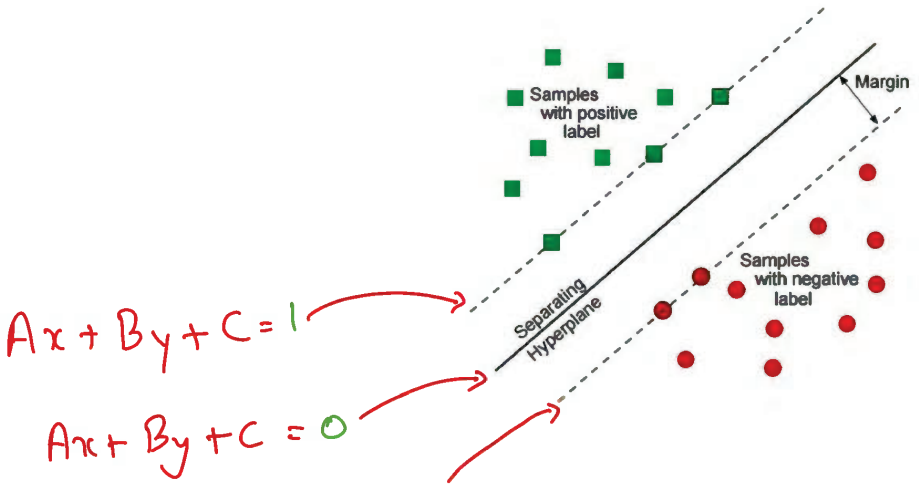
SVM

Maximal (Hard) Margin Classifier

A Maximal Margin Classifier is the foundation of Support Vector Machine (SVM) for linearly seperable data

The maximal margin classifier finds the hyperplane that seperates two classes with the largest possible margin - The distance between the hyperplane and the nearest data points from each class.



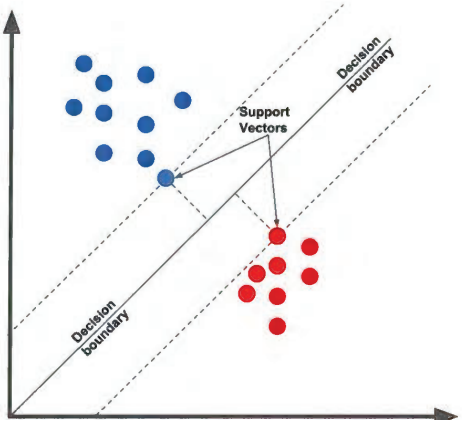
Why this special form eq? $Ax + By + C = 1/0/-1$?

Because equation in this form are always parallel. And the R.H.S denotes what region does the point lie. $1 \rightarrow$ +ve region, $-1 \rightarrow$ -ve region

Support Vectors

Support vectors are very important foundation in SVM.

Support vectors are simply the base (foundation), of the data. Meaning the first wall of the data. Anything that comes from outside meets the first points (first wall) which are nothing but support vectors.



Mathematical Intuition.

Simply, we have to check if that point is in the +ve or -ve region from the hyperplane.

$$Ax + By + C \geq 1 \quad \text{if } y_i = 1$$

$$Ax + By + C \leq 1 \quad \text{if } y_i = -1$$

We combine these equations in one by multiplying the class region of the point. if it's +ve then we multiply with 1 else -1

$$\text{Argmax}_{A,B,C} \xrightarrow{\text{such that}} \boxed{y_i(Ax_{i1} + Bx_{i2} + C) \geq 1}$$

↓

To maximize the distance d between two margins

↓

To find the distance between two parallel lines (margins), we use the standard formula

$$d = \frac{|C_2 - C_1|}{\sqrt{a^2 + b^2}}$$

Therefore, substituting this, $d = \frac{|c+1-c+1|}{\sqrt{a^2+b^2}} = \frac{2}{\sqrt{a^2+b^2}}$

$$\text{Argmax}_{A,B,C} \frac{2}{\sqrt{A^2+B^2}} \text{ given } \{ y_i(Ax_i + Bx_{i2} + C) \geq 1 \}$$

$$\underset{A\,B\,C}{\text{argmax}} \quad \frac{2}{\sqrt{A^2+B^2}} \quad \text{given} \left\{ y_i \left(A x_i + B x_{i2} + C \right) \geq 1 \right\}$$

Solving:

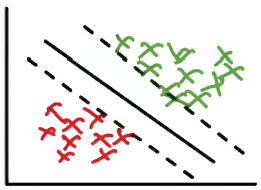
We have to simply find the values of A, B, C in such a way that the value of d (distance) is maximum. (We generally use gradient descent) But here the catch is that we also have to maintain the constrain given with finding the max values.

This is called as Constraint Optimization Problem.

This is where the math is a bit more complex. We use Quadratic programming to solve this.
This is complex advance math, which we will not do.

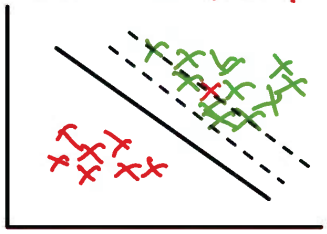
But by using this math, we finally get the values of A, B, C with maximum distance (d) and within the constraint.

Problems with Hard Margin SVM:



1. This requires linearly seperable data, which is often not possible in real world.

2. Extremely Sensitive to Outliers: If even one single data point is on the other class region, this will not work at all. This is because that outlier will act as a Support Vector



Soft Margin SVM

Slack Variable also called as Hinge loss function

The concept of slack variables is used in the formulation of "soft-margin" SVM to handle cases where data is not linearly seperable, or when one allows for some degree of error in classification.

