points and the regression line), also known as the residual. The residuals are visualized below,



If a particular line is far from all the points, the residuals will be higher, and so will the cost function. If a line is close to the points, the residuals will be small, and hence the cost function.

Optimization using Gradient Descent

Each value of the weight vector w gives us a corresponding cost J(w). We want to find the value of weights for which cost is minimum. We can visualize this as follows:

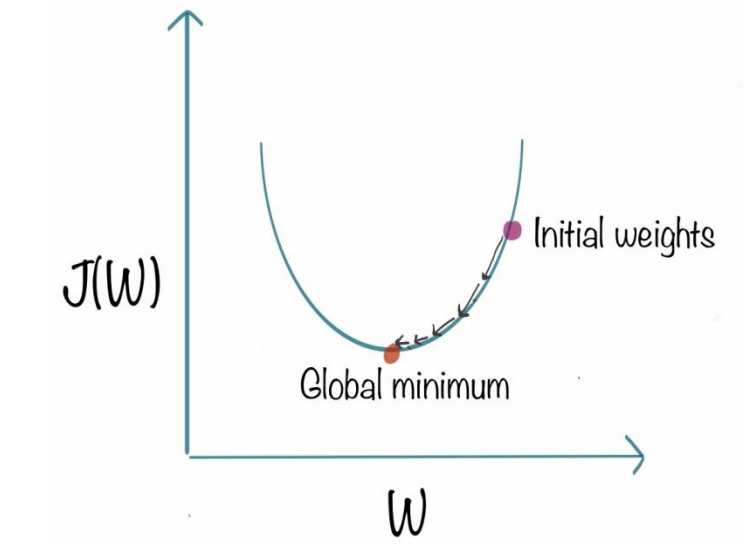


Note: Above we have used the word "global" because the shape of the cost-function for linear regression is convex (i.e. like a bowl). It has a single minimum, and it smoothly increases in all directions around it.

Given the linear regression model and the cost function, we can use Gradient Descent (covered in the next article) to find a good set of values for the weight vector. This process of finding the best model out of the many possible models is called optimization.

Gradient Descent

Gradient Descent is one of the most popular and widely used optimization algorithms. Given a machine learning model with parameters (weights and biases) and a cost function to evaluate how good a particular model is, our learning problem reduces to that of finding a good set of weights for our model which minimizes the cost function.



Gradient descent is an iterative method. We start with some set of values for our model parameters (weights and biases), and improve them slowly. To improve a given set of weights, we try to get a sense of the value of the cost function for weights similar to the current weights (by calculating the gradient) and move in the direction in which the cost function reduces. By repeating this step thousands of times we'll continually minimize our cost function.

Pseudocode for Gradient Descent

Gradient descent is used to minimize a cost function J(w) parameterized by a model parameters w. The gradient (or derivative) tells us the incline or slope of the cost function. Hence, to minimize the cost function, we move in the direction opposite to the gradient.

- initialize the weights w randomly
- calculate the gradients $G=\nabla wJ(w)$ of cost function w.r.t parameters
- update the weights by an amount proportional to G, i.e. $w=w-\eta \cdot G$
- repeat till J(w) stops reducing or other pre-defined termination criteria is met

In step 3, η is the learning rate which determines the size of the steps we take to reach a minimum. We need to be very careful about this parameter since high values of η may overshoot the minimum and very low value will reach minimum very slowly.

A popular sensible choice for the termination criteria is that the cost J(w) stops reducing on the validation dataset.

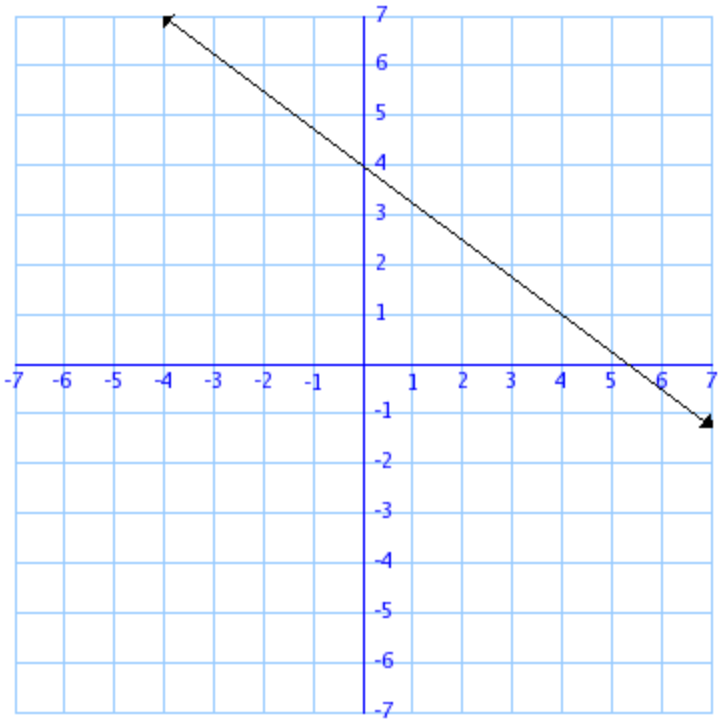
Polynomial Regression

When there are multiple features, to better fit our hypothesis function and cost function graph we can combine multiple features into a single one.

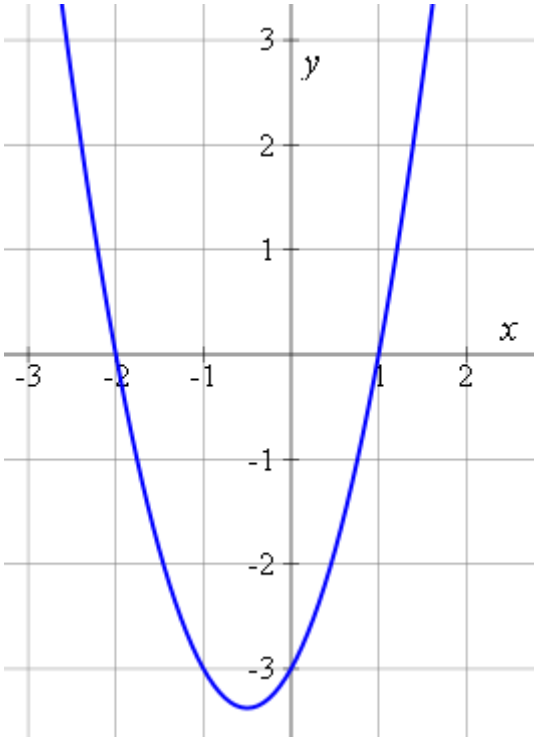
For example, if our features are length and breadth of a house then, we can combine the length and breadth and make a new feature called area which will be the multiplication of length and breadth.

There is no reason for our hypothesis function to be just a linear function. It can be a quadratic function or a cubic function or a square root function.

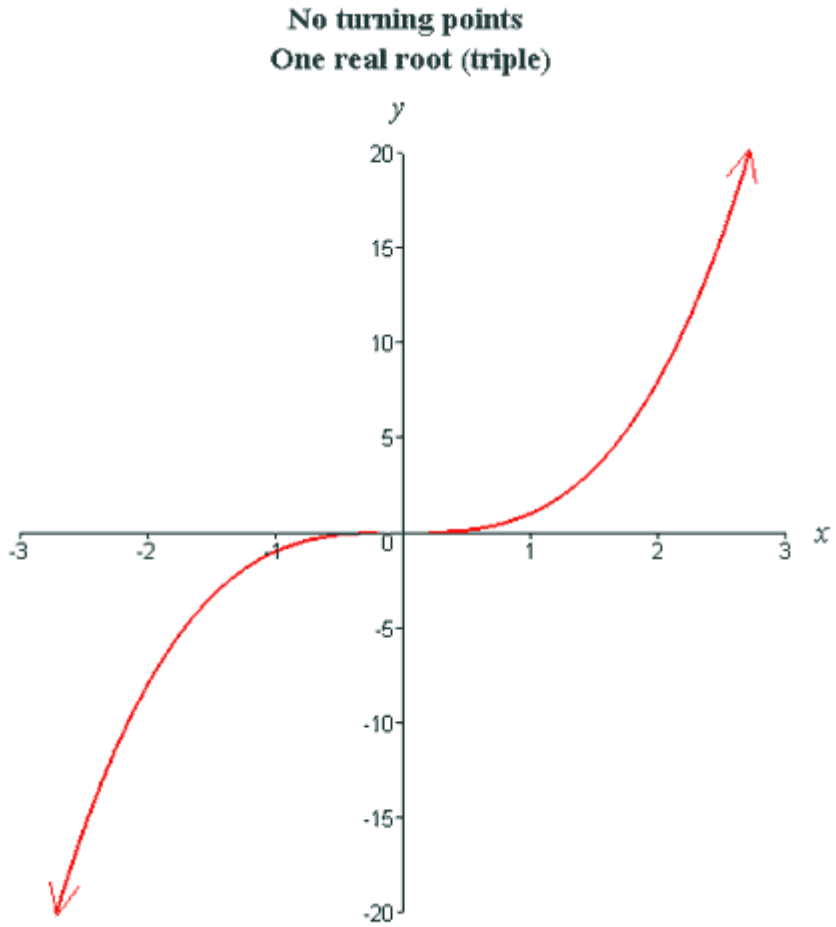
For example, if our hypothesis function is $h_\theta(x) = \theta_0 + \theta_1 x_1$ then we can create additional features based on x_1, to get the quadratic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ or the cubic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$



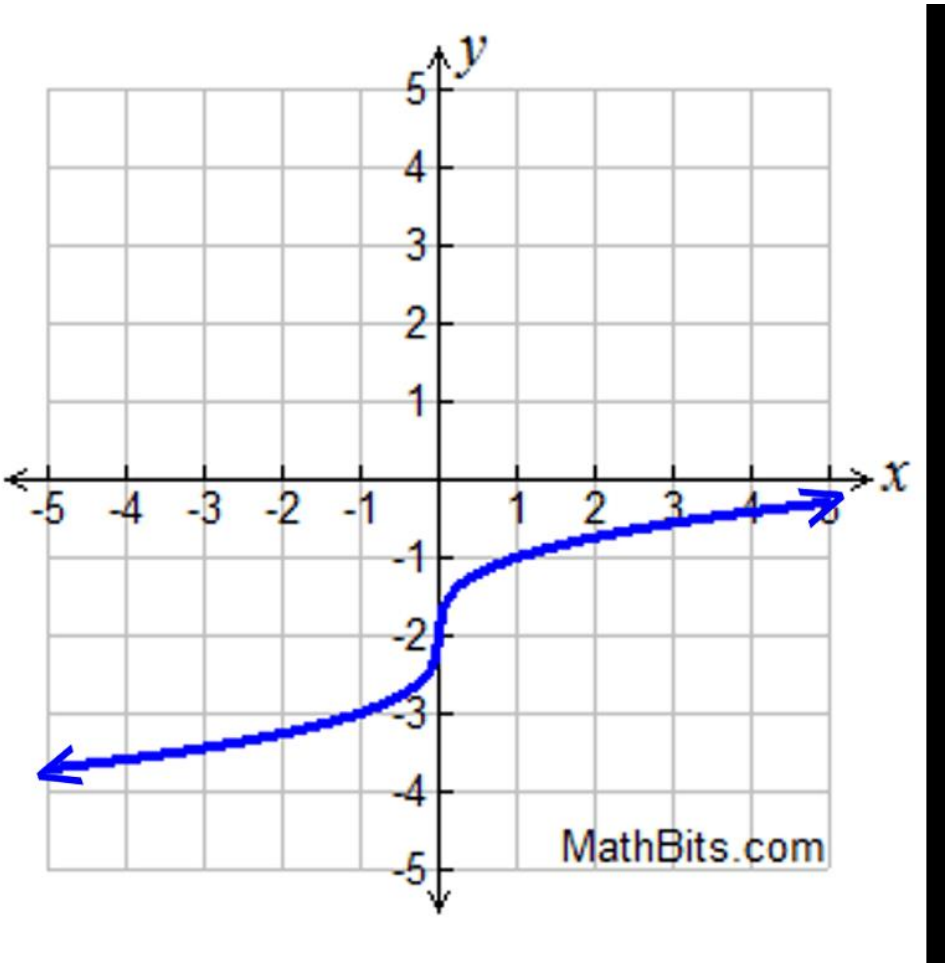
a typical graph for a linear equation



a typical quadratic graph is a parabola



a typical graph for a cubic function.



a typical graph of a root function

Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

The simplest method is rescaling the range of features to scale the range in [0, 1] or [−1, 1]. The general formula is given as:

X' = (X—Xmin)/(Xmax—Xmin)

where, X' is the value we want to rescale, X is the given value, Xmax is the largest value of X and Xmin the smallest.

Let us consider, old weights = [115,140,175] and we are going to scale for the value 140.

X' = (140–115)/(175–115) = 0.41666

Therefore, the range is, [0,0.41666,1]

Feature Selection

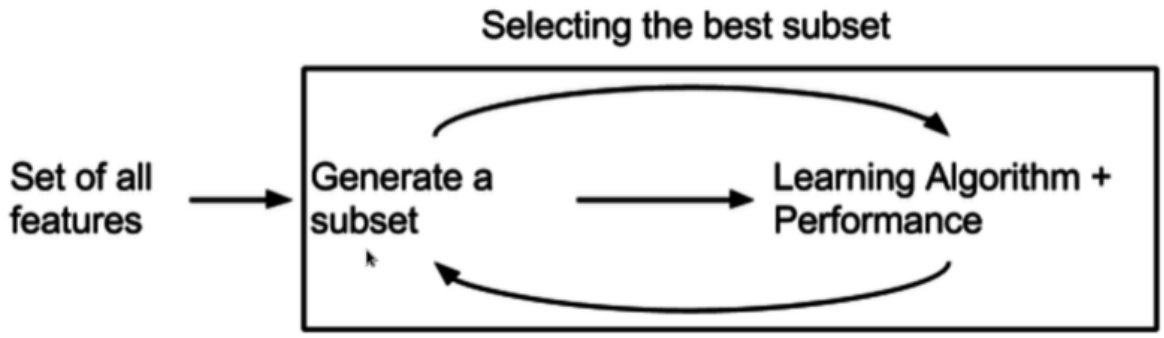
Why do we have to perform feature selection?

- Knowledge discovery, Interpretability and to gain some insights.
- Curse of dimensionality.

There are two methods of Feature Selection :

- Filtering—Filter type methods select variables regardless of the model. They are based only on general features like the correlation with the variable to predict. Filter methods suppress the least interesting variables. They are mainly used as a pre-process method.

- Wrapping—Wrapper methods evaluate subsets of variables which allows, unlike filter approaches, to detect the possible interactions between variables.



Feature Transformation:

Feature transformation is a group of methods that create new features (predictor variables). Feature selection is a subset of feature transformation.

Consider an 'X' space having 'n' features, using feature transformation we are going to transform X to have 'm' features, where m < n.

This is done by defining some matrix Px which is a subspace to which we are going to project 'X' space. The new features are combination of the old features.

There are many types explained below -

Principal Component Analysis (PCA)

A movie camera takes a 3-D information and flatten it to 2-D without too much loss of information.

What does all of this have to do with PCA?

PCA takes a dataset with a lot of dimensions and flatten it to two or three dimensions so we can look at it.

It tries to find a meaningful way to flatten the data by focusing on the things that are different between cells.

Here, the weights are termed Loadings. And array of loadings is called "Eigen Vector".

PCA review :

- Systematized way to transform input features into principal components (PC)
- Use new PCs as new features.
- PCs are directions in data that maximize variance when you project/compress down onto them.
- The more variance of data along a PC, the higher that PC is ranked.
- Most variance, most information would be the first PC.
- Second-most variance would be the second PC.
- Max number of PCs = number of input features.

