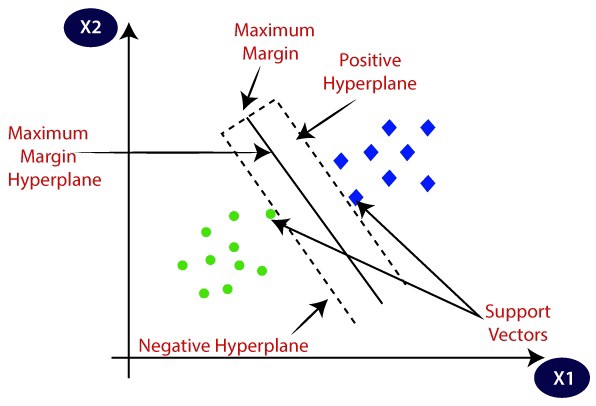
**What are Support Vector Machines (SVMs)?**

**Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.** The objective of the **support vector machine** algorithm is to find a **hyperplane** in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

SVM is a supervised machine learning algorithm that works on both classification and regression problem statements.

 For classification problem statements, it tries to differentiate data points of different classes by finding a hyperplane that maximizes the margin between the classes in the training data.

 In simple words, SVM tries to choose the hyperplane which separates the data points as widely as possible since this margin maximization improves the model’s accuracy on the test or the unseen data.



Scatter chart

Description automatically generated

### What is the basic principle of a Support Vector Machine?

Text, letter

Description automatically generated

Diagram

Description automatically generated with low confidence

### What are hard margin and soft Margin SVMs?

* **Hard-Margin** SVMs have *linearly separable* training data. No data points are allowed in the margin areas. This type of linear classification is known as Hard margin classification.
* **Data point must be beyond the margin**
* **Soft-Margin** SVMs have training data that are not *linearly separable*. Margin violation means choosing a hyperplane, which can allow some data points to stay either in between the margin area or on the incorrect side of the hyperplane.
* **Data point can be in the margin**

### Chart, radar chart, scatter chart Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

If a point Xi satisfies the equation ***Yi(WT\*Xi +b) ≥ 1,***then Xi is correctly classified else incorrectly classified. So we can see that if the points are linearly separable then only our hyperplane is able to distinguish between them and if any outlier is introduced then it is not able to separate them. So these type of SVM is called***hard margin SVM****(since we have very strict constraints to correctly classify each and every data point).*

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

To overcome this, we introduce a term***( ξ )***(pronounced as Zeta)



*if ξi= 0, the points can be considered as correctly classified.*

*if ξi> 0 , Incorrectly classified points.*

Hinge Loss is a loss function which penalises the SVM model for inaccurate predictions.

**Hinge loss** is a loss function used for training classifiers. Hinge loss is most notably used for soft-margin SVMs. The hinge loss penalizes the SVM model for inaccurate predictions (misclassifications).

If**Yi(WT\*Xi +b) ≥ 1**, hinge loss is ‘**0**’ i.e the points are correctly classified. When

**Yi(WT\*Xi +b) < 1**, then hinge loss increases massively.

As **Yi(WT\*Xi +b)**increases with every misclassified point, the upper bound of hinge loss {**1- Yi(WT\*Xi +b)**} also increases exponentially.

Hence, the points that are farther away from the decision margins have a greater loss value, thus penalising those points.

A picture containing timeline

Description automatically generated

We can formulate hinge loss as **max[0, 1- Yi(WT\*Xi +b)]**

#### **What affects the decision boundary in SVMs?**

Decision boundaries are affected only by the support vectors; instances located on the margin of the SVM. Instances not lying on the margin of the hyperplane do not affect the decision boundary.

#### **What technique can be used to solve the optimization problem cast by Support Vector Machines?**

The problem with Hinge loss is that it is not differentiable. Since we can’t take the derivative of it, we can’t find the gradient of the function. The technique to mitigate this issue is called Sub-Gradient Descent which in particular we can use PEGASOS (**primal estimated sub-gradient solver**) for SVMs.

