METODI STATISTICI PER L'APPRENDIMENTO

Luca Cappelletti
Prof. Nicolo Cesa Bianchi

6 CFU

2019
Informatica Magistrale
Università degli studi di Milano
Italia
2 febbraio 2020

Indice

1 Introduzione

Definizione 1.1 | Funzione di perdita

Si tratta di una funzione *non negativa* $\ell(y, \hat{y})$ che misura la discrepanza fra un etichetta predetta ed una etichetta vera.

$$\ell(y, \hat{y})$$

Definizione 1.2 | Esempio

Un esempio è una coppia $(\underline{\boldsymbol{x}}, y)$ composta da un dato $\underline{\boldsymbol{x}}$ e dalla sua etichetta y, che è quella che riteniamo corretta per quel dato.

Definizione 1.3 | Training set

Si tratta di un insieme $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m)$ di esempi.

Definizione 1.4 | Test set

Un test set è un insieme $(\underline{\boldsymbol{x}}'_1, y'_1), \ldots, (\underline{\boldsymbol{x}}'_n, y'_n)$ di esempi a cui l'algoritmo di apprendimento non ha accesso.

Viene tipicamente costruito assieme al training set.

Definizione 1.5 | Test error

Dato un classificatore o regressore f, stimiamo la capacità predittiva di f attraverso il **test error**:

$$\hat{\text{er}}(f) = \frac{1}{n} \sum_{t=1}^{n} \ell(y'_t, f(\underline{\boldsymbol{x}}'_t))$$

Osservazione 1.1 | A cosa serve il Test Error?

Il **test error** serve a stimare il comportamento del predittore "sul campo", ovvero su dati non precedentemente osservati. Il nostro scopo é quindi formulare una teoria che ci permetta di sviluppare algoritmi di apprendimento in grado di generare predittori con basso test error.

Definizione 1.6 | Metodo di minimizzazione del rischio empirico (ERM)

Sia $\mathbb{F}$ un insieme dato di classificatori o regressori. Il metodo di **minimizzazione del rischio empirico** (o ERM, da *empirical risk minimization*) indica l'algoritmo di apprendimento che sceglie la funzione in $\mathbb{F}$ che minimizza il training error:

$$\hat{f} = \operatorname*{argmin}_{f \in \mathbb{F}} \hat{\text{er}}(f)$$

Definizione 1.7 | Overfitting

Il fenomeno per il quale un algoritmo di apprendimento tende a generare predittori con basso training error e alto test error prende il nome di **overfitting**. Si verifica soprattutto quando i dati sono affetti da rumore.

Il predittore "impara" molto bene i dati di esempio e riconosce esattamente quelli.

Definizione 1.8 | Rumore

Diremo che un dataset è affetto da **rumore** quando un'istanza $\underline{\boldsymbol{x}}$ può comparire a volte con un'etichetta e a volte con un'altra.

2

Algoritmo Nearest Neighbour

Definizione 2.1 | Algoritmo Nearest Neighbour

Dato un training set S contenente gli esempi $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m)$, l'algoritmo **nearest neighbour** (NN) genera un classificatore $h_{NN} : \mathbb{R}^d \longrightarrow \{-1, +1\}$ definito come:

$$h_{\text{NN}}(\underline{\boldsymbol{x}}) = \begin{array}{l}\text{etichetta } y_t \text{ del punto } \underline{\boldsymbol{x}} \in S \\ \text{più vicino a } \underline{\boldsymbol{x}}.\end{array}$$

Se esiste più di un punto in S a distanza minima da $\underline{\boldsymbol{x}}$, allora prediciamo con la maggioranza delle etichette di questi punti più vicini. Se c'è un uguale numero di punti più vicini ad $\underline{\boldsymbol{x}}$ con etichette positive e negative, prediciamo un valore di default in $\{-1, +1\}$.

In particolare, il classificatore è costruito in modo tale che $h_{\text{NN}}(\underline{\boldsymbol{x}}_t) = y_t$ per ogni esempio $(\underline{\boldsymbol{x}}_t, y_t)$ del training set. La distanza fra $\underline{\boldsymbol{x}}$ e $\underline{\boldsymbol{x}}_t$, denotata con $\|\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}_t\|$, viene misurata con la formula della distanza euclidea fra due vettori.

Definizione 2.2 | Distanza euclidea

Dati due vettori $\underline{\boldsymbol{x}}$ e $\underline{\boldsymbol{x}}_t$, la distanza euclidea tra di essi è definita come:

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

Definizione 2.3 | Celle di Voronoi

Il classificatore generato dall'algoritmo Nearest Neighbour induce una partizione di $X = \mathbb{R}^n$ in **celle di voronoi**, dove ogni istanza $\underline{\boldsymbol{x}}_t$ del training set è il centro di una stella e il confine tra due celle è costituito dai punti $\underline{\boldsymbol{x}}$ equidistanti dai due centri.

Osservazione 2.1 | I limiti del classificatore Nearest Neighbour

L'algoritmo Nearest Neighbour ha diverse controindicazioni, tra cui:

1. Esso deve memorizzare l'intero training set, quindi l'algoritmo non è pratico per dataset di grandi dimensioni.
2. Dato un qualunque punto $\underline{\boldsymbol{x}}$ di test, il calcolo dell'etichetta $h_{\text{NN}}(\underline{\boldsymbol{x}})$ è computazionalmente oneroso in quanto richiede, in generale, il calcolo delle istanze fra $\underline{\boldsymbol{x}}$ e ogni $\underline{\boldsymbol{x}}_t$ del training set.
3. L'algoritmo realizza sempre un classificatore tale che $\widehat{\text{er}}(h_N N) = 0$, cosa non sorprendente dato che esso memorizza tutto il training set.

Definizione 2.4 | Algoritmi k-NN

Dai predittori NN è possibile ottenere la famiglia di algoritmi k-NN: dato un training set $S = (\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m)$, k-NN genera un classificatore $h_{k\text{-NN}}$ tale che $h_{k\text{-NN}}(\underline{\boldsymbol{x}})$ è l'etichetta $y_t \in \{-1, +1\}$ che appare nella maggioranza dei k punti $\underline{\boldsymbol{x}}_t \in S$ più vicini a $\underline{\boldsymbol{x}}$. Per valutare $h_{k\text{-NN}}(\underline{\boldsymbol{x}})$, si compiono le seguenti operazioni:

1. Trovo i k punti $\underline{\boldsymbol{x}}_{t_1}, \ldots, \underline{\boldsymbol{x}}_{t_k}$ del training set più vicini a $\underline{\boldsymbol{x}}$ e siano $y_{t_1}, \ldots, y_{t_k}$ le etichette di questi punti.
2. Se la maggioranza delle etichette $y_{t_1}, \ldots, y_{t_k}$ è pari a $+1$, allora $h_{k\text{-NN}}(\underline{\boldsymbol{x}}) = +1$, altrimenti se è pari a -1 allora $h_{k\text{-NN}}(\underline{\boldsymbol{x}}) = +1$. Nel caso in cui non vi sia una maggioranza, allora predico un valore di default in $\{-1, +1\}$.

Osservazione 2.2 | Applicazioni di algoritmi k-NN

Gli algoritmi k-NN si prestano bene a risolvere problemi di classificazione multiclasse (dove l'immagine contiene più di due simboli) e di regressione.

Osservazione 2.3 | Test error nell'algoritmo k-NN

Nell'algoritmo k-NN avremo in generale $\widehat{\text{er}}(h_{k\text{-NN}}) > 0$.

3

Algoritmo Nearest Neighbour

Definizione 3.1 | Albero ordinato

Un albero ordinato è un albero dove i figli di ogni nodo sono numerati progressivamente. Se il nodo interno v ha k figli, distingueremo il figlio $1, \dots, k$.

Definizione 3.2 | Struttura di un predittore ad albero

La struttura di un predittore ad albero è quella di un albero ordinato con radice.

Definizione 3.3 | Predittore ad albero

Fissiamo $\mathbb{X} = \mathbb{X}_1 \times \dots \times \mathbb{X}_d$, dove $\mathbb{X}_i$ è l'insieme di valori che può assumere l'i-esimo attributo x_i. Un **predittore ad albero** $h_T : \mathbb{X} \longrightarrow \mathbb{Y}$ è un predittore associato ad un albero ordinato T, i cui nodi interni sono etichettati da *test* e le cui foglie sono etichettate da elementi di $\mathbb{Y}$.

Un test per un nodo interno con k figli è una funzione $f : \mathbb{X}_i \longrightarrow \{1, \dots, k\}$ dove i è l'indice di un attributo. Quindi f mappa ciascun valore dell'i-esimo attributo in un figlio del nodo.

Il valore $h_T(\underline{\boldsymbol{x}})$ viene calcolato iniziando assegnando $v \leftarrow r$, dove r è la radice di T:

1. Se v è una foglia ℓ, allora mi fermo assegnando a $h_T(\underline{\boldsymbol{x}})$ l'etichetta $y \in \mathbb{Y}$ associata a ℓ.
2. Altrimenti, se $f : \mathbb{X}_i \longrightarrow \{1, \dots, k\}$ è il test associato a v eseguo l'assegnamento $v \leftarrow v_j$ dove $j = f(x_i)$ e v_j indica il j-esimo figlio di v.
3. Ritorno al passo 1

Definizione 3.4 | Foglia finale

Diciamo che ℓ è la foglia finale di $\underline{\boldsymbol{x}}$ se terminiamo il calcolo di $h_T(\underline{\boldsymbol{x}})$ nella foglia ℓ.

Definizione 3.5 | Costruzione di albero generale

Descriviamo ora un metodo generico di costruzione dell'albero a partire dal training set S.

1. **Inizializzazione:** creo T con la sola radice ℓ. Associo alla radice l'insieme $S_\ell = S$. Come etichetta della radice scelgo la maggioranza delle etichette degli esempi in S_ℓ.
2. **Loop principale:** scelgo una foglia ℓ e la trasformo in nodo intorno creando due figli ℓ' (primo figlio) e ℓ'' (secondo figlio). Scelgo un attributo i e un test $f : \mathbb{X}_i \longrightarrow \{1, 2\}$. Associo il test f alla ex-foglia ℓ e partiziono l'insieme S_ℓ nei due sottoinsiemi:

$$S_{\ell'} = \{(\underline{\boldsymbol{x}}_t, y_t) \in S_\ell : f(x_{t,i}) = 1\}$$
$$S_{\ell''} = \{(\underline{\boldsymbol{x}}_t, y_t) \in S_\ell : f(x_{t,i}) = 2\}$$

   Come etichetta di ℓ' scelgo la maggioranza delle etichette degli esempi in $S_{\ell'}$ e come etichetta di ℓ'' scelgo la maggioranza delle etichette degli esempi in $S_{\ell''}$.

Definizione 3.6 | Funzione di Gini

La funzione di Gini viene definita come:

$$\psi_2(p) = 2p(1-p)$$

Definizione 3.7 | Funzione di entropia scalata

La funzione di entropia scalata viene definita come:

$$\psi_3(p) = -\frac{p}{2}\ln p - \frac{1-p}{2}\ln 1 - p$$

Osservazione 3.1 | Overfitting negli alberi di decisione

Gli alberi di decisione possono subire il fenomeno dell'overfitting: il parametro rilevante risulta essere il numero di nodi nell'albero. Consideriamo un training set affetto da rumore, se l'albero viene fatto crescere fino ad azzerare il training error, il classificatore tenderà a mostrare un test error più elevato di un altro albero la cui crescita venga arrestata prima.

Osservazione 3.2 | Alberi di decisione e forma normale disgiuntiva

Una caratteristica interessante dei classificatori basati su alberi di decisione è che possiamo rappresentarli come una formula di logica proposizionale in forma disgiuntiva: questa si ottiene facendo la disgiunzione delle clausole ottenute dai cammini che dalla radice conducono a tutte le foglie etichettate con $+1$.

4

Rischio statistico e sua analisi

Osservazione 4.1 | Esempio come estrazione indipendente

Nel modello statistico di apprendimento assumiamo che ogni esempio $(\underline{\boldsymbol{x}}, y)$ sia ottenuto tramite un'estrazione indipendente da una distribuzione di probabilità su $\mathbb{X} \times \mathbb{Y}$ fissata ma ignota. Per evidenziare che $\underline{\boldsymbol{x}}$ e y sono variabili casuali spesso scriveremo $(\underline{\mathbb{X}})$.

Osservazione 4.2 | Esempio come campione casuale

Dato che nel modello statistico ogni esempio $(\underline{\mathbb{X}}, Y)$ è ottenuto tramite un'estrazione indipendente dalla stessa distribuzione di probabilità congiunta, ogni dataset sarà un **campione casuale** nel senso statistico del termine.

Osservazione 4.3 | Assunzione di indipendenza

In generale, l'assunzione di indipendenza è comoda dal punto di vista della trattabilità analitica del problema ma risulta poco plausibile in realtà.

Definizione 4.1 | Funzione indicatrice

La **funzione indicatrice** $\mathbb{I}\{A\} \in \{0,1\}$ di un evento A è definita pari a 1 se A è vero, 0 altrimenti.

Definizione 4.2 | Rischio statistico

Le prestazioni di un predittore $h : \mathbb{X} \longrightarrow \mathbb{Y}$ rispetto ad un modello statistico dato ed a una funzione di perdita $\ell : \mathbb{Y} \times \mathbb{Y} \longrightarrow \mathbb{R}$ dato vengono valutate con il **rischio statistico**, definito da:

$$\operatorname{er}(h) = \mathbb{E}\left[\ell\left(Y, h\left(\underline{\boldsymbol{X}}\right)\right)\right]$$

ovvero, il valore atteso della funzione di perdita rispetto ad un esempio $(\underline{\boldsymbol{X}}, Y)$ generato dal modello statistico.

Definizione 4.3 | Bayes Error

Il rischio $\operatorname{er}(f^*)$ del predittore Bayesiano ottimo è detto **Bayes error**. In generale il **Bayes error** è maggiore di zero in quanto i predittori sono deterministici mentre le etichette sono probabilistiche.

Definizione 4.4 | Predittore Bayesiano ottimo

Ipotizzando di conoscere il modello statistico, possiamo costruire il **predittore Bayesiano ottimo** $f^*\left(\underline{\boldsymbol{x}}\right) : \mathbb{X} \longrightarrow \mathbb{Y}$. Esso è definito come:

$$f^*\left(\underline{\boldsymbol{x}}\right) = \operatorname*{argmin}_{\widehat{y} \in \mathbb{Y}} \mathbb{E}\left[\ell\left(Y, \widehat{y}\right) \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right]$$

ovvero la predizione $\widehat{y}$ che minimizza il rischio condizionato, cioè la perdita attesa rispetto alla distribuzione di Y condizionata su $\underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}$. Si noti che, per definizione di f^*, vale:

$$\mathbb{E}\left[\ell\left(Y, f^*(x)\right) \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] \leqslant \mathbb{E}\left[\ell\left(Y, h(x)\right) \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right]$$

per ogni classificatore $h : \mathbb{X} \longrightarrow \mathbb{Y}$.

Dato che la disuguaglianza sopra vale per ogni $\underline{\boldsymbol{x}} \in \mathbb{X}$, vale anche in media rispetto all'estrazione di $\underline{\boldsymbol{X}}$. Ma siccome la media del rischio condizionato coincide col rischio, abbiamo che $\operatorname{er}\left(f^*\right) \leqslant \operatorname{er}(h)$.

Esempio 4.1 | Bayesiano ottimo con loss quadratica

Iniziamo ora a calcolare il predittore Bayesiano ottimo per la funzione di perdita quadratica $\ell\left(y, \widehat{y}\right) = \left(y - \widehat{y}\right)^2$ nel caso $\mathbb{Y} \equiv \mathbb{R}$:

$$\begin{aligned} f^*\left(\underline{\boldsymbol{x}}\right) &= \operatorname*{argmin}_{\widehat{y} \in \mathbb{R}} \mathbb{E}\left[\left(Y - \widehat{y}\right)^2 \,\middle|\, \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] \\ &= \operatorname*{argmin}_{\widehat{y} \in \mathbb{R}} \left(\mathbb{E}\left[Y^2 \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] + \widehat{y^2} - 2\widehat{y}\,\mathbb{E}\left[Y \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right]\right) \\ &= \operatorname*{argmin}_{\widehat{y} \in \mathbb{R}} \left(\widehat{y^2} - 2\widehat{y}\,\mathbb{E}\left[Y \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right]\right) \\ &= \mathbb{E}\left[Y \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] \end{aligned}$$

Ovvero, il predittore Bayesiano ottimo per la funzione di perdita quadratica è il valore atteso dell'etichetta condizionato sull'istanza.