Typical example of PCA is in eigenfaces.

PCA is a linear algebraic approach.

Independent Components Analysis (ICA)

It is a computational method for separating a multivariate signal into additive sub-components. This is done by assuming that the sub-components are non-Gaussian signals and that they are statistically independent from each other.

A common example application is the "cocktail party problem" of listening in on one person's speech in a noisy room.

ICA is a probabilistic approach.

Random Component Analysis (RCA)

Uses random way to transform input features into principal components (PC)

Linear Discriminant Analysis (LDA)

Finds a projection that descriminates based on the label.

Fundamental assumption is different, although they do the same thing, which is to capture the original data in some new transform space that is somehow better.

Logistic regression

Logistic regression can be used to model and solve such problems, also called as binary classification problems.

A key point to note here is that Y can have 2 classes only and not more than that. If Y has more than 2 classes, it would become a multi class classification and you can no longer use the vanilla logistic regression for that.

Yet, Logistic regression is a classic predictive modelling technique and still remains a popular choice for modelling binary categorical variables.

Another advantage of logistic regression is that it computes a prediction probability score of an event. More on that when you actually start building the models.

Some real world examples of binary classification problems

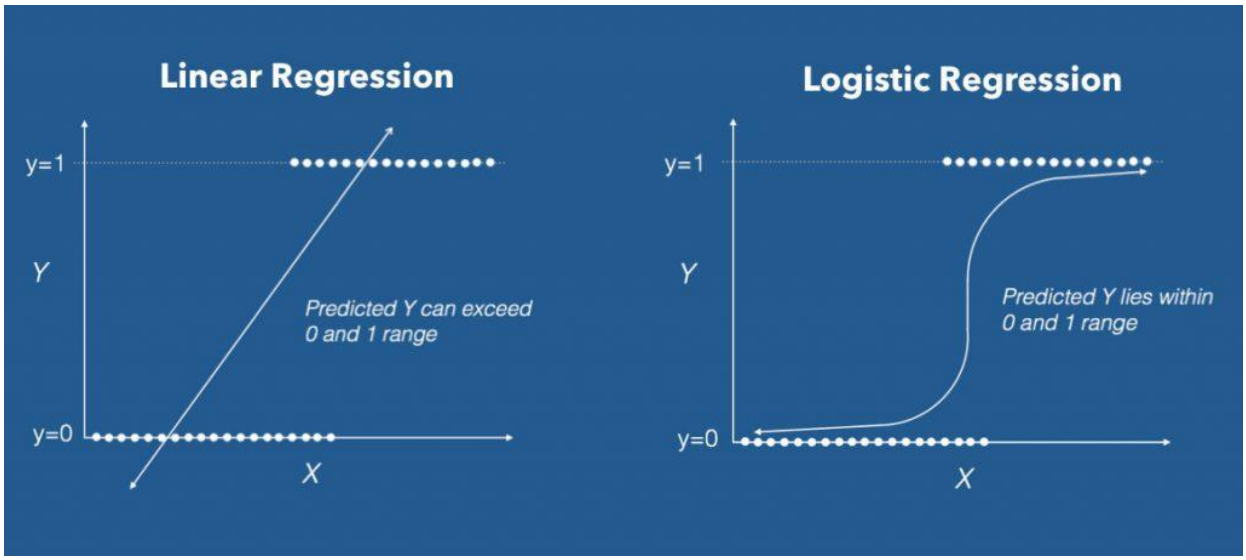
You might wonder what kind of problems you can use logistic regression for. Here are some examples of binary classification problems:

- Spam Detection : Predicting if an email is Spam or not
- Credit Card Fraud : Predicting if a given credit card transaction is fraud or not
- Health : Predicting if a given mass of tissue is benign or malignant
- Marketing : Predicting if a given user will buy an insurance product or not
- Banking : Predicting if a customer will default on a loan.

Why not linear regression?

When the response variable has only 2 possible values, it is desirable to have a model that predicts the value either as 0 or 1 or as a probability score that ranges between 0 and 1.

Linear regression does *not* have this capability. Because, If you use linear regression to model a binary response variable, the resulting model may not restrict the predicted Y values within 0 and 1.



LINEAR VS LOGISTIC REGRESSION

This is where logistic regression comes into play. In logistic regression, you get a probability score that reflects the probability of the occurence of the event.

An event in this case is each row of the training dataset. It could be something like classifying if a given email is spam, or mass of cell is malignant or a user will buy a product and so on.

Types of Logistic Regression

The types of Logistic Regression are,

1.**Ordinal logistic regression**

2.**Multinomial Logistic regression**

3.**Binomial Logistic regression**

Ordinal logistic regression

If the values of dependent variable are ordinal, then it is called as Ordinal logistic regression. Ordinal regression is used to predict the dependent variable with 'ordered' multiple categories given one or more independent variables.

Example : To predict the belief that the tax is too high, the dependent variable ranges from strongly agree to strongly disagree and the independent variables are age and income. In this case, we will use the ordinal logistic regression.

Multinomial Logistic regression

Multinomial logistic regression is used to predict a nominal dependent variable given one or more independent variables. It is sometimes considered as extension of binomial logistic regression.

Example : To understand which type of drink consumers prefer based on location in the US and age. The dependent variables would be type of the drink (Coffee, Soft Drink, Tea and Water) and the independent variables would be the nominal variable, location in US and the age (in years).

Binomial Logistic regression

A binomial logistic regression, predicts the probability that an observation falls into one of two categories of a dichotomous dependent variable based on one or more independent variables that can be either continuous or categorical. This is often called as simple logistic regression.

Example : Let us predict, whether students will pass or not (i.e. The dependent variables are Pass and Fail.) in their final exam based on the internal marks , assignment submission and few other independent variables.

Type	Dependent	Independent
Multi Logistic Regression	Categorical	More than one X variable
Binary Logistic Regression	Categorical with only two levels(Converted, Not Converted)	One or More X variables
Ordinal Logistic Regression	Ordinal with more than two levels (High,Meduim,Low)	One or More X variables
Multinomial	Nominal with more than two	One or More X variables

Logistic Regression	levels (Green,Blue,Red)	

Difference between Linear and Logistic Regression

1. Variable Type : Linear regression requires the dependent variable to be continuous i.e. numeric values (no categories or groups). While Binary logistic regression requires the dependent variable to be binary - two categories only (0/1). Multinominal or ordinary logistic regression can have dependent variable with more than two categories.

2. Algorithm : Linear regression is based on ***least square estimation*** which says regression coefficients should be chosen in such a way that it minimizes the sum of the squared distances of each observed response to its fitted value.

While logistic regression is based on ***Maximum Likelihood Estimation*** which says coefficients should be chosen in such a way that it maximizes the **Probability of Y given X** (likelihood). With ML, the computer uses different "iterations" in which it tries different solutions until it gets the maximum likelihood estimates.

3. Equation :

Multiple Regression Equation :

$$Y = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \ldots\ldots\ldots + b_kX_k$$

Linear Regression Equation

Y is target or dependent variable, b0 is intercept. x1,x2,x3...xk are predictors or independent variables. b1,b2,b3....bk is coefficients of respective predictors.

Logistic Regression Equation :

P(y=1) = e(b0 + b1x1 + b2x2 +-----bkxk) / (1+e(b0+b1x1+ b2x2+------bkxk))

Which further simplifies to :

P(y=1) = 1 / (1+ exp -(b0+b1x1+ b2x2+------bkxk))

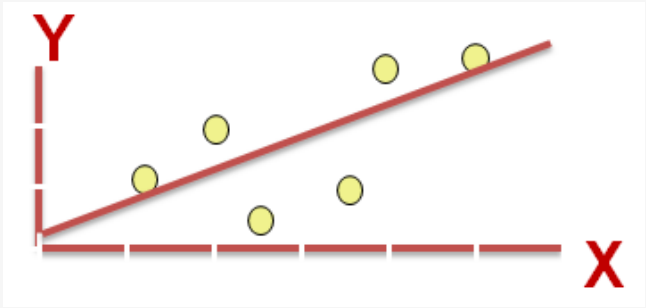
$$p = \frac{1}{1 + e^{-(b_0 + b_1X_1 + b_2X_2 + --- + b_kX_k)}}$$

Logistic Regression Equation

The above function is called logistic or sigmoid function.

4. Curve :

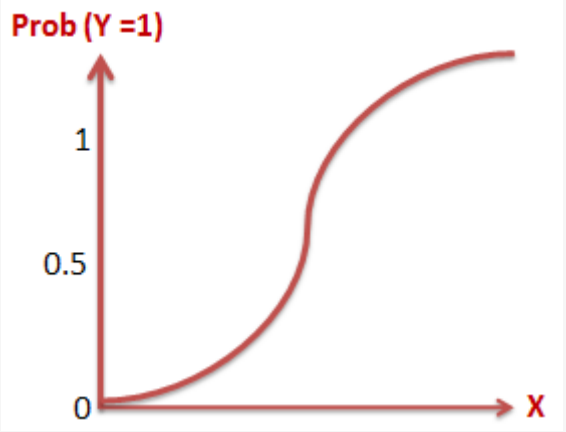
Linear Regression : Straight line



Straight Line : Linear Regression

Linear regression aims at finding the best-fitting straight line which is also called a regression line. In the above figure, the red diagonal line is the best-fitting straight line and consists of the predicted score on Y for each possible value of X. The distance between the points to the regression line represent the errors.

Logistic Regression : S Curve



Logistic S-shaped curve

Changing the coefficient leads to change in both the direction and the steepness of the logistic function. It means positive slopes result in an S-shaped curve and negative slopes result in a Z-shaped curve.

5. Linear Relationship : Linear regression needs a linear relationship between the dependent and independent variables. While logistic regression does not need a linear relationship between the dependent and independent variables.

6. Normality of Residual : Linear regression requires error term should be normally distributed. While logistic regression does not require error term should be normally distributed.

7. Homoscedasticity : Linear regression assumes that residuals are approximately equal for all predicted dependent variable values. While Logistic regression does not need residuals to be equal for each level of the predicted dependent variable values.

8. Sample Size : Linear regression requires 5 cases per independent variable in the analysis.While logistic regression needs at least 10 events per independent variable.

9. Purpose : Linear regression is used to estimate the dependent variable incase of a change in independent variables**.** For example, relationship between number of hours studied and your grades. Whereas logistic regression is used to calculate the probability of an event. For

example, an event can be whether customer will attrite or not in next 6 months.

10. Interpretation : Betas or Coefficients of linear regression is interpreted like below.

Keeping all other independent variables constant, how much the dependent variable is expected to increase/decrease with an unit increase in the independent variable.

In logistic regression, we interpret odd ratios -

The effect of a one unit of change in X in the predicted odds ratio with the other variables in the model held constant.

11. Distribution :

Linear regression assumes normal or gaussian distribution of dependent variable. Whereas, **Logistic regression** assumes binomial distribution of dependent variable. **Note :** Gaussian is the same as the normal distribution.

For Example

```
set.seed(123)
y = ifelse(runif(100) < 0.5, 1,0)

x = sample(1:100,100)

y1 = sample(100:1000,100, replace=T)
```

Linear Regression

```
glm(y1 ~ x, family = gaussian(link = "identity"))
```

Coefficients:

(Intercept) x 600.6152 -0.8631

Degrees of Freedom: 99 Total (i.e. Null); 98 Residual

Null Deviance: 7339000

Residual Deviance: 7277000 AIC: 1409

Logistic Regression

*glm(y ~ x, family = **binomial(**link = "logit"))*

Coefficients:

(Intercept) x -0.024018 0.005279

Degrees of Freedom: 99 Total (i.e. Null); 98 Residual

Null Deviance: 137.2

Residual Deviance: 136.6 AIC: 140.6

12. Link Function Linear regression uses **Identity** link function of gaussian family. Whereas, logistic regression uses **Logit** function of Binomial family.

13. Computational Time: Linear regression is very fast as compared to logistic regression as logistic regression is an iterative process of maximum likelihood.

Part B (2 Marks)

1. Define Regression.
2. What is the difference between dependent and independent variable?
3. What are functions to be used for best fit line?
4. Define Gradient.
5. Write brief on polynomial regression.
6. Brief short note on Feature scaling.
7. Simplify on Linear regression with multiple variable.

Part C (6 Marks)

1. Differentiate between Linear and Logistics Regression.
2. Describe about Linear Regression in machine learning.
3. How to predict House value using linear regression.
4. Discuss on Logistic regression.
5. Illustrate on Classification using logistic regression

KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of CS,CA & IT

III B.Sc(CS) (BATCH 2016-2019)

Machine Learning

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

SN(	QUESTIONS	OPTION 1	OPTION 2	OPTION 3	OPTION 4	ANSWER
1	What's the point of the added bias term in models that use linear combinations of inputs?	Due to the bias-variance tradeoff, it helps reduce overfitting	There is no point, it's just convention at this point.	In logistic/linear regression, it helps shape the curvature of the model, and centers it at the	Aids in a model's ability to fit the data (e.g affine transformations)	Aids in a model's ability to fit the data (e.g affine transformations)
2	Which of the following is false regarding linear regression?	Scaling the features (not bias) by a constant has NO effect on an unregularized	Mean centering the features (not bias) has NO effect on an unregularized model's prediction	Standardization on the data has an effect on the unregularized model's prediction	Scaling the features (not bias) by a constant has NO effect on a regularized model's prediction	Scaling the features (not bias) by a constant has NO effect on a regularized model's prediction
3	Which is true regarding fitting a linear regression model when we're dealing with millions of samples?	We can utilize the normal equations since matrix inversions are computationally	Gradient descent is a more viable solution as opposed to solving the normal equations	Numerical computer scientist implement standard matrix inversion to	We can utilize the normal equations since vector inversions are computationally easy	Gradient descent is a more viable solution as opposed to solving the normal equations

4	How are logistic regression and linear regression related?	The logistic regression model utilizes a polynomial/higher-order model. This gives it the	Logistic regression inputs the linear regression model through a logarithmic	Since the logit function ranges from 0 to 1, just like the linear regression model, the	The logit function ranges from negative infinity to positive infinity. Equating it to the linear regression model and inverting it	**The logit function ranges from negative infinity to positive infinity. Equating it to the linear regression model and**
5	Logistic regression is often used when:	the dependent variable is categorical.	the underlying distribution for p(y|x) is Gaussian.	the dependent variable is continuous.	we need a closed form analytical solution.	**the dependent variable is categorical.**
6	Is the logistic regression model a Generalized Linear Model (GLM)? What is a possible reason?	No, it is not possible since the sigmoid shape of the function is highly non-	Yes, it's a GLM but only by convention. It's properties don't match with the current definition	Yes, it's a GLM since it's parameterized by a linear combination of its features, and	No, it's not a GLM since it belongs to the 'exponential' family of distributions and not the 'linear' family of distributions.	**Yes, it's a GLM since it's parameterized by a linear combination of its features, and shares a linear relationship with**
7	Which of the following is false regarding logistic regression?	We can solve the maximum log-likelihood problem via an analytical closed form solution,	We can solve the maximum log-likelihood problem via gradient descent.	We can solve the maximum log-likelihood problem via the Raphson-Newton Method	The conditional probability is modeled as a Bernoulli distribution.	**We can solve the maximum log-likelihood problem via an analytical closed form solution, though it is a bit slow.**
8	A regression model in which more than one independent variable is used to predict the dependent variable is called	a simple linear regression model	a multiple regression models	an independent model	none of the above	an independent model

