**Question 1**

a) A Machine Learning practitioner needs to estimate the parameters 𝜃 of a model, given some data 𝑋. Suppose, that the parameters can take only two values, 𝜃1 or 𝜃2 and that it is given that 𝑝(𝑋|𝜃1) = 0.6, 𝑝(𝑋|𝜃2) = 0.5, 𝑝(𝜃1) = 0.3, and 𝑝(𝜃2) = 0.7.

i) What does each of the terms 𝑝(𝑋|𝜃𝑖) and 𝑝(𝜃𝑖), 𝑖 ∈ {1,2} denote?

ii) Which model will he/she choose under the Maximum Likelihood criterion? Which model will he/she choose under the Maximum Aposteriori Probability criterion? Show the calculations.

Answer:-

1. P(X|𝜃) denotes maximum likelihood. This resulting conditional probability is referred to as the likelihood of observing the data given the model parameters. Whereas P(𝜃) denotes the prior probability over the distribution of parameters and it acts a regularizer to prevent the model from overfitting.
2. 2.While considering Maximum Likelihood criteria we will choose 𝑝(𝑋|𝜃1) = 0.6 as its value is more than 𝑝(𝑋|𝜃2).  
   MAP (1) = P(X|𝜃1)\*P(𝜃1) = 0.6\*0.3 = 0.18  
   MAP (2) = P(X|𝜃2)\*P(𝜃2) = 0.5\*0.7 = 0.35  
   While considering Maximum Aposteriori Probability criterion we will choose model with 2 as its value is more than model with 1.

b) Explain the “gradient descent” algorithm. What is the learning rate in this context and how does a Machine Learning practitioner set its value?

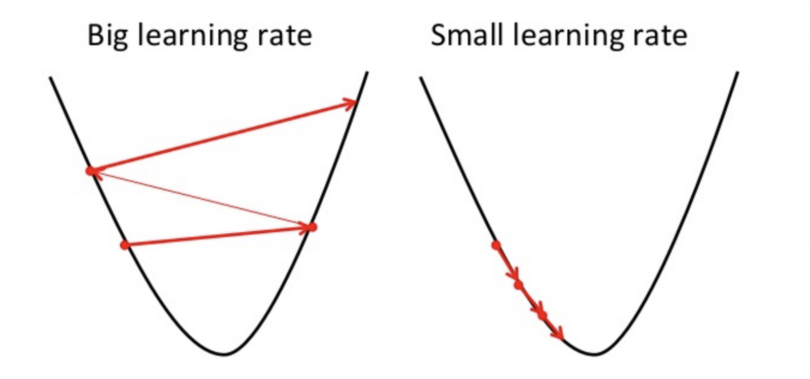
Answer:-

Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. Gradient descent is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible

Gradient descent is an optimization algorithm which is mainly used to find the minimum of a function. In machine learning, gradient descent is used to update parameters in a model. Parameters can vary according to the algorithms, such as *coefficients* in Linear Regression and weights in Neural Networks.

How big the steps are gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights.

For gradient descent to reach the local minimum we must set the learning rate to an appropriate value, which is neither too low nor too high. This is important because if the steps it takes are too big, it may not reach the local minimum because it bounces back and forth between the convex function of gradient descent (see left image below). If we set the learning rate to a very small value, gradient descent will eventually reach the local minimum but that may take a while (see the right image).



So, the learning rate should never be too high or too low for this reason. You can check if you’re learning rate is doing well by plotting it on a graph.

A good way to make sure gradient descent runs properly is by plotting the cost function as the optimization runs. Put the number of iterations on the x-axis and the value of the cost-function on the y-axis. This helps you see the value of your cost function after each iteration of gradient descent, and provides a way to easily spot how appropriate your learning rate is. You can just try different values for it and plot them all together.

c) In a Machine Learning problem, you are asked to fit a linear regression model to data from a dataset 𝑋 = {𝑥1, 𝑥2, … , 𝑥𝑚}, where 𝑥𝑚 ∈ 𝑅𝑑, with 𝑚 = 105 and 𝑑 =108.