Esempio 4.2 | Rischio con loss quadratica

Sostituendo nella formula del rischio condizionato $\mathbb{E}\left[\left(Y - f^*\left(\underline{\boldsymbol{X}}\right)\right)^2 \,\middle|\, \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right]$ il predittore ottimo $f^*\left(\underline{\boldsymbol{x}}\right) = \mathbb{E}\left[Y \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right]$ otteniamo:

$$\begin{aligned} \mathbb{E}\left[\left(Y - f^*\left(\underline{\boldsymbol{X}}\right)\right)^2 \,\middle|\, \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] &= \mathbb{E}\left[\left(Y - \mathbb{E}\left[Y \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right]\right)^2 \,\middle|\, \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] \\ &= \operatorname{Var}\left(Y \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right) \end{aligned}$$

Ovvero il rischio condizionato del predittore Bayesiano ottimo per la perdita quadrata è la varianza dell'etichetta condizionata sull'istanza.

Definizione 4.5 | Distribuzione congiunta

In probabilità, date due variabili aleatorie X e Y, definite sullo stesso spazio di probabilità, si definisce la loro **distribuzione congiunta** come la distribuzione di probabilità associata al vettore (X, Y). Nel caso di due sole variabili, si parla di distribuzione **bivariata**, mentre nel caso di più variabili si parla di distribuzione **multivariata**.

$$F(x_1, \dots, x_m) = \mathbb{P}(X_1 \leqslant x_1, \dots, X_m \leqslant x_m)$$

Definizione 4.6 | Distribuzione marginale

La **distribuzione marginale** di un sottoinsieme di una collezione di variabili casuali è la distribuzione di probabilità delle variabili contenute nel sottoinsieme. Il termine **variabile marginale** è usato per riferirsi a quelle variabili nel sottoinsieme delle variabili che vengono trattenute ovvero utilizzate.

Questo termine, marginale, è attribuito ai valori ottenuti ad esempio sommando in una tabella di valori lungo le righe oppure lungo le colonne, trascrivendo il risultato appunto a margine rispettivamente della riga o colonna sommata.

Osservazione 4.4 | Distribuzione congiunta degli esempi nel caso di classificazione binaria

Nel caso di classificazione binaria, con $\mathbb{Y} = \{-1, +1\}$, è conveniente specificare una distribuzione congiunta sugli esempi $(\underline{\boldsymbol{X}}, Y)$ con la coppia (D, μ), dove D è la **distribuzione marginale** su $\mathbb{X}$ e μ rappresenta la distribuzione su $\{-1, +1\}$ condizionata su $\mathbb{X}$.

Dato che μ è una distribuzione concentrata su due valori, possiamo rappresentarla con la funzione $\mu : \mathbb{X} \longrightarrow [0, 1]$ dove $\mu(\underline{\boldsymbol{x}}) = \mathbb{P}(Y = +1 \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}})$ è la probabilità che $\underline{\boldsymbol{x}}$ assuma l'etichetta $+1$.

Osservazione 4.5 | Esempi come estrazioni indipendenti nella marginale

Per interpretare il modello statistico, possiamo pensare che ogni esempio $(\underline{\boldsymbol{x}}, y)$ disponibile sia ottenuto attraverso un'estrazione indipendente di $\underline{\boldsymbol{x}} \in \mathbb{X}$ secondo la distribuzione D seguito dall'attribuzione dell'etichetta $y \in \{-1, +1\}$ secondo la distribuzione $\{1 - q(\underline{\boldsymbol{x}}), q(\underline{\boldsymbol{x}})\}$.

Esempio 4.3 | Rischio statistico con loss zero-uno

Il rischio statistico rispetto alla funzione di perdita zero-uno $\ell(y, \hat{y}) = \mathbb{I}\{\hat{y} \neq y\}$ risulta quindi essere:

$$\begin{aligned} \mathrm{er}(h) &= \mathbb{E}\left[\ell\left(Y, h(\underline{\boldsymbol{X}})\right)\right] \\ &= \mathbb{E}\left[\mathbb{I}\{h(\underline{\boldsymbol{X}}) \neq Y\}\right] \\ &= \mathbb{P}\left(h(\underline{\boldsymbol{X}}) \neq Y\right) \end{aligned}$$

In altre parole, il rischio di h è la probabilità che h sbagli a classificare un $\underline{\boldsymbol{X}}$ estratto dalla distribuzione D su $\mathbb{X}$ e avente etichetta Y estratta a caso dalla distribuzione $\{1 - \mu(\underline{\boldsymbol{X}}), \mu(\underline{\boldsymbol{X}})\}$ su $\{-1, +1\}$.

Osservazione 4.6 | È possibile calcolare direttamente il rischio rispetto a un modello statistico?

Non possiamo calcolare direttamente il rischio $\mathrm{er}(h)$ di un qualunque classificatore h rispetto ad un modello statistico (D, μ): per calcolarlo infatti è richiesto di conoscere esattamente (D, μ). Ma se conoscessimo μ potremmo direttamente trovare il classificatore Bayesiano ottimo per il problema.

Esempio 4.4 | Rischio per la classificazione binaria

Per stimare il rischio di un dato classificatore h si usa un **test set**, ovvero un insieme $(\underline{\boldsymbol{x}}'_1, y'_1), \dots, (\underline{\boldsymbol{x}}'_n, y'_n)$ di dati etichettati: il rischio viene approssimato usando il **test error**, ovvero la frazione degli esempi del test set classificati scorrettamente da h:

$$\mathrm{er}_{D,\mu}(h) \approx \widetilde{\mathrm{er}}(h) = \frac{1}{n} \sum_{t=1}^{n} \ell\left(y'_t, h(\underline{\boldsymbol{x}}'_t)\right)$$

Sotto l'ipotesi che il test sia stato generato mediante estrazioni indipendenti dal modello statistico (D, μ), il test error altro non è che la media campionaria del rischio in quanto, per ogni $t = 1, \dots, n$ abbiamo che $(\underline{\boldsymbol{X}}'_t, Y'_t)$ è un'estrazione indipendente da (D, μ). Quindi:

$$\mathbb{E}\left[\ell\left(Y'_t, h(\underline{\boldsymbol{X}}'_t)\right)\right] = \mathbb{P}\left(h(\underline{\boldsymbol{X}}'_t) \neq Y'_t\right) = \mathrm{er}(h)$$

Esempio 4.5 | Classificatore Bayesiano ottimo per classificazione binaria

Ipotizzando di conoscere il modello statistico (D, μ), possiamo costruire il classificatore Bayesiano ottimo $f^* : \mathbb{X} \longrightarrow \{-1, +1\}$. Allora:

$$\begin{aligned} f^*(\underline{\boldsymbol{x}}) &= \underset{\hat{y}\in\{-1,+1\}}{\operatorname{argmin}} \mathbb{E}\left[\ell(Y,\hat{y}) \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] \\ &= \underset{\hat{y}\in\{-1,+1\}}{\operatorname{argmin}} \mathbb{E}\left[\mathbb{I}\{Y=+1\}\mathbb{I}\{\hat{y}=-1\} + \mathbb{I}\{Y=-1\}\mathbb{I}\{\hat{y}=+1\} \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right] \\ &= \underset{\hat{y}\in\{-1,+1\}}{\operatorname{argmin}} \left(\mathbb{P}\left(Y=+1 \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right)\mathbb{I}\{\hat{y}=-1\} + \mathbb{P}\left(Y=-1 \mid \underline{\boldsymbol{X}} = \underline{\boldsymbol{x}}\right)\mathbb{I}\{\hat{y}=+1\}\right) \\ &= \underset{\hat{y}\in\{-1,+1\}}{\operatorname{argmin}} \left(\mu(\underline{\boldsymbol{x}})\mathbb{I}\{\hat{y}=-1\} + (1-\mu(\underline{\boldsymbol{x}}))\mathbb{I}\{\hat{y}=+1\}\right) \\ &= \begin{cases} -1 & \text{se } \mu(\underline{\boldsymbol{x}}) < \frac{1}{2} \\ +1 & \text{se } \mu(\underline{\boldsymbol{x}}) \geqslant \frac{1}{2} \end{cases} \end{aligned}$$

Quindi il classificatore Bayesiano ottimo predice l'etichetta che massimizza la probabilità condizionata sull'istanza. Non è difficile verificare che il **Bayes error** è pari a:

$$\operatorname{er}(f^*) = \mathbb{E}\left[\min\left(\mu(\underline{\boldsymbol{X}}), 1-\mu(\underline{\boldsymbol{X}})\right)\right]$$

Lemma 4.1 | Lemma di Chernoff-Hoeffding

Siano $Z_1, \ldots, Z_n$ variabili casuali indipendenti, identicamente distribuite con media μ, e tali che $0 \leqslant Z_t \leqslant 1$, per ogni $t = 1, \ldots, n$. Allora, per ogni $\varepsilon > 0$ fissato, vale che:

$$\mathbb{P}\left(\frac{1}{n}\sum_{t=1}^{n} Z_t > \mu + \varepsilon\right) \leqslant e^{-2\varepsilon^2 n} \quad \text{e} \quad \mathbb{P}\left(\frac{1}{n}\sum_{t=1}^{n} Z_t < \mu - \varepsilon\right) \leqslant e^{-2\varepsilon^2 n}$$

Osservazione 4.7 | Precisione della stima del rischio utilizzando il maggiorante di Chernoff-Hoeffding

Ponendo $Z_t = \ell(y_t, h(x_t))$ dove ℓ è la funzione di perdita per la classificazione binaria, possiamo valutare la precisione della nostra stima come:

$$\mathbb{P}\left(|\operatorname{er}(h) - \widetilde{\operatorname{er}}(h)| > \varepsilon\right) \leqslant 2e^{-2\varepsilon^2 n}$$

$$\mathbb{P}\left(\operatorname{er}(h) - \widetilde{\operatorname{er}}(h) > \varepsilon\right) + \mathbb{P}\left(\widetilde{\operatorname{er}}(h) - \operatorname{er}(h) > \varepsilon\right) \leqslant 2e^{-2\varepsilon^2 n}$$

dove la probabilità è calcolata rispetto all'estrazione del test set. Questa disuguaglianza ci dice che la misura (secondo D e μ) dei test set che producono stime $\widetilde{\operatorname{er}}(h)$ che differiscono dal valor medio $\operatorname{er}(h)$ per più di ε decresce rapidamente con l'aumentare di n, cioè del numero di esempi nel test set.

In particolare, ponendo $e^{-2\varepsilon^2 n} = \delta$, per un qualunque $\delta \in (0,1)$ fissato e risolvendo la disequazione per ε otteniamo:

$$\varepsilon = \sqrt{\frac{1}{2n}\ln\left(\frac{2}{\delta}\right)}$$

Sostituendo il valore ottenuto nella formula si ottiene:

$$\mathbb{P}\left(|\operatorname{er}(h) - \widetilde{\operatorname{er}}(h)| > \sqrt{\frac{1}{2n}\ln\left(\frac{2}{\delta}\right)}\right) = \delta$$

$$\mathbb{P}\left(|\operatorname{er}(h) - \widetilde{\operatorname{er}}(h)| \leqslant \sqrt{\frac{1}{2n}\ln\left(\frac{2}{\delta}\right)}\right) = 1 - \delta$$

La disuguaglianza ottenuta ci indica come stimare il rischio di un'ipotesi prodotta da un qualche algoritmo di apprendimento mediante un test set. Viceversa, la stessa disuguaglianza ci mostra come il test set, che è il modo con cui in pratica misuriamo le prestazioni di un classificatore su dati ignoti, sia ben correlato col rischio nel modello di apprendimento statistico.

Osservazione 4.8 | Compare Overfitting nel modello statistico delineato?

Procediamo a studiare un algoritmo di apprendimento nel modello statistico che abbiamo appena delineato e verifichiamo se compare l'overfitting. Come di consueto, assumiamo che l'algoritmo riceva in ingresso un training set $(\underline{\boldsymbol{x}}_1, y_1), \dots, (\underline{\boldsymbol{x}}_m, y_m)$, dove $(\underline{\boldsymbol{x}}_t, y_t) \in \mathbb{X} \times \{-1, +1\}$, e generi un classificatore $h : \mathbb{X} \longrightarrow \{-1, +1\}$ appartenente a un dato spazio di modelli $\mathbb{H}$.

Sotto queste ipotesi la cosa migliore che l'algoritmo può fare è scegliere il miglior classificatore possibile $h \in \mathbb{H}$, ovvero il classificatore h^* tale che:

$$\mathrm{er}(h^*) = \min_{h \in \mathbb{H}} \mathrm{er}(h)$$

Grazie alla legge dei grandi numeri, sappiamo che il training error $\widehat{\mathrm{er}}(h^*)$ è vicino a $\mathrm{er}(h^*)$ con alta probabilità rispetto all'estrazione del training set su cui $\widehat{\mathrm{er}}(h^*)$ viene calcolato. Consideriamo quindi l'algoritmo che sceglie $\widehat{h} \in \mathbb{H}$ in modo da minimizzare il training error, ovvero:

$$\widehat{h} = \operatorname*{argmin}_{h \in \mathbb{H}} \widehat{\mathrm{er}}(h)$$

Purtroppo non possiamo applicare direttamente a $\widehat{h}$ il maggiorante di Chernoff-Hoeffding al fine di dimostrare che $\mathrm{er}(\widehat{h})$ è vicino $\widehat{\mathrm{er}}(\widehat{h})$, per poi concludere che $\mathrm{er}(\widehat{h})$ è vicino a $\mathrm{er}(h^*)$. Il motivo è che $\widehat{h}$ è una funzione del training set e quindi una variabile casuale. Chernoff-Hoeffding dice che $\widehat{\mathrm{er}}(h)$ è vicino a $\mathrm{er}(h)$ per ogni h fissato, mentre $\widehat{h}$ non è fissato, ma dipende dal campione rispetto al quale calcoliamo la probabilità.

Per analizzare il rischio di $\widehat{h}$ procediamo allora come segue:

$$\mathrm{er}(\widehat{h}) = \underbrace{\mathrm{er}(\widehat{h}) - \mathrm{er}(h^*)}_{\text{Errore di varianza}} + \underbrace{\mathrm{er}(h^*) - \mathrm{er}(f^*)}_{\text{Errore di bias}} + \underbrace{\mathrm{er}(f^*)}_{\text{Bayes error}}$$

dove f^* è il classificatore Bayesiano ottimo per il modello statistico (D, μ) soggiacente.

Il Bayes error non è controllabile, dato che dipende unicamente dal modello (D, μ). **L'errore di bias** dipende dal fatto che $\mathbb{H}$ può non contenere il classificatore Bayesiano ottimo. **L'errore di varianza** dipende dal fatto che $\mathrm{er}(\widehat{h})$ è generalmente diverso dall'errore di $\widehat{h}$ sul training set.

Di conseguenza, scegliere $\widehat{h}$ comporta un errore dovuto al fatto che la frazione di esempi classificati scorrettamente nel training set da un classificatore h è soltanto una stima, possibilmente imprecisa, del rischio $\mathrm{er}(h)$.

