Support Vector Machine

Perceptron

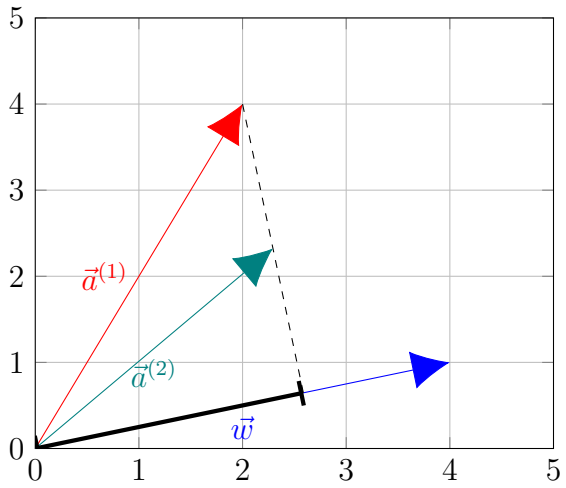
One of the simplest way to classify data into class 1 and -1 is to use a line and see which side of line the data point is at.

Let us try to write a function that determine which side of the line the data lies in. We have learned before that equation for line is just $x_2 = mx_1 + c$ but let us derive another form of a straight line using geometry.

Let $\vec{x} = [x_1, x_2, \ldots]$. Then the equation for line/plane/hyperplane is given by

$$\vec{w} \cdot \vec{x} + b = 0 \tag{1}$$

Geometrically this means that a line is a set of points whose projection along vector $\vec{w}$ is a constant[1].

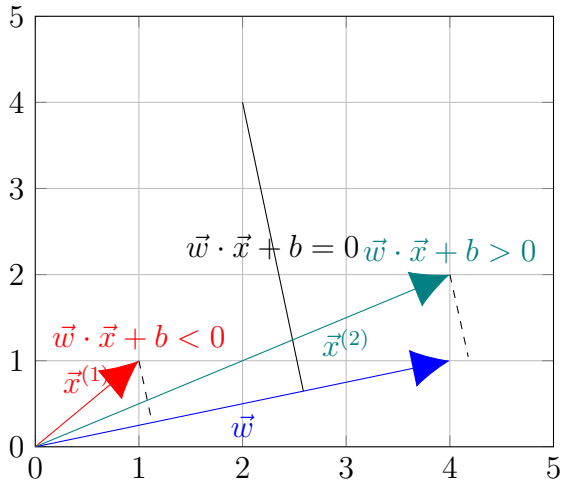


As the picture above shows, all the points on the line has exactly the same projection length along $\vec{w}$. It should be noted that one line can be represented by many $\vec{w}$ and b. We can divide or multiply Equation 1 by any constant and it still represent the same line. This means that a line is defined by $\vec{w}$ and b up to a constant factor.

This interpretation of line as set of point with constant projection allows us to decide which side of line a random point. Since the points on the line has the projection of its position vector along $\vec{w}$ equal to $-b/|\vec{w}|$, the points on the side closer to origin will have the projection shorter than $-b/|\vec{w}|$ or that means $\vec{w} \cdot \vec{x} + b < 0$ on one side.

Similarly the side away from the origin will have the projection longer than $b/|\vec{w}|$. This is illustrated in the picture below.