9	A term used to describe the case when the independent variables in a multiple regression model are correlated is	regression	correlation	multicollinearity	none of the above	multicollinearity
10	A multiple regression model has	only one independent variable	more than one dependent variable	more than one independent variable	none of the above	more than one dependent variable
11	_____ is a systematic method for adding and removing terms from a multilinear model based on their statistical significance in a regression.	more than one dependent variable	more than one dependent variable	Stepwise regression	none of the above	Stepwise regression
12	_____ is used when _____	The entropy function	The squared error	Linear regression	Multinomial logistic regression	Multinomial logistic regression
13	most commonly used algorithm for solving all classification problems. It is also one of the first methods people get their hands dirty on.	Linear regression	Logistic Regression	Multinomial logistic regression	Polynomial Regression	Logistic Regression

14	_____ is a is a *universal* approximator so it can implement linear regression algorithm.	Neural network	Linear regression	Logistic Regression	Multinomial logistic regression	Neural network
15	Which of the following methods do we use to best fit the data in Logistic Regression?	Least Square Error	Maximum Likelihood	Jaccard distance	Both A and B	Maximum Likelihood
16	Which of the following evaluation metrics can not be applied in case of logistic regression output to compare with target?	AUC-ROC	Accuracy	Logloss	Mean-Squared-Error	Mean-Squared-Error
17	to analyze the performance of Logistic Regression is AIC, which is similar to R-Squared in Linear Regression. Which of the following is true about AIC?	We prefer a model with minimum AIC value	We prefer a model with maximum AIC value	Both but depend on the situation	None of these	We prefer a model with minimum AIC value
18	Which of the following algorithms do we use for Variable Selection?	LASSO	Ridge	Both	None of these	LASSO

19	Which of the following option is true?	Regression errors values has to be normally distributed but in case of Logistic Regression it is	Regression errors values has to be normally distributed but in case of Linear Regression it is	Regression and Logistic Regression error values have to be normally distributed	Both Linear Regression and Logistic Regression error values have not to be normally distributed	Linear Regression errors values has to be normally distributed but in case of Logistic Regression it is not the case
20	_____ is used to estimate / predict the discrete valued output such as success or failure, 0 or 1 etc.	The entropy function	Linear regression	Logistic regression	Polynomial Regression	Logistic regression
21	How many types in logistic regression	4	3	2	4	3
22	_____ is used to estimate the likelihood of outcome dependent variable instead of actual value as like linear regression model	The entropy function	Linear regression	Logistic regression	Polynomial Regression	Logistic regression
23	Logistic regression is used to predict _____ valued output?	Continuous	Categorical	Discrete	Ordinal	**Discrete**

24	How much marks a strudent can get in a competitive exam based on hours of study can be solved using _____ regression model	Multi-linear	Linear	Logistic	Ploynomial	
25	Logistic regression is _____ when the observed outcome of dependent variable can have only two values such as 0 and 1 or success and failure.	Binomial	Multinomial	Ordinal	Discreste	**Binomial**
26	Whether a strudent will pass or fail in the competitive exam based on hours of study can be solved using _____ regression model.	Multi-linear	Logistic	Linear	Polynomial	**-Logistic**
27	_____ regression can be termed as a special case of _____ regression when the outcome variable is categorical.	Logistic,Linear	Linear	Linear	Logistic	**Logistic,Linear**
28	In logistic regression	the goal is to predict _____	Actual value of outcome dependent variable	Odds of outcome dependent variable	agd	**Odds of outcome dependent variable**

29	Which of the following can be used to evaluate the performance of logistic regression model?	Adjusted R-Squared	AIC	Entropy	ROC	**AIC**
30	Which of the following is link function in logistic regression	Identity	Logit	Error		**Logit**
31	Logistic regression is _____ when the observed outcome of dependent variable can have multiple possible types	Binomial	Multinomial	Ordinal	Cardinal	**Multinomial**
32	In logistic regression	following technique is used to measure the goodness of the fit.	Sum of squares calculations	Deviance calculations	The Entropy function	**Deviance calculations**
33	Which of the following can be used to evaluate the performance of logistic regression model?.AIC	Null and Residual Deviance	Both of the above	Null Deviance only	Null Deviance Only	**Both of the above**

34	Given two model with different AIC value	which one would be preferred model?	One with higher AIC value	One with lower AIC value	Dependent variable equalling a given case	**One with lower AIC value**
35	Deviance is a measure of difference between a _____ model and the _____ model.	saturated,fitted	fitted	Fitted,saturated	saturated	**Fitted,saturated**
36	Logistic regression is _____ when the observed outcome of dependent variable are ordered.	Binomial	Multinomial	Ordinal	sum of squares calculations	**Ordinal**
37	Logit transformation is log of _____.	Odds of the event happening for different levels of each independent variable	Ratio of odds of the event happening for different levels of each independent variable	Deviance,sum of squares calculations	Dependent variable equalling a given case	**Ratio of odds of the event happening for different levels of each independent variable**
38	Logistic function is _____.	Dependent variable equalling a given case	Probability that dependent variable equals a case	Deviance,sum of squares calculations	Exponential function of the linear regression function	**Probability that dependent variable equals a case**

39	Deviance is is a function of _____.	Exponential function of likelihood ratio	Logrithmic function of likelihood ratio	Exponential function of the linear regression function	Maximum likelihood estimation method	**Logrithmic function of likelihood ratio**
40	The odds of the dependent variable equaling a case (given some linear combination x of the predictors) is equivalent to _____	Log function of the linear regression expression	Exponential function of the linear regression function	sum of squares calculations	Maximum likelihood estimation method	**Exponential function of the linear regression function**
41	Regression coefficients in logistic regression are estimated using _____.	Ordinary least squares method	Maximum likelihood estimation method	Dependent variable equalling a given case	sum of squares calculations	**Maximum likelihood estimation method**
42	_____ is analogous to _____ in linear regression.	Sum of squares calculations,deviance	deviance	Deviance,sum of squares calculations	sum of squares calculations	**Deviance,sum of squares calculations**
43	Deviance can be shown to follow _____.	t-distribution	F-distribution	Chi-square distribution	None of the above	**Chi-square distribution**

44	_____ value of deviance represents the better fit of model.	Higher	Lower	Smaller	Very Lower	**Lower**
45	If the model deviance is significantly _____ than the null deviance then one can conclude that the predictor or set of predictors significantly improved model fit.	Smaller	Larger	Moderate	Higher	**Smaller**
46	Which of the following is analogous to R-Squared for logistic regression.	Likelihood ration R-squared	McFadden R-squared	Cox and Snell R-Squared	All of the above	**All of the above**
47	Estimation in logistic regression chooses the parameters that _____ the likelihood of observing the sample values.	Minimizes	Maximizes	Higher	Lower	**Maximizes**
48	Which of the following tests can be used to assess whether the logistic regression model is well calibrated.	Hosmer-Lemeshow test	ROC Curve	Both of the above	Regression Operating Characteristic	**Hosmer-Lemeshow test**

49	ROC related with ROC curve stands for _____.	Regression Optimization Characteristic	Regression Operating Characteristic	Receiver Operating Characteristic	Regression Operating Characteristic	**Receiver Operating Characteristic**
50	Which of the following is used to identify the best threshold for separating positive and negative classes.	Hosmer-Lemeshow test	ROC Curve	Both of the above	AIC	**ROC Curve**
51	ROC curve is a plot of _____ vs _____.	Sensitivity,1-specificity	1-specificity	1-specificity	Sensitivity	**Sensitivity,1-specificity**
52	_____ the value of AUC,better is the prediction power of the model.	Moderate	Lower	Higher	Very Higher	**Higher**

Unit-IV

Regularization: Regularization and its utility: The problem of Overfitting, Application of Regularization in Linear and Logistic Regression, Regularization and Bias/Variance.

Regularization

Regularization is used to prevent the model from overfitting the training sample. It is basically a parameter that is used by minimizing the error function. A very good introduction to **regularization** is given by Prof Andrew NG in his **machine learning** class tutorial under the section "Linear and Logistic Regression".

The Problem of overfitting regularization

A common issue in machine learning or mathematical modeling is overfitting, which occurs when you build a model that not only captures the signal but also the noise in a dataset.

Because we want to create models that generalize and perform well on different data-points, we need to avoid overfitting.

In comes regularization, which is a powerful mathematical tool for reducing overfitting within our model. It does this by adding a penalty for model complexity or extreme parameter values, and it can be applied to different learning models: linear regression, logistic regression, and support vector machines to name a few.

Below is the linear regression cost function with an added regularization component.

$$J(\theta) = \frac{1}{2m}[\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})^2 + \underbrace{\lambda \sum_{j=1}^{n} \theta_j^2}_{\text{Regularization}}]$$

The regularization component is really just the sum of squared coefficients of your model (your beta values), multiplied by a parameter, lambda.

Lambda

Lambda can be adjusted to help you find a good fit for your model. However, a value that is too low might not do anything, and one that is too high might actually cause you to underfit the model and lose valuable information. It's up to the user to find the sweet spot.

Cross validation using different values of lambda can help you to identify the optimal lambda that produces the lowest out of sample error.

Regularization methods (L1 & L2)

The equation shown above is called Ridge Regression (L2) - the beta coefficients are squared and summed. However, another regularization method is Lasso Regreesion (L1), which sums the absolute value of the beta coefficients. Even more, you can combine Ridge and Lasso linearly to get Elastic Net Regression (both squared and absolute value components are included in the cost function).

L2 regularization tends to yield a "dense" solution, where the magnitude of the coefficients are evenly reduced. For example, for a model with 3 parameters, B1, B2, and B3 will reduce by a similar factor.

However, with L1 regularization, the shrinkage of the parameters may be uneven, driving the value of some coefficients to 0. In other words, it will produce a sparse solution. Because of this property, it is often used for feature selection- it can help identify the most predictive features, while zeroing the others.

It also a good idea to appropriately scale your features, so that your coefficients are penalized based on their predictive power and not their scale.

As you can see, regularization can be a powerful tool for reducing overfitting.

In the words of the great thinkers:

- "when you have two competing theories that make exactly the same predictions, the simpler one is the better." - William of Ockham
- "Everything should be made as simple as possible, but not simpler." - Albert Einstein

An in-depth look into theory and application of regularization.

Feature Scaling

The scikit-learn module makes it surprisingly easy to implement a wide range of machine learning algoirthms in Python. Often times, the only parameters you need to specify are your model and data, and scikit-learn will do the rest.

But before putting your raw data into the model, it's important that it's in the right format. This means ensuring that your predictor variables are numerical and properly scaled. This post will focus on properly scaling your features.

Scaling methods

Standard scale

Features can be scaled to have mean 0 and variance 1, which is also known as calculating Z-scores for every observation. This will usually transform the data so that most of your data falls between [-3, 3].

Min-max scale

Another way of scaling is to subtract each value by the minimum value and divide by the range. This is called min-max scaling and it transforms your feature to a scale of [0, 1]. For better or worse, this scaling technique does not change the distribution of your data.

scikit-learn implementations

Scikit has easy implementations of scaling using its preprocessing module.

sklearn**.**preprocessing**.**scale(X)

X can be an array or matrix, by default normalizes to mean 0 and variance 1.

If you want to consider scaling as part of your modeling step rather than your preprocessing step, you can fit your scalar to your training set and apply these same parameters to scaling your test set.

```
std_scalar = sklearn.preprocessing.StandardScalar()
std_scalar.fit(X_train)
X_train_std = std_scalar.transform(X_train)
X_test_std = std_scalar.transform(X_test)
std_scalar = sklearn.preprocessing.MinMaxScalar()
std_scalar.fit(X_train)
X_train_std = std_scalar.transform(X_train)
X_test_std = std_scalar.transform(X_test)
```

Normalizing to standard normal scale has a big impact on the quality of some models (SVM, kNN), a small impact on others (logistic regression, decision trees), and no impact on Naive Bayes .

Overfitting and regularization

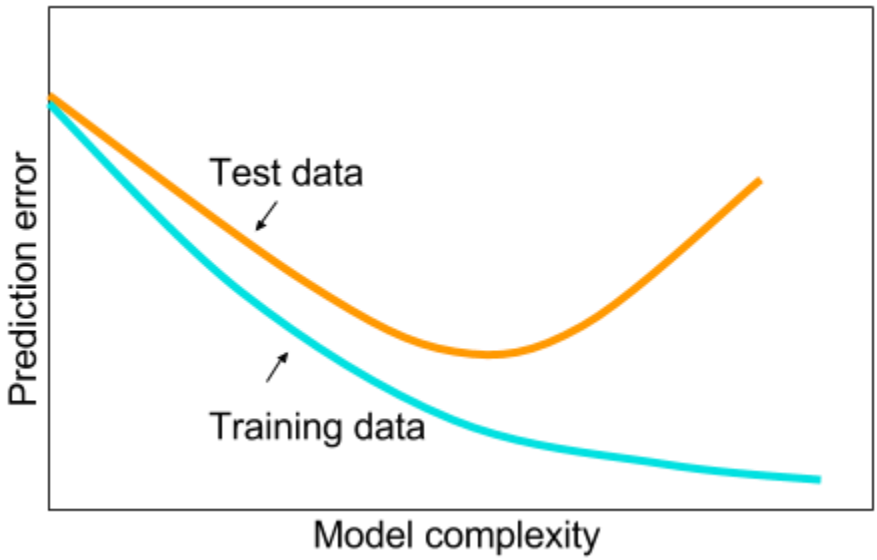
. We showed how you can tackle this problem with a linear model called logistic regression. Owing to some amount of randomness, you might get slightly different results, but when I ran the notebook, the model achieved 88.1% accuracy on the training data and actually did slightly (but not significantly) better on the test data than on the training data.

Not every algorithm that performs well on training data will also perform well on test data. Take, for example, a trivial algorithm that memorizes its inputs and stores the associated labels. This model would have 100% accuracy on

training data but would have no way of making any prediction at all on previously unseen data.

The goal of supervised learning is to produce models that *generalize* to previously unseen data. When a model achieves low error on training data but performs much worse on test data, we say that the model has *overfit*. This means that the model has caught on to idiosyncratic features of the training data (e.g. one "2" happened to have a white pixel in the top-right corner), but hasn't really picked up on general patterns.

We can express this more formally. The quantity we really care about is the test error ee. Because this quantity reflects the error of our model when generalized to previously unseen data, we commonly call it the *generalization error*. When we have simple models and abundant data, we expect the generalization error to resemble the training error. When we work with more complex models and fewer examples, we expect the training error to go down but the generalization gap to grow. Fixing the size of the dataset, the following graph should give you some intuition about what we generally expect to see.



What precisely constitutes model complexity is a complex matter. Many factors govern whether a model will generalize well. For example a model with more parameters might be considered more complex. A model whose parameters can take a wider range of values might be more complex. Often with neural networks, we think of a model that takes more training steps as more complex, and one subject to *early stopping* as fewer complexes.

It can be difficult to compare the complexity among members of very different model classes (say decision trees versus neural networks). Researchers in the field of statistical learning theory have developed a large body of mathematical analysis that formulizes the notion of model complexity and provides guarantees on the generalization error for simple classes of models. *We won't get into this theory but may delve deeper in a future chapter*. For now a simple rule of thumb is quite useful: A model that can readily explain *arbitrary* facts is what statisticians view as complex, whereas one that has only a limited

expressive power but still manages to explain the data well is probably closer to the truth. In philosophy this is closely related to Popper's criterion of falsifiability of a scientific theory: a theory is good if it fits data and if there are specific tests which can be used to disprove it. This is important since all statistical estimation is post hoc, i.e. we estimate after we observe the facts, hence vulnerable to the associated fallacy. Ok, enough of philosophy, let's get to more tangible issues.

To give you some intuition in this chapter, we'll focus on a few factors that tend to influence the generalizability of a model class:

1. **The number of tunable parameters.** When the number of tunable parameters, sometimes denoted as the number of degrees of freedom, is large, models tend to be more susceptible to overfitting.
2. **The values taken by the parameters.** When weights can take a wider range of values, models can be more susceptible to over fitting.
3. **The number of training examples.** It's trivially easy to overfit a dataset containing only one or two examples even if your model is simple. But overfitting a dataset with millions of examples requires an extremely flexible model.