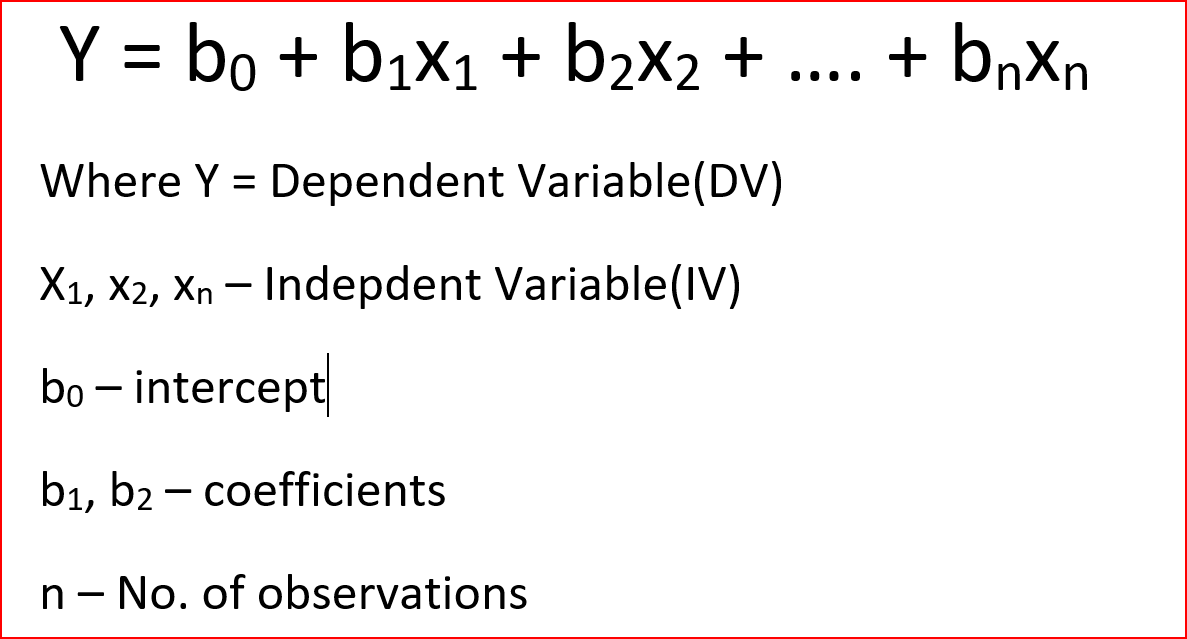
i) What is the specific issue that arises given those values of 𝑚 and 𝑑 and how can you address it?

ii) Give the equation that you will use to estimate the parameters of the model in that case and explain what each term denotes.

Answer:-

i)In this problem we have see that the dataset are very far away from that the best fit line if we used the value of m and d and this produces the error in the linear regression fit model.in this problem the dataset goes very very away from the line of action which would you go the file for this dataset and the line may not be converged to the specific equation that’s why this specific issue arises in the dataset.

ii)In this equation we see the equation below



From this equation we put the value of above parameter so we will get the result according to that from that we see that the values of x1,x2,….xm will be putting in the equation so we will get the result of the output which should be given in the above equation from this result we see that the model may be converge to a specific values which regards to the value of m and d.

**Question 2**

1. The probability density function of the Gaussian distribution is given by 𝑝(𝑥; 𝜇, Σ) = 1 (2𝜋) 𝑑/2|Σ| 1/2 1 𝑒 −2 𝑥−𝜇 ) 𝑇Σ−1 (𝑥−𝜇) . Explain what each symbol denotes and give its dimensionality. Sketch the probability density function in the 1-dimensional case and use it to support your explanations.

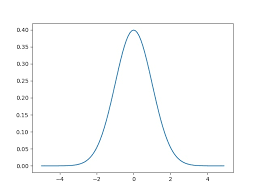
Answer:-

d is the dimensionality of the data vectors (scalar)

x is an input data vector (dimensionality m)

