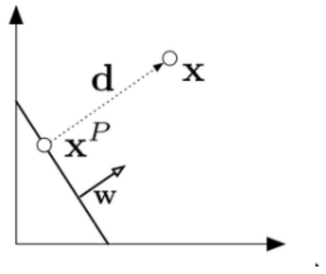
SVM notes

Matteo Lugli

December 2, 2023

1 Margins



Considering the above figure, $x^p = x - d$ and $d = w\alpha$. But x^{p} is also a point on the hyperplane, so $w^t x^p + b = 0$. By substituting we get $w^t(x - \alpha w) + b = 0$.

$$\begin{gathered}w^t - w^t \alpha w + b = 0 \\ \frac{w^t x + b}{w^t w} = \alpha \\ d = \frac{w^t x + b}{w^t w} w\end{gathered}$$

Now we want to calculate the magnitute of the distance d, which is

$$\begin{aligned}||d||_2 &= \sqrt{d^t d} = \sqrt{\alpha^2 w^t w} = \sqrt{\alpha^2 w^t w} = \alpha\sqrt{w^t w} \\ &= \frac{w^t x + b}{w^t w}\sqrt{w^t w} = \frac{w^t x + b}{\sqrt{w^t w}} = \frac{w^t x + b}{||w||_2}\end{aligned}$$

So the margin is defined as follows:

$$\gamma(w,b) = min_{x\in D}\Big(\frac{w^tx+b}{||w||_2}\Big) \tag{1}$$

In the SVM problem we want to find the separating hyperplane that maximizes the margin:

$$\begin{aligned} max_{w,b}(\gamma(w,b)) &= max_{w,b}\Bigg(min_{x\in D}\Big(\frac{w^tx+b}{||w||_2}\Big)\Bigg) \\ &= max_{w,b}\Bigg(\frac{1}{||w||_2}min_{x\in D}\Big(w^tx+b\Big)\Bigg) \end{aligned}$$

If we consider the hyperplane $w^tx + b$ it is scale invariant, meaning that we can multiply w and b for whatever number β and nothing will change. Intuitively, you can thing of it this way: imagine that the weights of your classifier are $[0.1, 0.2, 0.3]$. It means that feature 1 of your vector has importance 0.1 on the overall classification, feature 2 has importance 0.2, ecc. . .
The important thing here is that feature 2 is twice more important than feature 1, so if you multiply everything by the same amount nothing will change. It means that we can set β so that $w^t + b = 1$, obtaining the following objective function:

$$max_{w,b}\frac{1}{||w||_2}(1) = min_{w,b}||w|| \tag{2}$$

Given that $f(z) = z^2$ is a monotonic function, we can write our full optimization problem like this:

$$\begin{aligned} & min_{w,b}||w||^2 \\ s.t \quad & \forall_i \quad y_i(w^tx_i+b) \geq 0 \\ & min_i \quad w^tx_i + b = 1 \end{aligned}$$

We can intuitively combine the two costraints in a single one and get the final formulation:

$$\begin{aligned} & min_{w,b}||w||^2 \\ s.t \quad & \forall_i \quad y_i(w^tx_i+b) \geq 1 \end{aligned} \tag{3}$$

This problem can be solved by using a quadratic programming solver, and it's totally fine. Moreover there are many ways to re-formulate and solve the problem, which are particularly usefull when our data is not linearly separable.

2 Lagrangian

In machine learning we usually deal with problems in which points are not linearly separable. The previous problem can be re-written in a way that allows us to use the so called **Kernel Trick**. To do that we need to introduce the *Lagrangian formulation.* It is basically a way to reduce a constrained optimization problem into a single equation.

$$min f(x)$$
$$s.t \quad h_i(x) \geq 0 \quad \forall i = 1...m$$
$$\Downarrow$$

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i h_i(x) \tag{4}$$

where λ_i are called **lagrangian multipliers** and are all ≥ 0.
The lagrangian dual function is defined like this:

$$g(\lambda) = min_x \Big(L(x, \lambda) \Big) \tag{5}$$

According to the **strong duality** we can write that

$$d^* = max_{\lambda \geq 0} g(\lambda) = min f(x) = p^* \tag{6}$$

where d* is the solution of the dual and p* is the solution of the primal. So to find the equation of the dual, we must set $\frac{dL}{dx_1} = 0$, $\frac{dL}{dx_2} = 0$, $\ldots$, $\frac{dL}{dx_n} = 0$. If we plug what we obtain in L we get a function of λ only, which is what we need. So let's try to do that for our specific minimization problem and see what we get. Remember that we have a constraint for each one of the points in our dataset, so $x^1 \ldots x^i$

$$L(w, b, \lambda) = \frac{1}{2} w^t w + \sum_{i=1}^{m} \lambda_i (1 - y_i(w^t x^{(i)} + b)) \tag{7}$$

