MACHINE LEARNING

MACHINE LEARNING ASSIGNMENT - 5 Q1 to Q15 are subjective answer type questions, Answer them briefly.

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?

Ans: Both R-squared and Residual Sum of Squares (RSS) are measures of goodness of fit in regression analysis, but they capture different aspects of the model's performance.

R-squared (also known as the coefficient of determination) measures the proportion of variation in the dependent variable that is explained by the independent variables in the model. In other words, it indicates how well the model fits the data, with values ranging from 0 to 1. Higher R-squared values indicate a better fit, as they mean that a larger proportion of the variation in the dependent variable is explained by the independent variables in the model.

On the other hand, RSS measures the total sum of squared differences between the actual values of the dependent variable and the predicted values by the model. It represents the amount of unexplained variation in the data, and lower RSS values indicate a better fit, as they mean that the model is able to explain more of the variation in the data.

Therefore, both measures are useful in evaluating the goodness of fit of a model, but they serve different purposes. R-squared is a useful measure to assess the overall fit of the model and to compare different models, while RSS is useful to identify the degree of the error in the model's predictions.

In general, a good model should have both a high R-squared value and a low RSS value, indicating that it explains a large proportion of the variation in the dependent variable and has a low degree of error in its predictions. However, in some cases, one measure may be more important than the other, depending on the research question and the nature of the data being analysed.

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.

Ans: The total sum of squares (TSS) measures how much variation there is in the observed data.

The explained sum of squares (ESS) is the sum of the squares of the deviations of the predicted values from the mean value of a response variable, in a standard regression model.

The residual sum of squares (RSS) measures the variation in the error between the observed data and modelled values.

In statistics, the values for the residual sum of squares (RSS) and the total sum of squares (TSS) are oftentimes compared to each other.

TSS = ESS + RSS, **where TSS is Total Sum of Squares, ESS is Explained Sum of Squares and RSS is Residual Sum of Square's**.

- The residual sum of squares (RSS) measures the level of variance in the error term, or residuals, of a regression model.
- The smaller the residual sum of squares, the better your model fits your data; the greater the residual sum of squares, the poorer your model fits your data.
- A value of zero means your model is a perfect fit.
- Statistical models are used by investors and portfolio managers to track an investment's price and use that data to predict future movements.
- The RSS is used by financial analysts in order to estimate the validity of their econometric models.

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

Ans: Regularization is a set of techniques that can **prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model when facing completely new data from the problem domain**.

Regularization refers to techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting.

A scenario where the machine learning model tries to learn from the details along with the noise in the data and tries to fit each data point on the curve is called Overfitting.

In the figure depicted below, we can see that the model is fit for every point in our data. If given new data, the model curves may not correspond to the patterns in the new data, and the model cannot predict very well in it.

Figure 1: Overfitted Model

Conversely, in a scenario where the model has not been allowed to look at our data a sufficient number of times, the model won't be able to find patterns in our test dataset. It will not fit properly to our test dataset and fail to perform on new data too.

A scenario where a machine learning model can neither learn the relationship between variables in the testing data nor predict or classify a new data point is called Underfitting.

The below diagram shows an under-fitted model. We can see that it has not fit properly to the data given to it. It has not found patterns in the data and has ignored a large part of the dataset. It cannot perform on both known and unknown data.

Figure 2: Underfitted Model

Regularization in Machine Learning

Regularization refers to techniques that are used to calibrate machine learning models in order to minimize the adjusted loss function and prevent overfitting or underfitting.

Figure 5: Regularization on an over-fitted model

Using **Regularization,** we can fit our machine learning model appropriately on a given test set and hence **reduce the errors in it.**

Need of Regularization in Machine learning:

In Machine Learning we often divide the dataset into training and test data, the algorithm while training the data can either
1) learn the data too well, even the noises which is called over fitting

2) do not learn from the data, cannot find the pattern from the data which is called under fitting.

Now, both over fitting and underfitting are problems one need to address while building models.

Regularization in Machine Learning is used to minimize the problem of overfitting, the result is that the model generalizes well on the unseen data once overfitting is minimized.
To avoid overfitting, regularization discourages learning a more sophisticated or flexible model. Regularization will try to minimize a loss function by inducing penalty.
For Example
The residual sum of squares is our optimization function or loss function in simple linear regression (RSS).