[1] with a factor of $|\vec{w}|$

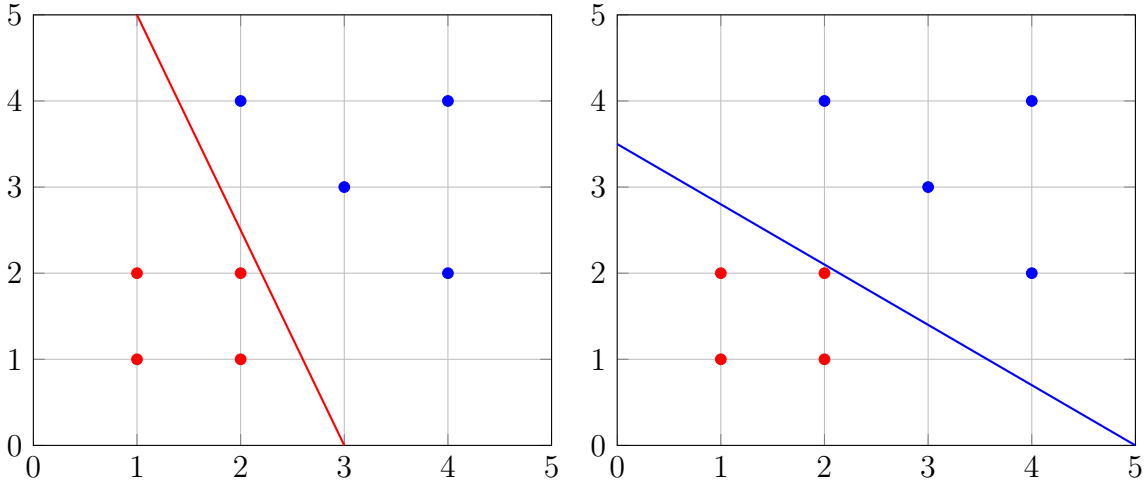


Using this fact we can build a simple classifier using a straight line. That given a $\vec{w}$ we can classify a new data point $\vec{w}$ by deciding which side it belongs to. Without loss of generality[2] we classify a data point using

$$\text{Perceptron}_{\vec{w},b}(\vec{x}) = \begin{cases} +1 & \text{if } \vec{w} \cdot \vec{x} + b \geq 0 \\ -1 & \text{if } \vec{w} \cdot \vec{x} + b < 0 \end{cases} \tag{2}$$

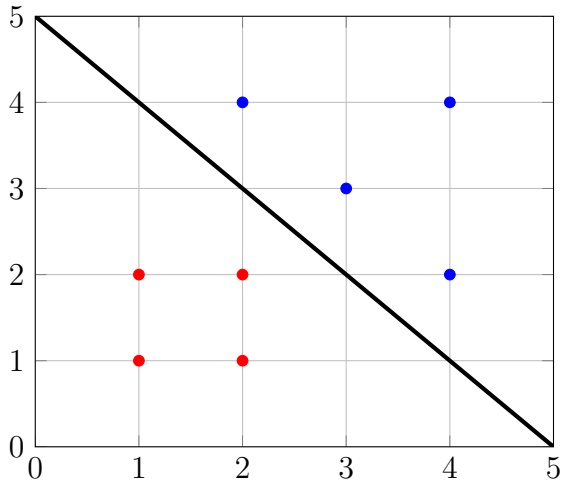
Margin

Given a set of linearly separable data points[3] there are many lines that can separate the classes. All of the lines shown below have exactly the same E_{in} of 0. But it seems more natural to us that the black line will perform the best out of sample.

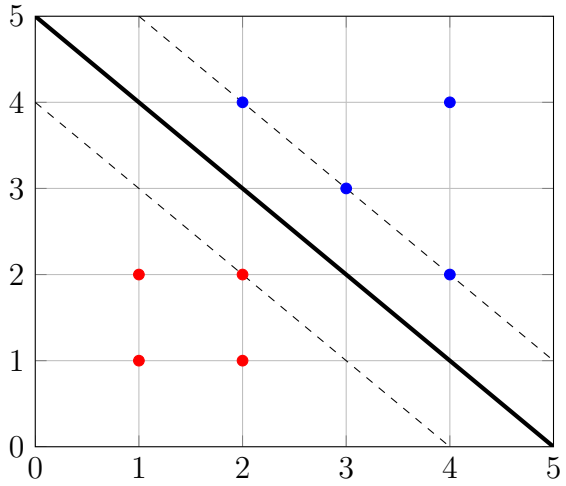


[2] If we want to flip the side, we just need to put a minus sign in front of both $\vec{w}$ and b.

[3] Data points that you separate it by a line.



The reason why the black line seems like the best choice for us is because the "margin" of the black line is the greatest. The distance of the closest point to the black line is the greatest. This magin is illustrated in the figure below.



We will learn how to find the line that has the greatest margin later. But the figure above shows us one very important thing: the line is defined by only the points on the boundary. The three red dots on the far left doesn't do anything. This implies two things $\vec{w}$ and b will depends only on the boundary points. This is the most important fact about SVM and this is why it is so powerful.

To see why, let us imagine that this is not 2 dimension. Let us say this is 100 dimensional data. Since our model($\vec{w}$, b) depends on only the boundary point. w is actually not a 100 dimension parameter which will mostlikely lead to overfitting. The underlying dimension of the line is only the number of point on the boundary.

