**Linear Models**

Sommario

[Modelli Lineari 2](#_Toc118390105)

[Risoluzione dell’equazione matriciale – caso ideale 3](#_Toc118390106)

[Problemi dei Least Squares 6](#_Toc118390107)

[Least Squares Overfitting 6](#_Toc118390108)

[Least Squares per la classificazione 9](#_Toc118390109)

[Support Vector Machines 9](#_Toc118390110)

[Margine funzionale 10](#_Toc118390111)

[Margine geometrico 11](#_Toc118390112)

[Formulazione duale 13](#_Toc118390113)

[Permettere errore sul margine 16](#_Toc118390114)

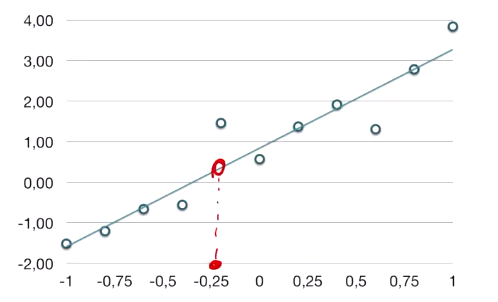
[Kernels trick 20](#_Toc118390115)

Lezione 1

# Modelli Lineari

Immaginiamo un problema di **regressione** in cui abbiamo un data set e il nostro obiettivo è tracciare un modello lineare che passi il più possibile vicino i punti in modo da poter fare predizione.

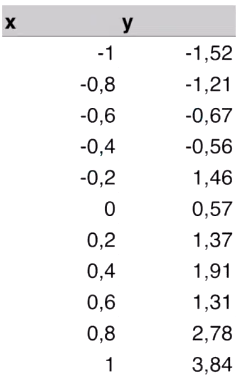
La variabile target y sarà un numero reale. Nel caso più semplice la descrizione dell’esempio apparterrà anch’esso ai numeri reali (le sue features saranno numeri reali).



In questo caso sull’asse delle x abbiamo la descrizione dell’esempio e su quello delle y abbiamo la nostra variabile target.

**Vogliamo trovare la retta che passa attraverso i punti nel grafico, che minimizzi gli errori di predizione**. In questo caso data la x noi vorremmo che il modello ci restituisse la y corretta.

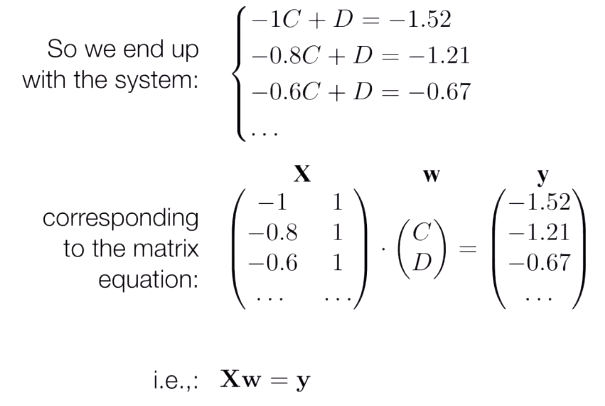
Nel formalizzare il problema iniziamo con la rappresentazione matriciale dei nostri dati.



Vogliamo trovare un modello del tipo y = ax + b (che è come dire Cx + D = y). Costruiremo un sistema di questo tipo che vorremmo risolvere



Il passo successivo è riscrivere il problema in forma matriciale



Nel costruire la matrice bisogna mettere sempre la colonna che costituisce il termine noto (detto anche bias), 1 in questo caso. Senza il termine noto apprendiamo una retta che passa per l’origine.

## Risoluzione dell’equazione matriciale – caso ideale

Nel caso ideale, x è una matrice quadrata e a rango pieno, x è invertibile e si può risolvere il problema immediatamente perché il valore di w lo trovo andando a scrivere

X-1 \* X \* w = X-1 \* y

Dove X-1 \* X è uguale alla matrice identità e quindi si può omettere. Si avrà



Il punto è che nei casi di interesse la matrice X non è invertibile (perciò non è quadrata e a rango pieno). Dobbiamo trovare un nuovo approccio.

Es.

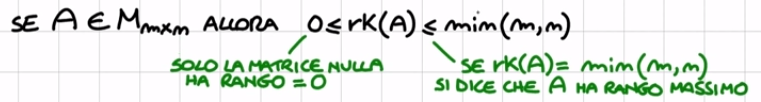
Immagine che contiene testo

Descrizione generata automaticamente

Detto ciò, il vettore dei pesi **w** che troveremo, in quanto non saremo mai nel caso ideale di matrice X invertibile, non sarà la vera w, ma una sua **approssimazione** (che è anche meglio in quanto evita di fare overfitting).

Nozioni di algebra lineare:

* **Matrice a rango pieno**: i vettori nella matrice x sono tutti indipendenti tra di loro
* **Rango**: il rango di una matrice rappresenta il massimo numero di righe o colonne linearmente indipendenti.



* Se due vettori sono linearmente dipendenti allora stanno allora esiste una loro combinazione lineare, a coefficienti non nulli, che dà come risultato il vettore nullo



* Se n vettori v1, v2, …, vn sono linearmente dipendenti allora stanno sullo stesso piano, se invece sono linearmente indipendenti in Rx allora **non sono complanari**.
* Vediamo come riscrivere il prodotto matrice-vettore

Immagine che contiene testo, lavagnabianca

Descrizione generata automaticamente

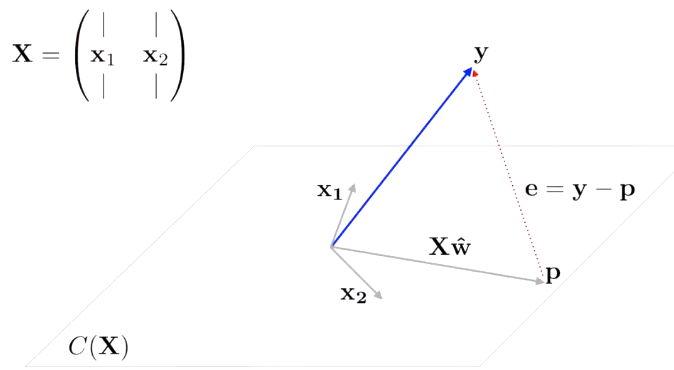
Abbiamo una combinazione lineare. 2 e 4 in realtà saranno w1 e w2 e noi non li conosciamo, sono i pesi (coefficienti) che vogliamo trovare. Avremo quindi

Immagine che contiene testo, lavagnabianca

Descrizione generata automaticamente

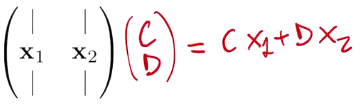
Considerando tutte le combinazioni lineari tra x1 e x2 ottengo uno **spazio vettoriale**, che avendo due vettori sarà un piano.

Detto ciò, variando il vettore w noi stiamo **spazzolando** lo spazio dei vettori x e notiamo anche che y non appartiene a questo.



Partendo dal piano di partenza, formato da x (esempio) e y (target), passiamo ad avere un nuovo piano (spazio vettoriale), che chiameremo C(x). C rappresenta le colonne di X, quindi la combinazione lineare delle colonne di X.

Moltiplicando le colonne di X (i vettori x1 e x2) per (C D) avremo



Il risultato è l’equazione che genera lo spazio C(X), dove C(X) è lo **spazio delle colonne di X**.

**La soluzione che stiamo cercando (Xw), appartiene allo spazio delle colonne di X.**

Dall’immagine notiamo che y non appartiene allo spazio delle colonne di x; quindi, noi vorremo trovare il valore che più si avvicina a y, per farlo, ci basterà andare a **minimizzare** la differenza y e p, dove p sarà un valore presente nello spazio vettoriale (dato da un certo w, tale che Xw = p), minimizzando quindi il vettore e

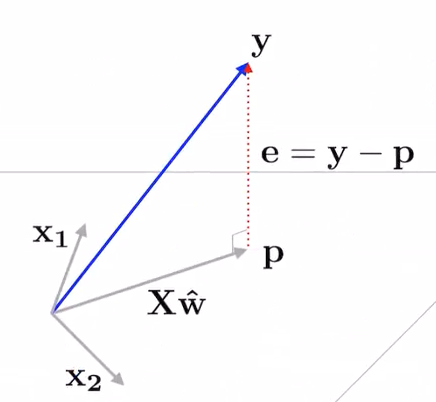
**e = y – p**

si tratta di un **errore** che vogliamo minimizzare.

Il punto p che minimizza la distanza con y è il punto che forma il vettore e ortogonale con c(x), cioè dato qualsiasi vettore x appartenente a c(x), andando a fare il prodotto scalare tra x ed e si ottiene 0. Se x appartiene a c(x) allora x = Xw = x1\*w1 + x2\*w2. Se xw = 0 deve essere vero per ogni vettore in c(x), quale condizione devo imporre? Imporre che un vettore sia ortogonale ad un piano, corrisponde a imporre che sia ortogonale a una base del piano (cioè deve essere ortogonale ai vettori che hanno generato il piano x1 e x2). Quindi, in questo caso, imporremo

* x1\*e = 0
* x2\*e = 0

senza dover considerare W.



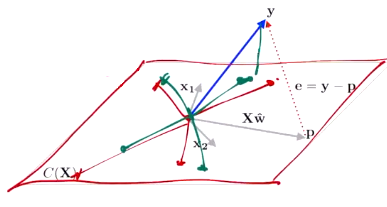
Come scriviamo in forma matriciale ciò che abbiamo detto?



Da notare che la prima matrice non è X, ma XT.

PS. Le basi del piano sono i vettori che hanno creato il piano (x1 e x2).

Noi vogliamo che e sia ortogonale alle basi del piano.



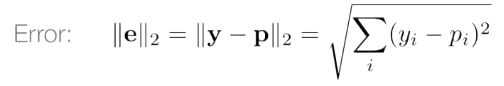
C(X) è un iperpiano in questo caso.