Mathematically, for each data point, a slack variable $\xi_i \geq 0$ is introduced. The slack variable ξ_i measures the degree of misclassification of the data point x_i.

$\longrightarrow$ $\xi_i = 0$ if x_i is on the correct side of margin

$\longrightarrow$ $0 < \xi_i < 1$ if x_i is on correct side of the hyperplane but on the wrong side of the margin

$\longrightarrow$ $\xi_i \geq 1$ if x_i is on the wrong side of the hyperplane. It is misclassified.

Hinge Loss function $= \max(0, 1 - (y_i(Ax_i + Bx_{2i} + C)))$

If we fit this into SVM, it looks like this:

$$y_i(Ax_{i} + Bx_{2i} + C) \geq 1 - \xi_i$$

But, now this is no longer a constrained formula, because "$1 - \xi_i$" allows all the points to be a part of any class.

So, we tweak the hard margin SVM loss function.

taking inverse of $\frac{\sqrt{A^2+B^2}}{2}$, because since this follows the naming loss function, by inversing, now we have to minimize this to get max distance. And, we take average of the "ξ_i".

$$\underset{\beta_0\,\beta_1\,\beta_2}{\operatorname{argmin}} \quad \frac{\sqrt{\beta_1^2+\beta_2^2}}{2} + \frac{1}{n}\sum_{i=1}^{n}\xi_i$$

Therefore, this finally becomes

$$\underset{\beta_0\,\beta_1\,\beta_2}{\operatorname{argmin}} \quad \frac{\sqrt{\beta_1^2+\beta_2^2}}{2} + \frac{1}{n}\sum_{i=1}^{n}\xi_i$$

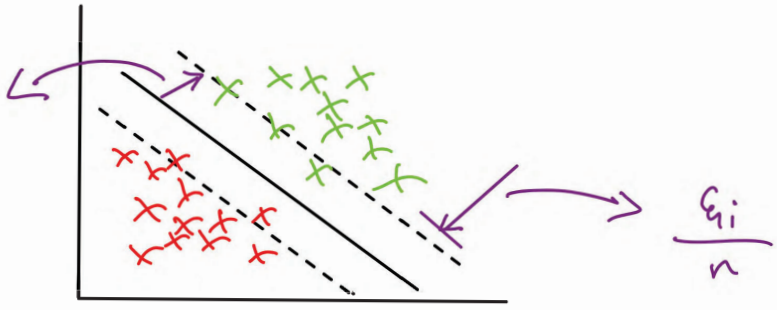
such that

$$y_i\,(\beta_1 x_{1i} + \beta_2 x_{2i} + c) \geq 1 - \xi_i$$

for all x_i and $\xi_i \geq 0$

Therefore here, The "$\frac{\sqrt{\beta_1^2+\beta_2^2}}{2}$" tries to expand the margins, while the "$\frac{\xi_i}{n}$" tries to shrink the margins.

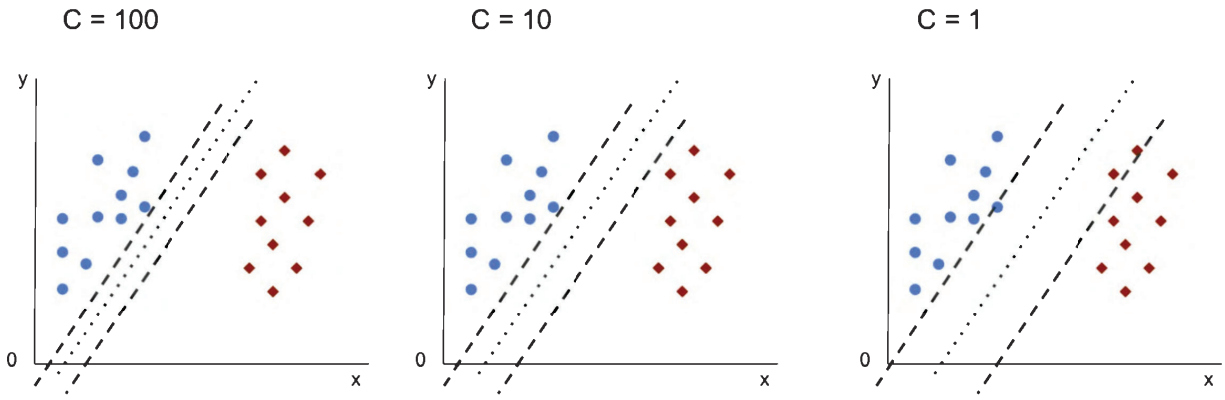
$\frac{\sqrt{\beta_1^2+\beta_2^2}}{2}$



But Since there's no control over $\frac{\xi_i}{n}$, therefore we add "C" to control and do hyperparameter tuning. By this we will be able to control both parameters.

$$\underset{\beta_0\,\beta_1\,\beta_2}{\text{Argmin}} \frac{\sqrt{\beta_1^2+\beta_2^2}}{2} + C\frac{1}{n}\sum_{i=1}^{n}\xi_i$$

C → new hyperparameter.