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Here,

y is the dependent variable,
x_1, x_2, x_3,..... x_n are independent variables.
b_0, b_1 ,b_2..... b_n, are the coefficients estimates for different variables of x, these can also be called weights or magnitudes
Regularization will shrink these coefficients towards Zero,
Minimizing the loss means less error and model will be a good fit.

The way regularization can be done is by
1) RIDGE also known as L-2 Regularization
2) LASSO (Least Absolute and Selection Operator) also known as L-1 Regularization

4. What is Gini–impurity index?

Ans: Gini Index, also known as Gini impurity, **calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly**. If all the elements are linked with a single class then it can be called pure.

5. Are unregularized decision-trees prone to overfitting? If yes, why?

Ans: Decision trees are prone to overfitting, especially when a tree is particularly deep.

6. What is an ensemble technique in machine learning?

Ans: Ensemble learning refers to algorithms that combine the predictions from two or more models. Although there is nearly an unlimited number of ways that this can be achieved, there are perhaps three classes of ensemble learning techniques that are most commonly discussed and used in practice.

Stacking: is an Ensemble Learning technique that uses predictions from multiple models (for example, decision tree, knn or svm, etc.) to form a new efficient predictive model.

In machine learning, boosting is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones.

Gradient Boosting or GBM is another ensemble machine learning algorithm that works for both regression and classification problems. GBM uses the boosting technique, combining a number of weak learners to form a strong learner.

7. What is the difference between Bagging and Boosting techniques?

Ans:(Bagging and **boosting** are both ensembles learning methods in machine learning)

(1) Bagging is a method of merging the same type of predictions. Boosting is a method of merging different types of predictions.

(2) Bagging decreases variance, not bias, and solves over-fitting issues in a model. Boosting decreases bias, not variance.

(3) In Bagging, each model receives an equal weight. In Boosting, models are weighed based on their performance.

(4) Models are built independently in Bagging. New models are affected by a previously built model's performance in Boosting.

(5) In Bagging, training data subsets are drawn randomly with a replacement for the training dataset. In Boosting, every new subset comprises the elements that were misclassified by previous models.

(6) Bagging is usually applied where the classifier is unstable and has a high variance. Boosting is usually applied where the classifier is stable and simple and has high bias.

8. What is out-of-bag error in random forests?

Ans: 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 DTj that has been fitted on a subset of the sample data. For every training observation or sample z(i)*=(x(I),y(I))* not in the sample subset of DTj where x(I) is the set of features and y(I) is the target, we use DTj to predict the outcome o(I)for x(I).

The error can easily be computed as *|o(I)-x(I)|*.
The out-of-bag error is thus the average value of this error across all decision trees.

Out-of-bag (**OOB**) **error**, also called **out-of-bag estimate**, is a method of measuring the prediction error of random forests, boosted decision trees, and other machine learning models utilizing bootstrap aggregating (bagging). Bagging uses subsampling with replacement to create training samples for the model to learn from. OOB error is the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.

The prediction error on that sample is known as the out-of-bag error. The OOB score is the number of **correctly predicted data on OOB samples** taken for validation. It means that the more the error bottom model does, the Less the OOB score for the bottom model.

9. What is K-fold cross-validation?

Ans: K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data. K refers to the number of groups the data sample is split into. For example, if you see that the k-value is 5, we can call this a 5-fold cross-validation. Each fold is used as a testing set at one point in the process. **K-fold Cross-Validation Process:**

1. Choose your k-value
2. Split the dataset into the number of k folds.
3. Start off with using your k-1 fold as the test dataset and the remaining folds as the training dataset
4. Train the model on the training dataset and validate it on the test dataset
5. Save the validation score
6. Repeat steps 3 – 5 but changing the value of your k test dataset. So, we chose k-1 as our test dataset for the first round, we then move onto k-2 as the test dataset for the next round.
7. By the end of it you would have validated the model on every fold that you have.
8. Average the results that were produced in step 5 to summarize the skill of the model.

10. What is hyper parameter tuning in machine learning and why it is done?

Ans: Hyperparameter tuning is an essential part of controlling the behaviour of a machine learning model. If we don’t correctly tune our hyperparameters, our estimated model parameters produce suboptimal results, as they don’t minimize

the loss function. This means our model makes more errors. In practice, key indicators like the accuracy or the confusion matrix will be worse.

