**Unlocking the ideas behind of SVM(Support Vector Machine)**

[[Sachinsoni](https://medium.com/@sachinsoni600517?source=post_page-----1db47b025376--------------------------------)](https://medium.com/@sachinsoni600517?source=post_page-----1db47b025376--------------------------------)

[Sachinsoni](https://medium.com/@sachinsoni600517?source=post_page-----1db47b025376--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F24b290d3b6e8&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40sachinsoni600517%2Funlocking-the-ideas-behind-of-svm-support-vector-machine-1db47b025376&user=Sachinsoni&userId=24b290d3b6e8&source=post_page-24b290d3b6e8----1db47b025376---------------------post_header-----------)

17 min read

·

Aug 22, 2023

53

Imagine teaching a computer to be like a clever detective, finding the best ways to group things together. That’s what Support Vector Machines (SVM) do —**they draw smart lines to separate different things**, even when they’re jumbled up. Whether it’s telling apart cats and dogs or predicting numbers, SVMs use their special skills to make sense of messy data and help computers make really good guesses. In this blog, we’ll uncover how SVMs work using simple examples and stories, so you’ll soon be a pro at understanding their secret sorting tricks!



*In this blog for better understanding, we divide SVM into 5 parts:*

*Hard Margin Classifier*

*Soft Margin SVM(SVC)*

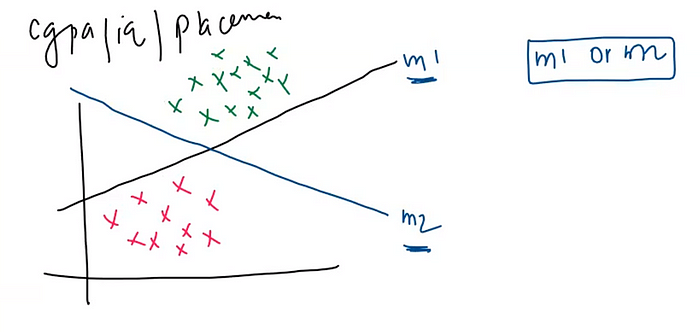
*SVC with kernels (SVM)*

*Mathematics behind Constrained Optimization Problem in SVM*

*Concept of Duality*

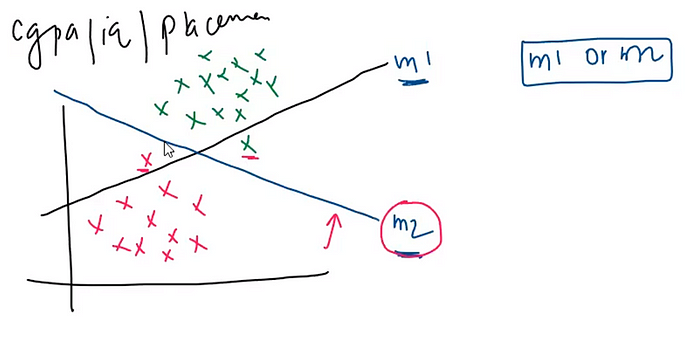
*Math's behind SVM Kernels*

**1. Hard Margin Classifier**

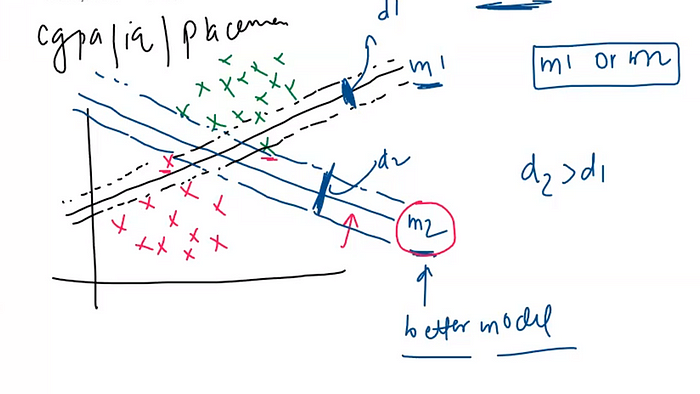


The provided graph illustrates the relationship between student CGPA and IQ, with green points representing placed students and red points indicating non-placed students. We have two prediction models, namely m1 and m2, designed to determine student placement. Now, the critical inquiry emerges: which model, m1 or m2, is the superior choice?

Considering m1, we observe that its decision boundary closely follows the data points. **This proximity suggests that if new test data points fall slightly above the m1 boundary, the model might not perform optimally.**

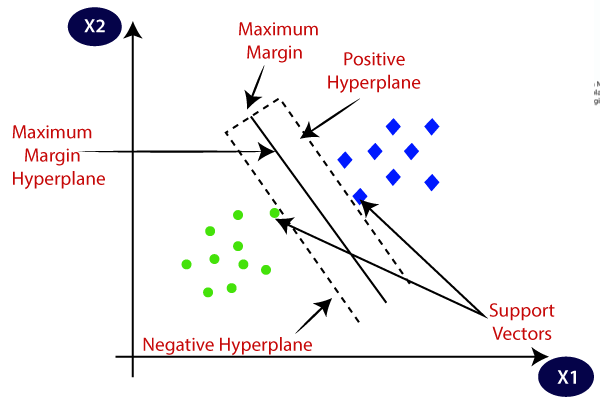


**Conversely, model m2 exhibits better performance in such cases due to its wider margin that accommodates data points from both sides.**



Because the margin of the m2 model is larger in comparison to the m1 model, it indicates that m2 holds an advantage. **The principle of maximum margin serves as the cornerstone of SVM’s operation, with larger margins being a fundamental factor in SVM’s ability to determine the better model between the two.**

In a n-dimensional plane, all the points are referred to as vectors. Among these points, the ones lying on near the positive or negative hyperplane are termed support vectors.



