

# Statistical methods for machine learning

Mauro Tellaroli

## Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Definizioni fondamentali . . . . .	2
1.1.1	<i>Label set</i> $\mathcal{Y}$ . . . . .	2
1.1.2	<i>Loss function</i> $\ell$ . . . . .	2
1.1.3	<i>Data domain</i> $\mathcal{X}$ . . . . .	3
1.1.4	Predittori $f$ . . . . .	4
1.1.5	Esempi . . . . .	4
1.1.6	<i>Test set</i> e <i>test error</i> . . . . .	4
1.1.7	<i>Learning algorithm</i> $A$ . . . . .	4
1.1.8	<i>Training error</i> $\ell_S$ . . . . .	4
1.2	Empirical Risk Minimization (ERM) . . . . .	5
1.2.1	Definizione . . . . .	5
1.2.2	Predittori con <i>test error</i> elevato . . . . .	5
1.2.3	<i>Overfitting</i> e <i>underfitting</i> . . . . .	6
1.2.4	Etichette rumorose . . . . .	6
<b>2</b>	<b>L'algoritmo <i>Nearest Neighbor</i> (NN)</b>	<b>7</b>

# 1 Introduzione

## 1.1 Definizioni fondamentali

La *data inference* è lo studio dei metodi che utilizzano i dati per predire il futuro. Il *Machine Learning* è uno strumento potente che può essere usato per risolvere una grossa parte dei problemi di *data inference*, inclusi i seguenti:

- **Clustering**: raggruppare i *data points* in base alle loro similarità;
- **Prediction**: assegnare delle etichette (*label*) ai *data points*;
- **Generation**: generare nuovi *data points*;
- **Control**: eseguire una sequenza di azioni in un ambiente con l'obiettivo di massimizzare una nozione di utilità.

Con *data point* si intende una serie di informazioni legate ad un unico elemento; un'analogia può essere un *record* in un database.

Gli algoritmi che risolvono una *learning task* in base a dei dati già semanticamente etichettati lavorano in modalità ***supervised learning***. A etichettare i dati saranno delle persone o la natura. Un esempio dell'ultimo caso sono le previsioni del meteo. D'altra parte, gli algoritmi che utilizzano i dati senza la presenza di etichette lavorano in modalità ***unsupervised learning***.

In questo corso ci si focalizzerà sul *supervised learning* e la progettazione di sistemi di *machine learning* il cui obiettivo è apprendere dei **predittori**, ovvero funzioni che mappano i *data points* alla loro etichetta.

### 1.1.1 Label set $\mathcal{Y}$

Verrà usata  $\mathcal{Y}$  per indicare il *label set*, ovvero l'insieme di tutte le possibili etichette di un *data point*. Le etichette potranno essere di due tipi differenti:

1. **Categoriche** ( $\mathcal{Y} = \{\text{sport, politica, economia}\}$ ): si parlerà di problemi di **classificazione**;
2. **Numeriche** ( $\mathcal{Y} \subseteq \mathbb{R}$ ): si parlerà di problemi di **regressione**.

È importante sottolineare come la reale differenza tra le due tipologie di etichetta sia il significato e non la sua rappresentazione in quanto, si potrà sempre codificare un'etichetta categorica in un numero.

A sottolineare ciò è il fatto che nella regressione l'errore è tipicamente una funzione della differenza  $|y - \hat{y}|$ , dove  $\hat{y}$  è la predizione di  $y$ . Nella classificazione, invece, l'errore è tipicamente binario: predizione corretta ( $\hat{y} = y$ ) o errata ( $\hat{y} \neq y$ ).

Quando ci sono solo due possibili etichette ( $|\mathcal{Y}| = 2$ ), si ha un **problema di classificazione binario** e, convenzionalmente, verrà usata una codifica numerica  $\mathcal{Y} = \{-1, 1\}$ .