Parameters in a machine learning algorithm learns or estimates model parameters for the given data set, then continues updating these values as it continues to learn. After learning is complete, these parameters become part of the model. For example, each weight and bias in a neural network is a parameter.

Hyperparameters, on the other hand, are specific to the algorithm itself, so we can't calculate their values from the data. We use hyperparameters to calculate the model parameters. Different hyperparameter values produce different model parameter values for a given data set.

Note that the learning algorithm optimizes the loss based on the input data and tries to find an optimal solution within the given setting. However, hyperparameters describe this setting exactly.

For instance, if we work on natural language processing (NLP) models, we probably use neural networks, support-vector machines (SVMs), Bayesian networks, and Extreme Gradient Boosting (XGB) for tuning parameters.

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

Ans: Deep learning neural networks are trained using the stochastic gradient descent algorithm.

Stochastic gradient descent is an optimization algorithm that estimates the error gradient for the current state of the model using examples from the training dataset, then updates the weights of the model using the back-propagation of errors algorithm, referred to as simply backpropagation.

The amount that the weights are updated during training is referred to as the step size or the "*learning rate*."

Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more training epochs given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.

Because of this effect, a learning rate that is too large **takes longer to train**, because it is continually overshooting its objective and “unlearning” what it has learned, thus requiring expensive backtracking or causing unproductive oscillations.

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

Ans: No, logistic regression only forms linear decision surface

Logistic regression is a very commonly used method for predicting a target label from structured tabular data. Logistic Regression has traditionally been used as a linear classifier, i.e., when the classes can be separated in the feature space by linear boundaries. That can be remedied however if we happen to have a better idea as to the shape of the decision boundary…

13. Differentiate between Adaboost and Gradient Boosting.

Ans: Comparison between AdaBoost and Gradient Boost

S.No	Adaboost	Gradient Boost
1	An additive model where shortcomings of previous models are identified by high-weight data points.	An additive model where shortcomings of previous models are identified by the gradient.
2	The trees are usually grown as decision stumps.	The trees are grown to a greater depth usually ranging from 8 to 32 terminal nodes.
3	Each classifier has different weights assigned to the final prediction based on its performance.	All classifiers are weighed equally, and their predictive capacity is restricted with learning rate to increase accuracy.
4	It gives weights to both classifiers and observations thus capturing maximum variance within data.	It builds trees on previous classifier's residuals thus capturing variance in data.

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

Ans: The bias-variance trade-off is the tension between bias and variance in ML models. Biased models fail to capture the true trend, resulting in underfitting, whereas low-bias high-variance models likely result in overfitting.

Bias error is technically defined as an error that arises between average model prediction and the ground truth. Low bias implies fewer assumptions about the target function.

High bias implies more assumptions about the target function.

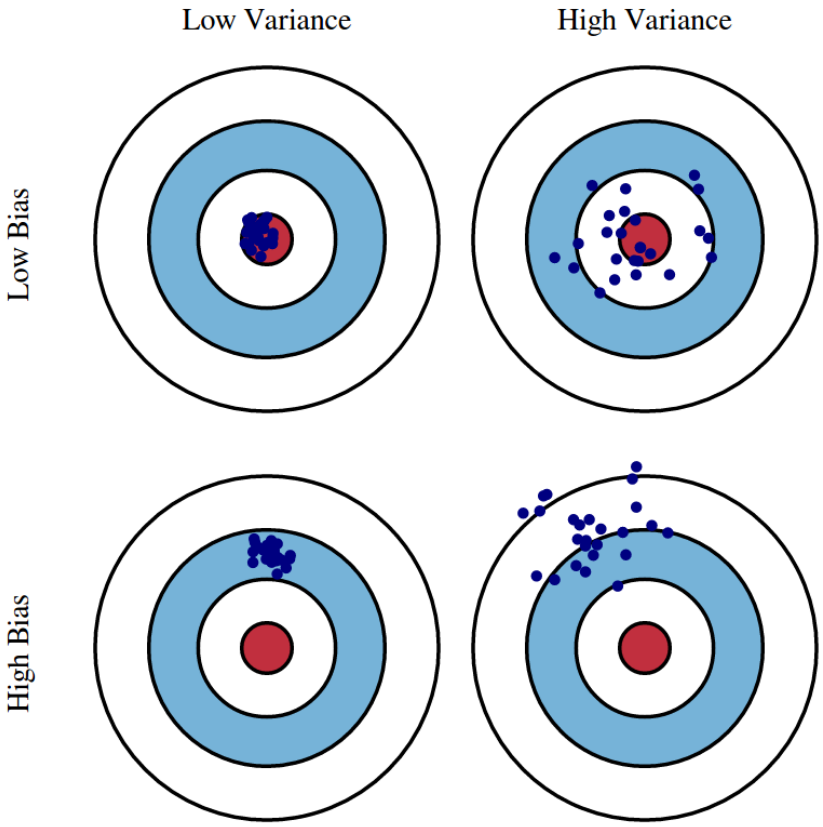
Variance refers to what extent the predictions of target or mapping functions change when subject to different datasets.