## Soluzione least squares

Immagine che contiene testo

Descrizione generata automaticamente

Se calcolo la differenza tra y e un punto p trovato ottengo un vettore, che chiamiamo e. quello che voglio fare è trovare il p, generato dalla moltiplicazione tra w e x e dato che x è costante quindi vogliamo trovare il w, tale che abbia la norma minima (quindi tale che il vettore e sia il più corto)



Per minimizzare l’errore posso minimizzare l’errore al quadrato, tanto la crescita è monotona, in modo tale da semplificarmi i calcoli successivi.

Quindi ||e||22  così da poter togliere la radice quadrata.

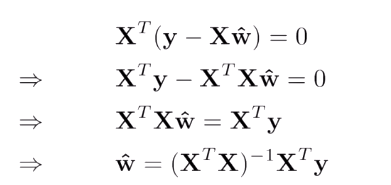
Posso ridurre il mio problema a



L’errore minimo si ottiene avendo vettore di errore perpendicolare al piano C(X). Dobbiamo imporre questo vincolo al problema per arrivare ad avere la soluzione. Affinché e sia perpendicolare a tutto il piano dev’essere perpendicolare ad ogni vettore che genera il piano. La condizione di perpendicolarità tra due vettori è che lo scalare tra loro faccia 0. Quindi e \* xi = 0.

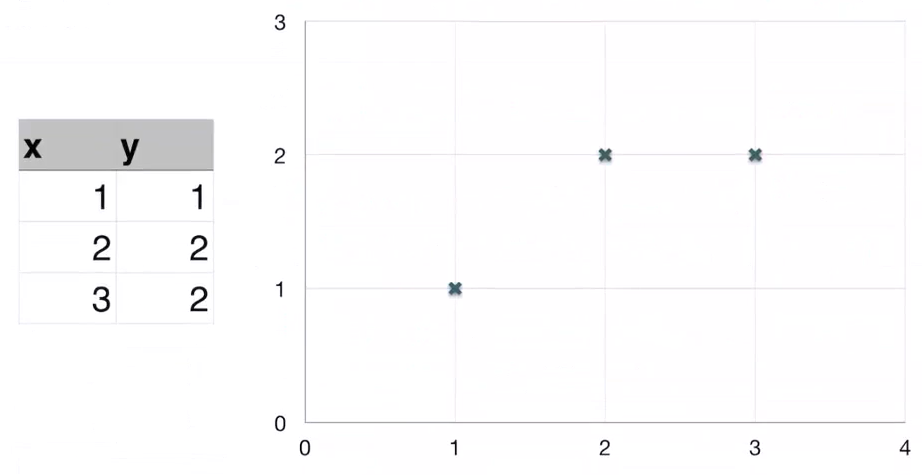
Basta imporre una moltiplicazione matriciale. Useremo X trasposta in modo da imporre tutti i vettori presenti nello spazio delle colonne di X perpendicolari a e. Imporremo che





Facendo così scopriamo la **pseudo-inversa di X** che appunto è **(XTX)-1XT**.

### Esempio



Usiamo la formula calcolata per trovare w, avendo X e y come segue

Immagine che contiene testo

Descrizione generata automaticamente

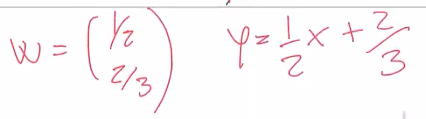
PS. Occhio alla colonna x2 tutta formata a 1.

Per calcolare l’inversa della matrice viene usato il metodo di Gauss-Jordan (non serve saperlo fare).

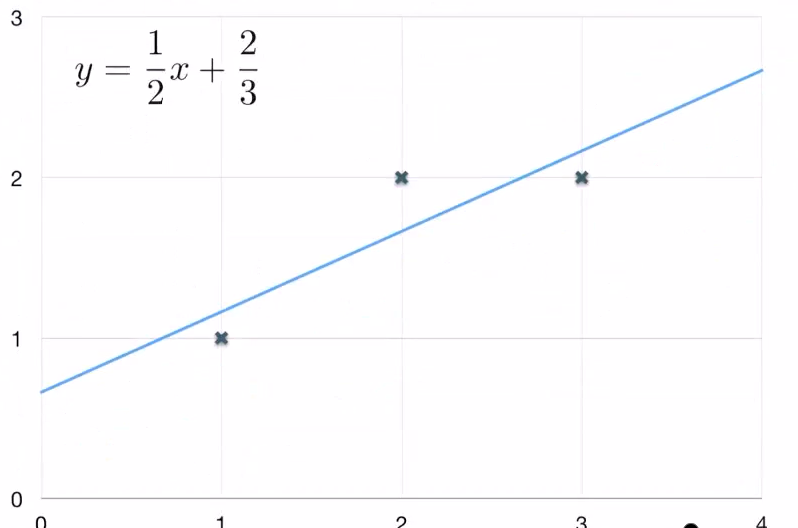
Immagine che contiene testo

Descrizione generata automaticamente

Otteniamo quanto segue



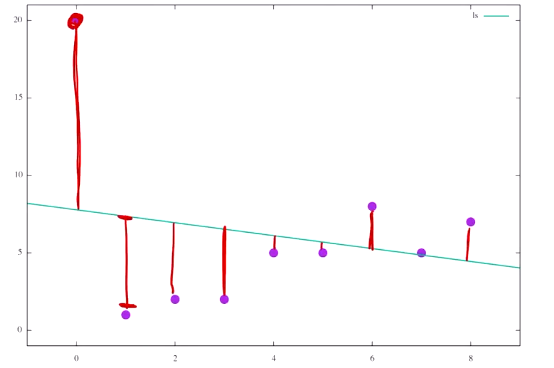
Andando ad ottenere quanto segue



### Problemi dei Least Squares

#### Least Squares Overfitting

È prono all’overfitting, in particolare minimizza la funzione obiettivo che dipende dal quadrato delle differenze tra i nostri valori e la nostra soluzione.



Un esempio come quello più a sinistra nel grafico peserà tantissimo nella costruzione della funzione perché sarà molto lontano dalla retta e quindi avrà una penalità molto alta e andrà a spostare la retta anche se diventa meno precisa.

Un outlayer è un punto estratto da un’altra distribuzione, come ad esempio è l’esempio a sinistra.

Come si risolve? Con la tecnica di **regolarizzazione della soluzione**.

#### Regolarizzazione

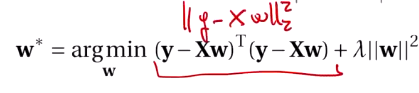
È un metodo generale per evitare l’overfitting. L’idea è aggiungere dei vincoli sulla soluzione che sto cercando che vincolano la forma dell’insieme dei pesi w che sto utilizzando. La maggior parte delle volte sono dei vincoli che chiedono che il vettore in questione sia un vettore piccolo, cioè che non possa crescere in tutte le dimensioni a piacere ma che debba essere il vettore più piccolo possibile. Questo metodo di regolarizzazione si chiama **shrinkage**.

Noi vogliamo che w assuma valori troppo grandi, che equivale a imporre il vincolo che la norma di w deve essere piccola.

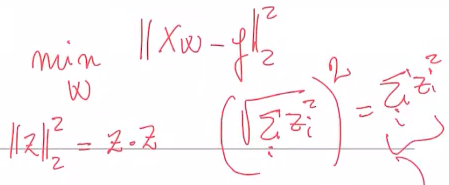
Questa idea può essere implementata con due tecniche diverse

* **ridge** **regression**
* **lasso** **regression**

##### Ridge regression



PS. Il problema che abbiamo risolto inizialmente poteva essere visto come la minimizzazione di Xw-y che equivale a argmin…



Nella formula **lambda** è uno scalare che determina la quantità di regolarizzazione. Così facendo il vettore dei pesi non cambierà a piacere ma avrà una lieve modifica.

Se lambda è tanto grande, w sarà tanto piccolo, se lambda è preponderante w sarà il vettore di tutti 0.

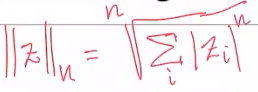
**Questo problema ha una soluzione in forma chiusa** che è molto **simile a quella del problema originale**. L’unica differenza è che al posto di calcolare (XTX)-1 calcolo (XTX + lambda\*I)-1 dove I è la matrice identità. Questo corrisponde ad aggiungere lambda alla diagonale di XTX. Questo è un trucchetto ben noto in ambito di calcolo numerico per migliorare la stabilità dell’inversione della matrice, così la matrice durante l’inversione si comporta meglio.



Questo modo di fare regolarizzazione si chiama **ridge regression** (per indicare che aggiungo + lambda||w||22).

##### Lasso regression

Cambia che al posto di usare la 2-norm, **usiamo la 1-norm**.



Un altro modo per fare regolarizzazione negli Squares è utilizzare la **lasso regression**, dove lasso sta per “least absolute shrinkage and selection operator”. La differenza rispetto alla ridge regressione è che sostituiamo ||w||22  con ||w||1 . Quindi il termine di regolarizzazione diviene lambda||w||1.

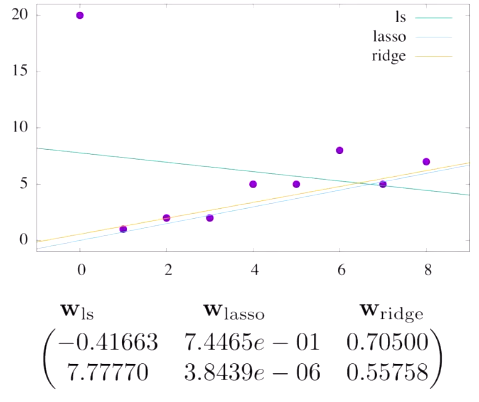
**||w||1 = ∑|wi|**

Nel nome, least absolute shrinkage fa riferimento al fatto stiamo cercando di minimizzare i valori assoluti delle singole componenti w.

**Uno degli effetti della regressione lasso è che la soluzione che troviamo è quella più sparsa** (quindi avremo un vettore con tanti zeri). Il vantaggio di ciò è perché ci permette di avere un modello più interpretabile (capiamo quale feature usa e quale no).