Bias Variance tradeoff

$$\underset{\beta_0\,\beta_1\,\beta_2}{\text{Argmin}} \frac{\sqrt{\beta_1^2+\beta_2^2}}{2} + C\frac{1}{n}\sum_{i=1}^{n}\xi_i$$

Increasing the value of c will shrink the margins (push inwards)

high c → Overfitting (low bias / High Variance)

low c → Underfitting (high bias / low Variance)

SVM - Support Vector Machines

SVM are supervised learning algorithms used for classification, regression, and outlier detection.

The flow :

1) Input data ⟶ transformed into higher dimension using kernel.

2) Apply SVC (generally)

3) Project down from top to get the view.

Kernel.

This is simply a function that converts the data from lower to higher dimension

There are four types of kernels

1. linear
2. RBF - Radial basis function
3. Poly
4. Sigmoid

The Trick

The trick here is that the dimension change never actually happens here! Why? How? Explained below. with math.

Constrained Optimization Problem

let's start with a simpler problem to understand the topic better.

Our Main function

$$\underset{x,y}{\text{argmax}} \; x^2 y \quad \text{such that} \quad \underbrace{x^2 + y^2 = 1}_{\text{Constraint}}$$

firstly, we increase the dimension, and then plotting the graph for x^2y in 3-Dimension

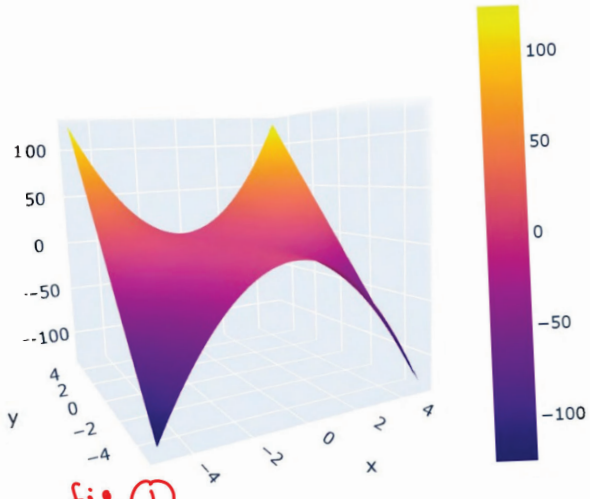


fig. ①

plot for x^2y

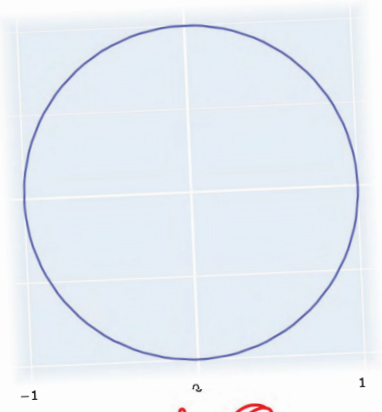
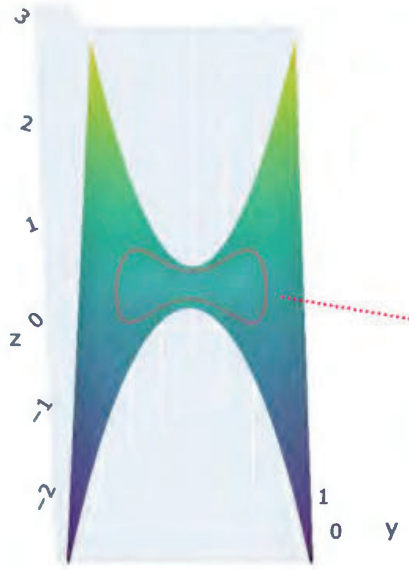


fig ②

plot for constraint $x^2 + y^2 = 1$

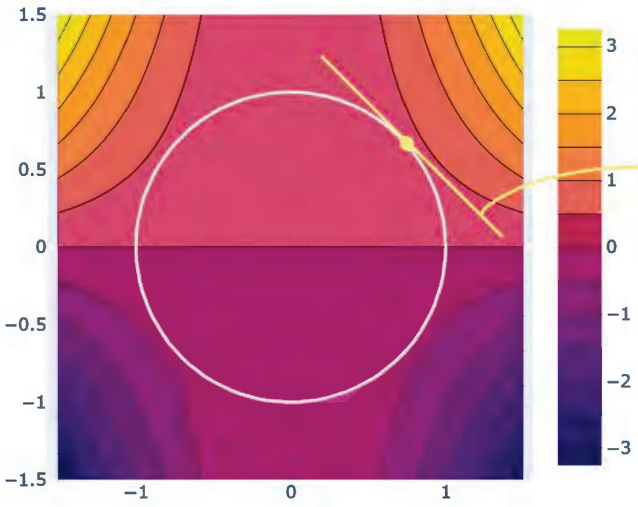
In figure ①, we can notice that the gellow region has the max values for z-axis. But since, this is a constraint problem, we plot the constraint inside the main function's graph to make sure the the values we get are within the constraint, This makes sure that we are following the given condition along with finding the maximum value for function x^2y.



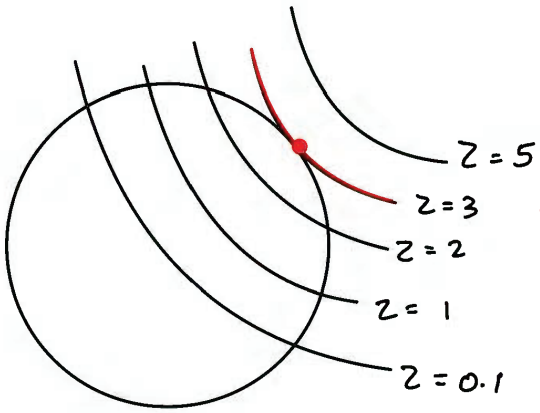
Now that we have the constraint too, we can go ahead and get the maximum value of x^2y.