Recall when we do non-linear transformation on the data to get curvy separating region. The number of dimension grows really fast and the same goes for the complexity of the mode. But with SVM, the real underlying number of dimension is low. Overfitting with SVM would be quite difficult. We can even go to infinite dimension with this method.

Computing Margin

Let us comeback to the problem of finding margin. Notice that Equation 1 is equivalent to

$$\vec{w} \cdot \vec{x} + b = 0 \tag{3}$$

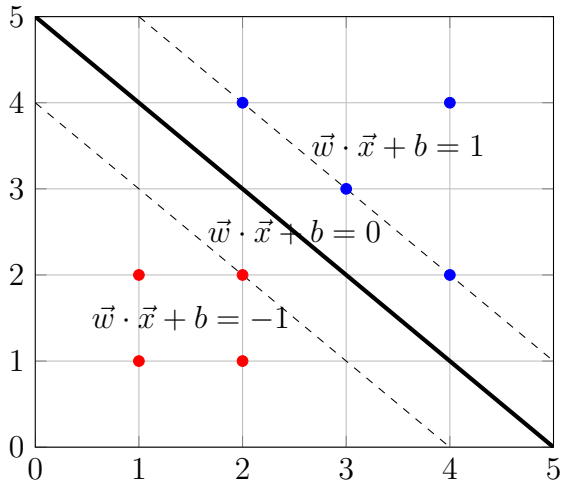
The equation for the margin line for the left and for the right is given by

$$\vec{w} \cdot \vec{x} + b = c \quad \text{Right margin} \tag{4}$$
$$\vec{w} \cdot \vec{x} + b = -c \quad \text{Left margin} \tag{5}$$

for some constant c since it is symmetric.

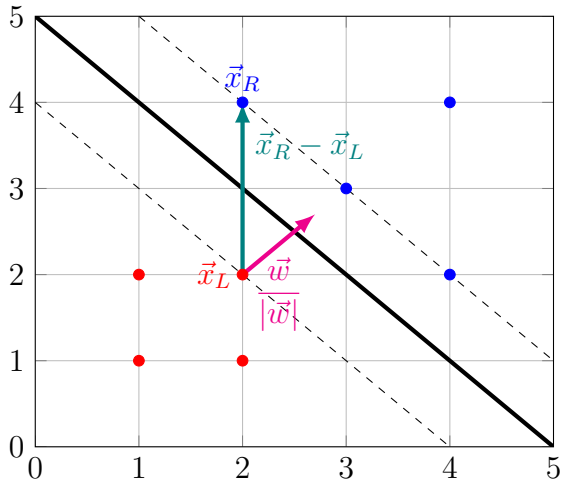
Recall that we have a choice on the normalization of $\vec{w}$ and b make the math easier, let us choose the normalization such that $c = 1$. [4]. This will make the math much easier later on.

$$\vec{w} \cdot \vec{x} + b = 1 \quad \text{Right margin} \tag{6}$$
$$\vec{w} \cdot \vec{x} + b = -1 \quad \text{Left margin} \tag{7}$$



With Equation 7 and 6, we are now ready to compute the width of the margin. Consider the following figure

[4] just dividing the whole equation by c and rename $\vec{w}$ and b



The vector $\vec{x}_R - \vec{x}_L$ points from one side of the margin to the other side of the margin. That means all we need to find the width of the margin is to project it on to unit vector along $\vec{w}$ which is $\vec{w}/|\vec{w}|$. We have

$$\text{width} = \frac{\vec{w}}{|\vec{w}|} \cdot (\vec{x}_R - \vec{x}_L) \tag{8}$$

$$= \frac{1}{|\vec{w}|} (\vec{w} \cdot \vec{x}_R - \vec{w} \cdot \vec{x}_L) \tag{9}$$

with our choice of normalization from Equation 7 and 6 we know that since $\vec{x}_L$ lies on the left margin, $\vec{w} \cdot \vec{x}_L + b = -1$. Similarly since $\vec{x}_R$ lies on the right margin, $\vec{w} \cdot \vec{x}_R + b = +1$. Plugging this back in we got

$$\text{width} = \frac{1}{|\vec{w}|}(1 - b - (-1 - b)) \tag{10}$$

$$= \frac{2}{|\vec{w}|} \tag{11}$$

This is the quantity that we want to maximize. This is equivalent to minimization of $|\vec{w}|$. And for mathematical convenient, minimize $|\vec{w}|$ is the same thing as minimization of $\frac{1}{2}|\vec{w}|^2$.

