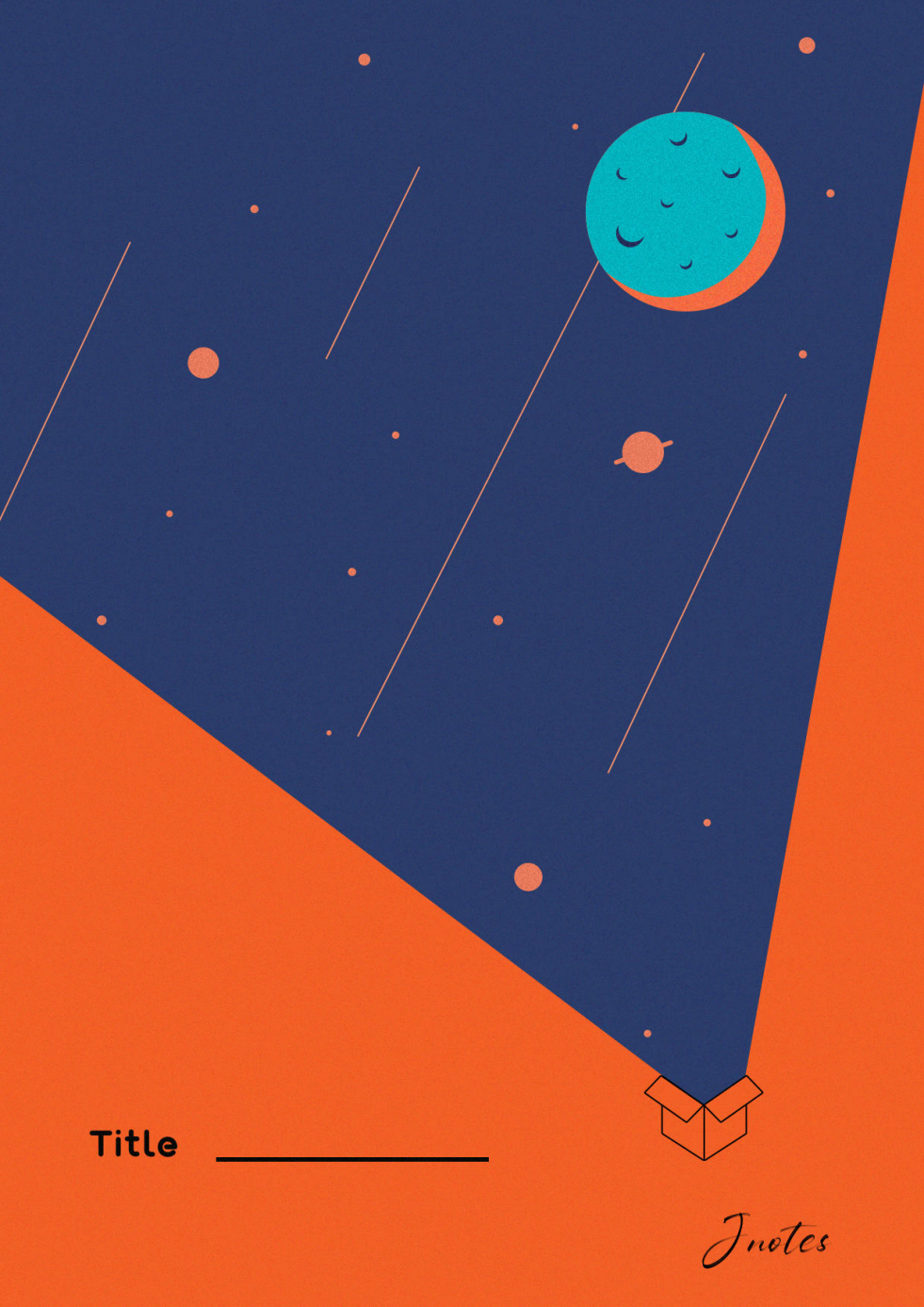


Title _____

Jnotes

Let's consider again a binary classification problem

We have shown that logistic regression provides a linear classification rule that maximizes the log–probability of class assignments $\sum_{i=1}^{n} \log P(C_i = c_i | \boldsymbol{X}_i = \boldsymbol{x}_i, \boldsymbol{w}, b)$ for the training set

The LR solution can be found by minimizing the logistic loss

As we have mentioned, it is often useful to add a regularization term of the form $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$ to prevent overfitting:

$$\boldsymbol{w}^*, b = \arg\min_{\boldsymbol{w},b} \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}\right)$$

where z_i is the class label for sample $\boldsymbol{x}_i$, encoded as

$$z_i = \begin{cases} +1 & \text{if } c_i = \mathcal{H}_T \\ -1 & \text{if } c_i = \mathcal{H}_F \end{cases}$$

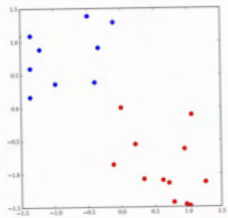
We can alternatively minimize a scaled version of the objective:

$$\boldsymbol{w}^*, b = \arg\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}\right)$$

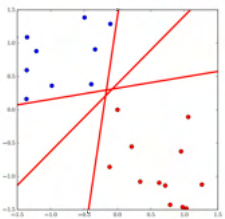
where $C = \frac{1}{n\lambda}$

Logistic regression has a fundamental issue when classes are perfectly separable: the optimization has no true minimum.
For perfectly separate class we could always make our model more confident by scaling up the weight vector $\boldsymbol{w}$. As $\|\boldsymbol{w}\|$ increase the model push class probabilities closer to 0 or 1, continually decreasing the loss function. This means there's an infimum (greatest lower bound) for the loss at $\|\boldsymbol{w}\| \to \infty$ but no actual minimum value.

Assume that we have two classes that are linearly separable (i.e., we can find a linear separation hyperplane that correctly classifies all points of our training set)

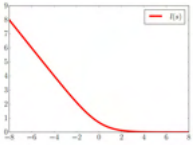


Problem: there is an infinite number of separating hyperplanes, which one should we select?



An optimal LR solution is not defined, since the objective function has an infimum for $\|w\|^2 \to +\infty$, but not a minimum

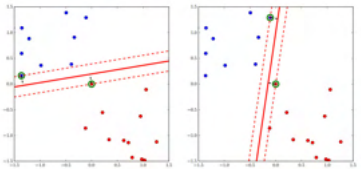
LR optimizes the Logistic Loss. As we move further along the x axis the loss tends to 0



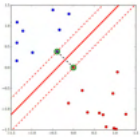
When multiple hyperplanes could separate our classes, SVMs answer the question. "Which hyperplane is optimal?"

The core insight is to select the hyperplane that creates the maximum margin between classes. The margin is define as the distance from the decision boundary to the nearest data point from either class.

Intuitively, we can select the hyperplane that separates the classes with the largest margin



Intuitively, we can select the hyperplane that separates the classes with the largest margin



The margin is defined as the distance of the closest point w.r.t. the separation hyperplane[1]

[1] Some authors separately introduce the margin for each class. This distinction has no practical effects on the derivation of the SVM objective.

Starting with our linear classifier function:

$$f(x) = w^T x + b$$

This function represent our separating hyperplane, where point with $f(x)=0$ lie directly on the boundary.

For any point in our feature space, we want to know how far it is from our decision boundary.