$x^2+y^2=1$ is plotted on x^2y

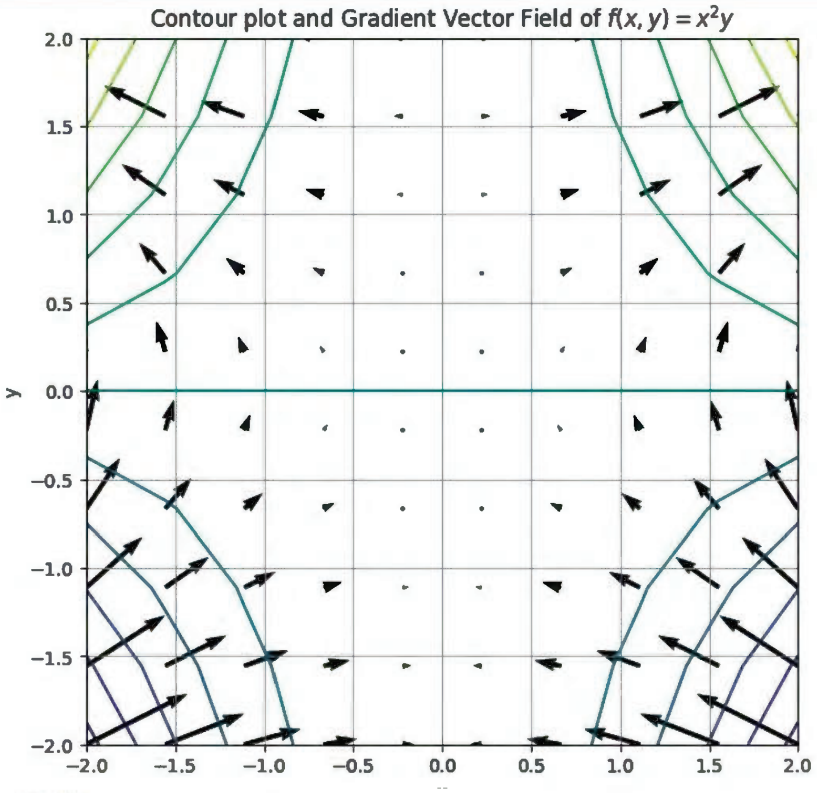
plotting the contour plot for better understanding.



The yellow region represents more value of z as compared to purple region



The further we go towards the direction where value of z increases, there comes a point where the value of z is maximum within the constraint, This is what we have to find. But how do we know where is the z value increasing?



The gradient Vector field is simply a visualization, that points in the direction of the steepest (fastest) increase of the function. (increase of value z here)

The gradient of a vector field is denoted as ∇f

Gradient Vector Field

The gradient of a function at a point is a vector that points in the direction of the steepest ascent of the function at that point. The magnitude (or length) of the gradient vector is equal to the rate of increase of the function in that direction.

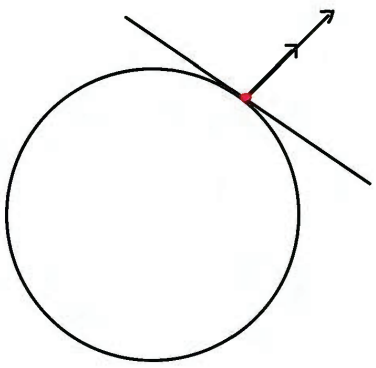
Contour lines (or level sets) of a function are curves that connect points where the function has the same value. For a 2D function, these are like the lines of constant altitude on a topographic map.

There are a few key relationships between gradients and contour lines:

The gradient at a point is perpendicular (or orthogonal) to the contour line passing through that point. This is because the contour line represents the direction of no change in the function value, while the gradient represents the direction of maximum change.

1. [redacted] If you were to walk along the contour line (where the function value doesn't change), the direction you'd need to go to start climbing as steeply as possible is the direction of the gradient.
2. The magnitude of the gradient (how long the gradient vector is) indicates how steeply the function is increasing. If the contour lines are close together, that means the function is changing rapidly, so the gradient is large. If the contour lines are far apart, the function is changing slowly, so the gradient is small.

Therefore, we come to a conclusion,



The point where the vectors have the same direction. Make sure that the magnitude of the vectors are different.

Therefore we get this formula

$$\nabla f(x,y) = \lambda \nabla g(x,y)$$

$f(x,y) = x^2 y$ — λ: Scaler — $g(x,y) = x^2 + y^2$ (constraint)

This Scaler 'λ' is also known as "Lagrange's Multiplier"

Now let's solve this...

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{\partial (x^2 y)}{\partial x} \\ \frac{\partial (x^2 y}{\partial y} \end{bmatrix} \longrightarrow \begin{bmatrix} 2xy \\ x^2 \end{bmatrix}$$

Same for $\nabla g(x,y)$

$$\nabla g(x,y) = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} \longrightarrow \begin{bmatrix} \frac{\partial (x^2+y^2)}{\partial x} \\ \frac{\partial (x^2+y^2)}{\partial y} \end{bmatrix} \longrightarrow \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

Therefore we get

$$\begin{bmatrix} 2xy \\ x^2 \end{bmatrix} = \lambda \begin{bmatrix} 2x \\ 2y \end{bmatrix}$$

Solving these in form of equations

$2xy = \lambda 2x$

$x^2 = \lambda 2y$

$x^2 + y^2 = 1$ ⟶ This third equation is the constraint.