### 1.1.2 Loss function $\ell$

Come già visto precedentemente, si vuole misurare l'errore che un predittore commette su una determinata predizione. Per farlo si userà una **funzione di loss**  $\ell$  non negativa che misurerà la discrepanza  $\ell(y, \hat{y})$  tra l'etichetta predetta  $\hat{y}$  e quella corretta  $y$ . Si assumerà sempre  $\ell(y, \hat{y}) = 0$  quando  $\hat{y} = y$ .

La funzione di loss più semplice per la classificazione è la **zero-one loss**:

$$\ell(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & \text{altrimenti} \end{cases}$$

Nella regressione, le tipiche funzioni di loss sono:

- la **absolute loss**:  $\ell(y, \hat{y}) = |y - \hat{y}|$

- la **quadratic loss**:  $\ell(y, \hat{y}) = (y - \hat{y})^2$

In alcuni casi può essere conveniente scegliere l'etichetta predetta da un insieme  $\mathcal{Z}$  diverso da  $\mathcal{Y}$ . Per esempio, si consideri il problema di assegnare una probabilità  $\hat{y} \in (0, 1)$  all'evento  $y = \text{"pioverà domani"}$ . In questo caso,  $\mathcal{Y} = \{\text{"piove", "non piove"}\}$  e  $\mathcal{Z} = (0, 1)$ . Indicando questi due eventi con 1 (piove) e 0 (non piove), si può usare una funzione di loss per la regressione, come la *absolute loss*:

$$\ell(y, \hat{y}) = |y - \hat{y}| = \begin{cases} 1 - \hat{y} & y = 1 \quad (\text{piove}) \\ \hat{y} & y = 0 \quad (\text{non piove}) \end{cases}$$

Per penalizzare maggiormente le predizioni che distano troppo dalla realtà, si può usare una **logarithmic loss**:

$$\ell(y, \hat{y}) = \begin{cases} \ln \frac{1}{\hat{y}} & y = 1 \quad (\text{piove}) \\ \ln \frac{1}{1-\hat{y}} & y = 0 \quad (\text{non piove}) \end{cases}$$

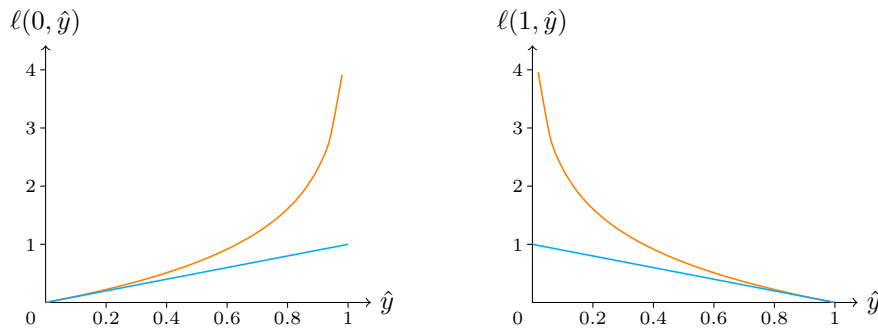


Figura 1: Confronto tra *absolute loss* e *logarithmic loss*; a sinistra il caso  $y = 0$ , a destra  $y = 1$ .

Si noti in figura 1 come la *logarithmic loss* tenda ad infinito quando la predizione è opposta all'etichetta reale:

$$\lim_{\hat{y} \rightarrow 1^-} \ell(0, \hat{y}) = \lim_{\hat{y} \rightarrow 0^+} \ell(1, \hat{y}) = +\infty$$

In pratica questo previene l'utilizzo di predizioni  $\hat{y}$  troppo sicure, quindi troppo vicine a zero o uno.

### 1.1.3 Data domain $\mathcal{X}$

Verrà usata  $\mathcal{X}$  per indicare l'insieme dei *data points*; ogni suo punto  $x \in \mathcal{X}$  è tipicamente un record di un database formato da *feature*:

$$x = (x_1, \dots, x_d)$$

Spesso un *data point* può essere codificato come un vettore i cui elementi sono le sue *feature*. Questa codifica risulta naturale in presenza di quantità omogenee, come i pixel di un'immagine o una lista di occorrenze di parole in un testo. Quando invece i dati presenti utilizzano unità di misura differenti, come "età" e "altezza", la codifica non risulta più immediata. Ci sarà bisogno di una procedura che codifichi i dati in modo da ottenere uno spazio vettoriale omogeneo e coerente con i dati iniziali.