The bias-variance trade-off is inescapable – there is no avoiding the tension between the two:

- Increasing the bias will always decrease the variance.
- Increasing the variance will always decrease the bias.

In an ideal world, a model would accurately capture the regularities and trends of its training data but generalise well to training sets or unseen data. Here, the model would have low bias and low variance.

However, this is usually not the case, and different algorithms tend to fall in one of the two camps, i.e., linear models risk high bias and low variance. In contrast, nonlinear models risk high variance and low bias. There are positives to glean from both bias and variance, so long as they're balanced for the problem space.

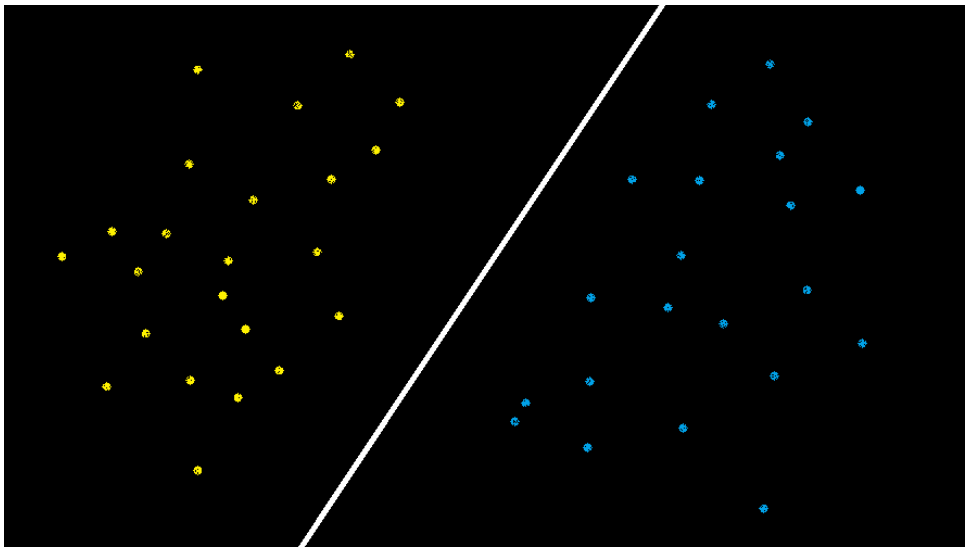


Bias-Variance Trade-off

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

Ans: Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. One of the examples where there are a lot of features, is **Text**

Classification, as each alphabet is a new feature. So we mostly use Linear Kernel in Text Classification.



In the above image, there are two set of features "**Blue**" features and the "**Yellow**" Features. Since these can be easily separated or in other words, they are linearly separable, so the Linear Kernel can be used here.

Advantages of using Linear Kernel:

1. Training a SVM with a Linear Kernel is **Faster** than with any other Kernel.

2. When training a SVM with a Linear Kernel, only the optimisation of the **C Regularisation** parameter is required. On the other hand, when training with other kernels, there is a need to optimise the ***γ*** parameter which means that performing a grid search will usually take more time.