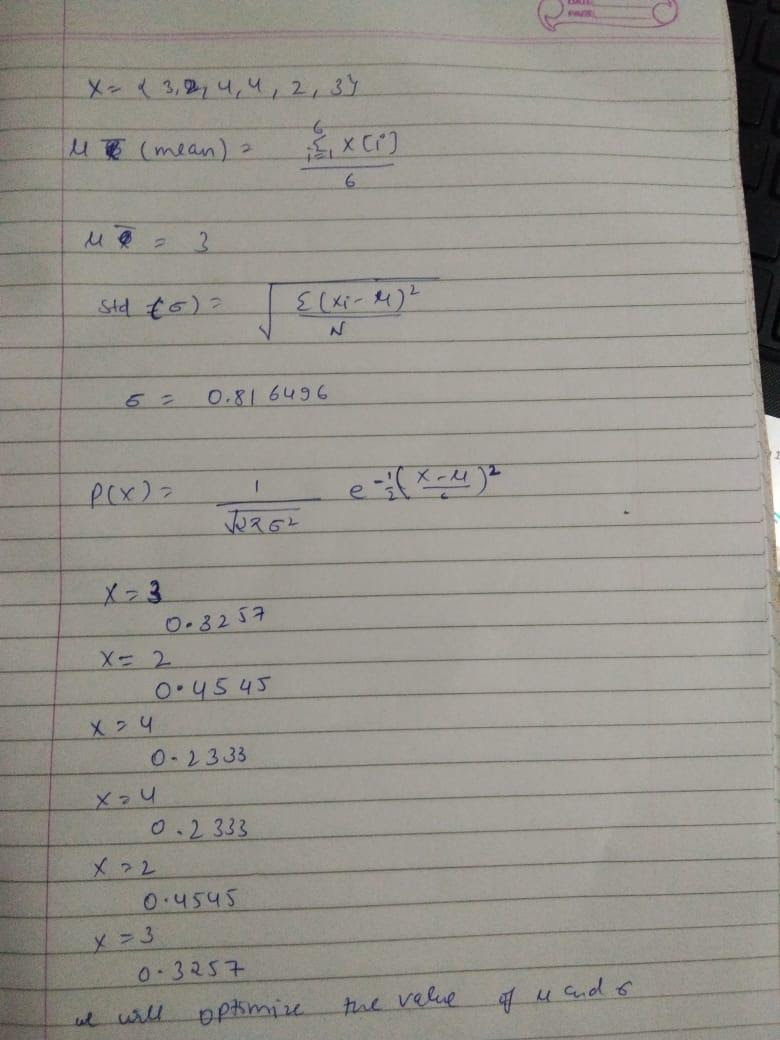
μ is the vector of means (dimensionality m)

Σ is the covariance matrix (dimensionality m\*m)



1. Fit a Gaussian distribution to the following set of 1-dimensional data 𝑋 = {3,2,4,4,2,3}. Show the formulas that you use for your estimation and state what is the criterion that is optimised.

Answer:-



1. In several cases the covariance matrix is singular and cannot be inverted. In which cases is this more likely to happen? Why is this a problem? Give a solution to it.

Answer:- A matrix in linear algebra is said to be singular if its determinant is zero and this basically happens when either the rows of the matrix or the columns within the matrix are correlated with each other. By correlation, it means they are a multiple of each other and due to this when their determinant is calculated, it comes out to be 0, and hence the matrix is said to be singular. It is a problem as it leads to redundancy and many times, it also leads to problems such as noise, etc. while you are working in the field of data science.

In order to deal with such kind of problem, you need to make sure that the columns or the rows that are present within the matrix are not linearly related or they are not multiple of each other, if they are then you need to make sure that they are removed and only then you can actually make a matrix non-singular and its determinant would be non-zero otherwise it would always be singular and would pose problems in a longer run.

**Question 3**

a) You need to design a Neural Network that solves the problem of facial attribute recognition. More specifically the network should receive in the input an image of a face, and should recognise whether the depicted subject wears glasses or not, has long or short hair, smiles or not and should recognise its apparent age. Design the first and the last layers of such a network, detailing your choices. Define the total cost function and give the format of a training example and the corresponding ground truth associated with it.

[Hint: You can treat the recognition of the age either as a regression problem, or as a classification problem – either choice is equally valid.]