When classifying handwritten digits before, we didn't overfit because our 60,000 training examples far out numbered the 784×10=7,840784×10=7,840 weights plus 1010 bias terms, which gave us far fewer parameters than training examples. Let's see how things can go wrong. We begin with our import ritual.

In [ ]:

```
from __future__ import print_function
import mxnet as mx
import mxnet.ndarray as nd
from mxnet import autograd
import numpy as np
ctx = mx.cpu()
mx.random.seed(1)
```

```
# for plotting purposes
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
```

Load the MNIST dataset

In [ ]:

```
mnist = mx.test_utils.get_mnist()

num_examples = 1000

batch_size = 64

train_data = mx.gluon.data.DataLoader(

    mx.gluon.data.ArrayDataset(mnist["train_data"][:num_examples],

                               mnist["train_label"][:num_examples].astype(np.float32)),

                               batch_size, shuffle=True)

test_data = mx.gluon.data.DataLoader(

    mx.gluon.data.ArrayDataset(mnist["test_data"][:num_examples],

                               mnist["test_label"][:num_examples].astype(np.float32)),

                               batch_size, shuffle=False)
```

Allocate model parameters and define model

We pick a simple linear model f(x)=Wx+bf(x)=Wx+b with subsequent softmax, i.e. p(y|x)∝exp(f(x)y)p(y|x)∝exp⁡(f(x)y). This is about as simple as it gets.

In [ ]:

```
W = nd.random_normal(shape=(784,10))
```

```
b = nd.random_normal(shape=10)

params = [W, b]

for param in params:

    param.attach_grad()

def net(X):

    y_linear = nd.dot(X, W) + b

    yhat = nd.softmax(y_linear, axis=1)

    return yhat
```

Define loss function and optimizer

A sensible thing to do is to minimize the negative log-likelihood of the data, i.e. $-\log p(y|x) - \log p(y|x)$. Statisticians have proven that this is actually the most *efficient* estimator, i.e. the one that makes the most use of the data provided. This is why it is so popular.

In [ ]:

```
def cross_entropy(yhat, y):

    return - nd.sum(y * nd.log(yhat), axis=0, exclude=True)
```

```
def SGD(params, lr):
    for param in params:
        param[:] = param - lr * param.grad
```

Write evaluation loop to calculate accuracy

Ultimately we want to recognize digits. This is a bit different from knowing the *probability* of a digit - when given an image we need to *decide* what digit we are seeing, *regardless* of how uncertain we are. Hence we measure the number of actual misclassifications.

For diagnosis purposes, it is always a good idea to calculate the average loss function.

In [ ]:

```
def evaluate_accuracy(data_iterator, net):
    numerator = 0.
    denominator = 0.
    loss_avg = 0.
    for i, (data, label) in enumerate(data_iterator):
        data = data.as_in_context(ctx).reshape((-1,784))
```

```
        label = label.as_in_context(ctx)
        label_one_hot = nd.one_hot(label, 10)
        output = net(data)
        loss = cross_entropy(output, label_one_hot)
        predictions = nd.argmax(output, axis=1)
        numerator += nd.sum(predictions == label)
        denominator += data.shape[0]
        loss_avg = loss_avg*i/(i+1) + nd.mean(loss).asscalar()/(i+1)
    return (numerator / denominator).asscalar(), loss_avg
```

Write a utility function to plot the learning curves

Just to visualize how loss functions and accuracy changes over the number of iterations.

In [ ]:

```
def plot_learningcurves(loss_tr,loss_ts, acc_tr,acc_ts):
    xs = list(range(len(loss_tr)))
```

```
f = plt.figure(figsize=(12,6))

fg1 = f.add_subplot(121)

fg2 = f.add_subplot(122)

fg1.set_xlabel('epoch',fontsize=14)

fg1.set_title('Comparing loss functions')

fg1.semilogy(xs, loss_tr)

fg1.semilogy(xs, loss_ts)

fg1.grid(True,which="both")

fg1.legend(['training loss', 'testing loss'],fontsize=14)

fg2.set_title('Comparing accuracy')

fg1.set_xlabel('epoch',fontsize=14)

fg2.plot(xs, acc_tr)

fg2.plot(xs, acc_ts)

fg2.grid(True,which="both")

fg2.legend(['training accuracy', 'testing accuracy'],fontsize=14)
```

Execute training loop

We now train the model until there is no further improvement. Our approach is actually a bit naive since we will keep the learning rate unchanged but it fits the purpose (we want to keep the code simple and avoid confusing anyone with further tricks for adjusting learning rate schedules).

In [ ]:

```
epochs = 1000

moving_loss = 0.

niter=0

loss_seq_train = []

loss_seq_test = []

acc_seq_train = []

acc_seq_test = []

for e in range(epochs):

    for i, (data, label) in enumerate(train_data):

        data = data.as_in_context(ctx).reshape((-1,784))

        label = label.as_in_context(ctx)

        label_one_hot = nd.one_hot(label, 10)
```

```
    with autograd.record():
        output = net(data)
        loss = cross_entropy(output, label_one_hot)
    loss.backward()
    SGD(params, .001)
    ##########################
    #  Keep a moving average of the losses
    ##########################
    niter +=1
    moving_loss = .99 * moving_loss + .01 * nd.mean(loss).asscalar()
    est_loss = moving_loss/(1-0.99**niter)


test_accuracy, test_loss = evaluate_accuracy(test_data, net)
train_accuracy, train_loss = evaluate_accuracy(train_data, net)
# save them for later
loss_seq_train.append(train_loss)
```

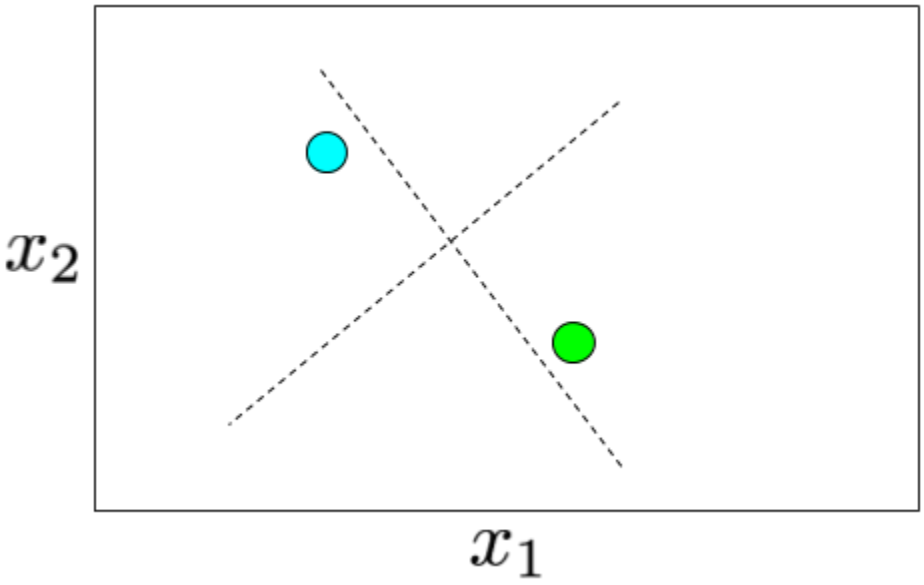
```
    loss_seq_test.append(test_loss)

    acc_seq_train.append(train_accuracy)

    acc_seq_test.append(test_accuracy)

    if e % 100 == 99:

        print("Completed epoch %s. Train Loss: %s, Test Loss %s, Train_acc %s,
Test_acc %s" %

              (e+1, train_loss, test_loss, train_accuracy, test_accuracy)

## Plotting the learning curves

plot_learningcurves(loss_seq_train,loss_seq_test,acc_seq_train,acc_seq_test)
```

What Happened?

By the 700th epoch, our model achieves 100% accuracy on the training data. However, it only classifies 75% of the test examples accurately. This is a clear case of overfitting. At a high level, there's a reason this went wrong. Because we have 7450 parameters and only 1000 data points, there are actually many settings of the parameters that could produce 100% accuracy on training data.

To get some intuition imagine that we wanted to fit a dataset with 2 dimensional data and 2 data points. Our model has three degrees of freedom, and thus for any dataset can find an arbitrary number of separators that will

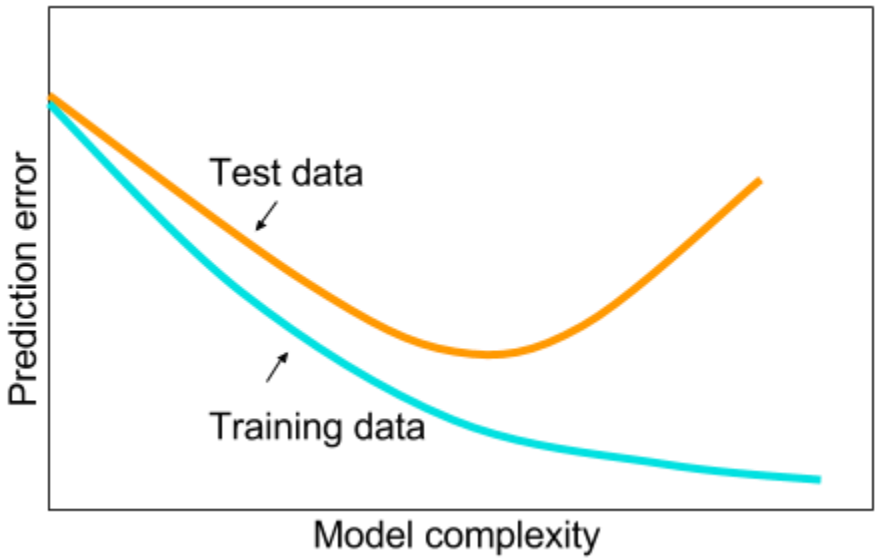
perfectly classify our training points. Note below that we can produce completely orthogonal separators that both classify our training data perfectly. Even if it seems preposterous that they could both describe our training data well.



Regularization

Now that we've characterized the problem of overfitting, we can begin talking about some solutions. Broadly speaking the family of techniques geared towards mitigating overfitting are referred to as *regularization*. The core idea is this: when a model is overfitting, its training error is substantially lower than its test error. We're already doing as well as we possibly can on the training data, but our test data performance leaves something to be desired. Typically, regularization techniques attempt to trade off our training performance in exchange for lowering our test error.

There are several straightforward techniques we might employ. Given the intuition from the previous chart, we might attempt to make our model less complex. One way to do this would be to lower the number of free parameters. For example, we could throw away some subset of our input features (and thus the corresponding parameters) that we thought were least informative.



Another approach is to limit the values that our weights might take. One common approach is to force the weights to take small values. [give more intuition with example of polynomial curve fitting] We can accomplish this by changing our optimization objective to penalize the value of our weights. The most popular regularizer is the ℓ22ℓ22 norm. For linear models, ℓ22ℓ22 regularization has the additional benefit that it makes the solution unique, even when our model is overparametrized.

∑i(y^−y)2+λ‖w‖22∑i(y^−y)2+λ‖w‖22

Here, ‖w‖‖w‖ is the ℓ22ℓ22 norm and λλ is a hyper-parameter that determines how aggressively we want to push the weights towards 0. In code, we can express the ℓ22ℓ22 penalty succinctly:

In [ ]:

```
def l2_penalty(params):
    penalty = nd.zeros(shape=1)
    for param in params:
        penalty = penalty + nd.sum(param ** 2)
    return penalty
```

Re-initializing the parameters

Just for good measure to ensure that the results in the second training run don't depend on the first one.

In [ ]:

```
for param in params:
    param[:] = nd.random_normal(shape=param.shape)
```

Training L2-regularized logistic regression

In [ ]:

```
epochs = 1000

moving_loss = 0.

l2_strength = .1

niter=0



loss_seq_train = []

loss_seq_test = []

acc_seq_train = []

acc_seq_test = []




for e in range(epochs):

    for i, (data, label) in enumerate(train_data):

        data = data.as_in_context(ctx).reshape((-1,784))

        label = label.as_in_context(ctx)

        label_one_hot = nd.one_hot(label, 10)
```

```
        with autograd.record():
            output = net(data)
            loss = nd.sum(cross_entropy(output, label_one_hot)) + l2_strength *
l2_penalty(params)
        loss.backward()
        SGD(params, .001)


        ##########################
        #  Keep a moving average of the losses
        ##########################
        niter +=1
        moving_loss = .99 * moving_loss + .01 * nd.mean(loss).asscalar()
        est_loss = moving_loss/(1-0.99**niter)



    test_accuracy, test_loss = evaluate_accuracy(test_data, net)
```

```
    train_accuracy, train_loss = evaluate_accuracy(train_data, net)


    # save them for later
    loss_seq_train.append(train_loss)
    loss_seq_test.append(test_loss)
    acc_seq_train.append(train_accuracy)
    acc_seq_test.append(test_accuracy)


    if e % 100 == 99:
        print("Completed epoch %s. Train Loss: %s, Test Loss %s, Train_acc %s,
Test_acc %s" %
              (e+1, train_loss, test_loss, train_accuracy, test_accuracy))
```

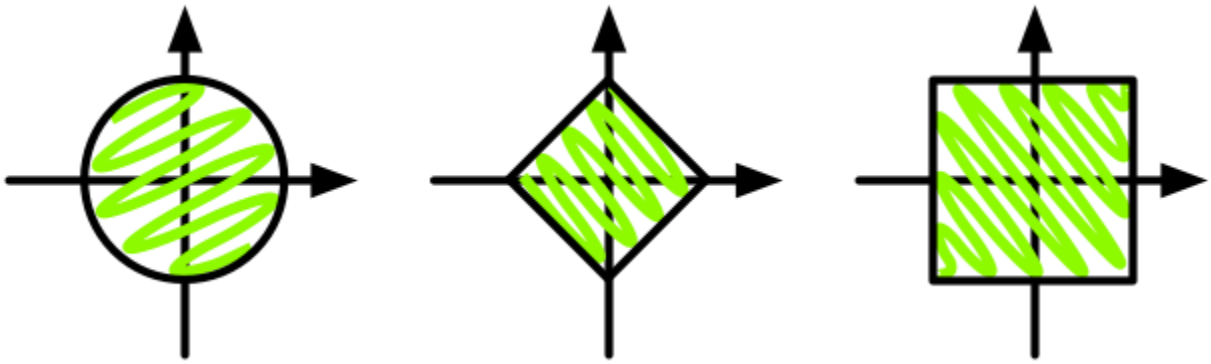
In [ ]:

```
## Plotting the learning curves
plot_learningcurves(loss_seq_train,loss_seq_test,acc_seq_train,acc_seq_test)
```

Analysis

By adding L2L2 regularization we were able to increase the performance on test data from 75% accuracy to 83% accuracy. That's a 32% reduction in error. In a lot of applications, this big an improvement can make the difference between a viable product and useless system. Note that L2 regularization is just one of many ways of controlling capacity. Basically we assumed that small weight values are good. But there are many more ways to constrain the values of the weights:

- We could require that the total sum of the weights is small. That is what L1L1 regularization does via the penalty ∑i|wi|∑i|wi|.
- We could require that the largest weight is not too large. This is what L∞L∞ regularization does via the penalty maxi|wi|maxi|wi|.
- We could require that the number of nonzero weights is small, i.e. that the weight vectors are *sparse*. This is what the L0L0 penalty does, i.e. ∑iI{wi≠0}∑iI{wi≠0}. This penalty is quite difficult to deal with explicitly since it is nonsmooth. There is a lot of research that shows how to solve this problem approximately using an L1L1 penalty.



