INDEX

S. No	Name	Sign
1	To get the histogram of an image using matplotlib.	
2	To implement principal Component Analysis on a given dataset.	
3	Implementing a Gaussian Naïve Bayes Classifier .	
4	To implement Naive Bayes on a small Language dataset using NLP.	
5	To implement K-Means Classifier.	
6	To implement a K-Nearest Neighbors Classifier.	
7	To implement Linear Regression.	
8	To implement Support Vector Machine (SVM) Classifier.	
9	To use MLP to implement various gates	
10	To implement artificial neural network	

Aim :

To get the histogram of an image using matplotlib.

Theory :

Histogram is considered as a graph or plot which is related to frequency of pixels in an grayscale image with pixel values (ranging from 0 to 255). Grayscale image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information where pixel value varies from 0 to 255. Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest where Pixel can be considered as a every point in an image.

CONCLUSION :

Histogram can be used to find entire tonal distribution of a specific image at a glance. In the field of computer vision, image histograms can be useful tools for thresholding. Because the information contained in the graph is a representation of pixel distribution as a function of tonal variation, image histograms can be analyzed for peaks and/or valleys.

Aim:

To implement Principal Component Analysis on a given dataset

Theory:

PCA is a linear transformation that finds the "principal components", or directions of greatest variance, in a data set. It can be used for dimension reduction among other applications such as image compression.

In simple words, principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set. It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible. With fewer variables, visualization also becomes much more meaningful. PCA is more useful when dealing with 3 or higher dimensional data.

It is always performed on a symmetric correlation or covariance matrix. This means the matrix should be numeric and have standardized data.

In this exercise, we implement PCA and apply it to a simple 2-dimensional data set to understand its working.

Steps: -

1. Standardize the data.
2. Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix, or perform Singular Vector Decomposition.
3. Sort eigenvalues in descending order and choose the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace ($k \leq d$).
4. Construct the projection matrix W from the selected k eigenvectors.
5. Transform the original dataset X via W to obtain a k-dimensional feature subspace Y.

Result:

We performed PCA on a 2-dimensional dataset (visualized using a heatmap as shown below) and reduced it into 1 dimensions (shown in the scatterplot below) while retaining approximately 96% of the information.

Conclusion:

PCA is a powerful decomposition technique for linear dimensionality reduction to project data into a lower dimensional space.

Aim:

Implementing a Gaussian Naïve Bayes Classifier

Theory:

It is a classification technique based on Bayes Theorem.

Along with an assumption of independence among predictors i.e. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity.

Result:

We performed naïve Bayes classification on the given dataset to classify test data into one of the 2 classes representing "Diabetic" or "Non-Diabetic" with appreciable (approximately 75%) accuracy.

Conclusion:

Naive Bayes is known to outperform even highly sophisticated classification methods.

Pros:

It is easy and fast to predict class of test data set. It also perform well in multi class prediction

When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.

Cons:

If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction. This is often known as "Zero Frequency".

Aim:

To implement Naive Bayes on a small Language dataset using NLP

Theory:

Naive Bayes is a family of probabilistic algorithms that take advantage of probability theory and Bayes' Theorem to predict the category of a sample (like a piece of news or a customer review). They are probabilistic, which means that they calculate the probability of each category for a given sample, and then output the category with the highest one. The way they get these probabilities is by using Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature.

Steps: -

1. Get word frequencies.
2. Obtain word probabilities for positive and negative samples
3. calculate the class model.
4. Calculate the prior probabilities.
5. Calculate the needed probabilities using naive bayes.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Result:

test_not_sports : 0.0002915451895043731

test_sports : 0.012244897959183673

Conclusion:

We calculated the needed probabilities using naive bayes classifier

Aim:

To implement K-Means Clustering

Theory:

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster.

These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step.

After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more.

Results:

We implemented a K-Means Clustering algorithm from scratch in python and used it to classify data points generated randomly from mean and covariance matrix.

Conclusion:

Pros

1. With zero to little training time, it can be a useful tool for off-the-bat analysis of some data set you are planning to run more complex algorithms on
2. Easy to interpret the clustering results.
3. Fast and efficient in terms of computational cost.

Cons

1. Uniform Effect: Often produce clusters with relatively uniform size even If the input data has different cluster size.
2. K value not known: how to solve K? 1)for small range of K value, say 2-10, for each K value run lots of times(20-100 times), take the clustering result with the lowest J value among all K values; using Elbow method to decide K value; 3) GAPs; 4) decide the K down streams: decide by the purposes/goals of the projects
3. Sensitive to initial points and local optimal, and there is no unique solution for a certain K values, thus it needs to run for a lot of iterations.