Nel caso della ridge regression come detto abbiamo una soluzione in forma chiusa, **nel caso di lasso regression NON esiste la soluzione in forma chiusa**. Utilizzeremo dei **solver numerici**, per risolvere il problema di minimizzazione.

#### Esempio



Lasso notiamo che dà dei termini più piccoli rispetto alla ridge. Ad esempio, w2 è 3.8\*10-6, infatti vediamo che la retta parte praticamente da zero.

### Domande

1. **Perché per ovviare al problema del least squares dobbiamo minimizzare la norma del vettore dei pesi?**
2. **Perché lasso favorisce soluzioni sparse?**

#### Risposte

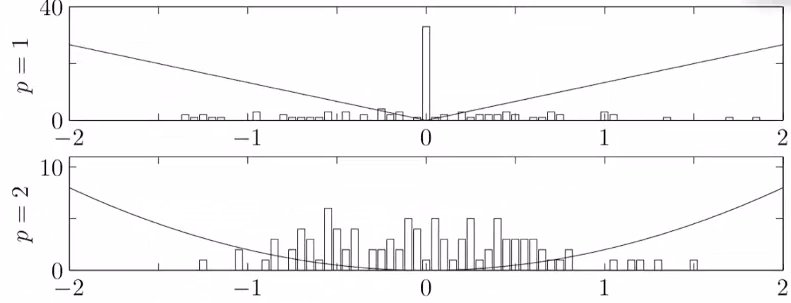
1. Iniziamo col vedere diverse motivazioni (quando vedremo le SVM vedremo un ulteriore motivo di ciò):
   1. Minimizzando la w, minimizziamo anche l’effetto dell’errore D.

Immagine che contiene testo

Descrizione generata automaticamente

* 1. Avere un vettore dei pesi più piccolo ci permette anche di avere un **modello semplice** (rasoio di Occam).
  2. Aggiungendo il vincolo di norma piccola, stiamo aggiungendo un **bias** **induttivo** (tipo di bias che permette agli algoritmi di apprendere qualcosa), che **riduce l’errore sulla** **varianza**, perché andiamo a dire che non potranno essere scelte tutte le w possibili, ma solo quelle con norma piccola, facendo così scendere l’errore dovuto alla varianza (sperando di non urtare troppo l’errore dovuto alla polarizzazione)

1. Perché l’approccio lasso favorisce soluzioni sparse?



In questa figura, sulle x abbiamo i valori assunti dai wi e sulla y abbiamo la frequenza dei valori assunti dai wi.

Vediamo che nel grafico della 1-norm, ci sono tanti valori che hanno esattamente zero. Mentre su x=1 vediamo che ci sono pochi wi con valore 1.

Le righe (o semicerchio) che partono da zero danno un’altra informazione: corrisponde alle forme dei due fattori di regolarizzazione (lasso e ridge).  
Per quanto riguarda ridge, vediamo che più ci avviciniamo allo zero, più la derivata è zero (ad occhio possiamo dire che da circa -0.3 a 0.3 la derivata è 0).

Per quanto riguarda lasso, vediamo che la derivata è zero solo in x=0, mentre negli altri punti è lambda.

Ciò a cosa ci serve? Quando facciamo ridge regressione e decidiamo di portare un wi verso zero piuttosto che un altro, quale scegliamo? Quello che ci da vantaggio maggiore. Il vantaggio lo calcoliamo in base a quanto la derivata diminuisce in base a quanto l’ho spostato. La funzione obiettivo li vuole allontanare dallo zero, mentre la regolarizzazione li vuole avvicinare, dobbiamo bilanciare le due cose, vorremo il più grande vantaggio con il minimo spostamento verso zero. Nel caso del ridge ci verrà da spostare wi lontani a zero (perché ci danno più guadagno).

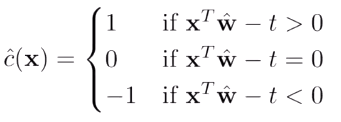
Nel caso del lasso invece spostare un valore vicino a zero e spostare un valore lontano da zero ci danno lo stesso guadagno, quindi li sposteremo tutti verso zero, avendo un grosso accumulo di valori vicino a zero.

Quindi nel ridge quando siamo vicino a zero ci fermiamo con gli spostamenti, con la lasso invece ci impegniamo per portarli a zero (per questo abbiamo sparsità)

Lezione 19

### Least Squares per la classificazione

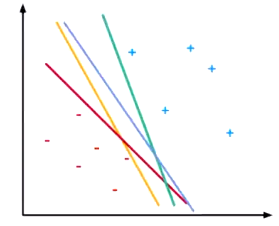
Nel caso della classificazione vogliamo trovare una funzione C(X) che restituisce la classe degli esempi che sto guardando. Guardiamo il caso binario



In questo caso il termine t viene messo esternamente e non nella matrice W. Al posto di trovare una retta che passa attraverso l’insieme di punti vogliamo una retta che separa due insiemi di punti.

## Support Vector Machines

Modello lineare per la **classificazione**.



Qual è la riga migliore? È quella blu, perché è maggiormente distante dagli esempi positivi e negativi.

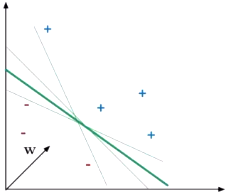
L’idea delle support vector machines, invece di fare come il percettrone, e quindi lasciare al caso la scelta della riga, do un’indicazione precisa e dico che voglio la linea di separazione che passa più in centro possibile tra gli esempi positivi e quelli negativi. Detto in maniera tecnica vogliamo cercare l’iperpiano che massimizza il minimo margine tra gli esempi.

Chiamiamo **margine** la distanza tra ognuno degli esempi e l’iperpiano che separa le due classi (è una distanza positiva quando il punto è etichettato positivamente e negativa quando è etichettato negativamente).

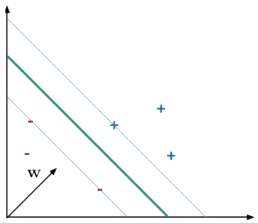
Quello che si cerca è di **massimizzare il** **minimo margine**, cioè allontano la retta il più possibile dall’esempio più vicino (gli esempi cambieranno finché non ci sarà stabilità).

Vedremo che per riuscire a capire bene come arrivare alla soluzione di questo problema dovremo introdurre due tipi di margine

1. uno lo chiameremo **margine funzionale**, e ci permetterà di garantire che la retta che troviamo separi esattamente i punti positivi da quelli negativi e che quindi non ci siano errori di classificazione.



1. Il secondo lo chiameremo **margine geometrico**, e ci garantirà che la retta che troviamo massimizza il minimo margine



In tutto questo discorso assumeremo che gli esempi siano **linearmente separabili** (quindi che è possibile separarli tramite iperpiano).

### Margine funzionale

Partiamo dal margine funzionale. Definiamo la funzione f in un punto xi come

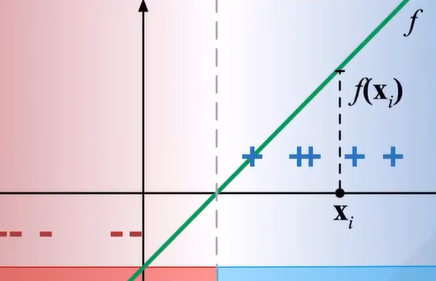
**f(xi) = wxi – t**

La funzione della retta che vogliamo trovare. Notiamo che il valore di f nel punto xi è un valore che può andare da -infinito a +infinito. Noi decideremo di etichettare positivamente gli esempi per cui **f(xi) > 0** e negativamente tutti gli altri.

Consideriamo il problema nella formula più semplice possibile, ad una sola dimensione



Nella retta dei reali il separatore sarà un certo punto presente sulla retta. La nostra funzione f(xi) sarà su un piano bidimensionale. Se l’altezza (quindi f(xi)) è positiva allora l’esempio sarà etichettato come positivo, negativo altrimenti.



Le rette che vediamo in realtà non sono delle rette, ma dei piani che intersecano il grafico degli esempi.

Gli esempi vanno pensati nel piano x, y in questo caso, mentre l’iperpiano si trova sull’asse z.

Il **margine** **funzionale** non è altro che la quantità calcolata in questo modo



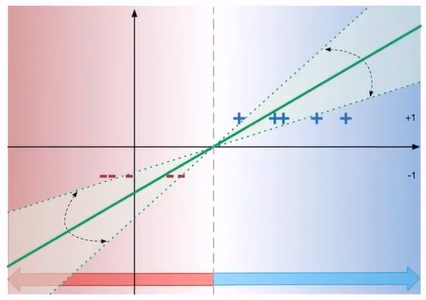
Cioè il valore yi (il valore target dell’esempio), che può essere +1 o -1, moltiplicato per la definizione di f, cioè l’altezza tra i punti (xi, f(xi)) e l’asse delle x moltiplicata per yi. Il margine sarà positivo sse l’esempio è classificato correttamente (quindi, ad esempio, yi = +1 (target di xi) e f(xi) = +1).

PS. **Se u(xi) > 0 allora xi è stato classificato correttamente**

Detto ciò, capiamo che il primo vincolo che vogliamo imporre è che il margine funzionale sia maggiore di 0 per tutti gli esempi



Il fatto è questo, cambiando w e t cambia l’inclinazione del piano



Possiamo dire questo, se esiste **yi(wxi – t) > 0**, allora esisterà anche una costante c, tale che **yi(wxi – t) >= c** (c è il valore più piccolo che può assumere u(xi)). Dividendo ambo i membri per 1/c



Otteniamo

yi(wxi – t) >= 1

PS. 1/c a sinistra viene inglobato dentro w e t. più precisamente succede questo



Andiamo a porre che l’uguaglianza valga per gli esempi sul margine

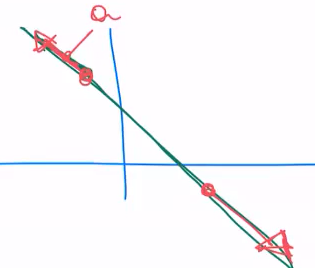
yi(wxi – t) = 1

Mettendo questo vincolo, gli esempi più vicini alla retta avranno margine funzionale = 1.