$$\max_{\vec{w},b} \text{width} \longrightarrow \min_{\vec{w},b} |\vec{w}| \longrightarrow \min_{\vec{w},b} \frac{1}{2}|\vec{w}|^2 \tag{12}$$

Constraints

Let us come back to our choice of normalization, Equation 7 and 6. With linear separable data, we require that it classify every single point correctly. This means that every data point correctly. Without loss of generality, this means that

- For data point with $y^{(i)} = +1$ we require that $\vec{w} \cdot \vec{x}^{(i)} + b > 0$ But since the point on the right margin line already have $\vec{w} \cdot \vec{x}_R + b = 1$ and all other points must go farther than that. This means that for every data point with class $+1$,

$$\vec{w} \cdot \vec{x}^{(i)} + b \geq 1$$

- Similarly for data point of class $y^{(i)} = -1$. We require that

$$\vec{w} \cdot \vec{x}^{(i)} + b < -1$$

The two inequality can be written succinctly as for every data point

$$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) > 1 \tag{13}$$

Optimization Problem

Combining what we learned in Equation 12 and 13. The problem of find the line with the maximum margin become

$$\begin{aligned} \underset{\vec{w}}{\text{minimize}} \quad & \frac{1}{2}|\vec{w}|^2 \\ \text{subject to} \quad & y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 1 \; i = 1, \ldots, m. \end{aligned} \tag{14}$$

This is a contrained optimization problem, normal gradient descent wouldn't help us much here since we have a complex contraints. In fact, the number of constraints is equal to the number of data points. First we will get away with this problem by using slack variable and then we will deal with it head on since it will give us a useful insight.

Slack Variable

So far we have dealt with linearly separable data. But more often than not we will have a non linearly separable data. That means the condition in the previous section might never be satisfies. Instead of rejecting it out right, we will introduce the penalty for not lying behind the right side of margin. It should be noted that this lying within the margin zone will also be penalized.

When our model defined by $\vec{w}$ and b fail the constraints this means that for that datapoint $x^{(i)}, y^{(i)}$

$$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) < 1$$

Let us define a positive number ξ_i called slack variable. The above can be written as.

$$y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) = 1 - \xi_i$$

We can think of ξ_i as how far off this datapoint is from the correct side. But first let us rewrite the above equation

$$\xi_i = 1 - y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \tag{15}$$

This expression is really close to what we want to penalize, the problem is that we only want to penalize only those that are on the wrong side. Those that are on the wrong side will have a positive ξ_i. When it is on the correct side ξ_i is negative So the penalty called soft margin penalty looks like

$$\text{Soft Margin} = \sum_i \max(0, \xi_i) \tag{16}$$

Combine this with margin size we found earlier we have our cost function

$$cost(\vec{w}, b) = \frac{1}{2}|\vec{w}|^2 + C\sum_i \max(0, 1 - y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b)) \tag{17}$$

where C is a constant that tells how much we are willing to trade off between having a large margin and having some point on the wrong side of the margin. Minimizing this is not that hard but one need to be a little bit careful since the derivative is not well defined at some point.

Relation to Logistic Regression and Regularization

Equation 17 looks awfully familiar especially the term $|\vec{w}|^2$. This is the regularization term we have seen before. Recall that when we did regularization we try to minimize the error and we penalize the complexity by $|\vec{w}|^2$ and we end up with a cost function that looks like.

$$\text{cost} = \text{Error} + \lambda\text{Complexity}$$

Equation 17 is the same idea but from different view. This time we minimize the complexity while penalize the error. This leads us to exactly the same idea.

In fact what we found is closely related to logistic regression. Recall the cost function for logistic regression with regularization term[5]