Procediamo quindi a controllare l'errore di varianza: per ogni training set fissato abbiamo che:

$$\begin{aligned}
\mathrm{er}(\widehat{h}) - \mathrm{er}(h^*) &= \mathrm{er}(\widehat{h}) - \widehat{\mathrm{er}}(\widehat{h}) + \widehat{\mathrm{er}}(\widehat{h}) - \mathrm{er}(h^*) \\
&\leqslant \mathrm{er}(\widehat{h}) - \widehat{\mathrm{er}}(\widehat{h}) + \widehat{\mathrm{er}}(h^*) - \mathrm{er}(h^*) \\
&\leqslant |\mathrm{er}(\widehat{h}) - \widehat{\mathrm{er}}(\widehat{h})| + |\widehat{\mathrm{er}}(h^*) - \mathrm{er}(h^*)| \\
&\leqslant 2 \max_{h \in \mathbb{H}} |\widehat{\mathrm{er}}(h) - \mathrm{er}(h)|
\end{aligned}$$

Dove abbiamo usato l'ipotesi che $\widehat{h}$ minimizza $\widehat{\mathrm{er}}$ in $\mathbb{H}$. Quindi, $\forall \varepsilon > 0$ vale che:

$$\begin{aligned}
\mathrm{er}(\widehat{h}) - \mathrm{er}(h^*) > \varepsilon \quad &\Rightarrow \quad \max_{h \in \mathbb{H}} |\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2} \\
&\Rightarrow \quad \exists h \in \mathbb{H} : |\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}
\end{aligned}$$

Dato che la catena di implicazioni qui sopra vale per qualsiasi realizzazione del training set, possiamo scrivere:

$$\mathbb{P}\left(\mathrm{er}(\widehat{h}) - \mathrm{er}(h^*) > \varepsilon\right) \leqslant \mathbb{P}\left(\exists h \in \mathbb{H} : |\widehat{\mathrm{er}}(h)| - \mathrm{er}(h) > \frac{\varepsilon}{2}\right)$$

Studiamo il caso in cui lo spazio dei modelli contiene un numero finito di classificatori, cioè $|\mathbb{H}| < \infty$. Dato che l'evento: $\exists h \in \mathbb{H} : |\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}$ è l'unione su ogni $h \in \mathbb{H}$ degli eventi, non necessariamente disgiunti $|\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}$ ed usando la regola della somma $\mathbb{P}(A_1 \cup \cdots \cup A_n) \leqslant \sum_{i=1}^{n} \mathbb{P}(A_i)$ che vale per qualsiasi collezione di eventi otteniamo che:

$$\begin{aligned}
\mathbb{P}\left(\exists h \in \mathbb{H} : |\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}\right) &= \mathbb{P}\left(\bigcup_{h \in \mathbb{H}} \left(|\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}\right)\right) \\
&\leqslant \sum_{h \in \mathbb{H}} \mathbb{P}\left(|\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}\right) \\
&\leqslant |\mathbb{H}| \max_{h \in \mathbb{H}} \mathbb{P}\left(|\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}\right) \\
&\leqslant |\mathbb{H}| 2 e^{-m\varepsilon^2/2}
\end{aligned}$$

dove nell'ultimo passaggio abbiamo usato il maggiorante di Chernoff Hoeffding. In conclusione otteniamo che:

$$\begin{aligned}
\mathbb{P}\left(\mathrm{er}(\widehat{h}) - \mathrm{er}(h^*) > \varepsilon\right) &\leqslant \mathbb{P}\left(\exists h \in \mathbb{H} : |\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| > \frac{\varepsilon}{2}\right) \\
&\leqslant 2|\mathbb{H}| e^{-m\varepsilon^2/2}
\end{aligned}$$

Ponendo quindi il membro di destra uguale a δ e risolviamo rispetto a ε, otteniamo che $\mathrm{er}(\widehat{h}) \leqslant \mathrm{er}(h^*) + \sqrt{\frac{2}{m} \ln \frac{2|\mathbb{H}|}{\delta}}$ vale con probabilità almeno $1 - \delta$ rispetto all'estrazione casuale di un training set di cardinalità m.

Vediamo quindi che il rischio di $\widehat{h}$ si scompone in due attributi: $\mathrm{er}(h^*)$ e $\sqrt{\frac{2}{m} \ln \frac{2|\mathbb{H}|}{\delta}}$. In mancanza di informazioni su (D, μ), e per una fissata cardinalità m del training set, per ridurre $\sqrt{\frac{2}{m} \ln \frac{2|\mathbb{H}|}{\delta}}$, ovvero il maggiorante sull'errore di varianza, dobbiamo ridurre $\mathbb{H}$. Ma questo fa potenzialmente aumentare $\mathrm{er}(h^*)$ e quindi l'errore di bias. La necessità di bilanciare l'errore di varianza e l'errore di bias per controllare il rischio di $\widehat{h}$ è il materializzarsi nella teoria del fenomeno dell'overfitting. Nella dimostrazione del maggiorante sull'errore di varianza abbiamo anche dimostrato che $\forall h \in \mathbb{H} \quad |\widehat{\mathrm{er}}(h) - \mathrm{er}(h)| \leqslant \sqrt{\frac{1}{2m} \ln \frac{2|\mathbb{H}|}{\delta}}$ con probabilità almeno $1 - \delta$ rispetto all'estrazione del training set. Questo significa che quando la cardinalità m del training set è abbastanza grande rispetto a $\ln|\mathbb{H}|$, allora il training error $\widehat{\mathrm{er}}(h)$ diventa una buona stima del rischio $\mathrm{er}(h)$ **simultaneamente** per tutti i classificatori $h \in \mathbb{H}$. In queste condizioni, cioè quando la legge dei grandi numeri vale uniformemente rispetto alla scelta $h \in \mathbb{H}$, è chiaro che qualsiasi algoritmo che sceglie classificatori da $\mathbb{H}$ è protetto dall'overfitting.

5

Rischio nei predittori ad albero

Osservazione 5.1 | Analisi di rischio in classi molto grandi

L'analisi di rischio ci dice che per un training set di taglia m, con probabilità almeno $1-\delta$ si ha che:

$$\mathrm{er}(\widehat{h}) \leqslant \min_{h\in\mathbb{H}} \mathrm{er}(h) + \sqrt{\frac{2}{m}\ln\frac{2|\mathbb{H}|}{\delta}}$$

Se la classe $\mathbb{H}$ è molto grande questo risultato è debole.

Fatto 5.1 | Predittori ad albero e funzioni su vettori binari

Sia $\mathbb{H}$ l'insieme di tutti i classificatori calcolati da predittori ad albero su $\{0,1\}^d$. Allora $\mathbb{H}$ contiene tutte le funzioni della forma $h:\{0,1\}^d \longrightarrow \{-1,+1\}$.

Dimostrazione 5.1 | Predittori ad albero e funzioni su vettori binari

Questo è una conseguenza del fatto che ogni funzione booleana su $\{0,1\}^d$ è rappresentabile con un albero di decisione con $N=\Theta\left(2^d\right)$ nodi.

Osservazione 5.2 | Overfitting nei predittori ad albero

Dato che ci sono 2^{2^d} funzioni della forma $h:\{0,1\}^d \longrightarrow \{-1,+1\}$, con $|\mathbb{H}|=2^{2^d}$ il maggiorante diviene:

$$\mathrm{er}(\widehat{h}) \leqslant \min_{h\in\mathbb{H}} \mathrm{er}(h) + \sqrt{\frac{2}{m}\left(2^d\ln 2+\ln\frac{2}{\delta}\right)}$$

Quindi, perché il rischio di $\widehat{h}$ sia piccolo, il training set deve contenere un numero m di esempi dell'ordine di 2^d, ma dato che $\left|\{0,1\}^d\right|=2^d$ ciò equivale a dire che il training set deve contenere tutte le istanze possibili: **si tratta di un tipico esempio di overfitting**.

Osservazione 5.3 | Controllare l'overfitting nei predittori ad albero

Per controllare l'overfitting possiamo minimizzare il training error focalizzandoci sulla classe $\mathbb{H}_N$ dei predittori ad albero con N nodi su $\{0,1\}^d$, dove $N \ll 2^d$.

Definizione 5.1 | Disuguaglianza di Kraft

Ogni codice binario univocamente decodificabile per i simboli $\alpha \in A = \alpha_1,\ldots,\alpha_N$ deve soddisfare la disuguaglianza:

$$\sum_{i=1}^{N} 2^{-l(\alpha_i)} \leqslant 1$$

Dove $l(\alpha_i)$ è la lunghezza della stringa binaria corrispondente ad α_i. Dati invece gli interi $l(\alpha_i)$, con $i=1,\ldots,N$ soddisfacenti la disuguaglianza precedente, esiste un codice binario per l'alfabeto A tale per cui la parola $C(\alpha_i)$ ha lunghezza $l(\alpha_i)$ e non esiste alcuna parola uguale ad un suo prefisso.

Fatto 5.2 | Limite alla dimensione dei predittori ad albero

Si $\mathbb{H}_N$ la classe di predittori ad albero con N nodi. Allora vale che:

$$|\mathbb{H}_N| \leqslant (2de)^N$$

Dimostrazione 5.2 | Limite alla dimensione dei predittori ad albero

$|\mathbb{H}_N|$ è esprimibile come il prodotto fra:

1. Il numero di alberi binari con N nodi.
2. Il numero di modi di assegnare test binari su attributi ai nodi interni.
3. Il numero di modi di assegnare etichette binarie alle foglie.

Assegnando convenzionalmente il figlio sinistro al risultato negativo di un test e il figlio destro al risultato positivo, un test è definito solamente dall'indice i dell'attributo testato.

Quindi, se l'albero ha M nodi interni, ci sono d^M modi per assegnare i test ai nodi interni. Inoltre, dato che le foglie sono $N-M$, ci sono 2^{N-M} modi per assegnare etichette binarie alle foglie.

Quindi, ogni albero di N nodi può implementare fino a $d^M 2^{N-M} \leqslant d^N$ (dato che $d \geqslant 2$) classificatori.

Infine, il numero di alberi binari con N nodi è dato dall'$(N-1)$-esimo **numero di Catalano**: $C_{N-1} = \frac{1}{N}\binom{2N-2}{N-1}$. Quindi, utilizzando la maggiorazione $\binom{n}{k} \leqslant \left(\frac{en}{k}\right)^k$ derivata dall'approssimazione di Stirling per i coefficienti binomiali, otteniamo:

$$|\mathbb{H}_N| \leqslant \frac{1}{N}\left(\frac{2e(N-1)}{N-1}\right)^{N-1} d^N \leqslant (2ed)^N$$

Osservazione 5.4 | Dimensione del training set per controllare il rischio

Se $\widehat{h} = \operatorname{argmin}_{\mathbb{H}_N} \hat{\text{er}}(h)$ per un N fissato, il maggiorante diviene:

$$\text{er}(\widehat{h}) \leqslant \min_{h \in \mathbb{H}_N} \text{er}(h) + \sqrt{\frac{2}{m}\left(N(1+\ln(2d)) + \ln\frac{2}{\delta}\right)}$$

Da ciò si deduce che in questo caso un training set la cui taglia è dell'ordine di $N \ln d$ è sufficiente per controllare il rischio di $\widehat{h}$ in $\mathbb{H}_N$.

Osservazione 5.5 | Generalizzazione

Il risultato appena dimostrato vale per un predittore specifico, quello che minimizza il training error in $\mathbb{H}_N$ per un dato N fissato. In pratica, non è chiaro come scegliere N, che dovrebbe dipendere dalle caratteristiche del training set. Per aggirare questo problema, invece di limitare l'errore di varianza del predittore che minimizza il training error, come abbiamo fatto finora, mostriamo un risultato diverso. Cioè, maggioriamo simultaneamente il rischio di tutti i predittori ad albero, dove il maggiorante del rischio di ogni albero dipende dal training error e dal numero di nodi dell'albero.

A questo scopo introduciamo una funzione $w : \mathbb{H} \longrightarrow [0,1]$ e chiamiamo $w(h)$ il peso del predittore h. Assumiamo che valga:

$$\sum_{h \in \mathbb{H}} w(h) \leqslant 1$$

Possiamo allora scrivere la seguente catena di disuguaglianze, dove $\varepsilon_h > 0$ è scelto alla fine:

$$\begin{aligned}\mathbb{P}(\exists h \in \mathbb{H} : |\hat{\text{er}}(h) - \text{er}(h)| > \varepsilon_h) &\leqslant \sum_{h \in \mathbb{H}} \mathbb{P}(|\hat{\text{er}}(h) - \text{er}(h)| > \varepsilon_h) \\ &\leqslant \sum_{h \in \mathbb{H}} 2e^{-2m\varepsilon_h^2}\end{aligned}$$

Si noti che abbiamo usato il maggiorante di Chernoff-Hoeffding all'ultimo passo. Scegliendo ora:

$$\varepsilon_h = \sqrt{\frac{1}{2m}\left(\ln\frac{1}{w(h)} + \ln\frac{2}{\delta}\right)}$$

abbiamo che:

$$\mathbb{P}(\exists h \in \mathbb{H} : |\hat{\text{er}}(h) - \text{er}(h)| > \varepsilon_h) \leqslant \sum_{h \in \mathbb{H}} \delta\, w(h) \leqslant \delta$$

dove abbiamo sfruttando la proprietà della funzione w definita precedentemente.

Una conseguenza di questa analisi è che, con probabilità almeno $1-\delta$ rispetto all'estrazione del training set abbiamo che

$$\text{er}(h) \leqslant \hat{\text{er}}(h) + \sqrt{\frac{1}{2m}\left(\ln\frac{1}{w(h)} + \ln\frac{2}{\delta}\right)}$$

simultaneamente per ogni $h \in \mathbb{H}$. Questo suggerisce un algoritmo alternativo alla minimizzazione del training error. Infatti, mentre ERM suggerisce di usare $\widehat{h} = \operatorname{argmin}_{h \in \mathbb{H}_N} \hat{\text{er}}(h)$ per un dato N fissato, l'approccio suggerito dalla nuova analisi propone di usare:

$$\widehat{h} = \operatorname*{argmin}_{h \in \mathbb{H}} \left(\hat{\text{er}}(h) + \sqrt{\frac{1}{2m}\left(\ln\frac{1}{w(h)} + \ln\frac{2}{\delta}\right)}\right)$$

Osservazione 5.6 | Una nuova prospettiva sull'overfitting

La funzione w può essere naturalmente vista come una misura di complessità del classificatore h. Si noti che questa analisi offre una nuova prospettiva sull'**overfitting**: $\hat{\text{er}}(h)$ diventa una buona stima di $\text{er}(h)$ quando viene penalizzato dal termine:

$$\sqrt{\frac{1}{2m}\left(\ln\frac{1}{w(h)} + \ln\frac{2}{\delta}\right)}$$

che rende conto del fatto che abbiamo usato gli m esempi del training set per scegliere un predittore h di complessità $w(h)$.

Esempio 5.1 | Esempio sui predittori ad albero

Sia $\mathbb{H}$ l'insieme dei 2^{2^d} predittori ad albero che codificano tutti i classificatori $h : \{0,1\}^d \longrightarrow \{-1,+1\}$.
Usando tecniche di teoria dei codici, possiamo codificare ogni predittore ad albero h con N_h nodi usando una stringa binaria $\sigma(h)$ di lunghezza $|\sigma(h)| = (N_h+1)\lceil \log_2(d+3)\rceil + 2\lfloor \log_2 N_h \rfloor + 1 = \mathbb{O}\left(N_h \log d\right)$ in modo che non ci siano due predittori h e h' tali che $\sigma(h)$ è prefisso di $\sigma\left(h'\right)$. Codici di questo tipo si chiamano **istantanei** e soddisfano la **disuguaglianza di Kraft**.

Grazie alla disuguaglianza di Kraft possiamo assegnare il peso $w(h) = 2^{-|\sigma(h)|}$ ad un classificatore h da un predittore ad albero con N_h nodi.

Applicando il maggiorante otteniamo che, con probabilità almeno $1-\delta$ rispetto all'estrazione del training set,

$$\mathrm{er}(h) \leqslant \hat{\mathrm{er}}(h) + \sqrt{\frac{1}{2m}\left(|\sigma(h)| + \ln\frac{2}{\delta}\right)} \quad \text{con } |\sigma(h)| = \mathbb{O}\left(N_h \log d\right)$$

simultaneamente per ogni $h \in \mathbb{H}$. Quindi, un algoritmo di apprendimento con alberi può controllare l'overfitting generando predittori $\widehat{h}$ definiti come:

$$\widehat{h} = \operatorname*{argmin}_{h\in\mathbb{H}} \left(\hat{\mathrm{er}}(h) + \sqrt{\frac{1}{2m}\left(|\sigma(h)| + \ln\frac{2}{\delta}\right)} \right)$$

Questo tipo di analisi giustifica l'osservazione empirica che, a parità d training error risulta generalmente più affidabile il classificatore ad albero con minor numero di nodi. D'altra parte, esiste un'arbitrarietà nella scelta della funzione di complessità w. In particolare, non è detto che w debba necessariamente essere inversamente proporzionale al numero di nodi dell'albero: possiamo scegliere una qualsiasi altra w a patto che soddisfi la proprietà $\left(\sum_{h\in\mathbb{H}} w(h) \leqslant 1\right)$.

È quindi corretto interpretare w come un bias che orienta la nostra preferenza verso certi tipi di alberi rispetto ad altri.