Gli esempi che realizzano margine funzionale esattamente uguale a 1 vengono detti **vettori di supporto**.

### Margine geometrico

Breve ragionamento:

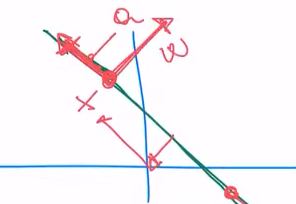


**Che relazione c’è tra la retta wx – t (cioè a) e w?**

PS. w è il coefficiente angolare della retta.

Come verifichiamo che siano paralleli? La direzione di a e w non cambia se sposto la retta nell’origine; quindi, la relazione vale comunque anche se considero wx = 0, ponendo t a 0 in quanto la retta passa per l’origine.

A questo punto, il vettore a, è semplicemente un vettore x che parte dall’origine. A questo punto, il prodotto scalare tra w e x è 0 (wx = 0), quindi vuol dire che **sono ortogonali**.



Riprendiamo con il margine funzionale:

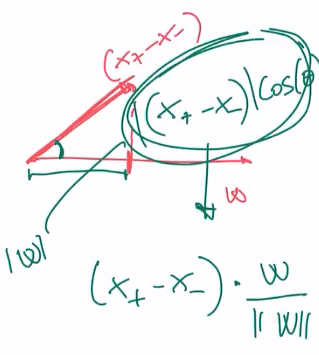
Vogliamo calcolare la linea rossa, cioè la distanza tra l’esempio più vicino all’iperpiano e l’iperpiano stesso. In realtà per semplificare i calcoli calcoleremo la linea arancione (che è il doppio di quella rossa)

Immagine che contiene testo, cielo, mappa

Descrizione generata automaticamente

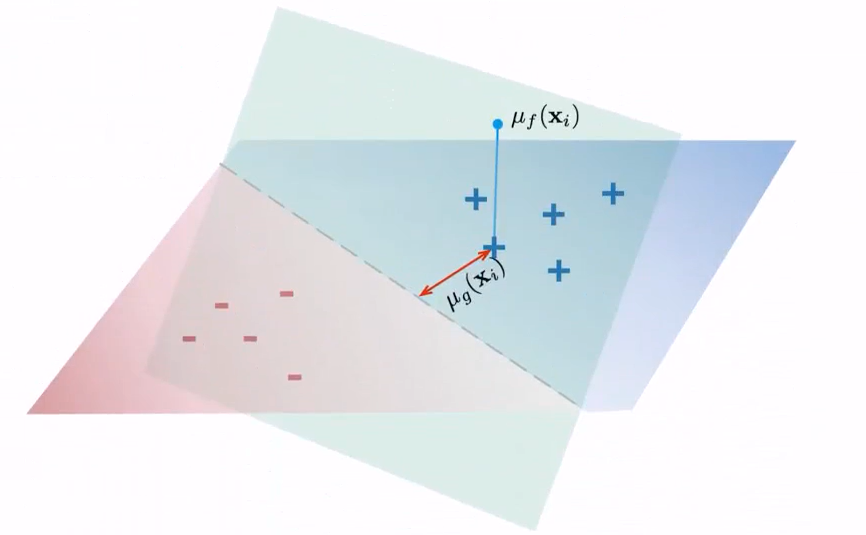
Per calcolare la distanza consideriamo l’esempio positivo più vicino e quello negativo più vicino (x+ e x-). Calcoliamo la differenza tra i due (x+ - x-), che ci dà un vettore che ha direzione da x+ a x-. Ricordiamo che x+ e x- sono due vettori, facendo la differenza tra i due ottengo la differenza tra x+ e x-.

Ottenuto ciò vorremo proiettare il vettore ottenuto nella direzione della w (freccia gialla). Per fare ciò moltiplicheremo il vettore per w, e per evitare che w vada ad incrementare lo scalare del vettore differenza allora dividiamo per ||w|| (anche perché w può essere grande come piccolo, dividendo per la norma gli diamo dimensione 1 così da non influire sulla dimensione del vettore proiettato).  
Il ragionamento è che noi vogliamo proiettare il vettore differenza lungo w, quindi vogliamo calcolare la lineetta verde.



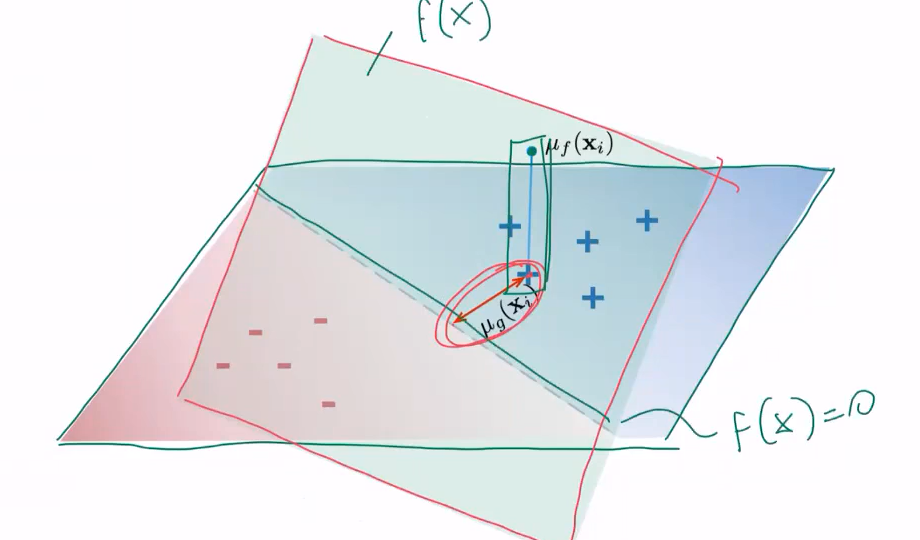
Per trovare la linea verde dovremmo fare |(x+ - x-)|\*cos(teta), visto che non conosciamo teta ci basterà usare la formula (x+-x-)\*w/||w||.

A questo punto possiamo formulare il problema delle **support vector machine**. Vediamo una rappresentazione grafica



Notiamo l’iperpiano che taglia lasse x, y.

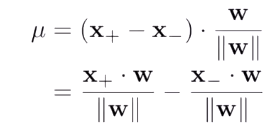
* Il margine geometrico è la distanza in rosso
* Quello funzionale è la distanza in blu



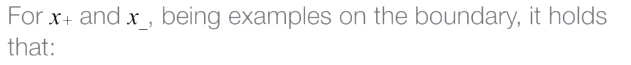
## Definizione del problema SVM

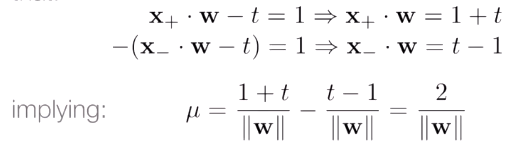
Vogliamo imporre all’algoritmo di massimizzare la distanza in arancione calcolata, quindi **massimizzare il margine geometrico**.

La dimensione del margine può essere valutata come

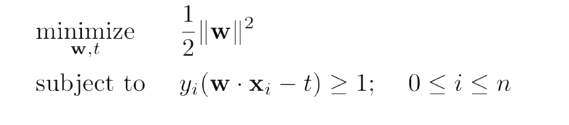


x+ e x- sono i vettori di supporto (quelli più vicini al margine geometrico)





Stiamo dicendo che vogliamo massimizzare 2/|w|, ma massimizzarlo è equivalente a minimizzare |w|. Questo è equivalente a minimizzare |w|2 (che ci permette di togliere la radice dalla norma), quindi tutto il problema di minimizzazione possono scriverlo come segue



PS. La norma sarà 2-norm

“subject to” è il vincolo che poniamo. Per far si che valga il dataset deve essere linearmente separabile.

Quello che facciamo ci concentriamo su tutte le rette che rispettano il vincolo, dopodiché tra tutte quelle mi concentro su quelle che hanno soluzione minima (quindi margine geometrico massimo).

Quello che stiamo facendo è **regolarizzazione**, tra tutte le soluzioni possibili vogliamo quella che minimizza la norma di w, come facevamo coi minimi quadrati. La differenza con i minimi quadrati era che prima non eravamo certi di avere w minimo.

Quest’ultimo è il problema che definisce le SVM, come si risolve? Dal punto di vista matematico è difficile e non esiste la risoluzione in forma chiusa, è possibile però calcolarne una. In quanto essendo un problema **convesso** avrà un unico minimo, che sarà la soluzione.

Ora trasformeremo questo problema in uno equivalente, detto problema **duale**, ed è quello usato nei software per trovare la soluzione delle SVM per diverse ragioni. Nel trovare il duale scopriremo cose di questa soluzione che ci aiuteranno meglio a capire questa versione appena trattata.

Sarà sempre un problema con risoluzione non in forma chiusa (che dovrà essere quindi risolto da un solver), ma avrà come vantaggi

* Dal punto di vista pratico, potremo applicare la tecnica di **kernel trick**, cosi da poter usare la soluzione anche per problemi non lineari (quindi per risolvere problemi non linearmente separabili)

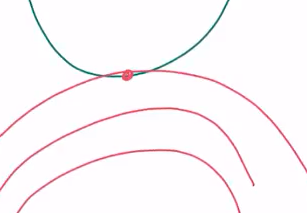
## Formulazione duale

Il termine dualità in questo caso indica la **dualità di Lagrange** ed è un modo per trovare un problema equivalente, al posto di essere espresso in termini di minimizzazione di una funzione è espresso in termini di massimizzazione.

Nel problema precedente noi dovevamo minimizzare f0(x) sotto alcuni vincoli dati da disuguaglianze, come fi(x) <= 0, e da uguaglianze, come gi(x) = 0.