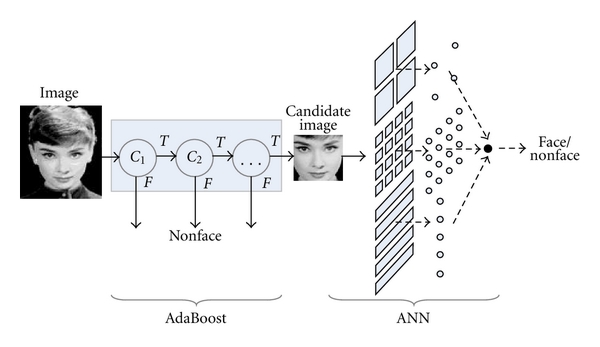
Answer:-

The face detection processing is the first step of the face recognition system. The step will decide the performance of the system, so it is the most important step of the recognition system. To carry out its efficiently, many researchers have proposed different approaches. In general, there are four groups of face detecting methods : (1) *Knowledge-based methods*; (2) *Invariant feature-based methods*; (3) *Template matching-based methods*; (4) *Machine learning-based methods*.

In this paper, we focus on only machine learning methods because they eliminate subjective thinking factors from human experience. Moreover, they only depend on training data to make final decisions. Thus, if training data is well organized and adequate, then these systems will achieve high performance without human factors.

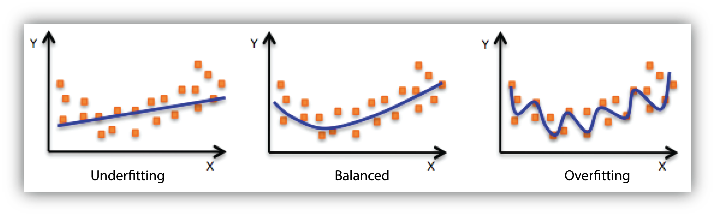
One of the most popular and efficient learning machine-based approaches for detecting faces is AdaBoost approach . Viola and Jones designed a fast, robust face detection system where AdaBoost learning is used to build nonlinear classifiers. AdaBoost is used to solve the following three fundamental problems: (1) learning effective features from a large feature set; (2) constructing weak classifiers, each of which is based on one of the selected features; (3) boosting the weak classifiers to construct a strong classifier. Viola and Jones make use of several techniques for effective computation of a large number of such features under varying scale and location which is important for real-time performance. Moreover, the cascade of strong classifiers which form cascade tree will make the computation even more efficient. Their system is the first real-time frontal-view face detector. However, their system still has some drawbacks. Since the detection results depend on weak classifiers, the detection results often have many false positives. To decrease the rate of false positives, it is compelled to increase the number of strong classifiers and Haar-like features in cascade tree, but this will cause a significant increase in the performance time, and detection rate can be decreased. Thus, to deal with the issue, we should combine AdaBoost with other machine learning techniques to achieve the same face detecting ratios but with the minimum number of false positives and the running time.

One of the popular methods having the same achievement as well is artificial neural networks (ANNs) . ANN is the term on the method to solve problems by simulating neuron’s activities. In detail, ANNs can be most adequately characterized as “computational models” with particular properties such as the ability to adapt or learn, to generalize, or to cluster or organize data, and which operation is based on parallel processing. However, many of the previously mentioned properties can be attributed to nonneural models. A hybrid approach combining AdaBoost and ANN is proposed to detect faces with the purpose of decreasing the performance time but still achieving the desired faces detecting rate.



b) How can one decide whether the network is overfitting or underfitting the data? Describe the process that one needs to follow.

**Answer:-** We can determine whether a predictive model is underfitting or overfitting the training data by looking at the prediction error on the training data and the evaluation data.



Your model is *underfitting* the training data when the model performs poorly on the training data. This is because the model is unable to capture the relationship between the input examples (often called X) and the target values (often called Y). Your model is *overfitting* your training data when you see that the model performs well on the training data but does not perform well on the evaluation data. This is because the model is memorizing the data it has seen and is unable to generalize to unseen examples.

Poor performance on the training data could be because the model is too simple (the input features are not expressive enough) to describe the target well. Performance can be improved by increasing model flexibility. To increase model flexibility, try the following:

* Add new domain-specific features and more feature Cartesian products, and change the types of feature processing used (e.g., increasing n-grams size)
* Decrease the amount of regularization used

If your model is overfitting the training data, it makes sense to take actions that reduce model flexibility. To reduce model flexibility, try the following:

* Feature selection: consider using fewer feature combinations, decrease n-grams size, and decrease the number of numeric attribute bins.
* Increase the amount of regularization used.

