MACHINE LEARNING

1) R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?

Answer - R squared score is best

eg – 1 being best

0 being worst

0 means none of the data fit by the model

0.8 or o.65 r2_score means 80% or 65% of the data are fit by model

Ratio of RSS/TSS deducted in 1. If it is near to one then the errors are less(means not much gap between actual and predicted)

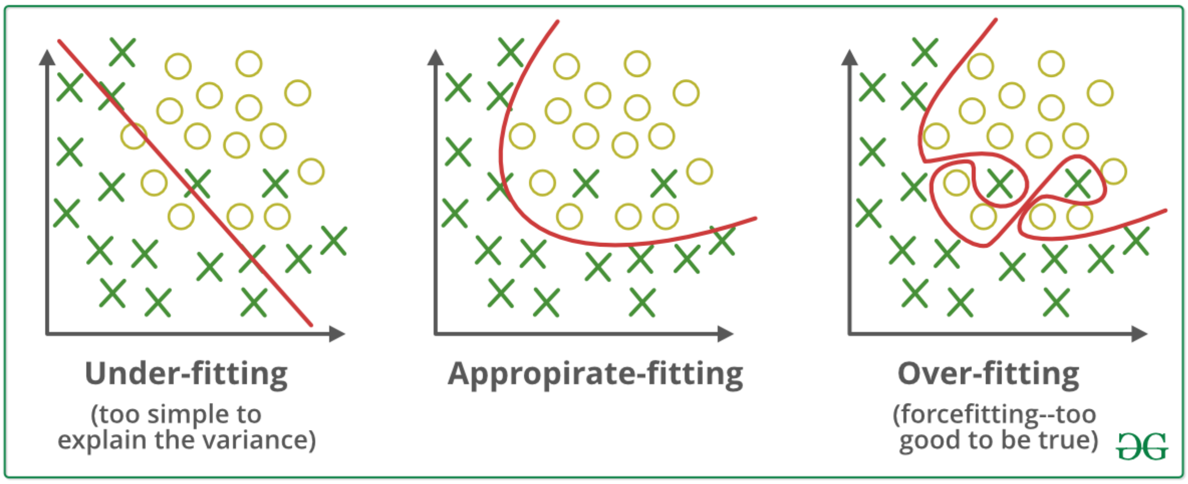
The residual sum of squares (RSS) is the absolute amount of explained variation, whereas R-squared is the absolute amount of variation as a proportion of total variation.

2) What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.

Answer - the **explained sum of squares** (**ESS**), alternatively known as the **model sum of squares** or **sum of squares due to regression** (**SSR** – not to be confused with the residual sum of squares (RSS) or sum of squares of errors), is a quantity used in describing how well a model, often a regression model, represents the data being modelled. In particular, the explained sum of squares measures how much variation there is in the modelled values and this is compared to the total sum of squares (TSS), which measures how much variation there is in the observed data, and to the residual sum of squares, which measures the variation in the error between the observed data and modelled values.

3) What is the need of regularization in machine learning?

Answer - **Overfitting** is a phenomenon that occurs when a Machine Learning model is constraint to training set and not able to perform well on unseen data.


Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting.
The commonly used regularization techniques are :

1. L1 regularization
2. L2 regularization
3. Dropout regularization

This article focus on L1 and L2 regularization.

A regression model which uses **L1 Regularization** technique is called **LASSO(Least Absolute Shrinkage and Selection Operator)** regression.
A regression model that uses **L2 regularization** technique is called **Ridge regression**.
Lasso Regression adds *“absolute value of magnitude”* of coefficient as penalty term to the loss function(L)

4) What is Gini–impurity index?
Answer- To make predictions it is necessary to start dividing the features starting from one feature and that feature where the other features are divided is root node or the node from where remaining nodes are divided is the root node

The remaining nodes are branch nodes

And the final node where decisions are being made are leaf nodes

To decide which feature will become root node and which one become branch nodes among all features (say 100), Decision tree uses some metrics and those are Entropy and Gini

These will priorities the features like which one become root node, which one become branch node and finally leaf node.