In questo corso si assumerà che i dati possano essere rappresentati da vettori di numeri:

$$\mathcal{X} \equiv \mathbb{R}^d$$

### 1.1.4 Predittori $f$

Un **predittore** è una funzione  $f : \mathcal{X} \rightarrow \mathcal{Y}$  che mappa i *data points* alle etichette (o  $f : \mathcal{X} \rightarrow \mathcal{Z}$ ). Si può quindi dire che in un problema di predizione l'obiettivo è ottenere una funzione  $f$  che genera delle predizioni  $\hat{y} = f(x)$  tali che  $\ell(y, \hat{y})$  sia basso per il maggior numero di punti  $x \in \mathcal{X}$  osservati. In pratica, **la funzione  $f$  è definita da un certo numero di parametri in un dato modello**. Un esempio sono i parametri di una rete neurale.

### 1.1.5 Esempi

Nel *supervised learning* un **esempio** è una coppia  $(x, y)$  dove  $x$  è un *data point* e  $y$  la sua reale etichetta.

In alcuni casi  $x$  ha un'unica  $y$ , come nel caso in cui  $y$  rappresenta una proprietà oggettiva di  $x$ ; in altri casi, invece,  $x$  può avere diverse  $y$  associate, come quando le  $y$  sono soggettivamente assegnate da persone.

### 1.1.6 Test set e test error

Per poter stimare la qualità di un predittore si usa un insieme di esempi detto **test set**:

$$\{(x'_1, y'_1), \dots, (x'_n, y'_n)\}$$

Data una *loss function*  $\ell$ , il *test set* viene usato per calcolare il **test error** di un predittore  $f$ :

$$\frac{1}{n} \sum_{t=1}^n \ell(\underbrace{y'_t}_{\text{reale}}, \underbrace{f(x'_t)}_{\text{predetta}})$$

Il *test error* ha quindi lo scopo di calcolare la prestazione media del predittore su dei dati reali.

### 1.1.7 Learning algorithm $A$

Si definisce *training set*  $S$  un insieme di esempi:

$$S = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

che viene usato dal **learning algorithm**  $A$  per produrre un predittore  $A(S)$ . Informalmente, il *learning algorithm* “impara” dal *training set*.

$$\underbrace{\{(x_1, y_1), \dots, (x_m, y_m)\}}_S \longrightarrow \boxed{A \overset{\ell}{\longrightarrow}} \longrightarrow A(S) = f : \mathcal{X} \rightarrow \mathcal{Y}$$

Il *test set* e il *training set* vengono solitamente prodotti assieme attraverso un processo di collezione dati e etichettamento. Dato l'insieme di esempi preparati, questo verrà partizionato in *test set* e *training set*, tipicamente tramite una divisione casuale. **Obiettivo del corso è lo sviluppo di una teoria che ci guidi nella progettazione di *learning algorithm* che generano predittori con un basso *test error*.**

### 1.1.8 Training error $\ell_S$

Sia  $S = \{(x_1, y_1), \dots, (x_m, y_m)\}$  il *training set*; viene definito, equivalentemente al *test error*, il **training error**:

$$\ell_S(f) = \frac{1}{m} \sum_{t=1}^m \ell(y_t, f(x_t))$$

Un approccio intuitivo alla progettazione di *learning algorithm* è quello di assumere che il *training error*  $\ell_S(f)$  del predittore  $f$  sia correlato con il suo *test error*.