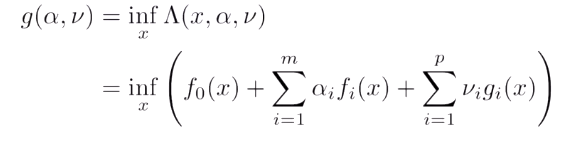


Questa definizione è estremamente generale.



Andremo a disegnare una serie di lower bound (curve rosse), tale cui il massimo sarà il minimo del problema originale.

Dato un problema scritto in questi termini, la funzione duale di Lagrange è definita in questo modo:



g è detta funzione lagrangiana.

Inf rappresenta un potenziamento del minimo (mentre il minimo prevede che l’elemento che troviamo faccia parte dell’insieme a cui x appartiene, mentre inf è il più piccolo elemento dell’insieme)

Es. data una retta di reali, il minimo punto che si avvicina al cerchio non lo possiamo definire perché qualsiasi reale troviamo ce ne sarà sempre uno più piccolo, l’inf invece è proprio il punto verde.



Ed è una funzione g non più funzione della variabile originale x, ma è funzione di altre due variabili.

Inf è la generalizzazione del minimo, si parla del minimo quando il risultato appartiene all’insieme originario, mentre l’infimo può anche essere fuori dall’insieme originario. Se ad esempio si ha la retta dei reali e considera soltanto x >= 0 e si vuole l’inf {x > 0} è 0, mentre il minimo non è definito perché non esiste un numero preciso che minimizza questo insieme.

Abbiamo definito una variabile alfai per ogni vincolo di diseguaglianze e una variabile vi (nui) per ogni vincolo di uguaglianza e poi abbiamo sommato tutto. Dopodiché, prendiamo l’infimo rispetto alle x e chiamo questa funzione g(alfa, nu) con alfa e nu >= 0.

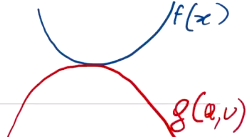
Notiamo che questa funzione g è sempre minore di f0(x), perché la sommatoria delle diseguaglianze sarà di sicuro minore di 0 e la sommatoria delle uguaglianze sarà di sicuro 0 e infine si prende l’infimum.

Immagine che contiene testo, lavagnabianca

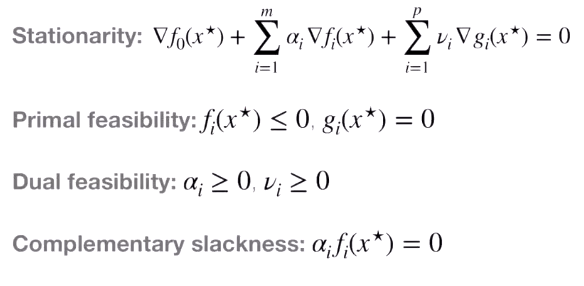
Descrizione generata automaticamente

PS. I vincoli li dobbiamo trasformare nella forma <= 0. Avremo un vincolo per ogni esempio.

Come dice **Stephen Boyd** (vedi corso su ottimizzazione convessa), la dualità può essere pensata come un modo organizzato per costruire dei lower bound completamente non triviali (difficile da pensare) per i problemi di ottimizzazione.



In un problema convesso e sottostà ad alcuni vincoli laschi, il bound dato alla funzione è di solito di solito stretto e vuol dire che max lower bound = min upper bound, e quindi risolvere il problema massimizzando g è come risolvere il problema minimizzando f. quando questo è vero si dice che c’è **dualità forte**, e per esserci ci sono un insieme di condizioni necessarie chiamate **Karush-Kuhn-Tucker (KKT) conditions**.



* **Stazionarietà**: la lagrangiana nel punto ottimale x\* ha gradiente nullo, cioè saremo in un minimo (in una funzione multivariata se siamo in un minimo abbiamo il gradiente nullo, esattamente come nelle funzioni monovariate, se siamo in un minimo avrò derivata = 0)
* **Primal** **feasibility**: nel punto ottimale i vincoli del problema primale valgono, se no avremmo trovato una soluzione che non rispetta i vincoli.
* **Dual feasibility**: valgono i vincoli del problema duale.
* **Complementary slackness**: nel punto ottimale, il prodotto tra le variabili duali e i vincoli a cui sono associate fa 0 (non banale che non dimostriamo).

### Costruiamo il sistema duale

Noi vogliamo riscrivere il nostro problema nella sua forma duale. Dobbiamo trovare l’inf di gamma, per farlo usiamo i mezzi forniti dall’analisi.

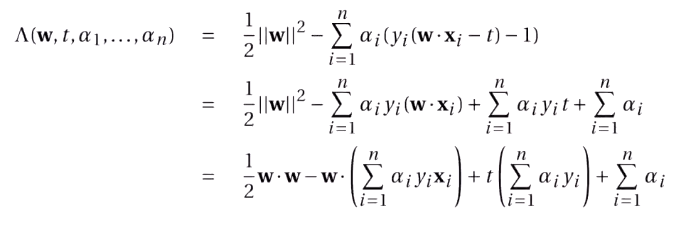
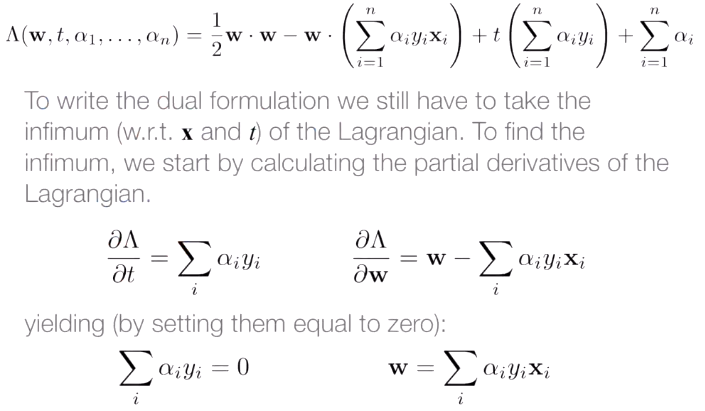


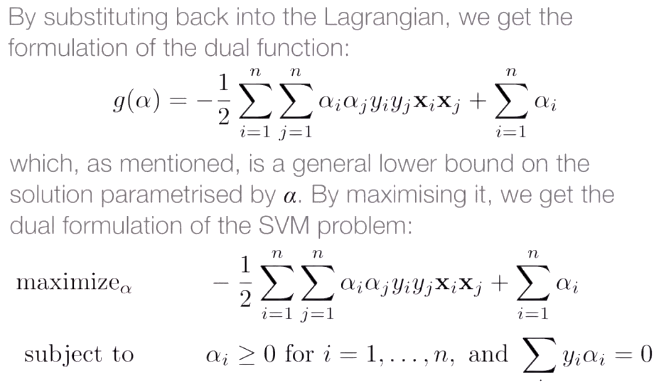
Immagine che contiene testo, lavagnabianca

Descrizione generata automaticamente

Ora deriviamo la seguente espressione prima rispetto a t e poi rispetto a w



Notiamo che w possiamo calcolarlo come combinazioni lineari degli esempi del nostro dataset. Vedremo che non saranno tutti gi esempi a contribuire alla soluzione, ma solo i vettori di supporto.



g non ha nu perché non ci sono vincoli di uguaglianza.

Cosi facendo abbiamo ottenuto il **problema duale**.

Lezione 20

## Dataset non linearmente separabile

Iniziamo con l’analizzare ciò che abbiamo appreso

* Noi abbiamo trovato il problema primale non risolvibile in forma chiusa, per cui bisogna per forza passare a dei solver (cioè, algoritmi numerici).
* Trovando il problema duale abbiamo lo stesso problema, non esiste risoluzione in forma chiusa, per cui bisogna passare all’uso di un solver.
* **Domanda**: perché abbiamo fatto ciò? Per diverse ragioni
  + Dal punto di vista teorico abbiamo imparato una cosa

Immagine che contiene testo, orologio

Descrizione generata automaticamente

Sappiamo quindi che la soluzione al nostro problema (w) si può scrivere come combinazione lineare degli esempi all’interno del nostro dataset (**osservazione non banale**).

* + Le alfa (moltiplicatori di Lagrange) sono positive, e saranno strettamente positive quando il vettore associato è quello di supporto.
  + Nel caso duale la soluzione è scritta guardando soltanto il prodotto scalare dei singoli esempi

Immagine che contiene testo, orologio

Descrizione generata automaticamente

Mentre prima era in termine di una funzione dei singoli esempi

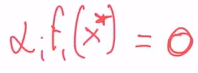
Immagine che contiene testo

Descrizione generata automaticamente

Questo ci dice che, per apprendere la soluzione finale, non ci serve guardare i singoli esempi, ma guardare la **distanza** tra le coppie di esempi. Ciò è **molto importante**, perché ci permetterà di usare il **kernel trick**, per generalizzare le support vector machine a problemi anche non linearmente separabili.

Vediamo perché **soltanto i vettori di supporto hanno moltiplicatori di Lagrange positivi associati a loro**:

* Avevamo detto che essendo il nostro un **problema di dualità forte**,valevano le proprietà di **KKT**. In particolare, a noi interessa la **Complementary slackness condition**, che dice



Ad ora, i nostri vincoli del primale, hanno questa forma qua



Possiamo portare l’1 a sinistra e moltiplicare per il moltiplicatore di Lagrange, ottenendo



Ciò ci dice che siccome l’equazione deve essere uguale a 0,

* + ai deve essere = 0, oppure
  + (yi (wxi – t) – 1) deve essere = 0,
  + O entrambi

Se non siamo su un vettore di supporto, però sappiamo una cosa, cioè che

* + (yi (wxi – t) – 1) sarà sicuramente > 0

E quindi, visto che la condizione ai (yi (wxi – t) – 1) = 0 deve continuare a valere, avremo sicuramente che ai sarà uguale a 0. **Quindi**, **ai potrà essere, ed in generale lo sarà, diverso da 0 soltanto quando siamo sui vettori di supporto**.Per tutti gli altri esempi ɑi è uguale a 0.

