Kernel Machines

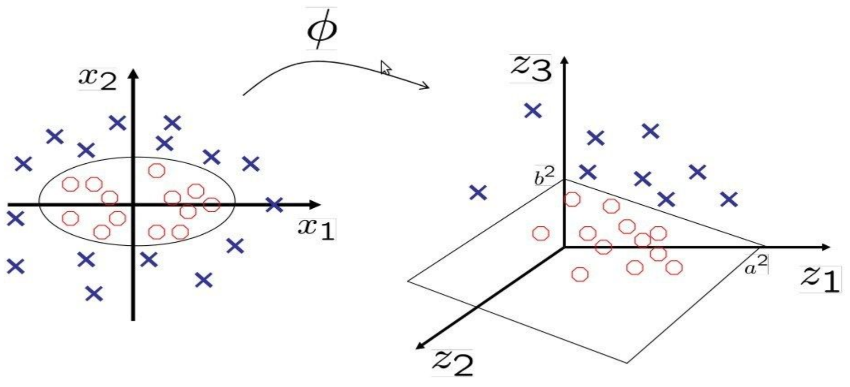
A kernel is a shortcut that helps us do certain calculation faster which otherwise would involve computations in higher dimensional space. Intuitively, a kernel function measures the similarity between two data points. The notion of similarity is task-dependent.

1) What are Kernels?

A kernel is a similarity function. It is a function that you provide to a machine learning algorithm. It takes two inputs and spits out how similar they are.

Kernel methods are a class of algorithms for pattern analysis, whose best known member is the Support Vector Machines. The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets. For many algorithms that solve these tasks, the data in raw representation have to be explicitly transformed into feature vector representations via a user-specified feature map: in contrast, kernel methods require only a user-specified kernel, i.e., a similarity function over pairs of data points in raw representation.

Intuitively, a kernel is just a transformation of our input data that allows us(or an algorithm like SVMs) to treat/process it more easily. Imagine that we have the toy problem of separating the red circles from the blue crosses on the plane as shown below.



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

Our separating surface would be the ellipse drawn on the left figure. However, transforming our data into a 3 dimensional space through the mapping shown in the figure would make the problem much easier since, now, our points are separated by a simple plane. This embedding on the higher dimension is called the kernel trick.

Mathematics:

Mercer's theorem: Let $x \in R^l$ and a mapping Φ,

$$\boldsymbol{x} \mapsto \boldsymbol{\phi}(\boldsymbol{x}) \in H \qquad \text{……(1)}$$

where H is a hilbert space. The inner product operation has an equivalent representation

$$\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{z}) \rangle = K(\boldsymbol{x}, \boldsymbol{z}) \qquad \text{..........(2)}$$

where $\langle\ .\,,\ .\ \rangle$ denotes the inner product operation in H and K(x, z) is a symmetric continuous function satisfying the following condition.

$$\int_C \int_C K(\boldsymbol{x}, \boldsymbol{z}) g(\boldsymbol{x}) g(\boldsymbol{z})\, d\boldsymbol{x}\, d\boldsymbol{z} \ \geq\ 0 \qquad \text{.............(3)}$$

for any $g(x)$, $x \ \in\ C\ \subset R^l$ such that

$$\int_C g(\boldsymbol{x})^2\, d\boldsymbol{x} < +\infty \qquad \text{……………..(4)}$$

where C is compact (finite) subset of R^l. The opposite is always true; that is, for any symmetric, continuous function $K(x,\ z)$ satisfying (3) and (4) there exists a space in which $K(x,\ z)$ defines an inner product. Such functions are known as **kernels** and the space H as Reproducing kernel Hilbert space (RKHS). What Mercer's theorem does not disclose to us, however, is how to find this space. That is, we do not have a general tool to construct the mapping $\Phi(\ .\ )$ once we know the inner product of the corresponding space**.** Furthermore, we lack the means to know the dimensionality of the space, which can even be infinite. This is the case, for example, for the radial basis (Gaussian) kernel.