#### **What is the benefit of using sub-gradient descent?**

sing the sub-gradient, we get a guaranteed minimum just like with classifiers like logistic regression. We would be able to update weights just like we would for gradient descent.

#### **Are SVMs sensitive to Feature Scaling?**

Yes, SVMs are sensitive to Feature Scaling as they utilize input data to find the margins around hyperplanes. They also end up getting biased for the variance in high values.

#### **What is the function of the Hyperparameter C?**

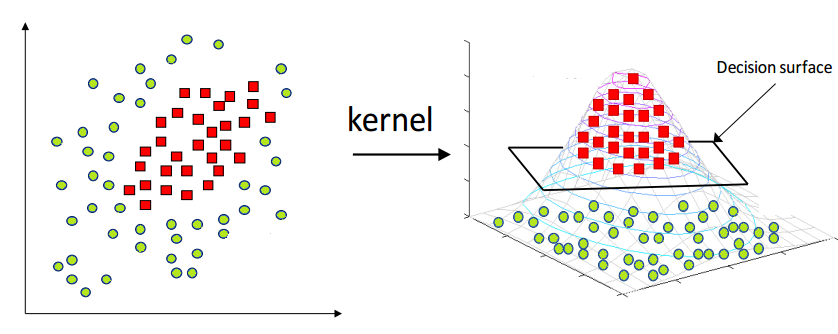
It helps us tune how much we want to penalize points lying either inside of our margin or complete misclassifications. It is recommended to use cross-validation to find out the best value of C

#### **What is the relationship between Slack and Margin?**

Margin is the space between the hyperplane and support vectors. In the case of soft margin Support vectors, margin includes the slack. Slack is the relaxing of the constraint that all example must lie outside the margin which creates the soft-margin SVM

**What’s the “kernel trick” and how is it useful?**

Explanation: Earlier we have discussed applying SVM on linearly separable data but it is very rare to get such data. Here, kernel trick plays a huge role. The idea is to map the non-linear separable data-set into a higher dimensional space where we can find a hyperplane that can separate the samples.



It reduces the complexity of finding the mapping function. So, **Kernel function defines the inner product in the transformed space.**Application of the kernel trick is not limited to the SVM algorithm. Any computations involving the dot products (x, y) can utilize the kernel trick.

The function which takes input vectors in the original states and returns the dot product in features space is called kernel trick

## What is the difference between Classification and Regression when using SVM?

Chart, radar chart

Description automatically generated

* These are functions which takes low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called **kernels**. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs you’ve defined.

Text

Description automatically generated with medium confidence

## Name some advantages of SVM

**Junior**

[](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[SVM](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[56](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)

**Answer**

* **Guaranteed Optimality**: Owing to the nature of Convex Optimization, the solution will always be global minimum not a local minimum.
* **Abundance of Implementations**: We can access it conveniently, be it from Python or Matlab.
* SVM can be used for linearly separable as well as non-linearly separable data. Linearly separable data is the hard margin whereas non-linearly separable data poses a soft margin.
* SVMs provide compliance to the semi-supervised learning models. **It can be used in areas where the data is labeled as well as unlabeled. It only requires a condition to the minimization problem which is known as the Transductive SVM.**
* Feature Mapping used to be quite a load on the computational complexity of the overall training performance of the model. However, with the help of **Kernel Trick**, SVM can carry out the feature mapping using a simple dot product.

## What happens when there is no clear Hyperplane in SVM?

* Data is rarely as clean that hyperplane is a line that linearly separates and classifies a set of data. In order to classify a dataset, it’s necessary to move away from a 2D view of the data to a 3D view.
* This ‘lifting’ of the data points represents the *mapping of data* into a *higher dimension*. This is known as kernelling.