The distance from a point x_i to a hyperplane is given by:

$$d(x_i) = \frac{|f(x_i)|}{\|w\|}$$

The numerator $|f(x_i)|$ tells us how strongly our function classifies the point while dividing by $\|w\|$ normalize this value to give us the actual perpendicular distance.

$$s(x) = w^T x + b \begin{cases} =0 \\ <0 \\ >0 \end{cases}$$

Why this formula??

we want D, the distance

$P_2(x_2, y_2, z_2)$

$P_0(x_0, y_0, z_0)$

$$D = \overrightarrow{P_0P_2} = \langle x_2 - x_0, y_2 - y_0, z_2 - z_0 \rangle$$

n is the normal vector (w), in order to find D, we want the scalar projection of D on n

$$D = \underset{\text{componente di } D \text{ lungo } n}{\text{comp}_n B} = |B| \cos\theta = \frac{|n \cdot B|}{\|n\|}$$

(prodotto scalare)

$$\frac{a \cdot b}{|a|} = \frac{|a||b|\cos\theta}{|a|}$$

$n = \langle a, b, c \rangle$

$$D = \frac{|n \cdot B|}{\|n\|} = \frac{|a(x_2 - x_0) + b(y_2 - y_0) + c(z_2 - z_0)|}{\sqrt{a^2 + b^2 + c^2}} =$$

$$= \frac{|(ax_2 + by_2 + cz_2) - (ax_0 + by_0 + cz_0)|}{\sqrt{a^2 + b^2 + c^2}} = \frac{|ax_2 + by_2 + cz_2 + d|}{\|n\|} = \frac{|f(P_2)|}{\|n\|}$$

the equation of a plane is $ax + by + cz = -d$

$$\Downarrow$$

$$f(x) = ax + by + cz + d$$

Since we are working with binary classifications, we use z_i to denote the class label of point x_i where:

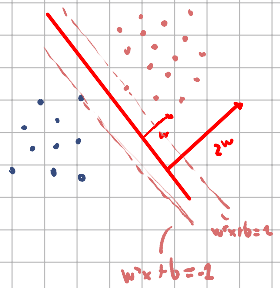
- $z_i = +1$ for the positive class
- $z_i = -1$ for the negative class

For a correctly classified point the sign of $f(x_i)$ will match the sign of z_i. This mean $z_i f(x_i) > 0$ for all correctly classified points.

Using this property we can rewrite the distance formula as:

$$d(x_i) = \frac{|z_i (w^T x_i + b)|}{\|w\|}$$

Visualizing how we can simplify the problem !!



(w, b) The hyperplane with (w,b) and $(2w,2b)$ is the same. When I'am
$(2w, 2b)$ thinking of a separation rule, the same rule can be represented in infinite way $(2w, 2b)$ with one I should select?

I will select only those planes (feasible plans that correct separate the point) for with the closest points to the plain has a score of $(w^T x + b = \pm 1)$

The w^* & b^* that hold ↑ this will be consider an optimal solution

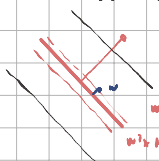
If the support vectors have this constraint $w^T x + b = \pm 1$ then

$\forall i \quad (w^T x_i + b) \geq 1 \quad z_i = 1$

$(w^T x_i + b) \leq -1 \quad z_i = -1$

With this limitation we can see that the range of the margin will be proportional to $\|w\|$

if we have $\|w\|$ is large then x ($xw^T + b$) need to change a bit in order to obtain a score of 1 so we will consider a point close to the surface ~



(hyperplane)

if $\|w\|$ is small, we will need point that are far from the surface. to obtain a score $|g(x)|$ of 1.

So $\|w\|$ to have a large margin we want a low $\|w\|$ or $\frac{1}{2}\|w\|^2$

Formal way

The goal of svm is to find the hyperplane that maximizes the minimum distance to any training point.

we select only the support vector

$$w^*, b^* = \arg\max_{w,b} \min_{i \in \{1,\dots,n\}} d(x_i)$$

Subject to the constraint that all point are correctly classified:

$$z_i (w^T x_i + b) > 0 \quad \forall i$$

We can do some semplifications (for semplify the problem, keeping the same solution)

① First we can remove the absolute value thanks to the constraint $z_i (w^T x_i + b) > 0 \ \forall i$

the class are linear separable, so for sure the minimum of the distance will be positive

② We can rewrite our optimization problem as:

$$w^*, b^* = \arg\max_{w,b} \frac{1}{\|w\|} \min_{i \in \{1,\dots,n\}} [z_i (w^T x_i + b)]$$

We are looking for the hyperplane that maximize the margin (represented by the factor $\frac{1}{\|w\|}$) while ensuring all points are correctly classified with some positive confidence value represented by $\min z_i (w^T x_i + b)$

Let's image an

We can show the two problems are equivalent:

- Hyperplanes that do not satisfy this constraint will have $\min_i z_i (w^T x_i + b) \not> 0$, and thus cannot be an optimal solution of the second formulation (classes are separable, so an optimal solution exists with $z_i (w^T x_i + b) > 0 \ , \forall i = 1 \dots n$)
- For hyperplanes that correctly classify all patterns, the objective functions are the same

The svm

Focus on boundary points: The minimum distance is determined by the closest points to the boundary (the support vectors), making the solution sparse.

Scale invariance: The decision boundary depends only on the direction of ww w, not its magnitude, providing a unique solution unlike logistic regression.

Robustness: By maximizing the minimum distance, SVMs create the most "cautious" boundary possible, increasing resilience to small perturbations in the data.

We need to transform our function in something more tractable

Step 1

Our objective function has a special property called <u>scale invariance</u>. This mean that if we multiply both w and b by any positive constant a the decision boundary remains the same

$$\frac{1}{\|w\|}\min_i\left[z_i(w^T x_i+b)\right]=\frac{1}{\|aw\|}\min_i\left[z_i(aw^T x_i+ab)\right]$$

Since $\|aw\| = a\|w\|$ we get

$$\frac{1}{a\|w\|}\min_i a\,z_i(w^T x_i+b)\right]$$

This mean that (w,b) and (aw,ab) give us the same objective value for any $a>0$. They represent the same hyperplane but with different scaling of the parameters.

$y=2x+b$ and $2y=4x+2b$ describe exactly the same line

if (w^*,b^*) is an optimal solution, then also $(w',b')=(aw^*,ab^*)$ is optimal e viceversa.

The collections of values $\{(aw,ab)\}_{a\in\mathbb{R}_+}$ forms an equivalent class of equivalent solutions.

For each of these equivalence classes, we are free to select any one of the equivalent solutions. In particular, we restrict our problem to solutions for which

$$\min_i z_i(w^T x_i+b)=1$$

we choose w, b that allow to have this constraint

This mean that for the point(s) closest to the decision boundary the <u>function marginal</u> is equal to 1. All other point will have function marginal greater or equal to 1

$$z_i(w^T x_i+b)\geq 1,\quad i=1,\dots,n$$

With this new constraint (the function has the same result as before) the objective function became

$$\arg\max_{w,b}\frac{1}{\|w\|}$$

$$\text{s.t.}\begin{cases} z_i(w^T x_i+b)\geq 1, & i=1,\dots,n \\ \min_i z_i(w^T x_i+b)=1 \end{cases}$$

the standard distance from the closest point became $\frac{1}{\|w\|}$

we still want to maximize this

This is equivalent to:

$$\arg\min_{w,b}\|w\|$$

or for more practical way

$$\arg\min_{w,b}\frac{1}{2}\|w\|^2 \quad (2)$$

[the function is <u>scale invariance</u>]

Minimizing $\|w\|$ and minimizing $\|w\|^2$ give exactly the same result because the square function is monotonically increasing for positive value. If $\|w\|<\|w'\|$, then $\|w\|^2<\|w'\|^2$

A Road Between Two Towns Analogy

Imagine you're designing a straight road between two towns. Each town has houses scattered around (these are your data points).

Town A represents the positive class (+1)
Town B represents the negative class (-1)

Your goal is to build a straight road (your decision boundary) that completely separates the two towns, with no houses from either town ending up on the wrong side of the road. ESISTE x IPOTESI UN HYPERPLANE CHE PERMETTE CIÒ
Now, for safety reasons, you want this road to be as far away as possible from the houses in both towns. In other words, you want to maximize the distance from the road to the closest house from either town.
The Road Design Problem
Let's say the position and direction of your road is defined by the equation w^T x + b = 0.

The vector w determines the direction of the road
The term b shifts the road's position

The distance from any house to the road is given by:
|w^T x + b| / w
To maximize the safety buffer (margin) between your road and the closest houses, you need to:

Find a road that completely separates the towns (all houses are on the correct side)
Maximize the distance to the closest house

Why We Minimize w^2

Here's where mathematics helps us solve this problem efficiently:

scale vector – VALORE MINIMO

First, we standardize our problem: we scale our road equation so that for the closest house, the value $|w^T x_i + b| = 1$.
With this standardization, the actual distance from the closest house to the road becomes 1/w.
To maximize this distance (1/w), we need to minimize w.
For mathematical convenience, we minimize $(1/2)w^2$ instead of just w, which gives us the same solution but with cleaner mathematics.