Typical examples of kernels used in pattern recognition applications are as follows:

1) Polynomials:

$$K(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^T \boldsymbol{z} + 1)^q, \quad q > 0$$

2) Radial Basis Function(Gaussian):

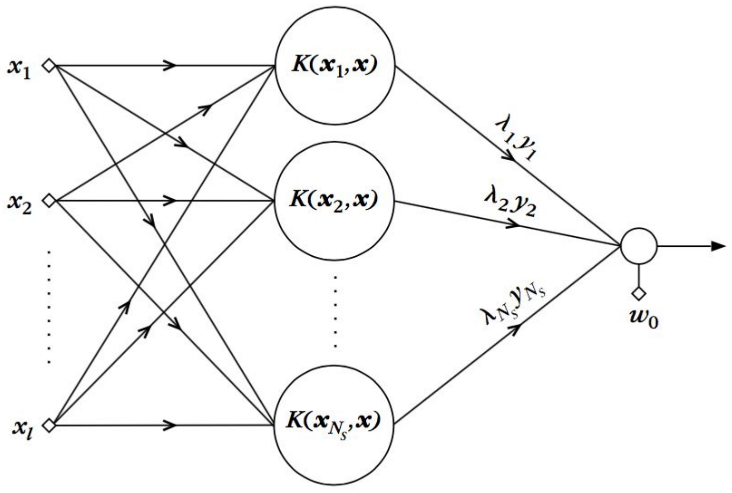
$$K(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{z}\|^2}{\sigma^2}\right)$$

3) Hyperbolic Tangent:

$$K(\boldsymbol{x}, \boldsymbol{z}) = \tanh\left(\beta \boldsymbol{x}^T \boldsymbol{z} + \gamma\right)$$

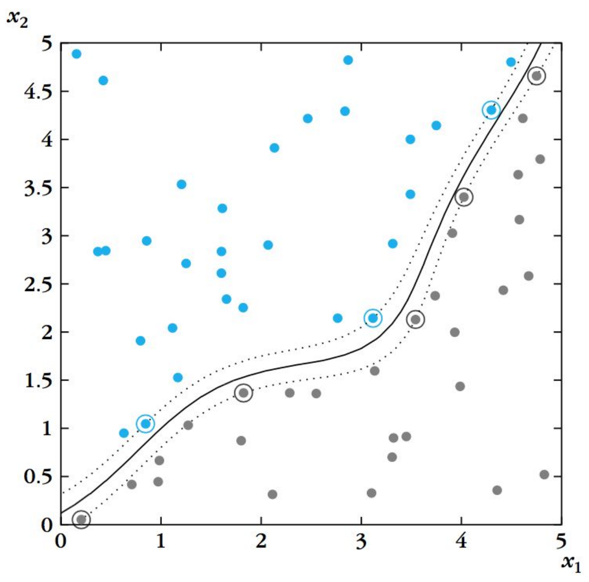
for appropriate values of   and   so that Mercer's conditions are satisfied. One possibility is $\beta = 2, \quad \gamma = 1$.

The Architecture:


x_1
x_2
x_l
$K(\boldsymbol{x}_1, \boldsymbol{x})$
$K(\boldsymbol{x}_2, \boldsymbol{x})$
$K(\boldsymbol{x}_{N_s}, \boldsymbol{x})$
$\lambda_1 y_1$
$\lambda_2 y_2$
$\lambda_{N_s} y_{N_s}$
w_0

where, x denotes feature vectors.
The above figure shows the SVM architecture employing kernel functions.
The number of nodes is determined by the number of support vectors N_s. The nodes perform the inner products between the mapping of x and the corresponding mappings of the support vectors in the high-dimensional space, via the kernel operation.

