Support Vector Machines, Dual Formulation, Quadratic Programming & Sequential Minimal Optimization

This is a math-oriented approach to the intuition behind SVMs and the optimization algorithms used to solve it. This article serves as a one-stop guide to demystify the working of SVMs internally.

Suraj Donthi Feb 10 · 12 min read

The Support-vector Machine (or called Support-vector Networks initially by the author — Vladimir Vapnik) takes a completely different approach to solving statistical problems (in specific Classification). This algorithm has been heavily used in several classification problems like Image Classification, Bag-of-Words Classifier, OCR, Cancer prediction, and many more. SVM is basically a binary classifier, although it can be modified for multi-class classification as well as regression. Unlike logistic regression and other neural network models, SVMs try to maximize the separation between two classes of points. A brilliant idea is used by the author.

Below are the *concepts we'll cover in this article*, that basically demystify SVMs step by step and then enhance the algorithm against its deficiencies.

Vanilla(Plain) SVM & its Objective Function

Let’s just take the formal definition of SVM from Wikipedia:

> A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.

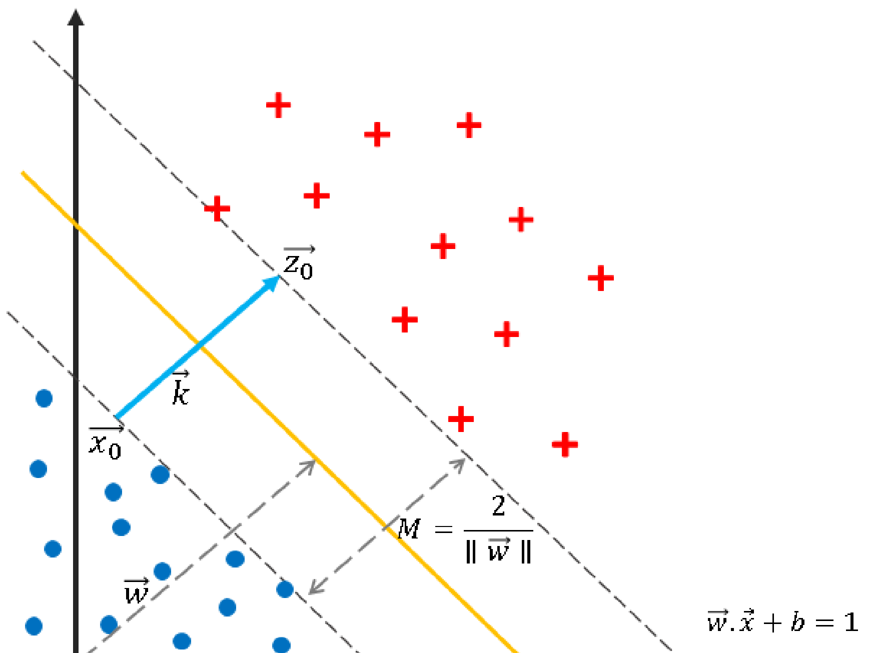
Wait, but there are too many technical terms! Let’s just simplify it & keep information that's only required!

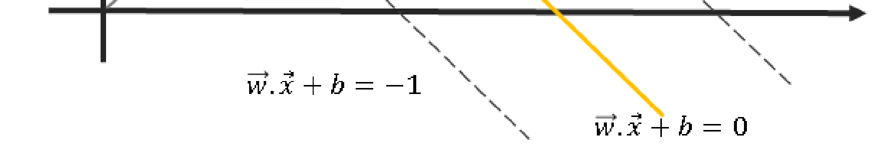
First things first, the SVM creates a hyperplane (*a simple line in n-dimensions*). As in the below GIF, this hyperplane needs to bisect the two classes in the best way possible. This is all that SVMs do… The rest are toppings on it! (*infinite-dimensional space, regression, outlier detection, etc.*)

Now to construct the *OPTIMAL HYPERPLANE* it takes the support of two other hyperplanes that are parallel & equidistant from it on either side!

