Kernel Methods

Rohit Singhatwadia

May 2018

1 Introduction

Historically, in Machine Learning, most of the underlying theory and algorithms developed were good in handling linearly separable data. However, modern data analysis problems require methods capable of dealing with complex nonlinear data. To be able to solve for such data, it is required to map the features to a new higher dimensional space so that it can become linearly separable, which makes it more convenient for us to apply traditional machine learning algorithms. For example, in the figure 1 below, consider the points having black, green, and red color in class 1 and the rest in class 2. Figure 2 displays its projection in two dimension. After comparing the two images, its easy to observe this functionality.

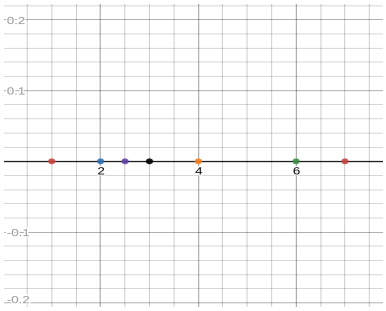


Fig 1 Original points in 1D

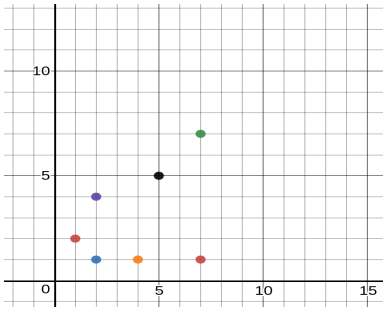


Fig 2 Transformed points in 2D

Consider an embedding map

$$\phi : x \epsilon R^n \mapsto \phi(x) \epsilon F \subset R^N$$

The choice of the map ϕ aims to convert the nonlinear relations into linear ones. Although, after projecting it to higher dimensions, what is actually required is the dot product between these features. Unfortunately, if the dimension of the new feature space, i.e., N is large, then computing the dot product can be an expensive operation both in terms of time and space(to store the higher

dimension features). The inner products can, however, sometimes be computed more efficiently as a direct function of the input features, without going to the higher dimension. This direct function is known as the kernel function.

2 Kernels

A kernel is a function that measures the similarity between two objects in a way such that there exists a dot product corresponding to this similarity in some higher-dimensional space. The inputs can be anything from integers, real-valued vectors, strings, etc. Without a doubt, the simplest example is of the linear kernel. Given two vectors, the similarity is the length of the projection of one vector on another. Another interesting example is the Gaussian kernel. Given two vectors, the similarity will diminish with the radius of σ. The success of learning with kernels depends very strongly on our choice, which is often problem-specific.
Let's define a kernel properly

A kernel is a function that for all x, z ϵ X satisfies

$$k(x, z) = \langle \phi(x), \phi(z) \rangle$$

where,

$$\phi : x \epsilon R^n \mapsto \phi(x) \epsilon F \subset R^N$$

Kernels act as a shortcut as neither there is a need to compute higher-dimensional features nor a requirement of the mapping ϕ. One can directly compute the dot product in a higher dimension through original features. This is known as the kernel trick. To explain this let's use an example. Take 2 vectors $x = (1, 2, 3) y = (4, 5, 6)$ and try to compute its dot product in some higher dimensional space. Assume the features in that space be
$f(x) = (x1x1, x1x2, x1x3, x2x1, x2x2, x2x3, x3x1, x3x2, x3x3)$. For our vectors $\phi(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)$ and $\phi(y) = (16, 20, 24, 20, 25, 30, 24, 30, 36)$. Finally, their dot product is $\langle \phi(x), \phi(y) \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$. However, kernels can compute these directly $k(x, y) = (\langle x, y \rangle)^2 = (4 + 10 + 18)^2 = 1024$.

3 Hilbert Space

A Hilbert Space F is an abstract inner product space that follows two additional properties completeness and separability.
Completeness states that every Cauchy sequence (a series whose elements become closer to each other as sequence progress) of elements of F converges to an element h ϵ F.

Separability states that for any $\epsilon > 0$ there is a finite set of elements $h_1 \dots h_N$ of F such that for all h ϵ F

$$min||h_i - h|| < \epsilon$$

Intuitively, by completeness, it means that there are no missing points from a set. For example, the set of rational numbers is incomplete as the value $(2)^{1/2}$ is missing from it. Moreover, a Cauchy sequence of rational numbers can converge to it. Separability implies that Hilbert spaces have countable orthonormal bases. Any space which is finite or countably infinite is separable as the whole space is a countable dense subset of itself. An example of an uncountable separable space is the real line, in which the rational numbers form a countable dense subset. A simple example of a space which is not separable is uncountable discrete space. The reason for the importance of the properties of completeness and separability is that together, they ensure that Hilbert spaces are isomorphic to R^n for some finite n.
Each element in this space is a function. This function is a linear combination of the input features. It is similar to the weight vector as it is a linear combination of the feature vectors of the training points. Thus, finding the weight vector is equivalent to finding an element of this space.
A reproducing kernel Hilbert space (RKHS) is a Hilbert space of functions in which point evaluation is a continuous linear function. Thus, it guarantees that the required element is linear.
Note : In an inner product space, if k is a kernel and $x_1, x_2 \epsilon X$, then

$$k(x1, x2) \leq k(x1, x1) * k(x2, x2)$$

this is called Cauchy–Schwarz inequality.

4 Gram Matrix

Suppose there are l input vectors $x_1 \dots x_l$ and a kernel k to evaluate the dot product in a feature space with feature map ψ. Then the gram matrix is defined as

$$G_{ij} = \langle \phi(x_i), \phi(x_j) \rangle = k(x_i, x_j)$$

It is also known as kernel matrix. This matrix is symmetric and it contains all the information needed to find the pairwise distance in a data set. However, it loses some information like the angle at which the feature is located w.r.t origin, which suggest that gram matrix is rotation invariant.

Positive semi-definite Kernels Let X be a nonempty set. A function k : $X * X \mapsto R$ that gives rise to a positive semi-definite gram matrix for all values of $x \epsilon X$ is called positive semi-definite kernel.

5 Construction of the reproducing kernel Hilbert space

Let us define a function k

$$k : XxX \mapsto R$$

With the assumption that k satisfies the positive semi-definite property, proceed to construct a feature mapping ϕ into a Hilbert space for which is the kernel. Firstly, define the mapping ϕ

$$\phi : x\epsilon X \mapsto \phi(x) = k(x, .)\epsilon F$$

Note, something is unusual about this construction. The elements of the feature space will also be functions. Use F to denote the function space. To define F as a vector space

$$f, g\epsilon F \Rightarrow (f + g)(x) = f(x) + g(x)$$

Let's introduce an inner product on F as follows. Allow $f, g\epsilon F$ be given by

$$f(x) = \sum_{i=1}^{l} \alpha_i k(x_i, x)$$

and

$$g(x) = \sum_{i=1}^{n} \beta_i k(z_i, x)$$

then their dot product is

$$\langle f, g\rangle = \sum_{i=1}^{l}\sum_{j=1}^{n} \alpha_i\beta_j k(x_i, z_j) = \sum_{i=1}^{l} \alpha_i g(x_i) = \sum_{j=1}^{n} \beta_j f(z_j)$$

In the previous expression, to get the second and third equality substitute the value of g and f respectively.
This is real valued, bilinear and symmetric. So, it satisfies the properties of an inner product, provided

$$\langle f, f\rangle \geq 0$$

But it follows from our assumption that the kernel is positive semi-definite,

$$\langle f, f\rangle = \sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i\alpha_j k(x_i, x_j) = \alpha' K\alpha \geq 0$$

where α_i and α_j are entries of vector α and K is the kernel matrix.
There is another property that follows if one takes g=k(x,.)

$$\langle f, k(x, .)\rangle = \sum_{i=1}^{l} \alpha_i k(x_i, x) = f(x)$$

This fact is known as reproducing property of the kernel. It is also required to demonstrate completeness and separability. Separability will follow if the input space is countable or the kernel is continuous. For completeness we consider a fixed input x and a Cauchy sequence $(f_n)_{n=1}^{\infty}$. $(f_n)_{n=1}^{\infty}$.

$$(f_n(x) - f_m(x))^2 = (\langle f_n - f_m, k(x,.)\rangle)^2 \leq ||f_n - f_m||^2 k(x,x)$$

by the Cauchy–Schwarz inequality. Thus, $f_n(x)$ is a bounded sequence of real numbers and has a limit. This satisfies the completeness condition.
By using the reproducing property we can also say that

$$\langle f, \phi(x)\rangle = \langle f, k(x,.)\rangle = f(x)$$

Thus, the function f can be represented as the linear function defined by an inner product, which also proves that the function f is linear. Since this function also corresponds to classification weights, it can be said that the features in the high dimensional space are linearly separable.

Given a function that satisfies the finitely positive semi-definite property, one should address the corresponding space F_k as its Reproducing Kernel Hilbert Space (RKHS).

Note: In the early years of kernel machine learning research, the notion of positive definite kernels was not popular. Instead, researchers considered kernels satisfying the conditions of Mercer's theorem.

Note: Let X be the set of all countable sequences of real numbers x $= x_1 \dots x_n \dots$ such that the sum

$$\sum_{i=1}^{\infty} x_i^2 < \infty$$

with the inner product between two sequences x and y defined by

$$\langle x, y\rangle = \sum_{i=1}^{\infty} x_i y_i$$

This space is known as L2.

6 Mercer's kernel

Mercer's theorem is used to construct feature space for a valid kernel just like the last section. The major difference is that it defines the feature space as vectors instead of functions.

Suppose K is a continuous symmetric function $K : R^m * R^m \mapsto R$, then there exist an inner product space H and a mapping $\phi : R^m \mapsto H$ such that

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2)\rangle$$

if for all square-integrable function g

$$\int\int K(x_1, x_2)g(x_1)g(x_2)dx_1dx_2 \geq 0$$

This can be illustrated with the help of an example Let our kernel be a positive constant function $K(x_1, x_2) = c$. Following conditions are required

$$\int\int cg(x_1)g(x_2)dx_1dx_2 \geq 0$$

$$c\int\int g(x_1)g(x_2)dx_1dx_2 \geq 0$$

Since both the integrals are same, it can be rewritten as

$$c(\int g(x_1)dx_1)^2 \geq 0$$

Thus, this kernel satisfies mercer theorem.

7 Properties of kernels

1) **The set of positive definite kernels is a closed convex cone**

(a) If $\alpha_1 and \alpha_2 are \geq 0$ and k_1, k_2 are positive semi-definite kernels then $\alpha_1 k_1 + \alpha_2 k_2$ is positive semi-definite.

(b)If k(x,x') = $\lim_{n\mapsto\infty} k_n(x, x')$ exist for all x,x' then k is positive semi-definite.

2) **The point-wise product $k_1 k_2$ is positive semi-definite.**

3) **The tensor product $k_1 \otimes k_2$ and direct sum $k_1 \oplus k_2$ are positive semi-definite.**

4) **Closure Properties**

(a) $k(x, y) = k_1(x, y) + k_2(x, y)$

(b) $k(x, y) = ak_1(x, y)$

(c) $k(x, y) = k_1(x, y)k_2(x, y)$

(d) $k(x, y) = f(x)f(y)$

(e)$k(x, y) = k_1(\phi(x), \phi(y))$

8 Kernel Construction

(a) $k(x, y) = p(k_1(x, y))$ where p(x) is a polynomial
(b) $k(x, y) = exp(k_1(x, y))$
(c) $k(x, y) = exp(-||x - y||^2/(2\sigma^2))$
(d) Kernels generated using closer properties

9 Examples of Kernels

9.1 Polynomial kernels

The derived polynomial kernel for a kernel k_1 is defined as

$$k(x, y) = p(k_1(x, y))$$

Frequently, it is also mentioned as the inhomogeneous polynomial.

$$k(x, y) = (\langle x, y\rangle + R)^d$$

It can be expanded using the Binomial Theorem. Features for each component in the sum together form the features of the whole kernel.
The polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. This kernel is not popular in SVM, as RBF outperforms it in most cases. However, it is frequently used in natural language processing. The most common degree is two since larger degree tend to overfit.
The problem with polynomial kernel is that it suffers from numerical instability when $x^T y + c < 1$ than $K(x, y) = (x^T y + c)^d$ tends to zero as d increases whereas $x^T y + c > 1$ tend to infinity.
In NLP, people need a combination of features to determine accuracy. While applying the polynomial kernel, we can improve the speed by using the inverted index. This inverted index table gives us for each feature the number of support vectors that contain that feature.