Pictorial example:

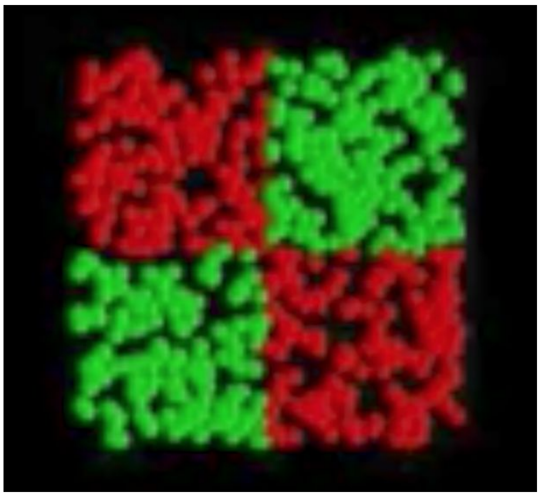


Example of a nonlinear SVM classifier for the case of two nonlinearly separable classes, where the Gaussian radial basis function kernel, with $\sigma = 1.75$, has been used. Dotted lines mark the margin and circled points the support vectors.

Working of kernels:

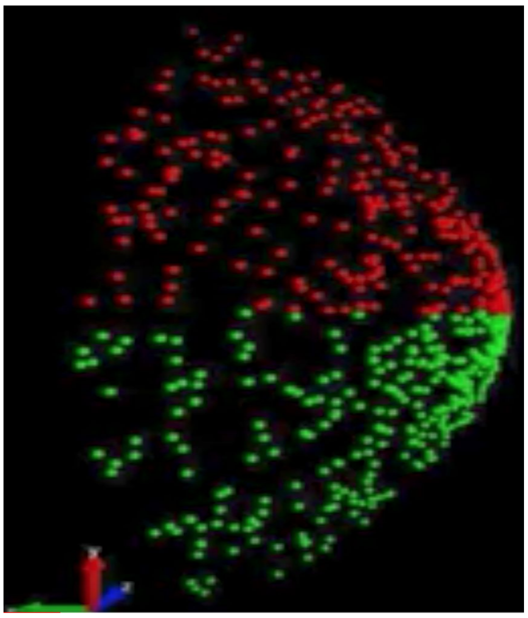
The below series of images clearly explains the working of kernels

1)



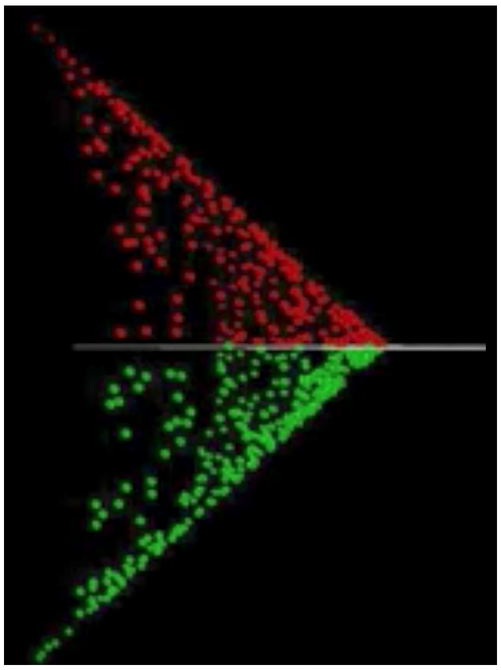
This image shows a dataset in two dimensions which cannot be linearly separable.

2)



This image shows the data in the previous image in 3 dimensional space

3)



This image shows the above data can be linearly separable in 3 dimensional space.

2) Feature Space Mapping:

Kernels are the functions which are used to project the data from lower dimensions to higher dimensions to separate the data linearly.

The input space (lower dimensional space) is usually denoted by X space and feature space is usually denoted by Z space. Then, now, we gonna project from X space to Z space.

The idea behind the kernel is that we want to go to higher dimensional Z space without paying actual the price for it.

In nonlinear transformation we still have to transform the points and perform the inner product between two z vectors with high dimensions.

The only thing we need from z space is the result of the inner product.

$$D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{z}_i \cdot \mathbf{z}_j)$$

That is If we can compute inner product without visiting Z space, even without knowing what Z space is, we can still use SV machinery.

For the data points x and x' in X space, we only need the inner product z · z'.