**These support vectors play a crucial role in determining the maximum margin, as the distance between them is pivotal in maximizing the margin.**When the support vectors are spread out, the resulting margin achieves its maximum potential.

***Mathematical formulation of Hard Margin Classifier :***

Given a set of data points xᵢ in a feature space and corresponding labels yᵢ (where yᵢ ∈ {-1, 1}), the hard margin classifier seeks to find a hyperplane represented by the equation:

w ⋅ x + b = 0

where w is the weight vector perpendicular to the hyperplane, and b is the bias term.

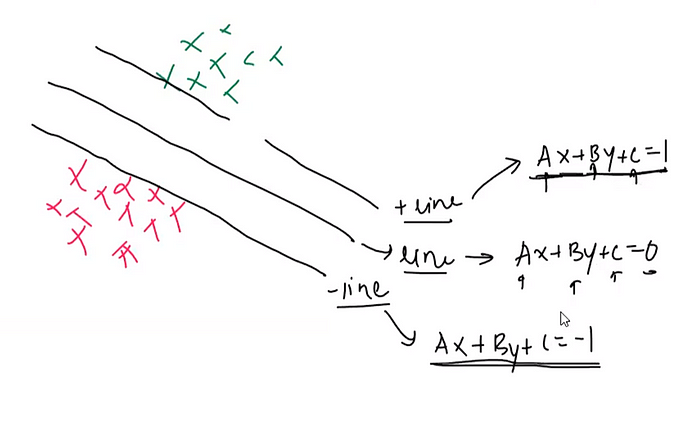
The primary objective is to find w and b that satisfy the following conditions for all data points (support vectors):

1. For data points with label yᵢ = 1, the condition is: w ⋅ xᵢ + b ≥ 1.
2. For data points with label yᵢ = -1, the condition is: w ⋅ xᵢ + b ≤ -1.

These conditions ensure that the hyperplane creates a margin of at least 1 unit between the two classes, thus achieving perfect separation. The goal is to maximize this margin while still satisfying the conditions. The margin is the perpendicular distance between the hyperplane and the nearest support vectors from each class.

Let us understand these mathematical concept by taking 2-dimensional example.

Consider the primary classifying line with the equation Ax + By + C = 0. Furthermore, we have the positive support line represented by Ax + By + C = 1, and the negative support line given by Ax + By + C = -1.



The reason for choosing these specific equations is to create a margin of width 2 units (1 unit on each side) between the positive and negative support lines, while ensuring that the data points are correctly classified. These equations are formulated in this manner to facilitate the optimization process and maximize the separation between classes while minimizing classification errors. The optimization process finds the values of A, B, and C that create the widest possible margin while still correctly classifying the data points.

***Why are the equations ax + by + c = 1 and ax + by + c = -1 chosen for defining the support lines in Support Vector Machines (SVM)? And why do we not opt for equations like ax + by + c = 2 and ax + by + c = -5 instead?***

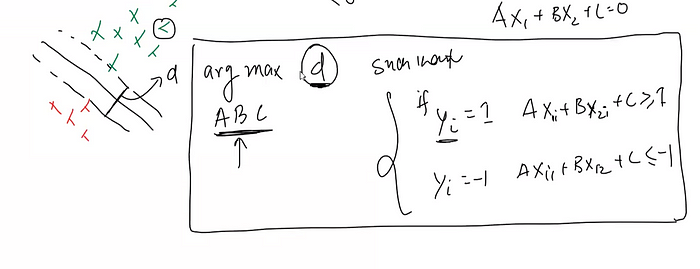
The answer is very simple for above question, When we set the lines at ax + by + c = 1 and ax + by + c = -1, it makes sure that the margin between the lines is exactly 2 units wide. This choice helps in making calculations easier and the process more balanced.

If we were to use ax + by + c = 2 for one line and ax + by + c = -5 for the other, the margin would become 7 units wide. While this seems bigger, too much margin can sometimes cause problems.

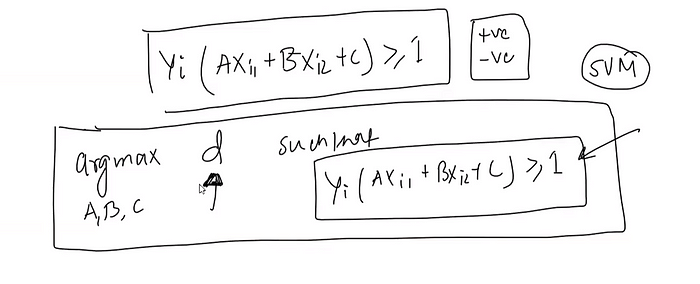
Having a balanced margin, like 2 units, helps the SVM perform well on both the data it was trained on and new data it hasn’t seen before. It’s like finding a good middle ground between being too strict and too lenient with the lines.

So, the choice of numbers in the equations is about finding a good balance that makes the SVM work effectively on various types of data.