## 1.2 Empirical Risk Minimization (ERM)

### 1.2.1 Definizione

Sia  $\mathcal{F}$  un insieme di predittori e  $\ell$  una *loss function*. L'*empirical risk minimizer* (ERM) è il *learning algorithm*  $A$  che restituisce un predittore in  $\mathcal{F}$  che **minimizza il *training error***:

$$A(S) \in \operatorname{argmin}_{f \in \mathcal{F}} \ell_S(f)$$

Si noti come  $A(S)$  appartenga e non uguagli il minimo; questo perchè ci potrebbero essere più  $f \in \mathcal{F}$  che minimizzano  $\ell_S(f)$ .

### 1.2.2 Predittori con *test error* elevato

Quando in  $\mathcal{F}$  tutti i predittori hanno un *test error* alto, ERM produrrà un pessimo predittore. **Per trovare un buon predittore, ovvero un predittore con un *test error* basso, ci sarà quindi bisogno che  $\mathcal{F}$  sia sufficientemente grande.**

**Tuttavia, se  $\mathcal{F}$  è troppo grande, anche in questo caso verrà prodotto un pessimo predittore.** Un esempio è il seguente.

Si consideri il seguente problema “giocattolo”:

$$\mathcal{Y} = \{-1, 1\} \quad \mathcal{X} = \{x_1, x_2, x_3, x_4, x_5\}$$

Si prenda l'insieme  $\mathcal{F}$  contenente un classificatore  $f : \mathcal{X} \rightarrow \mathcal{Y}$  per ognuna delle possibili combinazioni di etichettamento dei cinque *data points*.  $\mathcal{F}$  sarà quindi formata da  $2^5 = 32$  classificatori:

$$\mathcal{F} = \{f_1, \dots, f_{32}\}$$

$\mathcal{F}$	$f(x_1)$	$f(x_2)$	$f(x_3)$	$f(x_4)$	$f(x_5)$
$f_1$	1	1	1	1	1
$f_2$	1	1	1	1	-1
$f_3$	1	1	1	-1	1
$f_4$	1	1	1	-1	-1
$f_5$	1	1	-1	1	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f_{31}$	-1	-1	-1	-1	1
$f_{32}$	-1	-1	-1	-1	-1

Si supponga che il *training set*  $S$  contenga solo tre *data points* qualsiasi e il *test set* contenga gli altri due. Sia  $f^*$  il predittore usato per etichettare i dati che quindi avrà zero *test e training error*; ogni etichetta  $y_t$  sarà quindi ottenuta da  $f^*$ :

$$y_t = f^*(x_t) \quad \forall t = 1, \dots, 5$$

Per rendere l'idea, si prenda come esempio:

$$f^* = f_3$$

$$\begin{aligned} S &= \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\} \\ &= \{(x_1, 1), (x_2, 1), (x_3, 1)\} \end{aligned}$$

Nonostante ad avere *test error* nullo sia solo  $f_3$ , ad avere il *training error* nullo sono i quattro classificatori che hanno  $y_1, y_2, y_3 = 1$  ovvero  $f_1, f_2, f_3, f_4$ . Questo perchè il *training set*  $S$  contiene solo i primi 3 *data points*.

Siamo quindi nella situazione in cui ERM trova più predittori con  $\ell_S$  minimo e non ha abbastanza informazioni per capire quale di questi sia migliore a livello di *test error*.

Il problema dell'esempio appena visto è che  $\mathcal{F}$  è troppo grande rispetto al *training set*. La domanda che sorge spontanea è quindi: Quanto deve essere grande  $\mathcal{F}$  per poter ottenere un buon predittore tramite ERM?

La teoria dell'informazione ci suggerisce che  $S$  debba avere cardinalità  $\log_2 |\mathcal{F}|$  o, viceversa,  $\mathcal{F}$  debba avere cardinalità  $2^m$ . Quindi, nell'esempio di prima, il *training set* avrebbe dovuto contenere almeno  $\log_2 |\mathcal{F}| = 5$  *data points*.

