Statistical methods for machine learning

Mauro Tellaroli

Indice

1 Introduzione

1.1 Definizioni fondamentali

La *data inference* è lo studio dei metodi che utilizzano i dati per predirre 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** ($\mathcal{Y} = \{\text{sport}, \text{politica}, \text{economia}\}$): 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* ℓ

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** ℓ 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 $y =$ "pioverà domani". In questo caso, $\mathcal{Y} = \{$"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***:

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

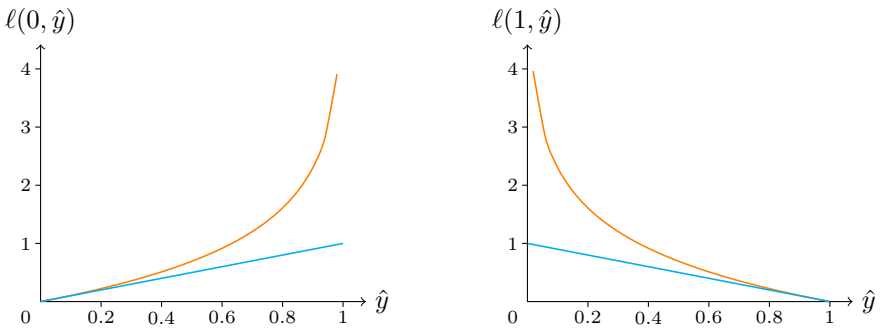

Figura 1: Confronto tra *absolute loss* e *logarithmic loss*; a sinistra il caso $y = 0$, a destra $y = 1$.

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

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

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. Spesso un *data point* può essere codificato come un vettore. 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 $f : \mathcal{X} \to \mathcal{Y}$ che mappa i *data points* alle etichette (o $f : \mathcal{X} \to \mathcal{Z}$). Sì 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***:

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

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

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

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:

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

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

$$\underbrace{\{(x_1, y_1), \ldots, (x_m, y_m)\}}_{S} \longrightarrow \boxed{A \;\; \boxed{\ell}} \longrightarrow A(S) = f : \mathcal{X} \to \mathcal{Y}$$

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* ℓ_S

Sia $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ 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 ℓ una *loss function*. L'*empirical risk minimizer* (ERM) è il *learning algorithm* A che restituisce alcuni predittori in $\mathcal{F}$ che **minimizzano 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 *Overfitting* e *underfitting*

Se in $\mathcal{F}$ tutti i predittori hanno un *test error* alto, ERM produrrà un pessimo predittore. **Per trovare un buon predittore, ci sarà quindi bisogno di un $\mathcal{F}$ 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":

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

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

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

$\mathcal{F}$	$f(x_1)$	$f(x_2)$	$f(x_3)$	$f(x_4)$	$f(x_5)$
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^*:

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

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 ℓ_S minimo e non ha abbastanza informazioni per capire quale di questi ha il *test error* migliore.

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*.