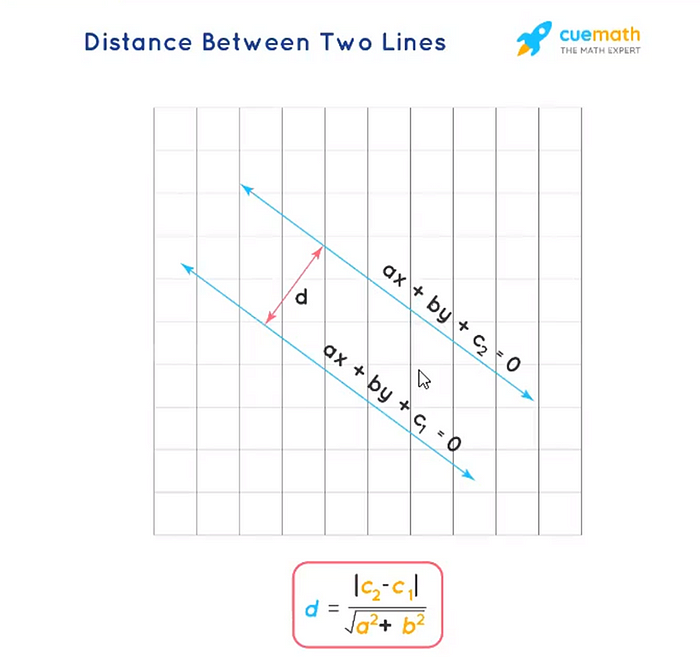
**In summary, our objective is to determine an equation ax + by + c = 0 that ensures the red points remain on the correct side of the negative line, and the green points stay on the appropriate side of the positive line. Simultaneously, we aim to maximize the distance between these lines, creating a wide margin. The mathematical condition for this is given below:-**



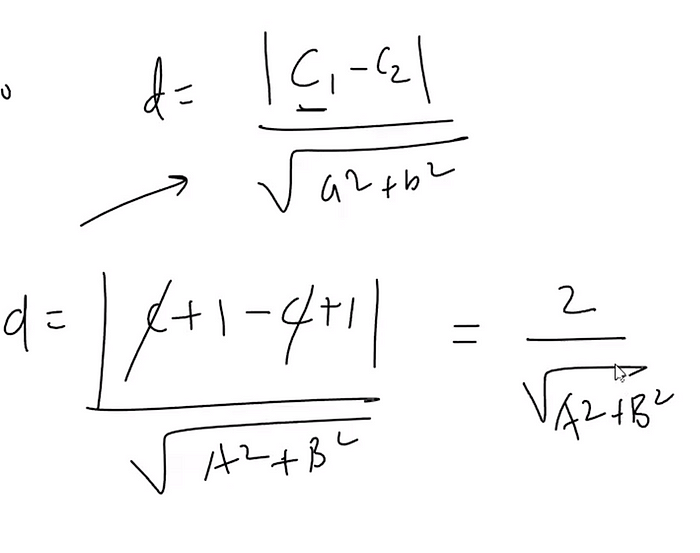
Now converting this two different condition in one single condition:-



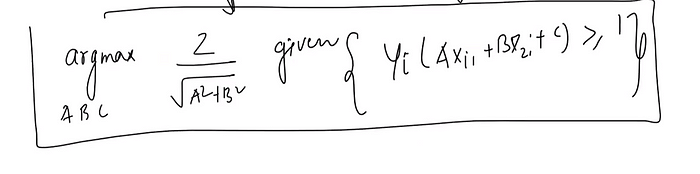
Now we need to calculate the distance between two parallel lines Ax+By+C=1 and Ax+By+C=-1.



and distance, d is calculated below:



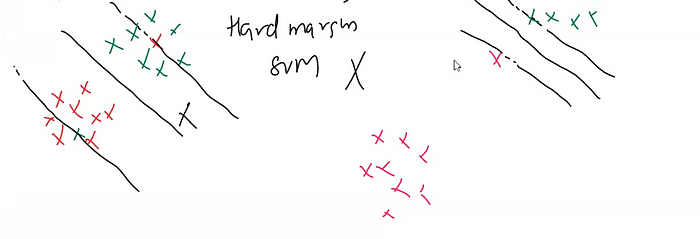
and final mathematical condition is:



This is the **constrained optimization problem**and solve with the help of **Quadratic Programming.**

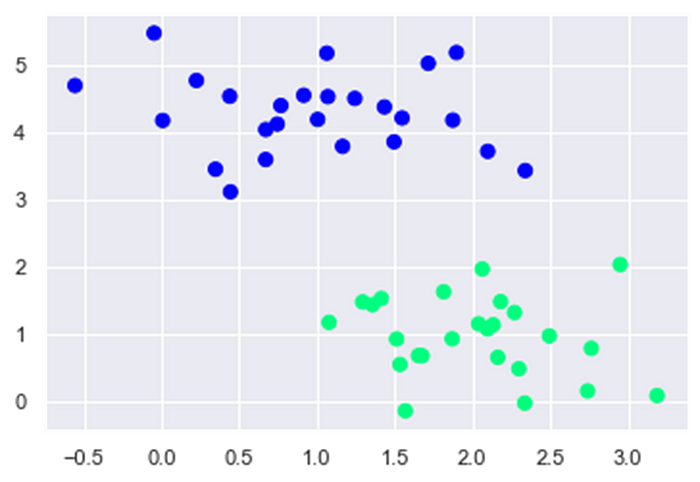
**Problems with Hard Margin Classifier:**

The primary issue lies in its that a single mispositioned red or green point can significantly impact the decision boundary and the margin. (See the below diagram)



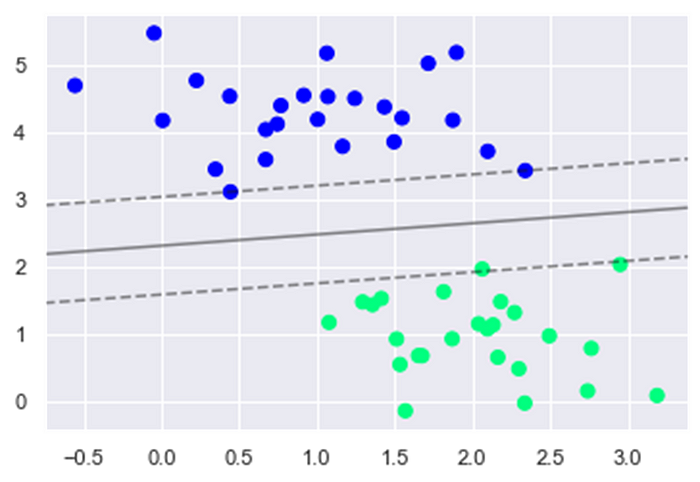
*How to implement Hard Margin Classifier in scikit-learn ?*