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \begin{cases} z_i(w^T x_i + b) \geq 1, & i = 1 \dots n \\ \min_i z_i(w^T x_i + b) = 1 \end{cases}$$

(second constraint struck through) QUESTO DOVREBBE È IMPLICITO NEL PIANO

Why we can remove that constraint?

Indeed, an optimal solution of (2) will automatically satisfy $\min_i z_i(w^T x_i + b) = 1$

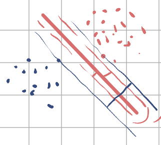
~~In fact, if~~ w^*, b^* was a solution for which $\min_i z_i(w^T x_i + b) = \psi > 1$, then we could build the solution $(w', b') = \left(\frac{w}{\psi}, \frac{b}{\psi}\right)$ which would still satisfy all constraints, and have a lower norm, thus w^*, b^* would not be optimal — w^* was not true minimum

Let's imagin w^*, b^* are optimal solution for this problem $\left[\arg\min_{w,b} \frac{1}{2}\|w\|^2 \text{ s.t. } z_i(w^T x_i + b) \geq 1\right.$ that does not respect $\min(z_i(w^T x_i + b)) = 1$ – QUINDI IL MINIMO NON È 1 È x MINIMO ψ

$\min_i z_i(w^T x_i + b) = \psi$

$\min_i z_i\left(\frac{w^T}{\psi} x_i + \frac{b}{\psi}\right) = 1$

QUESTO HA w PIÙ BASSO AND STILL SATISFIES THE CONSTRAINT ≥1 – È MINIMO =1 – QUINDI IL PUNTO PIÙ VICINO ALLA DECISION RULE DEVE AVERE SCORE PARI A 1



This is not the optimal solution, I can find one that has higher margin – so it will always touch the margin region

$$\begin{cases} \arg\min_{w,b} \frac{1}{2}\|w\|^2 \\ \text{s.t. } z_i(w^T x_i + b) \geq 1, \quad i = 1, \dots n \end{cases}$$

This is a convex quadratic programmin problem, which means:
- The objective function is a convex (it's a simple quadratic function of w) ✓
- The constraints form a convex set

[There is no local minimum but only one global minimum]

Note:
A set is convex if for any two points in the set, the entire line segment connecting those points also lines within the set. Visually this mean the set has no "dents" or "cavities" - if you can draw a straight line between any two points in the set, that entire line must be also in the set.

 is not a convex set

A half-space is one of the simplest examples of a convex set. it's the set of all points on one side of the hyperplane (a straight line in 2D, a flat plane in 3D...)
in SVM each constraint is of the form

$$z_i(w^T x_i + b) \geq 1$$

For a specific data point (x_i, z_i) this constraint defines a half-space in the parameter space of (w, b)

So to solve the problem we wanted to minimize $\frac{1}{2}\|w\|^2$ (to maximize the margin) but we had to ensure that all data points were correctly classified with a sufficient margin $(z_i(w^T x_i + b) \geq 1)$

This is what's known as a constrained optimization problem. We are trying to optimize one function while respecting certain boundaries or rules given by another set of functions.

We want to find the lowest point in a hill but we are allowed to walk only on certain paths

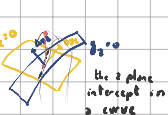
The method of **Lagrange multipliers** gives us an optimal way to convert this constrained problem in an unconstrained ones.

Note: Lagrange Multiplier with multiple constraint

Optimize: $f(x_1, x_2, z)$

Constraints: $g_1(x_1, x_2, z) = 0$

$g_2(x_1, x_2, z) = 0$



$\Delta f = \lambda \Delta g_1 + \mu \Delta g_2$ this intersection curve is where the two constraint are true

$g_1 = 0$

$g_2 = 0$

① For each constraint we introduce, a new variable called a Lagrange multiplier (denoted by α_i), $\alpha_i \geq 0$

② We create a new function called the Lagrangian that combines our original objective function with the constraints, weighted by these multipliers.

For our SVM problem, the Lagrangian is: (minus or surplus)

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

the minus comes from how we formulate the constraint (following the Lagrangian theory)

- For $g_i(w, b) \geq 0$ we use $-\alpha_i g_i(w, b)$
- For $g_i(w, b) = 0$ we use $\beta_i g_i(w, b)$ (no minus sign)

Our constraints are $z_i(w^T x_i + b) - 1 \geq 0$

Now, there are two different approaches to solve the problem that is finding the saddle point of $L(w, b, \alpha)$

① Minimize L with respect to the primal variables w and b, while handling the Lagrange multipliers α_i in a specif way:

For each α_i (for each point) one of two conditions must be true:

- The derivative of L with respect to α_i equals zero
- Or multiplier its self $\alpha_i = 0$

When $\frac{\partial L}{\partial \alpha_i} = 0$ $z_i(w^T x_i + b) - 1 = 0$, the datapoint x_i lies exactly on the margin boundary, they are support vector

When $\alpha_i = 0$, it means this data point does not influence the final decision boundary. it's doesn't matter for our solution, we don't need it. is not a support vector NON CONTRIBUISCE ALLA DEFINIZIONE DELLA SOLUZIONE OTTIMALE

(2) We first ensure that the derivative of L with respect to w and b are zero then maximize L with respect to the Lagrange multipliers α_i;

When L is expressed in function of α_i thanks to the value obtained from $\nabla_w L(w,b,\alpha)$ and $\nabla_b L(w,b,\alpha)$

We will use the second approach:
Let's start by finding the minimum with respect to w and b by setting the derivatives equal to zero.

- $\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^{n} \alpha_i z_i x_i = 0$

  $w = \sum_{i=1}^{n} \alpha_i z_i x_i$

- $\frac{\partial L(w,b,\alpha)}{\partial b} = \sum_{i=1}^{n} \alpha_i z_i = 0$

We substitute this result back into the Lagrangian to eliminate the primal variables w,b => We obtain a dual objective function

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i + \frac{1}{2} \left\| \sum_{i=1}^{n} \alpha_i z_i x_i \right\|^2$$

Let's focus on the second term

$\frac{1}{2}\|w\|^2 = \frac{1}{2} w^T w$

$$\frac{1}{2} \left\| \sum_{i=1}^{n} \alpha_i z_i x_i \right\|^2 = \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i z_i x_i \right)^T \left(\sum_{j=1}^{n} \alpha_j z_j x_j \right) =$$ (sum)

$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i z_i x_i^T \alpha_j z_j x_j = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j$$ (swap)

$$L_D(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j z_i z_j x_i^T x_j$$

↳ an optimal solution can be expressed as a linear combination of the training set sample

s.t

$\alpha_i \geq 0$

$\sum_{i=1}^{l} \alpha_i z_i = 0$

in matrix form

$$L_D(\alpha) = \alpha^T 1 - \frac{1}{2} \alpha^T H \alpha \qquad , 1 = [1, 1, \dots 1]$$

where H is the matrix

$$H_{ij} = z_i z_j x_i^T x_j \qquad , \alpha^T H \alpha = \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i H_{ij} \alpha_j = \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j z_i z_j x_i^T x_j$$

[We want to find $\max_{\alpha} L_D(\alpha)$]

Note that the actual form of the primal on the data only through dot products $x_i^T x_j$. This allow to extend SVMs to non-linear classification, we can replace these dot with more complex Kernel function without changing the structure of the optimization problem

Why we want to $\max_{\alpha}$??

↓

is a Lagrange rule for finding the optimal solution [if you want you can see the L_D as UN LIMITE INFERIORE ALL'OBIETTIVO PRIMALE (poiché la massima distanza) Massimizzando lo stiamo trovando il miglior LIMITE INFERIORE POSSIBILE, che coincide con il valore minimo della funzione primale ($\min_{w,b} \|w\|^2$) è come ruotare nello zero

α inoltre ($\alpha_i \geq 0$) serve ad identificare i vettori di supporto, e definirà quando ogni vettore di supporto contribuisce alla decisione finale. (alcuni punti x cattivi hanno α_i grandi quelli di cui ce ne dobbiamo il caso hanno $\alpha = 0$ e come da formula non contribuiscono al calcolo del $(w^* b^*)$ soluzione)

Poi se vedi primo $L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i (z_i (w^T x_i + b) - 1)$ non che i moltiplier di lagrange funzione da limitazioni, sono noi x far rispettare i constraint (di cui l'ottimale si ha per il loro massimo).