These two support hyperplanes lie on the most extreme points between the classes and are called **support-vectors**.

Therefore, what we just need to do is to find the support hyperplanes (*using for simplicity*), that have the maximum distance between them! From this, we can easily get the *OPTIMAL HYPERPLANE*. This is simply called the **Maximum Margin Hyperplane**. The distance between the support hyperplanes is called the **Margin**.





Source: Image by Author

Hence, our goal is to simply find the Maximum Margin M. Using vector operations, we can find that (given the *OPTIMAL HYPERPLANE (**w.x+b=0**)*), the **Margin** is equal to:

$$M = \frac{2}{\|\vec{w}\|}$$

Therefore, the **Maximum Margin M**:

$$\implies \max(M) = \max\left(\frac{2}{\|\vec{w}\|}\right) = \min(\|\vec{w}\|)$$

For mathematical convenience, the **Objective Function** becomes (a detailed explanation in Appendix 1):

$$\min\left(\frac{1}{2}\|\vec{w}\|^2\right)$$

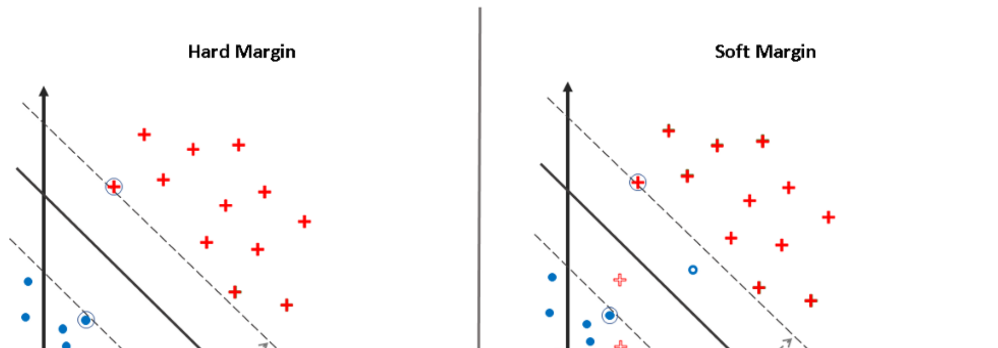
Observe here that our goal has been deduced to finding the optimal ***w*** from the equation ***w.x+b=0***!

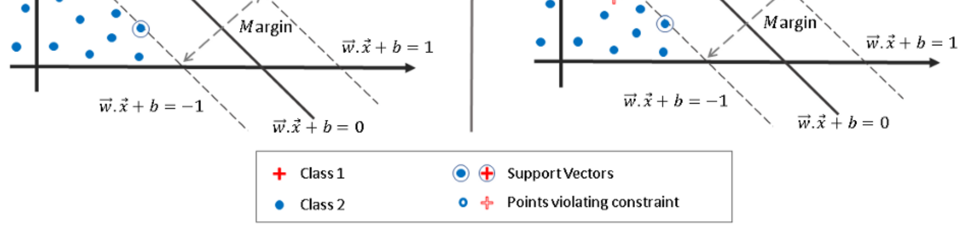
However, this also comes with a constraint that the points in a class must not lie within the two support hyperplanes! That can be mathematically represented as:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \forall \text{ Points } i = 1, \ldots, \text{to} \ldots, n$$

So what do these constraints mean?

It means that there can be no points inside of the hyperplane as shown in the below figure and this is called ***Hard Margin SVM (Vanilla SVM).***





Source: Image by Author

This is the big drawback of SVMs! The two classes need to be fully separable. This is never the case in real-world datasets! This is where ***Soft Margin SVMs*** come into play🎉😃!

Before we move on, just one question! ***Do SVMs use Gradient Descent to find the minima***? No way! (Most people fail to know this). The minima is directly found by solving derivatives of the *Objective function*. Since there are constraints, we'll need to first take the *Langrangian of the Objective Function* to solve for the minima. You can find the complete derivation in Appendix 2 (I encourage you to read it!).