9.2 All-subsets kernel

All-subset kernel considers different combination of features. Consider a space having feature ϕ_A for each subset A of the input features, including the empty subset. It can be represented as

$$\phi_i(x) = x_1^{i_1} \dots x_n^{i_n}$$

with the restriction that i=$(i_{1n})\epsilon 0, 1^n$. The feature ϕ_A is given by multiplying together the input features for all the elements of the subset. Now

$$\phi : x \mapsto \phi_A(x)$$

The corresponding kernel is

$$k(x, y) = \langle \phi(x), \phi(y)\rangle = \sum_A \prod_{i \epsilon A} x_i y_i = \prod_{i=1}^{n} (1 + x_i y_i)$$

9.3 Radial basis function kernel

For σ¿ 0, the Gaussian kernel is defined as

$$k(x, y) = exp(-||x - y||^2/(2\sigma^2))$$

Rewrite it as

$$k(x, y) = exp(-\gamma||x - y||^2)$$

This kernel signifies an infinite dimensional feature space i.e it works as if we have transformed our features in an infinite dimensional space. Although the expansion of this function will propagate to infinite length, the higher-order terms will fall quickly. This specifies a similarity measure as to when x is very close to y the kernel will attain its maximum value, and when its far away the kernel value will tend towards zero. The shape, however, will always be smooth. A small gamma means Gaussian with a large variance, so the influence of x_j is more, i.e., if x_j is a support vector, a small gamma implies the class of this support vector will have influence on deciding the class of the vector x_i even if the distance between them is large. If gamma is large, then the variance is small, implying the support vector does not have wide-spread influence. Technically speaking, large σ leads to high bias and low variance models, and vice-versa. Also, the scaling by γ occurs the same amount in all directions.
RBF kernel works well in practice, and it is relatively easy to tune. It is a stationary kernel, which implies that it is invariant to translation. To illustrate this property, suppose we are computing $K(x, y)$. A stationary kernel will yield the same value $K(x, y)$ for $K(x + c, y + c)$, where c may be vector.For the RBF, this is done by working on the difference between the two vectors. Also, note that the linear kernel does not have this property.
Other kernels that are very close to this are exponential Kernel

$$k(x, y) = exp(-||x - y||/(2\sigma^2))$$

and laplacian Kernel

$$k(x, y) = exp(-||x - y||/(\sigma))$$

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.
Let us address why the VC dimension of the RBF kernel is infinite. Gaussian Kernels can classify any set of points exactly. It can be justified by saying that the VC dimension corresponds to the number of dichotomies one can get, which depends on the effective number of parameters. These are simply the weights of the classifier, and it corresponds to a point in a Hilbert space. Moreover, with the RBF kernel, the model works with infinite-dimensional space. Thus, with this argument, it can be said that the VC dimension of the RBF kernel is infinite.

9.4 ANOVA kernels

The ANOVA kernel of degree d is like the all-subsets kernel except that it is restricted to subsets of the given cardinality d. Represent them as

$$\phi_i(x) = x_1^{i_1} \dots x_n^{i_n}$$

with the restriction that i=$(i_1 n)\epsilon 0, 1^n$. It has another restriction that

$$\sum_{j=1}^{n} i_j = d$$

ANOVA stands for ANalysis Of VAriance.

9.5 Locality Improved kernels

Suppose there is an image to test our kernel. Different kernels can be used like the polynomial kernel or Gaussian RBF. However, these kernels do not make use of the locality of the image. Permute the pixels of the image, and still, these kernels will give the same result. By using locality, better results can be obtained. One way is to use a pyramidal kernel. It takes inner product between corresponding image patches, then raises the latter to some power p1, and finally raises their sum to another power p2. While the overall degree of this kernel is p1 p2, the first factor p1 only captures short-range interactions.

9.6 Kernels on Sets

These are defined in non-vectorial space. Consider the set of subsets P(D) of a fixed domain D. The elements of the input space therefore set. They could include all the subsets of D or some sub-selection of the whole power set

$$X \subset P(D)$$

Consider two elements of X $A_1, A2$, which are subsets of domain D. Several kernels can be defined like intersection kernel, union complement kernel based on basic set operations like union, intersection, and complement.

9.7 Kernels for text

Let's see the bag of words model. A bag is a set in which repeated elements are allowed so that not only the presence of a word but also its frequency is taken into account. As the ordering of words is ignored, the grammatical information is lost. Since the phrases are broken, their meaning is also forgotten. Refer to this full set of documents as the corpus and the set of terms occurring in the corpus as the dictionary. It can be represented as a vector in a space in which each dimension is associated with one term from the dictionary. Store the terms which occur in the document and their corresponding frequency. Clearly, the size of this vector is very large. Nevertheless, most of its entries are 0.

Document Term Matrix: The document–term matrix of a corpus is the matrix whose rows are indexed by the documents of the corpus and whose columns are indexed by the terms.The $(i, j)^{th}$ entry gives the frequency of term t_j in document d_i. The term-document matrix is the transpose D' of the document–term matrix. D'D gives the term-by-term matrix while the document-by-document matrix is DD'.

Vector space kernel: Vector space kernel is defined by the document-by-document matrix DD'.

In order to compute the kernel k, first convert the document into a list of the terms that they contain, which is known as tokenization. Each term encountered in the corpus is assigned its unique number, which ensures that the list of terms in a document can be reordered into ascending term order, together with an associated frequency count.

9.8 Kernels for strings

Spectrum kernels: Spectrum kernel counts how many substrings of length p they have in common. Let's define the spectrum of order p (or p-spectrum) of a sequence s to be the histogram of frequencies of all its substrings of length p. Comparing the p-spectra of two strings can give important information about their similarity in applications where contiguity plays an important role. We can define a kernel as the inner product of their p-spectra.
Construct all possible substrings of length p and find out which of them are present in our string, which will create a corresponding vector. Now the job of the kernel is to find out the inner product of these vectors.

All-subsequences kernel: It's working is similar to the spectrum kernel. The difference is instead of matching all possible substrings of length p, it matches all possible subsequences. Clearly, the vector space of all-subsequences kernel is much larger than the vector space of spectrum kernel.

9.9 Convolution and structures

To define kernels on structured objects. Suppose the object x ϵ X is composed of $x_p \epsilon X_p$. Haussler investigated how to define a kernel between composite objects by building on similarity measures that assess their respective parts. Define the R-convolution of $k1 \dots kp$ as

$$[k_1 * \dots * k_p](x, y) = \sum_{x' \epsilon R(x) y' \epsilon R(y)} \prod_{p=1}^{P} k_p(x - p', y'_p)$$

where the sum runs over all possible ways R(x) and R(x') in which x can be decomposed into $x_1 \dots x_p$. R-convolution is a valid kernel.

10 Representer theorem

Representer theorem states that a minimizer f of regularized empirical risk function defined over a reproducing kernel Hilbert space can be represented as a finite linear combination of kernel products evaluated on the input points in the training set data.
The representer theorem shows that solutions of a large class of optimization problems can be expressed as kernel expansions over the sample points. We define $\Omega : [0, \infty) \mapsto R$ a strictly monotonic increasing function, a set X and an arbitrary loss function c: $(X * R * R)^n \mapsto RU\infty$. Then each minimizer f ϵ H of the regularized risk functional

$$c((x_1, y_1, f(x_1)), \dots (x_n, y_n, f(x_n))) + \Omega(||f||_H^2)$$

can be expressed in the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x)$$

Monotonicity of Ω does not prevent the regularized risk function from having multiple local minima. If we discard the strictness of the monotonicity, then it no longer follows that each minimizer of the regularized risk admits an expansion.
Let's see the significance of this theorem. Solving for an optimization problem in an infinite-dimensional space H, containing linear combinations of kernels centered on arbitrary points of X, the solution within the span of n particular kernels will be centered on the training points. On solving, one can find that most of the α will turn out to be zero. Let's see this further in the next section. The values for which α is not zero will determine the solution.

11 Support Vector Machine for Classification

In this section, let's show how kernels can help in classification.
Firstly, let's design the classifier for linearly separable data. The objective is to find a line that can classify the two classes. As there can be many such lines, we want to pick the one which has the maximum margin. It means the distance from the closest point to the separating line should be as large as possible. This line, like any other separating line, has the same in-sample error. The reason for picking this line is if the data generated is not accurate the classifier with fat margin has a better chance of classifying it correctly as compared to a classifier with a thin margin.
Suppose the input points are in d dimensional space. The goal is to find a plane that classifies the points correctly with a fat margin.
Equation of this plane is $w^T x = 0$. Let x_n be the nearest data point to it. It is obvious that for any point in the dataset not on the plane, $|w^T x_n| > 0$. Multiplying this weight by any number will not change the plane. So, let's choose the normalized equation of the plane for mathematical simplicity. It can

be done by enforcing the constraint that $|w^T x_n| = 1$. Let's also include a bias b. Thus, equation is $w^T x + b = 0$ with the constraint $|w^T x_n + b| = 1$. Before moving further, let's see why the vector w is perpendicular to our plane. Take any two points on the plane and take their difference. Observe, it's dot product with w is zero $w^T(x - x') = 0$.
Find the distance of point x_n from the plane.
Take any point x on the plane. Take the projection of $x_n - x$ on w. This is our distance. Unit vector of w

$$w/||w||$$

Projection is the dot product of our unit vector and $x_n - x$ which is

$$|w^T x_n - w^T x|/||w|| = |w^T x_n + b - (w^T x + b)|/||w|| = 1/||w||$$

Our optimization problem is to maximize $1/||w||$, which is mathematically similar to minimizing $w^T w/2$. Also, constraint for solving the equation is $y_n(w^T x_n + b) \geq 1$ for all n.
To solve them, let's use the Lagrange method

$$L(w, b, \alpha) = 1/2 w_T w - \sum_{n=1}^{N} \alpha_n (y_n (w^T x_n + b) - 1)$$

Minimize w.r.t w and b maximize w.r.t each $\alpha_n \geq 0$.
Take derivatives w.r.t w and b

$$\nabla_w L = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0$$

$$\nabla_b L = -\sum_{n=1}^{N} \alpha_n y_n = 0$$

which results in

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0$$

substitute these equations in the Lagrangian.

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - (1/2) \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m$$

Maximize w.r.t α such that $\alpha_n \geq 0$ for all n ϵ N and $\sum_{n=1}^{N} \alpha_n y_n = 0$ Solve this for α using quadratic programming technique. From the equation its easy to

observe that complexity depends on the number of examples N.
Solution of α can be used to compute w

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

After solving, note that most of the values in α are zero. It defines the KKT condition

$$\alpha_n(y_n(w^T x_n + b) - 1) = 0$$

Those points for which α_n is greater than zero are the support vectors. These are the points that define the margin. Since their α_n is greater than zero, they define w. These are relatively less in number.
Solve for b using any support vector

$$y_n(w^T x_n + b) = 1$$

To solve for nonlinear data, one can go to a higher dimension space Z in which data is linearly separable and apply the quadratic programming in that space to get the result. Our equation here is simply

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - (1/2) \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m z_n^T z_m$$

$$w = \sum_{n=1}^{N} \alpha_n y_n z_n$$

Where z features in a higher dimension space. The support vectors it outputs reside in the Z space. Their pre-image in the X space might not make much sense in some cases. The margin is also maintained in the Z space.

And, finally, our hypothesis as

$$g(x) = sign(w^T z + b)$$

Again, only inner product is required from the Z space, which can be computed using kernels. It is needed in the computation of $L(\alpha)$, b and hypothesis function. This won't have any effect on quadratic programming computation.

