FONDAMENTI DI SCIENZA DEI DATI

APPUNTI A CURA DI: RICCARDO LO IACONO

Università degli studi di Palermo

a.a. 2023-2024

Documento in WIP

Indice.

1 Pipeline di analisi dei dati **1**
1.1 Integrazione dei dati . . . 1
1.2 Trasformazione dei dati . . . 1
1.3 Riduzione della dimensionalità . . . 2

2 Stimatori e predittori **3**
2.1 MSE e bias-variance decomposition . . . 4

3 Tipologie di machine learning **5**
3.1 Apprendimento supervisionato . . . 5
3.2 Apprendimento non-supervisionato . . . 5

4 Riduzione della dimensionalità **6**
4.1 SVD: singular value decomposition . . . 6
4.2 Regressione . . . 8

5 Clustering **10**
5.1 Tipologie di clustering . . . 10
5.2 Misura della qualità di clustering . . . 10
5.3 K-mean clustering . . . 11

6 Classificatori **12**
6.1 SVM: support vector machine . . . 12

– 1 – Pipeline di analisi dei dati.

Il processo di analisi dei dati parte banalmente dall'ottenimento stesso dei dati, cosa che può essere effettuata da database, file di log, o altre fonti. In ogni caso quando si lavora con i dati si deve prestare attenzione a due problemi, uno legato alla struttura dei dati, l'altro alla numerosità degli stessi.

In generale comunque, prima di operare con i dati si procede ad una fase di pulizia. Questa è atta a gestire

- *valori nulli o erronei:* è ovvio che la presenza di valori errati possa influire negativamente sul modello;
- *valori duplicati:* se non gestiti potrebbero portare ad un cattivo addestramento;
- *struttura dei dati:* è opportuno uniformare la struttura con cui i dati sono rappresentati.

Una volta effettuata la fase di pulizia, si può passare alla parte principale dell'analisi dei dati: l'estrazione delle feature. Dal un punto di vista strettamente formale, queste rappresentano caratteristiche valorizzabili dei dati, utili alla risoluzione di un qualche problema. Una volta estratte le feature, queste sono adattate per essere utilizzate da un modello di machine learning, ed eventualmente combinate per definire caratteristiche supplementari.

– 1.1 – Integrazione dei dati.

Fase essenziale della analisi dei dati, *l'integrazione* è atta alla gestione di dati provenienti da fonti diverse. In generale, in questa fase ci si occupa di

Si assume che dati provenienti da fonti diverse, avranno anche una rappresentazione diversa.

- uniformare la struttura dei dati;
- gestire le inconsistenze: ossia dati che, sebbene riferiti ad uno stesso evento, sono discordi tra loro.

– 1.2 – Trasformazione dei dati.

Una volta eseguita la fase di integrazione, i dati sono sottoposti ad operazioni di trasformazione utili ad ottimizzare l'addestramento del modello. Tra queste si considerano

- *lo smoothing:* si tratta di un'ulteriore fase di pulizia dei dati, con l'obbiettivo di eliminare del rumore eventualmente permasto;
- *l'aggregazione:* i dati sono tra loro combinati, per poter descrivere concetti più generici;
- *la normalizzazione:* per ridurre il numero di valori assunti dai dati, si procede a ridurre l'intervallo che questi possono assumere.

– 1.2.1 – Normalizzazione.

Sebbene esistano decine di normalizzazione diverse, di interesse risultano

- *la normalizzazione max-min:* si tratta di una normalizzazione dipendente dai valore di massimo e minimo dei dati. Nello specifico, assunti v i dati, v' i dati normalizzati, MAX_0 e min_0 rispettivamente i valori di massimo e minimo iniziali, e assunti inoltre MAX e min gli estremi del nuovo range, segue

$$v' = \frac{v - min_0}{(MAX_0 - min_0)}(MAX - min) + min$$

In generale, poiché si preferisce operare con valori in $[0,1]$, $Max = 1, min = 0$.

- *la normalizzazione z-core:* è una normalizzazione dipendente dalla media e dalla deviazione standard dei dati. Nello specifico, posti v i dati, v' i dati normalizzati, μ la media e σ la deviazione standard, si ha

$$v' = \frac{v - \mu}{\sigma}$$

- *la normalizzazione decimale:* si normalizza dividendo per una potenza di 10 tale che, a seguito della normalizzazione, i valori cadano nel range $(0,1)$. Formalmente

$$v' = \frac{v}{10^k}, k = \min_{k \in \mathbb{N}} \{||v'|| \leq 1\}$$

– 1.3 – Riduzione della dimensionalità.