{ Since we can't find x, y, z using 2 eq, we have the 3rd equation already.

On Solving the equations, we get these values for x, y

$$\left(\sqrt{\frac{2}{3}}, \frac{1}{\sqrt{3}}\right) \quad \left(\sqrt{\frac{2}{3}}, \frac{-1}{\sqrt{3}}\right) \quad \left(-\sqrt{\frac{2}{3}}, \frac{1}{\sqrt{3}}\right) \quad \left(-\sqrt{\frac{2}{3}}, -\frac{1}{\sqrt{3}}\right)$$

Only these two values are considered since their y is +ve.

Substituting in x^2y, we get that both values are maximum.

Remember, since the plot showed that it's symmetrical, we always get two max values. you can consider any!

And we can notice that these x, y values also follow the constraint.

We get: $\left[\underset{x,y}{\arg\max} \; x^2y \qquad x^2+y^2-1 \right]$

$\downarrow$ optimization

$$\left[L(x, y, \lambda) = \underset{x,y}{\arg\max} \; f(x,y) - \lambda\left(g(x,y)-1\right) \right]$$

Lagrangian function
(L = symbol of lagrangian)

$\downarrow$ we finally get the most optimised form.

$$L(x, y, \lambda) = \underset{x,y,\lambda}{\arg\max} \; x^2y - \lambda\left(x^2+y^2-1\right)$$

This is called as Lagrange's Multiplier

Summary: We simply converted the constraint optimization problem into Lagrange's Multiplier.

We specifically converted it to Lagrange's multiplier because Lagrange's multiplier equation's are very computationally easy for computers to solve. Since there are no Constraints anymore! Now let's do this for SVM Loss function.

SVM Dual Problem

firstly, for more better understanding, we will consider the lagrange's formula for n dimensions

$$\therefore \quad \underset{A,B,C}{\operatorname{argmin}} \quad \frac{\sqrt{A^2+B^2}}{2} + C \sum_{i=1}^{n} \xi_i$$

such that $y_i(Ax_{1i} + Bx_{2i} + C) \geq 1-\xi$

and $\xi_i \geq 0$

$\left\| |w| \right\| \rightarrow$ norm $= \sqrt{A^2+B^2}$

adding square for mathematical solving

$$\underset{w,b}{\operatorname{argmin}} \quad \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \xi_i \quad \text{such that} \quad y_i(w^T x + b) \geq 1-\xi_i$$

$$\therefore \quad \underset{w,b}{\operatorname{argmin}} \quad \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \xi_i$$

Such that

$$y_i(w^T x + b) \geq 1 - \xi_i$$

This is called the primal form (Hard to Solve)

Convert

Dual

let's optimise it, but since it contains constraints with inequality, we introduce KKT conditions.

Primal form is simply the original optimization problem with objective and constraints.

Constraint Optimization with inequalities

Karush Kuhn Tucker Conditions

Karush Kuhn Tucker Conditions (KKT Conditions)
They generalize the method of Lagrange multipliers to handle inequality constraints. In the context of support vector machines (SVMs) and many other optimization problems, the KKT conditions play a key role in deriving the dual problem from the primal problem.

The KKT conditions are:

1. **Stationarity:** The derivative of the Lagrangian with respect to the primal variables, the dual variables associated with inequality constraints, and the dual variables associated with equality constraints are all zero.
2. **Primal feasibility:** All the primal constraints are satisfied.
3. **Dual feasibility:** All the dual variables associated with inequality constraints are nonnegative.
4. **Complementary slackness:** The product of each dual variable and its associated inequality constraint is zero. This means that at the optimal solution, for each constraint, either the constraint is active (equality holds) and the dual variable can be nonzero, or the constraint is inactive (strict inequality holds) and the dual variable is zero.

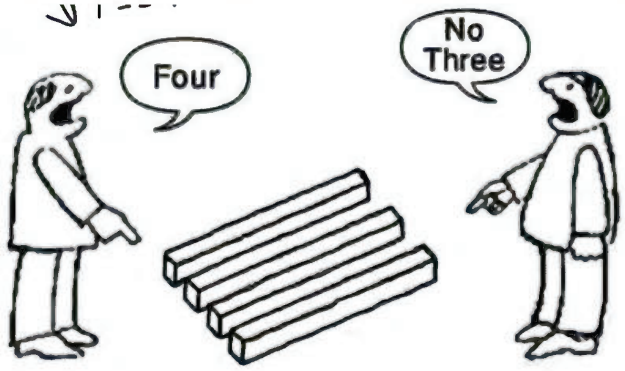
In summary, mathematically the condition states:

① $\frac{\partial L}{\partial x} = 0$, $\frac{\partial L}{\partial \lambda} = 0$ ② $x - 1 \leq 0$

③ $\lambda \geq 0$ ④ $\lambda(x-1) = 0$

All this four eq must be true for optimizing constraint with inequalities.

The duality principle is fundamental in optimization theory. It provides a powerful tool for solving difficult or complex optimization problems by transforming them into simpler or easier-to-solve problems. The solution to the dual problem provides a lower bound on the solution of the primal problem. If strong duality holds (i.e., the optimal values of the primal and dual problems are equal), then solving the dual problem can directly give the solution to the primal problem.