La scelta del bias che, a parità di training error, privilegia alberi più piccoli è conforme al principio del Rasoio di Occam: una regola euristica secondo la quale fra due spiegazioni alternative di uno stesso fenomeno, la più corta è anche quella più affidabile.

6

Rischio nell'algoritmo Nearest Neighbour

Definizione 6.1 | Predittore 1-NN

Dato un training set $S = (\underline{x}_1, y_1), \ldots, (\underline{x}_m, y_m)$ estratto da un modello statistico (D, μ) definiamo la mappa $\pi_S : \mathbb{R}^d \longrightarrow \{1, \ldots, m\}$ come:

$$\pi_S(\boldsymbol{x}) = \underset{t=1,\ldots,m}{\operatorname{argmin}} \|\boldsymbol{x} - \boldsymbol{X}_t\|$$

Il predittore 1-NN su input S è quindi definito da $\widehat{h}_S(\underline{\boldsymbol{x}}) = y_{\pi_S}(\underline{\boldsymbol{x}})$.

Teorema 6.1 | Rischio nell'algoritmo Nearest Neighbour

Una maggiorazione del rischio statistico del classificatore prodotto da 1-NN rispetto all'estrazione del training set valida é:

$$\mathbb{E}\left[\operatorname{er}\left(\widehat{h}_S\right)\right] \leqslant 2\operatorname{er}\left(f^*\right) + \varepsilon_m$$

dove $\widehat{h}_S$ indica il classificatore 1-NN prodotto sul training set S di taglia m, $\operatorname{er}\left(f^*\right)$ è il rischio del classificatore Bayesiano ottimo e ε_m è una quantità che dipende da m

Osservazione 6.1 | Differenze tra classificatore ERM e 1-NN

Le differenze sono principalmente di due tipi:

1. Il maggiorante del rischio del classificatore $1-NN$ in termini assoluti (cioè si confronta direttamente col Bayes risk), mentre il maggiorante dell'ERM lo maggiora relativamente al rischio del miglior classificatore in $\mathbb{H}$, che potrebbe essere arbitrariamente peggiore del Bayes risk.
2. Il maggiorante ERM descrive una proprietà della distribuzione del rischio dei classificatori ERM al variare di $\varepsilon > 0$, mentre l'altro limita semplicemente il rischio di un tipico classificatore $1-NN$.

Dimostrazione 6.1 | Rischio nell'algoritmo Nearest Neighbour

Assumiamo che i dati $\underline{\boldsymbol{x}}$ generati dalla sorgente siano tali che $\max_i |x_i| \leqslant 1$ con probabilità 1: vale a dire che le componenti x_i delle istanze $\underline{\boldsymbol{x}} = (x_1, \dots, x_d)$ generate dalla sorgente hanno sempre valori compresi fra -1 e $+1$. Sia $\mathbb{X} \subset \mathbb{R}^d$ il sottoinsieme delle istanze $\underline{\boldsymbol{x}}$ con questa proprietà.

Esprimeremo il maggiorante sul valore atteso del rischio di 1-NN in termini di una quantità che caratterizza il modello statistico (D, μ), ovvero il più piccolo $c > 0$ tale che:

$$|\eta(\boldsymbol{x}) - \eta(\boldsymbol{x}')| \leqslant c \|\boldsymbol{x} - \boldsymbol{x}'\| \qquad \text{per ogni } \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$$

Si noti che $c < \infty$ implica che μ è una funzione continua. Possiamo quindi scrivere:

$$\eta(\boldsymbol{x}') \leqslant \eta(\boldsymbol{x}) + c \|\boldsymbol{x} - \boldsymbol{x}'\|$$
$$1 - \eta(\boldsymbol{x}') \leqslant 1 - \eta(\boldsymbol{x}) + c \|\boldsymbol{x} - \boldsymbol{x}'\|$$

Siccome i dati sono estratti in modo indipendente, per ogni $(\underline{\boldsymbol{x}}, y)$ e $(\underline{\boldsymbol{x}}', y')$ vale che:

$$\mathbb{P}(Y = y, Y' = y' | \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{X}' = \boldsymbol{x}') = \mathbb{P}(Y = y | \boldsymbol{X} = \boldsymbol{x}) \mathbb{P}(Y' = y' | \boldsymbol{X}' = \boldsymbol{x}')$$

Ricordando che $\widehat{h}_S(\underline{\boldsymbol{x}}) = y_{\pi_S(\underline{\boldsymbol{x}})}$, notiamo che, per ogni coppia di istanze $\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}'$ vale che:

$$\begin{aligned}\mathbb{P}(Y \neq Y' | \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{X}' = \boldsymbol{x}') &= \mathbb{P}(Y = 1 | \boldsymbol{X} = \boldsymbol{x}) \mathbb{P}(Y' = -1 | \boldsymbol{X}' = \boldsymbol{x}') \\ &+ \mathbb{P}(Y = -1 | \boldsymbol{X} = \boldsymbol{x}) \mathbb{P}(Y' = 1 | \boldsymbol{X}' = \boldsymbol{x}') \\ &= \eta(\boldsymbol{x})(1 - \eta(\boldsymbol{x}')) + (1 - \eta(\boldsymbol{x}))\eta(\boldsymbol{x}')\end{aligned}$$

dove la probabilità è rispetto all'estrazione di S e $(\underline{\boldsymbol{X}}, Y)$ e abbiamo usato l'equazione delle probabilità precedente. Applicando le due disuguaglianze, possiamo allora scrivere:

$$\begin{aligned}&\mathbb{E}[\mathrm{er}(\widehat{h}_S)] = \mathbb{E}[\mathbb{I}\{\widehat{h}_S(\boldsymbol{X}) \neq Y\}] \\ &= \mathbb{P}(\widehat{h}_S(\boldsymbol{X}) \neq Y) \\ &= \mathbb{P}(Y_{\pi_S(\boldsymbol{X})} \neq Y) \\ &= \mathbb{E}[\mathbb{I}\{Y_{\pi_S(\boldsymbol{X})} \neq Y\}] \\ &= \mathbb{E}[\mathbb{E}[\mathbb{I}\{Y' \neq Y\} | \boldsymbol{X} = \boldsymbol{x}, \boldsymbol{X}_{\pi_S(\boldsymbol{X})} = \boldsymbol{x}']] \\ &= \mathbb{E}[\eta(\boldsymbol{X})(1 - \eta(\boldsymbol{X}')) + (1 - \eta(\boldsymbol{X}))\eta(\boldsymbol{X}')] \\ &\leqslant \mathbb{E}[\eta(\boldsymbol{X})(1 - \eta(\boldsymbol{X})) + \eta(\boldsymbol{X}) c \|\boldsymbol{X} - \boldsymbol{X}'\| + (1 - \eta(\boldsymbol{X}))\eta(\boldsymbol{X}) + (1 - \eta(\boldsymbol{X})) c \|\boldsymbol{X} - \boldsymbol{X}'\|] \\ &\leqslant 2\mathbb{E}[\eta(\boldsymbol{X})(1 - \eta(\boldsymbol{X}))] + c\mathbb{E}[\|\boldsymbol{X} - \boldsymbol{X}_{\pi_S(\boldsymbol{X})}\|]\end{aligned}$$

dove i valori attesi e le probabilità sono rispetto alle estrazioni indipendenti di S e di $(\underline{\boldsymbol{X}}, Y)$. Ora ricordando che il rischio del classificatore Bayesiano ottimo f^* soddisfa:

$$\mathrm{er}(f^*) = \mathbb{E}[\min\{\eta(\boldsymbol{X}), 1 - \eta(\boldsymbol{X})\}] \geqslant \mathbb{E}[\eta(\boldsymbol{X})(1 - \eta(\boldsymbol{X}))]$$

Quindi vale che:

$$\mathbb{E}[\mathrm{er}(\widehat{h}_S)] \leqslant 2\,\mathrm{er}(f^*) + c\mathbb{E}[\|\boldsymbol{X} - \boldsymbol{X}_{\pi_S(\boldsymbol{X})}\|]$$

TO BE CONTINUED...

Conclusione 6.1 |

Da questa analisi è possibile trarre due conclusioni:

1. Per $m \to \infty$, $\mathrm{er}(f^*) \leqslant \mathbb{E}[\mathrm{er}(\widehat{h}_S)] \leqslant 2\,\mathrm{er}(f^*)$: vale a dire che il rischio del 1-NN è compreso fra il Bayes error e due volte il Bayes error.
2. Perché $\mathbb{E}[\mathrm{er}(\widehat{h}_S)]$ sia al più $2\mathrm{er}(f^*) + \varepsilon$ il training set dev'essere almeno di taglia $m \geqslant \left(\frac{4c}{\varepsilon}\sqrt{d}\right)^{d+1}$, cioè esponenziale nel numero di d attributi. Questo mostra che 1-NN paga un prezzo alto in termini di overfitting per potersi confrontare direttamente con il Bayes risk.

Generalizzazione 6.1 | Da 1-NN a k-NN

La dimostrazione può essere generalizzata per studiare sotto le medesime assunzioni il rischio del classificatore $\widehat{h}_S$ prodotto da k-NN, che risulta essere maggiorato come segue:

$$\mathbb{E}\left[\mathrm{er}\left(\widehat{h}_S\right)\right] \leqslant \left(1+\sqrt{\frac{8}{k}}\right)\mathrm{er}\left(f^*\right)+\mathbb{O}\left(km^{-1/(d+1)}\right)$$

7

Consistenza e algoritmi nonparametrici

Definizione 7.1 | Algoritmo consistente

L'algoritmo A è detto consistente rispetto a un modello statistico (D,μ) quando:

$$\lim_{m\to\infty}\mathbb{E}\left[\mathrm{er}\left(A\left(S_m\right)\right)\right]=\mathrm{er}\left(f^*\right)$$

dove il valore atteso è rispetto all'estrazione del training set S_m da (D,μ). Quindi la **consistenza** è una proprietà asintotica che certifica la capacità dell'algoritmo di raggiugere in media le prestazioni del Bayesiano ottimo al crescere del training set.

Teorema 7.1 | No Free Lunch

Sia $a_1, a_2, \ldots$ una successione convergente a zero di numeri positivi tali che $\frac{1}{16}\geqslant a_1\geqslant a_2\geqslant\ldots$.

Per ogni algoritmo di apprendimento A esiste un modello statistico (D,μ) tale che $\mathrm{er}\left(f^*\right)=0$ e simultaneamente $\mathbb{E}\left[\mathrm{er}\left(A\left(S_m\right)\right)\right]\geqslant a_m$ per ogni $m\geqslant 1$.

Osservazione 7.1 | Significato del No Free Lunch

Una domanda che ci si può porre nel campo nel ML è "Quale è l'algoritmo di ML migliore?": la risposta del "No-Free Lunch" è "non ne esiste uno". Dati due algoritmi di ML, le loro performance medie sull'intero set dei problemi di ML che loro devono risolvere risulterebbe che nessuno dei due è migliore dell'altro.

Osservazione 7.2 | Cosa implica il No Free Lunch?

Un algoritmo consistente può risultare in pratica peggiore di uno non consistente per la sua **velocità di convergenza**: infatti, la consistenza per ogni modello (D,μ) può essere pagata con una velocità di convergenza *arbitrariamente lenta* al Bayes error.

Il teorema non impedisce a un algoritmo consistente di convergere rapidamente per specifiche distribuzioni (D,μ) ed analogamente un algoritmo non consistente può convergere rapidamente a specifici valori di rischio a patto che siano superiori al Bayes error. Il teorema afferma solo che se A arriva al Bayes error, allora non è possibile che ci arrivi in fretta per tutte le distribuzioni.

Definizione 7.2 | Algoritmi non-parametrici

Gli algoritmi consistenti vengono anche detti algoritmi **non-parametrici**: i classificatori prodotti da questi algoritmi non sono rappresentabili con un numero predeterminato di parametri. La struttura di un classificatore nonparametrico non è fissata, ma viene determinata dai dati di training.

Esempio 7.1 | Consistenza dell'algoritmo k-NN

Per ogni (D,μ) vale che:

$$\lim_{m\to\infty}\mathbb{E}\left[\mathrm{er}\left(k-\mathrm{NN}\left(S_m\right)\right)\right]\leqslant\mathrm{er}\left(f^*\right)+2\sqrt{\frac{\mathrm{er}\left(f^*\right)}{k}}$$

Perciò per ogni k fissato, k-NN non è consistente. Tuttavia, se rendiamo k dipendente da m in modo che non cresca troppo velocemente, cioè se $k=k_m$, tale che $k_m\to\infty$ e $k_m=o(m)$, allora è possibile dimostrare che k-NN diventa consistente. Possiamo confrontare il risultato con quello ottenuto tramite l'analisi di rischio:

$$\mathbb{E}\left[\mathrm{er}\left(k-\mathrm{NN}\left(S_m\right)\right)\right]\leqslant\left(1+\sqrt{\frac{8}{k}}\right)\mathrm{er}\left(f^*\right)+\mathbb{O}\left(km^{-1/(d+1)}\right)$$

che era stato ottenuto sotto l'assunzione che ci si trova nella condizione seguente:

$$\left|\eta(\boldsymbol{x})-\eta\left(\boldsymbol{x}'\right)\right|\leqslant c\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|\qquad\text{per ogni }\boldsymbol{x},\boldsymbol{x}'\in\mathbb{X}$$

Si noti che se $\mathrm{er}\left(f^*\right)=0$ (prima condizione del no free lunch theorem) allora dev'essere $\mu\left(\underline{\boldsymbol{x}}\right)\in\{0,1\}$ per ogni $\underline{\boldsymbol{x}}$. Ovvero μ è discontinua e la condizione non può essere rispettata.

Esempio 7.2 | Consistenza dell'algoritmo greedy di costruzione di classificatori ad albero

Sotto determinate assunzioni sul modello statistico (D,μ) è possibile mostrare che anche l'algoritmo greedy di costrizione di classificatori ad albero diventa consistente quando ogni foglia è finale per un numero sufficiente di punti del training set.

Osservazione 7.3 | Parametrico o nonparametrico?

In generale:

1. Se il training set è piccolo conviene utilizzare un algoritmo parametrico.
2. Se d è grande conviene usare un algoritmo parametrico perché anche quando μ soddisfa la condizione:

   $$\left|\eta(\boldsymbol{x})-\eta\left(\boldsymbol{x}'\right)\right|\leqslant c\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|\qquad\text{per ogni }\boldsymbol{x},\boldsymbol{x}'\in\mathbb{X}$$

   la convergenza **non parametrica** è comunque dell'ordine $\mathbb{O}\left(m^{-1}/(d+1)\right)$ mentre quella parametrica è dell'ordine $\mathbb{O}\left(m^{-1/2}\right)$.
3. Negli altri casi, con dataset grandi con pochi attributi, possiamo provare ad usare un algoritmo nonparametrico.

8

Compression bounds

Definizione 8.1 | sotto-sequenza

Per ogni $K \subseteq \{1,\dots,m\}$ indichiamo con S_K la sotto-sequenza del training set che contiene soltanto gli elementi indicizzati da K e indichiamo con $S_{\overline{K}}$ la sotto-sequenza che contiene soltanto gli elementi indicizzati da $\{1,\dots,m\} \setminus K$.

Definizione 8.2 | Sketch

Data una sequenza di esempi $S = \big((\underline{\boldsymbol{x}}_1, y_1),\dots,(\underline{\boldsymbol{x}}_m, y_m)\big)$, chiamiamo **sketch** per un algoritmo di classificazione A una qualsiasi sotto-sequenza S_K di S tale che $A\big(S'\big) = A(S)$ per ogni sotto-sequenza $S_K \subseteq S' \subset S$.

Quindi A con input S_K genera lo stesso classificatore di A con input S: una sotto-sequenza con questa proprietà esiste sempre dato che S_K può essere anche uguale ad S.

Obbiettivo 8.1 |

Denotando:

$$\widehat{h} = A(S) = A(S_K)$$

vogliamo **limitare il rischio** $\mathrm{er}\big(\widehat{h}\big)$ in termini di $\widetilde{\mathrm{er}}\big(\widehat{h}\big)$, dove:

$$\widetilde{\mathrm{er}}(\widehat{h}) = \frac{1}{\big|S_{\overline{K}}\big|} \sum_{(\boldsymbol{x}_t, y_t) \in S_{\overline{K}}} \mathbb{I}\{\widehat{h}(\boldsymbol{x}_t) \neq y_t\}$$

è la frazione di errori commessi da $\widehat{h}$ sulla sotto-sequenza di training set $S_{\overline{K}}$ che non include gli esempi dello sketch.