In generale, quando si opera con i dati, questi risultano essere di grandi quantità. Poiché gestirli tutti risulta complesso, si preferisce molto spesso ridurre al necessario i dati da utilizzare. Per far ciò si utilizza una delle diverse tecniche di riduzione della dimensionalità, tra queste

- *l'aggregazione:* simile a quella descritta nella fase di trasformazione;
- *la selezione degli attributi minimi:* si procede ad estrarre un sottoinsieme dei dati, tali che questi siano sufficienti a rappresentare l'intero insieme. Di questa tecnica si distinguono
  - *la selezione in avanti:* a partire dall'intero set di dati S, si costruisce un sottoinsieme S' estraendo di passo in passo un elemento da S e inserendolo in S', se e solo se questi migliora la qualità dei dati in S'.
  - *la selezione all'indietro:* da S si rimuove un elemento, se a seguito di una sua eventualmente estrazione, la qualità degli elementi in S risulta massimizzata.

– 2 – Stimatori e predittori.

L'intero processo di machine learning è basato sulla *teoria dell'apprendimento statistico*. Secondo tale teoria, i dati che rappresentano un qualunque fenomeno, possono essere visti come appartenenti ad una qualche distribuzione di probabilità ignota; e per tale motivo si può assumere che essi siano tra loro indipendenti ed equiprobabili, concludendo pertanto che il valore atteso dei dati di addestramento e quello dei dati di training coincida. A seconda del tipo di machine learning adoperato (si veda *Sezione 3*), l'intero processo fa uso di *stimatori* o di *predittori*.

Definizione: uno stimatore, dal punto di vista statistico, è una funzione che, in funzione dei dati, permette di stimare una quantità/funzione interessante dei dati.

Di questi, supposto Θ_n uno stimatore, Θ il valore da stimare, si distinguono

- *gli stimatori polarizzati:* ossia stimatori tali che

$$\mathbb{E}(\Theta_n) - \Theta \neq 0.$$

  Cioè, lo stimatore commette un certo errore nell'approssimare Θ;

- *gli stimatori non-polarizzati:* si tratta di stimatori tali che

$$\mathbb{E}(\Theta_n) - \Theta = 0.$$

  Ossia, nell'approssimareΘnon si commette alcun errore.

Osservazione. Se si verifica che

$$\lim_{n\to\infty} \mathbb{E}(\Theta_n) = \Theta$$

si dirà che Θ_n è uno stimatore asintoticamente non-polarizzato.
Ulteriori caratteristiche degli stimatori sono la correttezza e la coerenza, nello specifico

$$\Theta_n \text{ corretto} \iff \mathbb{E}(\Theta_n) = \Theta$$
$$\Theta_n \text{ coerente} \iff \lim_{n\to\infty} \mathrm{Var}(\Theta_n) = 0$$

Ovviamente un buon stimatore sarà sia corretto che coerente.

Definizione: un predittore è un algoritmo della forma

$$y = f(x) + \varepsilon$$

che permette di stimare la funzione f sulla base dei dati di addestramento x.

Segue dalla definizione che, affinché y sia buona, l'errore ε deve essere minimo.

– 2.1 – MSE e bias-variance decomposition.

Si è parlato di stimatori e predittori, rimane dunque da discutere come determinare la "bonta" degli stessi. Sebbene ne esistano altre, la tecnica che risulta di interesse è l'*MSE*. Questi nel caso degli stimatori è definito come

$$MSE(\Theta_n) = \mathbb{E}\left[(\Theta_n - \Theta)^2\right] = \underbrace{\mathbb{E}(\Theta_n)^2 + \Theta^2 - 2\,\mathbb{E}(\Theta_n)\,\Theta}_{\text{Bias}(\Theta_n)^2} + \underbrace{\mathbb{E}\left(\Theta_n^2\right) - \mathbb{E}(\Theta_n)^2}_{\text{Var}(\Theta_n)}$$

Osservazione. Tale decomposizione prende il nome di *bias-variance decomposition.*

Supposto infine $y = f(x) + \varepsilon$ un predittore, e $\tilde{f}$ l'approssimazione di f sulla base dei dati, si dimostra che

$$MSE(y) = \mathbb{E}\left[\left(y - \tilde{f}\right)^2\right] = \cdots = \text{Var}\left(f(x)\right) + \text{Var}\left(\varepsilon\right) + \text{Bias}\left(f(x)\right)^2$$

– 3 – Tipologie di machine learning.