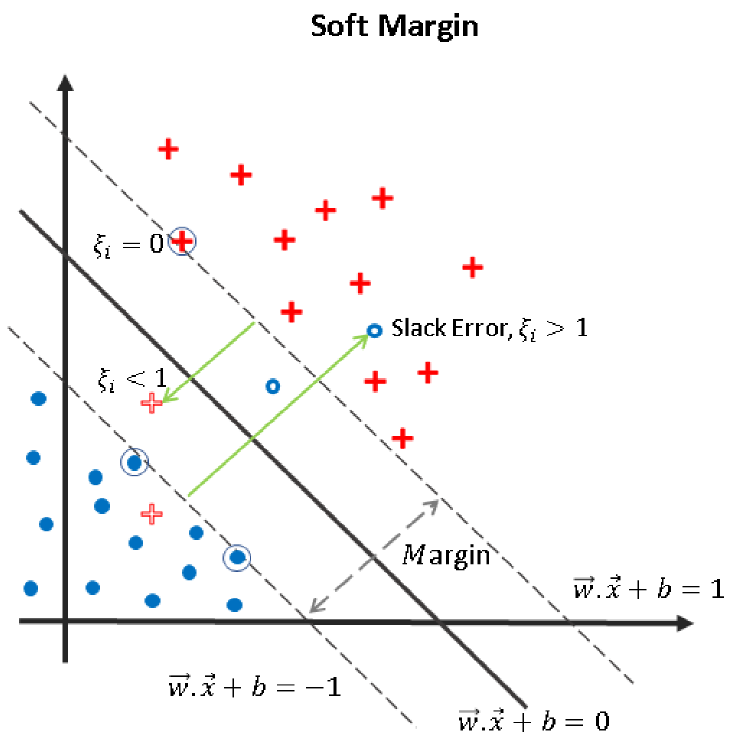
. . .

Soft Margin SVM

As mentioned above, Soft Margin SVMs can handle classes with inseparable datapoints. The figure below explains the same clearly!

So here's the gist of the idea behind Soft Margin:

> To allow the SVM to make some mistakes and yet keep the margin as wide as possible.



Source: Image by Author

Now how is it able to do it, but not Vanilla SVMs?🤔

We just use an additional penalizing factor in the Objective function called ξ_i. This factor is the distance exceeded from the respective support hyperplane by a data point towards the other class.

Therefore, if the data point is well within the boundary (support hyperplane), the penalizing factor ξ_i is 0. Else if the data point is on the other side, this factor ξ_i is equal to its distance between the datapoint and the support hyperplane. Hence, the value ξ_i is a non-negative number.

This can be summarized with the below equation:

$$\begin{array}{rcl} y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, & i = 1,\ldots,n & \& \\ \xi_i \geq 0, & i = 1,\ldots,n & \end{array}$$

And it should ideally represent the number of "*mistakes*" the SVM makes. Therefore, our new objective function would be:

$$\min\left(\frac{1}{2}w^2 + C(\#\text{ of mistakes})\right)$$

$$\implies \min\left(\frac{1}{2}w^2 + C.\,F\left(\sum_{i=1}^{n}\xi_i^{\sigma}\right)\right)$$

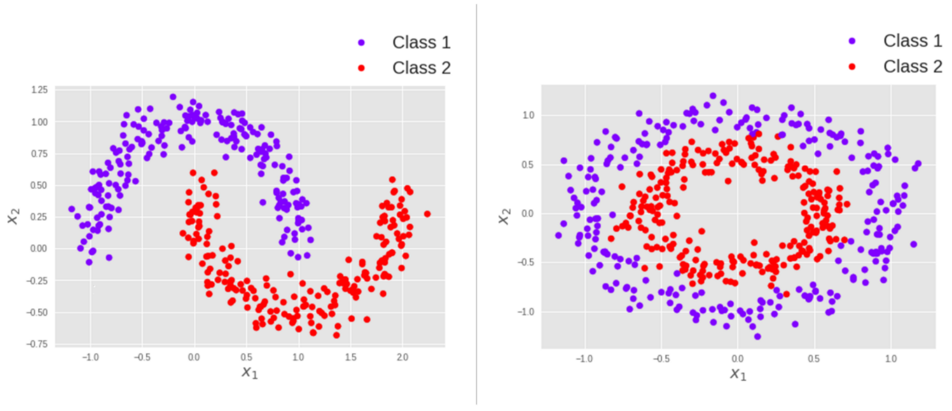
Here for a small power σ, the function ***F*** becomes the number of mistakes! (The summation is used to account for ξ_i of all points in the dataset)