A picture containing text, athletic game, sport

Description automatically generated

## Why would you use the Kernel Trick?

When it comes to **classification** problems, the goal is to establish a decision boundary that maximizes the margin between the classes. However, in the real world, this task can become difficult when we have to treat with **non-linearly separable data**. One approach to solve this problem is to perform a data transformation process, in which we map all the data points to a **higher dimension** find the boundary and make the classification.

That sounds alright, however, when there are more and more dimensions, computations within that space become more and more expensive. In such cases, the **kernel trick allows us to operate in the original feature space without computing the coordinates of the data** in a higher-dimensional space and therefore offers a more efficient and less expensive way to transform data into higher dimensions.

There exist different kernel functions, such as:

* *linear*,
* *nonlinear*,
* *polynomial*,
* *radial basis function (RBF)*, and
* *sigmoid*.

## Compare K-Nearest Neighbors (KNN) and SVM

* Linear **SVM** is considered parametric because it can only produce linear boundaries.
* ***kNN*** is a nonparametric algorithm because it avoids a priori assumptions about the shape of the class boundary and can thus adapt more closely to nonlinear boundaries as the amount of *training data increases*.
* ***kNN*** has *higher variance* than linear SVM but it has the advantage of producing classification fits that adapt to any boundary. Even though the true class boundary is unknown in most real-world applications, kNN has been shown to approach the theoretically optimal classification boundary as the training set increases to massive data.
* Because ***kNN*** does not impose any structure on the boundary, it can create class boundaries that may be less interpretable than those of *linear SVM*.

## Compare SVM and Logistic Regression in handling outliers

* For **Logistic Regression**, outliers can have an unusually large effect on the estimate of logistic regression coefficients. It will find a linear boundary if it exists to accommodate the outliers. To solve the problem of outliers, sometimes a sigmoid function is used in logistic regression.
* For **SVM**, outliers can make the decision boundary deviate severely from the optimal hyperplane. One way for SVM to get around the problem is to intrduce slack variables. There is a penalty involved with using slack variables, and how SVM handles outliers depends on how this penalty is imposed.

## What are Polynomial Kernels?

* The *polynomial kernel* is a kernel function commonly used with *SVMs* and other kernelized models, that represents the *similarity* of vectors in a feature space over *polynomials* of the original variables, allowing learning of *non-linear models*.
* The polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. In *regression analysis*, these combinations of features are called *interaction features*.
* For degree d polynomials, the polynomial kernel is defined as:

Text

Description automatically generated

where x and y are vectors in the *input space*.

* **Polynomial kernels are used for *Natural Language Processing***

#### **What is RBF in SVMs?**

RBF which is the Radial Basis Function is a kernel function that represents a separate dimension per data point that we have. It allows our data to be linearly separable by assigning each data point a Gaussian distribution. It then traces a line of the sum of these Gaussian distributions (now interconnected). The points are projected on this line (still within its Gaussian distribution) and then we separate our data and project it back to a 1-dimensional space.

RBF is the default kernel used for many Support Vector Machine algorithms due to its robustness and efficiency.

#### **How to deal with multiple classes with SVM?**

We can create an SVM per class. The idea here is to be able to classify between a particular class and every other class. This paradigm is known as 1 versus rest.

SVM(x, y, z) -> SVM(x, y==z), SVM(y, x==z), SVM(z, x==y)

So to choose a prediction, we plug in samples from every SVM, we then measure the margin it produces within the classes. We predict the class that produces the largest margin between the sample and the other class.