Ciò che abbiamo detto, implica una cosa sulla soluzione

Immagine che contiene testo, orologio

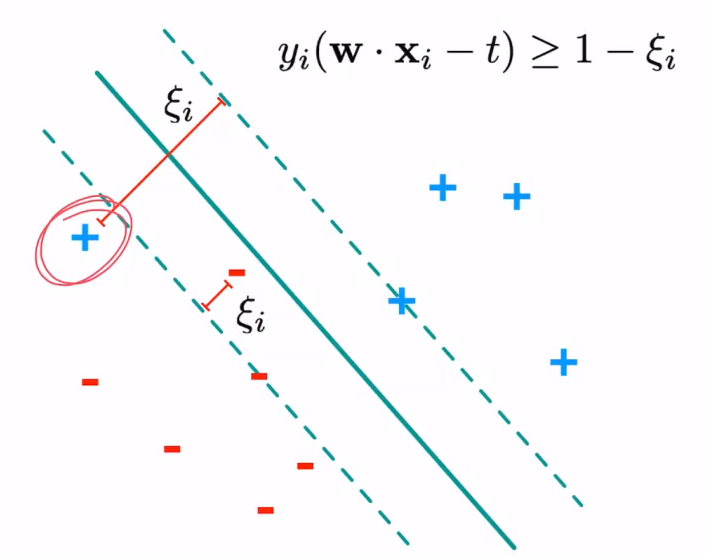
Descrizione generata automaticamente

Ovvero che le uniche somme valide saranno quelle fatte sui vettori di supporto, su tutte le altre ci sarà valore 0. E ciò implica che valutare ciò sarà poco costoso.

Quindi tutto ciò implica che la soluzione è combinazione lineare dei soli vettori di supporto.

### Permettere errore sul margine

Vogliamo permettere errore sul margine, per sopperire agli **errori di campionamento**. Oltre a ciò ci permetterà di “risolvere” il problema ammettendo errori all’interno del margine, quando il problema è non linearmente separabile.



Il nostro problema primale era

Immagine che contiene testo

Descrizione generata automaticamente

Abbiamo imposto un **vincolo** **hard**, ovvero se non c’è soluzione al problema allora questi viene lasciato perdere. **Vogliamo trasformalo in un vincolo soft**. Per farlo allora creiamo il vincolo

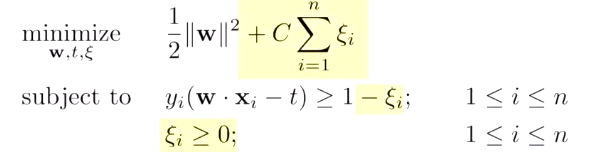


Dove in pratica aggiungiamo una psii (una per ogni esempio del dataset) con l’idea che sia la più piccola possibile. Le psii verranno chiamate **slack variables**.

PS. Psi ci dice quanto sto violando il margine funzionale

**Questa però non sarà l’unica modifica perché se altrimenti questo vincolo non vincola nulla (perché psii può crescere all’infinito)**

**Quello che facciamo è aggiungere un termine alla funzione obiettivo** **in cui diciamo che quello che vogliamo minimizzare sono i parametri w, t e ora anche psi.**



Creando quindi un nuovo modello, chiamato **SVM with soft margin**.

Nella funzione obiettivo c’è anche la somma di una costante C moltiplicata per la somma di tutte le variabili psii. Ogni volta che psii cresce un po' l’obiettivo di minimizzare la funzione principale viene penalizzata un pochino.

La somma di c ecc. aggiunta alla funzione è molto simile a una regolarizzazione a norma 1 (regolarizzazione lasso) con la differenza che non vi è il valore assoluto su psi ma che tanto non serve per il vincolo psii >= 0. Lasso tende a mettere a 0 il più alto numero possibile di variabili psii.

Le variabili psii si chiamano **slack variables**, mentre la costante C si chiama **costante di complessità del sistema**.

La costante C è un parametro definito dall’utente che mi dice quanto voglio penalizzare le psii.

Cosa succede quando metto C molto alto o molto basso?

* Ponendo un C molto alto si tornerebbe al valore iniziale perché per ottimizzare il minimizzatore metterebbe tutte le psii = 0.
* Ponendo C = 0 si otterrebbe il minimo di w è il vettore nullo 0.

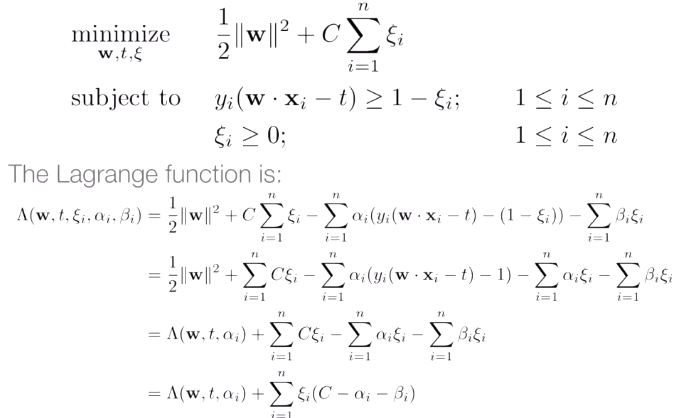
PS. Se penalizzo tanto le slack variables tendono a essere molto piccole, molto grandi altrimenti.

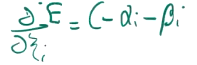
#### Formulazione duale

Se ammettiamo più errori sul margine avremo bisogno di meno vettori di supporto (perché gli altri saranno oltre il margine) e quindi possiamo dire che C sta controllando la complessità del sistema e per questo si chiama costante di complessità del sistema.

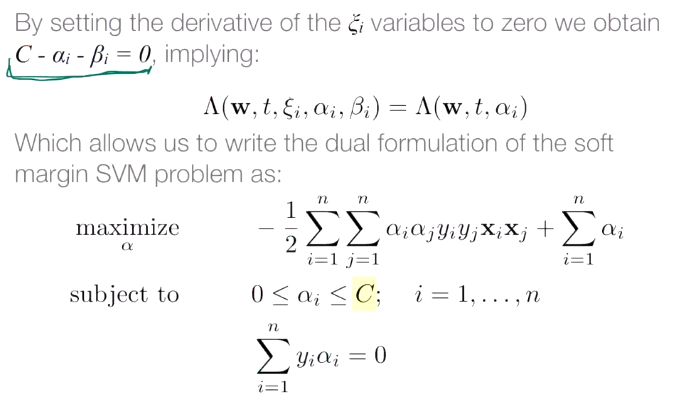
Spieghiamo perché è vero ciò.

Per farlo dobbiamo passare alla versione duale del problema. È molto semplice perché riutilizziamo ciò che abbiamo fatto la volta scorsa. L’unica cosa che dobbiamo fare è cambiare la funzione obiettivo (che diverrà ½||w||^2 + C ...)





Se C, ai e bi sono uguali a zero allora la Lagarngiana è uguale a quella che avevamo per l’svm hard



Il vincolo è stato trovato perché, sapendo

* ai >= 0
* bi >= 0

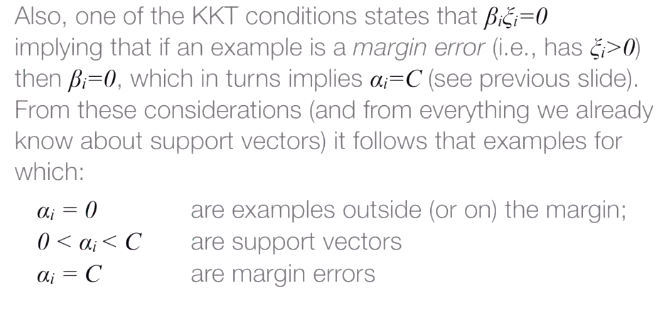
Avendo C – ai – bi = 0, spostando sto dicendo

* ai = C – bi 🡪 ai <= C

quindi, unendo i vincoli avrò

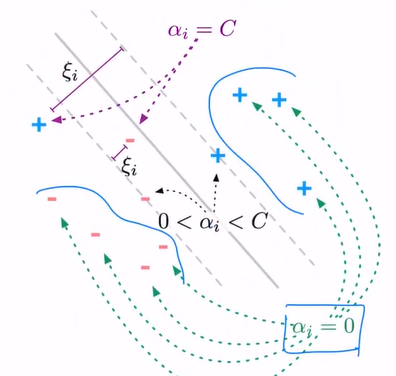
0 <= ai <= C

**Ha come unica differenza rispetto al problema primario che le ɑi non possano crescere più di C.**



PS. In scikitlearn come vettori di supporto indica anche i vettori che commettono errori, con ai = c. Questo ha senso se pensiamo come vettori di supporto non solo quelli con margine funzionale uguale a 1, ma quelli che ci aiutano a costruire la soluzione (quelli con ai = C contribuiscono alla soluzione)