Just like for SVMs, the minima can be found directly by solving its derivatives (Lagrangian of the derivatives).

. . .

The Kernel Trick

Now looking back at what we've derived, it is clear that we are only using ***w.x+b***. This is simply only a linear equation. That means SVM works best when you can classify the data linearly!

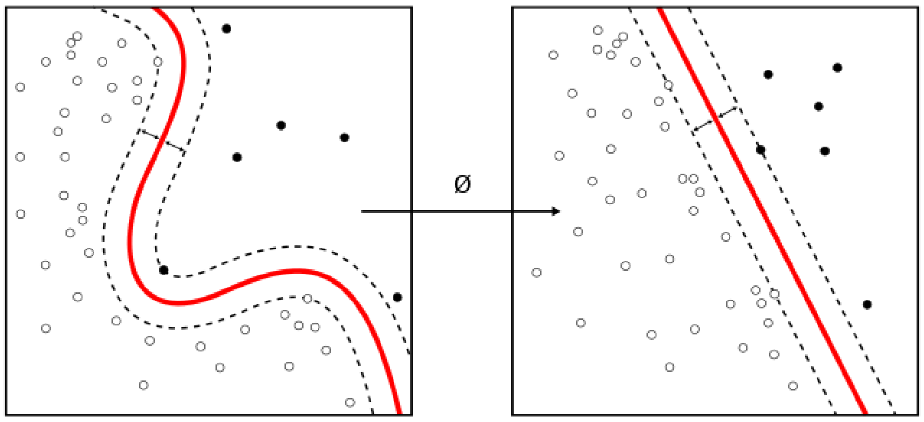


Non-linear data. Source: Image by Author

That is another really huge limitation! However, the authors have found a hack for this 💃 !! & that's the kernel trick. In simplistic terms:

> The Kernel simply converts the non-linear datapoints to linear datapoints, so that the SVM can bisect two classes.

The same is shown in the below figure.



Conversion from n-dimension to N-dimension. Source: Wikimedia

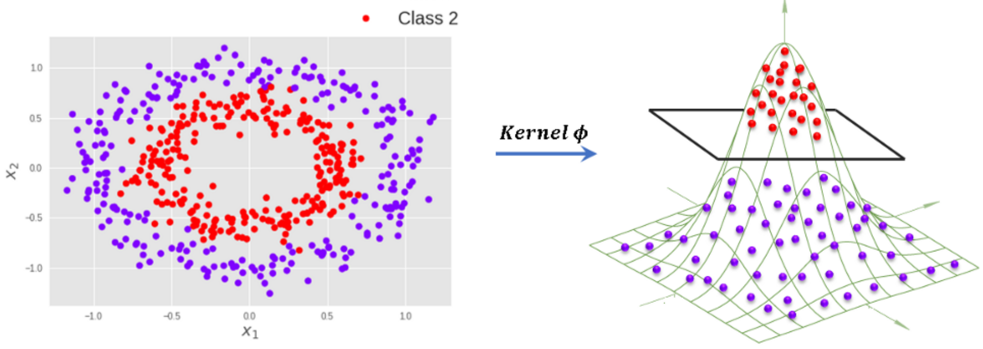
Hence the new line equation would be

$$w.\phi(x) + b$$

Here, a neat trick is used to convert the points in *n-dimension* to *N-dimension* where *N>n*.