Analisi 8.1 |

Per comodità consideriamo solo algoritmi che ammettano sempre uno sketch di cardinalità al più metà di quella del training set di partenza.

Fissiamo quindi un algoritmo A, un training set S di cardinalità m e uno sketch S_K per A di cardinalità $|K| \leqslant \frac{m}{2}$. È importante notare a questo punto che, per A fissato, K è una funzione del campione casuale S. Introducendo le costanti $\varepsilon_j > 0$ da determinare in seguito, notiamo che:

$$\mathrm{er}(A(S)) > \tilde{\mathrm{er}}(A(S)) + \varepsilon_{|K|} \quad \Rightarrow \quad \exists \text{ sketch } J, |J| \leqslant \frac{m}{2}, \mathrm{er}\big(A\big(S_J\big)\big) > \tilde{\mathrm{er}}\big(A\big(S_J\big)\big) + \varepsilon_{|J|}$$

La precedente implicazione permette di spezzare la variabile casuale $|K|$ nell'unione dei suoi possibili valori $|J|$, ottenendo:

$$\begin{aligned}\mathbb{P}\left(\mathrm{er}(A(S)) > \tilde{\mathrm{er}}(A(S)) + \varepsilon_{|K|}\right) &\leqslant \mathbb{P}\left(\exists \mathrm{sketch}\, J, |J| \leqslant \frac{m}{2}, \mathrm{er}\big(A\big(S_J\big)\big) > \tilde{\mathrm{er}}\big(A\big(S_J\big)\big) + \varepsilon_{|J|}\right) \\ &\leqslant \sum_{j=0}^{m/2} \sum_{J \text{ sketch:}} \mathbb{P}\left(\mathrm{er}\big(A\big(S_J\big)\big) > \tilde{\mathrm{er}}\big(A\big(S_J\big)\big) + \varepsilon_j\right)\end{aligned}$$

dove abbiamo usato la regola della somma $\mathbb{P}(A \cup B) \leqslant \mathbb{P}(A) + \mathbb{P}(B)$ ed ora le ε_j sono quantità deterministiche. Si noti che $\tilde{\mathrm{er}}\big(A\big(S_J\big)\big)$ denota la frazione di errori di $A\big(S_J\big)$ sugli esempi $(\underline{\boldsymbol{x}}_t, y_t)$ del training set tali che $t \notin J$. Ora, in ciascuna probabilità

$$\mathbb{P}\left(\mathrm{er}\big(A\big(S_J\big)\big) > \tilde{\mathrm{er}}\big(A\big(S_J\big)\big) + \varepsilon_j\right)$$

il classificatore $A\big(S_J\big)$ per definizione è indipendente da tutti gli $m - j$ esempi $(\underline{\boldsymbol{x}}_t, y_t)$ di training tali che $t \notin J$. Quindi $\tilde{\mathrm{er}}\big(A\big(S_J\big)\big)$, che è proprio determinato da questi $m - j$ esempi, è una media campionaria di un classificatore fissato ed ha valore atteso $\mathrm{er}\big(A\big(S_J\big)\big)$.

Possiamo dunque applicare il maggiorante di Chernoff-Hoeffding ottenendo:

$$\mathbb{P}\left(\mathrm{er}(A(S)) > \tilde{\mathrm{er}}(A(S)) + \varepsilon_{|K|}\right) \leqslant \sum_{j=0}^{m/2} \sum_{J:|J|=j} e^{-2(m-j)\varepsilon_j^2}$$

Ora, scegliendo come ε_j:

$$\varepsilon_j = \sqrt{\frac{1}{m}\left(\ln\frac{1}{w_j} + \ln\frac{1}{\delta}\right)}$$

dove i pesi $w_j \geqslant 0$, che determineremo in seguito, soddisfano

$$\sum_{j=0}^{m/2} \sum_{J:|J|=j} w_j \leqslant 1$$

otteniamo

$$\begin{aligned}\sum_{j=0}^{m/2} \sum_{J:|J|=j} e^{-2(m-j)\varepsilon_j^2} &\leqslant \sum_{j=0}^{m/2} \sum_{J:|J|=j} e^{-m\varepsilon_j^2} \\ &\leqslant \sum_{j=0}^{m/2} \sum_{J|=j} \delta w_j \\ &\leqslant \delta\end{aligned}$$

Quindi, con probabilità almeno $1 - \delta$ rispetto all'estrazione del training set abbiamo:

$$\mathrm{er}(A(S)) \leqslant \tilde{\mathrm{er}}(A(S)) + \sqrt{\frac{1}{m}\left(\ln\frac{1}{w_{|K|}} + \ln\frac{1}{\delta}\right)}$$

Si noti ora che per soddisfare $\sum_{j=0}^{m/2} \sum_{J:|J|=j} w_j \leqslant 1$ è sufficiente definire il peso w_j come:

$$w_j = \frac{1}{m\binom{m}{j}}$$

Utilizzando il maggiorante $\binom{m}{j} \leqslant \left(\frac{em}{j}\right)^j$ vediamo che:

$$\ln\frac{1}{w_j} = \ln m + \ln\binom{m}{j} \leqslant \ln m + j\ln\frac{em}{j} \leqslant j + (j+1)\ln m$$

Quindi, con probabilità almeno $1 - \delta$ rispetto all'estrazione del training set abbiamo infine

$$\mathrm{er}(A(S)) \leqslant \tilde{\mathrm{er}}(A(S)) + \sqrt{\frac{1}{m}\left(|K| + (|K|+1)\ln m + \ln\frac{1}{\delta}\right)}$$

9 Classificatori lineari

Definizione 9.1 | Norma euclidea

La norma euclidea di un vettore $\underline{\boldsymbol{x}} = (x_1, \ldots, x_d)$ è pari a:

$$\|\boldsymbol{x}\| = \sqrt{\sum_{i=1}^{d} x_i^2}$$

Definizione 9.2 | Prodotto interno

Si ricordi che, detto θ l'angolo fra due vettori $\underline{\boldsymbol{v}}$ e $\underline{\boldsymbol{x}}$, la quantità $\underline{\boldsymbol{v}}^T\underline{\boldsymbol{x}}$, la quantità $\underline{\boldsymbol{v}}^T\underline{\boldsymbol{x}} = \|\underline{\boldsymbol{v}}\|\,\|\underline{\boldsymbol{x}}\|\cos\theta$ è la lunghezza della proiezione di $\underline{\boldsymbol{x}}$ su $\underline{\boldsymbol{v}}$ moltiplicata per $\|\underline{\boldsymbol{v}}\|$ o, equivalentemente, la lunghezza della proiezione di $\underline{\boldsymbol{v}}$ su $\underline{\boldsymbol{x}}$ moltiplicata per $\|\underline{\boldsymbol{x}}\|$.

Definizione 9.3 | Iperpiano

Algebricamente, un iperpiano è il luogo dei punti $\underline{\boldsymbol{x}} \in \mathbb{R}^d$ che soddisfano l'equazione $v_1x_1 + \ldots + v_dx_d = c$ dove $v_1, \ldots, v_n, c$ sono coefficienti reali. Se definiamo la notazione

$$\boldsymbol{u}^\top\boldsymbol{v} = \sum_{i=1}^{d} u_i v_i$$

per il prodotto interno, possiamo riscrivere l'iperpiano come:

$$S(\boldsymbol{v}, c) = \left\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{v}^\top\boldsymbol{x} = c\right\}$$

Osservazione 9.1 | Interpretazione geometrica di classificazione binario

Possiamo rappresentare geometricamente un classificatore binario $h : \mathbb{R}^d \longrightarrow \{-1, +1\}$ con la partizione $\{S^+, S^-\}$ di $\mathbb{R}^d$ tale che:

$$h(\underline{\boldsymbol{x}}) = \begin{cases} +1 & \text{se } \underline{\boldsymbol{x}} \in S^+ \\ -1 & \text{se } \underline{\boldsymbol{x}} \in S^- \end{cases}$$

Definizione 9.4 | Classificatore lineare per il caso $\mathbb{R}^d$

I classificatori lineari sono quei classificatori h dove S^+ e S^- sono i semispazi definiti da un iperpiano S in $\mathbb{R}^d$.

Proprietà 9.1 | Vettore perpendicolare ad iperpiano

Dato un iperpiano $S = S(\underline{\boldsymbol{v}}, c)$ e considerando $\underline{\boldsymbol{v}}^T\underline{\boldsymbol{x}}$ come la lunghezza della proiezione di $\underline{\boldsymbol{x}}$ su $\underline{\boldsymbol{v}}$ moltiplicata per $\|\underline{\boldsymbol{v}}\|$, si ha che il vettore $\underline{\boldsymbol{v}}$ è perpendicolare all'iperpiano S che lo taglia alla distanza $\frac{c}{\|\underline{\boldsymbol{v}}\|}$ dall'origine.

Definizione 9.5 | Semispazi S^+ ed S^-

I semispazi S^+ e S^- definiti da $S = \{\underline{\boldsymbol{x}} : \underline{\boldsymbol{v}}^T\underline{\boldsymbol{x}} = c\}$ sono:

$$S^+ = \{\boldsymbol{x} : \boldsymbol{v}^\top\boldsymbol{x} > c\} \quad \text{e} \quad S^- = \{\boldsymbol{x}' : \boldsymbol{v}^\top\boldsymbol{x}' \leqslant c\}$$

ovvero l'insieme dei vettori $\underline{\boldsymbol{x}}$ tali che la proiezione su $\underline{\boldsymbol{v}}$ è almeno $\frac{c}{\|\underline{\boldsymbol{v}}\|}$ e l'insieme dei vettori $\underline{\boldsymbol{x}}'$ tali che la proiezione su $\underline{\boldsymbol{v}}$ è minore di $\frac{c}{\|\underline{\boldsymbol{v}}\|}$.

Definizione 9.6 | Classificatore lineare associato all'iperpiano

Possiamo rappresentare un classificatore lineare h associato all'iperpiano $S(\underline{\boldsymbol{v}}, c)$ come $h(\underline{\boldsymbol{x}}) = \operatorname{sgn}(\underline{\boldsymbol{v}}^T\underline{\boldsymbol{x}} - c)$, dove la funzione sgm è definita come:

$$\operatorname{sgn}(x) = \begin{cases} 1 & \text{se } x > 0 \\ -1 & \text{altrimenti} \end{cases}$$

Definizione 9.7 | Iperpiano omogeneo

Iperpiani della forma $S(\underline{\boldsymbol{v}}, 0) = \{\underline{\boldsymbol{x}} \in \mathbb{R}^d : \underline{\boldsymbol{v}}^T\underline{\boldsymbol{x}} = 0\}$ passano per l'origine e vengono detti **iperpiani omogenei**.

Definizione 9.8 | Equivalenza tra iperpiano omogeneo e non omogeneo

Un iperpiano non omogeneo $S(\underline{\boldsymbol{v}}, c)$ in d dimensioni è equivalente all'iperpiano omogeneo $S(\underline{\tilde{\boldsymbol{v}}}, 0)$ in $d+1$ dimensioni con $\underline{\tilde{\boldsymbol{v}}} = (x_1, \ldots, x_d, 1) \in \mathbb{R}^{d+1}$. Infatti, $\operatorname{sgn}(\boldsymbol{v}^\top\boldsymbol{x} - c) = \operatorname{sgn}(\tilde{\boldsymbol{v}}^\top\tilde{\boldsymbol{x}})$.

Definizione 9.9 | Problema di apprendimento di classificatori lineari

Detta $\mathbb{H}_d$ la famiglia dei classificatori lineari $h(\underline{\boldsymbol{x}}) = \operatorname{sgn}(\boldsymbol{w}^\top\boldsymbol{x})$ per $\underline{\boldsymbol{\omega}} \in \mathbb{R}^d$, consideriamo l'algoritmo ERM che, dato un training set $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m) \in \mathbb{R}^d \times \{-1, +1\}$ trova:

$$\widehat{h} = \operatorname*{argmin}_{h \in \mathbb{H}_d} \frac{1}{m} \sum_{t=1}^{m} \mathbb{I}\{h(\boldsymbol{x}_t) \neq y_t\}$$

Il problema di decisione associato è NP-completo, anche quando $\underline{\boldsymbol{x}}_t \in \{0, 1\}^d$ per $t = 1, \ldots, m$.

Definizione 9.10 | MinDisagreement

Il problema di decisione MinDisagreement è definito come:

Istanza: Coppie $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m) \in \{0,1\}^d \times \{-1,+1\}$ ed un numero intero k.

Domanda: Esiste un vettore $\underline{\boldsymbol{\omega}} \in \mathbb{Q}^d$ tale che $y_t \underline{\boldsymbol{\omega}}^T \underline{\boldsymbol{x}}_t \leqslant 0$ per al più k indici $t = 1, \ldots, m$?

Il problema **MinDisagreement** è NP-completo.

Definizione 9.11 | MinDisOpt

Il problema di MinDisOpt è definito come:

Istanza: Coppie $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m) \in \{0,1\}^d \times \{-1,+1\}$.

Domanda: Un vettore $\underline{\boldsymbol{\omega}} \in \mathbb{Q}^d$ che minimizzi il numero di indici $t = 1, \ldots, m$ tali che $y_t \underline{\boldsymbol{\omega}}^T \underline{\boldsymbol{x}}_t \leqslant 0$.

Data un'istanza S di MinDisOpt, cioè un training set, sia $\mathrm{Opt}(S)$ il numero di esempi di S classificati in modo errato dall'iperpiano ottimo.

Teorema 9.1 |

Se $P \neq NP$, per ogni $c > 0$ non esiste un algoritmo polinomiale che risolva ogni istanza S di MinDisOpt con un numero di esempi classificati in modo errato pari ad al più

$$c \cdot \mathrm{Opt}(S)$$

Questo significa che a meno che $P = NP$, cosa ritenuta improbabile, non è possibile trovare un algoritmo che approssimi la soluzione di $\widehat{h} = \operatorname*{argmin}_{h \in \mathbb{H}_d} \frac{1}{m} \sum_{t=1}^{m} \mathbb{I}\{h(\boldsymbol{x}_t) \neq y_t\}$ a meno di un fattore costante in tempo polinomiale nella dimensione dell'input, ovvero polinomiale in m e d.

Definizione 9.12 | Margine di un training set

Dato un training set $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m)$, per ogni iperpiano $\underline{\boldsymbol{u}}$ definiamo il **margine**:

$$\gamma(\boldsymbol{u}) \stackrel{\text{def}}{=} \min_{t=1,\ldots,m} y_t \boldsymbol{u}^\top \boldsymbol{x}_t$$

Definizione 9.13 | Training set linearmente separabile

Un training set è linearmente separabile quando esiste un iperpiano separatore $\underline{\boldsymbol{u}} \in \mathbb{R}^d$ tale che $\gamma(\boldsymbol{u}) > 0$.

Si noti che $\frac{\gamma(\boldsymbol{u})}{|\boldsymbol{u}|}$ è la distanza dall'iperpiano separatore $\underline{\boldsymbol{u}}$ dell'elemento del training set ad esso più vicino. Dato che $\gamma(\underline{\boldsymbol{u}})$ può essere moltiplicato per un qualunque fattore positivo riscalando $\underline{\boldsymbol{u}}$, convenzionalmente assumiamo che un iperpiano separatore soddisfi sempre $\gamma(\underline{\boldsymbol{u}}) \geqslant 1$.

Osservazione 9.2 | Problema di apprendimento su dataset linearmente separabili

In caso di training set linearmente separabile, il problema $\widehat{h} = \operatorname*{argmin}_{h \in \mathbb{H}_d} \frac{1}{m} \sum_{t=1}^{m} \mathbb{I}\{h(\boldsymbol{x}_t) \neq y_t\}$ è equivalente al seguente problema di programmazione lineare: trova un vettore $\underline{\boldsymbol{\omega}} \in \mathbb{R}^d$ che soddisfi le disequazioni lineari:

$$y_t \boldsymbol{w}^\top \boldsymbol{x}_t > 0 \quad t = 1, \ldots, m$$

Questo problema può quindi essere risolto in tempo polinomiale usando un qualsiasi algoritmo efficiente di programmazione lineare.