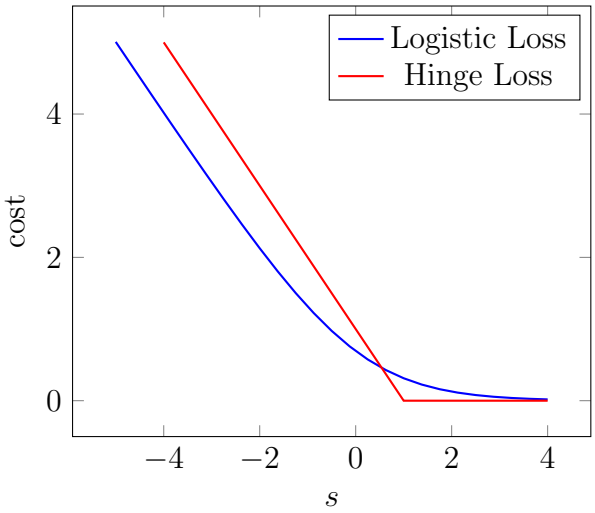
$$cost(\vec{w}, b) = \frac{1}{N}\sum_i \log\left(1 + e^{-y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)}\right) + \lambda|\vec{w}|^2$$

Let us plot out the each of error term versus $s = y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b)$.

$$\text{Logistic Loss} = \text{Log Loss} = \log\left(1 + e^{-s}\right)$$

The error term for soft margin shown in Equation 17.

$$\text{Soft Margin Loss} = \text{Hinge Loss} = \max(0, 1 - s)$$



We can see that the two loss function looks very close. The difference is that hinge loss doesn't penalize anything above one thus making no effort of adjusting the model to push these point any further away from the margin.

[5]Here I expand the padded feature term explictly and call w_0 a b. This so that we have the same convention.

Generalized Lagrange Multiplier

Lagrange multiplier we have learned so far only deal with the equality constriant. That is we can solve the problem of the form

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & f(\vec{x}) \\ \text{subject to} \quad & g_i(\vec{x}) = 0,\ i = 1, \ldots, m. \end{aligned}$$

by just using lagrange multiplie and the new objective function is called Lagragian

$$\mathcal{L}(\vec{x}, \lambda_i) = f(\vec{x}) + \sum_i \lambda_i g_i(\vec{x}) \tag{18}$$

Then the whole thing just becomes

$$\underset{\vec{x}, \lambda_i}{\text{minimize}} \quad L(\vec{x}, \lambda_i)$$

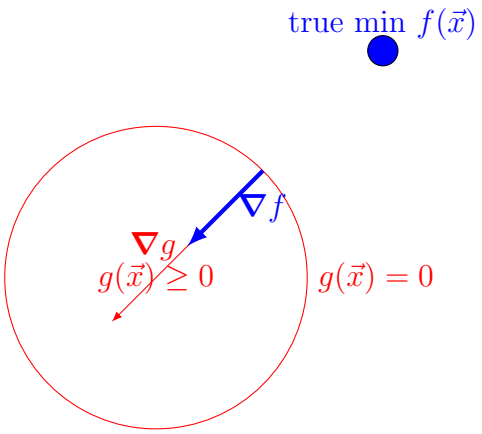
One useful thing about lagrangian is that we already encode the contraints right into the Lagragian. Notice carefully that the minimization is also over λ_i. The partial derivative of $\mathcal{L}$ with respect to λ gives us exactly the constaint.

But the problem we have in 14 force us to deal with inequality constain. The general form of this problem looks like[6][7]

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & f(\vec{x}) \\ \text{subject to} \quad & g_i(\vec{x}) \geq 0,\ i = 1, \ldots, m. \end{aligned}$$

The same idea that is used to give us lagrange multiplier can be used here. $g(\vec{x}) \geq 0$ defines a closed region. There are two cases on where the minimimum can occur.

1. First the true minimum of f lies inside the region $g(\vec{x}) \geq 0$. In this case, the constraint does not do anything and all we need to do is to minimize f. These contraints are call inactive contraints. Alternatively, we can think about this as having the same old lagrangian except that $\lambda = 0$. $\boldsymbol{\nabla} f(\vec{x}) = \lambda \boldsymbol{\nabla} g(x)$
2. Second, the true minimum of f lies outside the region $g(\vec{x}) \geq 0$



[6] Read the inequality sign and whether it's minimize or maximize carefully.

[7] Sometime people write the constraints as $\vec{g}(\vec{x}) \succeq 0$.

In this case we say that the contrain $g(\vec{x})$ is active. Then we know that the minimum with constrain has to lies on the boundary line of $g(\vec{x}) \geq 0$ which is the line defined by $g(\vec{x}) = 0$.