Entropy:
(Theoretical part is sufficient from interview point of view)
Try to find information from all features w.r.t label i.e., Information Gain. The feature giving high that will be the root node, the feature with second highest information gain will be the branch node and so on. The feature with least information gain is our label/output.

Gini index/impurity:
19:32Opposite to entropy, finds impurity in all the features w.r.t label. Whichever feature giving least impurity or index will be root node, feature with second least impurity will be first branch node and so on.

If features are categorical then metrics will be Entropy
If features are continuous then metric will be Gini index

Gini Impurity is a measurement used to build Decision Trees to determine how the features of a dataset should split nodes to form the tree. More precisely, the Gini Impurity of a dataset is a number between 0-0.5, which indicates the likelihood of new, random data being misclassified if it were given a random class label according to the class distribution in the dataset.

5) Are unregularized decision-trees prone to overfitting? If yes, why?
Answer - they are prone to this because they are very data intensive - that is, they examine the data in a lot of ways. At each node, they look at every possible split of every independent variable (sometimes they impose a rule of monotonicity - if the variable is continuous or ordinal).

Even with a relatively small number of variables, that can be a lot of things to examine, especially if one of
them is a categorical variable with more than a few levels

6) What is an ensemble technique in machine learning?
Answer - Ensemble learning is a technique in machine learning which takes the help of several base models and combines their output to produce an optimized model. This type of machine learning algorithm helps in improving the overall performance of the model. Here the base model which is most commonly used is the Decision tree classifier. A decision tree basically works on several rules and provides a predictive output, where the rules are the nodes and their decisions will be their children and the leaf nodes will constitute the ultimate decision. As shown in the example of a decision tree.
The above decision tree basically talks about whether a person/customer can be given a loan or not. One of the rules for loan eligibility yes is that if (income = Yes && Married = No) Then Loan = Yes so this is how a decision tree classifier works. We will be incorporating these classifiers as a multiple base model and combine their output to build one optimum predictive model. Figure 1.b shows the overall picture of an ensemble learning algorithm.

7) What is the difference between Bagging and Boosting techniques?
Answer - Bagging and boosting are two main types of ensemble learning methods, the main difference between these learning methods is the way in which they are trained. In bagging, weak learners are trained in parallel, but in boosting, they learn sequentially. This means that a series of models are constructed and with each new model iteration, the weights of the misclassified data in the previous model are increased. This redistribution of weights helps the algorithm identify the parameters that it needs to focus on to improve its performance. AdaBoost, which stands for “adaptative boosting algorithm,” is one of the most popular boosting algorithms as it was one of the first of its kind. Other types of boosting algorithms include XG Boost, Gradient Boost, and Brown Boost.
Another difference in which bagging and boosting differ are the scenarios in which they are used. For example, bagging methods are typically used on weak learners which exhibit high variance and low bias, whereas boosting methods are leveraged when low variance and high bias is observed.

8) What is out-of-bag error in random forests?
Answer - The out-of-bag error is the average error for each predicted outcome calculated using predictions from the trees that do not contain that data point in their respective bootstrap sample. This way, the Random Forest model is constantly being validated while being trained. Let us consider the *j*th decision tree ��� that has been fitted on a subset of the sample data. For every training observation or sample ��=(��,��) not in the

sample subset of ��� where �� is the set of features and �� is the target, we use ��� to predict the outcome �� for ��. The error can easily be computed as |��−��|.
The out-of-bag error is thus the average value of this error across all decision trees.

9) What is K-fold cross-validation?
Answer -Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.
Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.
It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.
The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

Importantly, each observation in the data sample is assigned to an individual group and stays in that group for the duration of the procedure. This means that each sample is given the opportunity to be used in the hold out set 1 time and used to train the model k-1 times.

10) What is hyper parameter tuning in machine learning and why it is done?
Answer- It is rare that a model will perform at the level you need for production just in the first instance. To find the right solution for your business problem, often you have to go through an iterative cycle. There are multiple pieces that come together to solve the intended machine learning puzzle. You may need to train and evaluate multiple models that include different data setup and algorithms, perform feature engineering a few times or even augment more data. This cycle also involves tweaking your model's **hyperparameters**.
Hyperparameters are the knobs or settings that can be tuned before running a training job to control the behavior of an ML algorithm. They can have a big impact on model training as it relates to training time, infrastructure resource requirements (and as a result cost), model convergence and model accuracy.