We have linear dataset given below:



from sklearn.svm import SVC # "Support vector classifier"  
model = SVC(kernel='linear', C=1)  
# By setting kernel= linear and C=1, we use hard margin classifier  
model.fit(X, y)

After apply Hard Margin Classifier, now decision boundary looks like:



[***Click here to see the code details***](https://github.com/sachin0612/SVM.git)***.***

In conclusion, the hard margin classifier, while aiming for perfect separation between classes, often encounters challenges in real-world datasets due to its sensitivity to outliers and noise. Its rigidity can lead to overfitting and difficulty in handling cases where data points are not linearly separable. As a result, while the hard margin classifier offers a clear theoretical foundation, practical applications demand more adaptive and flexible approaches such as the soft margin classifier and kernel methods to achieve better generalization and robust classification performance.

**2. Soft Margin SVM**

The hard margin classifier aims for a perfect separation, making it extremely sensitive to outliers or noisy data points. If even a single point is mispositioned, it can dramatically alter the decision boundary and the margin. Slack variables provide a solution by allowing some data points to be on the wrong side of the margin or even within it. By permitting a controlled amount of misclassification through slack variables, the SVM becomes more robust to outliers.

***Concept of Slack Variable:***

The concept of slack variables was introduced by Vladimir Vapnik in 1995 and is used in the formulation of the “soft-margin” SVM to handle cases where data is not linearly separable, or when one allows for some degree of error in classification.  
Mathematically, for each data point i, a slack variable ξi ≥ 0 is introduced. The slack variable ξi measures the degree of misclassification of the data point xi.  
**• ξi = 0 if xi is on the correct side of the margin.  
• 0 < ξi < 1 if xi is on the correct side of the hyperplane but on the wrong side of the margin.  
• ξi ≥ 1 if xi is on the wrong side of the hyperplane, i.e., it is misclassified.**

Formula for slack variable is **ξᵢ = max(0, 1 — w ⋅ xᵢ — b)**and in our 2D case **ξᵢ = max(0, 1 — yᵢ (Ax1ᵢ+Bx2ᵢ+C))**

Let’s consider a 2D example where we have a simple decision boundary equation ax + by + c = 0. We’ll calculate the slack variable for a data point in the context of a soft margin SVM.

Assuming our decision boundary equation is: 2x + 3y — 6 = 0

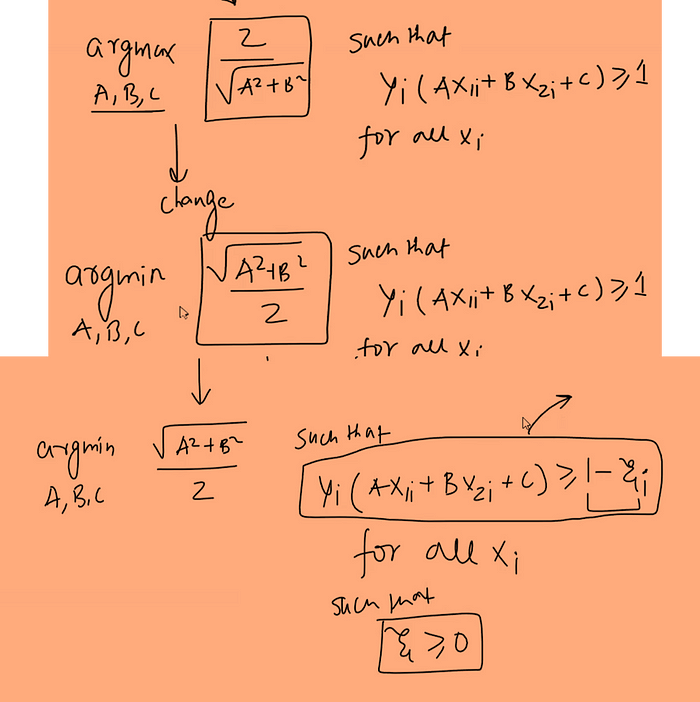
And we have the following data point: Data point: (4, 2) Label: y = 1 (positive class)

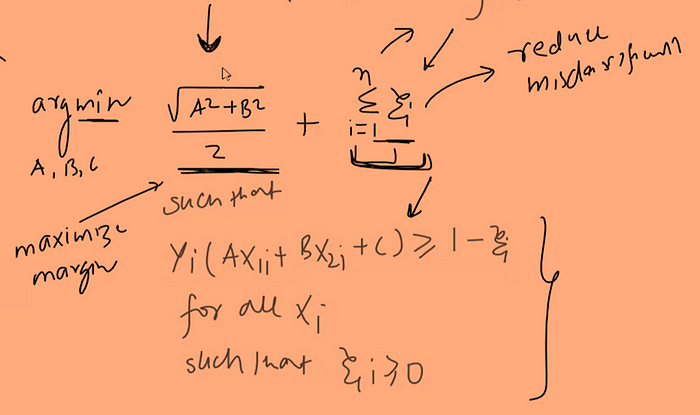
Now, let’s calculate the slack variable ξ for this data point. The formula for ξ when y = 1 (positive class) is ξ = max(0, 1 — (ax + by + c)).

Substitute the given values: ξ = max(0, 1 — (2 \* 4 + 3 \* 2–6)) ξ = max(0, 1 — (8 + 6–6)) ξ = max(0, 1–8)

Since the value of ξ should be at least 0, in this case, ξ = max(0, -7) = 0, indicating that the data point is correctly classified and lies on the correct side of the margin.