The primal problem is the original optimization problem that you are trying to solve. It involves finding the minimum or maximum of a particular objective function, subject to certain constraints.

The dual problem is a related optimization problem that is derived from the primal problem. It provides a lower or upper bound on the solution to the primal problem.

This simply means that even though something is the same, it might look different from different perspective. Meaning, Here in our case, the primal equation and it's dual equation (we will prove it) does not look similar at all, but is same.

We simply convert the primal equation to dual equation by derivating it to different variables (w, b) and then it gets converted into a different form.

Primal form (for hard margin SVM)

$$\underset{w,b}{\arg\min} \left[\frac{||w||^2}{2} \quad \text{such that} \quad y_i(wx_i + b) \geq 1 \; \forall i \right]$$

$\downarrow$ { α is simply λ here

$$L(w, b, \alpha) = \frac{||w||^2}{2} - \alpha_1[y_1(w^T x_1 + b) - 1)] -$$

$$\alpha_2[y_2(w^T x_2 + b) - 1] - \ldots \ldots$$

$$\ldots - \alpha_n[y_n(w^T x_n + b) - 1]$$

$$L(w, b, \alpha_i) = \frac{||w||^2}{2} - \sum_{i=1}^{n} \alpha_i[y_i(w^T x_i + b) - 1]$$

$$= \frac{||w||^2}{2} - \sum_{i=1}^{n} \alpha_i y_i w^T x_i + \alpha_i y_i b - \alpha_i$$

$$L(w, b, \alpha_i) = \frac{||w||^2}{2} - \sum_{i=1}^{n} \alpha_i y_i w^T x_i - \sum_{i=1}^{n} \alpha_i y_i b - \sum_{i=1}^{n} \alpha_i$$

Differentiation ....

$$\frac{\partial L}{\partial w} = \frac{2w}{2} - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \qquad \frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\boxed{w = \sum_{i=1}^{n} \alpha_i y_i x_i} \quad -(1) \qquad \boxed{\sum_{i=1}^{n} \alpha_i y_i = 0} \quad -(2)$$

So,

$$L(w, b, \alpha_i) = \frac{||w||^2}{2} - \sum_{i=1}^{n} \alpha_i y_i w x_i - \sum_{i=1}^{n} \alpha_i y_i b + \sum_{i=1}^{n} \alpha_i$$

Substitute value of eq (1) and eq (2) into above eq.

$$= \frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)\left(\sum_{j=1}^{n} \alpha_j y_j x_j\right) - \left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)\left(\sum_{i=j}^{n} \alpha_j y_j x_j\right) + \sum_{i=1}^{n} \alpha_i$$

Substracting

$$= -\frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)\left(\sum_{j=1}^{n} \alpha_j y_j x_j\right) + \sum_{i=1}^{n} \alpha_i$$

$$\boxed{\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{j} \alpha_i \alpha_j y_i y_j (x_i x_j)} \quad \text{— Dual form}$$

Notice that here we have to now again maximize the function instead of minimizing. Due to KKT, the role reverses.

Green highlight in page 15.

Summary :

There are mainly two major benefits of Dual form.

1. Easy to Solve

Since, there are only very few support vectors in any given data, rest of the values in the equation below will be zero except those for the support vectors.

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{j} \alpha_i \alpha_j y_i y_j (x_i x_j)$$

Here x, y is the data itself (input, output rows)

2. It is kernel friendly.

SVM Dual formulation

$$\max_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i=j}^{n} \alpha_i \alpha_j y_i y_j \boxed{(x_i x_j)}$$

We change this term to a bit different. (from dot product to (It's linear by default (cosine etc). $(x_i x_j)$)

This is where the concept of kernels came.

Instead of $x_i x_j$ dot product, we instead pass a kernel function (It is similar but bit different in few places).

kernel SVM $\longrightarrow$ $K(x_i, x_j)$

Polynomial / RBF

Polynomial Kernel

Polynomial kernel is defined as $K(x, y) = (r + \gamma x_i x_j)^d$

Here $x_i x_j$ → dot product of two vectors

r → Bias / offset term (generally $r = 0$)

d → Degree of polynomial

γ → gamma (scaling parameter - controls influence of dot product).

Steps:

1. Simply takes their dot product
2. Scaling it by γ (gamma)
3. Adding the bias term (r)
4. Raising the entire expression to the power d.

This implicitly maps the data into higher-dimensional polynomial feature space without actually computing the transformation

The trick!

This says that the kernel computes what the dot product would be in the transformed space without ever creating the transformed vectors.

Method ① : Explicit transformation
- Transform → Creates high dimensional space
- Compute → dot products in transformed space

Method ② : Implicit transformation
- Compute - dot product in original space
- Apply kernel $(\gamma(x \cdot y) + r)^d$

Therefore, Method ② clearly requires less computation.