Accuracy on training and test data could be poor because the learning algorithm did not have enough data to learn from. You could improve performance by doing the following:

* Increase the amount of training data examples.
* Increase the number of passes on the existing training data.

c) Why are the neural networks weights initialised with random values?

Answer:-

Random initial weights might give you better results that would be somewhat closer to what a trained neural network should look like, but it might as well be the exact opposite of what it should be, while 0.5, or some other average for the range of reasonable weights' values,The initial weights in a neural network are initialized randomly because the gradient based methods commonly used to train neural networks do not work well when all of the weights are initialized to the same value. While not all of the methods to train neural networks are gradient based, most of them are, and it has been shown in several cases that initializing the neural network to the same value makes the network take much longer to converge on an optimum solution. Also, if you want to retrain your neural network because it got stuck in a local minima, it will get stuck in the same local minima. For the above reasons, we do not set the initial weights to a constant value.

**Question 4**

a) Describe the difference between supervised and unsupervised learning. Give an example of a supervised learning problem and an example of an unsupervised learning problem.

**Answer:-**

Supervised learning (SL) trains the machine using labeled data, so, the predictions can then be compared and adjusted accordingly. SL creates direct links between input and output data and is less computationally complex. A real world example that may be solved with SL is email spam detection. The inputs are the emails received to the inbox and the output is whether or not that email is spam. Unsupervised learning (UL) learns only on input data that is unlabeled. UL is more computationally complex and takes place in real time. A real world problem that can be solved with UL is a recommendation system on a platform such as Amazon. The inputs are the data about a user and the outputs are similar/recommended items.

b) Describe in detail the steps of the K-means algorithm. Make sure that you define the input to the algorithm, the output, and the dimensionality of all the variables that you use.

Answer:-

The k-means algorithm attempts to group similar items into K clusters/groups. Inputs to this algorithm are K and the input data.

The algorithm:

1.Place K points randomly into the object data space representing an initial group of centroids encompassing the data

2.Each object is assigned to the closest K

3.After all objects are assigned, positions of the K centroids are recalculated using coordinate descent

4.Steps 2 and 3 are repeated until the coordinates are optimised.

The K-Means algorithm works iteratively and makes K number of groups out of several data points. It starts with a set of data points (for which we don't have any label) and groups them into K groups. It works iteratively, and consists of two steps. It begins with a random initialisation about where the centres of the K clusters are and:

It assigns points to the nearest cluster and

It recalculates the cluster centres

until the algorithm reaches convergence (i.e., the cluster centres do not change).

Input:

N-point dataset D={x1,x2,...,xn}

K number of clusters

Cost function: find cluster centres μ(1:K)and cluster assignments c(1:N)so as to minimise the sum squared distances of points from assigned clusters.

EKM(D,μ1:K,c1:N)=i=1N(xi-μci)2

Each μk will be D-dimensional if each xi is also D-dimensional.

Initialise randomly K number of centres c1,c2,...,ck

Repeat:

("E-step") for i=[1:N]

labelsi = cluster centroid closest to xi

aka "find the cluster closest to each point (fix μ, minimise for c)"

("M-step") for k=[1:K]

c) After applying K-Means clustering on unlabelled data, a Machine Learning practitioner runs a linear classifier using as pseudolabels the cluster assignments. Why would he/she want to do that? What type of regularisation would you advice her/him to use and why?

Answer:-

After applying K-means clustering on unlabelled data with pseudo-labeling you have a set of labeled data as well as a set of unlabeled data. You first train a model on only the labeled data. You then use that initial data to classify (attach provisional labels to) the unlabeled data. You then feed both the labeled and unlabeled data back into your model training, (re-)fitting to both the known labels and the predicted labels. (Iterate this process, re-labeling with the updated model.)

The claimed benefits are that you can use the information about the structure of the unlabeled data to improve the model. A variation of the following figure is often shown, "demonstrating" that the process can make a more complex decision boundary based on where the (unlabeled) data lies. Because the Here, the learned cluster boundaries ***will not*** be the same, because clustering doesn't care for class labels, all it accounts for is transforming the feature space. The clustering generates a latent feature space, on which the classification boundary is learned, and this depends only on labelled data.Algorithms that do not perform any sort of clustering, I believe, will not be able to change their optimum based on the unlabelled data set.so he will use Lasso regularisation for this purpose because of removing the multicolinearinty among this dataset.