So if we simply compute the derivatives $\frac{dL}{dw} = 0$ and $\frac{dL}{db} = 0$ we get:

$$\begin{aligned} \frac{dL}{db} &= 0 \Rightarrow \sum_{i=1}^{m} \lambda_i y_i = 0 \\ \frac{dL}{dw} &= 0 \Rightarrow w = \sum_{i=1}^{m} \lambda_i y_i x_i \end{aligned} \tag{8}$$

Let's plug the easiest of the two conditions (8) in the lagrangian and see what we get:

$$\frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j {x^{(i)}}^t x^{(j)} + \sum_i \lambda_i - \sum_{i,j}\lambda_i y_i \lambda_j y_j {x^{(i)}}^t x^{(j)} \tag{9}$$

This sum is composed by three terms: if you look carefully the first and the last one are the same (except for the $\frac{1}{2}$), so we can simplify:

$$\sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j {x^{(i)}}^t x^{(j)}$$

let's write the formulation of the new problem:

$$\begin{aligned} &max_{\lambda\geq 0}\sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \langle x^{(i)} x^{(j)}\rangle \qquad (10)\\ &s.t \quad \sum_{i=1}^{m}\lambda_i y_i = 0 \end{aligned}$$

Now we have a new objective function that only depends on λ. It also contains a inner product on which we can apply a **Kernel** (explained later).

2.1 Lagrangian in non linear case

When we deal with non-linearly separable data, we introduce the slack variables ξ_i, needed to insert some amount of tolerance when separating the data. The original problem is the following:

$$\begin{aligned} &\min \frac{1}{2}w^t w + C\sum_i \xi_i \\ &s.t \quad y_i(wx_i + b) \geq 1 - \xi_i \quad i = 1\cdots N \qquad (11) \end{aligned}$$

Let's write the formulation of the lagrangian:

$$L(w,b,\xi,\alpha,\beta,C) = \frac{1}{2}w^t w + C\sum_i \xi_i - \sum_i \alpha_i[y^{(i)}({x^{(i)}}^t w + b) - 1 + \xi_i] - \sum_i \beta_i \xi_i \tag{12}$$

remembering that:

- $x^{(i)}$ are the vectors corresponding to the samples (each sample is a vector indeed);
- $y^{(i)}$ are the labels;

- α and β are the lagrangian multipliers. We want the new dual problem to have only these as variables;

As usual, to write the dual formulation we need to compute the derivative with respect to each one of the variables that we don't want in the new problem, ξ, w, b in this case. We don't derive with respect to C because we'll se that it will cancel out.

$$\frac{dL}{dw} = 0 \Rightarrow w^* = \sum_i \alpha_i^* y^{(i)} x^{(i)} \tag{13}$$

$$\frac{dL}{db} = 0 \Rightarrow \sum_i \alpha_i^* y^{(i)} = 0 \tag{14}$$

$$\begin{aligned} \frac{dL}{d\xi} &= 0 \Rightarrow C - \alpha_i - \beta_i = 0 \\ &\Rightarrow C = \alpha_i + \beta_i \qquad (15) \\ &\Rightarrow 0 \leq \alpha_i \leq C \end{aligned}$$

the last relation has to be true because the lagrangian multipliers need to be positive. Let's plug 13 in the lagrangian (using y_i instead of $y^{(i)}$ to have a lighter notation):

$$\frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i^t x_j + C\sum_i \xi_i - \sum_{i,j} a_i a_j y_i y_j x_i^t x_j - b\sum_i \alpha_i y_i + \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \beta_i \xi_i \tag{16}$$

- The 1^o and the 2^o term are the same except for the $\frac{1}{2}$;
- The 4^o term is null because of 14;

$$-\frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i^t x_j + C\sum_i \xi_i + \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \beta_i \xi_i \tag{17}$$

Let's use equation 15 and re-write the C:

$$\begin{aligned} &-\frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i^t x_j + (\alpha_i + \beta_i)\sum_i \xi_i + \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \beta_i \xi_i \\ &-\frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i^t x_j + \sum_i \xi_i \alpha_i + \sum_i \xi_i \beta_i + \sum_i \alpha_i - \sum_i \alpha_i \xi_i - \sum_i \beta_i \xi_i \\ &-\frac{1}{2}\sum_{i,j} a_i a_j y_i y_j x_i^t x_j + \sum_i \alpha_i \qquad (18) \\ &s.t \quad 0 \leq \alpha_i \leq C \\ &s.t \quad \sum_i a_i y_i = 0 \end{aligned}$$

If we carefully look at equation 18 we notice that the formulation is basically the same as the one in the linearly separable case, we just have an extra bound on the value of α_i (quite an easy bound to consider), that now is limited by C.