From left to right: L2L2 regularization, which constrains the parameters to a ball, L1L1 regularization, which constrains the parameters to a diamond (for lack of a better name, this is often referred to as an L1L1-ball), and L∞L∞ regularization, which constrains the parameters to a hypercube.

All of this raises the question of **why** regularization is any good. After all, choice is good and giving our model more flexibility *ought* to be better (e.g. there are plenty of papers which show improvements on ImageNet using deeper networks). What is happening is somewhat more subtle. Allowing for many different parameter values allows our model to cherry pick a combination that is *just right* for all the training data it sees, without really learning the underlying mechanism. Since our observations are likely noisy, this means that we are trying to approximate the errors at least as much as we're learning what the relation between data and labels actually is. There is an entire field of statistics devoted to this issue - Statistical Learning Theory. For now, a few simple rules of thumb suffice:

- Fewer parameters tend to be better than more parameters.

- Better engineering for a specific problem that takes the actual problem into account will lead to better models, due to the prior knowledge that data scientists have about the problem at hand.
- L2L2 is easier to optimize for than L1L1. In particular, many optimizers will not work well out of the box for L1L1. Using the latter requires something called *proximal operators*.
- Dropout and other methods to make the model robust to perturbations in the data often work better than off-the-shelf L2L2 regularization.

We conclude with an XKCD Cartoon which captures the entire situation more succinctly than the preceding paragraph.

Applications :

Bayesian learning methods make use of a prior probability that (usually) gives lower probability to more complex models. Well-known model selection techniques include the Akaike information criterion (AIC), minimum description length (MDL), and the Bayesian information criterion (BIC). Alternative methods of controlling overfitting not involving regularization include cross-validation.

Examples of applications of different methods of regularization to the linear model are:

Model
AIC/BIC
Ridge regression[5]
Lasso[6]
Basis pursuit denoising
Rudin–Osher–Fatemi model (TV)
Potts model
RLAD[7]
Dantzig Selector[8]
SLOPE[9]

Regularization and Bias/Variance

- Regularization can avoid underfitting/overfitting. But how it does acttually affect the learning algorithms
- Remember the regularization indexes from 1
- Set lambda = 1000, and each parameters will be highly penalized and will be tend to flat graph, resulting to underfitting
- In contrast, set lambda to 0, the parameters will not be penalized and resulting in overfitting problems
- So how we choose the correct value of regularization (lambda)?
- Using extra lambda, just using average of the training set

- Jtrain, Jcv,Jtest in earlier without the regularization

- Try variant range of lambda by multiple sets of two
- Iterate each model of theta use it to cost function
- Use the theta into cross validation set
- Pick whichever model that has the lowest value in cross validation error
- And compare it to Jtest error
- Concretely, use the selection model of thetas with selection of lambda,(model 5 with lambda no.5), and pick whichever has the lowest error of Jcv
- That's the summary of model selection for regularization
- These show how variance/bias vary based on variant of regularization parameter
- lambda small, regularization not being used == overfitting
- lambda high, regularization highly used == underfitting
- These are the example where the higher the lambda, making more underfitting, that is the cost function of Jtheta in training set is higher
- In choosing lambda, often plotting the graph …. making a better intuition of choosing the right lambda
- Bias and variance by now is seen from a lot of different perspective
- Next learning curve as a tool to indentify whether the learning algorithm has bias/variance problem.

Part-B (2marks)

1. Define Regularization.
2. Write short notes on regularization utility.
3. Define Overfitting.
4. What is Bias/Variance?
5. Write brief notes on overfitting problems.

Part-C (6 Marks)

1. Explain in detail about Regularization.
2. Discuss about Bias/Variance with example.
3. Briefly explain about Regularization in Linear Regression.
4. Illustrate on Application of Regularization in Logistic Regression
5. Describe about problem of overfitting in machine learning.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc(CS) (BATCH 2016-2019)

Machine Learning

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

QUESTIONS	OPTION1	OPTION2	OPTION3	OPTION4	ANSWER
_____ is a is a *universal* approximator so it can implement linear regression algorithm.	Neural network	Linear regression	Logistic Regression	Multinomial logistic regression	Neural network

Which of the following methods do we use to best fit the data in Logistic Regression?	**Least Square Error**	**Maximum Likelihood**	**Jaccard distance**	**Both A and B**	**Maximum Likelihood**
A _____ can only contain objects of the same class.	**list**	**vector**	**data frames**	**factor**	**vector**
>m <- matrix(nrow = 2, ncol = 3) >m > attributes(m)	**2 3**	**3 2**	**dim**	**NA**	**dim**
_____ function to find the data type of the variable	**datatype()**	**class()**	**type()**	**cls()**	**class()**

The _____ Function get the current working directory	**get()**	**getwd()**	**getw()**	**wd()**	**getwd()**
Which function is used to transpose data frame?	**t()**	**ti()**	**transpose()**	**trans()**	**t()**
Vectors come in two parts:_____ and _____.	**atomic vectors and matrix**	**atomic vectos and array**	**atomic vectors and list**	**atomic vectors and functions**	**atomic vectors and list**
Which of the following is a base package for R language ?	**util**	**lang**	**tools**	**math**	**tools**

Which of the following is used for reading in saved workspaces ?	**unserialize**	**load**	**get**	**read**	**load**
The _____ function can be useful for reading in lines of webpages	**Load()**	**readLines()**	**read()**	**readpage()**	**readLines()**
Text files can be read line by line using the _____function.	**Load()**	**readpage()**	**read()**	**readLines()**	**readLines()**
The _____ package is recently developed by Hadley Wickham to deal with reading in large flat files quickly.	**readr**	**dplyr**	**read**	**dr**	**readr**

The _____ and _____ functions are useful because the resulting textual format is editable, and in the case of corruption, potentially recoverable.	**dump() and dget()**	**dump() and dput()**	**dget() and dput()**	**dump() and dp()**	**dump() and dput()**
_____ opens a connection to a file	**file**	**gzfile**	**bzfile**	**url**	**file**
_____ opens a connection to a file compressed with gzip	**file**	**gzfile**	**bzfile**	**url**	**gzfile**
_____ opens a connection to a file compressed with bzip2	**file**	**gzfile**	**bzfile**	**url**	**bzfile**

_____ opens a connection to a webpage	file	gzfile	bzfile	url	url
The ____ function has a number of arguments that are common to many other connection	f()	close()	file()	open()	file()
_____ open file in read only mode	“r”	“a”	“w”	"ab"	“r”
_____ open a file for writing (and initializing a new file)	“r”	“a”	“w”	"ab"	“w”

_____ open a file for appending	“r”	“a”	“w”	"ab"	“a”
The_____ operator can be used to extract multiple elements of a vector by passing the operator an integer sequence	$	[	[[	((	[
We can also create an empty list of a prespecified length with the_____function	create()	file()	vector()	list()	vector()
Which of the following is example of vectorized operation as far as subtraction is concerned ? > x <- 1:4 > y <- 6:9	x+y	x-y	x*y	x/y	x-y

In simulating linear model can also simulate from_____ where the errors are no longer from a Normal distribution but come from some other distribution.	generalized model	generalized linear model	linear model	ungeneralized linear model	generalized linear model
Simulating _____ numbers is useful but sometimes we want to simulate values that come from a specific model.	arbitrary	sample	random	sequence	random
The function call stack is the _____ of functions that was called before the error occurred.	arbitrary	sample	random	sequence	sequence
A nearest neighbor approach is best used	with large-sized datasets.	when irrelevant attributes have been removed from the data.	when a generalized model of the data is desireable.	when an explanation of what has been found is of primary importance.	when irrelevant attributes have been removed from the data.

If a customer is spending more than expected, the customer's intrinsic value is _____ their actual value.	**greater than**	**less than**	**less than or equal to**	**equal to**	**less than**
_____ can be any unprocessed fact, value, text, sound or picture that is not being interpreted and analyze	**data**	**knowledge**	**information**	**machine**	**data**
Database query is used to uncover this type of knowledge.	**deep**	**hidden**	**shallow**	**multidimensional**	**shallow**
A statement to be tested.	**theory**	**procedure**	**principle**	**hypothesis**	**hypothesis**

A person trained to interact with a human expert in order to capture their knowledge.	**knowledge programmer**	**knowledge developer**	**knowledge engineer**	**knowledge extractor**	**knowledge engineer**
Logistic regression is _____ when the observed outcome of dependent variable are ordered.	**Binomial**	**Multinomial**	**Ordinal**	**sum of squares calculations**	**Ordinal**
Logit transformation is log of _____.	**Odds of the event happening for different levels of each independent variable**	**Ratio of odds of the event happening for different levels of each independent variable**	**Deviance,sum of squares calculations**	**Dependent variable equalling a given case**	**Ratio of odds of the event happening for different levels of each independent variable**
Logistic regression is _____ when the observed outcome of dependent variable can have multiple possible types	**Binomial**	**Multinomial**	**Ordinal**	**Cardinal**	**Multinomial**

Regression coefficients in logistic regression are estimated using _____.	Ordinary least squares method	Maximum likelihood estimation method	Dependent variable equalling a given case	sum of squares calculations	Maximum likelihood estimation method
_____ acts as an outstanding tool for visulaizing technical data	C	C++	Java	MATLAB	MATLAB
The _____ command will display a list of possible help topics in the command window	help	helper	lookfor	order	help
The _____ function accepts an array arugment and displayus the value of athe array in the command window	disp	format	special	fprintf	disp

_____ are operations performed between arrays on an element by element basis	**matrix operations**	**array operations**	**vector operations**	**arthimetic operations**	**array operations**

Unit-V(Syllabus)

Neural Networks: Introduction, Model Representation, Gradient Descent vs. Perceptron Training, Stochastic Gradient Descent, Multilayer Perceptrons, Multiclass Representation, Backpropagation Algorithm.

Neural Networks for Machine Learning

Reasons to study neural computation

• To understand how the brain actually works. – Its very big and very complicated and made of stuff that dies when you poke it around. So we need to use computer simulations.

• To understand a style of parallel computation inspired by neurons and their adaptive connections. – Very different style from sequential computation.

• should be good for things that brains are good at (e.g. vision)

• Should be bad for things that brains are bad at (e.g. 23 x 71)

• To solve practical problems by using novel learning algorithms inspired by the brain (this course) – Learning algorithms can be very useful even if they are not how the brain actually works.

A typical cortical neuron

• Gross physical structure: – There is one axon that branches – There is a dendrites tree that collects input from other neurons.

• Axons typically contact dendrite trees at synapses – A spike of activity in the axon causes charge to be injected into the post-synaptic neuron.

• Spike generation: – There is an axon hillock that generates outgoing spikes whenever enough charge has flowed in at synapses to depolarize the cell membrane. axon body dendrites tree axon hillock

Neural network

The term neural network was traditionally used to refer to a network or circuit of biological neurons. The modern usage of the term often refers to artificial neural networks, which are composed of artificial neurons or nodes. Thus the term has two distinct usages:

1. Biological neural networks are made up of real biological neurons that are connected or functionally related in the peripheral nervous system or the central nervous system. In the field of neuroscience, they are often identified as groups of neurons that perform a specific physiological function in laboratory analysis.

2. Artificial neural networks are made up of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system. The real, biological nervous system is highly complex and includes some features that may seem superfluous based on an understanding of artificial networks.

Model-based machine learning

The central idea of the model-based approach to machine learning is to create a custom bespoke model tailored specifically to each new application. In some cases, the model (together with an associated inference algorithm) might correspond to a traditional machine learning technique, while in many cases it will not.

Typically, model-based machine learning will be implemented using a model specification language in which the model can be defined using compact code, from which the software implementing that model can be generated automatically.

The key goals of a model-based approach include the following —

The ability to create a very broad range of models, along with suitable inference or learning algorithms, in which many traditional machine learning techniques appear as special cases. —

Each specific model can be tuned to the individual requirements of the particular application: for example, if the application requires a combination of clustering and classification in the context of time-series data, it is not necessary to mash together traditional algorithms for each of these elements (Gaussian mixtures, neural networks and hidden Markov models (HMMs), for instance), but instead a single, integrated model capturing the desired behaviour can be constructed. —

Segregation between the model and the inference algorithm: if changes are made to the model, the corresponding modified inference software is created automatically. Equally, advances in techniques for efficient inference are available to a broad range of models. —

Transparency of functionality: the model is described by compact code within a generic modelling language, and so the structure of the model is readily apparent. Such modelling code can easily be shared and extended within a community of model builders. —

Pedagogy: newcomers to the field of machine learning have only to learn a single modelling environment in order to be able to access a wide range of modelling solutions. Because many traditional methods will be subsumed as special cases of the model-based environment, there is no need for newcomers to study these individually, or indeed to learn the specific terminology associated with them.

A variety of different approaches could be envisaged for achieving the aims of model-based machine learning. In this study, we focus on a powerful framework based on Bayesian inference in probabilistic graphical models, and so we begin with a brief introduction to the Bayesian view of machine learning.

Gradient descent

Gradient descent is a search strategy used in continuous search spaces. It is the basis for the error backpropagation algorithm used to train multilayer artificial neural networks. Below I explain how **gradient descent** works in general and how it applies in particular to **training** a single layer **perceptron** network.

Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).

Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.

Intuition for Gradient Descent

Think of a large bowl like what you would eat cereal out of or store fruit in. This bowl is a plot of the cost function (f).

Large Bowl

A random position on the surface of the bowl is the cost of the current values of the coefficients (cost).

The bottom of the bowl is the cost of the best set of coefficients, the minimum of the function.

The goal is to continue to try different values for the coefficients, evaluate their cost and select new coefficients that have a slightly better (lower) cost.

Repeating this process enough times will lead to the bottom of the bowl and you will know the values of the coefficients that result in the minimum cost.

What is Gradient Descent?

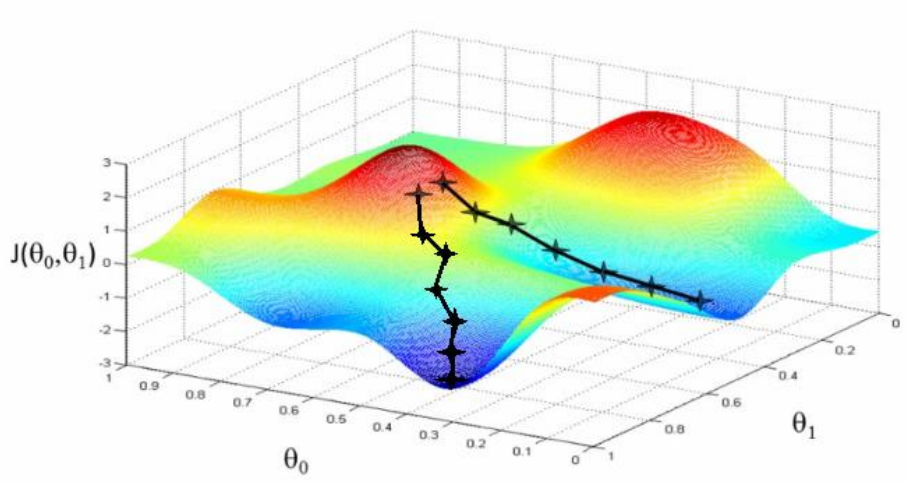
To explain Gradient Descent I'll use the classic mountaineering example.

Suppose you are at the top of a mountain, and you have to reach a lake which is at the lowest point of the mountain (a.k.a valley). A twist is that you are blindfolded and you have zero visibility to see where you are headed. So, what approach will you take to reach the lake?

Source

The best way is to check the ground near you and observe where the land tends to descend. This will give an idea in what direction you should take your first step. If you follow the descending path, it is very likely you would reach the lake.

To represent this graphically, notice the below graph.



Source

Let us now map this scenario in mathematical terms.