***Changes in Mathematical Formulation of Hard Margin Classifier:***

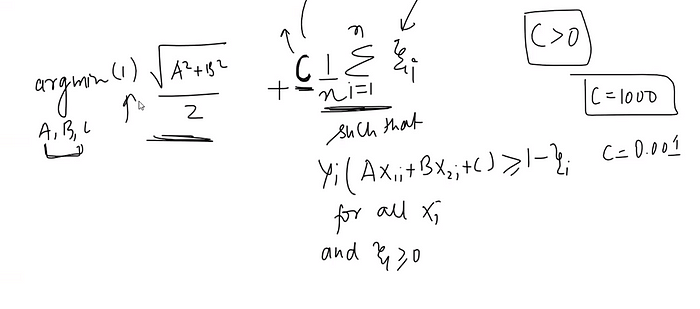




The above formula represents the optimization problem for the soft margin Support Vector Machine (SVM).

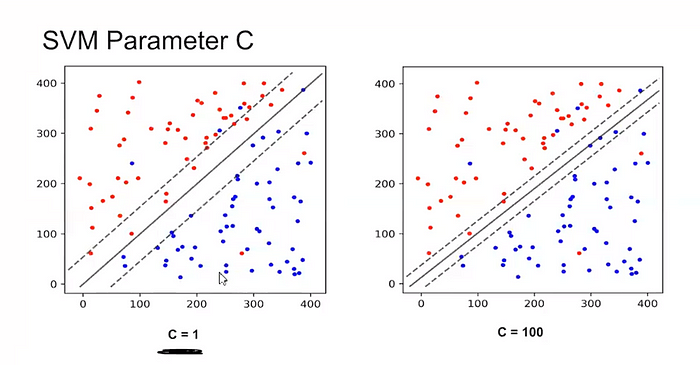
Σ ξᵢ: This part accounts for the slack variables ξᵢ associated with each data point. Slack variables allow some data points to be on the wrong side of the margin or even within it. By minimizing the sum of these slack variables, the optimization process encourages a balance between maximizing the margin and minimizing classification errors.

***Introduction of C :***



C is hyperparameter and used in the formulation of the optimization problem for soft margin SVM.

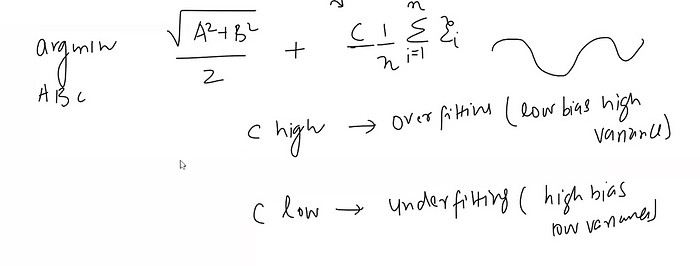
***Comparison between high and low value of C :***



Here’s how C works:

1. **Smaller C:** When C is small, the SVM places a higher priority on achieving a wide margin, even if that means allowing more misclassifications. In this case, the SVM is more tolerant of misclassified points and focuses on finding a larger margin.
2. **Larger C:** When C is large, the SVM becomes more sensitive to misclassifications and tries to minimize them as much as possible. This can lead to a narrower margin in order to correctly classify more points.

***Bias- Variance Trade Off with C :***



In a sense, C acts as a control knob that adjusts the balance between margin maximization and error minimization. It’s important to choose an appropriate value for C based on the specific problem and dataset at hand. A smaller C might be preferred when there is significant noise or outliers in the data, whereas a larger C might be suitable for cleaner datasets where misclassifications are less acceptable.

In summary, the introduction of the soft margin Support Vector Machine (SVM) addresses the limitations of its hard margin counterpart and offers a more versatile solution for complex real-world datasets. By allowing for controlled misclassifications through the use of slack variables and the regularization parameter C, the soft margin SVM strikes a balance between maximizing the margin width and minimizing classification errors. This flexibility enables the model to handle outliers, noise, and linearly inseparable data more effectively. The parameter C empowers users to tailor the SVM’s behavior according to the dataset’s characteristics, ensuring robust generalization and improved performance on diverse data.

**3. SVC with Kernels (SVM)**

The soft margin SVM, like its hard margin counterpart, works best in scenarios where linear separation is possible. When dealing with complex non-linear relationships and limited feature space, even the soft margin SVM might struggle without additional techniques like kernel methods. So for dealing with non-linear data, we need to learn soft margin SVM with kernel concept.

***Introduction of Kernels :***

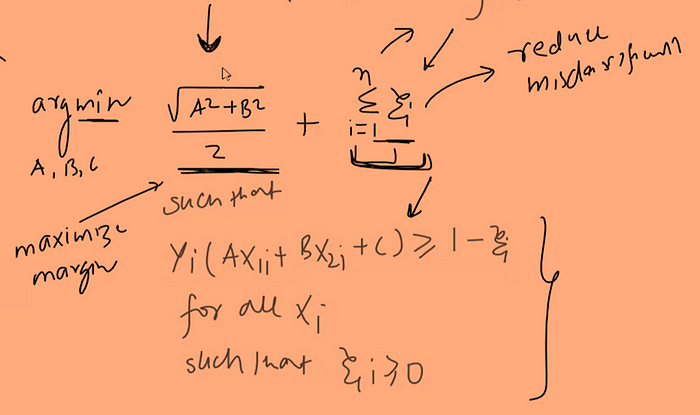
Kernels provide a way to implicitly transform data into a higher-dimensional space without actually computing the transformations explicitly. This higher-dimensional representation enables the algorithm to find complex relationships and patterns that might not be apparent in the original feature space.

Kernels allow SVMs (and other algorithms) to work effectively in scenarios where the data is not linearly separable in the original feature space. By applying a kernel function, the data is effectively “lifted” into a higher-dimensional space where it might become linearly separable. This process is called the “kernel trick.”