## What are some similarities between SVMs and Neural Networks?

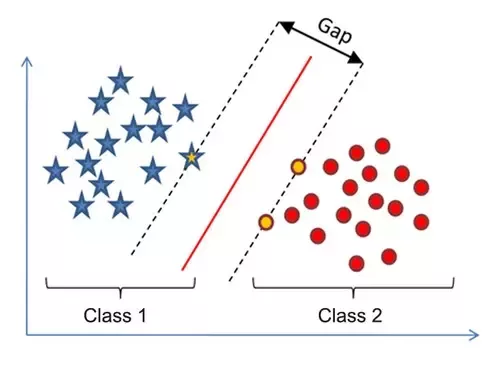
* **Parametric:** SVM and neural networks are both parametric but for different reasons.
  + For **SVM** the typical parameters are; *soft-margin parameter* (C), *parameter of the kernel function* (gamma).
  + **Neural networks** also have parameters but it is a lot more than SVM. Some NN parameters are the number of layers and their size, number of training epochs, and the learning rate.
* **Embedding Non-Linearity:** Both the methods can embed non-linear functions.
  + **SVM** does this through the usage of *kernel method*.
  + **Neural Networks** embed non-linearity using non-linear activation functions.
* **Comparable Accuracy:**
  + If both SVM and Neural Networks are trained in the *same dataset*, given the *same training time*, and the *same computation power* they have comparable accuracy.
  + If neural networks are given as much computation power and training time as possible then it outperforms SVMs.

## What is the difference between Classification and Regression when using SVM?

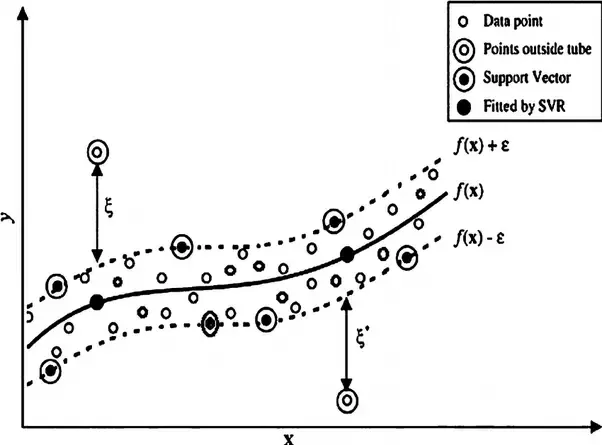
Explain how would you use a Support Vector Machine in *regression* tasks (SVM)?

**Answer**

SVM is a model for **classification**. For **regression**, you use *support vector regression* (SVR). The idea of SVR is closely related to that of SVM:

* In SVM, you want to find a separating hyperplane such that all points are at a certain distance from this plane. If there are points that are too close to the separating hyperplane, there is a penalty [the hinge loss]:
* In **SVR**, you want to find a function such th
* at *all points are within a certain distance* from this function. Unlike other Regression models that try to minimize the *error* between the real and predicted value, the SVR tries to *fit the best line within a threshold value* (if points are outside this distance [“ 𝜖 -tube”], there is a penalty):

In **SVR**, you want to find a function such that all points are within a certain distance from this function. Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value (if points are outside this distance [“ 𝜖 -tube”], there is a penalty):



#### **What is the difference in idea for using Support Vector Machine for regression and classification?**

## For classification with SVM, our goal is to maximize the distance between our decision boundary and our support vectors (margin).  For regression, the goal now is to keep all the points within the margin.

## When SVM is not a good approach?

SVM is not a good approach if:

1.When data is not in features form

2.SVM is not suitable for multiclass classification

3.When your data is imbalanced

4.When you want model explainability.SVM is black box model.

* Data is hardly given as feature vectors. In reality, more often than not, data might be images, videos, speech, text/language, music, and so on. How do you engineer the features in order to feed an SVM?
* If your data is non-linear and you choose a linear kernel to model it.
* If kernel and its parameters are not properly estimated/chosen.
* The quadratic programming problem is computationally intensive and with the increase in training data, it can be very slow on normal machines.
* SVM is mainly developed for binary classification problems. For handling multi-class problems, either you perform a 1-against-all strategy or do other tricks like optimizing classes together
* Your data is imbalanced.
* SVM can only be feasibly trained with a few 10’s of thousands of samples at most