Suppose we want to find out the best parameters (θ1) and (θ2) for our learning algorithm. Similar to the analogy above, we see we find similar mountains and valleys when we plot our "cost space". Cost space is nothing but how our algorithm would perform when we choose a particular value for a parameter.

So on the y-axis, we have the cost J(θ) against our parameters θ1 and θ2 on x-axis and z-axis respectively. Here, hills are represented by red region, which have high cost, and valleys are represented by blue region, which have low cost.

Now there are many types of gradient descent algorithms. They can be classified by two methods mainly:

- **On the basis of data ingestion**

1. Full Batch Gradient Descent Algorithm
2. Stochastic Gradient Descent Algorithm

In full batch gradient descent algorithms, you use whole data at once to compute the gradient, whereas in stochastic you take a sample while computing the gradient.

- **On the basis of differentiation techniques**
  1. First order Differentiation
  2. Second order Differentiation

Gradient descent requires calculation of gradient by differentiation of cost function. We can either use first order differentiation or second order differentiation.

2. Challenges in executing Gradient Descent

Gradient Descent is a sound technique which works in most of the cases. But there are many cases where gradient descent does not work properly or fails to work altogether. There are three main reasons when this would happen:

1. Data challenges
2. Gradient challenges
3. Implementation challenges

2.1 Data Challenges

- If the data is arranged in a way that it poses a **non-convex optimization problem.** It is very difficult to perform optimization using gradient descent. Gradient descent only works for problems which have a well defined convex optimization problem.
- Even when optimizing a convex optimization problem, there may be numerous minimal points. The lowest point is called global minimum, whereas rest of the points are called local minima. Our aim is to go to global minimum while avoiding local minima.
- There is also a saddle point problem. This is a point in the data where the gradient is zero but is not an optimal point. We don't have a specific way to avoid this point and is still an active area of research.

2.2 Gradient Challenges

- If the execution is not done properly while using gradient descent, it may lead to problems like vanishing gradient or exploding gradient problems. These problems occur when the gradient is too small or too large. And because of this problem the algorithms do not converge.

2.3 Implementation Challenges

- Most of the neural network practitioners don't generally pay attention to implementation, but it's very important to look at the resource utilization by networks. For eg: When implementing gradient descent, it is very important to note how many resources you would require. If the memory is too small for your application, then the network would fail.

- Also, its important to keep track of things like floating point considerations and hardware / software prerequisites.

3. Variants of Gradient Descent algorithms

Let us look at most commonly used gradient descent algorithms and their implementations.

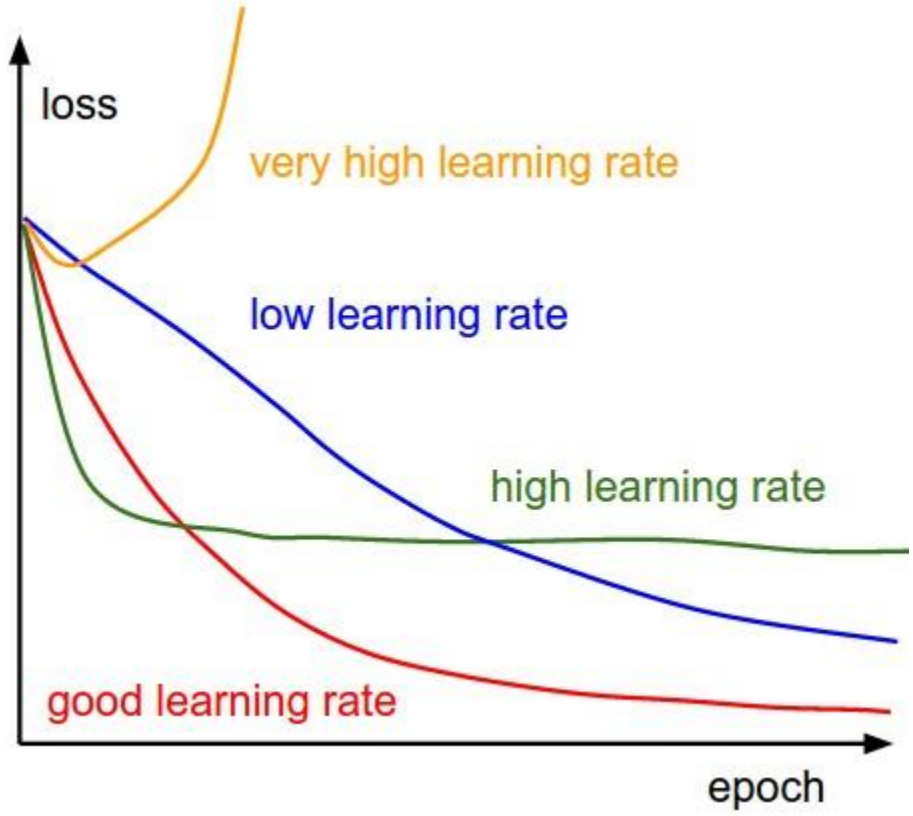
3.1 Vanilla Gradient Descent

This is the simplest form of gradient descent technique. Here, vanilla means pure / without any adulteration. Its main feature is that we take small steps in the direction of the minima by taking gradient of the cost function.

Let's look at its pseudocode.

```
update = learning_rate * gradient_of_parameters

parameters = parameters - update
```

Here, we see that we make an update to the parameters by taking gradient of the parameters. And multiplying it by a learning rate, which is essentially a constant number suggesting how fast we want to go the minimum. Learning rate is a hyper-parameter and should be treated with care when choosing its value.



3.2 Gradient Descent with Momentum

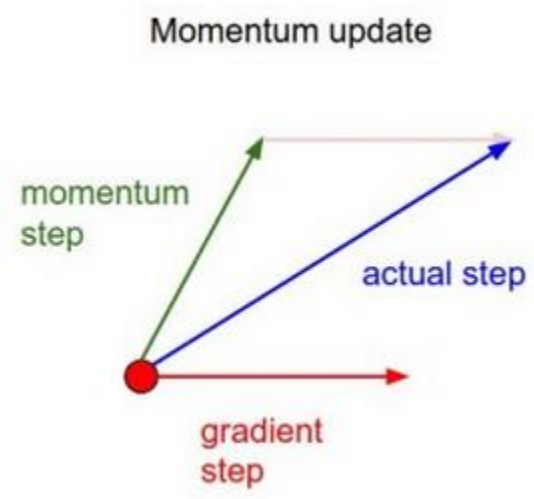
Here, we tweak the above algorithm in such a way that we pay heed to the prior step before taking the next step.

Here's a pseudocode.

```
update = learning_rate * gradient

velocity = previous_update * momentum
```

parameter = parameter + velocity – update

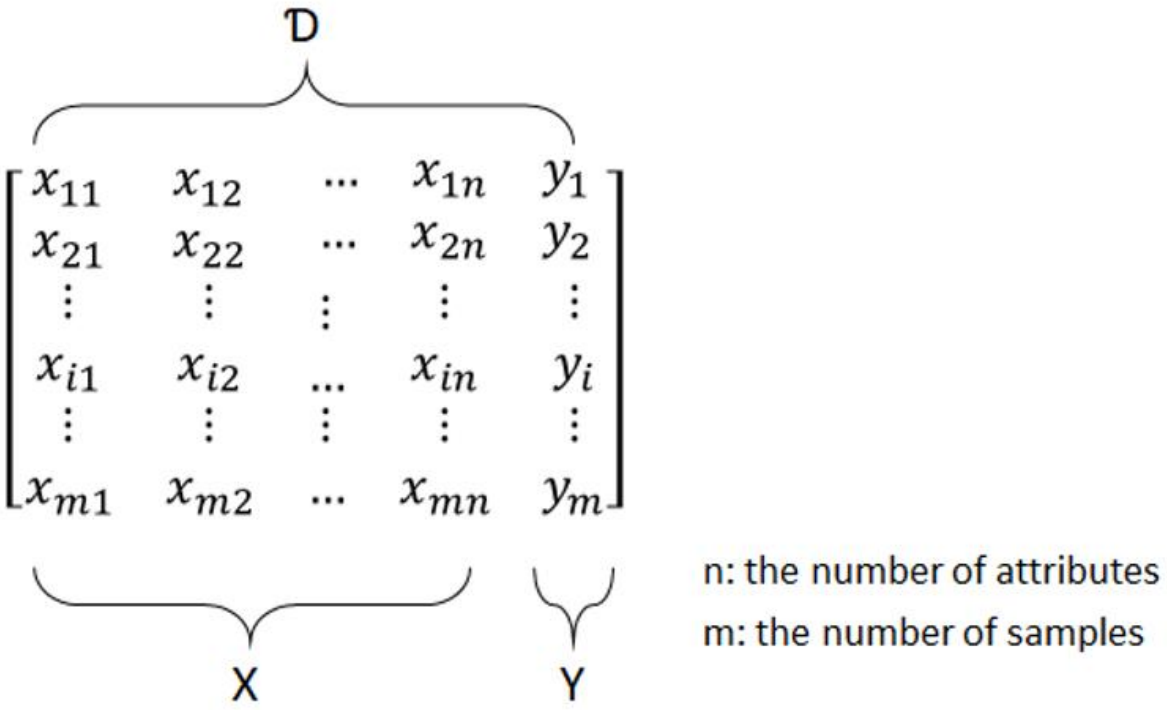
Here, our update is the same as that of vanilla gradient descent. But we introduce a new term called velocity, which considers the previous update and a constant which is called momentum.



Perceptron Learning Algorithm

Perceptron Learning Algorithm is the simplest form of artificial neural network, i.e., single-layer perceptron. A perceptron is an artificial neuron conceived as a model of biological neurons, which are the elementary units in an artificial neural network. An artificial neuron is a linear combination of certain (one or more) inputs and a corresponding weight vector. That is to say that a perceptron has the following definitions:

Given a data set D which contains training data set X and output labels Y, and can be formed as a matrix.



Each x_i is one sample of X, $x_i = \{x_{i1}, x_{i2}, ..., x_{in}\} \forall i \in N, i = 1, 2, ..., n$

Each y_i is the actual label and has a binary value: $\{-1, 1\}$.

W is the weight vector. Each x_{ij} has a correspoinding weight w_j

Since a perceptron is a linear combination of X and W, it can be denoted as

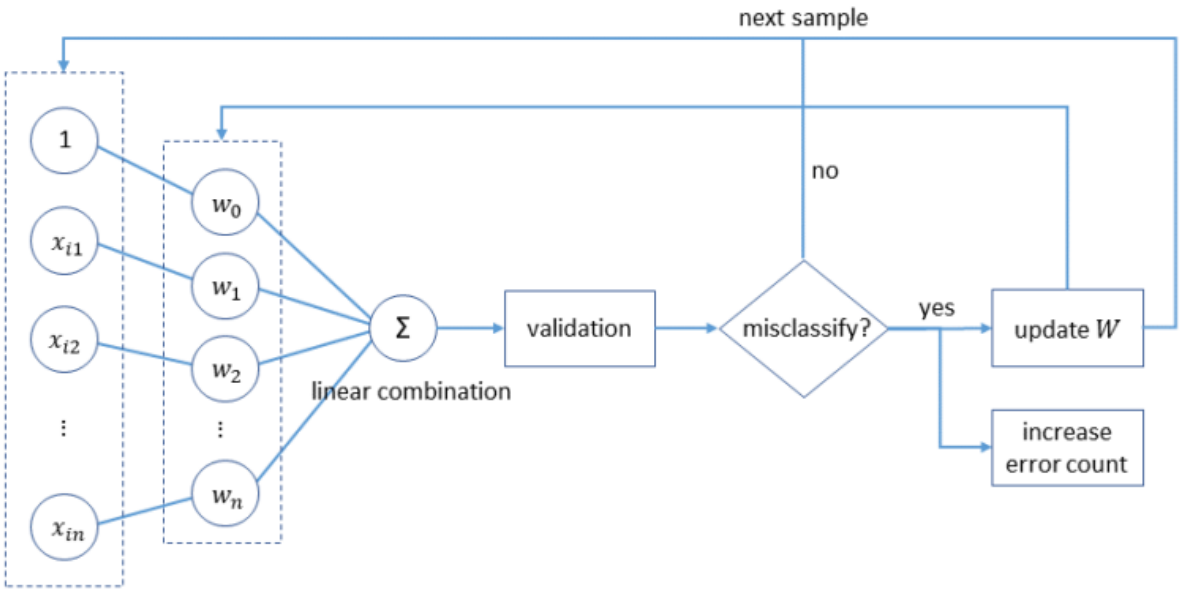
For each neuron y_j, the output is $y_j = \phi\left(\sum_{j=1}^{n} x_{ij} w_j\right)$, where the ϕ is the transfer function:

For the sake of simplicity, ϕ can be treated as $x_{i0}w_0$, where $x_{i0} = 1$

Therefore, the linear combination of X and W can be rewritten as

The output of jth neuron is $y_j = \phi\left(\sum_{j=0}^{n} x_{ij}w_j\right)$, where the ϕ is the transfer function:

The Learning Steps



Given the definition of Perception, the Perceptron Learning Algorithms works by the following steps:

1. Initialize the weights to 0 or small random numbers.
2. For each training sample, x_i, perform the following sub steps:
1. Compute ϕ, the linear combination of X and W, to get the predicted output, class label p_j.
2. Update the weights W
3. Record the number of misclassification

3. If the number of misclassification is not 0 after training the whole set, X, repeat steps 2 and start from the beginning of the trainin set, i.e., start from x_{0j}. Repeat this step until the number of mislassification is 0.

Note:

The output of 2.1 is the class predicted by the ϕ function.

In the step 2.2, the update of each weight, w_j, of W follows the rule:

$w_j(t+1) = w_j(t) + \Delta w_j = w_j(t) + (p_j - y_j)\, x_{ij}$, where t indicatees the step: t means current step, and $t+1$ means next step. Therefore, $w_j(t)$ indicates the current weight and $w_j(t+1)$ indicates the weight after updated.

If no misclassify, $\Delta w_j = (-1-(-1))\, x_{ij} = 0$ or $\Delta w_j = (1-1)\, x_{ij} = 0$. In this case, $w_j(t+1) = w_j(t)$. No update.

If misclassify, $\Delta w_j = (-1-1)\, x_{ij} = -2x_{ij}$ or $\Delta w_j = (1-(-1))\, x_{ij} = 2x_{ij}$. In this case, $w_j(t+1) = w_j(t) + 2x_{ij}$ or $w_j(t+1) = w_j(t) - 2x_{ij}$. Weight updates.

In the step 2.3, the convergence of PLA is only guaranteed if the two classes are linearly separable. If they are not, the PLA never stops. One simple modification is Pocket Learning Algorithm, which will be discussed in a future post.