In this case we just get our same old Lagrangian. The extremum has to happen at the point where $\boldsymbol{\nabla} f \parallel \boldsymbol{\nabla} g$. But, we need to be a little bit careful with the directions. Since the region is defined by $g(\vec{x}) \geq 0$ all the gradient of g at the boundary will be pointing inward. Since the true minimum of f lies outside of g the gradient will need to be pointing away from the true minimum. So we have that f must be pointing in the same direction and parallel to g.[8]

This means $\boldsymbol{\nabla} f = \lambda \boldsymbol{\nabla} g \longrightarrow \boldsymbol{\nabla} f - \lambda \boldsymbol{\nabla} g = 0$ with $\lambda > 0$.

Combining the two cases we have our lagrangian

$$\mathcal{L}(\vec{x}, \lambda) = f(\vec{x}) - \lambda g(\vec{x}) \tag{19}$$

with $\lambda \geq 0$. The first case just put an equal sign in addition to the $>$ sign we got in the second case. Similar to the normal case that if we have more than one constraint then we just need to add up the lagrange multiplier and the function.

The problem of minimize $f(\vec{x})$ subject to constraints is then turned into the problem of optimizing the lagrangian with a catch.

- First, we want to maximize $\mathcal{L}$ over λ. It can be understand this way. We want to have as many inactive contraints as possible since that means the minimum can go even lower. This means that $\lambda = 0$ would give us the best minimum.

  But since $g(\vec{x}) > 0$ and $\lambda \geq 0$. The last term with the minus sign is always ≤ 0 so $\lambda = 0$ will be the maximum of $\mathcal{L}$. [9]

- Second, obviously we want to minimize this with respect to $\vec{x}$.

Let us summarize what we found. We found that

$$\begin{aligned} \underset{\vec{x}}{\text{minimize}} \quad & f(\vec{x}) \\ \text{subject to} \quad & g_i(\vec{x}) \geq 0,\ i = 1, \ldots, m. \end{aligned}$$

is equivalent to minimizing with respect to $\vec{x}$ and maximize with respect to α_i the Lagrangian given by

$$\mathcal{L}(\vec{x}, \alpha_i) = f(\vec{x}) - \sum_i \alpha_i g_i(\vec{x}) \tag{20}$$

with $\alpha_i \geq 0 \ \ \forall i$

Dual Form

Our problem defined in 14 reads

$$\begin{aligned} \underset{\vec{w}}{\text{minimize}} \quad & \frac{1}{2}|\vec{w}|^2 \\ \text{subject to} \quad & y^{(i)}(\vec{w} \cdot \vec{x}^{(i)} + b) \geq 1 \ i = 1, \ldots, m. \end{aligned} \tag{21}$$

[8] Hand wavy argument here but you can think about it as that the point on the other side will give you the maximum instead.

[9] Hand-waving argument here if you need a rigorous treatment. Google Stanford CS229 SVM.

Let us first transform it to the form we know how to find Lagrangian. All we need to do is to modify the contrain so that it becomes ≥ 0.

$$\begin{array}{ll} \underset{\vec{w}}{\text{minimize}} & \frac{1}{2}|\vec{w}|^2 \\ \text{subject to} & y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1 \geq 0 \; i=1,\ldots,m. \end{array} \tag{22}$$

The equation above is called the primal form of the optimization problem.

Comparing this with Equation 20

$$g_i(\vec{w}) = y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1 \tag{23}$$

Thus the Lagrangian is given by

$$\mathcal{L}(\vec{x},\alpha_i) = \frac{1}{2}|\vec{w}|^2 - \sum_i \alpha_i(y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1) \tag{24}$$

with $\alpha_i \geq 0 \;\; \forall i$.

The minimization problem then becomes

$$\begin{array}{ll} \underset{\vec{w},b}{\text{minimize}}, \underset{\vec{\alpha_i}}{\text{maximize}} & \frac{1}{2}|\vec{w}|^2 - \sum_i \alpha_i(y^{(i)}(\vec{w}\cdot\vec{x}^{(i)}+b)-1) \\ \text{subject to} & \alpha_i \geq 0 \; i=1,\ldots,m. \end{array} \tag{25}$$

Let us take derivative of the Lagrangian and set it equal to zero.

$$\frac{\partial\mathcal{L}}{\partial w_j} = w_j - \sum_i \alpha_i y^{(i)} x_j^{(i)} = 0 \tag{26}$$

In other word,

$$\vec{w} = \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} \tag{27}$$

the vector $\vec{w}$ is just a linear combination of the features. But one should be a little bit worred here that $\vec{w}$ is a sum over all the data. The number of dimension would be large and we would definitely overfit. However, the fact that constrain is active only for the point on the boundary gives us that only those points on the boundary will have $\alpha_i \neq 0$. These data points are called **support vector**. So we are not going to overfit here.