## When would you use SVMs over Random Forest and vice-versa?

The choice depends very much on what data you have and what is your purpose:

* ***Random forest* is suited for multiclass problems.**
* ***SVM* is suited for two-class problems**. If the problem is multiclass, then the problem needs to be reduced to multiple binary classification problems. Usually this consists of building binary classifiers which distinguish:
  + (i) between one of the labels and the rest (one-versus-all) or
  + (ii) between every pair of classes (one-versus-one).
* ***Random forest* works well with both *numerical* or *categorical* features**. **Features need not be scaled either. It is generally okay to keep the dataset without processing it in random forest.**
* ***SVM* tries to maximize the *distance* between different points. It is up to the designer to define what the *distance* represents. As a consequence, one-hot-encoding, and scaling data is very important in *SVM*.**
* For a classification problem *Random Forest* gives you the *probability* of belonging to the class.
* *SVM* gives you the *distance* to the boundary, you still need to convert it to probability somehow if you need probability.
* If you have data with 𝑛 n points and 𝑚 m features, an intermediate step in SVM is constructing an 𝑛×𝑛 matrix (think about memory requirements for storage) by calculating 𝑛2 dot products (computational complexity). Therefore, as a rule of thumb, SVM is hardly scalable beyond 105 points.

## While designing an SVM classifier, what values should the designer select?

**Mid**

[](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[SVM](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[56](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)

**Answer**

While designing an SVM classifier, the designer should select:

* The **Regularization Weight** - It helps to include the problematic points into the equation and weight them using the C-Parameter. This parameter is a tradeoff between maximizing the margin and minimizing the error.
* The **Type of Kernel Function** - The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with a little computational cost even in very high-dimensional spaces. Kernel function examples are:
  + linear,
  + nonlinear,
  + polynomial,
  + Gaussian kernel,
  + Radial basis function (RBF),
  + sigmoid
* The **Parameters of the Kernel Function** - like Gamma for Gaussian Radial Basis function kernel.

These values should be selected by hand-tuning or by cross-validation.

## Is there a relation between the Number of Support Vectors and the classifiers performance?

* The number of support vectors depends on how much *slack* we allow and the distribution of the data.
* If we allow a large amount of slack, we will have a large number of support vectors.
* If we allow very little slack, we will have very few support vectors.
* The accuracy (or model *performance*) depends on finding the right level of slack for the data being analyzed.
* Chart, scatter chart

  Description automatically generated

## What are Slack Variables in SVM?

**Senior**

[](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[SVM](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[56](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)

**Answer**

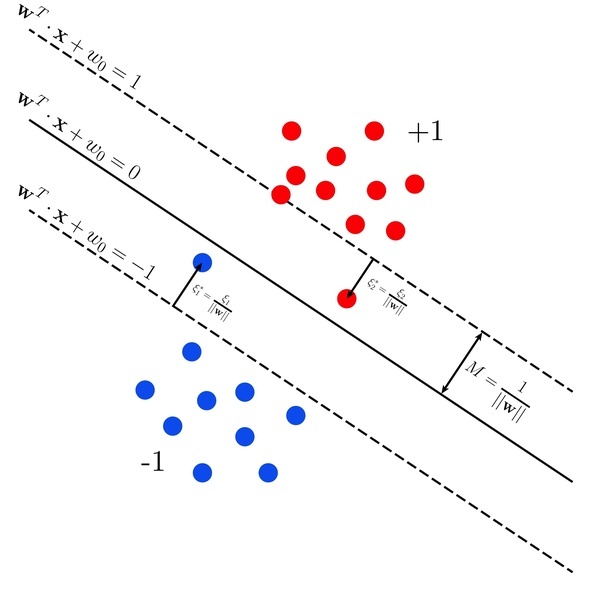
The standard SVM classifier works only if you have well-separated categories. To be more specific, they need to be linearly separable. It means there exists a line (or hyperplane) such that all points belonging to a single category are either below or above it. In many cases that condition is not satisfied, but still, the two classes are pretty much separated except for some small training data where the two categories overlap. It wouldn’t be a huge error if we would draw a line (somewhere in between) and accept some level of error - having training data on the wrong side of the marginal hyperplanes.