When we $\max_{\alpha} L(\alpha)$ we found $\alpha_i \; \forall i$ i è pari al numero di sample

Summarize we have for svm two problem

① **Primal Problem**: Finding w and b that minimize $L_p(w,b) = \frac{1}{2}\|w\|^2$, with constraint $z_i(w^T x_i + b) \geq 1$ for all training points.

② **Dual Problem**: Find α that maximize $L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j$, with constraint $\alpha_i \geq 0$ for all i and $\sum_{i=1}^{n} \alpha_i z_i = 0$

The dual and primal formulations are connected through the duality gap.

$$\text{Duality Gap} = L_p(w,b) - L_D(\alpha)$$

The difference between the primal objective value and the dual objective value

For any feasible w, b (meaning they satisfy the primal constraint) and any feasible α meaning they satisfy the dual constraint, we always have:

$$L_D(\alpha) \leq L_p(w,b)$$

Th

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(z_i(w^T x_i + b) - 1\right)$$

So for any feasible primal solution (w,b) and fasible α

$$\Downarrow$$

The term $\sum_{i=1}^{n} \alpha_i \left(z_i(w^T x_i + b) - 1\right)$ is always positive

$$\Downarrow \text{This mean}$$

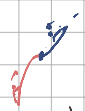
$$L(w,b,\alpha) \leq \frac{1}{2}\|w\|^2 = L_p(w,b)$$

funthermore the $L_D(\alpha)$ is the min $L(w,b,\alpha)$ so the α fesible generate a value for sure

$$L_D(\alpha) \leq L(w,b,\alpha) \leq \frac{1}{2}\|w\|^2$$

For convex optimization problems like svms, when both primal and dual problem reach their optimal solutions, the duality gap completly close.

$$L_D(\alpha^*) = L_p(w^*, b^*)$$



This is called "strong duality" and it means that solving either the primal or the dual problem will give us the same optimal value.

Observation 🔎

The primal objective $\frac{1}{2}\|w\|^2$ is related to the margin of the classifier. Specially the geometric margin is $\frac{1}{\|w\|}$, so minimizing $\|w\|^2$ is equivalent to maximizing the margin.

The dual objective, $\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j$ can be interpretate as finding the most important point (support vector) that define the margin.

The duality gap represents how far our current margin estimate (from the dual problem) is from the true maximum margin (in the primal problem). When the gap is zero we have identified exactly the right set of support vectors that define the maximum margin hyperplane.

- For a solution to be optimal, it must satisfy specific conditions called the Karush-Kuhn-Tucker (KKT) conditions. These aren't arbitrary rules- they're the mathematical requirements that any optimal solution must fulfill

The KKT conditions provide the bridge between the primal and the dual formulations. They are both necessary and sufficient conditions for optimality in convex problems like SVMs.

Karush–Kuhn–Tucker (KKT) conditions[2]

$$\nabla_w L(w,b,\alpha) = w - \sum_{i=1}^{n} \alpha_i z_i x_i = 0$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = -\sum_{i=1}^{n} \alpha_i z_i = 0$$

$$z_i\left(w^T x_i + b\right) - 1 \geq 0 \quad \forall i$$

$$\alpha_i \geq 0 \quad \forall i$$

$$\alpha_i\left[z_i\left(w^T x_i + b\right) - 1\right] = 0 \quad \forall i$$

Complementary slackness

[2]These are both necessary and sufficient conditions for optimality of the SVM solution

23

The first and second equations encode that, at the optimal primal solution (w, b), the gradient of the Lagrangian with respect to w and b becomes zero.

The next two equations encode that both the primal solution (w, b) and the corresponding dual solution α are feasible.

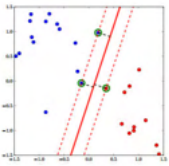
Complementary Slackness

$\alpha_i[z_i(w^T x_i + b) - 1] = 0$ [related to the first way of solving the problem]

For each datapoint, one of two things must be true:

- Either $\alpha_i = 0$
- Or $z_i(w^T x_i + b) = 0$ (the point lies exactly on the margin.

- For all points that do not lie on the margin (i.e. $z_i(w^T x_i + b) > 1$) the corresponding Lagrange multiplier is $\alpha_i = 0$
- $\alpha_i \neq 0$ implies that the corresponding point is on the margin, i.e., it is a Support Vector
- After we have obtained α, the KKT conditions allow us to estimate b



This means that that in svm the decision boundary depends only on the support vectors.
This is significant:

① When making predictions we only need to compute the dot products between the test point and the support vector not all (the training data points):

$$s(x_t) = w^T x_t + b = \sum_{\substack{i=1 \\ i \in SV}}^{n} \alpha_i z_i x_i^T x_t + b$$

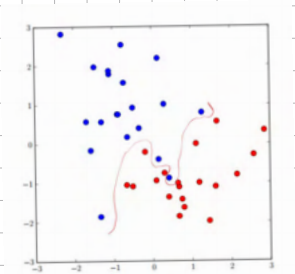
SV is the set of support vector ($\alpha_i > 0$)

The model focus only on the most informative points making it less sensitive to outliers

To make a predictions if $s(x_t) > 0$, the point belong to the positive class (H_1), if $s(x_t) < 0$ it will be classified as belonging to the H_2 class.

Non linearly separable data

In many practical applications, dataset aren't perfectly separable by a linear boundary. No matter how we position our hyperplane, some point will inevitably been in the wrong side or within the margin



So No matter the value of w, some points will violate the constraint $z_i(w^T x_i + b) \geq 1$

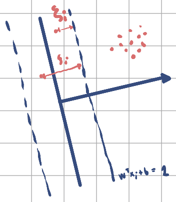
To handle this situation we introduce what are called "slack variables" ($\xi_i \geq 0$) for each training point. These variable measure how much a particular point violates the margin constraint.

We relax our origina constraint into

$$z_i(w^T x_i + b) \geq 1 - \xi_i \quad , \quad \xi_i \geq 0$$

Geometrically:

- if $\xi_i = 0$ the point satisfies the original constraint, and is on the correct side of the margin.
- if $0 < \xi_i < 1$ the point is on the correct side of the decision boundary but inside the margin
- if $\xi_i = 1$ The point is exactly on the decision boundary.
- if $\xi_i > 1$ The point is on the wrong side of the decision boundary (misclassified)



The point can be ξ_i inside the region

I want to keep the point that violate the constraint as few as possible for witch ($\xi_i > 0$)

we want to find an hyperplane that does this

Ideally we want to minimize the number of points that violate the margin constraints, that is points with $\xi_i > 0$. This can be formulated mathematically as minimizing:

$$\sum_{i=0}^{n} I(\xi_i > 0)$$

Where I is the indicator function that equals 1 if its argument is true and 0 otherwise

We will use

$$\Phi(\xi) = \sum_{i=1}^{n} \xi_i^{\sigma} \quad , \sigma > 0$$

For very small values of σ, this function approximate the count of non zero slack variables, effectively counting how many points violate the margin constraints.

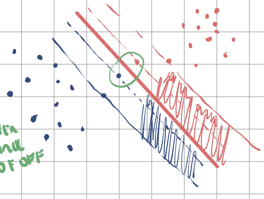
If we remove these data points the classes become linearly separable and we can thus train a maximum margin hyperplane over the remaining points

the one beyond the margin

Now combining our goal of maximizing the margin with minimizing constraint violations, we get the soft margin svm objective:

$$\min_{w, b, \xi} \frac{1}{2}\|w\|^2 + C \cdot F\left(\sum_{i=1}^{n} \xi_i^{\sigma}\right)$$

IT'S AN HYPERPARAMETER THAT CONTROL THIS TRADE OFF



IF NOW PERMETTO AI PUNTI DI STARE NEL MARGINE HO UN SMALL MARGIN

SE PERMETTO AI PUNTI DI ESSERE NEL MARGINE AVRÒ UN MARGINE + AMPIO MA MINOR PRECISIONE

THAT'S LIKE CREATING AND IMPROVE GENERALIZATION

Subject to: (s.t)

- $z_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i$
- $\xi_i \geq 0 \quad \forall i$

Where:

- F is a monotone convex function
- C is a positive constant that controls the trade-off between margin maximization and constraint violation minimization

in fact:

- A large C heavily penalizes constraint violations, leading to fewer margin violations but potentially smaller margin
- A small C allows more margin violations in exchange for a potentially larger margin.