11) What issues can occur if we have a large learning rate in Gradient Descent?

Answer - The learning rate can seen as step size, η�. As such, gradient descent is taking successive steps in the direction of the minimum. If the step size η� is too large, it can (plausibly) "jump over" the minima we are trying to reach, i.e we overshoot. This can lead to osculations around the minimum or in some cases to outright divergence. It is important to note that the step gradient descent takes is a function of step size η� as well as the gradient values g�. If we are in a local minimum with zero gradient the algorithm will not update the parameters p� because the gradient is zero, similarly if p� is in a "steep slope", even a small η� will lead to a large update in p�'s values.

Particular for the case of divergence what happens is that as soon as an oversized step η� is taken from an initial point $p_{i=0}$��=0, the gradient descent algorithm lands to a point $p_{i=1}$��=1 that is worse than $p_{i=0}$��=0 in terms of cost. At this new but cost function-wise worse point $p_{i=1}$��=1, when recalculating the gradients, the gradient values are increased, so the next (hopefully corrective) step is even larger. Nevertheless if this next step leads to a point $p_{i=2}$��=2 with even larger error because we overshoot again, we can be led to use even larger gradient values, leading ultimately to a vicious cycle of ever increasing gradient values and "exploding coefficients" p_i��.

In the code you provided you might wish add a print(gradient(X, y, p)) statement in the param update function. If that is add, we can monitor the gradient in each iteration and see that in the case of a reasonably valued η� the gradient values slowly decrease while in the case of unreasonably large η� the gradient values get steadily larger and larger.

Advanced variants of gradient descent use the concept to adaptive learning rate, the optimisation algorithm Adadelta is a famous example of this. We might wish to play with a toy version of this notion by using a steadily decreasing step size. Assuming that we start with $\eta=\eta_0$�=�0, we can scale the step size η_t�� used for the t� iteration according to: $\eta_t=\eta_{0t}$��=�0�. Notice you if we adaptively decrease η� we need to start with a reasonably large η_0�0 (say 1.0 for your example).

12) Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

Answer –

Advantages:

- Logistic Regression is a simple and easy-to-implement algorithm.
- It is fast and efficient for small datasets.
- It can handle both continuous and categorical independent variables.
- Logistic Regression models are easy to interpret, as the coefficients of the independent variables indicate their effect on the dependent variable.

Disadvantages:

- Logistic Regression is not suitable for complex relationships between the dependent variable and independent variables.
- It assumes that the relationship between the dependent variable and independent variables is linear, which may not always be the case.
- Logistic Regression may not perform well on highly imbalanced datasets.

Where to use:

- Logistic Regression is ideal for binary classification problems where the goal is to predict one of two possible outcomes.
- It is also a good starting point for more complex classification problems, as it provides a baseline for comparison.

Where to not use:

- Logistic Regression is not suitable for complex non-linear relationships between the dependent variable and independent variables.
- It is also not recommended for multi-class classification problems, as it can only handle binary classification.

13) Differentiate between Adaboost and Gradient Boosting.

Answer –AdaBoost

AdaBoost or Adaptive Boosting is the first Boosting ensemble model. The method automatically adjusts its parameters to the data based on the actual performance in the current iteration. Meaning, both the weights for re-weighting the data and the weights for the final aggregation are re-computed iteratively.

In practice, this boosting technique is used with simple classification trees or stumps as base-learners, which resulted in improved performance compared to the classification by one tree or other single base-learner.

Gradient Boosting

Gradient Boost is a robust machine learning algorithm made up of Gradient descent and Boosting. The word 'gradient' implies that you can have two or more derivatives of the same function. Gradient Boosting has three main components: additive model, loss function and a weak learner.

The technique yields a direct interpretation of boosting methods from the perspective of numerical optimisation in a function space and generalises them by allowing optimisation of an arbitrary loss function.

The Comparison

Loss Function:

The technique of Boosting uses various loss functions. In case of Adaptive Boosting or AdaBoost, it minimises the exponential loss function that can make the algorithm sensitive to the outliers. With Gradient Boosting, any differentiable loss function can be utilised. Gradient Boosting algorithm is more robust to outliers than AdaBoost.