Aim:

To implement a K-Nearest Neighbors Classifier

Theory:

We use x to denote a *feature* (aka. predictor, attribute) and y to denote the *target* (aka. label, class) we are trying to predict.

KNN falls in the **supervised learning** family of algorithms. Our goal is to learn a function $h:X \rightarrow Y$ so that given an unseen observation x, h(x) can confidently predict the corresponding output y.

The KNN classifier is also a **non parametric** and **instance-based** learning algorithm.

- **Non-parametric** means it makes no explicit assumptions about the functional form of h, avoiding the dangers of mis-modeling the underlying distribution of the data. For example, suppose our data is highly non-Gaussian but the learning model we choose assumes a Gaussian form. In that case, our algorithm would make extremely poor predictions.

- **Instance-based** learning means that our algorithm doesn't explicitly learn a model. Instead, it chooses to memorize the training instances which are subsequently used as "knowledge" for the prediction phase. Concretely, this means that only when a query to our database is made (i.e. when we ask it to predict a label given an input), will the algorithm use the training instances to spit out an answer.

It is worth noting that the minimal training phase of **KNN comes both at a *memory cost***, since we must store a potentially huge data set, as well as a ***computational cost*** during test time since classifying a given observation requires a run down of the whole data set. Practically speaking, this is undesirable since we usually want fast responses.

Results:

We implemented a K-nearest neighbors algorithm from scratch in python and used it to classify data points from "Iris Dataset" with a remarkable ~96% accuracy.

Conclusion:

Pros and Cons of KNN

Pros

1. One of the most attractive features of the K-nearest neighbors algorithm is that is simple to understand and easy to implement.
2. With zero to little training time, it can be a useful tool for off-the-bat analysis of some data set you are planning to run more complex algorithms on
3. KNN works just as easily with multiclass data set whereas other algorithms are hardcoded for the binary setting.

4. the non-parametric nature of KNN gives it an edge in certain settings where the data may be highly “unusual”.

Cons

1. One of the obvious drawbacks of the KNN algorithm is the computationally expensive testing phase which is impractical in industry settings.

2. KNN can suffer from skewed class distributions. For example, if a certain class is very frequent in the training set, it will tend to dominate the majority voting of the new example (large number = more common).

3. The accuracy of KNN can be severely degraded with high-dimension data because there is little difference between the nearest and farthest neighbors.

Aim:

To implement Linear Regression

Theory:

In statistics, linear regression is a linear approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression

Least Squared Error is used as the loss function. For learning the weights, Stochastic Gradient Descent is used in this experiment to update the weights in order to lead to minimum loss.

Linear regression is widely used in biological, behavioral and social sciences to describe possible relationships between variables. It ranks as one of the most important tools used in these disciplines.

Conclusion:

Pros

1. A linear model is much more simple than a quadratic model, and often works just as well for most purposes.
2. Linear regression implements a statistical model that, when relationships between the independent variables and the dependent variable are almost linear, shows optimal results.

Cons

1. It is sensitive to outliers.
2. Overfitting - It is easy to overfit your model such that your regression begins to model the random error (noise) in the data, rather than just the relationship between the variables. This most commonly arises when you have too many parameters compared to the number of samples
3. Linear regressions are meant to describe linear relationships between variables. So, if there is a nonlinear relationship, then you will have a bad model.

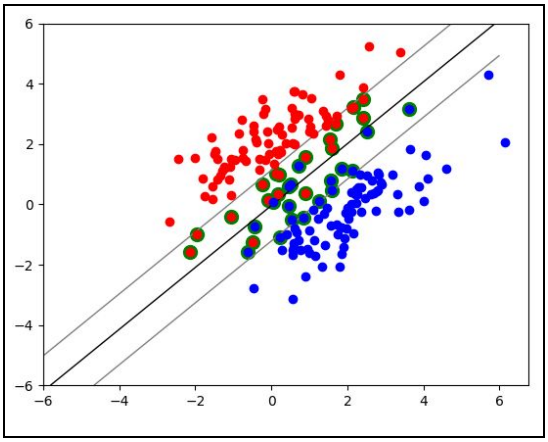
Aim :

To implement SVM Classifier.

Theory :

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

RESULT :



CONCLUSION :

Pros

- Guaranteed Optimality: Due to the nature of Convex Optimization, the solution is guaranteed to be the global minimum not a local minimum.
- Abundance of Implementations: from libsvm (Python) to >>fitcsvm (Matlab) it is conveniently accessed. These libraries may be utilized even though the user is indifferent to the KKT Conditions, Lagrange Multipliers etc.
- It is useful for both Linearly Separable (hard margin) and Non-linearly Separable (soft margin) data. The only thing to do is to come up with the optimal penalty variable C (the one that multiplies slack variables)

Cons:

- In Natural Language Processing, structured representations of text yield better performances. Sadly, SVMs can not accommodate such structures(word embeddings) and are used through Bag-of-Words representation which loses sequentiality information and leads to worse performance.