Note: Why C?

C È UN MOLTIPLICATORE DI LAGRANGE, DIFATTI QUANDO TRASFORMIAMO UN PROBLEMA VINCOLATO IN UNO NON VINCOLATO, AGGIUNGIAMO DEI TERMINI DI PENALITÀ MOLTIPLICATI PER COSTANTI. IL PARAMETRO C È PROPRIO QUESTA COSTANTE CHE DETERMINA QUANTO PESO DARE ALLA PENALITÀ PER LE VIOLAZIONI RISPETTO ALLA MASSIMIZZAZIONE DEL MARGINE.

QUESTO IPERPARAMETRO È SOLITO SCELTO MEDIANTE CROSS VALIDATION.

Note: Why F?

LA FUNZIONE F È INTRODOTTA CON UNA FUNZIONE CONVESSA MONOTONA CHE OPERA SULLA SOMMA DELLE VARIABILI DI SLACK (ξ_i) ELEVATE ALLA POTENZA 6. This function is a way to approximate IL CONTEGGIO DELLE VIOLAZIONI DEL MARGINE.

IN TEORIA CON QUESTO VOLEMMO MINIMIZZARE ESATTAMENTE IL NUMERO DI PUNTI PER CUI $\xi_i > 0$ (CHE VIOLANO IL MARGINE). PALESI SI USA I IN QUANTO NON È DIFFERENZIABILE (è convessa) E DA PROBLEMI A LIVELLO DI OTTIMIZZAZIONE

└ F È UN APPROSSIMAZIONE TRATTABILE DI QUESTA FUNZIONE DI CONTEGGIO

While the general formulation with arbitrary 6 and function F provide theoretical flexibility, it leads to difficult optimization problem (for ξ_i near 0 the problem become will be no more convex for $\xi_i > 0$)

So the standar semplification uses:

- $6 = 2$ (linear penalty on slack variables)
- $F(u) = u$ (identity function)

This gives us the practical soft margin SVM objective:

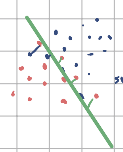
$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

I want to minimize how much on average a point is on the margin region

$$s.t.$$

$$z_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

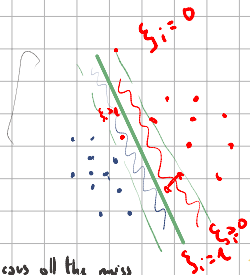


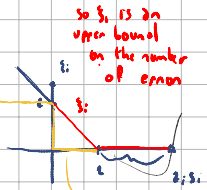
This is a convex quadratic programming problem

The terms ξ_i do not describe the number of errors, but act as a penalty

- Points inside the margin have $\xi_i > 0$, and miss–classified points have $\xi_i > 1$
- The further the point is from the hyperplane, the larger the value of ξ_i
- $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of miss–classified points (errors)
- C allows selecting a trade–off between margin and errors on the training set
- The solution is often called a *soft margin* hyperplane

because all the miss classificid point have $\xi_i \geq 1$

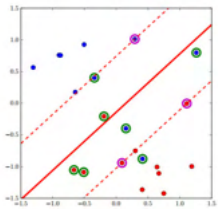




So ξ_i is an upper bound on the number of error

Function that count the number of error

Soft margin classification:



As we have seen with the hard-margin svm, we can solve the soft-margin svm using lagrange multipliers. We will then find the partial derivative of $L(w,b,\xi,\lambda,\mu)$ for w,b,ξ_i then substitute the value to obtain $L_0(\lambda)$ and finally $\max_\lambda L_0(\lambda)$

We have two set of Lagrange multiplier

- $\lambda_i \geq 0$ for the constraint $z_i(w^T x_i + b) = 1 - \xi_i$
- $\mu_i \geq 0$ " " $\xi_i \geq 0$

$$L(w,b,\xi,\lambda,\mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=0}^{n}\xi_i - \sum_{i=0}^{n}\lambda_i[z_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=0}^{n}\mu_i\xi_i$$

The KKT conditions are

The necessary and sufficient KKT conditions are

$$\nabla_w L(w,b,\alpha,\mu) = w - \sum_{i=1}^{n}\alpha_i z_i x_i = 0$$

$$\frac{\partial L(w,b,\alpha,\mu)}{\partial b} = -\sum_{i=1}^{n}\alpha_i z_i = 0$$

$$\frac{\partial L(w,b,\alpha,\mu)}{\partial \xi_i} = C - \alpha_i - \mu_i = 0 \quad \forall i$$

$$z_i(w^T x_i + b) - 1 + \xi_i \geq 0 \quad \forall i$$

$$\xi_i \geq 0 \quad \forall i$$

$$\alpha_i \geq 0 \quad \forall i$$

$$\mu_i \geq 0 \quad \forall i$$

$$\alpha_i\left[z_i\left(w^T x_i + b\right) - 1 + \xi_i\right] = 0 \quad \forall i$$

$$\mu_i\xi_i = 0 \quad \forall i$$

Non zero α_i tell us what vector are support vector

if $\lambda_i \neq 0$ then

$z_i(w^T x_i + b) - 1 + \xi_i = 0$

so we have two option

$\lambda_i \neq 0 \longrightarrow \lambda_i < C \Rightarrow \mu_i > 0 \Rightarrow \xi_i = 0$ ⇓ $z_i(w^T x_i + b) = 1$

$\lambda_i = C$ ⇓ $\mu_i = 0 \Rightarrow \xi_i \geq 0$

From the KKT conditions (1)–(3)

$$w = \sum_{i=1}^{n} \alpha_i z_i x_i \,, \qquad \sum_{i=1}^{n} \alpha_i z_i = 0 \,, \qquad \alpha_i = C - \mu_i$$

Note that $\mu_i \geq 0$ implies that $\alpha_i \leq C$

We can then optimize the Lagrangian w.r.t. w, b and ξ to obtain the dual problem

$$L_D(\alpha) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i \xi_i + \sum_{i=1}^{n} \mu_i \xi_i + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \xi_i - \sum_{i=1}^{n} \mu_i \xi_i$$

$$- \sum_{i=1}^{n} \alpha_i z_i \left(\sum_{j=1}^{n} \alpha_j z_j x_j \right)^T x_i =$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j$$

So the Dual problem is:

$$\max_{\alpha} L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j z_i z_j x_i^T x_j$$

s.t

$$0 \leq \alpha_i \leq C \quad i = 1, \dots n$$

$$\sum_{i=1}^{n} \alpha_i z_i = 0$$

The only difference with the hard-margin is the upper bound on the Lagrange multipliers $\alpha_i \leq C$. The constraint on α_i, in the soft-margin case have important implications.

① if $\alpha_i = 0$ the point is correctly classified and outside the margin. it doesn't influence the decision boundary

② if $0 < \alpha_i < C$ For complementary slackness, we know that:

- $\alpha_i (w^T x_i + b) - 1 + \xi_i = 0$ (the point lies exactly on the margin)
- $\mu_i \xi_i = 0$ and since $\mu_i = C - \alpha_i > 0$ we must have $\xi_i = 0$

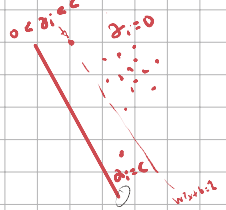
This point are the **support vector**

③ if $\alpha_i = C$ The point either:

- Inside the margin but correctly classified ($0 < \xi_i < 1$)
- Exactly on the decision boundary ($\xi_i = 1$) ($w^T x + b = 0$)
- Misclassified ($\xi_i > 1$)

These are also support vector but they are bounded or stored at the upper limit.

inside the margin region or misclassified



Once we have solved for the optimal α values, we make prediction using the score.

$$s(x_t) = w^T x_t + b = \sum_{i=1}^{n} \alpha_i z_i x_i^T x_t + b$$

the boundary depends only on the support vector

Again depends only on the dot products $x_i^T x_k$

Several methods to solve the *dual* problem

Packages are available for several solvers

If we are interested in linear separation we can directly solve the *primal* SVM problem

Let's recall the primal formulation

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

we use only the primal formulation

s.t.