The Perceptron Learning Algorithm can be simply implemented as following:

Hide Shrink ▲ Copy Code

```
import numpy as np
```

```
class PerceptronClassifier:
    '''Preceptron Binary Classifier uses Perceptron Learning Algorithm
       to do classification with two classes.

       Parameters
       ----------
       number_of_attributes : int
          The number of attributes of data set.

       Attributes
       ----------
       weights : list of float
          The list of weights corresponding &amp;lt;g class="gr_ gr_313 gr-alert gr_gramm
gr_inline_cards gr_run_anim Grammar multiReplace" id="313" data-gr-
id="313"&amp;gt;with&amp;lt;/g&amp;gt; input attributes.

       errors_trend : list of int
          The number of misclassification for each training sample.
       '''
    def __init__(self, number_of_attributes: int):
        # Initialize the weigths to zero
        # The size is the number of attributes plus the bias, i.e. x_0 * w_0
        self.weights = np.zeros(number_of_attributes + 1)

        # Record of the number of misclassify for each train sample
        self.misclassify_record = []
```

```
    self._label_map = {}
    self._reversed_label_map = {}

def _linear_combination(self, sample):
    '''linear combination of sample and weights'''
    return np.inner(sample, self.weights[1:])

def train(self, samples, labels, max_iterator=10):
    '''Train the model

    Parameters
    ----------
    samples : two dimensions list
        Training data set
    labels : list of labels
        Class labels. The labels can be anything as long as it has only two types of labels.
    max_iterator : int
        The max iterator to stop the training process
        in case the training data is not converaged.
    '''
    # Build the label map to map the original labels to numerical labels
    # For example, ['a', 'b', 'c'] -&amp;gt; {0 : 'a', 1 : 'b', 2 : 'c'}
    self._label_map = {1 : list(set(labels))[0], -1 : list(set(labels))[1]}
    self._reversed_label_map = {value : key for key, value in self._label_map.items()}

    # Transfer the labels to numerical labels
    transfered_labels = [self._reversed_label_map[index] for index in labels]
```

```
    for _ in range(max_iterator):
        misclassifies = 0
        for sample, target in zip(samples, transfered_labels):
            linear_combination = self._linear_combination(sample)
            update = target - np.where(linear_combination &amp;gt;= 0.0, 1, -1)

            # use numpy.multiply to multiply element-wise
            self.weights[1:] += np.multiply(update, sample)
            self.weights[0] += update

            # record the number of misclassification
            misclassifies += int(update != 0.0)

        if misclassifies == 0:
            break
        self.misclassify_record.append(misclassifies)

def classify(self, new_data):
    '''Classify the sample based on the trained weights

    Parameters
    ----------
    new_data : two dimensions list
        New data to be classified

    Return
```

```
        ------
        List of int
            The list of predicted class labels.
        '''
        predicted_result = np.where((self._linear_combination(new_data) + self.weights[0])
&amp;gt;= 0.0, 1, -1)
        return [self._label_map[item] for item in predicted_result]
```

Apply Perceptron Learning Algorithm onto Iris Data Set

Normally, the first step to apply machine learning algorithm to a data set is to transform the data set to something or format that the machine learning algorithm can recognize. This process may involve normalization, dimension reduction, and feature engineering. For example, most machine learning algorithms only accept numerical data. Therefore, a data set needs to transfer to a numerical format.

In the Iris Data Set, the attributes, sepal length, sepal width, petal length, and petal width, are a numerical value, but the class labels are not. Therefore, in the implementation of PerceptronClassifier, the train function transfers the class labels to a numerical format. A simple way is to use numbers to indicates these labels: , which means 0 indicates Setosa, 1 implies Versicolour, and 2 means Virginica. Then, the Iris Data Set can be viewed as the form below to feed into Perceptron Learning Algorithm.

$$X = \{x_0, x_1, x_2, x_3, x_4\} = \{1, sepal - length, sepal - width, petal - legnth, petal - width\}$$

$$Y = \{0, 1, 2\}$$

Perceptron is a binary classifier. However, the Iris Data Set has three labels. There are two common ways to deal with multiclass problems: one-vs-all and one-vs-one. For this section, we use a simplified one-vs-one strategy to determinate the types of the iris plant.

One-vs-one approach trains a model for each pair of classes and determines the correct class by a majority vote. For example, the Iris Data Set has three classes. That means all the combinations of a pair of the three classes. That
means $\{\{setosa, versicolour\}, \{setosa, virginica\}, \{versicolour, virginica\}\}$.

Besides, machine learning is not restricted to use all features that the data set has. Instead, only important features are necessary. Here we only consider the two features, Sepal width, and Petal width. In fact, choosing the right features is so important that there is a subject called Feature Engineering to deal with this problem.

Stochastic gradient descent

Stochastic gradient descent (often shortened to **SGD**), also known as **incremental** gradient descent, is an iterative method for optimizing a differentiable objective function, a stochastic approximation of gradient descent optimization.

Both statistical estimation and machine learning consider the problem of minimizing an objective function that has the form of a sum:

where the parameter which minimizes is to be estimated. Each summand function is typically associated with the - observation in the data set (used for training).

In classical statistics, sum-minimization problems arise in least squares and in maximum-likelihood estimation (for independent observations). The general class of estimators that

arise as minimizers of sums are called M-estimators. However, in statistics, it has been long recognized that requiring even local minimization is too restrictive for some problems of maximum-likelihood estimation. Therefore, contemporary statistical theorists often consider stationary points of the likelihood function (or zeros of its derivative, the score function, and other estimating equations).

The sum-minimization problem also arises for empirical risk minimization. In this case, is the value of the loss function at example, and is the empirical risk.

When used to minimize the above function, a standard (or "batch") gradient descent method would perform the following iterations :

where is a step size (sometimes called the *learning rate* in machine learning).

In many cases, the summand functions have a simple form that enables inexpensive evaluations of the sum-function and the sum gradient. For example, in statistics, one-parameter exponential families allow economical function-evaluations and gradient-evaluations.

However, in other cases, evaluating the sum-gradient may require expensive evaluations of the gradients from all summand functions. When the training set is enormous and no simple formulas exist, evaluating the sums of gradients becomes very expensive, because evaluating the gradient requires evaluating all the summand functions' gradients. To economize on the computational cost at every iteration, stochastic gradient descent samples a subset of summand functions at every step. This is very effective in the case of large-scale machine learning problems.

multilayer perceptron

A **multilayer perceptron** (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear

activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then linear algebra shows that any number of layers can be reduced to a two-layer input-output model. In MLPs some neurons use a *nonlinear* activation function that was developed to model the frequency of action potentials, or firing, of biological neurons.

The two common activation functions are both sigmoids, and are described by

The first is a hyperbolic tangent that ranges from -1 to 1, while the other is the logistic function, which is similar in shape but ranges from 0 to 1. Here is the output of the node (neuron) and is the weighted sum of the input connections. Alternative activation functions have been proposed, including the rectifier and softplus functions. More specialized activation functions include radial basis functions (used in radial basis networks, another class of supervised neural network models).

Layers

The MLP consists of three or more layers (an input and an output layer with one or more *hidden layers*) of nonlinearly-activating nodes. Since MLPs are fully connected, each node in one layer connects with a certain weight to every node in the following layer.

Applications:

MLPs are useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely complex problems like fitness approximation.

MLPs are universal function approximators as showed by Cybenko's theorem,[3] so they can be used to create mathematical models by regression analysis. As classification is a particular case of regression when the response variable is categorical, MLPs make good classifier algorithms.

MLPs were a popular machine learning solution in the 1980s, finding applications in diverse fields such as speech recognition, image recognition, and machine translation software,[6] but thereafter faced strong competition from much simpler (and related) support vector machines. Interest in backpropagation networks returned due to the successes of deep learning.

Multi-class Representation:

In machine learning, **multiclass** or **multinomial classification** is the problem of classifying instances into one of three or more classes. (Classifying instances into one of the two classes is called binary classification.)

While some classification algorithms naturally permit the use of more than two classes, others are by nature binary algorithms; these can, however, be turned into multinomial classifiers by a variety of strategies.

Multiclass classification should not be confused with multi-label classification, where multiple labels are to be predicted for each instance.

Neural networks

Multilayer perceptrons provide a natural extension to the multi-class problem. Instead of just having one neuron in the output layer, with binary output, one could have N binary neurons leading to multi-class classification. In practice, the last layer of a neural network is usually a softmax function layer, which is the algebraic simplification of N logistic classifiers, normalized per class by the sum of the N-1 other logistic classifiers.

Backpropagation Algorithm

The backpropagation algorithm was originally introduced in the 1970s, but its importance wasn't fully appreciated until a famous 1986 paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams.

Why We Need Backpropagation?

While designing a Neural Network, in the beginning, we initialize weights with some random values or any variable for that fact.

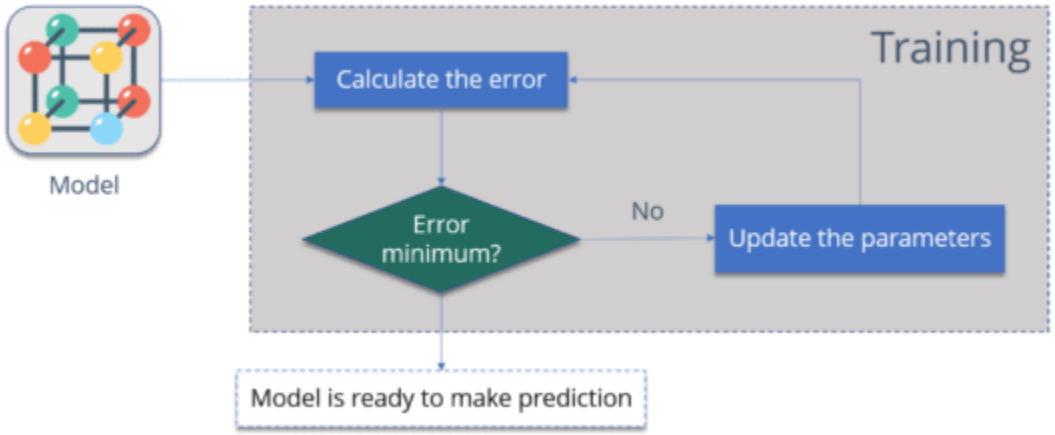
Now obviously, we are not *superhuman*. So, it's not necessary that whatever weight values we have selected will be correct, or it fits our model the best.

Okay, fine, we have selected some weight values in the beginning, but our model output is way different than our actual output i.e. the error value is huge.

Now, how will you reduce the error?

Basically, what we need to do, we need to somehow explain the model to change the parameters (weights), such that error becomes minimum.

One way to train our model is called as Backpropagation. Consider the diagram below:



Let me summarize the steps for you:

- **Calculate the error** – How far is your model output from the actual output.
- **Error minimum?** – Check whether the error is minimized or not.
- **Update the parameters** – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
- **Model is ready to make a prediction** – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.

I am pretty sure, now you know, why we need Backpropagation or why and what is the meaning of training a model.

Now is the correct time to understand what is Backpropagation.

What is Backpropagation?

The Backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.

Let's understand how it works with an example:

You have a dataset, which has labels.

Consider the below table:

Input	Desired Output
0	0
1	2
2	4

Now the output of your model when 'W" value is 3:

Input	Desired Output	Model output (W=3)
0	0	0
1	2	3
2	4	6

Notice the difference between the actual output and the desired output:

Input	Desired Output	Model output (W=3)	Absolute Error	Square Error
0	0	0	0	0
1	2	3	1	1
2	4	6	2	4

Let's change the value of 'W'. Notice the error when 'W' = '4'

Input	Desired Output	Model output (W=3)	Absolute Error	Square Error	Model output (W=4)	Square Error
0	0	0	0	0	0	0
1	2	3	1	1	4	4
2	4	6	2	4	8	16

Now if you notice, when we increase the value of 'W' the error has increased. So, obviously there is no point in increasing the value of 'W' further. But, what happens if I decrease the value of 'W'? Consider the table below:

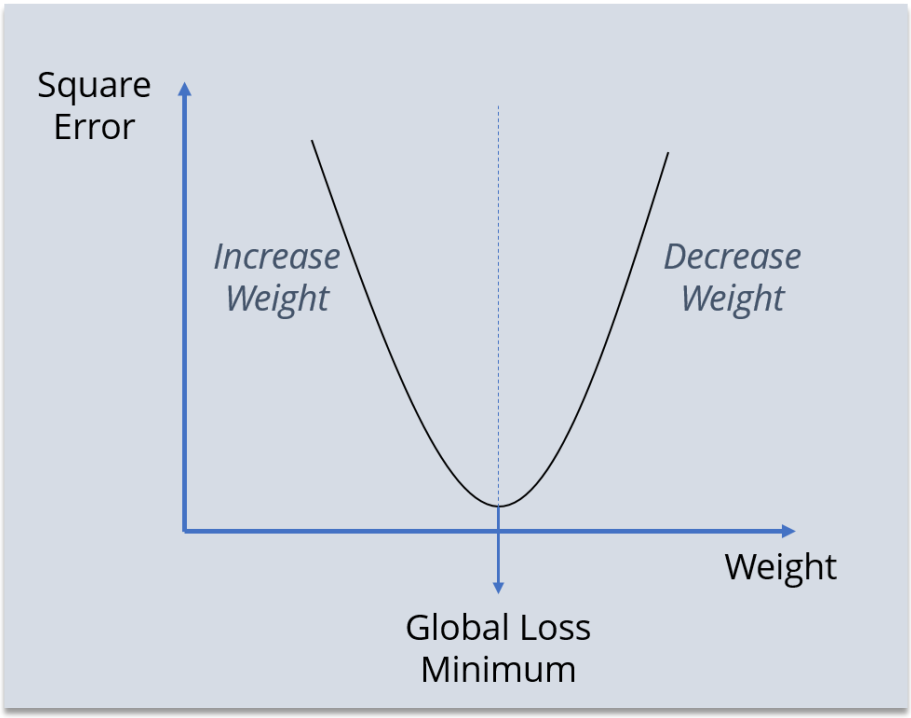
Input	Desired Output	Model output (W=3)	Absolute Error	Square Error	Model output (W=2)	Square Error
0	0	0	0	0	0	0
1	2	3	2	4	3	0
2	4	6	2	4	4	0

Now, what we did here:

- We first initialized some random value to 'W' and propagated forward.
- Then, we noticed that there is some error. To reduce that error, we propagated backwards and increased the value of 'W'.
- After that, also we noticed that the error has increased. We came to know that, we can't increase the 'W' value.
- So, we again propagated backwards and we decreased 'W' value.
- Now, we noticed that the error has reduced.

So, we are trying to get the value of weight such that the error becomes minimum. Basically, we need to figure out whether we need to increase or decrease the weight value. Once we know that, we keep on updating the weight value in that direction until error becomes minimum. You might reach a point, where if you further update the weight, the error will increase. At that time you need to stop, and that is your final weight value.

Consider the graph below:



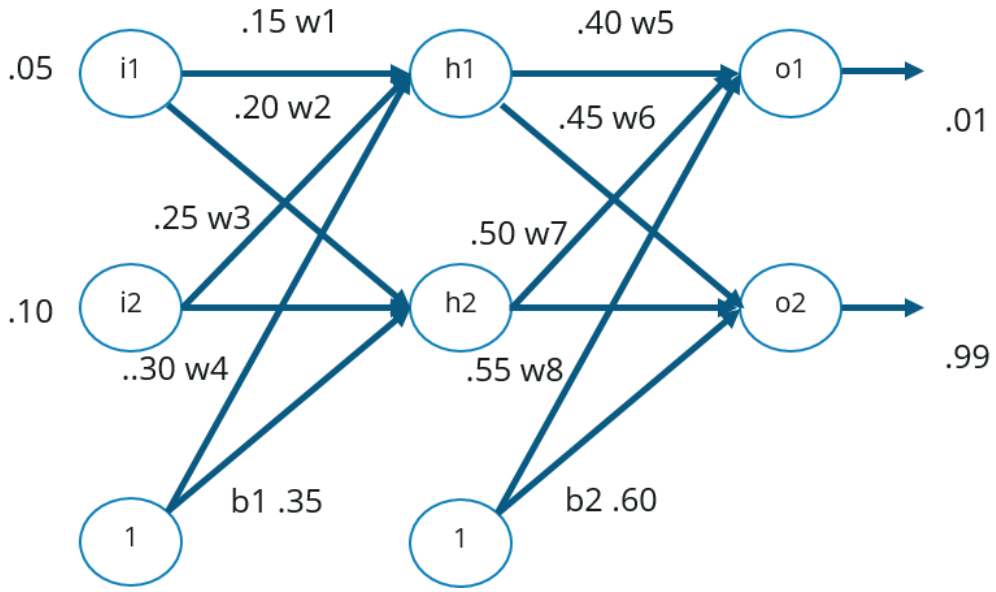
We need to reach the 'Global Loss Minimum'.