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - (1/2) \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m K(x_n, x_m)$$

$$g(x) = sign(\sum_{n=1}^{N} \alpha_n y_n K(x_n, x) + b)$$

Soft Margin Classifier

Previously, hard margin classifier was discussed in which there was no scope of making any error. A model in which errors can be made is defined as the soft margin classifier. The solution to this is very similar to the one just discussed.However, soft margin classifiers consider margin violations. Let's introduce a new factor E_n which defines the amount of margin violation caused by point n. The total violations is

$$\sum_{n=1}^{N} E_n$$

Now our optimization problem is to minimize $w^T w/2 + C\sum_{n=1}^{N} E_n$. And our constraint is $y_n(w^T x_n + b) \geq 1 - E_n$ and $E_n \geq 0$ for all n.
Again apply Lagrange method

$$L(w, b, E, \alpha, \beta) = 1/2 w_T w + C\sum_{n=1}^{N} E_n - \sum_{n=1}^{N} \alpha_n(y_n(w^T x_n + b) - 1 + E_n) - \sum_{n=1}^{N} \beta_n E_n$$

Minimize w.r.t w,b and E maximize w.r.t each $\alpha_n \geq 0$ and $\beta_n \geq 0$.
Take the derivatives w.r.t w,b and E_n

$$\nabla_w L = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0$$

$$\nabla_b L = -\sum_{n=1}^{N} \alpha_n y_n = 0$$

$$\nabla_{E_n} L = C - \alpha_n - \beta_n = 0$$

Substituting it, our solution get reduced to

$$L(\alpha) = \sum_{n=1}^{N} \alpha_n - (1/2)\sum_{n=1}^{N}\sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m x_n^T x_m$$

Maximize w.r.t α such that $\alpha_n \geq 0$ and $\alpha_n \leq C$ for all n ϵ N and $\sum_{n=1}^{N} \alpha_n y_n = 0$. Again feed it to the quadratic programming technique and get the value of α and eventually the support vectors. Among these margin support vectors, there are the some which have $E_n = 0$.

12 Regression Estimation

Let's first understand the ϵ insensitive loss function. It ignores errors that are less than a threshold , which enables us to apply the kernel trick in regression. It extends the idea of soft margin to regression problems such that instead of

requiring yf(x) exceed some margin value, the condition is to bound yf(x) by a margin on both sides. Thus, it introduces the following constraints

$$y_i - f(x_i) \leq \epsilon_i - E_i$$

$$f(x_i) - y_i \leq \epsilon_i - E_i^*$$

where E_i and E_i^* are non negative. If $y_i - f(x_i) \leq \epsilon$ then no penalty is imposed. The objective function is given by the sum of the slack variables E_i and E_i^* and $1/2||w||^2$, which is equivalent to solving

$$minimize 1/2||w||^2 + \sum_{i=1}^{n} \varphi(y_i - f(x_i))$$

For different values of loss function φ
Least Square regression

$$\varphi(E) = (1/2)E_2$$

Least absolute deviations

$$\varphi(E) = |E|$$

Use the Lagrange and quadratic programming techniques to solve similar to the previous section.

13 Multicategory classification

Many estimation problems cannot be described by assuming that y=+1,-1. In this case it is advantageous to go beyond simple functions f(x), which depends on x only. Instead, encode a larger degree of information by estimating a function f(x,y), where y is obtained as a solution of an optimization problem over f(x,y). The objective is to match our predicted solution with the actual answer.
The loss may be more than just a simple 0–1 loss. Denote $\delta(y, y')$ the loss incurred by estimating y' instead of y. Without loss of generality, we require that $\delta(y, y) = 0$ and that $\delta(y, y') \geq 0$ for all y, y'ϵ Y.
To deal with this situation define f as

$$f(x, y) = \langle \phi(x, y), w \rangle$$

Corresponding kernel function is given by

$$k(x, y, x', y') = \langle \phi(x, y), \phi(x', y') \rangle$$

The resulting optimization problem

$$minimize (1/2)w^T w + C \sum_{i=1}^{N} E_i$$

such that

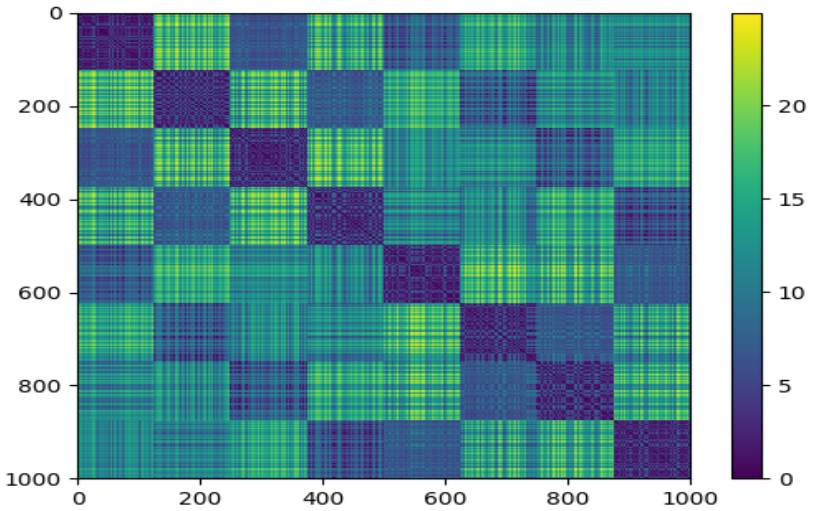
$$\langle w, \phi(x_i, y_i) - \phi(x_i, y_i) \rangle \geq \delta(y, y_i) - E_i$$

It can be solved efficiently if constraints are evaluated without heavy computation.

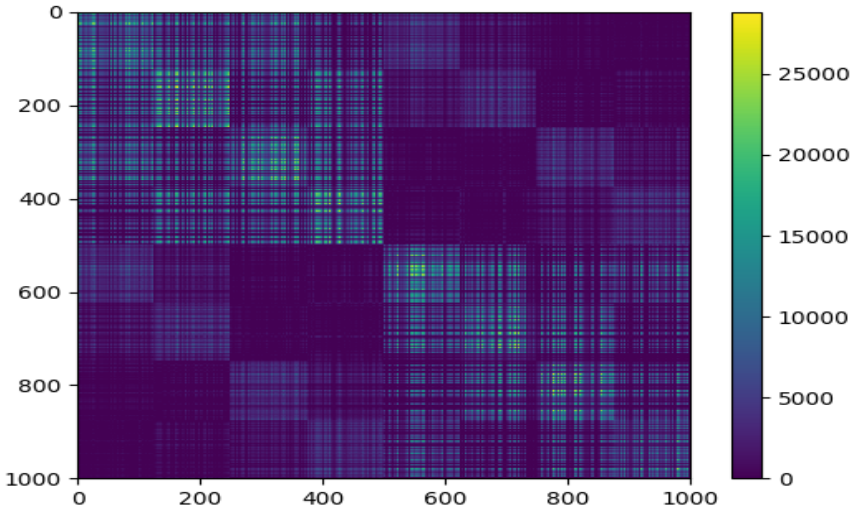
14 Results

Performance of different kernels can be visualised using the Euclidean Distance and Gram Matrix. Moreover, in the case of RBF kernel, three matrices have been computed for each of the σ value. From this experiment, it is easy to conclude that for RBF kernel a low magnitude of σ will assign large similarity values to nearby points whereas a higher σ will fail to distinguish the data. The diagonal has relatively low magnitude in the euclidean matrix but higher similarity value in the gram matrix for both polynomial and RBF kernel. This is because kernels assign high similarity value to nearby points. In RBF kernel, for lower σ values, the gram matrix is relatively dark. On the other hand for higher values the matrix is clear due to large variance. These matrices depend a lot on the type of datasets. In general, if there is a dataset in which all points are close then the corresponding gram matrix will be bright as the points are assigned higher similarity values. For better clarity refer to dataset images in the appendix. For the two spiral dataset, polynomial kernel doesn't give any valuable information as it assigns same similarity value to all data points. Due to this behaviour polynomial kernel gives only 51.8 percent accuracy on two spiral data.