2.2 Quadratic formulation derived from dual formulation

In some textbooks you'll find the dual formulation written this other way: (note that bold symbols indicate vectors)

$$-\frac{1}{2}\boldsymbol{\alpha^t} Q \boldsymbol{\alpha} + \sum_i \alpha_i \tag{19}$$

Where Q is a square matrix, so the objective function is quadratic. In such context it is easy to perform gradient descent, because we have a convex function of which we can analitically compute the derivative. Let's see how to go from 18 to 19. We will se a easy example to give you an idea. Let's suppose that we only have 2 constraints.

$$Q_{i,j} = y^{(i)} y^{(j)} x^{(i)^t} x^{(j)} \tag{20}$$

$$\sum_{i,j} a_i a_j y^{(i)} y^{(j)} x^{(i)t} x^{(j)} = \sum_{i,j} a_i a_j Q_{i,j} \tag{21}$$

The matrix Q is the following (let's use the underscore notation for vectors for simplicity):

$$Q = \begin{bmatrix} y_1 y_1 x_1^t x_1 & y_1 y_2 x_1^t x_2 \\ y_2 y_1 x_1^t x_2 & y_2 y_2 x_2^t x_2 \end{bmatrix} \tag{22}$$

If we do the summation iterations manually, so in a programming fashion with nested loops (for i, for j), we end up with the following:

$$\alpha_1\alpha_1 y_1 y_1 x_1^t x_1 + \alpha_1\alpha_2 y_1 y_2 x_1^t x_2 + \alpha_2\alpha_1 y_2 y_1 x_1^t x_2 + \alpha_2\alpha_2 y_2 y_2 x_2^t x_2 \tag{23}$$

Let's now try to do the computation written in equation 19. Remember that:

- $\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}$
- $\boldsymbol{\alpha^t} = [\alpha_1 \alpha_2]$

Let's also do a quick trick to analyze the dimensions that we should have in output (forgive the not so formal notation):

$$\boldsymbol{\alpha}^{\boldsymbol{t}} Q \Rightarrow 1 \times n \cdot n \times n \Rightarrow 1 \times n \tag{24}$$

$$\Rightarrow 1 \times n \cdot \boldsymbol{\alpha} \Rightarrow 1 \times n \cdot n \times 1 \Rightarrow scalar \tag{25}$$

Let's perform 24 on our example:

$$[\alpha_1 \alpha_2] \begin{bmatrix} y_1 y_1 x_1^t x_1 & y_1 y_2 x_1^t x_2 \\ y_2 y_1 x_1^t x_2 & y_2 y_2 x_2^t x_2 \end{bmatrix} = [\alpha_1 y_1 y_1 x_1^t x_1 + a_2 y_2 y_1 x_1^t x_2, \alpha_1 y_1 y_2 x_1^t x_2 + \alpha_2 y_2 y_2 x_2^t x_2] \tag{26}$$

Let's now perform 25:

$$[\alpha_1 y_1 y_1 x_1^t x_1 + a_2 y_2 y_1 x_1^t x_2, \alpha_1 y_1 y_2 x_1^t x_2 + \alpha_2 y_2 y_2 x_2^t x_2] \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \tag{27}$$

$$\alpha_1 \alpha_1 y_1 y_1 x_1^t x_1 + \alpha_1 \alpha_2 y_2 y_1 x_1^t x_2 + \alpha_2 \alpha_1 y_1 y_2 x_1^t x_2 + \alpha_2 \alpha_2 y_2 y_2 x_2^t x_2 \tag{28}$$