Taking derivative with respect to b gives[10]

$$\frac{\partial\mathcal{L}}{\partial b} = \sum_i \alpha_i y^{(i)} = 0 \tag{28}$$

[10] if we use softmargin we need to modify this.

Taking derivative with respect to α is a bit tricky. The derivative with respect to α_i is defined everywhere except for i that makes $\alpha_i = 0$. So, let us take derivative with respect to α_k where $\alpha_k \neq 0$. These are the α_i that belongs to support vector that we discuss before. This means that for these k's

$$y^{(k)}(\vec{w} \cdot \vec{x}^{(k)} + b) - 1 = 0 \longrightarrow b = \frac{1}{y^{(k)}} - \vec{w} \cdot \vec{x}^{(k)} \tag{29}$$

where w can be obtained from Equation(27). Since $y^{(k)}$ can only be ± 1 we can simplify it further to

$$b = y^{(k)} - \vec{w} \cdot \vec{x}^{(k)} \tag{30}$$

This is good in mathematics but for practical reason we pick k such that α_k is largest among all the α's.

To get the dual form of the problem all we need to do is eliminate the primary variable $\vec{w}$ and b. What we are going to have is everything in terms of α_i. Pluging $\vec{w}$ back into the original objective function gives.

$$\begin{aligned}
\mathcal{L} =& \frac{1}{2} \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} \cdot \sum_k \alpha_k y^{(k)} \vec{x}^{(k)} - \sum_i \alpha_i (y^{(i)} (\sum_k \alpha_k y^{(k)} \vec{x}^{(k)} \cdot \vec{x}^{(i)} + b) - 1) && (31)\\
=& \frac{1}{2} \sum_i \sum_k \alpha_i \alpha_k y^{(i)} y^{(k)} \vec{x}^{(i)} \cdot \vec{x}^{(k)} - \sum_i \sum_k \alpha_i \alpha_k y^{(i)} y^{(k)} \vec{x}^{(i)} \cdot \vec{x}^{(k)} - \sum_i \alpha_i y_i b + \sum_i \alpha_i && (32)\\
=& -\frac{1}{2} \sum_i \sum_k \alpha_i \alpha_k y^{(i)} y^{(k)} \vec{x}^{(i)} \cdot \vec{x}^{(k)} - \sum_i \alpha_i y_i b + \sum_i \alpha_i && (33)\\
=& -\frac{1}{2} \sum_i \sum_k \alpha_i \alpha_k y^{(i)} y^{(k)} \vec{x}^{(i)} \cdot \vec{x}^{(k)} + \sum_i \alpha_i && (34)
\end{aligned}$$

where in the last step we use Equation 28.

And our classification becomes

$$\vec{w} \cdot \vec{x}' + b \longrightarrow \sum_i \alpha_i y^{(i)} \vec{x}^{(i)} \cdot \vec{x}' + b \tag{35}$$

So we have that our lagrangiang depends solely on the dot product between feature vector and the classification function depends solely on the dot product. This is a very important fact which allows us to do classification in infinite dimension.

Let us wrap up here about what exactly we are trying to optimize. Recall that we want to maximize $\mathcal{L}$ with respect to α_i. This means that we want to minize the negative of what we got in 34 subject to all the constraints we have along the way. That is

$$\begin{aligned}
\underset{\alpha_i}{\text{minimize}} \quad & \frac{1}{2} \sum_i \sum_k \alpha_i \alpha_k y^{(i)} y^{(k)} \vec{x}^{(i)} \cdot \vec{x}^{(k)} - \sum_i \alpha_i \\
\text{subject to} \quad & \alpha_i \geq 0, \ i = 1, \ldots, m. \\
& \sum_i \alpha_i y_i = 0
\end{aligned} \tag{36}$$

This can be easily plugged in to constrained minimization routine like scipy.optimize.minimize with method SLSQP[11] The optimization routine will gives us back the α_i which we expect to find that most of them are zero. Then we compute b using equation 30. Then we are ready to classify data point using 35.