Sia assunto

$$y = f(x, z)$$

un algoritmo di machine learning. Da un punto di vista matematico, y deve fornire una valutazione statistica della relazione tra l'input e l'output dell'algoritmo. Formalmente, f è un modello di ML e (x, z) i suoi parametri.

– 3.1 – Apprendimento supervisionato.

Si tratta di una tipologia di machine learning in cui i dati di addestramento sono "etichettati": cioè presentano un campo che descrive la classe di appartenenza degli stessi. Tra questi si distinguono

Entrambe le categorie saranno discusse nelle sezioni a seguire.

- gli algoritmi di regressione;
- gli algoritmi di classificazione.

In generale, si utilizzano quando l'obbiettivo è quello di separare in classi i dati. Si osserva però che affinche l'addestramento possa definirsi "buono", l'algoritmo deve minimizzare l'errore relativo i dati di addestramento: il cosiddetto *training error*, e l'errore relativo i dati di test: il cosiddetto *test/generalization error*. In fine, per quanto detto e quanto discusso in *Sezione 2*, si deve fare attenzione

- all'*over-fitting:* ossia un fenomeno per cui il modello si è troppo adattato ai dati, non riuscendo dunque a generalizzare;
- all'*under-fitting:* fenomeno opposto all'over-fitting, è una condizione in cui il modello ha appreso poco dai dati, pertanto non ha le capacità sufficienti a generalizzare.

– 3.2 – Apprendimento non-supervisionato.

È un caso di machine learning in cui i dati non sono "etichettati". Tra questi, gli algoritmi di interesse sono

Come per gli algoritmi di ML supervisionato, entrambe le categorie saranno discusse nelle sezioni a seguire.

- quelli di riduzione della dimensionalità;
- quelli di clustering.

In tal senso, l'obbiettivo è quello di determinare similarità tra i dati.

– 4 – Riduzione della dimensionalità.

Come anticipato in *Sezione 3*, la riduzione della dimensionalità è una tecnica utilizzata dalla'apprendimento non-supervisionato. L'idea alla base è quella di ridurre la quantità dei dati che si deve analizzare, riducendoli al minimo. In generale, alla base di una qualsiasi tecnica di riduzione vi è la SVD, nel seguito descritta.

– 4.1 – SVD: singular value decomposition.

Tecnica fondamentale per la riduzione della dimensionalità, permette di decomporre la matrice rappresentanti i dati in tre sotto-matrici. Nel dettaglio, questa è definita come segue.

Definizione: sia $X \in \mathbb{R}^{n,m}$, allora

$$\exists U, \Sigma, V : X = U\Sigma V^T, \forall X \in \mathbb{R}^{n,m}$$

ove $U \in \mathbb{R}^{n,n}, \Sigma \in \mathbb{R}^{n,m}$ e $V \in \mathbb{R}^{m,m}$.

Ciò equivale a dire, da un punto di vista numerico, che

$$X = \sum_{i=1}^{n} \sigma_i u_i v_i^T$$

ove $u_i \in U, v_i \in V$ e $\sigma_i \in \Sigma, \sigma_i \geq \sigma_j, \forall i < j < n$.

Si definiscono i σ_i valori singolari.

Geometricamente segue che

- Σ è una matrice diagonale, nello specifico è data dagli autovalori di XX^T e X^TX;
- rispettivamente, U e V, sono le matrici date dagli autovettori di XX^T e X^TX.

Dalla definizione si è detto che la SVD esiste comunque scelta la matrice X. La dimostrazione di ciò è sostenuta dal seguente teorema.

Teorema. Sia $C = X^TX \in \mathbb{R}^{m,m}$, allora C è diagonale, simmetrica e definita positiva.

Dimostrazione: da dimostrazione segue banale dal *Teorema di decomposizione spettrale*. Per esso si ha che $C = VTV^T$, con $T = \text{Diag}(\lambda_1, \ldots, \lambda_m)$ e $r = \text{Rank}(X)$. Posto allora $\sigma_i = \sqrt{\lambda_i}$, segue che

$$\Sigma = \begin{pmatrix} \text{Diag}(\sigma_1, \ldots, \sigma_r) & 0_{r,(m-r)} \\ 0_{r,(m-r)} & 0_{m-r,m-r} \end{pmatrix}$$

e ponendo inoltre

$$u_i = \frac{X}{\sigma_i} v_i$$

che si dimostrano ortogonali, segue, completando a base

$$U = \begin{bmatrix} u_1 & \cdots & u_r & u_{r+1} & \cdots & u_n \end{bmatrix}$$

– 4.1.1 – Low-Rank approximation.