The Kernel ***φ*** must obey the following conditions:

$$\begin{aligned} \phi(x) &= 1 \qquad , if\ \|x\| \leq 1 \\ \phi(x) &= 0 \qquad , otherwise \end{aligned}$$



Source: Image by Author

There are several kernel functions used for SVMs. Some of the popular ones are:

- **Gaussian Radial Basis Function (RBF)**:

$$\phi(x, y) = exp(-\gamma \|x_i - x_j\|^2)$$

where $\gamma > 0$.

A special case is $\gamma = 1/2\sigma^2$

- **Gaussian Kernel**:

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

- **Polynomial Kernel**:

$$k(x_i, x_j) = (x_i . x_j + 1)^d$$

- **Sigmoid kernel**:

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

. . .

Appendix 1 — Deriving the Maximum Margin Eq. & the Objective function

For deriving the Objective function, we assume that the dataset is linearly separable.

We know that the two support hyperplanes lie on the Support Vectors and there are no points between them. Hence, we can first consider the midhyperplane between these two support hyperplanes to be:

$$w.x + b = 0$$

And considering that the distance between this and the support hyperplanes is 1, we get

$$w.x + b = 1$$

$$\&$$

$$w.x + b = -1$$

Since there are no points exceeding these two hyperplanes we can deduce that:

For all values with label ***y_i=+1,***

$$w.x_i + b \geq 1$$

& for all values with label ***y_i=-1,***

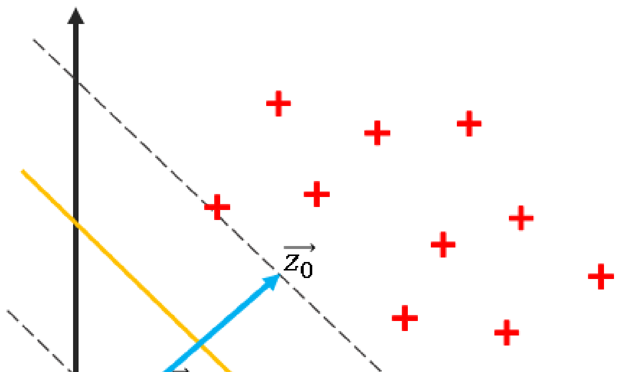
$$w.x_i + b \leq -1$$

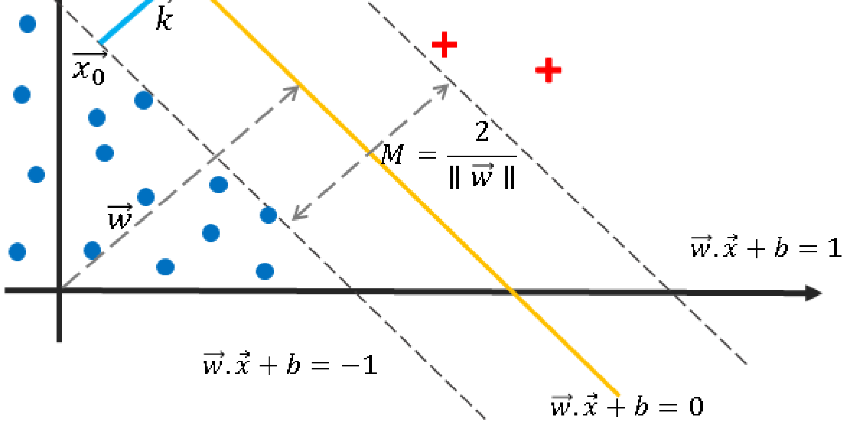
We can combine the two as below:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \forall \text{ Points } i = 1, \ldots, \text{to} \ldots, n$$

Therefore, this is the constraint for our objective function (Refer to same at the start of the article).

Now to find the Maximum Margin, it has to be represented in terms of the hyperplanes. So let's derive the relation between Margin ***M*** and the hyperplanes.