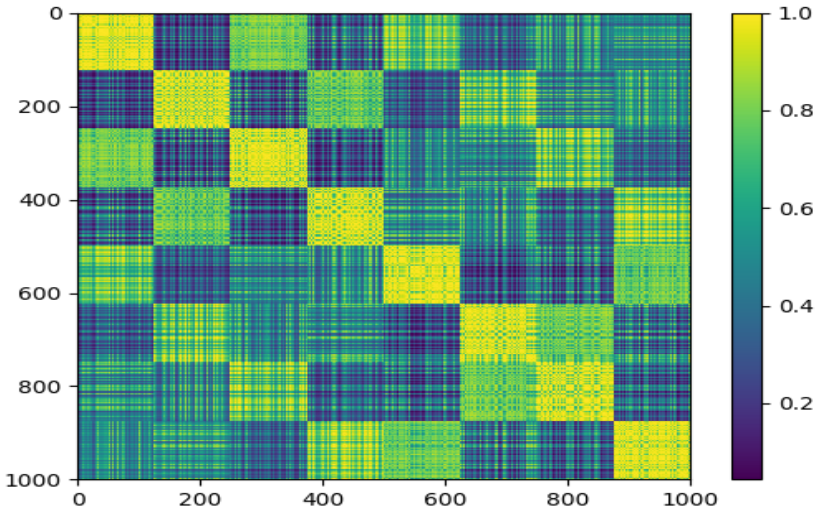
Corner Dataset



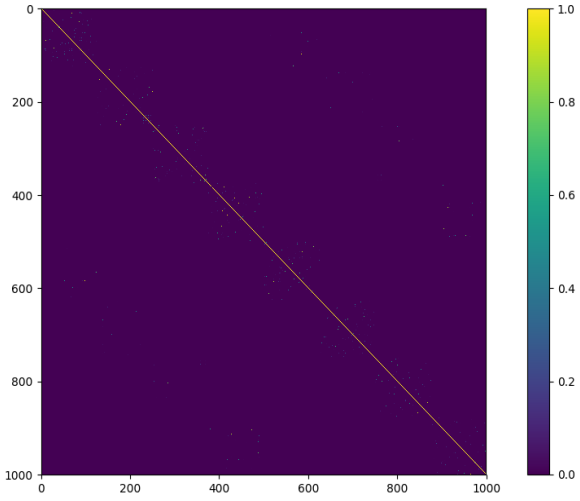
Euclidian Distance Matrix



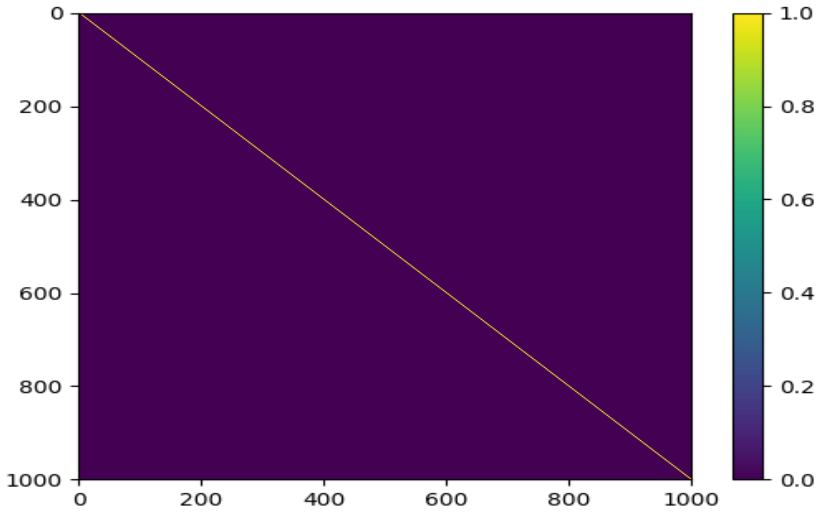
Polynomial kernel gram Matrix



RBF kernel gram Matrix with sigma 10

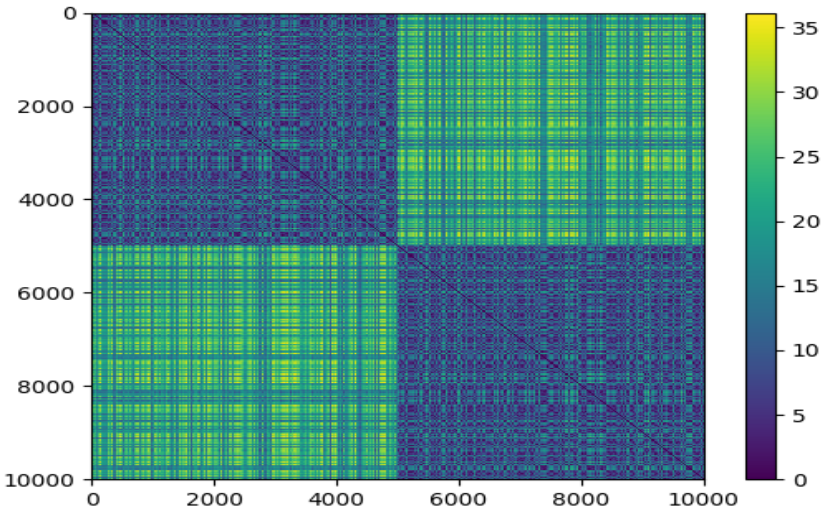


RBF kernel gram Matrix with sigma 0.1

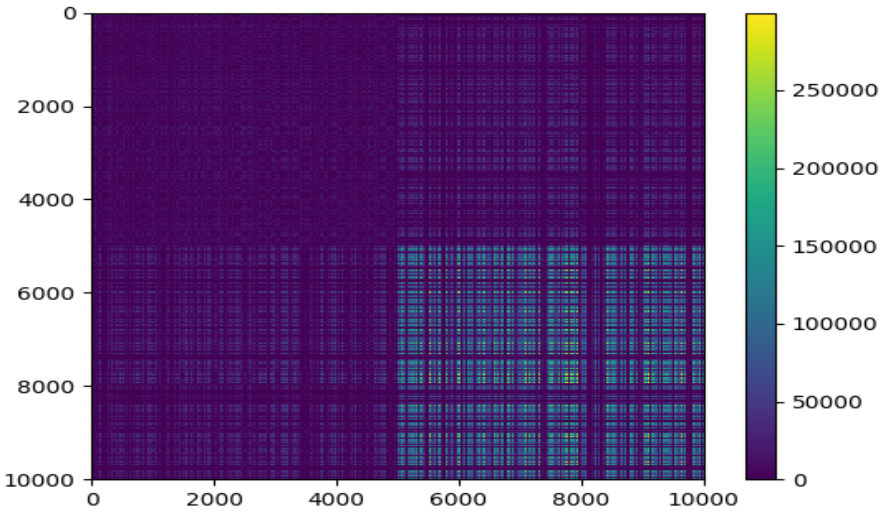


RBF kernel gram Matrix with sigma 0.001

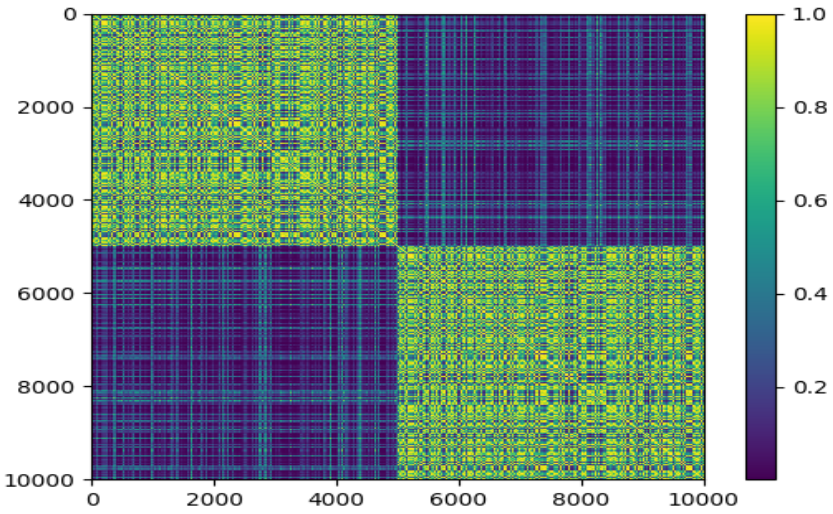
Double Moon Dataset



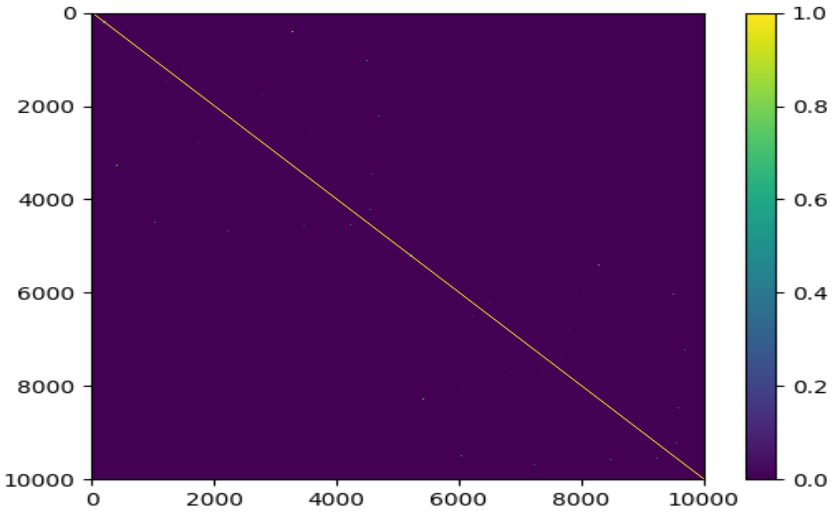
Euclidian Distance Matrix



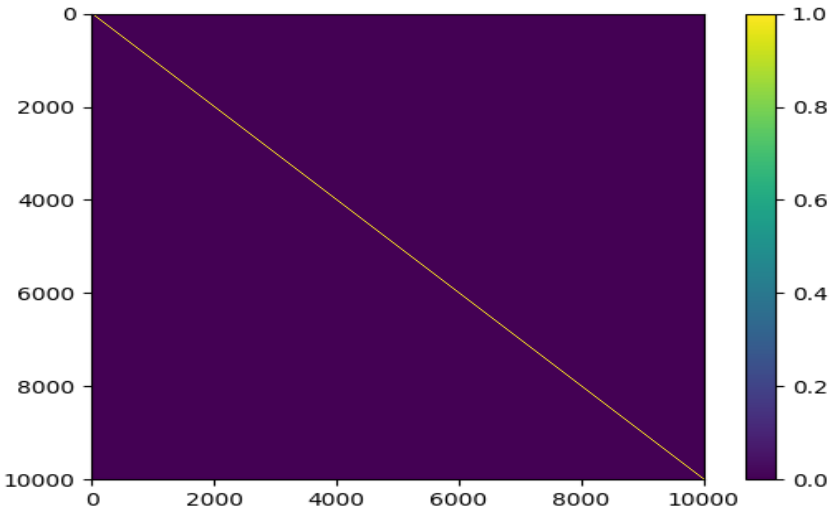
Polynomial kernel gram Matrix



RBF kernel gram Matrix with sigma 10

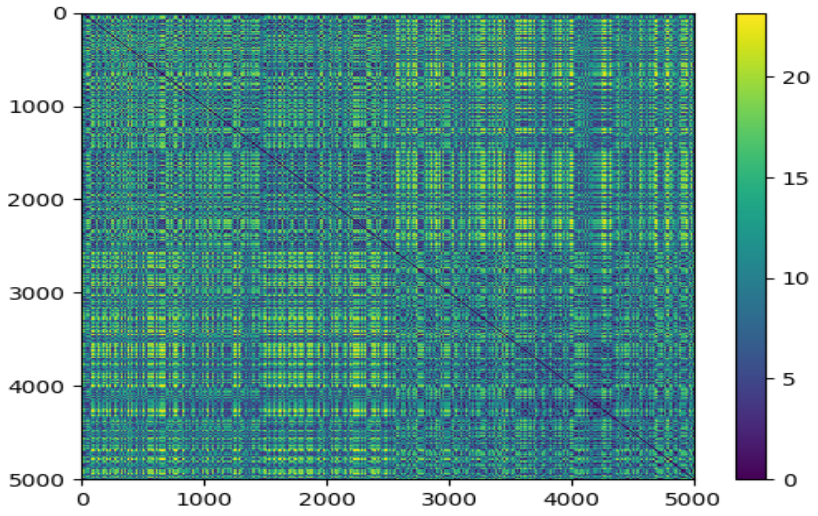


RBF kernel gram Matrix with sigma 0.1

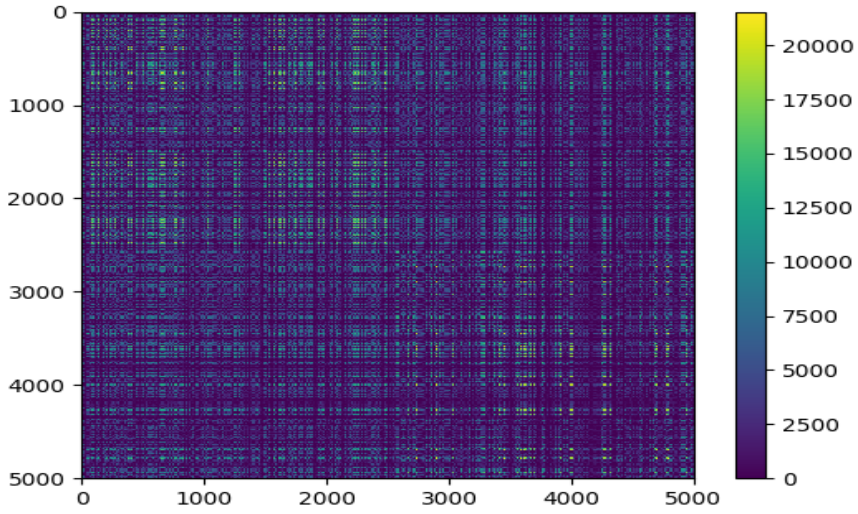


RBF kernel gram Matrix with sigma 0.001