Example using polynomial kernel is shown below:
Let X be two dimensional feature vector

$$\mathbf{x} = (x_1, x_2)$$

Let k be a polynomial kernel with degree two given by,

$$K(\mathbf{x},\mathbf{x}') = (1 + \mathbf{x} \cdot \mathbf{x}')^2 = (1 + x_1 x'_1 \; + \; x_2 x'_2)^2$$

which will be reduced to,

$$K(\mathbf{x},\mathbf{x}') = 1 + x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2 x_1 x'_1 + 2 x_2 x'_2 + 2 x_1 x'_1 \, x_2 x'_2$$

Now we got everything in terms of x and x' but the two's are extra. So our *z* and *z'* will become as follows,

$$\mathbf{z} = (1, \; x_1^2 \;, \; x_2^2 \;, \; \sqrt{2} x_1 \;, \; \sqrt{2} x_2 \;, \; \sqrt{2} x_1 x_2 \;)$$

$$\mathbf{z}' = (1, \; x'^2_1, \; x'^2_2, \; \sqrt{2} x'_1, \; \sqrt{2} x'_2, \; \sqrt{2} x'_1 \, x'_2)$$

This example is for degree 2 polynomial kernel with two features.
In this way we can map from input space X to feature space Z.

3) Why Kernels?

In many cases, computing the kernel is easy, but computing the feature vector corresponding to the kernel is very hard. The feature vector for even simple kernels can blow up in size, and for kernels like the RBF kernel the corresponding feature vector is infinite dimensional.

Many machine learning algorithms can be written to only use dot products, and then we can replace the dot products with kernels. By doing so, we don't have to use the feature vector at all. This means that we can work with highly complex, efficient-to-compute, and yet high performing kernels without ever having to write down the huge and potentially infinite dimensional feature vector. Thus if not for the ability to use kernel functions directly, we would be stuck with relatively low dimensional, low-performance feature vectors.

Recently, the use of kernels in learning systems has received considerable attention. The main reason is that kernels allow to map the data into a high dimensional feature space in order to increase the computational power of linear machines. Thus, it is a way of extending linear hypotheses to nonlinear ones, and this step can be performed implicitly.

Support vector machines, kernel principal component analysis, kernel Gram-Schmidt, Bayes point machines, Gaussian processes, are just some of the algorithms that make crucial use of kernels for problems of classification, regression, density estimation, and clustering.

Beyond the SVM paradigm:

One of the most attractive properties of the support vector machines, which has contributed to their popularity, is that their computational structure allows for the use of a kernel function, as discussed in the previous section. Sometimes this is also known as the kernel trick. This powerful tool makes the design of a linear classifier in the high-dimensional space independent of the dimensionality of this space.

The success of the SVMs in practice has inspired a research effort to extend a number of linear classifiers to nonlinear ones, by embedding the kernel trick in their structure. This is possible if all the computations can be expressed in terms of inner product operations.

4) Types of Kernels:

a) Linear Kernel:

The Linear Kernel is the simplest kernel function. It is given by the inner product <x, y> plus an optional constant c. Kernel algorithms using a linear kernel are often equivalent to their non-kernel counterparts.

$$k(x, y) = x^T y + c$$

b) Polynomial Kernels:

In machine learning, the polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of nonlinear models.

Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these.

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

$$K(x, y) = (x^\mathsf{T} y + c)^d$$

where x and y are vectors in the input space, i.e. vectors of features computed from training or test samples and $c \geq 0$ is a free parameter trading off the influence of higher-order versus lower-order terms in the polynomial. When $c = 0$, the kernel is called homogeneous.