Definizione 9.14 | Perceptrone

L'algoritmo del perceptrone costruisce un classificatore lineare omogeneo esaminando gli elementi del training set in modo incrementale: l'analisi del training set avviene aggiornando il modello lineare corrente, che è rappresentato da un iperpiano omogeneo con parametri $\underline{\boldsymbol{\omega}}$, ogni volta che questa sbaglia a classificare il prossimo elemento $(\underline{\boldsymbol{x}}_t, y_t)$ del training set.

Input: Training set $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m)$.

Inizializza: $\underline{\boldsymbol{\omega}} = (0, \ldots, 0)$.

Ripeti: ˜

Leggi il prossimo esempio $(\underline{\boldsymbol{x}}_t, y_t)$ nel training set. Se $y_t \underline{\boldsymbol{\omega}}^T \underline{\boldsymbol{x}}_t \leqslant 0$, allora $\underline{\boldsymbol{\omega}} \leftarrow \underline{\boldsymbol{\omega}} + y_t \underline{\boldsymbol{x}}_t$

Fino a che: $y_t \underline{\boldsymbol{\omega}}^T \underline{\boldsymbol{x}}_t > 0$ per ogni $t = 1, \ldots, m$

L'algoritmo termina se $\underline{\boldsymbol{\omega}}$ è un iperpiano separatore. L'aggiornamento $\underline{\boldsymbol{\omega}} \leftarrow \underline{\boldsymbol{\omega}} + y_t \underline{\boldsymbol{x}}_t$ quando $y_t \underline{\boldsymbol{\omega}}^T \underline{\boldsymbol{x}}_t \leqslant 0$ aumenta il valore di $y_t \underline{\boldsymbol{\omega}}^T \underline{\boldsymbol{x}}_t$. Infatti:

$$y_t (\boldsymbol{w} + y_t \boldsymbol{x}_t)^\top \boldsymbol{x}_t = y_t \boldsymbol{w}^\top \boldsymbol{x}_t + \|\boldsymbol{x}_t\|^2 > y_t \boldsymbol{w}^\top \boldsymbol{x}_t$$

Geometricamente, l'algoritmo sposta $\underline{\boldsymbol{\omega}}$ verso $\underline{\boldsymbol{\omega}}_t$ se $y_t = 1$ e lo allontana da $\underline{\boldsymbol{x}}_t$ se $y_t = -1$.

Corollario 9.1 |

Sia S un training set di m esempi estratti in modo indipendenti ed identicamente distribuiti da un modello statistico (D, μ) fissato ma ignoto. Supponiamo che l'algoritmo del **Perceptrone** con inpit S termini con output $\underline{\boldsymbol{\omega}}$ e sia $h_{\underline{\boldsymbol{\omega}}}$ il classificatore lineare definito da $\underline{\boldsymbol{\omega}}$.

Se $M \leqslant \frac{m}{2}$ è la cardinalità del sottoinsieme di S degli esempi su cui è stato effettuato almeno un aggiornamento, allora:

$$\mathrm{er}(h_{\underline{\boldsymbol{\omega}}}) \leqslant \widetilde{\mathrm{er}}(h_{\underline{\boldsymbol{\omega}}}) + \sqrt{\frac{1}{m}\left((M+1)\ln m + \ln \frac{e}{\delta}\right)}$$

con probabilità almeno $1 - \delta$ rispetto all'estrazione del training set S, dove $\widetilde{er}(h_{\underline{\boldsymbol{\omega}}})$ denota la frazione di errori compiuti da $h_{\underline{\boldsymbol{\omega}}}$ sui $m - M$ esempi in $S \setminus S'$.

Teorema 9.2 | Convergenza del Perceptrone

Per ogni training set $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m)$ linearmente separabile e per ogni iperpiano separatore $\underline{\boldsymbol{u}}$ con margine $\gamma(\underline{\boldsymbol{u}}) \geqslant 1$, l'algoritmo del perceptrone determina un iperpiano separatore $\underline{\boldsymbol{\omega}}$, generalmente diverso da $\underline{\boldsymbol{u}}$ dopo al più:

$$M \leqslant \|\boldsymbol{u}\|^2 \left(\max_{t=1,\ldots,m} \|\boldsymbol{x}_t\|^2 \right)$$

aggiornamenti.

Vale a dire che l'algoritmo termina sempre nel caso in cui il training set sia linearmente separabile.

Dimostrazione 9.1 | Convergenza del Perceptrone

Sia $\underline{\boldsymbol{\omega}}_0 = (0, \ldots, 0)$ l'iperpiano iniziale. Indichiamo con $\underline{\boldsymbol{\omega}}_M$ l'iperpiano dopo M aggiornamenti e indichiamo con $t_M \in \{1, \ldots, m\}$ l'indice dell'esempio $\left(\underline{\boldsymbol{x}}_{t_M}, y_{t_M}\right)$ del training set che ha causato l'M-esimo aggiornamento, ovvero $\underline{\boldsymbol{\omega}}_M = \underline{\boldsymbol{\omega}}_{M-1} + y_{t_M} \underline{\boldsymbol{x}}_{t_M}$. Ricaviamo un maggiorante di M derivando un maggiorante e un minorante a $\left\|\underline{\boldsymbol{\omega}}_M\right\| \|\underline{\boldsymbol{u}}\|$ come segue

$$\begin{aligned} \|\boldsymbol{w}_M\|^2 &= \left\|\boldsymbol{w}_{M-1} + y_{t_M} \boldsymbol{x}_{t_M}\right\|^2 \\ &= \|\boldsymbol{w}_{M-1}\|^2 + \left\|\boldsymbol{x}_{t_M}\right\|^2 + 2 y_{t_M} \boldsymbol{w}_{M-1}^\top \boldsymbol{x}_{t_M} \leqslant \|\boldsymbol{w}_{M-1}\|^2 + \left\|\boldsymbol{x}_{t_M}\right\|^2 \end{aligned}$$

in quanto $y_{t_M} \boldsymbol{w}_{M-1}^\top \boldsymbol{x}_{t_M} \leqslant 0$ dato che è stato eseguito l'aggiornamento $\boldsymbol{w}_M = \boldsymbol{w}_{M-1} + y_{t_M} \boldsymbol{x}_{t_M}$.

Iterando questo ragionamento per M volte, e ricordando che $\underline{\boldsymbol{\omega}}_0 = (0, \ldots, 0)$, otteniamo:

$$\|\boldsymbol{w}_M\|^2 \leqslant \|\boldsymbol{w}_0\|^2 + \sum_{i=1}^{M} \left\|\boldsymbol{x}_{t_i}\right\|^2 \leqslant M \left(\max_{t=1,\ldots,m} \|\boldsymbol{x}_t\|^2 \right)$$

Quindi:

$$\|\boldsymbol{w}_M\| \|\boldsymbol{u}\| \leqslant \|\boldsymbol{u}\| \left(\max_{t=1,\ldots,m} \|\boldsymbol{x}_t\| \right) \sqrt{M}$$

Ora, per il minorante, consideriamo un qualunque iperpiano separatore $\underline{\boldsymbol{u}}$ e sia θ l'algolo fra $\underline{\boldsymbol{u}}$ e $\underline{\boldsymbol{\omega}}_M$. Abbiamo:

$$\begin{aligned} \|\boldsymbol{w}_M\| \|\boldsymbol{u}\| &\geqslant \|\boldsymbol{w}_M\| \|\boldsymbol{u}\| \cos(\theta) \\ &= \boldsymbol{w}_M^\top \boldsymbol{u} \\ &= \left(\boldsymbol{w}_{M-1} + y_{t_M} \boldsymbol{x}_{t_M}\right)^\top \boldsymbol{u} \\ &= \boldsymbol{w}_{M-1}^\top \boldsymbol{u} + y_{t_M} \boldsymbol{u}^\top \boldsymbol{x}_{t_M} \\ &\geqslant \boldsymbol{w}_{M-1}^\top \boldsymbol{u} + 1 \end{aligned}$$

in quanto $1 \leqslant \gamma(\boldsymbol{u}) \leqslant y_t \boldsymbol{u}^\top \boldsymbol{x}_t$ per ogni $t = 1, \ldots, m$. Iterando per M volte otteniamo:

$$\|\boldsymbol{w}_M\| \|\boldsymbol{u}\| \geqslant \boldsymbol{w}_0^\top \boldsymbol{u} + M = M$$

dove abbiamo usato $\underline{\boldsymbol{\omega}}^T{}_0 \underline{\boldsymbol{u}} = 0$ dato che $\underline{\boldsymbol{\omega}}_0 = (0, \ldots, 0)$. Mettendo insieme maggiorante e minorante abbiamo:

$$M \leqslant \|\boldsymbol{u}\| \left(\max_{t=1,\ldots,M} \|\boldsymbol{x}_t\| \right) \sqrt{M}$$

Risolvendo rispetto a M otteniamo:

$$M \leqslant \|\boldsymbol{u}\|^2 \left(\max_{t=1,\ldots,m} \|\boldsymbol{x}_t\|^2 \right)$$

Da cui la tesi.

Dato che $\|\underline{\boldsymbol{u}}\|$ è costante, il perceptrone esegue un numero M di aggiornamenti maggiorato da una costante, e perciò converge ad un iperpiano separatore in un numero di aggiornamenti al più pari a tale costante.

10

Online Gradient Descent

Definizione 10.1 | Protocollo di predizione sequenziale

Dato un algoritmo di apprendimento A per classificazione binaria e data una sequenza arbitraria $(\underline{\boldsymbol{x}}_1, y_1), (\underline{\boldsymbol{x}}_2, y_2), \ldots$ di dati, l'algoritmo genera un modello di partenza $\underline{\boldsymbol{\omega}}_1$ e per $t = 1, 2, \ldots$ procede come segue:

1. Il modello corrente $\underline{\boldsymbol{\omega}}_t$ viene testato sul prossimo esempio $(\underline{\boldsymbol{x}}_t, y_t)$.
2. L'algoritmo A aggiorna il modello $\underline{\boldsymbol{\omega}}_t$ generando un nuovo modello $\underline{\boldsymbol{\omega}}_{t+1}$.

In questo protocollo di predizione sequenziale, l'algoritmo genera una sequenza $\underline{\boldsymbol{\omega}}_1, \underline{\boldsymbol{\omega}}_2, \ldots$ di modelli.

Osservazione 10.2 | Differenze tra apprendimento sequenziale e statistico

Il modello di apprendimento sequenziale si distingue da quello statistico perché nel primo gli algoritmi apprendono in modo incrementale, ovvero tramite ottimizzazioni progressive di un modello predittivo iniziale.

Queste ottimizzazioni sono locali, cioè definite rispetto a singoli esempi della sequenza osservata.

Al contrario, gli algoritmi sviluppati all'interno del modello statistico operano tipicamente risolvendo un problema di ottimizzazione globale, cioè definito sull'intero training set.

Il modello sequenziale è vantaggioso rispetto a quello statistico in tutte quelle situazioni dove non è possibile o non è pratico apprendere tramite ottimizzazione globale.

Definizione 10.2 | Rischio sequenziale

Nel **protocollo di predizione sequenziale** le prestazioni vengono valutate misurando il rischio sequenziale, ovvero la quantità:

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\}$$

che conta, al variare di T la frazione di errori di classificazione compiuta dalla sequenza di modelli sui primi T esempi.

Il rischio sequenziale sostituisce la nozione di rischio statistico.

Definizione 10.3 | Online gradient descent

L'algoritmo della discesa del gradiente o **online gradient descent** lavora con qualunque funzione di perdita convessa e differenziabile $\ell : \mathbb{R}^d \to \mathbb{R}$.

A partire da un punto arbitrario $\underline{\boldsymbol{\omega}}_1$, la discesa del gradiente applica ripetutamente:

$$\underline{\boldsymbol{\omega}}_{t+1} = \underline{\boldsymbol{\omega}}_t - \eta_t \nabla \ell\left(\underline{\boldsymbol{\omega}}_t\right) \quad \text{con } \eta_t > 0$$

Se il punto corrente non è un minimo della funzione, allora $\nabla \ell\left(\underline{\boldsymbol{\omega}}_t\right) > 0$ e pertanto $\underline{\boldsymbol{\omega}}_{t+1}$ si sposterà in direzione del minimo della funzione.

Osservazione 10.1 | A cosa serve il rischio sequenziale?

Come nell'apprendimento statistico siamo interessati a studiare quanto velocemente decresce il rischio all'aumentare della taglia del training set, così nell'apprendimento online siamo interessati a studiare quanto velocemente decresce il rischio sequenziale all'aumentare di T.

Definizione 10.4 | Algoritmo di discesa del gradiente con proiezione

Assumiamo che $\ell_1, \ell_2, \ldots$ sia una sequenza di funzioni di perdita convesse e due volte differenziabili. Ricevono due parametri: una costante η che rappresenta lo step di discesa ed il raggio. Il vettore $\underline{\boldsymbol{\omega}}_1$ viene inizializzato con degli zeri, quindi per $t = 1, 2 \ldots$ si procede:

1. $\boldsymbol{w}'_{t+1} = \boldsymbol{w}_t - \frac{\eta}{\sqrt{t}} \nabla \ell_t(\boldsymbol{w}_t)$
2. $\boldsymbol{w}_{t+1} = \underset{\boldsymbol{w}:\|\boldsymbol{w}\| \leqslant U}{\operatorname{argmin}} \left\|\boldsymbol{w} - \boldsymbol{w}'_{t+1}\right\|$

Obbiettivo 10.1 | Analisi del rischio sequenziale di OGD

Scopo dell'analisi è limitare la differenza fra il rischio sequenziale dell'algoritmo e quello di un qualsiasi modelli $\underline{\boldsymbol{u}}$ tale che $\|\underline{\boldsymbol{u}}\| \leqslant U$. Vogliamo controllare la differenza:

$$\frac{1}{T}\sum_{t=1}^{T}\left(\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u})\right)$$

Sia $\boldsymbol{u}_T^* = \underset{\boldsymbol{u}:\|\boldsymbol{u}\|\leqslant U}{\operatorname{argmin}} \frac{1}{T}\sum_{t=1}^{T}\ell_t(\boldsymbol{u})$ il miglior predittore per i primi T passi: vogliamo dimostrare che:

$$\frac{1}{T}\sum_{t=1}^{T}\ell_t(\boldsymbol{w}_t) - \frac{1}{T}\sum_{t=1}^{T}\ell_t\left(\boldsymbol{u}_T^*\right) = o(1)$$

ovvero che il rischio sequenziale di OGD converge alla perdita media del predittore ottimo $\underline{\boldsymbol{u}}^*{}_T$ per $T \to \infty$.

Lemma 10.1 | Formula di Taylor per funzioni multivariate

Sia $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ una funzione due volte differenziabile. Allora, per ogni $\underline{\boldsymbol{\omega}}, \underline{\boldsymbol{u}} \in \mathbb{R}^d$ vale:

$$f(\boldsymbol{u}) = f(\boldsymbol{w}) + \nabla f(\boldsymbol{w})^\top(\boldsymbol{u}-\boldsymbol{w}) + \frac{1}{2}(\boldsymbol{u}-\boldsymbol{w})^\top\nabla^2 f(\boldsymbol{\xi})(\boldsymbol{u}-\boldsymbol{w})$$

dove $\nabla^2 f(\xi)$ è la matrice Hessiana di f calcolata in un punto ξ sulla retta che congiunge $\underline{\boldsymbol{u}}$ a $\underline{\boldsymbol{\omega}}$.

Analisi 10.1 | Analisi del rischio sequenziale di OGD

Fissiamo un $\underline{\boldsymbol{u}}$ arbitrario con norma limitata da U e notiamo che ad ogni istante t, il teorema di Taylor implica:

$$\begin{aligned}\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u}) &= \nabla\ell_t(\boldsymbol{w}_t)^\top(\boldsymbol{w}_t - \boldsymbol{u}) - \frac{1}{2}(\boldsymbol{u}-\boldsymbol{w}_t)^\top\nabla^2\ell_t(\boldsymbol{\xi})(\boldsymbol{u}-\boldsymbol{w}_t)\\ &\leqslant \nabla\ell_t(\boldsymbol{w}_t)^\top(\boldsymbol{w}_t - \boldsymbol{u})\end{aligned}$$

La disuguaglianza vale perché stiamo assumendo che ℓ_t sia due volte differenziabile e convessa, il che implica che la matrice $\nabla^2\ell_t(\xi)$ sia positiva semidefinita, quindi $z^\top\nabla^2\ell_t(\xi)z \geqslant 0$ per ogni $z \in \mathbb{R}^d$.