La Low-Rank approximation è una tecnica appartenente alle cosiddette *budget* SVD, dove per "budget" si intende il fatto che dimensioni delle matrici sono ridotte a un certo k. Più rigorosamente, la low-rank approximation è definita come segue.

Definizione: sia $X \in \mathbb{R}^{n,m}$ e sia $r = \text{Rank}(X)$, se

$$X_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

è una SVD di X, allora scelto un $k < r$ è possibile definire una SVD ridotta ai primi k termini. Ossia,

$$X_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T.$$

Si definisce X_k matrice di miglior approssimazione di X di rango k.

Cioè sostanzialmente, X_k è la matrice di rango k che, fra tutte le matrici di rango k, approssima meglio X.

– 4.1.2 – PCA: principal component analysis.

La PCA (*principal component analysis*), è una tecnica con funzionalità duplice: essa permette di rilevare strutture (bi-, tri-dimensionali) all'interno dei dati; inoltre permette di stabilire eventuali colinearità tra i dati. Questa, assunta B la matrice rappresentante i dati, procede a considerare una nuova matrice C contenente le covarianze reciproche tra i dati. Più precisamente: sia B una matrice rappresentante un data-set, per la geometria esiste sempre B^T. Sia μ_j la media della colonna j di B. Allora sottraendo alle colonne di B le rispettive medie, si può definire

$$C = \frac{B^T \times B}{n-1}$$

contenente per ogni i, j la covarianza tra gli elementi delle colonne i e j. Segue da ciò che la diagonale di C conterrà unicamente le varianze.

Si dimostra che calcolando C in tal modo, i suoi autovettori sono ortogonali. Da ciò si può concludere che se alcuni autovalori descrivono bene la varianza del sistema, allora si possono utilizzare i relativi autovettori per descrivere l'interno sistema. Si conclude osservando che, per le sue caratteristiche, la SVD fornisce un approccio numerico robusto al calcolo delle componenti.

Gli autovettori di C sono detti componenti principali.

Ricordando che le osservazioni della matrice possono essere visti come vettori di uno spazio $n-dimensionale$, seguono le seguenti definizioni.

Definizione: le nuove coordinate dei vettori corrispondenti le osservazioni, sono definiti *scores*.

Definizione: i pesi che definiscono le componenti principali sono detti *loadings*.

– 4.2 – Regressione.

Il concetto di regressione si basa su quello di *curve fitting*: assunti x i dati indipendenti, y la variabile dipendente e β parametri sconosciuti, tramite funzioni semplici cerca di minimizzare l'errore di adattamento ai dati. In generale, curve fitting è formulato come soluzione di un sistema $Ax = b$.

Parlando di regressione, questa è utilizzata quando si ricerca un modello statistico che, a partire da variabili casuali $X_1, \dots, X_k$, determini una funzione dipendente Y. In tal senso, obbiettivo della regressione è determinare la forma della relazione funzionale tra le variabili.

Si distinguono du classi di regressione: lineare e non.

– 4.2.1 – Regressione lineare.

Come suggerito dal nome, è una tipologia di regressione utile quando, considerando il grafo di dispersione, i dati sono grossomodo distribuiti lungo una linea retta. Si distinguono:

- *il caso semplice:* y dipende da un'unica variabile x;
- *il caso multiplo:* y è dipendente da due o più variabili $x_1, \dots, x_k$.

Considerando unicamente il caso semplice (il caso multiplo si riduce ad una banale estensione), la relazione tra le variabili si può esprimere come

$$y = a + bx + \varepsilon$$

da ciò, calcolare gli y_i si riduce a calcolare

$$y_i = a_0 + b_1 x_i + \varepsilon_i$$

Ciò, comporta dover calcolare a_0, b_1, in generale realizzato con il metodo dei minimi quadrati.

Metriche di regressione lineare.