As a kernel, K corresponds to an inner product in a feature space based on some mapping φ:

$$K(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

The nature of φ can be seen from an example. For d = 2, so we get the special case of the quadratic kernel.

$$K(x, y) = \left(\sum_{i=1}^{n} x_i y_i + c \right)^2 = \sum_{i=1}^{n} \left(x_i^2\right) \left(y_i^2\right) + \sum_{i=2}^{n} \sum_{j=1}^{i-1} \left(\sqrt{2} x_i x_j\right) \left(\sqrt{2} y_i y_j\right) + \sum_{i=1}^{n} \left(\sqrt{2c} x_i\right) \left(\sqrt{2c} y_i\right) + c^2$$

From this it follows that the feature map is given by:

$$\varphi(x) = \langle x_n^2, \ldots, x_1^2, \sqrt{2} x_n x_{n-1}, \ldots, \sqrt{2} x_n x_1, \sqrt{2} x_{n-1} x_{n-2}, \ldots, \sqrt{2} x_{n-1} x_1, \ldots, \sqrt{2} x_2 x_1, \sqrt{2c} x_n, \ldots, \sqrt{2c} x_1, c \rangle$$

Applications:

This is most widely used in Natural Language Processing(NLP). The most common degree is d = 2 (quadratic), since larger degrees tend to overfit on NLP problems.

c) Radial Basis Function Kernel:

In machine learning, the (Gaussian) radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. In particular, it is commonly used in support vector machine classification.

The RBF kernel on two samples x and x', represented as feature vectors in some input space, is defined as

$$K(\mathbf{x}, \mathbf{x'}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x'}\|^2}{2\sigma^2}\right)$$

where,

$$\|\mathbf{x} - \mathbf{x'}\|^2$$

Is recognized as the squared euclidean distance between the two feature vectors, and σ is a free parameter

The feature space of the kernel has an infinite number of dimensions.
For σ=1, its expansion is,

$$\begin{aligned}\exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x'}\|^2\right) &= \sum_{j=0}^{\infty} \frac{(\mathbf{x}^\top \mathbf{x'})^j}{j!} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \exp\left(-\frac{1}{2}\|\mathbf{x'}\|^2\right) \\ &= \sum_{j=0}^{\infty} \sum_{\sum n_i = j} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp\left(-\frac{1}{2}\|\mathbf{x'}\|^2\right) \frac{{x'}_1^{n_1} \cdots {x'}_k^{n_k}}{\sqrt{n_1! \cdots n_k!}}\end{aligned}$$

d) Fisher Kernel:

The Fisher Kernel, named after Ronald Fisher, is a function that measures the similarity of two objects on the basis of sets of measurements for each object and a statistical model. In a classification procedure, the class for a new object (whose real class is unknown) can be estimated by minimising, across classes, an average of the Fisher kernel distance from the new object to each known member of the given class.

Mathematics:

Fisher Score:

The Fisher kernel makes use of the Fisher score, defined as:

$$U_X = \nabla_\theta \log P(X|\theta)$$

with θ being a set (vector) of parameters. The function taking θ to log $P(X \mid \theta)$ is the log-likelihood of the probabilistic model.

Fisher Kernel:

The Fisher Kernel is defined as:

$$K(X_i, X_j) = U_{X_i}^T I^{-1} U_{X_j}$$

with I the Fisher information matrix.

Application:

Information Retrieval:

The Fisher kernel is the kernel for a generative probabilistic model. As such, it constitutes a bridge between generative and probabilistic models of documents.

e) String Kernel:

A string kernel is a kernel function that operates on strings, i.e., finite sequences of symbols that need not be of the same length. String kernels can be intuitively understood as functions measuring the similarity of pairs of strings: the more similar two strings a and b are, the higher the value of a string kernel K(a, b) will be.

Definition:

The definition of a string subsequence kernel[1] on strings over an alphabet Σ . Coordinate-wise, the mapping is defined as follows:

$$\varphi_u : \begin{cases} \Sigma^n \to \mathbb{R}^{\Sigma^n} \\ s \mapsto \sum_{\mathbf{i}:u=s_{\mathbf{i}}} \lambda^{l(\mathbf{i})} \end{cases}$$

The **i** are multi indices and u is a string of length n: subsequences can occur in a non-contiguous manner, but gaps are penalized. The multi index i gives the positions of the characters matching *u* in *s*. $l(i)$ is the difference between first and last entry in **i,** that is: how far apart in *s* the subsequence matching *u* is. The parameter λ may be set to any value between 0 and 1.

e) Exponential Kernel:

The exponential kernel is closely related to the Gaussian kernel, with only the square of the norm left out.