RBF Kernel

Radial basis function

Rbf (is also called as the gaussian kernel) is a popular kernel function used in many ml algo.

$$K(x, y) = e^{-\frac{||x - y||^2}{2\sigma^2}}$$

↓ can also write as $\left(\gamma(\text{gamma}) = \frac{1}{2\sigma^2}\right)$

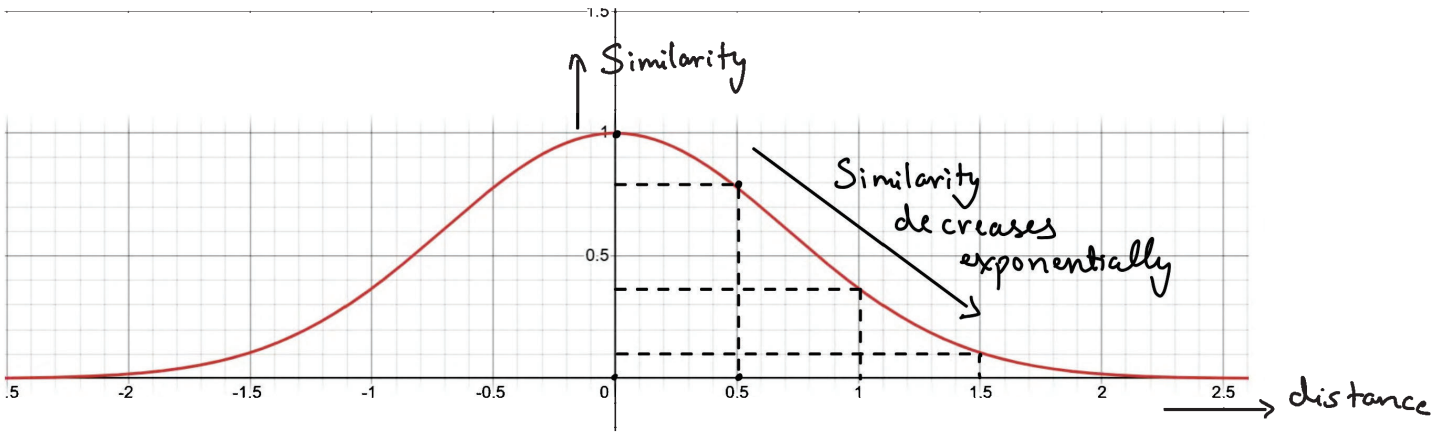
$$K(x, y) = e^{-\gamma ||x - y||^2}$$

Here,
- x, y are feature vectors
- $||x - y||^2$ is the euclidean distance between them.
- γ (gamma) is a hyperparameter that controls the "spread" or "width" of the kernel.

Local Decision Boundary :

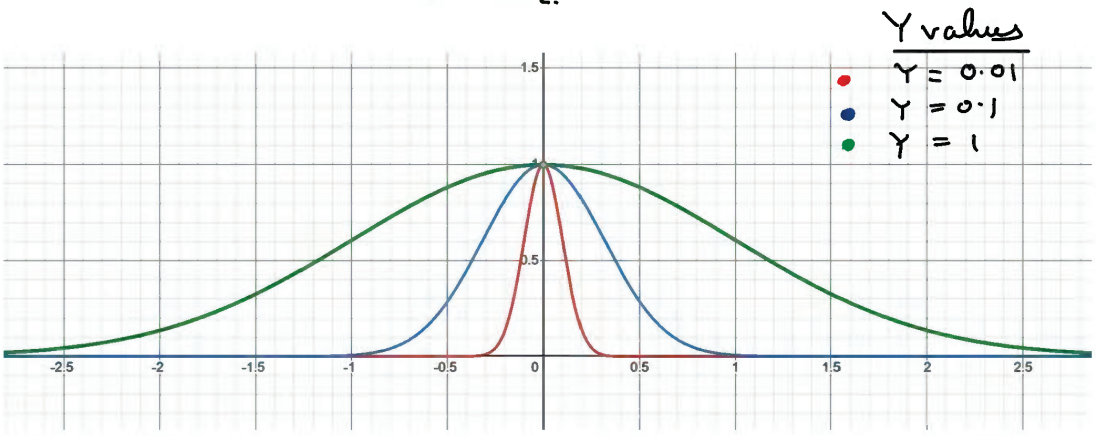
The RBF kernel measures the similarity between two data points. Its value is highest (1) when the points are identical and decreases towards zero as the points move further apart.

The gamma parameter controls how quickly this similarity decays.



If the distance is large, meaning the other point is out of the region, then there will be no similarity.
If distance is 3 here, then similarity is 0.

But we can control the region by tuning gamma.
gamma increases the width.



Therefore, by using gamma, we can control the similarity region.

Advantages

- **Non-linear Transformations:** The RBF kernel enables the use of non-linear transformations, which can map the original feature space to a higher-dimensional space where the data becomes linearly separable. This is particularly useful for problems where the decision boundary is not linear.

- **Local Decisions:** Unlike some other kernels, the RBF kernel makes "local" decisions. That is, the effect of each data point is limited to a certain region around that point. This can make the model more robust to outliers and create complex decision boundaries.

- **Flexibility:** The RBF kernel has a parameter γ (related to the standard deviation of the Gaussian distribution) that determines the complexity of the decision boundary. By tuning this parameter, we can adjust the trade-off between bias and variance, allowing for a flexible range of decision boundaries.

- **Universal Approximation Property:** The RBF kernel has a property known as the "universal approximation" property, meaning it can approximate any continuous function to a certain degree of accuracy given enough data points. This makes it highly versatile and capable of modelling a wide variety of relationships in data.

- **General-Purpose:** The RBF kernel does not make any strong assumptions about the data and can therefore be a good choice in many different situations, making it a versatile, general-purpose kernel.