Source: Image by Author

In the above figure, let vectors ***xo*** & ***zo*** be parallel vector on the lines ***w.x+b=-1*** & ***w.x+b=1*** respectively. Then,

$$\vec{z_0} = \vec{x_0} + \vec{k} \qquad --(1)$$

where vector ***k*** is a line perpendicular to the vector ***xo*** & ***zo.***

The magnitude of vector ***k*** is ***M.***

Therefore,

$$\vec{k} = M.\left(\frac{\vec{w}}{\|\vec{w}\|}\right) \qquad [\because \vec{k} \text{ is in the same direction as } \vec{w}] \qquad --(2)$$

Now since ***zo*** & ***xo*** lie on ***w.x+b=1*** & ***w.x+b=-1***,

$$\vec{w}.\vec{z_0} + b = 1 \quad \& \quad \vec{w}.\vec{x_0} + b = -1$$

From the above eq. substituting for ***zo***,

$$\begin{aligned} & \vec{w}.(\vec{x_0}+\vec{k}) + b = 1 \qquad [\text{From } (1)] \\ \implies & \vec{w}.\vec{x_0} + \vec{w}.\vec{k} + b = 1 \\ \implies & \vec{w}.\vec{k} - 1 = 1 \qquad [\because \vec{w}.\vec{x_0} + b = -1] \\ \implies & \vec{w}.\left(M.\frac{\vec{w}}{\|\vec{w}\|}\right) = 2 \qquad [\text{From } (2)] \\ \implies & M.\frac{\|\vec{w}\|^2}{\|\vec{w}\|} = 2 \\ \implies & M = \frac{2}{\|\vec{w}\|} \end{aligned}$$

Now that we have, we need to maximize it,

$$\max(M) = \max\left(\frac{2}{\|\vec{w}\|}\right) = \min(\|\vec{w}\|)$$
$$\therefore \max(M) = \min\left(\frac{1}{2}\|\vec{w}\|^2\right) \qquad \text{[For mathematical convinience]}$$

Hence, the new goal i.e., the Objective function is:

$$\min\left(\frac{1}{2}\|\vec{w}\|^2\right)$$

$$. \qquad --(3)$$

& the inequality constraint is:

$$y_i(\vec{w}\cdot\vec{x}_i + b) \geq 1, \forall \text{ Points } i = 1, \ldots, \text{to} \ldots, n$$

$$. \qquad --(4)$$

. . .

Appendix 2 — Finding optima of the Objective fn. using Lagrangian, Dual Formulation & Quadratic Programming

General method to solve for minima

To find the optima for a curve generally, we can just

1. Take the first-order derivative,
2. Equate the derivative to 0 (for maxima or minima), to get a differential equation.
3. Solve the differential equation to find the optimal points.
4. The 2nd order derivative can provide the direction & hence we can deduce whether the optima is a minimum or a maximum.

Solving for minima when constraints are present

If there are some constraints in the differential equation, like the one we have in our Objective function, we'll first need to apply the Lagrangian multipliers (which is honestly pretty straight forward)!

$$L(w, b, \alpha) = \frac{1}{2}w^2 - \sum_{i=1}^{n} \alpha_i \cdot [y_i(w \cdot x_i + b) - 1] \quad , \alpha_i \geq 0 \qquad --(5)$$

If you observe the above equation (5), it's just the Objective function, subtracted by the inequality constraint! It must only satisfy the criteria ***αᵢ* > *0*** when the constraint is an equality constraint. However, the above equations are **inequality constraint equations**. So an additional set of conditions called the Kuhn — Tucker conditions need to satisfy as well. We will go through the conditions later.

To find the optima, just like before, we take the first-order derivative and equate it to 0:

$$\begin{aligned}
&\frac{\partial L}{\partial w} = 0 \\
\Rightarrow\ & w - \sum_{i=1}^{n} \alpha_i \cdot y_i \cdot x_i = 0 \\
\therefore\ \ & w = \sum_{i=1}^{n} \alpha_i \cdot y_i \cdot x_i \qquad --(6) \\
\&\ \ & \frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i \cdot y_i = 0 \qquad --(7)
\end{aligned}$$