RBF kernels

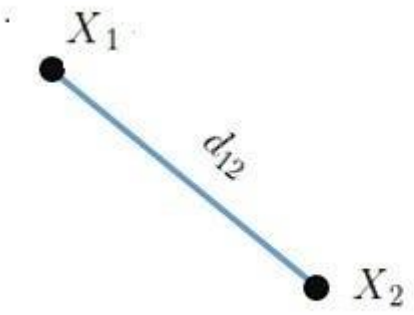
RBF kernels are the most generalized form of kernelization and is one of the most widely used kernels due to its similarity to the Gaussian distribution. The RBF kernel function for two points X_1 and X_2 computes the similarity or how close they are to each other. This kernel can be mathematically represented as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2\sigma^2})$$

where,

1. ‘σ’ is the variance and our hyperparameter
2. $||X_1 - X_2||$ is the Euclidean (L_2-norm) Distance between two points X_1 and X_2

Let d_{12} be the distance between the two points X_1 and X_2, we can now represent d_{12} as follows:



$$d_{12} = ||X_1 - X_2||^2$$

Fig 2: Distance between two points in space [Image by Author]

The kernel equation can be re-written as follows:

$$K(X_1, X_2) = exp(-\frac{d_{12}}{2\sigma^2})$$

The maximum value that the RBF kernel can be is 1 and occurs when d_{12} is 0 which is when the points are the same, i.e. $X_1 = X_2$.

9. When the points are the same, there is no distance between them and therefore they are extremely similar
10. When the points are separated by a large distance, then the kernel value is less than 1 and close to 0 which would mean that the points are dissimilar

Distance can be thought of as an equivalent to dissimilarity because we can notice that when distance between the points increases, they are less similar.

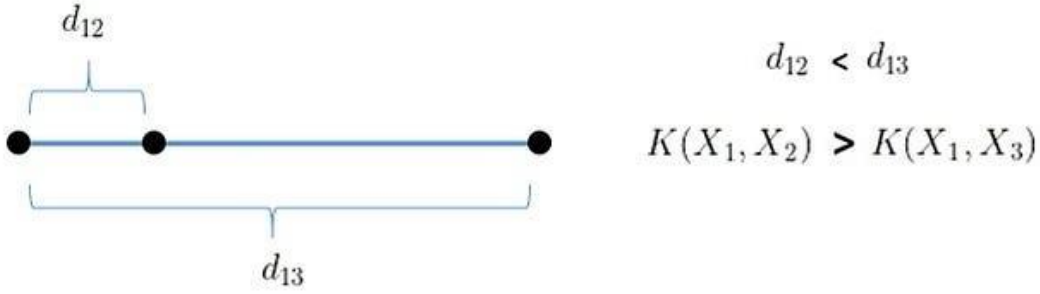


Fig 3: Similarity decreases as distance increases [Image by Author]

It is important to find the right value of 'σ' to decide which points should be considered similar and this can be demonstrated on a case-by-case basis.

a] σ = 1

When σ = 1, $\sigma^2 = 1$ and the RBF kernel's mathematical equation will be as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2})$$

The curve for this equation is given below and we can notice that as the distance increases, the RBF Kernel decreases exponentially and is 0 for distances greater than 4.