Possiamo quindi procedere maggiorando la quantità $\nabla\ell_t(\boldsymbol{w}_t)^\top(\boldsymbol{w}_t - \boldsymbol{u})$:

$$\begin{aligned}\nabla\ell_t(\boldsymbol{w}_t)^\top(\boldsymbol{w}_t - \boldsymbol{u}) &= -\frac{1}{\eta_t}\left(\boldsymbol{w}'_{t+1} - \boldsymbol{w}_t\right)^\top(\boldsymbol{w}_t - \boldsymbol{u})\\ &= \frac{1}{\eta_t}\left(\frac{1}{2}\|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \frac{1}{2}\left\|\boldsymbol{w}'_{t+1} - \boldsymbol{u}\right\|^2 + \frac{1}{2}\left\|\boldsymbol{w}'_{t+1} - \boldsymbol{w}_t\right\|^2\right)\\ &\leqslant \frac{1}{\eta_t}\left(\frac{1}{2}\|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \frac{1}{2}\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 + \frac{1}{2}\left\|\boldsymbol{w}'_{t+1} - \boldsymbol{w}_t\right\|^2\right)\end{aligned}$$

I passaggi sono motivati come segue:

Prima uguaglianza: usa il fatto che $\boldsymbol{w}'_{t+1} - \boldsymbol{w}_t = \eta_t\nabla\ell_t(\boldsymbol{w}_t)$.

Seconda uguaglianza: si tratta di un'identità algebrica.

Disuguaglianza: vale perché $\underline{\boldsymbol{u}}$ appartiene alla sfera di raggio U centrata sull'origine, e quindi proiettando $\underline{\boldsymbol{w}}'_{t+1}$ su questa sfera la distanza con $\underline{\boldsymbol{u}}$ non può aumentare.

Procediamo ad aggiungere e togliere lo stesso termine $\frac{1}{2\eta_{t+1}}\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2$ all'ultimo membro della catena di disuguaglianze e raggruppiamo:

$$\frac{1}{2\eta_t}\|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \frac{1}{2\eta_{t+1}}\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 - \frac{1}{2\eta_t}\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 + \frac{1}{2\eta_{t+1}}\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 + \frac{1}{2\eta_t}\left\|\boldsymbol{w}'_{t+1} - \boldsymbol{w}_t\right\|$$

Sommando su $t = 1, \dots, T$ notiamo che i primi due termini sono una somma telescopica, mentre i secondi due hanno un fattore comune:

$$\sum_{t=1}^{T}(\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u})) \leqslant \frac{1}{2\eta}\|\boldsymbol{w}_1 - \boldsymbol{u}\|^2 - \frac{1}{2\eta_{T+1}}\|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2 + \frac{1}{2}\sum_{t=1}^{T}\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2\left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right) + \frac{1}{2}\sum_{t=1}^{T}\frac{1}{\eta_t}\left\|\boldsymbol{w}'_{t+1} - \boldsymbol{w}_t\right\|^2$$

Semplifichiamo la disequazione utilizzando i seguenti fatti:

$\boldsymbol{w}_1 = \boldsymbol{0}$ Per definizione di OGD.

$\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 \leqslant 4U^2$ Dato che sia $\underline{\boldsymbol{w}}_{t+1}$ che $\underline{\boldsymbol{u}}$ appartengono alla sfera di raggio U.

$\left\|\boldsymbol{w}'_{t+1} - \boldsymbol{w}_t\right\|^2 = \eta_t^2\|\nabla\ell_t(\boldsymbol{w}_t)\|^2$ Per definizione di OGD.

Sostituendo queste relazioni nell'ultima disuguaglianza e scegliendo G tale che $\|\nabla\ell_t(\boldsymbol{w}_t)\| \leqslant G$ per ogni t, otteniamo:

$$\sum_{t=1}^{T}(\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u})) \leqslant \frac{U^2}{2\eta} - \frac{1}{2\eta_{T+1}}\|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2 + 2U^2\sum_{t=1}^{T-1}\left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t}\right) + \frac{1}{2\eta_{T+1}}\|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2 - \frac{1}{2\eta_T}\|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2 + \frac{G^2}{2}\sum_{t=1}^{T}\eta_t$$

Semplifichiamo la somma telescopica, cancellando i termini con segno opposto e maggioriamo omettendo il termine $-\frac{1}{2\eta_T}\|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2$:

$$\begin{aligned}\sum_{t=1}^{T}(\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u})) &\leqslant \frac{U^2}{2\eta} + \frac{2U^2}{\eta_T} - \frac{2U^2}{\eta} + \frac{G^2}{\eta}\sum_{t=1}^{T}\eta_t\\ &\leqslant \frac{2U^2\sqrt{T}}{\eta} + \frac{G^2\eta}{\eta}\sum_{t=1}^{T}\frac{1}{\sqrt{t}}\\ &\leqslant \frac{2U^2\sqrt{T}}{\eta} + G^2\eta\sqrt{T}\end{aligned}$$

dove abbiamo usato la maggiorazione:

$$\sum_{t=1}^{T}\frac{1}{\sqrt{t}} \leqslant 2\sqrt{T}$$

Scegliendo $\eta = \left(\frac{U}{G}\right)\sqrt{2}$ e dividendo tutto per T otteniamo il risultato finale:

$$\frac{1}{T}\sum_{t=1}^{T}\ell_t(\boldsymbol{w}_t) \leqslant \min_{\boldsymbol{u}:\|\boldsymbol{u}\|\leqslant U}\frac{1}{T}\sum_{t=1}^{T}\ell_t(\boldsymbol{u}) + UG\sqrt{\frac{8}{T}}$$

Definizione 10.5 | Hinge Loss

Si tratta di una particolare funzione di loss, che possiamo vedere come regola di aggiornamento del Perceptrone visto come discesa del gradiente:

$$h_t(\boldsymbol{w}) = \left[1 - y_t \boldsymbol{w}^\top \boldsymbol{x}_t\right]_+$$

dove $[z]_+ = \max\{0, z\}$.

La funzione è convessa e maggiore la funzione indicatrice di errore, ovvero $\mathbb{I}\{z \leqslant 0\} \leqslant [1-z]_+$ per ogni $z \in \mathbb{R}$.

Osservazione 10.3 | Gradiente della Hinge Loss

Il gradiente della Hinge Loss è calcolato come:

$$\nabla h_t(\boldsymbol{w}) = \begin{cases} -y_t \boldsymbol{x}_t & \text{se } y_t \boldsymbol{w}^\top \boldsymbol{x}_t \leqslant 1 \\ 0 & \text{otherwise} \end{cases}$$

Analisi 10.2 | Perceptrone come istanza di OGD

Per definire il Perceptrone come istanza di OGD dobbiamo aggiungere la condizione che l'aggiornamento venga fatto solo quando il modello corrente $\underline{\boldsymbol{\omega}}_t$ sbaglia a classificare $(\underline{\boldsymbol{x}}_t, y_t)$:

$$\begin{aligned} \boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \eta_t \nabla h_t(\boldsymbol{w}_t) \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\} \\ &= \boldsymbol{w}_t + \eta_t \boldsymbol{x}_t t \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\} \end{aligned}$$

Dato che $\underline{\boldsymbol{\omega}}_t$ cambia solo quando $y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0$, possiamo applicare l'analisi di OGD ai soli passi t dove $\underline{\boldsymbol{\omega}}_t$ sbaglia.

Scegliendo $\eta_t = \eta$ per ogni t e omettiamo la proiezione di $\underline{\boldsymbol{\omega}}'_{t+1}$ nella sfera di raggio U, cioè ponendo $\underline{\boldsymbol{\omega}}_{t+1} = \underline{\boldsymbol{\omega}}'_{t+1}$. La disuguaglianza dell'OGD diviene, omettendo il termine negativo $-\frac{1}{2\eta_1}\|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2$, ci dà:

$$\sum_{t=1}^{T} (h_t(\boldsymbol{w}_t) - h_t(\boldsymbol{u})) \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\} \leqslant \frac{1}{2\eta}\|\boldsymbol{u}\|^2 + \frac{1}{2}\sum_{t=1}^{T} \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 \left(\frac{1}{\eta} - \frac{1}{\eta}\right) \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\} + \frac{\eta G^2}{2} \sum_{t=1}^{T} \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\}$$

per un qualunque $\underline{\boldsymbol{u}} \in \mathbb{R}^d$.

Quindi, dato che $y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0$ implica $h_t(\boldsymbol{w}_t) \geqslant 1$, e ponendo $X = \max_t \|\boldsymbol{x}_t\| = \max_t \|\nabla h_t(\boldsymbol{w}_t)\|$ così da avere $X = G$, otteniamo:

$$\sum_{t=1}^{T} \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\} \leqslant \sum_{t=1}^{T} h_t(\boldsymbol{u}) + \frac{1}{2\eta}\|\boldsymbol{u}\|^2 + \frac{\eta X^2}{2} \sum_{t=1}^{T} \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\}$$

Sia $M_T = \sum_{t=1}^{T} \mathbb{I}\{y_t \boldsymbol{w}_t^\top \boldsymbol{x}_t \leqslant 0\}$ il numero di errori compiuti dal Perceptrone nei primi T passi.

Scegliendo $\eta = \|\boldsymbol{u}\| / \left(X\sqrt{M_T}\right)$, risolvendo per M_T e maggiorando otteniamo:

$$M_T \leqslant \sum_{t=1}^{T} h_t(\boldsymbol{u}) + (\|\boldsymbol{u}\| X)^2 + \|\boldsymbol{u}\| X \sqrt{\sum_{t=1}^{T} h_t(\boldsymbol{u})}$$

Questo è il maggiorante al numero di errori del Perceptrone nel caso generale, cioè con sequenze non linearmente separabili. Si noti che quando la sequenza è **linearmente separabile**, allora esiste $\underline{\boldsymbol{u}} \in \mathbb{R}^d$ tale che $y_t \boldsymbol{u}^\top \boldsymbol{x}_t \geqslant 1$ per ogni t, il che implica $h_t(\boldsymbol{u}) = 0$ per ogni t. Quindi il maggiorante si riduce a:

$$M_T \leqslant (\|\boldsymbol{u}\| X)^2$$

che corrisponde al teorema di convergenza del Perceptrone.

Definizione 10.6 | Convessità forte

Una funzione differenziabile ℓ è σ-fortemente convessa, per un dato $\sigma > 0$, se:

$$\ell(\boldsymbol{w}) - \ell(\boldsymbol{u}) \leqslant \nabla \ell(\boldsymbol{w})^\top (\boldsymbol{w} - \boldsymbol{u}) - \frac{\sigma}{2}\|\boldsymbol{u} - \boldsymbol{w}\|^2$$

Equivalentemente, possiamo dire che la matrice Hessiana di ℓ ha rango pieno, oppure che ha gli autovalori tutti strettamente maggiori di zero.

Esempio 10.1 | Funzione fortemente convessa

Un semplice esempio di funzione fortemente convessa è $\ell(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2$, infatti:

$$\frac{1}{2}\|\boldsymbol{w}\|^2 - \frac{1}{2}\|\boldsymbol{u}\|^2 = \boldsymbol{w}^\top (\boldsymbol{w} - \boldsymbol{u}) - \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{u}\|^2$$

Quindi la funzione è fortemente convessa per $\sigma = 1$.

Osservazione 10.4 | Algoritmo OGD su funzioni fortemente convesse

L'algoritmo OGD per funzioni fortemente convesse non ha bisogno del passo di proiezione ed è quindi completamente privo di parametri.

Definizione 10.7 | Algoritmo OGD senza proiezione per funzioni fortemente convesse

Il vettore $\underline{\boldsymbol{w}}_1$ viene inizializzato con degli zeri, quindi per $t = 1, 2 \dots$ si procede:

1. $\boldsymbol{w}'_{t+1} = \boldsymbol{w}_t - \eta_t \nabla \ell_t(\boldsymbol{w}_t)$

Analisi 10.3 | Analisi del rischio sequenziale di OGD per funzioni σ-fortemente convesse

Ripetiamo il passo iniziale dell'analisi dell'OGD sfruttando ora l'assunzione che $\ell_1, \ell_2, \dots$ sono tutte funzioni σ-fortemente convesse:

$$\begin{aligned}\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u}) &\leqslant \nabla \ell_t(\boldsymbol{w}_t)^\top (\boldsymbol{w}_t - \boldsymbol{u}) - \frac{\sigma}{2} \|\boldsymbol{u} - \boldsymbol{w}_t\|^2 \\ &= -\frac{1}{\eta_t} (\boldsymbol{w}_{t+1} - \boldsymbol{w}_t)^\top (\boldsymbol{w}_t - \boldsymbol{u}) - \frac{\sigma}{2} \|\boldsymbol{u} - \boldsymbol{w}_t\|^2 \\ &\leqslant \frac{1}{\eta_t} \left(\frac{1}{2} \|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \frac{1}{2} \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 + \frac{1}{2} \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 \right) - \frac{\sigma}{2} \|\boldsymbol{u} - \boldsymbol{w}_t\|^2\end{aligned}$$

Procedendo in modo completamente analogo al caso di OGD con proiezione, ma sfruttando la presenza dei termini aggiuntivi $-\frac{\sigma}{2} \|\boldsymbol{u} - \boldsymbol{w}_t\|^2$ otteniamo:

$$\begin{aligned}\sum_{t=1}^{T} (\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u})) &\leqslant \left(\frac{1}{\eta} - \sigma \right) \frac{1}{2} \|\boldsymbol{w}_1 - \boldsymbol{u}\|^2 - \frac{1}{2\eta_{T+1}} \|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2 \\ &\quad + \frac{1}{2} \sum_{t=1}^{T-1} \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 \left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} - \sigma \right) + \|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2 \frac{1}{2} \left(\frac{1}{\eta_{T+1}} - \frac{1}{\eta_T} \right) + \frac{G^2}{2} \sum_{t=1}^{T} \eta_t\end{aligned}$$

dove, analogamente a prima, $G \geqslant \max_t \|\nabla \ell_t(\boldsymbol{w}_t)\|$.

Omettendo il termine negativo $-\frac{1}{2\eta_T} \|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2$, semplificando il termine $\frac{1}{2\eta_{T+1}} \|\boldsymbol{w}_{T+1} - \boldsymbol{u}\|^2$ che appare con segni opposti e utilizzando la scelta $\eta_t = \frac{1}{\sigma t}$, osserviamo alcune ulteriori sorprendenti semplificazioni che ci conducono a:

$$\sum_{t=1}^{T} (\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{u})) \leqslant \frac{G^2}{2\sigma} \sum_{t=1}^{T} \frac{1}{t} \leqslant \frac{G^2}{2\sigma} \ln(T+1)$$

dove abbiamo usato un semplice maggiorante logaritmo alla somma armonica $1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{T}$.

Questo implica il risultato finale

$$\frac{1}{T} \sum_{t=1}^{T} \ell_t(\boldsymbol{w}_t) \leqslant \min_{\boldsymbol{u} \in \mathbb{R}^d} \frac{1}{T} \sum_{t=1}^{T} \ell_t(\boldsymbol{u}) + \frac{G^2}{2\sigma} \frac{\ln(T+1)}{T}$$

11

Da rischio sequenziale a rischio statistico

Obbiettivo 11.1 | Da rischio sequenziale a rischio statistico

Vogliamo mettere in relazione il rischio sequenziale di un algoritmo online (come OGD) con il rischio di un predittore da esso prodotto, assumendo che i dati sui quali l'algoritmo viene eseguito siano generati da una sorgente statistica. Questo ci permette di creare un ponte fra il modello di apprendimento sequenziale e quello statistico.

Analisi 11.1 | Da rischio sequenziale a rischio statistico

Fissiamo una funzione di perdita ℓ. Dato un predittore lineare $\underline{\boldsymbol{\omega}} \in \mathbb{R}^d$ e un esempio $(\underline{\boldsymbol{x}}_t, y_t) \in \mathbb{R}^d \times \mathbb{R}$, sia $\ell_t(\underline{\boldsymbol{\omega}}) = \ell(\underline{\boldsymbol{\omega}}^T \underline{\boldsymbol{\omega}}, y_t)$ la perdita associata. Assumiamo che la funzione ℓ_t sia convessa. Consideriamo il caso dell'apprendimento statistico, in cui gli esempi $(\underline{\boldsymbol{x}}_t, y_t)$ sono realizzazioni indipendenti di variabili casuali $(\underline{\boldsymbol{X}}_t, Y_t)$ con distribuzione comune $\mathbb{D}$ su $\mathbb{R}^d \times \mathbb{R}$ fissata ma ignota.