Kernel Trick

In this section we are going to exploit the fact that the final minimization problem 36 and the classification rule only depends on the dot product of the datapoints.

Recall what we have when we want a carved out a curved region, we transform it to higher dimesion for example[12]

$$\phi(x_1, x_2) = [x_1, x_2, x_1^2, x_2^2, x_1 x_2] \tag{37}$$

What we have done before is to explicitly compute the mapping and do all the classifcation base on that. The problem with this is that this size mapping grows exponentially fast so we can not use too high dimension.

The very neat thing about what we found in 36 is that it only depend on the dot product but not exactly the specific mapping of the variable. To see this, let

$$k(\vec{x}, \vec{x}') = \phi(\vec{x}) \cdot \phi(\vec{x}'). \tag{38}$$

The function $k(\vec{x}, \vec{x}')$ is called the kernel. If we can show that, a function that takes two vector and returns a dot product of some mapping then that function is a kernel. For example,

$$k(\vec{u}, \vec{v}') = u_1 v_1 + u_2 v_2$$

is a kernel since $\phi(\vec{u})$ is just identity.

Another example of kernel is

$$\begin{aligned} k(\vec{u}, \vec{v}) &= (\vec{u} \cdot \vec{v})^2 && (39)\\ &= \left(\sum_i u_i v_i\right)\left(\sum_j u_j v_j\right) && (40)\\ &= \sum_i \sum_j (u_i u_j)(v_i v_j) && (41)\\ &= \sum_{i,j} (u_i u_j)(v_i v_j) && (42) \end{aligned}$$

from the above we can see that the mapping(for 3 dimension) is just

$$\phi(\vec{u}) = [u_1 u_1, u_1 u_2, u_1 u_3, u_2 u_1, u_2 u_2, u_2 u_3, u_3 u_1, u_3 u_2, u_3 u_3]$$

This is an example of computing kernel without really having to compute the mapping. This will allow us to go to infinite dimension.

[11] usually people use CVXOPT.

[12] What we have done earlier has the offset 1 to it but since we separate out the offset b.

Another popular kernel is polynomial kernel defined by

$$k(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v} + c)^d$$

for some constant c. This function allows us to go to $\binom{n}{d}$ dimension with just $O(n)$.[13]

The main show case for this kernel trick is called the radial basis function(RBF)

$$k(\vec{u}, \vec{v}) = \exp -\frac{|\vec{u} - \vec{v}|^2}{\sigma^2} \tag{43}$$

This of course doesn't look like a dot product of any kind. But, let's us see why this one is a kernel with 1D vector. Let's expand it and do a taylor series expansion.[14]

$$\begin{aligned}
k(u, v) &= \exp -\frac{|\vec{u} - \vec{v}|^2}{\sigma^2} && (44)\\
&= \left(\exp -u^2\right)\left(\exp -v^2\right)\left(\exp(-2uv)\right) && (45)\\
&= \left(\exp -u^2\right)\left(\exp -v^2\right)\left(\sum_k -\frac{2^k}{k!}u^k v^k\right) && (46)\\
&= \sum_k -\frac{2^k}{k!}\left(u^k \exp -u^2\right)\left(v^k \exp -v^2\right) && (47)
\end{aligned}$$

The two things in the parenthes of the last line is then the mapping from 1 to infinite dimension. This proof can be applied if we start with more than 1 dimension. You are welcome to do it as an exercise. The computation of this kernel is not infinite it's just the same old $O(n)$.

Let us rewrite the optimization and the classification rule in terms of kernel.

$$\begin{aligned}
\underset{\alpha_i}{\text{minimize}} \quad & \frac{1}{2}\sum_i \sum_k \alpha_i \alpha_k y^{(i)} y^{(k)} k(\vec{x}^{(i)}, \vec{x}^{(k)}) - \sum_i \alpha_i \\
\text{subject to} \quad & \alpha_i \geq 0,\ i = 1, \ldots, m. \\
& \sum_i \alpha_i y_i = 0
\end{aligned} \tag{48}$$

and the classification rule

$$\sum_i \alpha_i y^{(i)} k(\vec{x}^{(i)}, \vec{x}') + b \tag{49}$$

One can also combine kernel trick with soft margin but it is one of those thing I wouldn't even bother to write myself. Just use a package like scikit learn. But now you understand all the parameters.

[13] If you want proof see wikipedia.

[14] Let's drop σ here to make the calculation a bit nicer