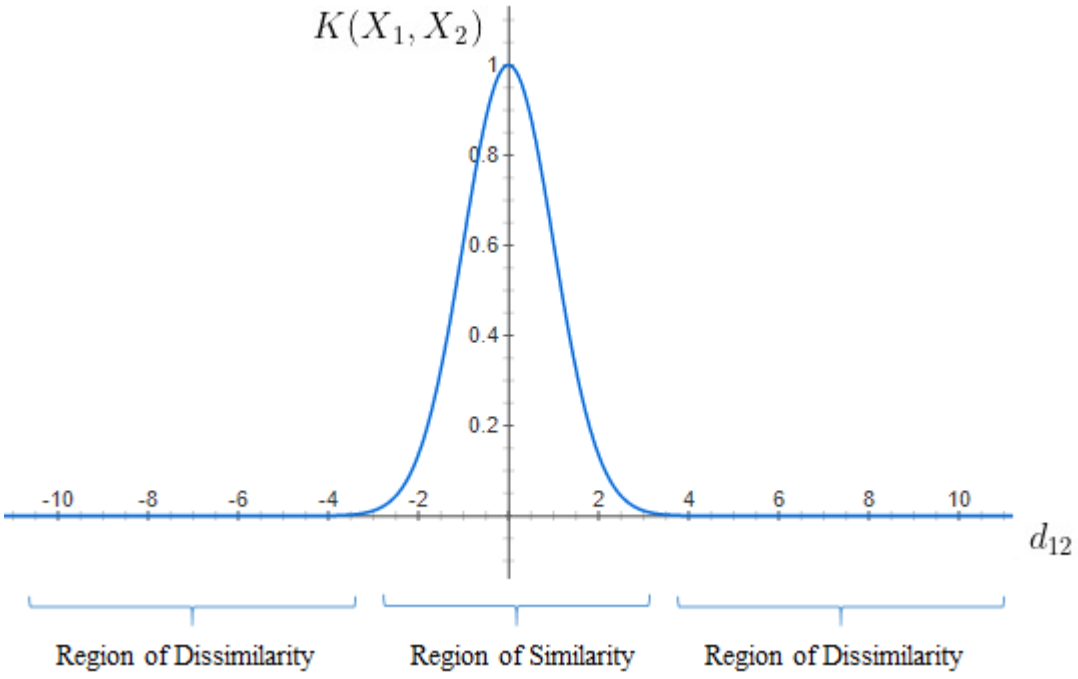


Fig 4: RBF Kernel for σ = 1 [Image by Author]

11. We can notice that when $d_{12} = 0$, the similarity is 1 and as d_{12} increases beyond 4 units, the similarity is 0
12. From the graph, we see that if the distance is below 4, the points can be considered similar and if the distance is greater than 4 then the points are dissimilar

b] σ = 0.1

When σ = 0.1, $\sigma^2 = 0.01$ and the RBF kernel's mathematical equation will be as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{0.01})$$

The width of the Region of Similarity is minimal for σ = 0.1 and hence, only if points are extremely close they are considered similar.

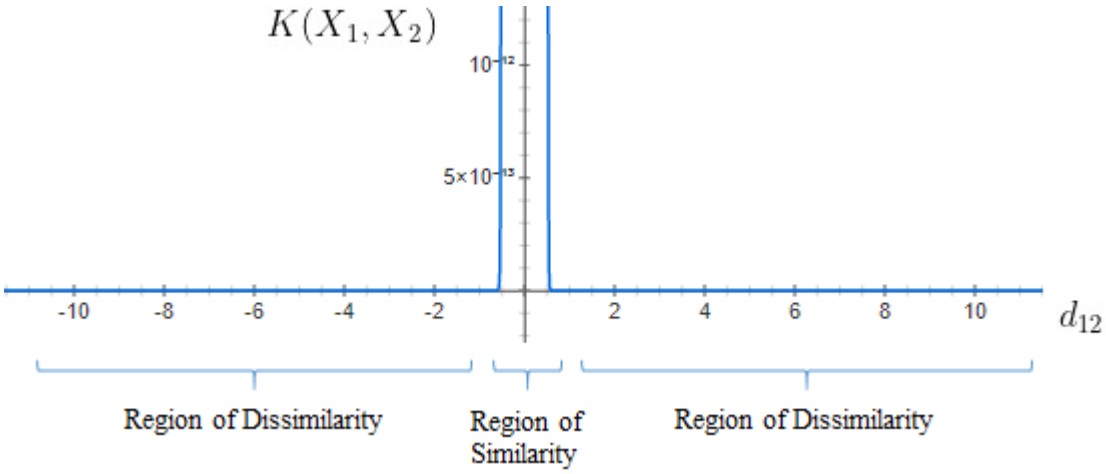


Fig 4: RBF Kernel for σ = 0.1 [Image by Author]

13. We see that the curve is extremely peaked and is 0 for distances greater than 0.2
14. The points are considered similar only if the distance is less than or equal to 0.2

b] σ = 10

When σ = 10, σ^2 = 100 and the RBF kernel's mathematical equation will be as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{100})$$

The width of the Region of Similarity is large for σ = 100 because of which the points that are farther away can be considered to be similar.