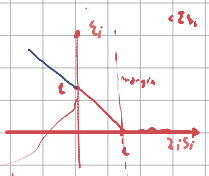
$$z_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

We can observe that for any data point, the optimal value of the slack variable ξ_i is:

- $\xi_i = 0$ if the point is correctly classified and outside the margin
- $\xi_i = 1 - z_i(w^T x_i + b)$ if the point violates the margin constraint

Putting these two cases together, for each data point, the optimal value of ξ_i is:

$$\xi_i = \max(0, 1 - z_i(w^T x_i + b))$$



cost that increases linearly in that way

The objective function then becomes:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\max[0, 1 - z_i(w^T x_i + b)]$$

This is an unconstrained optimization problem! We've effectively absorbed the constraint into the objective function through the max function, which is precisely the hinge loss

$$\ell(s) = \max(0, 1 - s)$$ cost that I pay for those point inside of the margin

$$\ell(s) = [1 - s]$$

The hinge loss has a clear geometric interpretation:

- When a point is correctly classified outside the margin ($z_i f(x_i) > 1$) the loss is 0. The point doesn't contribute to the objective function.
- When a point is inside the margin or misclassified ($z_i f(x_i) < 1$), the loss grows linearly with the distance from the margin boundary. The further a point is on the wrong side, the more it "pulls" on the decision boundary.

Only the point that are close to or on the wrong side of the margin exert (esercitano) influence.

The hinge loss formulation reveals that SVMs fit into the broader framework of regularized empirical risk minimization:

$$\min_{w,b} \quad \underbrace{\frac{\lambda}{2}\|w\|^2}_{\text{Regularization term}} + \underbrace{\frac{1}{n}\sum_{i=1}^{n} \ell(z_i, f(x_i))}_{\text{Empirical risk}}$$

Note C is obtained by cross validation I can multiplie for λ in some way

Where $\ell(z_i, f(x_i)) = \max(0, 1 - z_i f(x_i))$ is the hinge loss

- The first term regularize the complexity of the model (favoring large margin)
- The second term ensure the model fit the training data.

$$\min_{w,b} \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max\left[0, 1 - z_i(w^T x_i + b)\right]$$

Compare with Logistic Regression:

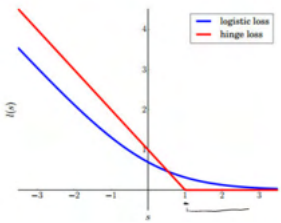
$$\min_{w,b} \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i} \log\left[1 + e^{-z_i(w^T x_i + b)}\right]$$

LR: Logistic Loss

$l(s) = \log\left(1 + e^{-s}\right)$

SVM: Hinge Loss

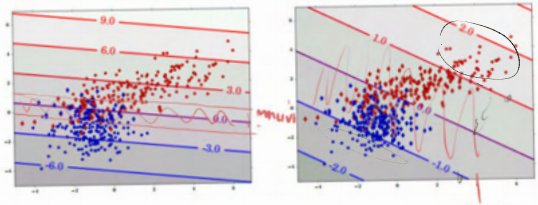
$l(s) = \max(0, 1 - s)$



Binary classification:

$C = 1.0$ ($\lambda = 0.0025$)

$C = 0.001$ ($\lambda = 2.5$)



we are trying to minimize the number of errors

One of the most powerfull aspect of the support vector machines it their ability to create non-linear decision boundaries through what's called "Kernel trick".

We have seen that we can obtain the separation hyperplane by solving either the primal or the dual problem.

Primal Formulation

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max[0, 1 - z_i(w^T x_i + b)]$$

Dual Formulation

$$\max_{\gamma} \sum_{i=1}^{n} \gamma_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \gamma_i \gamma_j z_i z_j x_i^T x_j =$$

$$= \gamma^T 1 - \frac{1}{2}\gamma^T H \gamma$$

s.t.

$0 \le \gamma_i \le C \quad \forall i$

$\sum \gamma_i z_i = 0 \quad \forall i$

At the optimal solution this formulation yield the same result, with the duality gap closing to 0

$$L_p(w^*, b^*) = L_D(\gamma^*)$$

Parameters $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$ (where D is the feature dimension)

Making prediction $s(x_t) = w^T x_t + b$ with complexity $O(D)$

Embedding a non-linear transformation requires explicitly defining the mapping from $\mathbb{R}^D$ to the Hilbert space $\mathcal{H}$ — EXPLICIT FEATURE

$$\Phi : \mathbb{R}^D \longrightarrow \mathcal{H}$$

$w^T \phi(x) + b$ as logistic regression — common use

we train the model in the expanded feature function

Parameters $\gamma \in \mathbb{R}^N$ (where N is the number of training samples)

Making predictions:

$$s(x_t) = \sum_{i=1}^{N} \gamma_i z_i x_i^T x_t + b$$

$$= \sum_{i:\gamma_i > 0} \gamma_i z_i x_i^T x_t + b$$

with complexity $O(sv)$, where sv is the number of support vector

Embedding a non-linear transformation only requires dot-products in the expanded space

$$\Phi(x_i)^T \Phi(x_j)$$

The fundamental observation is that in the dual formulation, both training and prediction only required to compute the dot products between datapoints. We never need the transformed features explicitly - we only need their dot products

If we have a function that efficiently computes the dot-products in the expanded space

$$k(x_1, x_2) = \Phi(x_1)^T \Phi(x_2)$$

then we can replace all the dot product in the dual problem

[Remember that $H_{ij} = z_i z_j \Phi(x_i)^T \Phi(x_j)$ for mapping Φ]

- **Training**: The matrix H became: $H_{ij} = z_i z_j \kappa(x_i, x_j)$
- **Prediction**: The scoring function became: $s(x_t) = \sum_{i=1 | \gamma_i > 0} \gamma_i z_i \Phi(x_i)^T \Phi(x_t) + b = \sum_{i=1 | \gamma_i > 0} \alpha_i z_i \kappa(x_i, x_t) + b$

Function k is called kernel function

A kernel function allows training a SVM in a large (even infinite) dimensional Hilbert space $\mathcal{H}$, without requiring to explicitly compute the mapping

Even if the expansion was finite, the complexity of the primal problem may be too large

On the other hand, the complexity of the dual problem only depends on the number of training points.

In practice, we are computing a linear separation surface in the expanded space, which corresponds to a non–linear separation surface in the original feature space

We have already seen an example of feature expansion that provide quadratic separation surface given by the mapping

$$\Phi(x) = \begin{bmatrix} \mathrm{vec}(xx^T) \\ \sqrt{2}x \\ 2 \end{bmatrix}$$ quadratic expansion

Note: $\mathrm{vec}(A)^T \mathrm{vec}(B) = \mathrm{Tr}(A^T B) = \langle A, B \rangle$

$$\Phi(x_1)^T \Phi(x_2) = \begin{pmatrix} \mathrm{vec}(x_1 x_1^T) \\ \sqrt{2}x_1 \\ 1 \end{pmatrix}^T \begin{pmatrix} \mathrm{vec}(x_2 x_2^T) \\ \sqrt{2}x_2 \\ 1 \end{pmatrix} = \langle x_1 x_1^T, x_2 x_2^T \rangle + 2 x_1^T x_2 + 1$$

$$\langle x_1 x_1^T, x_2 x_2^T \rangle = \mathrm{Tr}\left((x_1 x_1^T)^T (x_2 x_2^T)\right) = \mathrm{Tr}\left(x_1 x_1^T x_2 x_2^T\right) = \mathrm{Tr}\left((x_1^T x_2)(x_1 x_2^T)\right) = (x_1^T x_2)^2$$ (dot product)

$$\Phi(x_1)^T \Phi(x_2) = (x_1^T x_2)^2 + 2 x_1^T x_2 + 1 = (x_1^T x_2 + 1)^2 = k(x_1, x_2)$$

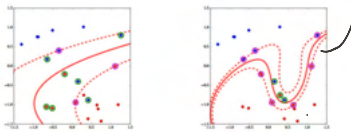
$\Rightarrow O(1)$ complexity

So instead of doing this transformation explicitly we can simply use the Kernel function $k(x_1, x_2) = (x_1^T x_2 + 1)^2$ directly. This becomes especially valuable for higher-degree polynomials where the explicit feature space will grow exponentially with the degree.