Effect of Gamma

The parameter γ in the Radial Basis Function (RBF) kernel of a Support Vector Machine (SVM) is a hyperparameter that determines the spread of the kernel and therefore the decision region.

The effect of γ can be summarized as follows:

- If γ is too large, the exponential will decay very quickly, which means that each data point will only have an influence in its immediate vicinity. The result is a more complex decision boundary, which might overfit the training data.
- If γ is too small, the exponential will decay slowly, which means that each data point will have a wide range of influence. The decision boundary will therefore be smoother and more simplistic, which might underfit the training data.
- In a sense, γ in the RBF kernel plays a role similar to that of the inverse of the regularization parameter: it controls the trade-off between bias (underfitting) and variance (overfitting). High γ values can lead to high variance (overfitting) due to more flexibility in shaping the decision boundary, while low γ values can lead to high bias (underfitting) due to a more rigid, simplistic decision boundary.
- Tuning the γ parameter using cross-validation or a similar technique is typically a crucial step when training SVMs with an RBF kernel.

$\sigma \downarrow \quad \gamma \uparrow$ (local boundary region $\downarrow$) $\longrightarrow$ Overfitting

$\sigma \uparrow \quad \gamma \downarrow$ (local boundary region $\uparrow$) $\longrightarrow$ Underfitting

Relationship between RBF and Polynomial Kernel

Infinite Dimensional Mapping : The RBF kernel implicitly maps input data to an infinite dimensional feature space, which allows for even greater flexibility in forming decision boundaries.

If we calculate $e^{-\gamma \|x_i - x_j\|^2}$ or $e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

$$= e^{-\frac{\|x_i - x_j\|^2}{2}} \quad \left\{ \sigma^2 = 1 \text{ for mathematical convenience} \right.$$

$$= e^{-\frac{(x_i - x_j)^T (x_i - x_j)}{2}}$$

$$= e^{-\frac{(x_i^T - x_j^T)(x_i - x_j)}{2}}$$

$$= e^{-\frac{(x_i^T x_i - x_i^T x_j - x_j^T x_i + x_j^T x_j)}{2}}$$

$$= e^{-\left[\frac{x_i^T x_i + x_j^T x_j - x_i^T x_j - x_j^T x_i}{2}\right]}$$

both term are same

$$= e^{-\left[\frac{x_i^T x_i + x_j^T x_j - 2 x_i^T x_j}{2}\right]}$$

$$= \underbrace{e^{-\frac{1}{2}[x_i^T x_i + x_j^T x_j]}}_{\text{Constant "C"}} e^{x_i^T x_j}$$

$$= C\, e^{x_i^T x_j}$$

$$= C\, e^{1 + x_i^T x_j - 1} \quad \{ \text{add substract 1}$$

$$= C\, e^{1 + x_i^T x_j}\, e^{-1}$$

$$= C'\, e^{1 + x_i^T x_j} \quad \{ C' = C e^{-1}$$

$$\downarrow$$

$$= C' \sum_{k=0}^{\infty} \frac{C(1 + x_i^T x_j)^k}{k!} \rightarrow \text{polynomial kernels}$$

Since e^x has infinite series,

$$RBF = \sum_{i=0}^{\infty} \underbrace{K(x_i, x_j)}_{\text{poly kernel.}}$$

This is why RBF kernel is so special. It can transform any data to n-dimensional (infitity) space without actually converting the data. Not even supercomputers can do this!

More Kernels.

1. **String kernels:** These are used for classifying text or sequences, where the input data is not numerical. String kernels measure the similarity between two strings. For example, a simple string kernel might count the number of common substrings between two strings.
2. **Chi-square kernel:** This kernel is often used in computer vision problems, especially for histogram comparison. It's defined as K(x,y)=exp (−γχ2(x,y))K(x,y)=exp(−γχ2(x,y)), where χ2(x,y)χ2(x,y) is the chi-square distance between the histograms xx and yy.
3. **Intersection kernel:** This is another kernel commonly used in computer vision, which computes the intersection between two histograms (or generally non-negative feature vectors).
4. **Hellinger's kernel:** Hellinger's kernel, or Bhattacharyya kernel, is used for comparing probability distributions and is popular in image recognition tasks.
5. **Radial basis function network (RBFN) kernels:** These are similar to the standard RBF kernel, but the centers and widths of the RBFs are learned from the data, rather than being fixed a priori.
6. **Spectral kernels:** These kernels use spectral analysis techniques to compare data points. They can be particularly useful for dealing with cyclic or periodic data.

Summary

In SVM, the goal is to maximize the margin between two classes by finding the optimal hyperplane. The problem can be formulated as a primal optimization problem where we directly try to minimize a loss function that includes a margin constraint.

However, solving this primal form directly can be inefficient, especially when dealing with constraints. To transform it, we use Lagrange multipliers – a mathematical technique that helps convert optimization problems with constraints into another form, called the dual form.

In the dual form, the optimization problem becomes one of maximizing or minimizing with respect to the Lagrange multipliers rather than the original parameters (weights and biases). This new formulation depends only on the dot products between data points, which is computationally simpler. Another benefit is that this form allows us to apply kernel functions. Kernels replace the dot products with non-linear transformations, enabling SVM to work in higher dimensions where the data is more easily separable without actually having to compute those higher dimensions explicitly.

This dual form is therefore more efficient and makes SVM highly flexible by enabling non-linear boundaries, especially when using the kernel trick.