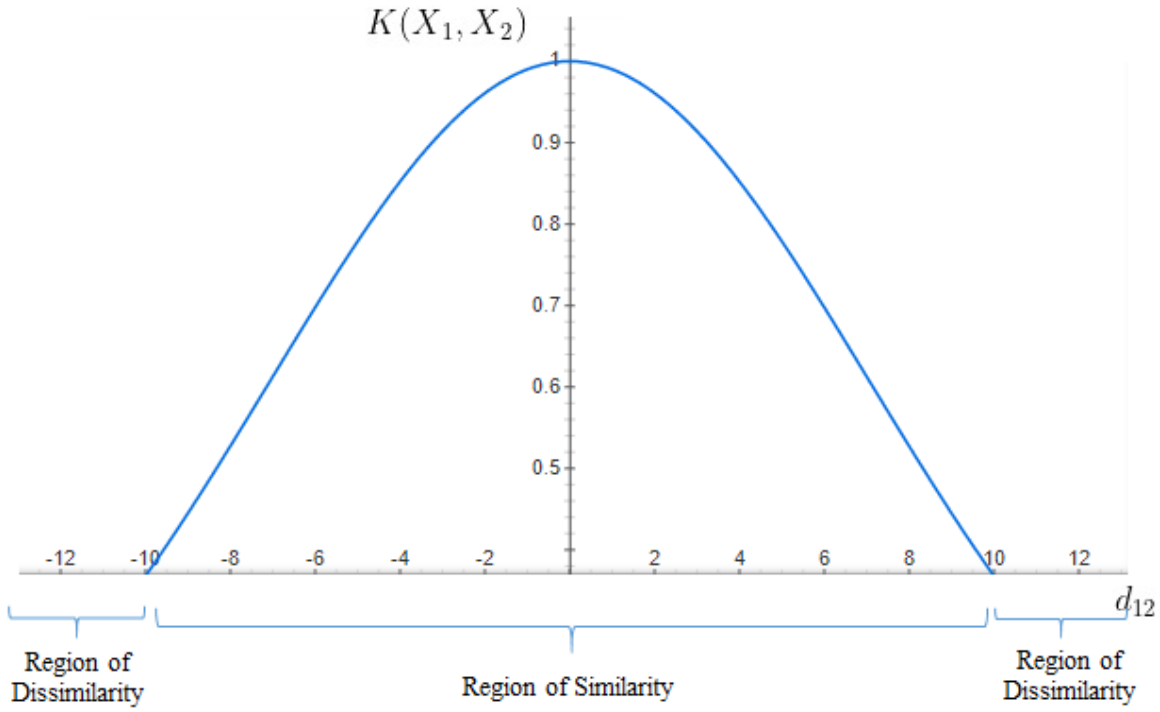


Fig 5: RBF Kernel for σ = 10 [Image by Author]

The width of the curve is large

The points are considered similar for distances up to 10 units and beyond 10 units they are dissimilar

A polynomial kernel

A polynomial kernel is a kind of SVM kernel that uses a polynomial function to map the data into a higher-dimensional space. It does this by taking the dot product of the data points in the original space and the polynomial function in the new space.

In a polynomial kernel for SVM, the data is mapped into a higher-dimensional space using a polynomial function. The dot product of the data points in the original space and the polynomial function in the new space is then taken. The polynomial kernel is often used in SVM classification problems where the data is not linearly separable. By mapping the data into a higher-dimensional space, the polynomial kernel can sometimes find a hyperplane that separates the classes.

The polynomial kernel has a number of parameters that can be tuned to improve its performance, including the degree of the polynomial and the coefficient of the polynomial.

For degree d polynomials, the polynomial kernel is defined as:

$$K(x_1, x_2) = (x_1^T x_2 + c)^d$$

where **c** is a constant and **x1** and **x2** are vectors in the original space.

The parameter **c** can be used to control the trade-off between the fit of the training data and the size of the margin. A large **c** value will give a low training error but may result in overfitting. A small **c** value will give a high training error but may result in underfitting. The degree **d** of the polynomial can be used to control the complexity of the model. A high degree **d** will result in a more complex model that may overfit the data, while a low degree **d** will result in a simpler model that may underfit the data.

When a dataset is given containing features x1 and x2, the equation can be transformed as:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1 x_2 \\ \\ x_1 x_2 & x_2^2 \end{bmatrix}$$

The important terms we need to note are **x1, x2, x1^2, x2^2**, and **x1 * x2.** When finding these new terms, the non-linear dataset is converted to another dimension that has features **x1^2, x2^2**, and **x1 * x2.**