Commonly used kernel functions include the linear kernel (which doesn’t actually transform the data), as well as non-linear kernels like the polynomial kernel, Gaussian (RBF) kernel, and sigmoid kernel. Each kernel has its own way of mapping the data to a higher-dimensional space, allowing the SVM to capture complex relationships.

**Mathematics behind Constrained Optimization Problem in SVM:**

We have a following constrained optimization problem in 2D and find the value of A,B and C such that following conditions are valid.



Let’s consider a simple example of optimizing a function subject to a constraint using **Lagrange multipliers**.

**Problem: You want to maximize the function f(x, y) = x + y, but the constraint is x² + y² = 1, which means your solution must lie on the unit circle.**

1. **Formulating the Lagrangian:** The Lagrangian combines the objective function and the constraint using a Lagrange multiplier (λ):

**L(x, y, λ) = f(x, y) -λ(g(x, y) — 1)** where g(x, y) represents the constraint equation (x² + y²).

**2. Setting up Equations:** Now, we have two equations:

* ∂L/∂x = ∂(x + y -λ(x² + y² — 1))/∂x = 1 -2λx = 0
* ∂L/∂y = ∂(x + y -λ(x² + y² — 1))/∂y = 1 -2λy = 0
* ∂L/∂λ = g(x, y) — 1 = x² + y² — 1 = 0 (constraint equation)

**3. Solving for Variables:** Solving the equations for x, y, and λ: From the first equation: x = 1/2λ From the second equation: y = 1/2λ Plugging x and y into the constraint equation: (1/2λ)² + (1/2λ)² = 1 ⇒ λ² = 1/2

4. Finding λ: Solving for λ: λ = ±1/√2

5. Finding x and y: Plugging λ = 1/√2 into x and y equations: x = √2 and y = √2

6. Finding f(x, y): Plugging x and y into the objective function: f(x, y) = (√2) + (√2) = 2√2

So, the maximum value of the function x + y under the constraint x² + y² = 1 is 2√2, and the solution lies at x = √2, y = √2 on the unit circle.

In this example, the Lagrange multiplier λ helped incorporate the constraint x² + y² = 1 into the optimization process, enabling us to find the maximum value of the function within the constraint.

The example I provided above was for an equality constraint problem.

Now, I am taking an inequality constraint example for better understanding SVM. But Before solving this we need to learn KKT theorem.

**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.

**Let’s work through an example using the function f(x, y) = x² + y² with the inequality constraint x + y — 1 ≤ 0.**

**Problem: You want to minimize the function f(x, y) = x² + y² subject to the constraint:**

* **x + y — 1 ≤ 0 (Inequality Constraint)**

1. **Formulating the Lagrangian:** The Lagrangian combines the objective function and the constraint using a Lagrange multiplier (λ):

**L(x, y, λ) = f(x, y) -λ(g(x, y) — 1)** where g(x, y) represents the constraint equation (x + y — 1).

**2. Setting up Equations:** The equations are:

* ∂L/∂x = ∂(x² + y² -λ(x + y — 1))/∂x = 2x -λ = 0
* ∂L/∂y = ∂(x² + y² -λ(x + y — 1))/∂y = 2y -λ = 0
* ∂L/∂λ = g(x, y) — 1 = x + y — 1 = 0 (constraint equation)

**3. Solving for Variables:** Solving the equations for x, y, and λ: From the first equation: 2x -λ = 0 ⇒ x = λ/2 From the second equation: 2y -λ = 0 ⇒ y = λ/2 Plugging x and y into the constraint equation: (λ/2) + (λ/2) = 1 ⇒ λ = 1  
and x= 0.5 and y=0.5

**4. Dual Feasibility:** The dual variable associated with the inequality constraint (λ) is non-negative, which follows the nonnegativity condition.

**5. Complementary Slackness:** The product of the dual variable and its associated inequality constraint is zero:

* λ \* (x + y — 1) = 2 \* (x + y — 1) = 0

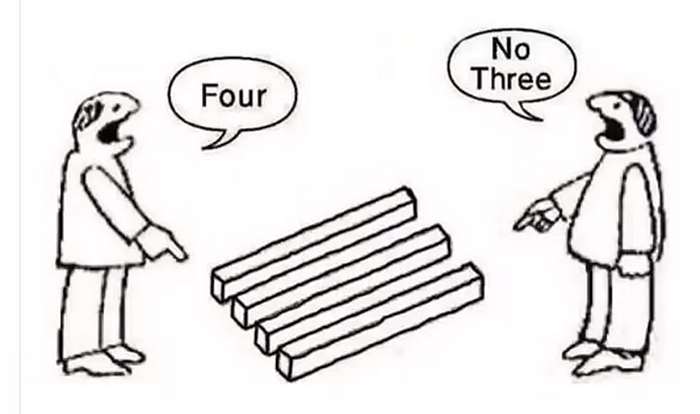
**6. Solving the KKT Conditions:** We observe that the solution satisfy the KKT conditions.

Now the minimum value of function =(0.5)² +(0.5)² = 0.50

So, in this example, the Lagrange multiplier and KKT conditions helped us find the correct optimal solution for the optimization problem.

***Concept of Duality :***

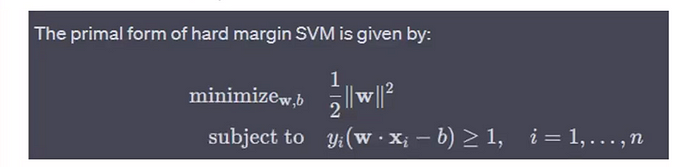
The duality principle is fundamental in optimization theory. It provides a powerful tool for solving difficult or complex optimization problems by transforming them into simpler or easier-to-solve problems. The solution to the dual problem provides a lower bound on the solution of the primal problem. 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.