Flexibility

AdaBoost is the first designed boosting algorithm with a particular loss function. On the other hand, Gradient Boosting is a generic algorithm that assists in searching the approximate solutions to the additive modelling problem. This makes Gradient Boosting more flexible than AdaBoost.

Benefits

AdaBoost minimises loss function related to any classification error and is best used with weak learners. The method was mainly designed for binary classification problems and can be utilised to boost the performance of decision trees. Gradient Boosting is used to solve the differentiable loss function problem. The technique can be used for both classification and regression problems.

Shortcomings

In the case of Gradient Boosting, the shortcomings of the existing weak learners can be identified by gradients and with AdaBoost, it can be identified by high-weight data points.

Wrapping Up

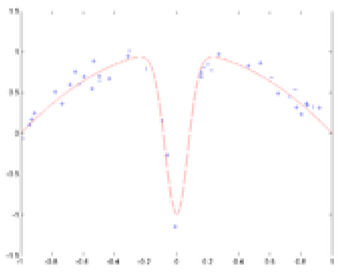
Though there are several differences between the two boosting methods, both the algorithms follow the same path and share similar historic roots. Both the algorithms work for boosting the performance of a simple base-learner by iteratively shifting the focus towards problematic observations that are challenging to predict.

In the case of AdaBoost, the shifting is done by up-weighting observations that were misclassified before, while Gradient Boosting identifies the difficult observations by large residuals computed in the previous iterations.

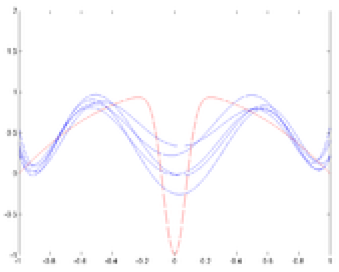
14) What is bias-variance trade off in machine learning?

Answer - In statistics and machine learning, the **bias–variance tradeoff** is the property of a model that the variance of the parameter estimated across samples can be reduced by increasing the bias in the estimated parameters. The **bias–variance dilemma** or **bias–variance problem** is the conflict in trying to simultaneously minimize these two sources of error that prevent supervised learning algorithms from generalizing beyond their training set:[1][2]

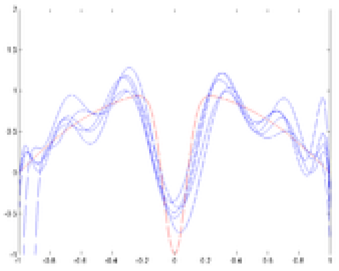
- The *bias* error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).
- The *variance* is an error from sensitivity to small fluctuations in the training set. High variance may result from an algorithm modeling the random noise in the training data (overfitting).



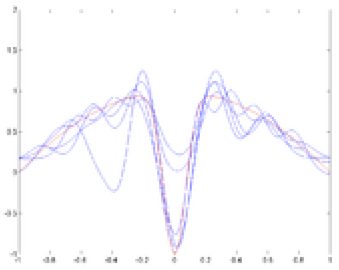
Function and noisy data



Spread=5

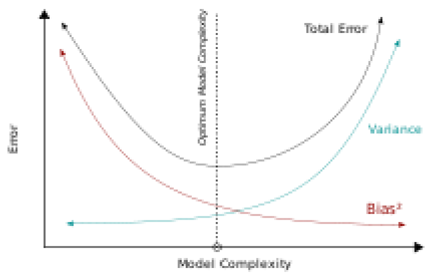


Spread=1



Spread=0.1

A function (red) is approximated using radial basis functions (blue). Several trials are shown in each graph. For each trial, a few noisy data points are provided as a training set (top). For a wide spread (image 2) the bias is high: the RBFs cannot fully approximate the function (especially the central dip), but the variance between different trials is low. As spread decreases (image 3 and 4) the bias decreases: the blue curves more closely approximate the red. However, depending on the noise in different trials the variance between trials increases. In the lowermost image the approximated values for x=0 varies wildly depending on where the data points were located.