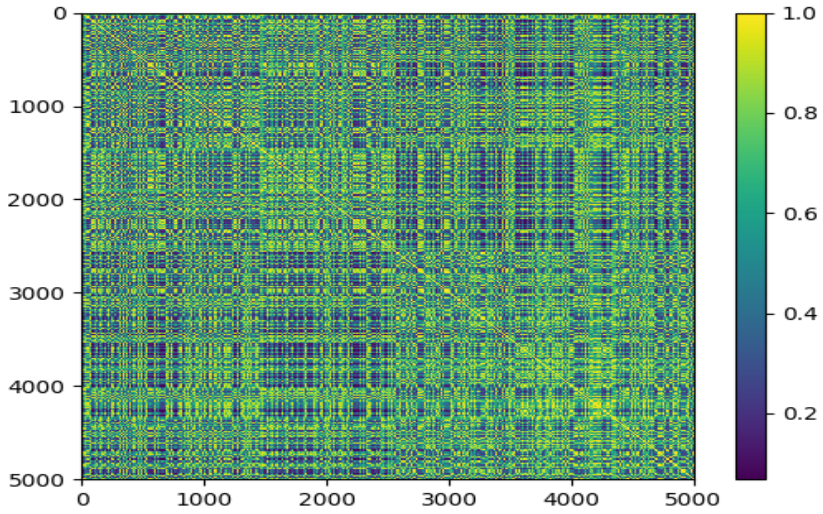
TwoSpiral Dataset



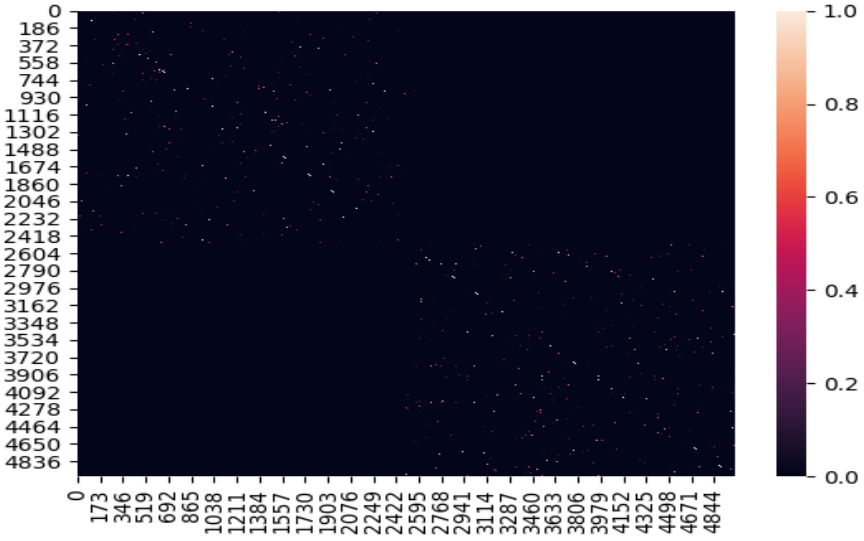
Euclidian Distance Matrix



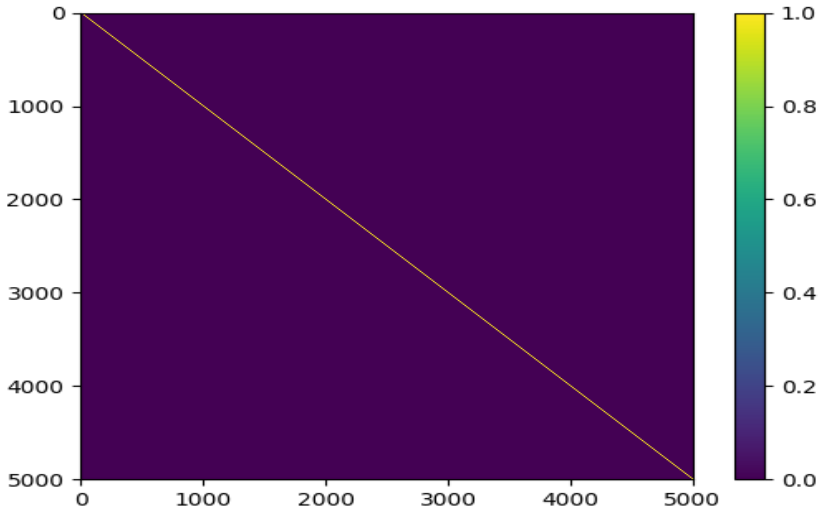
Polynomial kernel gram Matrix



RBF kernel gram Matrix with sigma 10

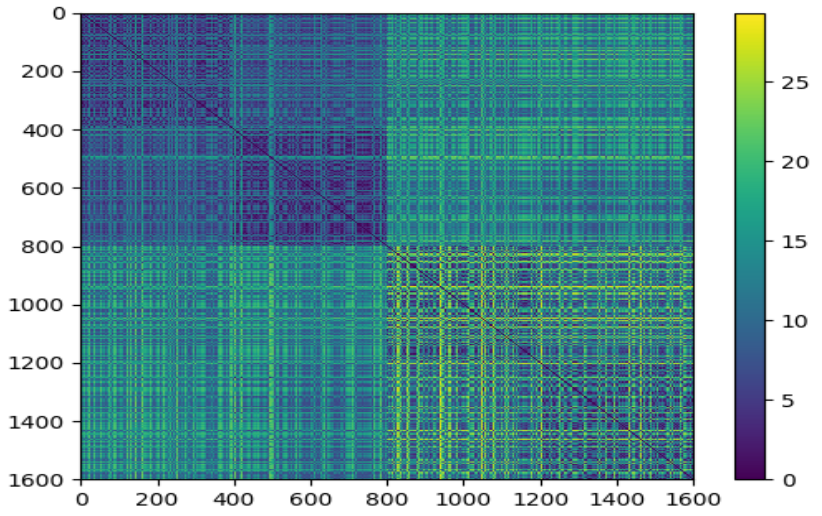


RBF kernel gram Matrix with sigma 0.1

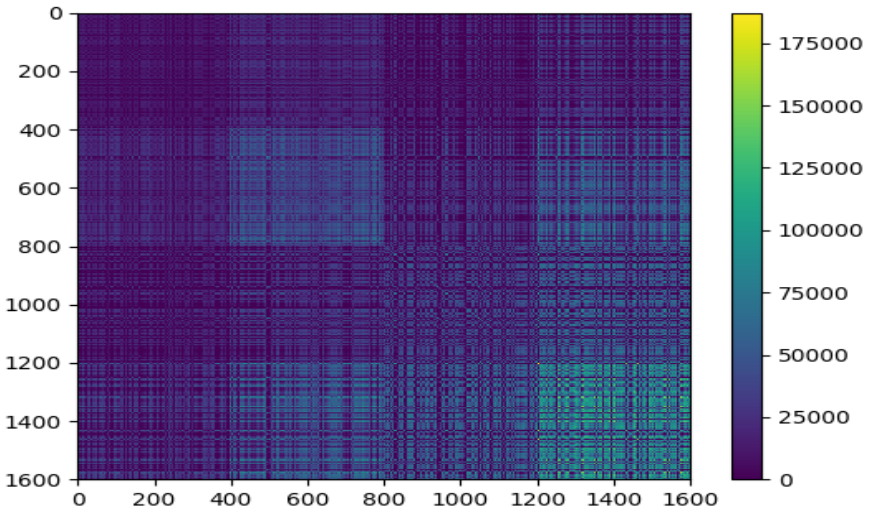


RBF kernel gram Matrix with sigma 0.001

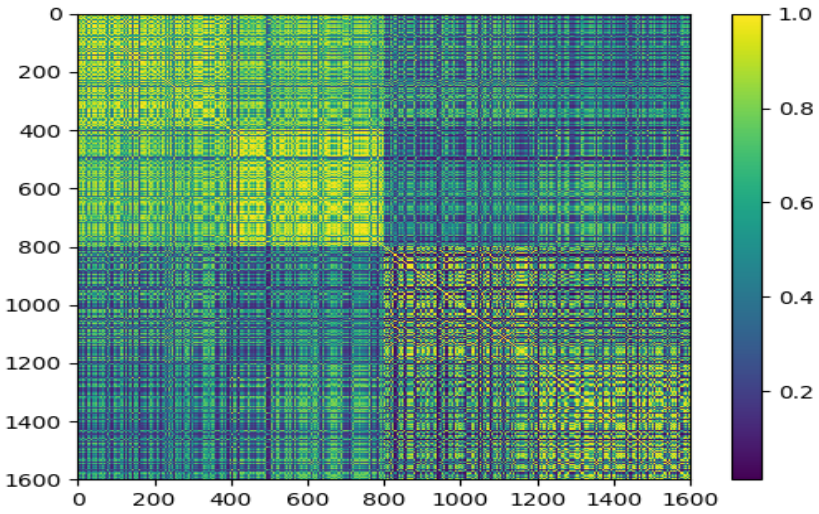
SwissRoll



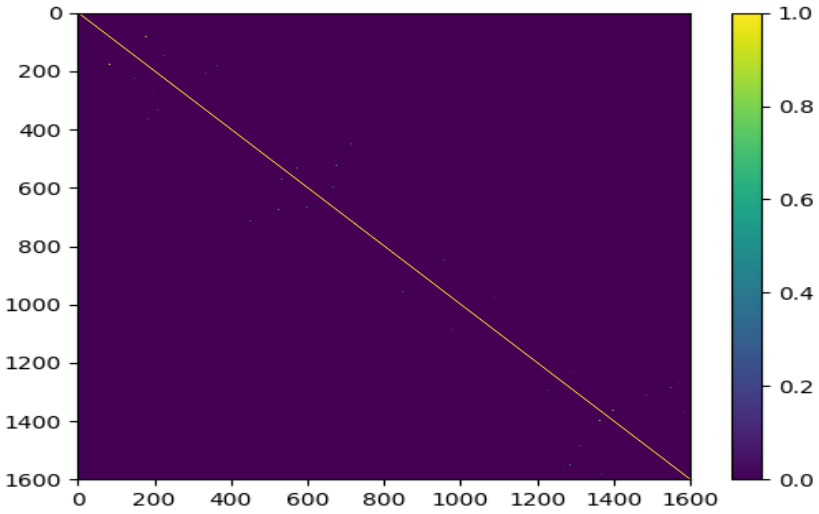
Euclidian Distance Matrix



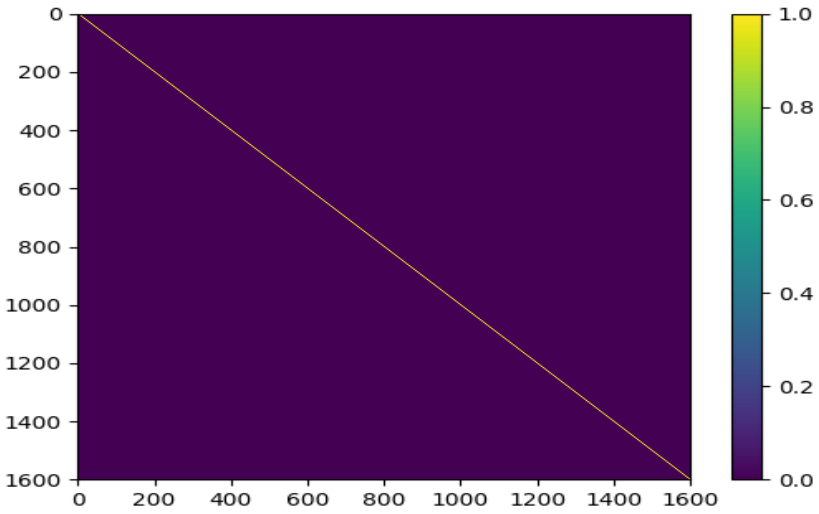
Polynomial kernel gram Matrix



RBF kernel gram Matrix with sigma 10



RBF kernel gram Matrix with sigma 0.1



RBF kernel gram Matrix with sigma 0.001

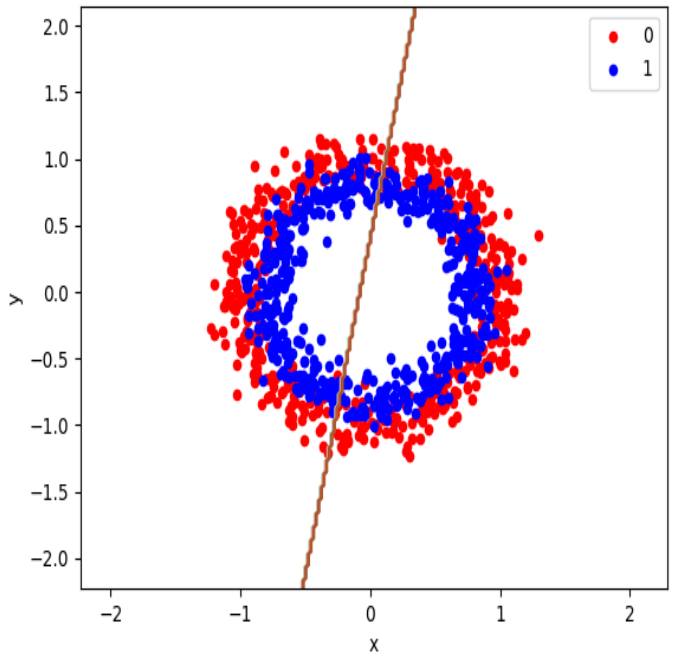
Accuracy of Generated datasets

Description of data in Appendix.