How do we measure the error? The answer is **Slack Variables**. One can relax the constraints by introducing so-called **slack variables** Note that each sample of the training set has its own slack variable. This gives us the following quadratic optimization problem:

\ A picture containing text, watch, clock

Description automatically generated

This is the **soft-margin SVM**. The variable C is the penalty strength, which specifies how much do we care about errors (training points that are on the wrong side). The 𝐶=∞ corresponds to a hard-margin (if possible).



## When would you use SVM vs Logistic regression?

**Senior**

[](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[SVM](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)**[56](https://www.mlstack.cafe/interview-questions/svm" \o "SVM Interview Questions)

**Answer**

Generally, the first approach is to try with ***logistic regression*** to see how the model performs. If we found that the data can't be linearly separable or the model does not have a good performance, then we can try using **SVM**.

We can also considerer the following rule of thumb: Given n number of features and m number of training examples:

* If n is less than 10,000 and m is less than 1,000 we can use either logistic regression or SVM with a linear kernel.
* If n is less than 1,000 and m is less than 10,000 we can use SVM with a Gaussian, polynomial, or another kernel.
* If n is less than 1,000, and m is greater than 50,000 we first add more features manually and then can use logistic regression or SVM with a linear kernel.

#### **What are the basic concepts behind Support Vector Machines?**

To make the separator more robust, and fit it with the greatest margin possible. To use kernel methods to effectively estimate bigger feature spaces.

***What is the role of C in SVM?***

Ans. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

***What is the difference between logistic regression and SVM***

Ans. Logistic regression assumes that the predictors aren’t sufficient to determine the response variable, but determine a probability that is a logistic function of a linear combination of them. If there’s a lot of noise, logistic regression (usually fit with maximum-likelihood techniques) is a great technique.

On the other hand, there are problems where you have thousands of dimensions and the predictors do nearly-certainly determine the response, but in some hard-to-explicitly-program way. An example would be image recognition. If you have a grayscale image, 100 by 100 pixels, you have 10,000 dimensions already. With various basis transforms (kernel trick) you will be able to get a linear separator of the data.

Non-regularized logistic regression techniques don’t work well (in fact, the fitted coefficients diverge) when there’s a separating hyperplane, because the maximum likelihood is achieved by any separating plane, and there’s no guarantee that you’ll get the best one. What you get is an extremely confident model with poor predictive power near the margin.

SVMs get you the best separating hyperplane, and they’re efficient in high dimensional spaces. They’re similar to regularization in terms of trying to find the lowest-normed vector that separates the data, but with a margin condition that favors choosing a *good* hyperplane. A hard-margin SVM will find a hyperplane that separates all the data (if one exists) and fail if there is none; soft-margin SVMs (generally preferred) do better when there’s noise in the data.

Additionally, SVMs only consider points near the margin (support vectors). Logistic regression considers all the points in the data set. Which you prefer depends on your problem.

Logistic regression is great in a low number of dimensions and when the predictors don’t suffice to give more than a probabilistic estimate of the response. SVMs do better when there’s a higher number of dimensions, and especially on problems where the predictors do certainly (or near-certainly) determine the responses.

9. **Suppose you are using RBF kernel in SVM with high Gamma value. What does this signify?**

Ans. The gamma parameter in SVM tuning signifies the influence of points either near or far away from the hyperplane.

For a low gamma, the model will be too constrained and include all points of the training dataset, without really capturing the shape.

For a higher gamma, the model will capture the shape of the dataset well.

10. **What is generalization error in terms of the SVM?**

Ans. Generalisation error in statistics is generally the out-of-sample error which is the measure of how accurately a model can predict values for previously unseen data.