Bias and variance as function of model complexity

The **bias–variance decomposition** is a way of analyzing a learning algorithm's expected generalization error with respect to a particular problem as a sum of three terms, the bias, variance, and a quantity called the *irreducible error*, resulting from noise in the problem itself.

15) . Give short description each of Linear, RBF, Polynomial kernels used in SVM

Answer - VM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example ***linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.***

Introduce Kernel functions for sequence data, graphs, text, images, as well as vectors. The most used type of kernel function is **RBF.** Because it has localized and finite response along the entire x-axis.

The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces.

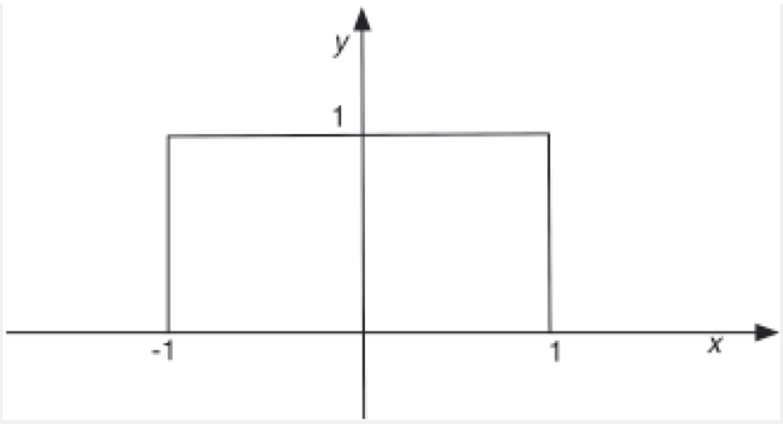
4. **Kernel Rules**

Define kernel or a window function as follows:

$$K(\overline{x}) = \begin{matrix} 1 & \text{if} \left\| \overline{x} \right\| \leq 1 \\ 0 & \text{otherwise} \end{matrix}$$

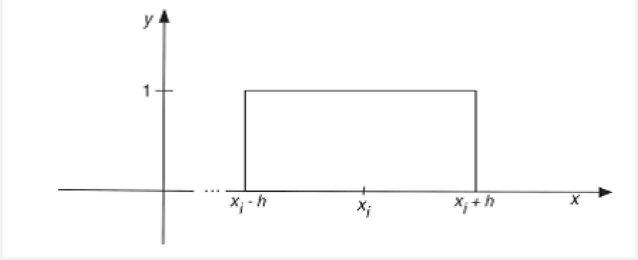
Kernel or a window function

This value of this function is 1 inside the closed ball of radius 1 centered at the origin, and 0 otherwise . As shown in the figure below:



Kernel or a window function

For a fixed xi, the function is K(z-xi)/h) = 1 inside the closed ball of radius h centered at xi, and 0 otherwise as shown in the figure below:



Kernel or a window function

So, by choosing the argument of K(·), you have moved the window to be centered at the point xi and to be of radius h.

4. Examples of SVM Kernels

Let us see some common kernels used with SVMs and their uses:

4.1. Polynomial kernel

It is popular in image processing.
Equation is:

$$k(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$$

Polynomial kernel equation

where d is the degree of the polynomial.

4.2. Gaussian kernel

It is a general-purpose kernel; used when there is no prior knowledge about the data.
Equation is:

$$k(x, y) = \exp\left(-\frac{||x - y||^2}{2\sigma^2}\right)$$

Gaussian kernel equation

4.3. Gaussian radial basis function (RBF)

It is a general-purpose kernel; used when there is no prior knowledge about the data.
Equation is:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma||\mathbf{x_i} - \mathbf{x_j}||^2)$$

Gaussian radial basis function (RBF)

, for:

$$\gamma > 0$$

Gaussian radial basis function (RBF)

Sometimes parametrized using:

$$\gamma = 1/2\sigma^2$$

Gaussian radial basis function (RBF)

4.4. Laplace RBF kernel

It is general-purpose kernel; used when there is no prior knowledge about the data. Equation is:

$$k(x,y) = \exp\left(-\frac{||x-y||}{\sigma}\right)$$

Laplace RBF kernel equation