### 1.2.3 Overfitting e underfitting

I due eventi visti nella sezione precedente, che portano alla generazione di un predittore con *test set* elevato, vengono chiamati:

- **Underfitting:** si verifica quando il *training error* è elevato;
- **Overfitting:** si verifica quando il *training error* è basso ma il *test error* è alto.

Quando  $A$  è ERM e  $S$  ha dimensione fissata  $|S| = m$ :

- Ci si aspetta *overfitting* quando  $\log_2 |\mathcal{F}| \gg m$ ;
- Ci si aspetta *underfitting* quando  $\log_2 |\mathcal{F}| \ll m$ .

### 1.2.4 Etichette rumorose

Il fenomeno dell'*overfitting* spesso accade quando le etichette sono rumorose, ovvero quando le etichette  $y$  non sono deterministicamente associate con i *data points*  $x$ . Questo può accadere per i seguenti motivi (non mutuamente esclusivi tra loro):

1. **Incertezza umana:** se ad etichettare  $S$  sono delle persone, ci sarà dell'incertezza in quanto persone diverse potrebbero avere opinioni diverse;
2. **Incertezza epistemica:** ogni *data point* è rappresentato da un vettore delle *feature* che non contiene abbastanza informazioni per determinare univocamente l'etichetta;
3. **Incertezza aleatoria:** il vettore delle *feature* che rappresenta il *data point* è ottenuto attraverso delle misurazioni rumorose.

Le etichette rumorose portano all'*overfitting* perchè possono ingannare l'algoritmo su quale sia la "vera" etichetta di una certo *data point*.

## 2 L'algoritmo *Nearest Neighbor* (NN)

Verrà introdotto ora l'algoritmo di *Nearest Neighbor* (NN) per la classificazione binaria con *feature* numeriche:

$$\mathcal{X} = \mathbb{R}^d \quad \mathcal{Y} = \{-1, 1\}$$

NN non è un'istanza di ERM in quanto non punta a minimizzare  $\ell_S$ .

L'idea di NN è la seguente:

- Predici ogni punto del *training set* con la propria etichetta;
- Predici gli altri punti con l'etichetta del punto del *training set* che è più vicino al punto interessato.

Più formalmente, dato un *training set*:

$$S = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

l'algoritmo  $A_{NN}$  genera un classificatore  $h_{NN} : \mathbb{R} \rightarrow \{-1, 1\}$  definito come segue:

$$h_{NN}(x) = \text{etichetta } y_t \text{ del punto } x_t \in S \text{ più vicino a } x$$

Se a minimizzare la distanza con  $x$  sono più punti, si predurrà l'etichetta più presente tra i punti vicini. Se non c'è una maggioranza di etichette tra i punti più vicini si predurrà un valore di default  $\in \{-1, 1\}$ .

Presi due punti  $x = (x_1, \dots, x_d)$  e  $x_t = (x_{t,1}, \dots, x_{t,d})$ , la distanza  $\|x - x_t\|$  verrà calcolata tramite la distanza euclidea:

$$\|x - x_t\| = \sqrt{\sum_{i=1}^d (x_i - x_{t,i})^2}$$

Ogni classificatore binario  $f : \mathbb{R}^d \rightarrow \{-1, 1\}$  partiziona  $\mathbb{R}^d$  in due regioni (come mostrato in figura 2):

$$\{x \in \mathbb{R}^d : f(x) = 1\} \quad , \quad \{x \in \mathbb{R}^d : f(x) = -1\}$$

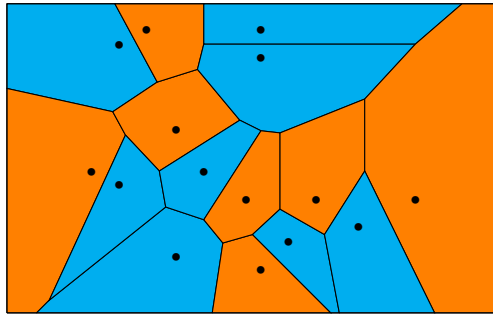


Figura 2: Diagramma di Voronoi in  $\mathbb{R}^2$