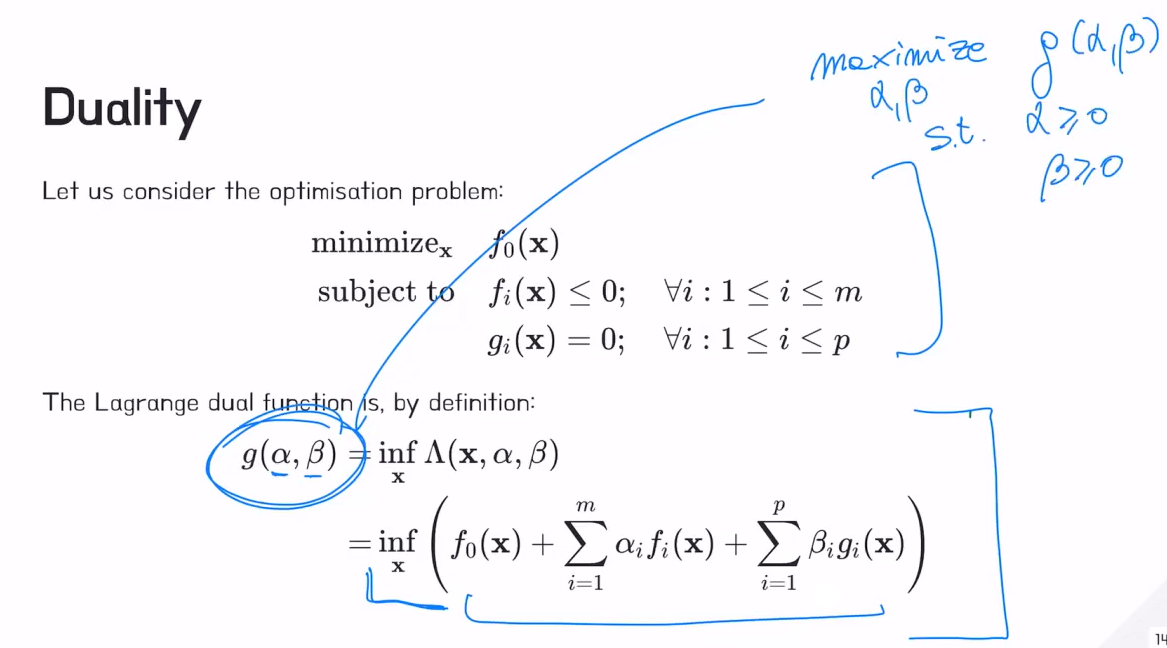
Ciò che abbiamo detto si può vedere graficamente come segue



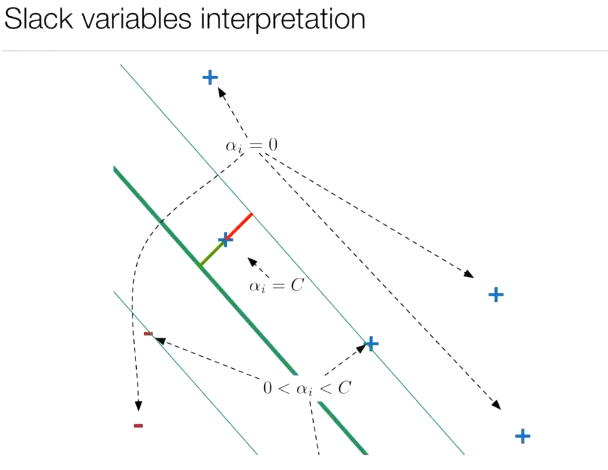
Ricapitolando chiamiamo vettori di supporto quei vettori con margine funzionale >= 1 e errori sul margine quei vettori con margine funzionale < 1.

Tips forma duale:

* vediamo come si costruisce la formulazione del problema duale



Notare che è coinvolta la minimizzazione precedente (inf), ed è in funzione di quella minimizzazione che pongo le derivate = 0, perché devo trovare il minimo di quelle funzioni

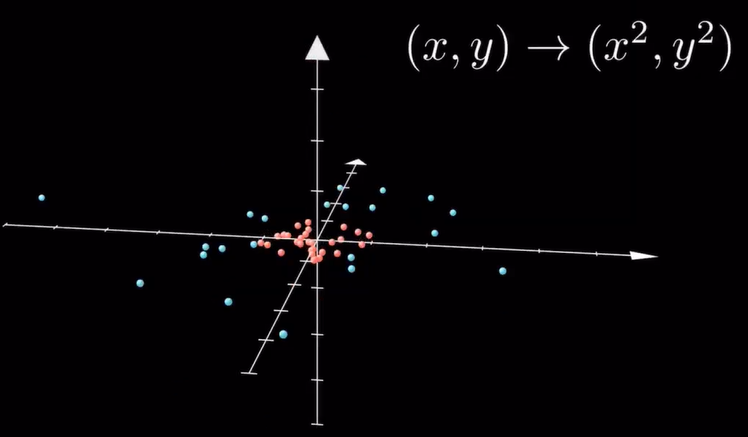


# Kernels trick

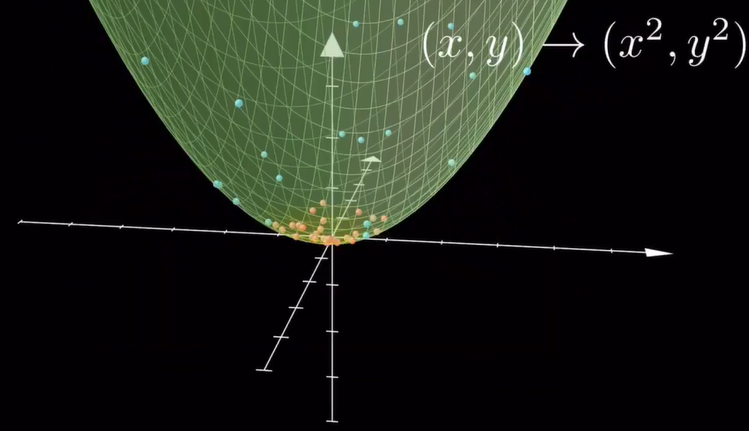
Perché parliamo di kernel? sono lo strumento principale per quello che viene chiamato trucco del kernel ed è un modo che precede le SVM per estendere modelli lineari a problemi non lineari.

Non viene usato solo per l’SVM, però l’SVM ha reso a renderlo molto popolare.

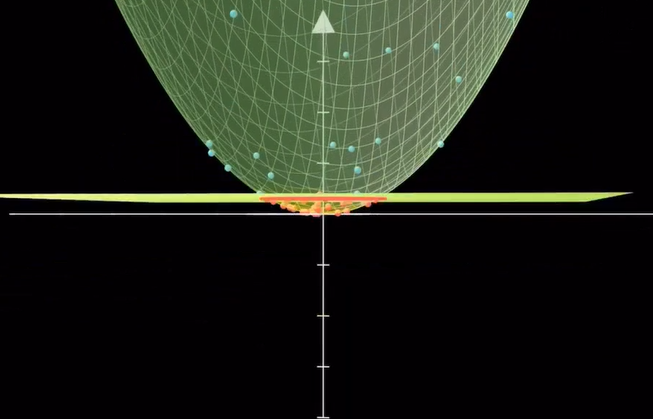
L’idea è di mappare lo spazio di origine non lineare in un nuovo spazio con più dimensioni. Ciò ci permette di rendere il problema linearmente separabile.



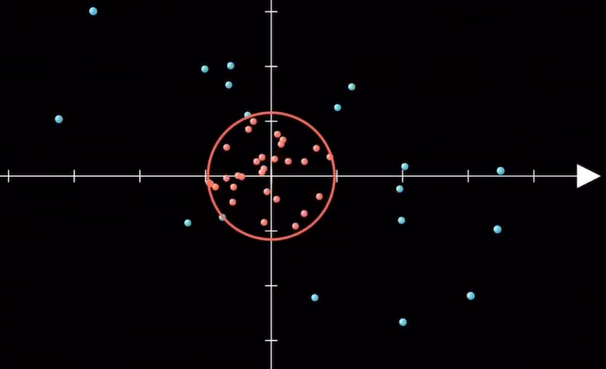
In questo grafico abbiamo un problema non lineare, che andremo a mappare su un paraboloide:



Cosi da poter dividere gli esempi attraverso l’uso di un iperpiano.



La soluzione, nello spazio originale corrisponderà ad un cerchio.



Ora cerchiamo di formalizzare l’idea

Immagine che contiene testo

Descrizione generata automaticamente

* Input: V x V sono due vettori e sono una coppia di esempi (es. x1 e x2) dello spazio originale.
* Output: restituisce un numero reale.

Il punto è che per poter applicare il kernel trick, e quindi usare la funzione K, deve esistere un mapping fi, che mappa l’esempio in un nuovo spazio F (**spazio di Hilbert**), tale per cui, fare K(x, x’) equivale a fare il **prodotto** **interno** tra gli esempi mappati in F.

PS.

* **Prodotto interno**: generalizzazione del concetto di prodotto scalare (possiamo vederlo come una misura di distanza tra gli esempi). La differenza tra prodotto scalare e interno è che il prodotto scalare assume che la base dello spazio su cui sto lavorando abbia dei vettori ortonormali. La cosa importante è che ha delle formule per essere calcolato. Il prodotto interno invece non ha una formula specifica perché sappiamo che la moltiplicazione tra vettori è un prodotto interno quando valgono queste tre proprietà qua

Immagine che contiene testo

Descrizione generata automaticamente

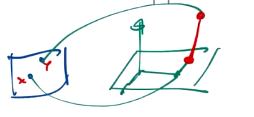
PS. Siccome è simmetrico è anche lineare per il secondo argomento

* **Spazio di Hilbert**: esiste una gerarchia degli spazi vettoriali, che viene fatta in base alle proprietà degli spazi. Uno spazio di Hilbert **è uno spazio vettoriale completo** (non ci sono buchi in questo spazio, come i reali) **ed è equipaggiato di prodotto interno**. Si tratta di uno degli spazi più generali possibili

Vediamo un esempio di kernel trick con uno spazio bidimensionale con x e y



Abbiamo un nuovo spazio diverso da quello che immaginiamo essere tridimensionale in cui vengono mappati x e y



E dopodiché calcola il prodotto interno tra i due, che è una misura che indica quanto sono simili i due vettori, come lo è il prodotto scalare. detto è la generalizzazione del prodotto scalare.

### Motivazioni

Vediamo i vantaggi che ci da il kernel trick

Immagine che contiene testo

Descrizione generata automaticamente

PS.

* Per via teorica, esistono delle **condizioni** che ci dicono che una funzione è un Kernel, che ci permette di non dover pensare al mapping nello spazio di Hilbert.
* Oltre a ciò, esistono delle **proprietà composizionali** che ci dicono come combinare i kernel tra di loro per ottenere un nuovo kernel, senza dover eseguire il mapping.