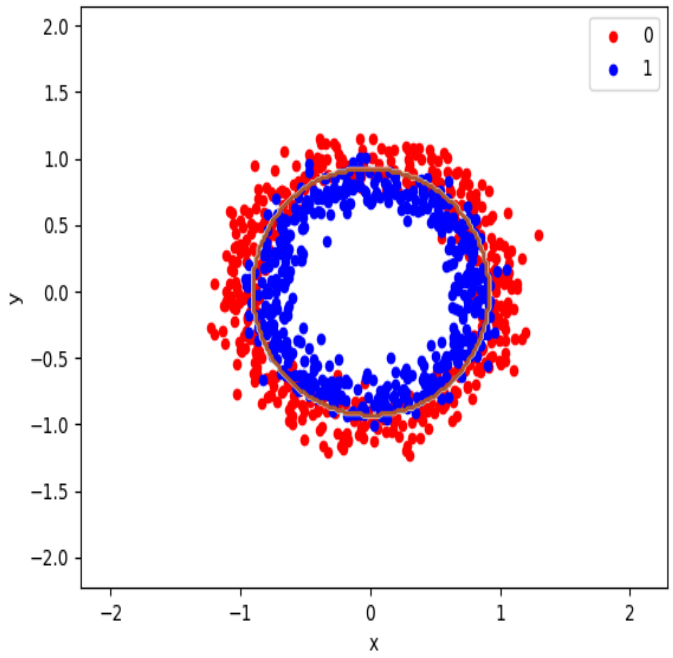
Dataset	Linear Kernel	RBF Kernel	Polynomial Kernel
DoubleMoon1	100	100	100
DoubleMoon2	100	100	100
Corners	100	100	100
TwoSpirals	64.45	100	51.8
CircleData	48	81.33	97.67
DoubleMoon3	85.33	86	86.33
Blob	85	86	86
CircleData2	49.67	83.33	83.33
DoubleMoon4	78.66	78.33	73.66
SwissRoll	76.458	97.916	97.916