To find the optimal values, we can simply substitute the values back into (5),

Expanding Eq. (5) we get,

$$\begin{aligned}
&\frac{1}{2}w^2 - \sum_{i=1}^{n}[\alpha_i \cdot (y_i \cdot w \cdot x_i + y_i \cdot b) - \alpha_i] \\
\Rightarrow\ & \frac{1}{2}w^2 - \sum_{i=1}^{n}[\alpha_i \cdot y_i \cdot w \cdot x_i + \alpha_i \cdot y_i \cdot b - \alpha_i] \\
\Rightarrow\ & \frac{1}{2}w^2 - \sum_{i=1}^{n} \alpha_i \cdot y_i \cdot w \cdot x_i - \sum_{i=1}^{n} \alpha_i \cdot y_i \cdot b + \sum_{i=1}^{n} \alpha_i
\end{aligned}$$

Now substituting the values in (6) & (7) to above,

$$\begin{aligned}
& \frac{1}{2}w^2 - w^2 + \sum_{i=1}^{n} \alpha_i \\
\Rightarrow\ \ & -\frac{1}{2}w^2 + \sum_{i=1}^{n} \alpha_i \\
\therefore\ \ & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}w^2
\end{aligned}$$

We can resubstitute the value of *w* in Eq. (6) to the above equation,

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

Here, ***W*** is called the *Objective functional* & it is a function of all (***α_i*** ... to ... ***α_n***) represented as **Λ** (Capital Lambda).

This derivation was necessary to account for the inequality constraint [Eq. (4)]. Since it's now accounted for, the Objective functional ***W*** is the new function that needs to be optimized instead of Eq. (3). This is called the *DUAL FORMULATION* because the initial Objective function has been modified!

While *Eq. (3) was minimized, **W** has to be maximized.*

Since ***W*** is a quadratic equation, it is a Quadratic Programming (QP) problem & it can be solved by an algorithm called **Sequential Minimal Optimization (SMO)**. If you've ever used the LibSVM package that is the base for SVMs in Scikit-Learn and most other SVM libraries, you'll find that the LibSVM package implements the SMO algorithm to solve for the **Maximum Margin**!

Lastly, moving onto the conditions to be satisfied for inequality constraints.

Kuhn — Tucker Conditions

1. **Dual Feasibility**:

$$\alpha_i \geq 0, \forall\, Points\ i = 1, \ldots, \ldots, n$$

2. **Complimentary Feasibility**:

$$\alpha_i \cdot [y_i(w \cdot x_i + b) - 1] = 0$$

3. **Stationary Condition**:

$$\text{These are equations (6) \& (7).}$$

4. **Primal feasibility** (our original constraint):

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1, \forall\, \text{Points}\ i = 1, \ldots, \text{to} \ldots, n$$

Duality & Complementary Slackness

When a problem can be converted to another problem whose solution is easier to compute and also provides the solution for the original problem, the two problems are said to exhibit **Duality** and the vice-versa holds true.

However, all dual functions need not necessarily have a solution providing the optimal value for the other. This can be inferred from the below *Fig. 1* where there is a *Duality Gap* between the primal and the dual problem. In *Fig. 2*, the dual problems exhibit strong duality and are said to have **complementary slackness**. Also, it is clear from the below graph that a minimization problem is converted to a maximization one. Hence we need to maximize ***W*** against $\boldsymbol{\alpha_i}$.

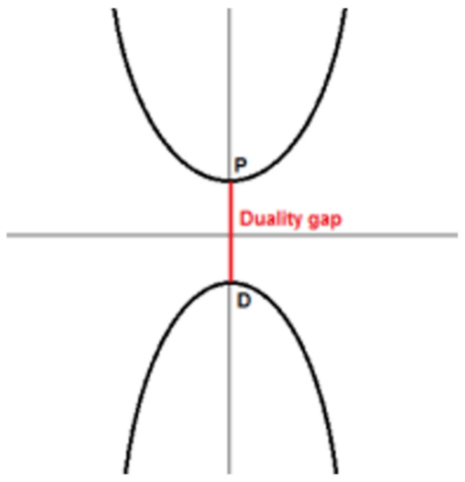


Figure 1: Weak Duality due to the Duality Gap.