### Definizione di Kernel importanti

#### Kernel polinomiale

Immagine che contiene testo

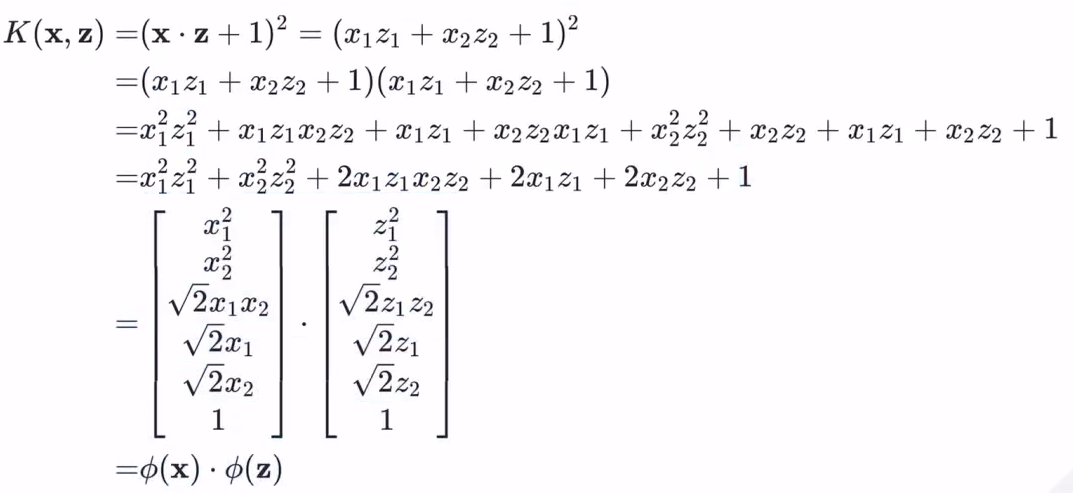
Descrizione generata automaticamente

Permette di apprendere con una funzione lineare concetti che corrispondono a polinomi di grado d.

Immagine che contiene testo

Descrizione generata automaticamente

Da notare che in questo caso non c’è il mapping.



Implicitamente sta calcolando la proiezione dei vettori in questione nello spazio polinomiale, e poi calcola il prodotto scalare tra i vettori coinvolti.

x e z hanno coordinate (x1, x2) e (z1, z2) e stanno x2.

Il mapping che viene fatto corrisponde ad usare i nostri vettori come se fossero dei polinomi.

Se, ad esempio, fossimo in R1, la nostra verrebbe mappata come segue:

Immagine che contiene testo, lavagnabianca

Descrizione generata automaticamente

Andando ad aggiungere dei coefficienti alle varie componenti otterremo un polinomio

Immagine che contiene testo, lavagnabianca

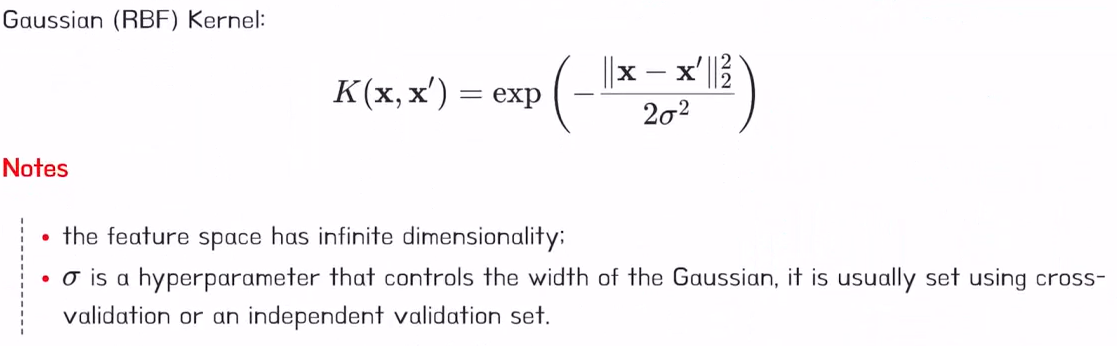
Descrizione generata automaticamente

Cambiando alfa, beta e gamma vado a selezionare un qualsiasi polinomio di secondo grado.

I coefficienti corrisponderanno alla soluzione del problema (i w che cercavamo nel problema primale), andando a cercare nello spazio dei polinomi.

#### Kernel Gaussiano

Si tratta del kernel più importante di tutti



**RBF**: radial basis function

È importante perché ci permette di apprendere funzioni più o meno di qualsiasi tipo (è universale).

Funziona un po’ come KNN, cioè, l’etichetta che associo ad un esempio dipende dagli esempi che lo circondano. Più è vicino l’esempio che lo circonda, più l’etichetta di quell’esempio influenzerà al mia classificazione.

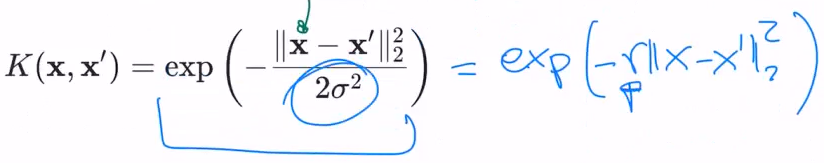
Immagine che contiene lavagnabianca

Descrizione generata automaticamente

Siccome + è più vicino al pallino verde contribuirà maggiormente alla scelta della classe.

Il contributo come cambia? In base la funzione RBF, si tratta di una **gaussiana** centrata sull’esempio. Il contributo decresce esponenzialmente più mi allontano dall’esempio

Si può anche trovare scritto come



**Importante**: lo spazio in cui vado a proiettare gli esempi è uno spazio di Hilbert a infinite dimensioni.

Quindi, se non avessi il modo di calcolarlo usando il Kernel, ma cercassi di fare la proiezione non saremmo in grado di trovare la soluzione (perché è su infinite dimensioni).

Come si calcola **sigma quadro** (o gamma)? Usando la cross-validation.

#### Altri Kernel

Immagine che contiene testo

Descrizione generata automaticamente

Il kernel è una misura di similarità in uno spazio trasformato. Se so costruire questa misura sulle feature che conosco, questa misura corrisponde probabilmente a qualche Kernel.

Uno Kernel usato in contesto bioinformatico è quello qua sopra (si nota che viene usato su feature categoriche).

### Intuizione

Immagine che contiene testo

Descrizione generata automaticamente

Il prodotto scalare è anche una misura di similarità, perché se x è simile a x’, l’angolo teta sarà molto vicino a zero, quindi, il coseno sarà prossimo a 1, e quindi x e x’ verranno reputati molto simili (si avrà un valore molto grande). Se sono molto diversi si avrà un valore molto piccolo oppure negativo.

Noi possiamo anche voler usare una misura di similarità senza però saper mappare in uno spazio i due esempio. Come facciamo a capire se si tratta di un Kernel? Bisogna controllare se valgono le **condizioni di Mercer**, se valgono allora si tratta di un Kernel.

Immagine che contiene testo

Descrizione generata automaticamente

Le condizioni di Mercer dicono che se io ho una **funzione** **K**, che non so se sia un kernel o meno (è la mia funzione di similarità). Se è vero che per ogni sequenza finita di vettori x1, …, xn, con n arbitrario, la **matrice M**, che costruisco mettendo in posizione **Mij** il risultato della funzione di similarità applicato sul vettore i e su j (**Mij = K (xi, xj)**), dove M viene anche detta **matrice di Gram**. Se M è **simmetrica** e **semi-definita positiva**, allora K è un Kernel, e viceversa.

PS. Una matrice M è semi-definita positiva, se per ogni Z, ZTMZ >= 0

Dimostriamo solo un verso di questo teorema

* **Se K è un kernel => M è simmetrica e semi definita positiva**
  + M simmetrico:

Siccome sappiamo che il prodotto interno è **simmetrico**, **Mij = <o(xi), o(xj)> = <o(xj), o(xi)> = Mji**, per ogni i, j

* + M semi definita positiva:

Immagine che contiene testo

Descrizione generata automaticamente

PS.

* + - Posso portare zi e zj dentro il prodotto interno perché so che è lineare.
    - Arriviamo all’ultimo passaggio a calcolare il prodotto interno tra un vettore e se stesso, e ciò sarà >= 0, per definizione di prodotto interno.

### Costruire un nuovo kernel

Abbiamo detto che i **kernel sono composizionali**

Immagine che contiene testo, tavolo

Descrizione generata automaticamente

Ciò ci permette di trattare in modo facile dataset non numerici.

Lezione 21

Esercitazione:

Abbiamo un data set fatto a mano la cui rappresentazione sul piano cartesiano è linearmente separabile.

Vogliamo applicare l’SVM

Iniziamo importando la libreria scipy, al cui interno è presente una funzione minimize presente dentro il modulo optimize, che si può raggiungere scrivendo

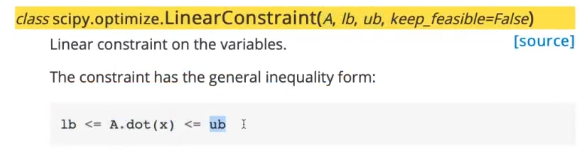
Scipy.optimize.minimize()

Che minimizza una data funzione sotto determinati vincoli.

L’obiettivo è riscrivere il nostro problema in una forma facilmente manipolabile da scipy. Iniziamo vedendo la documentazione di scipy.

La documentazione ci dice che per poter minimizzare bisogna passare al metodo

1. Una funzione da minimizzare f(x, \*args), dove x è il vettore dei pesi e \*args sono eventuali altri argomenti della funzione. La funzione deve restituire un numero che ci dice quanto siamo lontani dalla soluzione . la funzione calcolerà ½||w||2.
2. Un punto x0 da cui iniziare
3. Passare dei vincoli (constraints), come una lista (i quali possono essere lineari o non lineari), nel nostro caso ci interessano quelli lineari, che sono fatti nel seguente modo:
   1. Viene definito attraverso la matrice A e poi questa viene moltiplicata per il vettore de pesi w e poi la soluzione viene messa in mezzo tra un lower bound e un upper bound



Un dataset è in **coordinate omogenee** quando abbia aggiunto una colonna di 1 alla matrice x