Which is the same as 23

3 KKT conditions

There is a series of equations called KKT conditions that summarize the relation of the primal and the dual problem. If x* is a solution for the primal problem, f is convex and all of the $h_i(x)$ are linear constraints, then there exist some multipliers $\lambda_i \dots \lambda_m$ such that:

$$\nabla f(x^*) - \lambda^t \nabla h(x^*) = 0 \tag{29}$$

$$h(x^*) \geq 0 \tag{30}$$

$$\lambda_i \geq 0 \quad \forall i = 1 \dots m \tag{31}$$

$$\lambda^t h(x^*) = 0 \tag{32}$$

where λ^{t} is the solution of the dual! The first 3 conditions are quite intuitive, the last one is the tricky one. Let's write the demonstration for that. Let x^* be the solution of the primal and λ^* be the solution of the dual. Because of the strong duality, we can write the following:

$$\begin{aligned} f(x^*) = g(\lambda^*) = min_x\Big(f(x) + \sum_{i=1}^{m} \lambda_i^* h_i(x)\Big) \\ &\geq f(x^*) + \sum_{i=1}^{m} \underbrace{\lambda_i^*}_{\geq 0} \underbrace{h_i(x^*)}_{\geq 0} \\ &\geq f(x^*) \end{aligned}$$

which means that $\sum_{i=1}^{m} \lambda_i^* h_i(x^*) = 0 \Rightarrow {\lambda^*}^T h(x^*) = 0$.
This means that if we have a lagrangian multiplier $\lambda_i \geq 0$, then the relative constraint should be active (it is satisfied with equality), so $h_i(x^*) = 0$.

4 Kernels

$$\max -\frac{1}{2}\boldsymbol{\alpha_t} Q\boldsymbol{\alpha} + \sum_i \alpha_i$$

$$\text{s.t} \quad 0 \le \alpha_i \le C \tag{33}$$

$$\text{s.t} \quad \sum y_i \alpha_i = 0 \tag{34}$$

where $Q_{i,j} = y_i y_j x_i^t x_j$, or $Q_{i,j} = y_i y_j \langle x_i, x_j \rangle$. Once we find the solutions of the problem ($\boldsymbol{\alpha}^*$), we can compute the solutions of the primal $\boldsymbol{w}^*$ and b^* thanks to the kkt conditions. To classify a point, we do

$$\text{sign} \quad \boldsymbol{w}^* \cdot \varphi(\boldsymbol{x}) + b^* = \sum \alpha_i^* y_i \varphi(\boldsymbol{x_i}) \cdot \varphi(\boldsymbol{x}) + b* \tag{35}$$

where φ is a function that maps the samples in a higher feature space. Such a function for high dimensions is really complex, and hard to compute. Luckily we don't need to know it, because all we have to do is compute a inner product between the elements in the new feature space. So what we actually need is a function that describes **how to compute the inner product** in the new feature space. These functions are called kernels, and they are sort of easy:

$$\varphi(\boldsymbol{x_i}) \cdot \varphi(\boldsymbol{x_j}) = K(x_i \cdot x_j) \tag{36}$$

One of the most famous and used is the **gaussian kernel**:

$$K(\boldsymbol{x_i} \cdot \boldsymbol{x_j}) = \exp\Big(\frac{-||\boldsymbol{x_i} - \boldsymbol{x_j}||^2}{2\sigma^2}\Big) \tag{37}$$

Note that $||\boldsymbol{x_i} - \boldsymbol{x_j}||^2$ might be also a really high value, depending on the dataset. σ is used to re-scale it in a [0,1] range!

5 Pegasos

Pegasos is a training algorithm that solves the primal formulation of the problem by applying gradient descent. We will call L the "loss" function.

$$min_w L(w) = \frac{\lambda}{2}||w||^2 + \underbrace{\frac{1}{N}\sum_{i=1}^{N} max(0, 1 - y_i\langle w, x_i\rangle)}_{\text{Hinge loss}} \tag{38}$$

- λ is a parameter that controls the tradeoff between maximizing the margin and classifying the points correctly.
- The loss function increases if we missclassify a point, so if the inner product is > 0. If it is negative, the 0 gets taken.

For each step of the gradient descent algorithm we update the weights following this rule:

$$w^{t+1} = w^t - \eta \nabla L(w^t) \tag{39}$$

where η is the lenght of the step size. It decreases at each iteration:

$$\eta^t = \frac{1}{t\lambda} \tag{40}$$

Computing the gradient of the Loss is easy in this case:

$$\nabla L(w^t) = \lambda w^t - \frac{1}{N}\sum_{y_i\langle w^t, x^i\rangle < 1} y_i x_i \tag{41}$$

note that the loss takes into account only missclassified points!