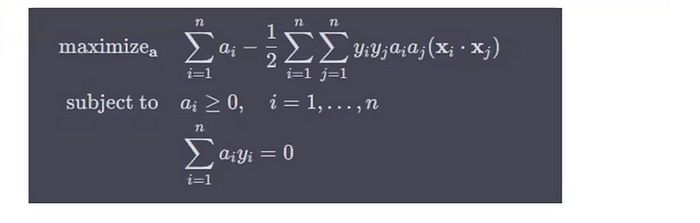
The primal problem is the original optimization problem that you are trying to solve. It involves finding the minimum or maximum of a particular objective function, subject to certain constraints.

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.

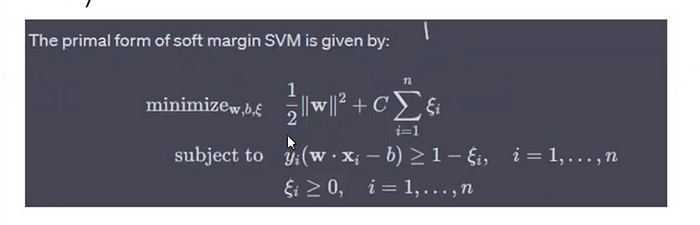
It is easier to solve SVM dual form as compared to SVM primal form, so we need to convert SVM primal form into SVM Dual form and solutions of SVM Dual form is also the solution of SVM primal form.



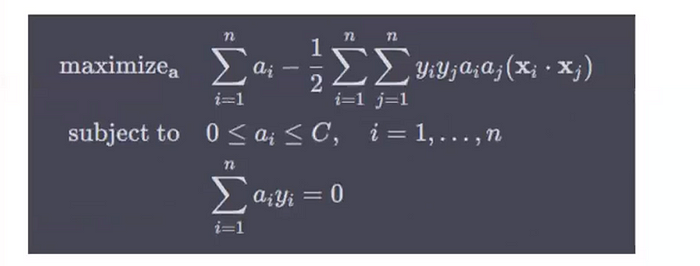
and its dual form is given by:



Here, *n* is the number of training samples, *α* is the vector of Lagrange multipliers (dual variables), *yi*​ is the label of the *i*-th sample, *xi*​ is the *i*-th input vector

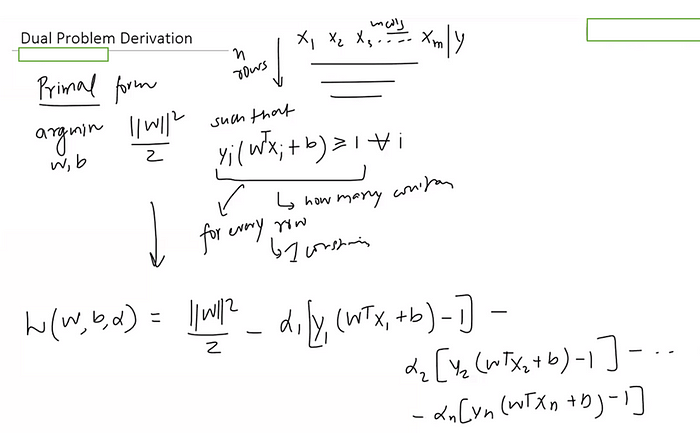


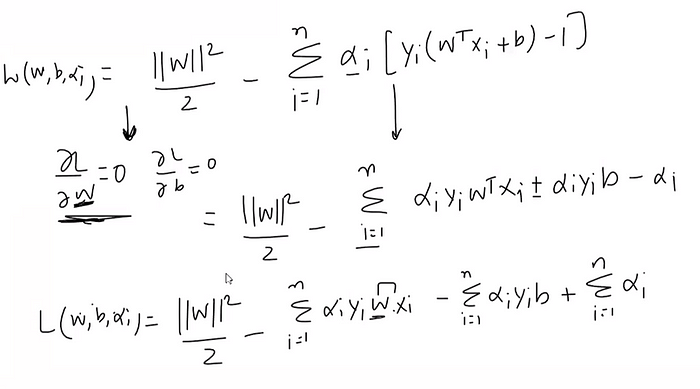
and its dual form is given by:

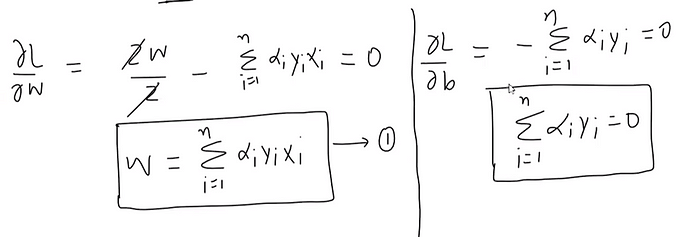


Here, similar to the hard margin case, *n* is the number of training samples, *α* is the vector of Lagrange multipliers (dual variables), *yi*​ is the label of the *i*-th sample, *xi*​ is the *i*-th input vector, and *C* is a constant representing the upper bound on the Lagrange multipliers.