In general we can define the polinomial Kernels of degree d as

$$k(x_1, x_2) = (x_1^T x_2 + 1)^d \quad O(d)$$

Note that the expanded space, although finite, grows very quickly



degree 4 polinomial hyperplane

If we know Φ, we can trivially define a corresponding kernel function $k(x_1, x_2) = \Phi(x_1)^T \Phi(x_2)$

However, we typically want to avoid defining explict mappings

We would rather like to know which functions $k(x_1, x_2)$ correspond to a feature mapping Φ in a suitable space $\mathcal{H}$

Mercer's condition provides a sufficient condition for k to define a dot-product in an expanded space

Let $K(u,v)$ be a symmetrical function in L^2 (the space of squer integrable function)
if for all function $g(u)$ such that $\int g(u)^2 du < \infty$, we have:

$$\iint K(u,v) g(u) g(v)\, du\, dv > 0$$

[this are sufficient conditions]
ALCUNI K NON SODDISFANO STE ROBE

then K defines a dot product in sum expanded space.
This mathematical condition essentially state that the Kernel function must be "positive definite" or more precisely "positive semi-definite". In practical term if we form a matrix K where $K_{ij} = K(x_i, x_j)$ for any set of point $\{x_1, x_2, ... x_n\}$ this matrix must have non-negative eigenvalues

Mercer's condition is important because it guarantees that a function can be expressed as a dot product in same features space.
if a function satisfies Mercer's condition, there exist same feature mapping Φ such that:

$$K(x_1, x_2) = \Phi(x_1)^T \Phi(x_2)$$

Even if we never explicitly compute or define Φ.

However, Mercer's condition, doesn't provide a constructive way to build new Kernels. Instade in practice

- we use well-estabilished Kernels that are known that satisfy the Mercer's condition.
- We applay operations that preserve the Kernel property to combine or modify existing Kernels (es. Sum ($K(x_1,x_2) = K_1(x_1,x_2) + K_2(x_1,x_2)$), Product $K_1 \cdot K_2$, scalar Moltiplication, function applications wher f is a function with a convergent Taylor series with non-negative coefficients.

DIPENDE DALLA DISTANZA DEI PUNTI

The "Gaussian Radial Basis Function Kernel" is one of the most widly used Kernel in practies:

$$K(x_1, x_2) = e^{-\gamma ||x_1 - x_2||}$$

it's corresponds to a mapping (Φ) into an infinite dimensional feature space

APPROFONDIMENTO

$$k(x_1, x_2) = e^{-\gamma ||x_1 - x_2||^2}$$

This kernel has a remarkable property: it corresponds to a mapping into an infinite-dimensional feature space. Let's understand why this is the case.

The RBF kernel can be expanded using the Taylor series of the exponential function:

$$e^{-\gamma ||x_1 - x_2||^2} = e^{-\gamma (||x_1||^2 + ||x_2||^2 - 2x_1^T x_2)}$$

This can be rewritten as:

$$e^{-\gamma ||x_1 - x_2||^2} = e^{-\gamma ||x_1||^2} \cdot e^{-\gamma ||x_2||^2} \cdot e^{2\gamma x_1^T x_2}$$

The term $e^{2\gamma x_1^T x_2}$ can be expanded as:

$$e^{2\gamma x_1^T x_2} = \sum_{n=0}^{\infty} \frac{(2\gamma)^n (x_1^T x_2)^n}{n!}$$

Each term $(x_1^T x_2)^n$ corresponds to a feature mapping that includes all possible products of n features. As we sum from $n = 0$ to ∞, we're effectively working in an infinite-dimensional feature space.

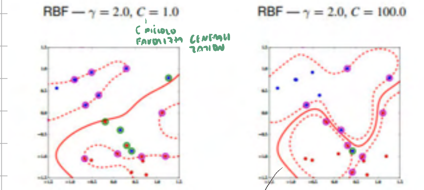
So RBF implicitly maps the data to an infinite-dimensional space wher linear separations becames much more likely, allowin it to create highly flexible decision boundaries

In the RBF the kernel value decrease with the distance between points. Point that are close in the input space have high kernel value (high similarity) while distance point have kernel value near 0

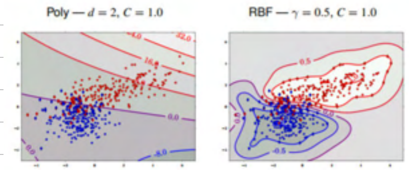
The parameter γ controls the width of the kernel determining how far the influence of a single example reaches:

- Small γ: wide kernel, where each support vector influences point in a large region
- Large γ: Narrow kernel, where the influence of support vectors is highly localized

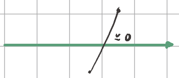
This mean that when γ is very large, the kernels becames extremely narrow, and the svm behaves similarly to a 1-nearest neighbor classifier.
While when γ is very small, the kernel becomes wide and smooth, leading to nearly linear decision boundaries.



γ control the smoothnes



$0 < k(x_1, x_2) < 1$

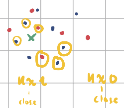


$s(x_t) = \sum_i \alpha_i z_i k(x_t, x_i) + b = 0$ this is the separation rule is control by the support vector

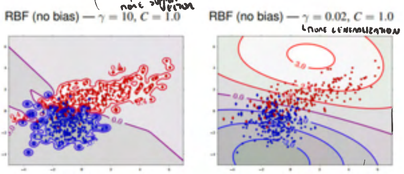
Approssimiamo tutti i support vector ... $\alpha_i = c$ e togliamo b

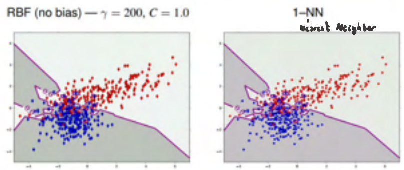
$\sum_i z_i k(x_t, x_i) = 0$ if two point are close kernel is 1 if the are far is 0
> 0
< 0

Test point
Support Vector



[So the score is controlled by the support vector near the test point (in a very soft way)]





Practical Consideration

One of the most crucial aspect of applaying svms effectivly is selecting appropiate hyperparameters. The two main category of hyperparameters are:

① Kernel-specific-parameters

Different kernel have different parameters that control their behavior:

- For polynomial kernels the degree d and the constant c
- For RBF kernels: the width parameter γ (gamma)

The choice of γ in RBF kernels is particularly important:

- Too small γ: The kernel becomes too wide, creating overly smooth decision boundaries that may underfit the data
- Too large γ: The kernel becomes too narrow, potentially creating complex boundaries that overfit the training data

② Regularization parameter C

The parameter C controls the trade-off between maximizing the margin and minimizing classification errors:

- Small C: Prioritizes a wider margin, potentially at the cost of a smaller margin
- Large C: Focuses on reducing training errors, potentially at the cost of a smaller margin

Cross-validation is the recommended approach for selecting these hyperparameters. This typically involves:

① Splitting the training data into multiple folds
② Training models with different hyperparameter combinations
③ Selecting the combination that performs best on the validation sets

Despite the power of kernel svms, linear svms (using the linear kernel) are often sufficient and preferable in certain scenarios:

① High-dimensional data: When working with very high-dimensional feature space, the data often becomes nearly linearly separable. This phenomenon is related to the "curse of dimensionality" becoming a blessing for linear separability.