This is nothing but Backpropagation.

Let's now understand the math behind Backpropagation.

How Backpropagation Works?

Consider the below Neural Network:



The above network contains the following:

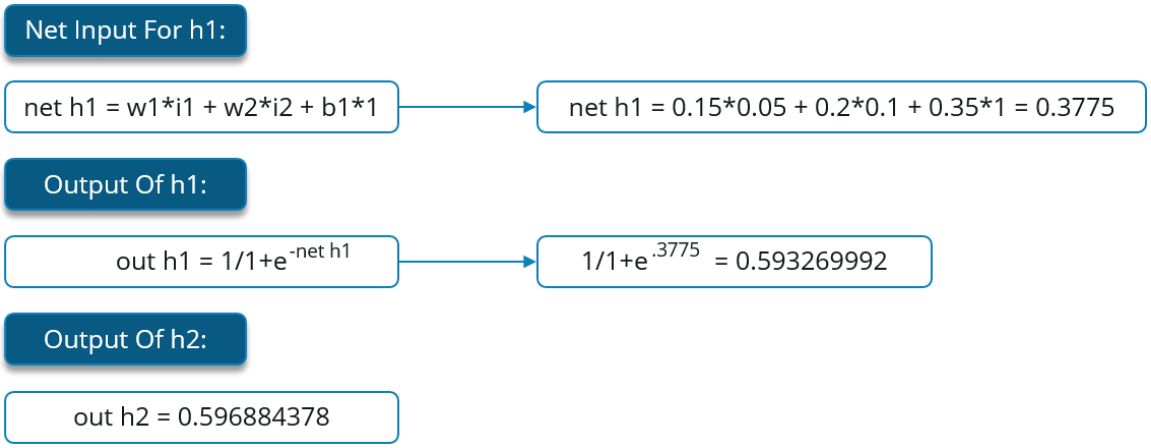
- two inputs
- two hidden neurons
- two output neurons
- two biases

Below are the steps involved in Backpropagation:

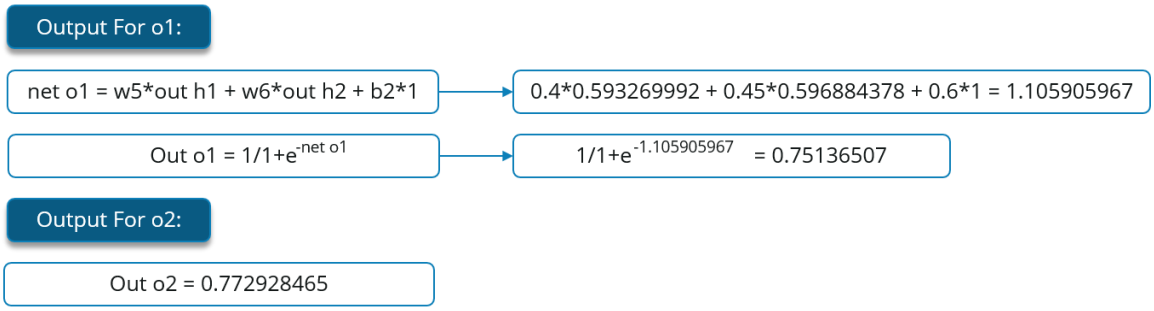
- Step – 1: Forward Propagation
- Step – 2: Backward Propagation
- Step – 3: Putting all the values together and calculating the updated weight value

Step – 1: Forward Propagation

We will start by propagating forward.



We will repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.



Now, let's see what is the value of the error:

Error For o1:

$E_{o1} = \Sigma 1/2(target - output)^2$ → ½ (0.01 – 0.75136507)² = 0.274811083

Error For o2:

E o2 = 0.023560026

Total Error:

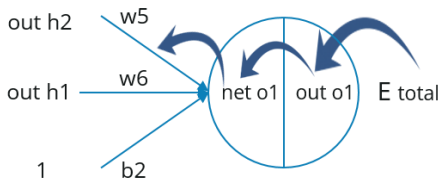
E_{total} = E o1 + E o2 → 0.274811083 + 0.023560026 = 0.298371109

Step – 2: Backward Propagation

Now, we will propagate backwards. This way we will try to reduce the error by changing the values of weights and biases.

Consider W5, we will calculate the rate of change of error w.r.t change in weight W5.

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$



Since we are propagating backwards, first thing we need to do is, calculate the change in total errors w.r.t the output O1 and O2.

$$E_{total} = 1/2(\text{target o1} - \text{out o1})^2 + 1/2(\text{target o2} - \text{out o2})^2$$

$$\frac{\delta Etotal}{\delta out\ o1} = -(\text{target o1} - \text{out o1}) = -(0.01 - 0.75136507) = 0.74136507$$

Now, we will propagate further backwards and calculate the change in output O1 w.r.t to its total net input.

$$\text{out o1} = 1/1+e^{-neto1}$$

$$\frac{\delta out\ o1}{\delta net\ o1} = \text{out o1 (1 - out o1)} = 0.75136507\ (1 - 0.75136507) = 0.186815602$$

Let's see now how much does the total net input of O1 changes w.r.t W5?

$$\text{net o1} = w5 * \text{out h1} + w6 * \text{out h2} + b2 * 1$$

$$\frac{\delta net\ o1}{\delta w5} = 1 * \text{out h1}\ w5^{(1-1)} + 0 + 0 = 0.593269992$$

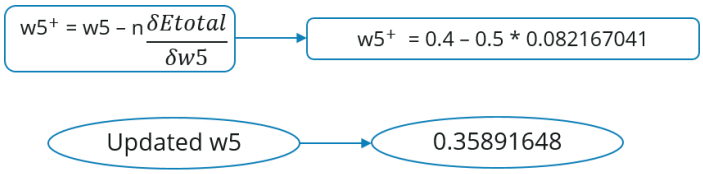
Step – 3: Putting all the values together and calculating the updated weight value

Now, let's put all the values together:

$$\frac{\delta Etotal}{\delta w5} = \frac{\delta Etotal}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5} \rightarrow 0.082167041$$

Let's calculate the updated value of W5:

- Similarly, we can calculate the other weight values as well.
- After that we will again propagate forward and calculate the output. Again, we will calculate the error.
- If the error is minimum we will stop right there, else we will again propagate backwards and update the weight values.
- This process will keep on repeating until error becomes minimum.



Backpropagation Algorithm:

initialize network weights (often small random values)

do

forEach training example named ex

prediction = neural-net-output(network, ex) *// forward pass*

actual = teacher-output(ex)

compute error (prediction - actual) at the output units

compute {\displaystyle \Delta w_{h}} for all weights from hidden layer to output layer *// backward pass*

compute {\displaystyle \Delta w_{i}} for all weights from input layer to hidden layer *// backward pass continued*

update network weights *// input layer not modified by error estimate*

until all examples classified correctly or another stopping criterion satisfied

return the network

Part-B:(2 marks)

1.what is neural networks?

2.Define Gradient Descent.

3. Define Perceptron Training.

4.what is Stochastic Gradient Descent?

5.Define Backpropagation Algorithm.

Part-C:(6 marks)

1.Explain about Neural networks.

2.Discuss briefly Multilayer Perceptron's.

3.Brief note on Backpropagation Algorithm.

4. Compare Gradient Descent vs. Perceptron Training with example.

KARPAGAM ACADEMY OF HIGHER EDUCATION

Department of Computer Science

III B.Sc(CS) (BATCH 2016-2019)

Machine Learning

PART-A OBJECTIVE TYPE/ MULTIPLE CHOICE QUESTIONS

QUESTIONS	OPT1	OPT2	OPT3	OPT4	ANSWER
What is back propagation?	It is another name given to the curvy function in the perceptron	It is the transmission of error back through the network to adjust the inputs	It is the transmission of error back through the network to allow weights to be adjusted so that the network can learn	None of the mentioned	**It is the transmission of error back through the network to adjust the inputs**
Neural Networks are complex _____ with many parameters.	Linear Functions	Nonlinear Functions	Discrete Functions	Exponential Functions	**Linear Functions**
A perceptron adds up all the weighted inputs it receives and if it exceeds a certain value, it outputs a 1, otherwise it just outputs a 0.	True	False	Sometimes – it can also output intermediate values as well	Can't say	**True**
Which of the following is an application of NN (Neural Network)?	Sales forecasting	Data validation	Risk management	All of the mentioned	**All of the mentioned**
Which is not a desirable property of a logical rule-based system?	Locality	Attachment	Detachment	Truth-Functionality	**Attachment**

In an Unsupervised learning	Specific output values are given	Specific output values are not given	No specific Inputs are given	Both inputs and outputs are given	**Specific output values are not given**
Neural Networks are complex _____with many parameters.	Linear Functions	Nonlinear Functions	Discrete Functions	Exponential Functions	**Nonlinear Functions**
A perceptron is a _____	Feed-forward neural network	Backpropagation algorithm	Backtracking algorithm	Feed Forward-backward algorithm	**Feed-forward neural network**
Which is true for neural networks?	It has set of nodes and connections	Each node computes it's weighted input	Node could be in excited state or non-excited state	All of the mentioned	**All of the mentioned**
A perceptron is:	a single layer feed-forward neural network with pre-processing	an auto-associative neural network	a double layer auto-associative neural network	a neural network that contains feedback	**a single layer feed-forward neural network with pre-processing**
What is plasticity in neural networks?	input pattern keeps on changing	input pattern has become static	output pattern keeps on changing	output is static	**input pattern keeps on changing**

Why do we need biological neural networks?	to solve tasks like machine vision & natural language processing	to apply heuristic search methods to find solutions of problem	to make smart human interactive & user friendly system	all of the mentioned	**all of the mentioned**
For what purpose Feedback neural networks are primarily used?	classification	feature mapping	pattern mapping	none of the mentioned	**none of the mentioned**
Back propagation is a learning technique that adjusts weights in the neural network by propagating weight changes.	Forward from source to sink	Backward from sink to source	Forward from source to hidden nodes	Backward from sink to hidden nodes	Backward from sink to source
Perceptron can learn Perceptron can learn	AND	XOR	Both A and B	None of these	AND
A perceptron is a ----------.	Feed-forward neural network	Back-propagation algorithm	Back-tracking algorithm	Feed Forward-backward algorithm	Back-propagation algorithm
An associative network is_____.	A neural network that contains no loop	A neural network that contains feedback	A neural network that has only one loop	None of These	**A neural network that contains feedback**

What is the objective of perceptron learning?	class identification	weight adjustment	adjust weight along with class identification	none of the mentioned	**adjust weight along with class identification**
In perceptron learning, what happens when input vector is correctly classified?	small adjustments in weight is done	large adjustments in weight is done	no adjustments in weight is done	weight adjustments doesn't depend on classification of input vector	**no adjustments in weight is done**
The perceptron convergence theorem is applicable for what kind of data?	binary	bipolar	both binary and bipolar	none of the mentioned	**both binary and bipolar**
Convergence in perceptron learning takes place if and only if:	a minimal error condition is satisfied	actual output is close to desired output	classes are linearly separable	all of the mentioned	**classes are linearly separable**
In a three layer network, shape of dividing surface is determined by?	number of units in second layer	number of units in third layer	number of units in second and third layer	none of the mentioned	**number of units in second layer**
If the output produces nonconvex regions, then how many layered neural is required at minimum?	2	3	4	5	**4**

When connect is interrupted by a caught signal that is not restarted,we must call_____to wait for the connection to complete.	wait	select	write	READS	select
For the process calling WAIT,the_____gives us control over which process to wait and whether or not to block	waitpid	wait	select	none	waitpad
If the client needs perform too many writes to the server before reading error from readline_____signal is sent to process.	SIGPIPE	SIG_IGN	SIGKILL	VC	SIGPIPE
When a process writes to a socket that has received an RST,_____signal is sent to the process.	EPIPE	SIGPIPE	SIGSTOP	none	SIGPIPE
When a unix system is shutdown the _____process normally sent the SIGTERM signal to call the process.	INIT	SIGKILL	select	Slash	INIT
_____is an optional flag when set,assistance call interrupt by the signal will be automatically restarted by the kernel.	SA_INTERRUPT	SIGALRM	SIG_RESTART	none	started

Posix allows us to specify a set of signal that will be _____when signal handler is called	delivered	blocked	started	none	blocked
Any signal that is blocked_____be delivered to the process.	blocked	can be	may be	Slash	blocked
compliment of SA_RESTART flag	SIGALRM	SA_START	blocked	none	blocked
If SA_INTERRUPT is defined,we set it if the signal being caught is	SIGALRM	SIGACTION	SIGSTOP	none	SIGALRM
If a process terminate and the process has children in the zombie state,the parent process up all the zombie children is set to_____	0	1	-1	none	1
Whenever we fork the children,we must_____for them to parent them from becoming zombies.	LISTEN	WAIT	WRITE	none	WAIT

We terminate the client by typing_____	EOF	Slash	/n	none	EOF
We used to term slow system call to discrete_____	return	terminate	accept	none	accept
Function that we cannot restart ourself is ?	wait	connect	readline	write	connect
We call the_____function to handle nthe terminate child.	wait	waitpid	connect	none	wait
Serverhost crashed and there were no response atall to theclients datasegment,the error is	EHOSTUREACH	ENETUNREACH	ETIMEOUT	none	ETIMEOUT
The server host was unreachable and respond with the ICMP ddestination unreachable message error is	ETIMEOUT	ECHOSTUNREACH	ECONNRESET	none	ECHOSTUNREACH

Our client is blocked in the call to read line when the__is received.	RST	CLR	ACK	none	RST
Berkely delivered implementation retransmit the data segment_____times and waiting for around_____minute.	10,11	12,9	11,12	none	12,9
When a client is handling multiple descriptors, _____ is used	I/O Multiplexing	Signal Handling	I/O Demultiplexing	I/O Messaging	I/O Multiplexing
When a client is handling multiple socket at the same time, _____ is used	Signal Handling	I/O Multiplexing	I/O Demultiplexing	I/O Messaging	Signal Handling
To handle both listening socket & its connected socket, a TCP server uses _____ .	I/O Demultiplexing	I/O Messaging	Signal Handling	I/O Multiplexing	I/O Multiplexing
If a server handles both TCP and UDP, _____ is used.	I/O Demultiplexing	I/O Messaging	I/O Multiplexing	Signal Handling	I/O Multiplexing

Capability of handling one or more I/O conditions is called _____	I/O Multiplexing	Signal Handling	I/O Demultiplexing	I/O Messaging	I/O Multiplexing
If a server handles multiple services, _____ is used.	Signal Handling	I/O Multiplexing	I/O Demultiplexing	I/O Messaging	Signal Handling
If a server handles multiple protocols, _____ is used.	I/O Demultiplexing	I/O Messaging	Signal Handling	I/O Multiplexing	I/O Multiplexing
fcntl stands for what?	Signal Handling	function control	file corrupt	function corrupt	Signal Handling
Which of the following is the IPv4 Socket option?	Signal Handling	ICMP6_FILTER	IPV6_HOPOPTS	TCP_KEEPALIVE	Signal Handling
Which of the following is the IPv6 Socket option?	IP_RECVDSTADDR	ICMP6_FILTER	IPV6_HOPOPTS	TCP_KEEPALIVE	IPV6_HOPOPTS

Which of the following is the TCP Socket option?	IP_RECVDSTADDR	ICMP6_FILTER	IPV6_HOPOPTS	TCP_KEEPALIVE	TCP_KEEPALIVE
Every UDP socket has a _____ .	send buffer	TCP_KEEPALIVE	send buffer & receive buffer	None of the above	TCP_KEEPALIVE
Every TCP socket has a _____ .	send buffer	receive buffer	send buffer & receive buffer	None of the above	send buffer & receive buffer