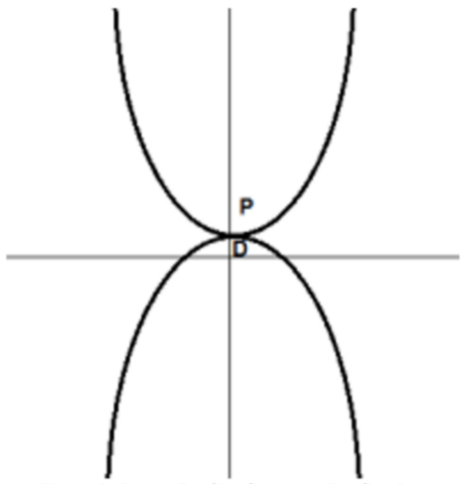


Figure 2: Strong Duality due to no Duality Gap (Exhibits Complementary Slackness)

Source: Image by Author

. . .

Appendix 3 — Deducing Optima for Soft Margin SVM

As described earlier, the Objective function for Soft Margin SVM is,

$$\frac{1}{2}w^2 + C.F\left(\sum_{i=1}^{n} \xi_i^{\sigma}\right)$$

Here, the function ***F*** is a monotonic convex function & ***C*** is a Regularization Constant.

Hence, the Lagrangian of the above Objective function is,

$$L(\mathbf{w}, \xi, b, \mathbf{\Lambda}, \mathbf{R}) = \tfrac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\left(\sum_{i=1}^{\ell} \xi_i\right)^k - \sum_{i=1}^{\ell} \alpha_i[y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i] - \sum_{i=1}^{\ell} r_i \xi_i$$

Using the same methodology as in Hard Margin SVM we get the *Objective functional* as,

$$W(\Lambda, \delta) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \frac{\delta^{k/k-1}}{(kC)^{1/k-1}}\left(1 - \frac{1}{k}\right)$$

If you observe, the extra term here is,

$$-\frac{\delta^{k/k-1}}{(kC)^{1/k-1}}\left(1-\frac{1}{k}\right)$$

Where,

$$\delta = \alpha_i + r_i$$

And the constraints for the Objective functional are:

$$\Lambda \geq 0$$

$$R \geq 0 \ \&$$

$$0 \leq \Lambda \leq \delta$$

Hence, solving the Objective functional, we can obtain the optimal Maximum Margin.

. . .

Conclusion

From the above sections, we can conclude that

1. Basic SVMs or Hard Margin SVMs are binary & linear classifiers that work only on separable data.
2. Soft Margin SVMs can work on inseparable data.
3. Kernels can be used to convert non-linear data to linear data, on which SVMs can be applied for binary classification.
4. A combination of SVMs can be used for performing multi-class classification.
5. Support Vector Machines are extremely fast algorithms as they directly solve for the Maximum Margin and do not use an iterative process like gradient descent to find the minima.

. . .

References

1. Support-Vector Networks paper.
2. https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/
3. https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-2/

4. https://www.svm-tutorial.com/2015/06/svm-understanding-math-part-3/
5. https://www.svm-tutorial.com/2016/09/unconstrained-minimization/
6. https://www.svm-tutorial.com/2016/09/convex-functions/
7. https://www.svm-tutorial.com/2016/09/duality-lagrange-multipliers/
8. Support Vector Machines Succinctly by Alexandre Kowalczyk
9. Ali Ghodsi, Lec 12: Soft margin Support Vector Machine (SVM)
10. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5822181/
11. https://www.youtube.com/watch?v=JTTiELgMyuM

11. http://fourier.eng.hmc.edu/e161/lectures/svm/

12. http://www.svcl.ucsd.edu/courses/ece271B-F09/handouts/SoftSVMs.pdf

13. Lecture 70 — Soft Margin SVMs | Mining of Massive Datasets | Stanford University

Support Vector Machine Machine Learning