Images of classifier for Generated datasets

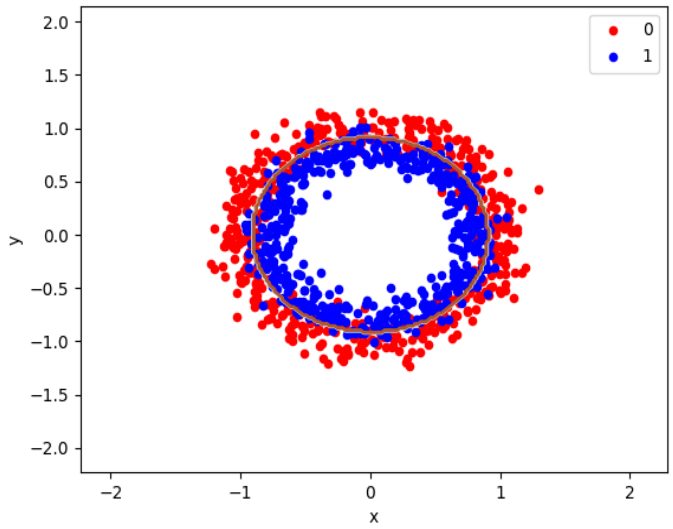
Cicles2 Dataset
Linear Kernel



RBF Kernel

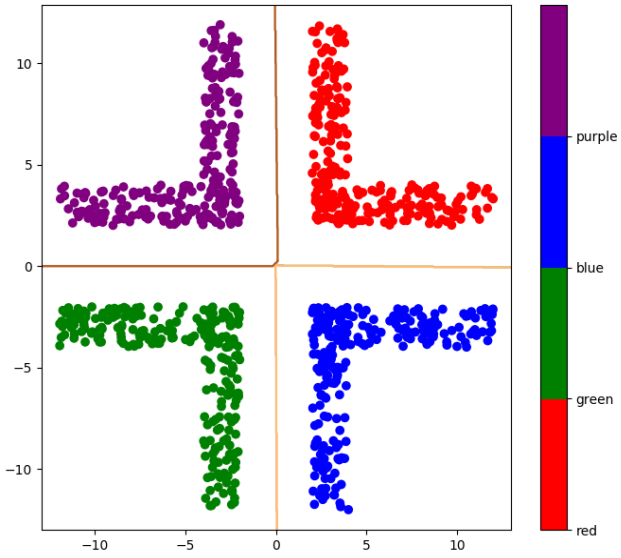


Polynomial Kernel

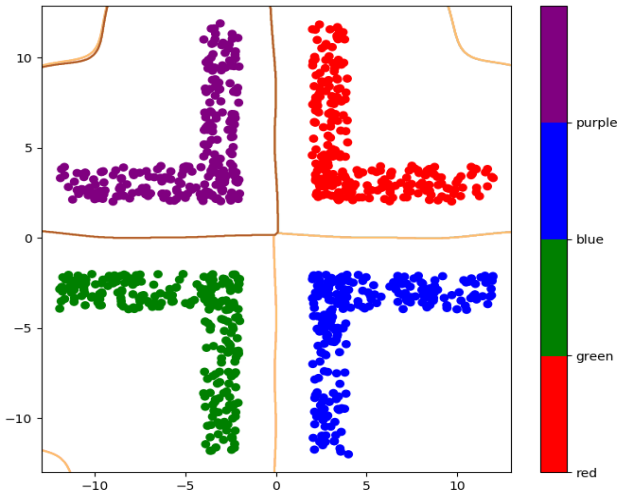


Corners Dataset

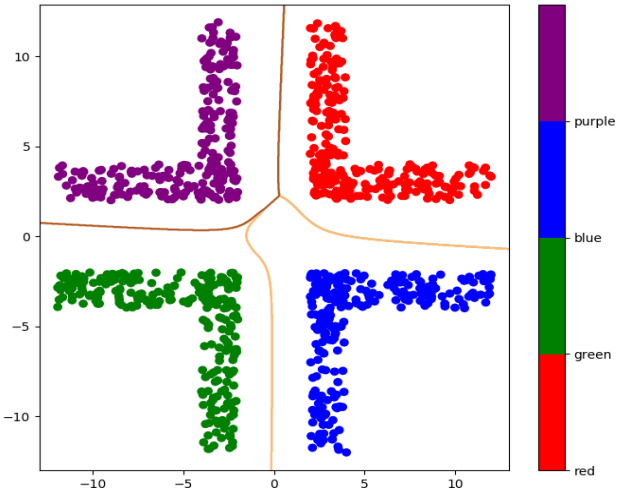
Linear Kernel



RBF Kernel

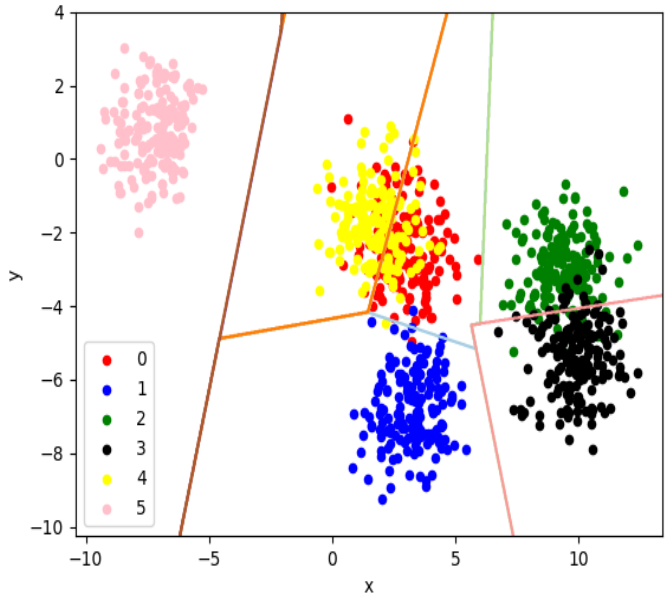


Polynomial Kernel

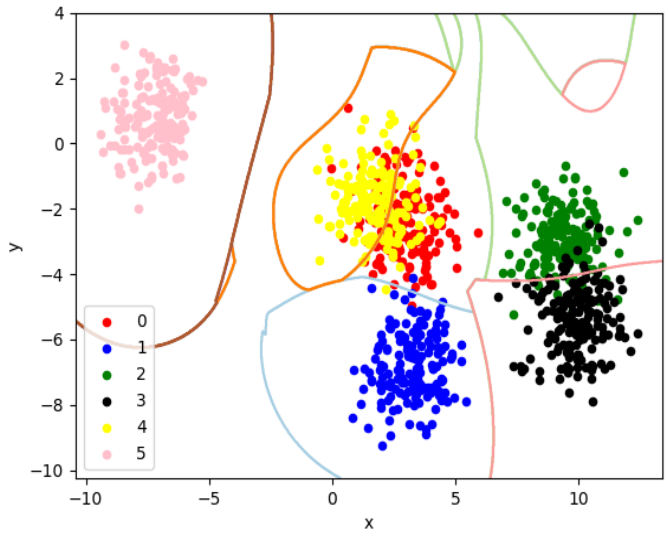


Blob Dataset

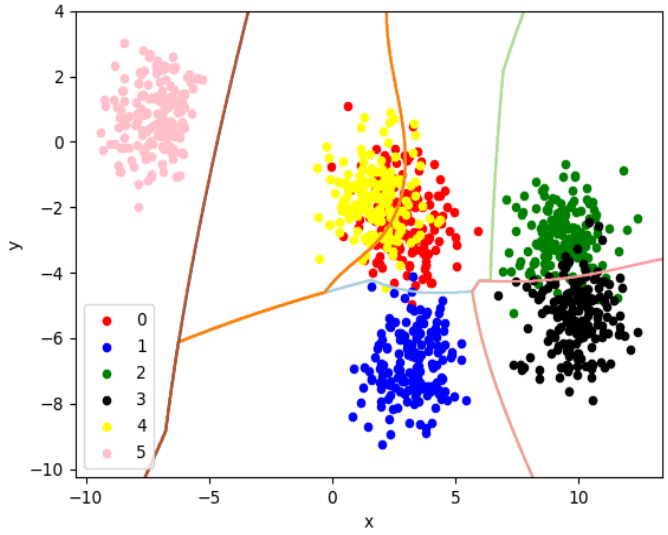
Linear Kernel



RBF Kernel

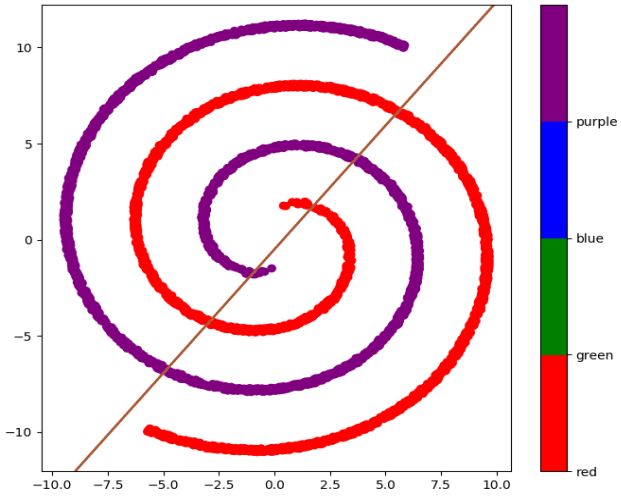


Polynomial Kernel

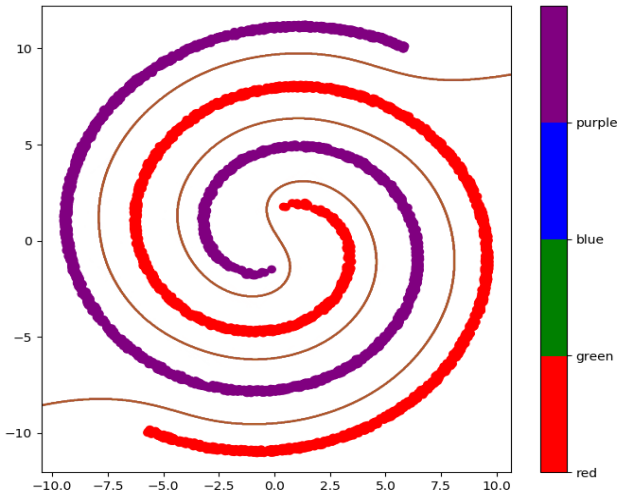


TwoSpirals Dataset

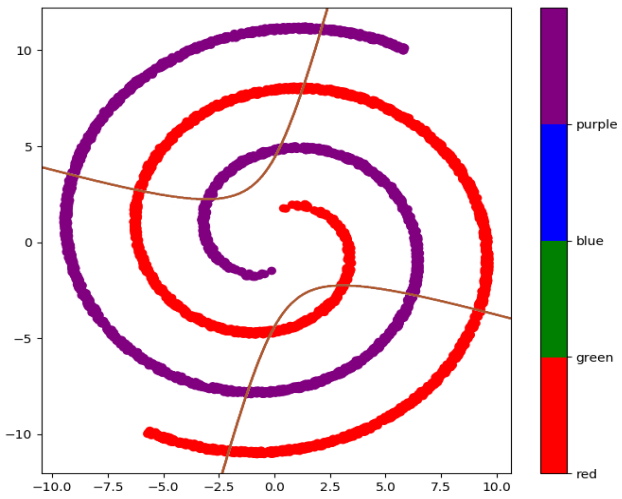
Linear Kernel



RBF Kernel

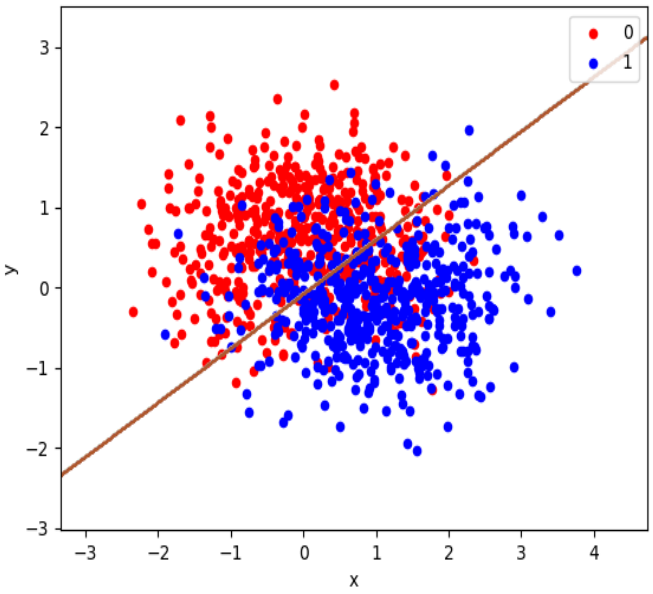


Polynomial Kernel

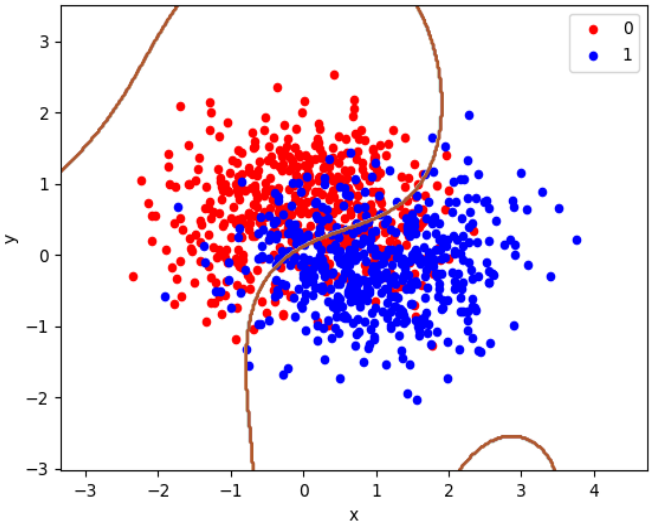


DoubleMoon4

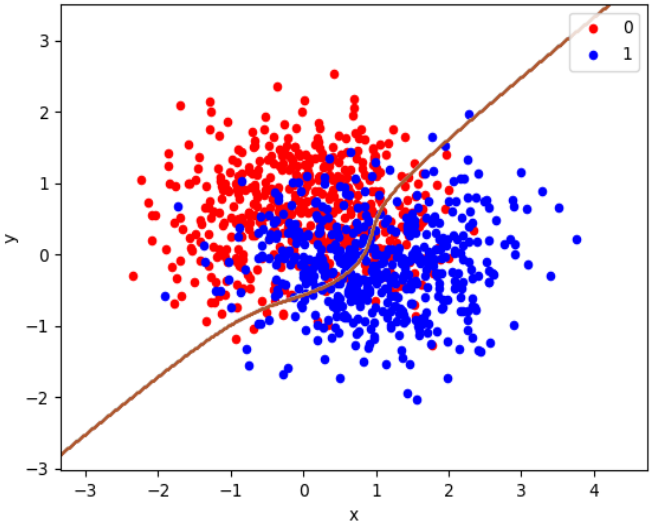
Linear Kernel



RBF Kernel



Polynomial Kernel



Accuracy for different parameters for Polynomial kernel

Generally, polynomial kernel uses a degree of 2 or 3 since with higher degrees there are greater chances to overfit the data. This behavior can be observed from the following experiment. In circle dataset, kernel with degree 2 performs better as the polynomial closely resembles the equation of circle. In double moon data, polynomial with degree 3 performs better since it efficiently matches the shape of the dataset. The images for these datasets are given in the appendix.

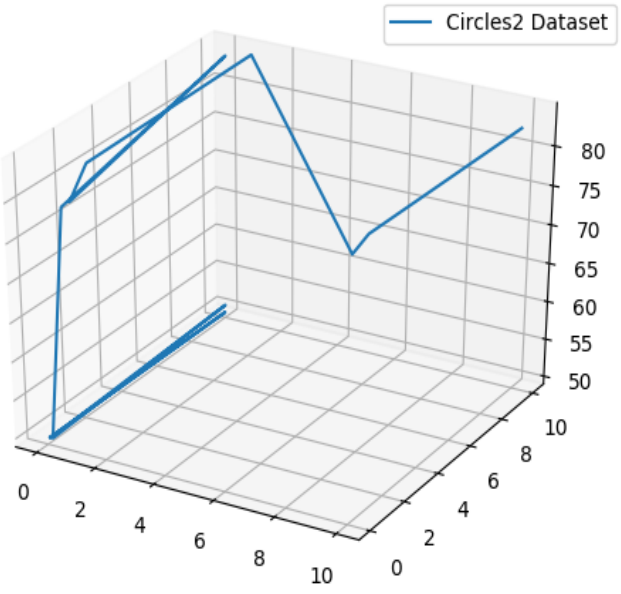
Dataset	Degree 2	Degree 3	Degree 4	Degree 5
Circle2	83.33	54	83	61.3
DoubleMoon4	62	73.67	59.33	71.67
Blob	84	86	84.33	84.66

Accuracy for different parameters for RBF kernel

It is well known that a large value of gamma results in higher variance and low bias. The choice of C and γ depend on the dataset used. For circular data, since more support vectors are needed for defining the shape, the value of γ has a important role as compared to C. Higher value of γ will provide better result due to high variance. For a noisy dataset such as double moon 4, the magnitude of C plays an important role in determining the result.

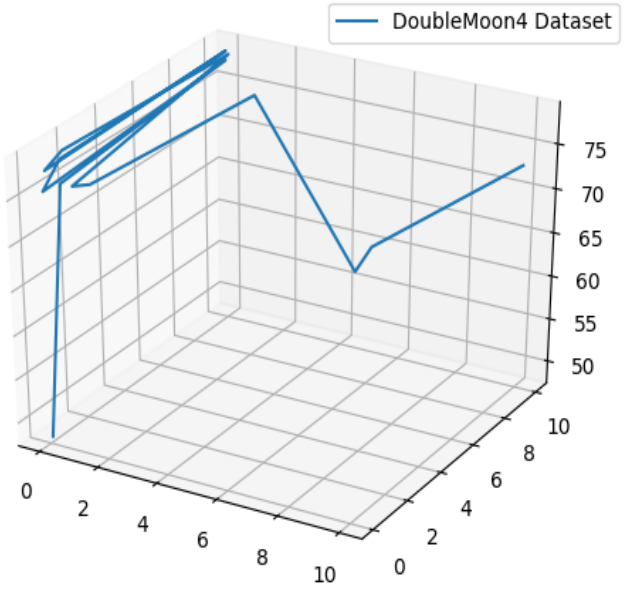
Circle2 Dataset

Gamma	C	Accuracy
0.001	0.1	49.67
0.001	1	49.67
0.001	10	49.67
0.01	0.1	49.67
0.01	1	49.67
0.01	10	50.67
0.1	0.1	49.67
0.1	1	77.33
0.1	1	83.33
1	0.1	80.66
1	1	84
1	10	84.66
10	0.1	82.33
10	1	83.33
10	10	82.66



DoubleMoon4 Dataset

Gamma	C	Accuracy
0.001	0.1	48
0.001	1	75.33
0.001	10	77.6
0.01	0.1	75.67
0.01	1	78
0.01	10	78.66
0.1	0.1	78
0.1	1	79
0.1	1	78.33
1	0.1	77
1	1	76
1	10	74.33
10	0.1	75
10	1	76.33
10	10	73



XOR Problem Illustration

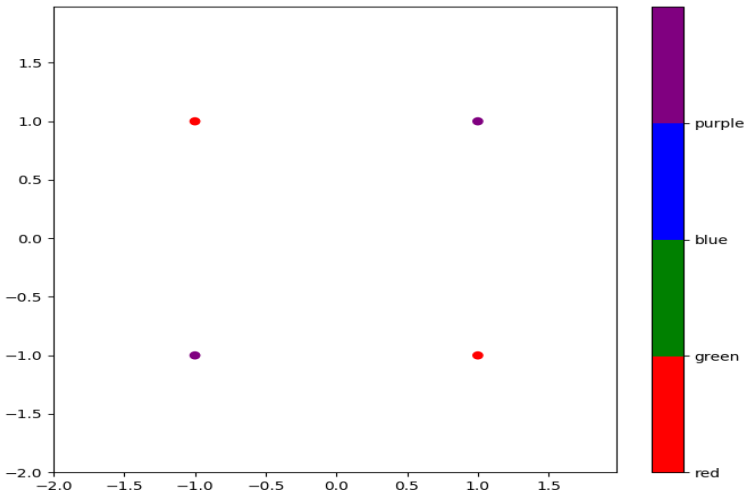


Fig: Two Dimensional Data

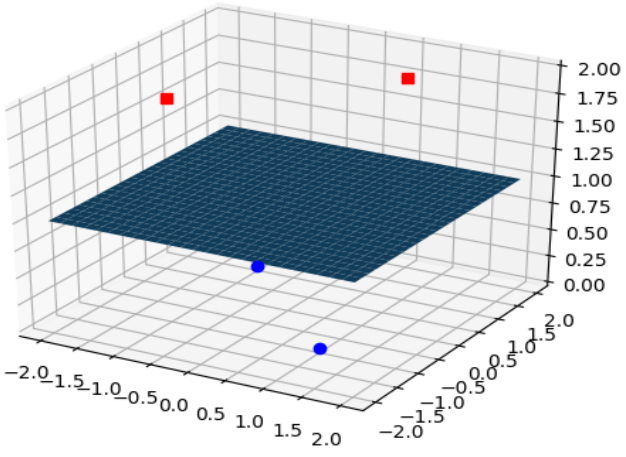


Fig: Three Dimensional Data

15 Appendix

15.1 Inner Product Space

A vector space X over the reals R is an inner product space if there exists a real-valued symmetric bilinear map, that satisfies

$$\langle x, x \rangle \geq 0$$

This bilinear map is known as the inner, dot or scalar product.

15.2 Positive semi-definite matrices

A symmetric matrix K is positive semi-definite, if it's eigen values are all non-negative. By Courant–Fisher theorem this holds if and only if

$$c'Kc \geq 0$$

or

$$\sum_{i,j} c_i cj K_{ij} \geq 0$$

for all $c_i \epsilon R$. Gram and Kernel matrices are poistive semi-definite.