② Computational efficiency: linear svms are significantly faster to train and evaluate. The scoring function is a simple dot product between the weight vector and the input features, making predictions extremely efficient. (O(n))

svms are not invariant to affine transformation of the data, meaning that scaling and transformations of features can significantly impact performance. Proper preprocessing typically includes:

Centering: Subtracting the mean of each feature to center the data around the origin

Scaling: Normalizing features to have comparable ranges or standardizing them to have zero mean and unit variance

Whitening: Transforming the data to have uncorrelated features with unit variance

These preprocessing steps are particularly important for distance-based kernels like the RBF kernel, which are sensitive to the scale of the input features.

Unlike logistic regression svm scores do not have a direct probabilistic interpretation. The raw output of an svm is simply the signed distance from the decision boundary, not a probability.

When probabilistic outputs are needed (for instance, to make cost-weighted decisions or combine svm predictions with other probabilistic models), post-processing methods can be applied

Platt scaling: Fitting a logistic regression model to transform SVM scores into probabilities
Isotonic regression: A more flexible non-parametric approach to calibrate scores to probabilities

These methods typically require a held-out validation set to avoid overfitting.

When working with imbalanced datasets (where one class is much more frequent than the other), standard SVMs may produce biased results favoring the majority class. There are several approaches to address this:

① Class-weighted SVMs: Assigning different C values to different classes to balance their importance

② Sample-weighted SVMs: Giving individual samples different weights based on their importance

To re-balance the classes we can use a different value of C for the different classes (or even for different samples):

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} C_i \left[1 - z_i(\mathbf{w}^T\mathbf{x}_i + b)\right]_+ = \min_{w,b} \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} C_i \max\left(0, 1 - z_i(w^T x_i + b)\right)$$

The corresponding dual problem is[5]

$$\arg\max_{\alpha} \boldsymbol{\alpha}^T\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T H\boldsymbol{\alpha}$$

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i z_i = 0\,, \quad 0 \le \alpha_i \le C_i,\ i = 1, \ldots, n$$

[5] In the laboratory we will omit the constraint related to the bias, since we will not explicitly model the bias term

For class balancing, we can set $C_i = C_T$ for samples of class $\mathcal{H}_T$ and $C_i = C_F$ for samples of class $\mathcal{H}_F$

We can select $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$

π_T^{emp} and π_F^{emp} are the empirical priors (i.e. sample proportions) for the two classes computed over the training set

π_T e π_F are the target (desired) class probabilities

In practice, we are simulating the application class proportions at training time

This approach effectively simulates the target class distribution during training, helping the model focus appropriately on both classes regardless of their representation in the training data.

MNIST — Comparison of average pairwise EER for different kernels (raw features)

Kernel	$C = 10$	$C = 1$	$C = 0.1$	LogReg ($\lambda = 1e^{-3}$)
Linear	1.6%	1.3%	1.1%	1.2%
Poly ($d = 2$)	0.3%	0.3%	0.3%	0.9%[6]
Poly ($d = 3$)	0.4%	0.4%	0.4%	—
RBF ($\gamma = 0.2$)	0.7%	0.7%	0.9%	—
RBF ($\gamma = 1.0$)	0.5%	0.5%	0.6%	—
RBF ($\gamma = 2.0$)	0.1%	0.1%	0.1%	—

we used PCA to reduce dimension to 50

[6] Quadratic expansion of PCA–reduced features

Kernel Choice Matters: Nonlinear kernels significantly outperform linear kernels on this image recognition task. This suggests that the relationship between pixel values and digit classes is inherently nonlinear.

RBF Kernel with γ = 2.0 Achieves Best Performance: This configuration achieves an impressive 0.1% error rate across different C values, showing that with the right kernel parameters, SVMs can be extremely effective for image classification.

Regularization Impact Varies by Kernel: For the linear kernel, the choice of C makes a noticeable difference (error rates ranging from 1.1% to 1.6%). However, for nonlinear kernels, performance is much more stable across different C values. This suggests that once the right kernel is selected, the exact regularization strength becomes less critical.

Comparison with Logistic Regression: SVMs with nonlinear kernels consistently outperform logistic regression on this task. Even with quadratic feature expansion, logistic regression achieves only 0.9% error rate, compared to 0.1% for the best SVM configuration.

Extending SVMs to Multi-class-Problem

① One-versus-All (OVA) Approach

This is the most common strategy:

- Train N different binary SVMs, each distinguishing one class from all others
- For a new sample, evaluate all N classifier
- Assign the sample to the class whose classifier produce the highest score.

While simple to implement, this approach may be affected by imbalance issues (since each classifier sees many negative examples but few positive ones) and may produce ambiguous regions where multiple classifiers give conflicting decisions

② One-versus-One (OVO) Approach

It handle multiclass classification by breaking it down into multiple binary classification problems - specifically one binary classifier for each for each possible pair of classes.

- Training Phase
  - For a problem with N classes, we train $N(N-1)/2$ binary SVM classifiers.
  - Each classifier is trained to distinguish between just two classes, ignoring all other classes.

For example, with 10 classes (like digits 0-9) we would train 45 different binary classifiers (one for each pair: 0 vs 1, 0 vs 2 ... 8 vs 9)

- Prediction Phase
  For a new sample, we run it through all $N(N-1)/2$ classifiers, each classifier votes for one of the two classes it was trained on, then we count the votes received by each class.
  The sample is assigned to the class that received the most votes.

Another possible way to operate is the **Direct Multiclass Formulation**

The direct multiclass formulation extends the SVM concept to handle all classes simultaneously in a single optimization problem, rather than breaking it down into multiple binary problems

- Introduce an objective function that maximizes the margin between each class and all the others. For example, (we ignore the bias terms)

$$\min_{\mathbf{W}} \frac{1}{2}\|\mathbf{W}\|_2^2 + C\sum_{i=1}^{n} \max_{y'} \left[(\mathbf{w}_{y'} - \mathbf{w}_{z_i})^T \mathbf{x}_i + \delta_{y',z_i}\right]$$

Compare with multiclass logistic regression

$$\min_{\mathbf{w}} \frac{\lambda}{2}\|\boldsymbol{W}\|_2^2 + \frac{1}{n}\sum_i \log\left(\sum_{y'} e^{(\mathbf{w}_{y'} - \mathbf{w}_{z_i}^T)^T \mathbf{x}_i}\right)$$

- $(\mathbf{w}_{y'})^T \mathbf{x}_i$ is the score for some class y'
- $(\mathbf{w}_{z_i})^T \mathbf{x}_i$ is the score for the true class z_i
- $(\mathbf{w}_{y'} - \mathbf{w}_{z_i})^T \mathbf{x}_i$ is the difference between these two scores
- δ_{y',z_i} equals 1 when $y' = z_i$ (same class) and 0 otherwise

The term $\max_{y'}[(\mathbf{w}_{y'} - \mathbf{w}_{z_i})^T \mathbf{x}_i + \delta_{y',z_i}]$ is calculating the loss based on the most problematic class.

Note: -

"the most competing/problematic class, is the incorrect class that receives the highest score for a given input example (training point)

In multi-class classification, each class has its own weight vector (or decision function). When we classify an example, we compute a score for each possible class and typically assign the example to the class with the highest score.

For an input example x_1 with true class z_1, we would calculate scores for all possible classes:

- Score for class 1: $w_1{}^T x_1$
- Score for class 2: $w_2{}^T x_1$
- Score for class 3: $w_3{}^T x_1$
- And so on...

Ideally, we want the score for the true class ($w_1{}^T x_1$) to be higher than the scores for all other classes. But among all the incorrect classes, one of them will have the highest score. This is what I'm calling "the most competing class" - it's the incorrect class that is most likely to be confused with the true class for that particular example.

The max is finds the class that maximizes this expression, in other words, it finds the class that most violates our desire to have the true class score higher than all other class score by some margin. the incorrect class and the higer score

The SVM is specially focused on increasing the margin between the true class and the most competing class rather than worring about classes that are already easily distinguished.

Kernel methods are not restricted to SVMs

- Logistic Regression can be extended to incorporate kernels
- In general, we can apply the kernel tricks to problems which depend only on dot–products
  - Kernel PCA
  - Gaussian Processes
  - ...