Se ne distinguono essenzialmente tre:

- **maximum error:** $\max\limits_{k\in\{1,\dots,n\}} \left\{ \left| f(x_k) - y_k \right| \right\}$;
- **mean absolute error:** $\frac{1}{n} \sum\limits_{k=1}^{n} \left| f(x_k) - y_k \right|$;
- **mean absolute error:** $\sqrt{\frac{1}{n} \sum\limits_{k=1}^{n} \left| f(x_k) - y_k \right|^2}$;

– 4.2.2 – Regressione non lineare.

Si tratta di un tipo di regressione in cui la funzione $f(x)$ utilizzata, risulta essere un polinomio di grado superiore o una funzione trascendentale (seni, coseni, ecc.). Ossia, si ha una funzione del tipo

$$f(x) = a + b_1 x + b_2 x^2 + \dots + \varepsilon$$

genericamente di grado non superiore a quattro, oppure

$$f(x) = a e^{bx} + \varepsilon$$

Gradient descent.

Nel caso di curve non lineari, si dimostra che il fitting porta alla risoluzione di equazioni non lineari, che risultano complesse da risolvere. Per tale ragione si preferisce utilizzare funzioni convesse, per la quale vi è garanzia di convergenza. In tali casi si applica il metodo di discesa del gradiente, il quale permette iterativamente di minimizzare le funzioni differenziabili.

K-Fold cross-validation.

Sia posto X un data-set; l'idea è quello di dividere X in k sotto-set, per ognuna di esse addestrare il modello con le restanti $k-1$ partizioni, e utilizzare la k-esima come train-set.

– 5 – Clustering.

Il concetto di clustering fa riferimento ad una classe di algoritmi di machine learning non-supervisionato, generalmente utilizzati per il data-mining o per la ricerca degli outliers[1].

In generale, a decidere i cluster è l'utente, scelta che deve però minimizzare la seguente funzione

$$O = \sum_{i=1}^{n} \min_{j} \left\{ dist\left(X_i, Y_j\right) \right\}$$

ove Y_j è il rappresentante del cluster C_j. Più nel dettaglio la definizione di clustering è la seguente.

Definizione: sia $X \in \mathbb{R}^n$, allora un *m-clustering* di X è un partizione $C_1, \dots, C_m$ tale che

- $C_i \neq \emptyset, \forall i \in \{1, \dots, m\}$;
- $\bigcup_{i=1}^{m} C_i = X$;
- $C_i \cap C_j = \emptyset,$ per $i \neq j$.

Come sarà discusso a seguire se $dist\left(X_i, Y_j\right) = \left\|X_i - Y_j\right\|_2^2$, si parlera di *k-mean* clustering. Circa il processo di clustering, questi si compone di cinque fasi, quali

- selezione della soglia di clustering;
- selezione della metodologia di clustering;
- scelta del modello di clustering;
- validazione dei risultati;
- interpretazione dei risultati.

– 5.1 – Tipologie di clustering.

Come intuibile, scelte diverse della funzione $dist\left(X_i, Y_j\right)$, portano ad algoritmi di clustering diversi. In generale, se ne distinguono due tipologie:

1. *clustering partizionale:* si tratta di una tipologia di clustering in cui ciascun cluster è nettamente distinto dagli altri;
2. *clustering gerarchico:* è una tipologia di clustering in cui è possibile che un cluster sia "sotto-cluster" di un'altro.

– 5.2 – Misura della qualità di clustering.

Ora sebbene gli algoritmi di clustering siano utilizzati per il ML non-supervisionato, è necessario stabilire la qualità dell'algoritmo, e per far ciò si definiscono delle misure relativamente ai cluster. Tali misure si dividono in

- *interne:* si tratta di misure dipendenti unicamente dai dati del cluster;
- *esterne:* misure da utilizzare, nel caso siano comunque presenti delle etichette.

Banalmente minore è SSE, migliore è la qualità del clustering.

Tra le pù diffuse ed utilizzate la *sum of square errors (SSE)*, definita come

$$SSE = \sum_{j=1}^{m} \sum_{x \in C_j} dist(x, y_j)$$

– 5.2.1 – Misure interne: coesione e separazione.

Si tratta di due misure essenziali, queste infatti stabiliscono, rispettivamente, quanto elementi di uno stesso cluster siano correlati (*coesione*) e, quanto in media gli elementi di due cluster distinti siano distanti.

– 5.2.2 – Misure interne: WSS.

È una misura di tipo strutturale; quel che fa è misurare la "bonta" di ogni cluster, in maniera indipendente dalle altre. In genere può anche essere utilizzata per stimare il numero di cluster necessario.