15.3 Dichotomy

It can be defined as a partition of a set S int two disjoint subset. Just like we define the hypothesis for our input space X, we can define dichotomy in a similar way on some points selected from this input space instead of the whole input space.

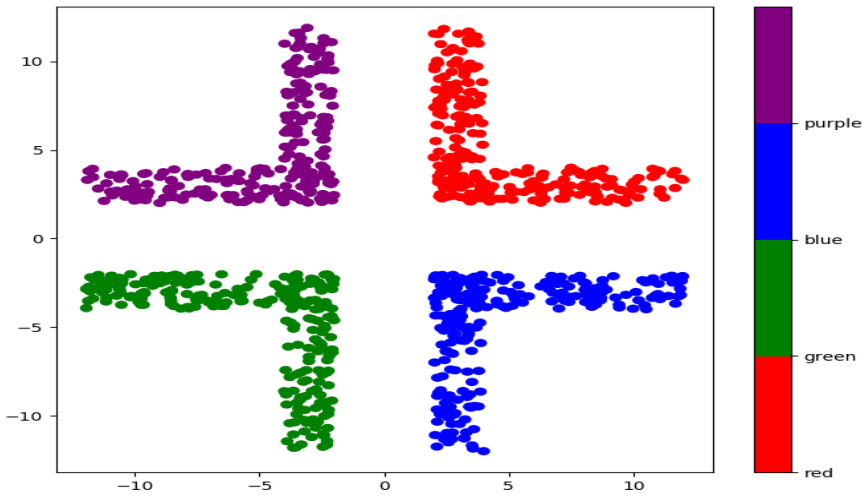
15.4 Growth Function

The growth function counts the most number of dichotomies on N points. It is given by $m_H(N)$. The growth function satisfies $m_H(N) \leq 2^N$ i.e all possible assignment of labels to n points.

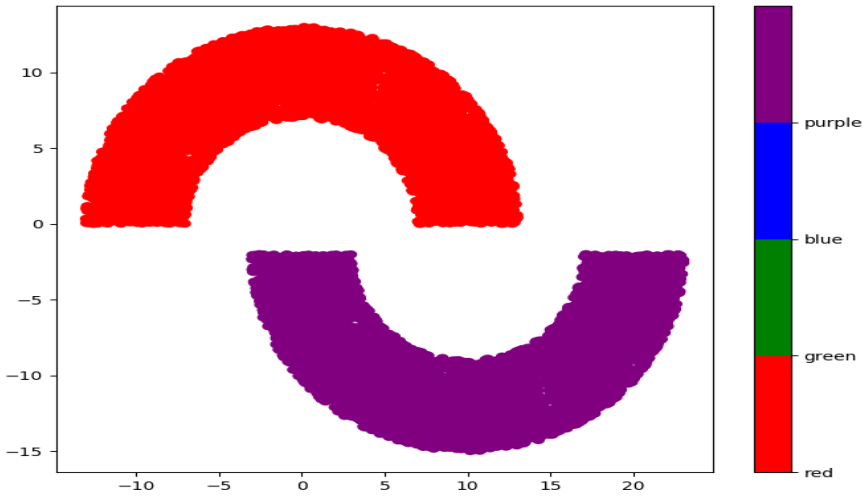
15.5 VC Dimension

VC dimension for a hypothesis set H can be defined as the largest value of N for which $m_H(N) = 2^N$, i.e., the most points H can shatter. VC dimension corresponds to the degree of freedom and thus on the effective number of parameters.

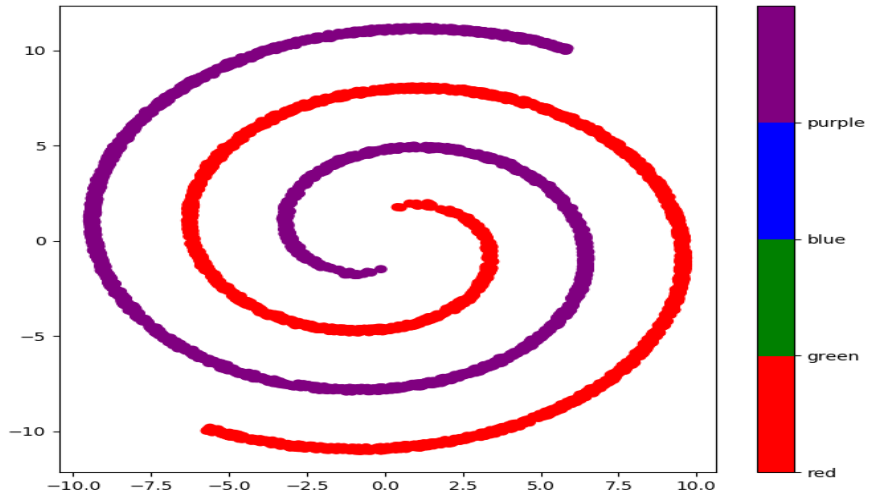
15.6 Datasets



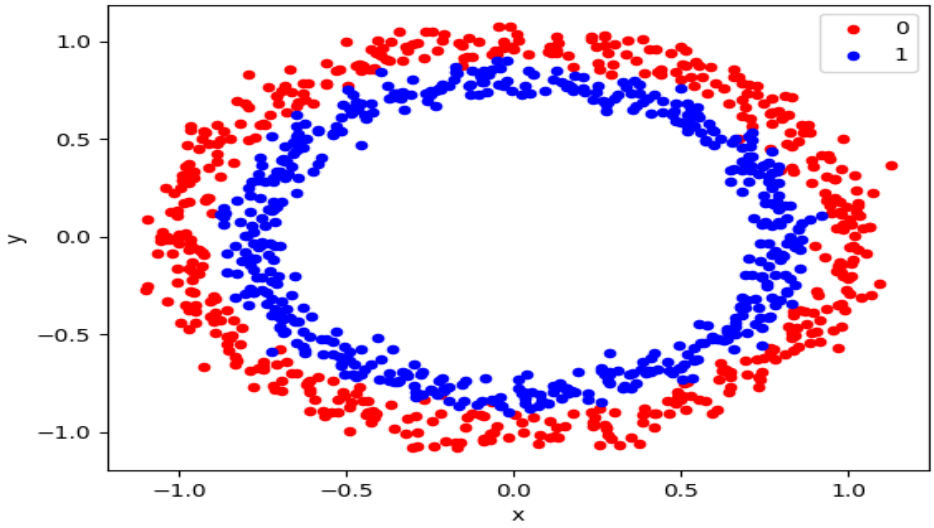
Corner Dataset



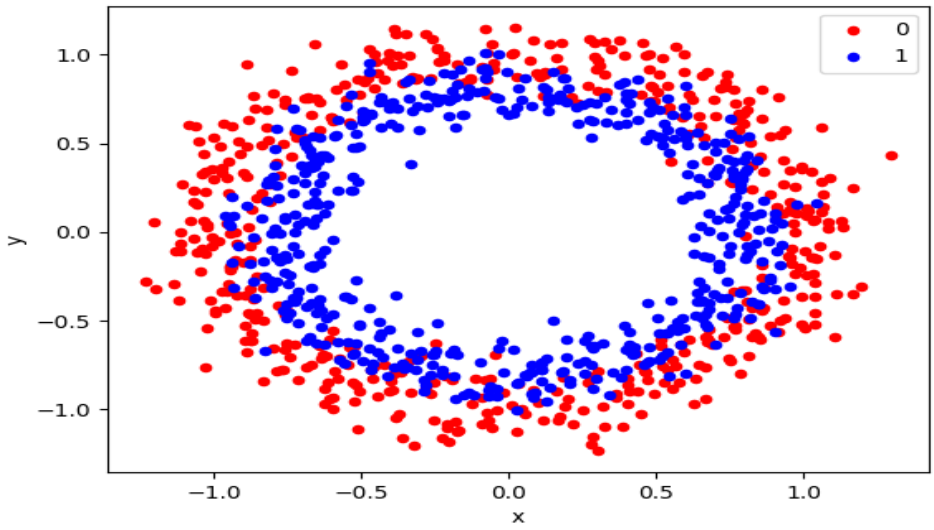
Double Moon Dataset



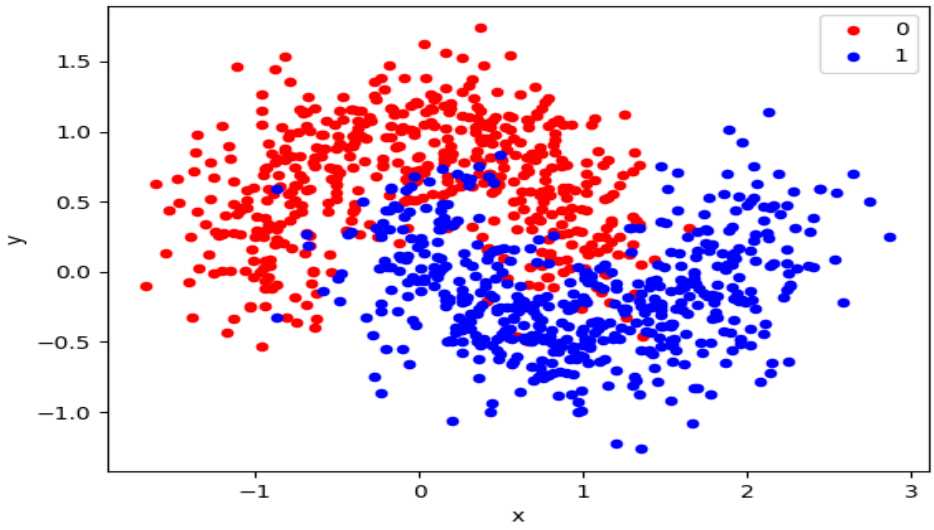
Two Spiral Dataset



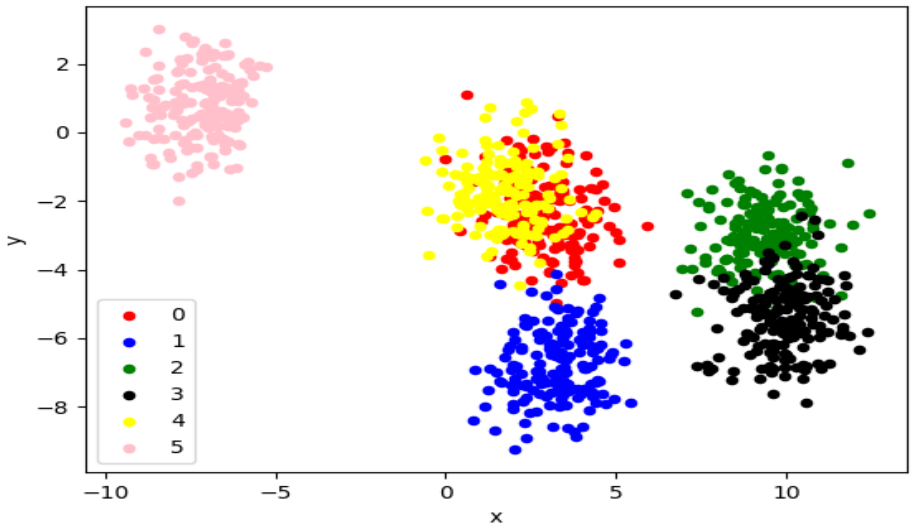
Circle Dataset



Circle2 Dataset



Double Moon 3 Dataset



Blob Dataset

References

[1] Thomas Hofmann, Bernhard Schölkopf and Alexander J. Smola [*KERNEL METHODS IN MACHINE LEARNING*]. The Annals of Statistics, 36(3):1171-1220, 2008.

[2] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, United States of America, 2004.

[3] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. [*Scikit-learn: Machine Learning in Python*]. Journal of Machine Learning Research, 12:2825–2830, 2011.

[4] Mostafa Samir: Machine Learning Theory - Part 1,Part 2
`https://mostafa-samir.github.io/ml-theory-pt1/`

[5] Taku Kudo and Yuji Matsumoto [*Fast Methods for Kernel-based Text Analysis*]. Association for Computational Linguistics, 41:24-31, July 2003.