Aim:

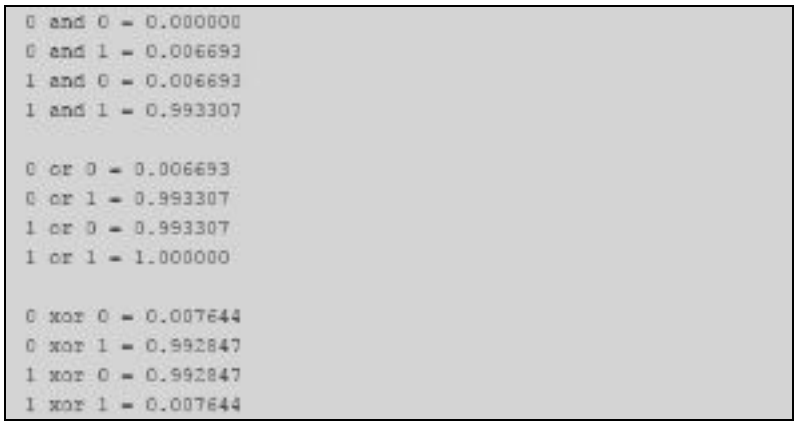
To design And , Or and XOR Gates using Multi Layer Perceptrons

Theory:

The sigmoid function [1/ (1+exp(-x))] has a range from 0 to 1 depending upon the value of x.

As shown in figure, as x tends to large negative value, sigmoid function would return a value close to zero whereas for large positive values, it will return a value closer to 1. We can use the same principal and add bias values as per the need of the logic gate to create one.

Results:



Conclusion:

Using the sigmoid function and the concepts of Multilayer Perceptrons, we have successfully designed the three LOGIC Gates as aimed.

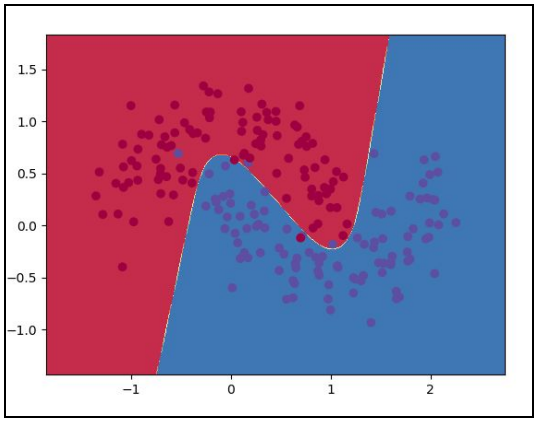
Aim:

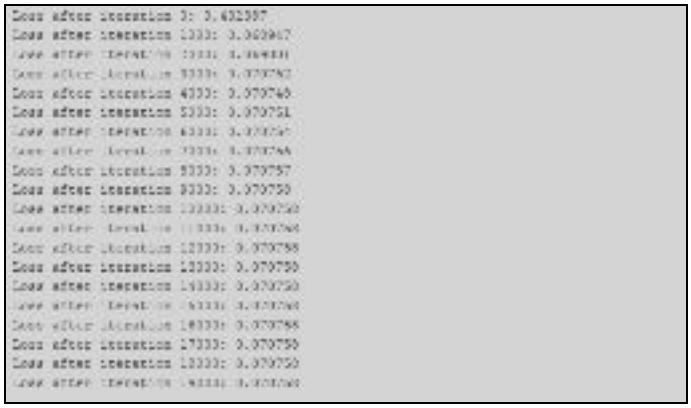
To implement artificial neural network

Theory:

Artificial neural networks (ANNs), a form of connectionism, are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express in a traditional computer algorithm using rule-based programming. An ANN is based on a collection of connected units called artificial neurons (analogous to biological neurons in an animal brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have a state, generally
represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Further, they may have a threshold such that only if the aggregate signal is below (or above) that level is the downstream signal sent.

Results :





Thus we implemented a neural network from scratch.

Conclusion:

Pros and Cons of Artificial Neural Nets

Pros

- 1. easy to conceptualize
- 2. large amount of academic research
- 3. used extensively in industry for many years
- 4. lots of libraries / implementations available

Cons

- 1. there are alternatives that are simpler, faster, easier to train, and provide
- better performance (svm, decision trees, regression)
- 2. multi-layer neural networks are usually hard to train, and require tuning
- lots of parameters.