– 5.2.3 – Stima del numero di cluster.

Per stabilire il numero di cluster si può operare in diversi modi, uno di questi è fornito dalle misure di validazione. Considerando il k-means, il valore ideale di k può essere ottenuto come segue:

- si esegue l'algoritmo per valori di k diversi;
- per ognuno dei k considerati, si calcola WSS;
- si considera il grafico dato dai valori di WSS.

– 5.3 – K-mean clustering.

Come anticipato se $dist\left(X_i, Y_j\right) = \left\|X_i - Y_j\right\|_2^2$, si ha a che fare con il *k-mean* clustering. Con tale algoritmo, l'utente deve solamente fornire i rappresentanti dei cluster, e da esse l'algoritmo procedera a comporli. Nello specifico l'algoritmo procede come segue.

1. per ogni elemento x, si calcola la distanza con ogni rappresentate μ_i definito dall'utente, e si assegna x al cluster il cui rappresentante ha la distanza minore da x;
2. per ogni cluster così formato si calcola il centroide (la media);
3. si ripete il processo finché non vi è convergenza[2].

Circa la complessità dell'algoritmo, questa è $\mathcal{O}(nkdi)$, ove n è il numero di elementi del dataset, k il numero di cluster desiderati, i il numero di iterazioni e d il numero di attributi delle osservazioni.

[1] Si tratta di osservazioni del dataset, che hanno poca correlazione con tutti le altre.

[2] Si raggiunge la convergenza se vi è una ripetizione di medie, o se si è definito un qualche criterio di arresto.

– 6 – Classificatori.

L'idea alla base della classificazione è molto semplice; supposto S un insieme di dati etichettati, che si ricordano essere dati che definiscono la propria classe di appartenenza, si costruisce un modello di ML tale che, fornendo allo stesso dati non ancora visionati, questo sia in grado di predirne la classe di appartenenza. Sebbene esistano vari algoritmi di classificazione, di interesse risulta essere *support vector machine*, descritto nel seguito.

– 6.1 – SVM: support vector machine.

Considerando la sua versione più elementare (il caso lineare) SVM costruisce un iper-piano della forma

$$\mathbf{w}\mathbf{x} + b = 0$$

con $\mathbf{w} \in \mathbb{R}^n$ e b costante, parametrizzanti l'iper-piano, tale da separare le osservazioni.
Osservazione. L'iper-piano, generalmente, non è unico.

Poiché generalmente l'iper-piano non è unico, come si stabilisce quello da utilizzare? Di norma si tende ad utilizzare quello che massimizza il cosiddetto *margine*: si tratta della distanza tra l'osservazione più vicina all'iper-piano e il limite decisionale dello stesso.

– 6.1.1 – Hard margin SVM.

Da quanto precedentemente detto, posto $\mathbf{x}$ il vettore associato a un punto p, $\mathbf{w}$ il vettore ortogonale all'iper-piano e c la distanza tra $\mathbf{w}$ e l'iper-piano, segue che

- se $\mathbf{w}\mathbf{x} = c$: il punto p sara sovrapposto al limite decisionale, e dunque non sarà classificabile;
- se $\mathbf{w}\mathbf{x} < c$: il punto sara classificato negativamente;
- in ultima istanza, p è classificato positivamente.

Da ciò, risulta fondamentale massimizzare c, e si dimostra che ciò si verifica massimizzando

$$\arg\min_{\mathbf{w},\mathbf{x}} \left\{ \frac{2}{\|\mathbf{w}\|} \right\} \tag{1}$$

– 6.1.2 – Soft margin SVM.

Si osserva però che l'hard margin SVM è applicabile unicamente al caso lineare. Poiché nella realtà tale situazione è estremamente rara, è necessario poter estendere SVM al caso non lineare. Tale estensione è data proprio dal *soft margin*.

Considerando la (1), e ricordando che massimizzare un funzione equivale a minimizzare l'inverso della stessa, si ha

$$\arg\min_{\mathbf{w},\mathbf{x}} \left\{ \frac{\|\mathbf{w}\|}{2} \right\}.$$

Si deve però tenere traccia dell'eventuale errore di classificazione, errore rappresentato dalla funzione ζ. Da cui in conclusione, si deve minimizzare

$$\arg\min_{\mathbf{w},\mathbf{x}} \left\{ \frac{\|\mathbf{w}\|}{2} \right\} + c \sum_{i=1}^{n} \zeta_i$$