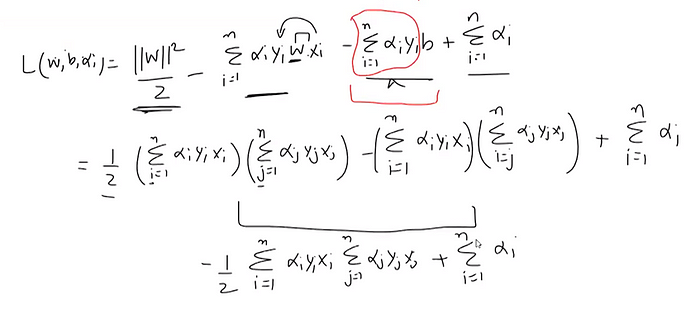
***Dual Problem Derivation for Hard Margin Classifier :***

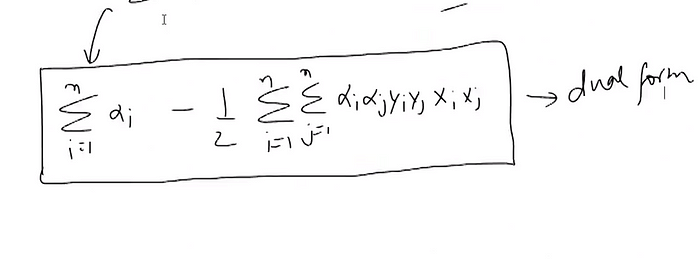






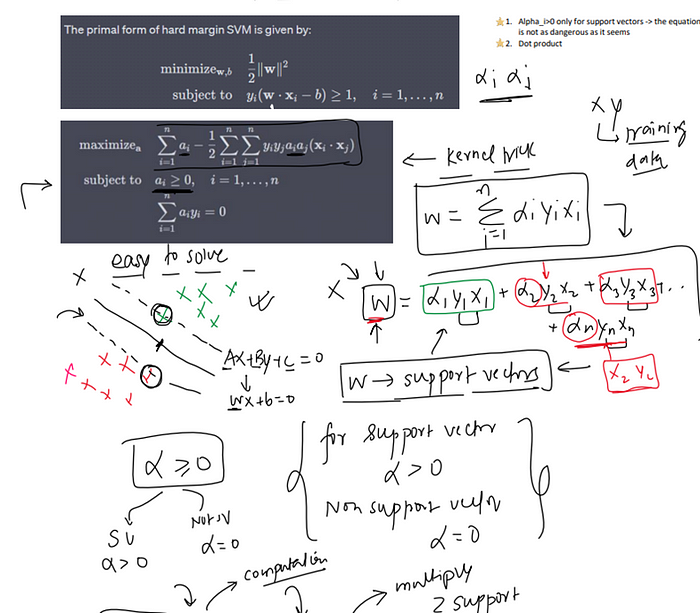
Now putting w value in below equation:



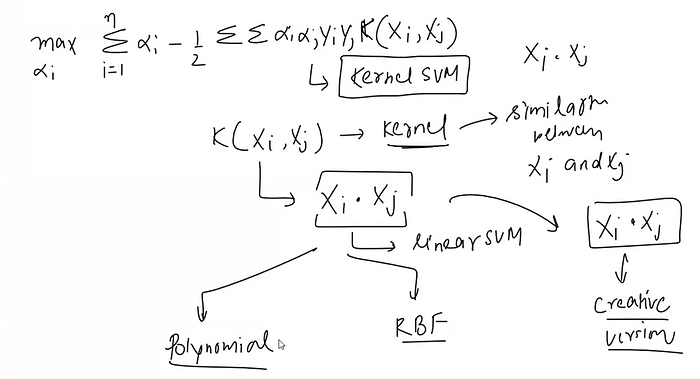


The above formula is the proof of Dual form of hard margin classifier.

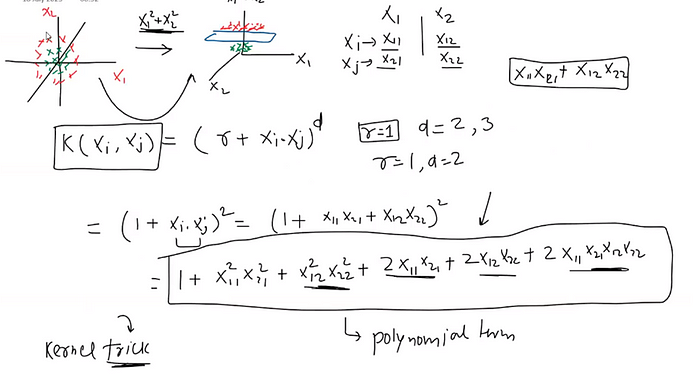
The dual form is easy to solve because we solve this equation for only support vector and for non support vector **α**is zero.



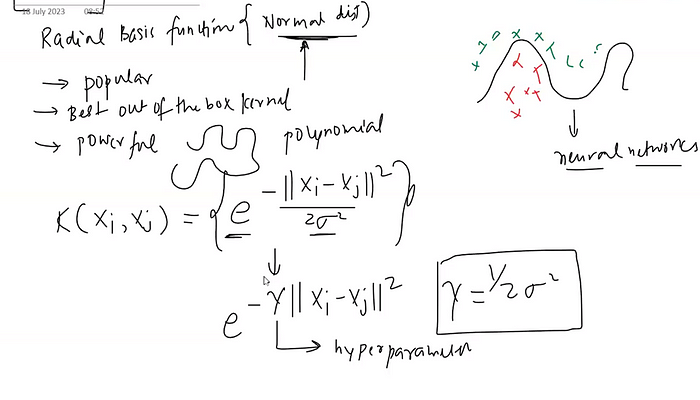
***Kernel SVM***



***a. Polynomial Kernel :***



***b. RBF Kernel :***



**Properties of rbf function :**

* **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.
* **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.

In conclusion, Support Vector Machines (SVM) stand as powerful tools in machine learning, offering effective solutions for both classification and regression tasks. From the hard margin classifier’s pursuit of clear separations to the soft margin’s flexibility in handling noisy data, SVMs provide versatile approaches to pattern recognition. Incorporating kernels allows for complex transformations without the computational burden, enabling the capture of intricate relationships in data. Through understanding the Lagrange multiplier and KKT conditions, we unveil the inner workings of SVM optimization, bridging theoretical foundations with practical application. My sincerest thanks for joining me on this exploration of SVMs, and I hope this journey enriches your understanding of this captivating machine learning technique.