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

f) Laplacian Kernel:

The Laplace Kernel is completely equivalent to the exponential kernel, except for being less sensitive for changes in the sigma parameter.

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

It is important to note that the observations made about the sigma parameter for the Gaussian kernel also apply to the Exponential and Laplacian kernels.

g) Hyperbolic Tangent (Sigmoid) Method:

The Hyperbolic Tangent Kernel is also known as the Sigmoid Kernel and as the Multilayer Perceptron (MLP) kernel. The Sigmoid Kernel comes from the Neural Networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons.

$$k(x, y) = \tanh(\alpha x^T y + c)$$

A SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

There are two adjustable parameters in the sigmoid kernel, the slope alpha and the intercept constant c. A common value for alpha is 1/N, where N is the data dimension.

5) Constructing New Kernels:

We can construct new kernels from previously defined kernels. Suppose we have kernels k1 and k2. Then the following are also kernels.

1)

$$k(x, z) = \alpha k_1(x, z) + \beta k_2(x, z)$$

where, $\alpha, \beta > 0$

2)

$$k(x, z) = k_1(x, z) k_2(x, z)$$

3)

$$k(x,\ z)\ =\ k_1(h(x), h(\ z))$$

where $h:\ X\ ->\ X$, i.e., h is a transformation in the same domain, k is simply a different kernel in that domain

4)

$$k\ (x,\ z)\ = g(x)g(z)$$

where, $g:\ X\ ->\ R$

5)

$$k(x,\ z)\ = r(k_1(x,\ z))$$

where r is any polynomial with positive coefficients.

6)

$$k(x,\ z)\ = exp(k_1(x,\ z))$$

where $exp(\ .\ )$ is an exponential function.

6) Utility of Kernels:

Recently the use of kernels in learning system has received considerable attention.The main reason is that kernels allow to map the data into a high dimensional feature space in order to increase the computational power of linear machines. Thus, it is a way of extending linear hypotheses to nonlinear ones, and this step can be performed implicitly.

Support vector machines, kernel principal component analysis, kernel Gram-Schmidt, Bayes point machines, Gaussian processes, are just some of the algorithms that make crucial use of kernels for problems of classification, regression, density estimation, and clustering.

Kernels are the similarity functions which is usually applied to nonlinearly separable data. This function converts the data in X space(i.e., the given space) to the data of Z space which is linearly separable without paying the actual processing cost for it.

Applications of some of the popular kernels are discussed below.

Kernels like Fisher Kernel is used in Information retrieval. It constitutes a bridge between generative and probabilistic model of documents.

The Fisher Kernel can also be applied to image representation for classification or retrieval problems. The Fisher kernel can result in a compact and dense representation, which is more desirable for image classification and retrieval problems.

Polynomial Kernel is quite popular in natural language processing(NLP).

Kernels are widely used in Support vector machines because this kernel trick makes the design of a linear classifier in the high-dimensional space independent of the dimensionality of this space.

The success of SVM has also lead the research to extend the linear classifiers to nonlinear ones using this powerful kernel trick.

The Below outputs are for Arrhythmia data set

Note: The Original dataset with 279 features yielded 48% accuracy

1) When Gaussian Kernel is used

Number of features generated	Accuracy
452	52.04%

2) When Polynomial Kernel is used

Degree	Number of features generated	Accuracy
2	452	50.90%
3	452	51.10%
4	452	50.99%

The Below outputs are for Internet Advertisement data set

Note: The Original dataset with 1558 features yielded 95.93% accuracy

1) When Gaussian Kernel is used

Number of features generated	**Accuracy**
3279	98.5145%

2) When Polynomial Kernel is used

Degree	**Number of features generated**	**Accuracy**
2	3279	90.4688%
3	3279	98.2017%
4	3279	97.9672%