Il rischio statistico rispetto alla funzione di perdita ℓ di un predittore lineare $\underline{\boldsymbol{\omega}}$ è definito da:

$$\mathrm{er}_{\mathbb{D}}(\boldsymbol{w}) = \mathbb{E}\left[\ell\left(\boldsymbol{w}^\top \boldsymbol{X}, Y\right)\right]$$

dove l'esempio $(\underline{\boldsymbol{X}}, Y)$ è estratto dalla distribuzione congiunta $\mathbb{D}$ su $\mathbb{R}^d \times \mathbb{R}$.

Consideriamo ora un training set S composto da realizzazioni $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m)$ di variabili casuali $(\underline{\boldsymbol{X}}_1, Y_1), \ldots, (\underline{\boldsymbol{X}}_m, Y_m)$ estratte da $\mathbb{D}$. Eseguiamo un algoritmo online come OGD sul training set S con la sequenza di funzioni di perdita $\ell_1, \ldots, \ell_m$ definite da $\ell_t(\boldsymbol{w}) = \ell\left(\boldsymbol{w}^\top \boldsymbol{x}_t, y_t\right)$.

Per definizione di algoritmo online, otteniamo una sequenza $\underline{\boldsymbol{\omega}}_1, \ldots, \underline{\boldsymbol{\omega}}_m$ di predittori lineari. Vogliamo stabilire un maggiorante sul rischio statistico di un predittore lineare ottenuto in modo naturale dalla sequenza, ovvero il **predittore medio**:

$$\overline{\boldsymbol{w}} = \frac{1}{m} \sum_{t=1}^{m} \boldsymbol{w}_t$$

Dato che ℓ è convessa, la disuguaglianza di Jensen ci dà che:

$$\begin{aligned} \mathrm{er}_{\mathbb{D}}(\overline{\boldsymbol{w}}) &= \mathbb{E}\left[\ell\left(\overline{\boldsymbol{w}}^\top \boldsymbol{X}, Y\right)\right] \\ &\leqslant \mathbb{E}\left[\frac{1}{m} \sum_{t=1}^{m} \ell\left(\boldsymbol{w}_t^\top \boldsymbol{X}, Y\right)\right] \\ &= \frac{1}{m} \sum_{t=1}^{m} \mathrm{er}_{\mathbb{D}}(\boldsymbol{w}_t) \end{aligned}$$

dove l'ultima uguaglianza vale per linearità dell'aspettazione, quindi il rischio del predittore medio è maggiorato dal rischio medio dei predittori della sequenza $\underline{\boldsymbol{\omega}}_1, \ldots, \underline{\boldsymbol{\omega}}_m$.

Il passo cruciale sta ora nel legare il rischio statistico medio con il rischio sequenziale. Questo viene fatto osservando che, sotto l'ipotesi che S sia un campione casuale estratto da $\mathbb{D}$, $\underline{\boldsymbol{\omega}}_t$ è determinato dai primi $t-1$ esempi $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_{t-1}, y_{t-1})$ estratti. Quindi, applicando la definizione di rischio al valore atteso della perdita di $\underline{\boldsymbol{\omega}}_t$ sul t-esimo esempio $(\underline{\boldsymbol{x}}_t, y_t)$, possiamo scrivere:

$$\mathbb{E}\left[\mathrm{er}_{\mathbb{D}}(\boldsymbol{w}_t) - \ell\left(\boldsymbol{w}_t^\top \boldsymbol{X}_t, Y_t\right) \mid (\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_{t-1}, Y_{t-1})\right] = 0$$

La relazione dice che se condizioniamo sui primi $t-1$ esempi, il valore atteso di $\ell_t(\underline{\boldsymbol{\omega}}_t)$ rispetto all'estrazione del t-esimo esempio è semplicemente (per definizione) il rischio di $\underline{\boldsymbol{\omega}}_t$.

Denotiamo con $\mathbb{E}_{t-1}$ il valore atteso condizionato come sopra. Se sommiamo entrambi i membri per $t = 1, \ldots, m$ e dividiamo per m otteniamo:

$$\frac{1}{m} \sum_{t=1}^{m} \mathbb{E}_{t-1}\left[\mathrm{er}_{\mathbb{D}}(\boldsymbol{w}_t) - \ell\left(\boldsymbol{w}_t^\top \boldsymbol{X}_t, Y_t\right)\right] = 0$$

éPer ogni $t = 1, \ldots, m$ sia Z_t la variabile casuale $\mathrm{er}_{\mathbb{D}}(\boldsymbol{w}_t) - \ell\left(\boldsymbol{w}_t^\top \boldsymbol{X}_t, Y_t\right)$. Le variabili casuali $Z_1, \ldots, Z_m$ sono tutte funzioni del medesimo campione S e sono tali che:

$$\frac{1}{m} \sum_{t=1}^{m} \mathbb{E}_{t-1}[Z_t] = 0$$

Assumiamo che $\ell_t \in [0, m]$, allora $|Z - t| \leqslant M$.

Questo tipo di variabili casuali, o più precisamente di processo, viene definito come una sequenza di differenze di Martingale con incrementi limitati da $2M$. Si noti che le Z_t non sono indipendenti. Da un punto di vista statistico, però, queste variabili si comportano come se lo fossero, almeno per certi aspetti.

In particolare, vale una legge dei grandi numeri del tipo:

$$\frac{1}{m} \sum_{t=1}^{m} Z_t \leqslant 2M \sqrt{\frac{1}{2m} \ln \frac{1}{\delta}}$$

con probabilità almeno $1 - \delta$ rispetto all'estrazione di S. Questo significa che:

$$\frac{1}{m} \sum_{t=1}^{m} \mathrm{er}_{\mathbb{D}}(\boldsymbol{w}_t) \leqslant \frac{1}{m} \sum_{t=1}^{m} \ell\left(\boldsymbol{w}_t^\top \boldsymbol{X}_t, Y_t\right) + M \sqrt{\frac{2}{m} \ln \frac{2}{\delta}}$$

con probabilità almeno $1 - \frac{\delta}{2}$ rispetto all'estrazione di S. Ritornando al predittore medio $\overline{\underline{\boldsymbol{\omega}}}$, il risultato che otteniamo può essere sintetizzato come:

$$\mathrm{er}_{\mathbb{D}}(\overline{\boldsymbol{w}}) \leqslant \frac{1}{m} \sum_{t=1}^{m} \ell\left(\boldsymbol{w}_t^\top \boldsymbol{x}_t, y_t\right) + \mathbb{O}\left(\frac{1}{\sqrt{m}}\right)$$

In altre parole, il rischio del predittore medio è limitato in probabilità del rischio sequenziale sul training set.

Analisi 11.2 | Da rischio sequenziale a rischio statistico: caso della regressione con perdita quadratica

Per regressione con perdita quadratica, se facciamo girare OGD con proiezione nell'insieme $\{\boldsymbol{u} \in \mathbb{R}^d : \|\boldsymbol{u}\| \leqslant U\}$ e assumiamo che $\|x_t\| \leqslant X$ e $|y_t| \leqslant UX$ per ogni t, otteniamo che, per ogni realizzazione $(\underline{\boldsymbol{x}}_1, y_1), \dots, (\underline{\boldsymbol{x}}_m, y_m)$ di S,

$$\frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{w}_t^\top \boldsymbol{x}_t, y_t\right) \leqslant \min_{\boldsymbol{u}\in\mathbb{R}^d : \|\boldsymbol{u}\| \leqslant U} \frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{u}^\top \boldsymbol{x}_t, y_t\right) + 8(UX)^2\sqrt{\frac{2}{m}}$$

Sostituendo nel risultato dell'analisi precedente e notando che $M = 4(UX)^2$ per la funzione di perdita quadratica, possiamo quindi scrivere che:

$$\mathrm{er}_{\mathbb{D}}(\overline{\boldsymbol{w}}) \leqslant \min_{\boldsymbol{u}\in\mathbb{R}^d : \|\boldsymbol{u}\| \leqslant U} \frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{u}^\top \boldsymbol{X}_t, Y_t\right) + 12(UX)^2\sqrt{\frac{2}{m}\ln\frac{2}{\delta}}$$

con probabilità almeno $1 - \frac{\delta}{2}$ rispetto all'estrazione di S.

Infine, possiamo notare che, detto:

$$\boldsymbol{u}^* = \operatorname*{argmin}_{\boldsymbol{u}\in\mathbb{R}^d : \|\boldsymbol{u}\| \leqslant U} \mathrm{er}_{\mathbb{D}}(\boldsymbol{u})$$

abbiamo, ovviamente,

$$\min_{\boldsymbol{u}\in\mathbb{R}^d : \|\boldsymbol{u}\| \leqslant U} \frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{u}^\top \boldsymbol{x}_t, y_t\right) \leqslant \frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{x}_t^\top \boldsymbol{u}^*, y_t\right)$$

Dato che, per ogni $t = 1, \dots, m$ si ha $\mathbb{E}\left[\ell\left(\boldsymbol{X}_t^\top \boldsymbol{u}^*, Y_t\right)\right] = \mathrm{er}_{\mathbb{D}}(\boldsymbol{u}^*)$, possiamo applicare il maggiorante di Chernoff-Hoeffding e dedurre che:

$$\frac{1}{m}\sum_{t=1}^{m} \ell\left(\boldsymbol{X}_t^\top \boldsymbol{u}^*, Y_t\right) \leqslant \mathrm{er}_{\mathbb{D}}\left(\boldsymbol{u}^*\right) + 4(UX)^2\sqrt{\frac{1}{2m}\ln\frac{2}{\delta}}$$

Abbiamo così ottenuto il seguente maggiorante esplicito sul rischio del predittore medio:

$$\mathrm{er}_{\mathbb{D}}(\overline{\boldsymbol{w}}) \leqslant \mathrm{er}_{\mathbb{D}}\left(\boldsymbol{u}^*\right) + 14(UX)^2\sqrt{\frac{2}{m}\ln\frac{2}{\delta}}$$

con probabilità almeno $1 - \delta$ rispetto all'estrazione di S.

12

Funzioni kernel

Osservazione 12.1 | Limiti dei predittori lineari

I predittori lineari possono soffrire di un errore di bias molto elevato in quanto il predittore Bayesiano ottimo f^* è spesso molto lontano dall'essere lineare. Questo crea un potenziale rischio di underfitting tutte le volte che utilizziamo un algoritmo di apprendimento che genra predittori lineari.

Osservazione 12.2 | Applicazione a predittori di mappe non lineari

Utilizzando una mappa non lineare per trasformare le istanze di un problema di apprendimento diventa possibile utilizzare predittori lineari per rappresentare predittori non lineari, alleviando il problema dell'underfitting.

Consideriamo infatti una funzione non lineari $\phi : \mathbb{R}^d \longrightarrow \mathbb{V}$ che mappa le istanze $\underline{\boldsymbol{x}} \in \mathbb{R}^d$ in un nuovo spazio $\mathbb{V}$ con un numero molto maggiore, anche infinito, di dimensioni.

Costruendo la funzione ϕ opportunamente, abbiamo che superfici non lineari in $\mathbb{R}^d$ vengono mappate in superfici lineari in $\mathbb{V}$.

Osservazione 12.3 |

La tecnica delle funzioni kernel ci dice come scegliere ϕ in modo tale che esista una funzione $K : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$ tale che:

$$K(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = \phi(\underline{\boldsymbol{x}})^T \phi(\underline{\boldsymbol{x}}') \quad \text{per ogni } \underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}' \in \mathbb{R}^d$$

Definizione 12.1 | Perceptrone con kernel

L'algoritmo perceptrone con kernel inizializza S con l'insieme vuoto e quindi per ogni $t = 1, 2, \ldots$:

1. Leggi il prossimo esempio $(\underline{\boldsymbol{x}}_t, y_t)$
2. Calcola $\hat{y}_t = \operatorname{sgn}\left(\sum_{s \in S} y_s K(\boldsymbol{x}_s, \boldsymbol{x}_t)\right)$
3. Se $\hat{y}_t = y_t$ aggiungi t a S

Definizione 12.2 | Kernel Polinomiale

Un kernel polinomiale è una funzione kernel del tipo:

$$K_n(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}') = \left(1 + \underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}'\right)^n \text{ con } \underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}' \in \mathbb{R}$$

Questo kernel permette di apprendere classificatori di grado n in $\mathbb{R}^d$ rappresentandoli come iperpiani in uno spazio $\mathbb{R}^N$.

Definizione 12.3 | Mappa per il Kernel Polinomiale

Possiamo esplicitare la funzione ϕ_n tale che, nel kernel polinomiale K_n:

$$K_n(\boldsymbol{x}, \boldsymbol{x}') = \phi_n(\boldsymbol{x})^\top \phi_n(\boldsymbol{x}')$$

usando il teorema del binomio di Newton:

$$\left(1 + \boldsymbol{x}^\top \boldsymbol{x}'\right)^n = \sum_{k=0}^{n} \binom{n}{k} \left(\boldsymbol{x}^\top \boldsymbol{x}'\right)^k$$

Osservando che vale:

$$\left(\boldsymbol{x}^\top \boldsymbol{x}'\right)^k = \left(\sum_{i=1}^{d} x_i x_i'\right)^k = \sum_{\boldsymbol{v} \in \{1,\ldots,d\}^k} \left(\prod_{s=1}^{k} x_{v_s} x'_{v_s}\right)$$

E quindi si ottiene:

$$\phi_n(\boldsymbol{x}) = \left(\sqrt{\binom{n}{k}} \prod_{s=1}^{k} x_{v_s}\right)_{k=0,\ldots,n, \boldsymbol{v} \in \{1,\ldots,d\}^k}$$

In altre parole, il mapping $\phi_n : \mathbb{R}^d \longrightarrow \mathbb{R}^N$ associato al kernel polinomiale K_n è un vettore le cui componenti sono indicizzate da tutti i monomi $\prod_{s=1}^{k} x_{v_s}$ di grado al più n pesati dalle radici dei coefficienti binomiali.

Nel caso del kernel polinomiale di grado n, la dimensione N del codominio di ϕ è:

$$\sum_{k=0}^{n} \left|\{1, \ldots, d\}^k\right| = \sum_{k=0}^{n} d^k = \frac{d^{n+1} - 1}{d - 1}$$

ovvero $N = \Theta(d^n)$, esponenziale nel grado del polinomio.

Definizione 12.4 | Kernel Gaussiano

Un utile kernel è il kernel gaussiano:

$$K_\gamma(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\gamma} \left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right) \quad \gamma > 0$$

Definizione 12.5 | Mappa per Kernel Gaussiano

Per identificare la mappa associata ad un kernel gaussiano K_γ procediamo come segue:

$$e^{\left(-\frac{1}{2\gamma}\|\boldsymbol{x}-\boldsymbol{x}'\|^2\right)} = e^{\left(-\frac{1}{2\gamma}\left(\|\boldsymbol{x}\|^2+\|\boldsymbol{x}'\|^2\right)\right)} e^{\left(\frac{1}{\gamma}\left(\boldsymbol{x}^\top \boldsymbol{x}'\right)\right)}{}_S$$

$$= e^{\left(-\frac{\|\boldsymbol{x}\|^2}{2\gamma}\right)} e^{\left(-\frac{\|\boldsymbol{x}'\|^2}{2\gamma}\right)} \sum_{n=0}^{\infty} \frac{1}{n!} \frac{\left(\boldsymbol{x}^\top \boldsymbol{x}'\right)^n}{\gamma^n}$$

dove utilizziamo l'espansione in serie di Taylor $e^x = 1 + x + \frac{x^2}{2!} + \cdots$.

È possibile notare che il kernel Gaussiano è una combinazione lineare di infiniti kernel polinomiali di grado crescente, ciascuno pesato dal reciproco del fattoriale del suo grado.

Il parametro γ svolge il ruolo di fattore di scala per i prodotti $\underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}'$. Infine i fattori $e^{-\|\boldsymbol{x}\|^2/(2\gamma)} e^{-\|\boldsymbol{x}'\|^2/(2\gamma)}$ normalizzano rispetto a $\underline{\boldsymbol{x}}$ e $\underline{\boldsymbol{x}}'$. Infatti, $K_\gamma\left(\underline{\boldsymbol{x}}, \underline{\boldsymbol{x}}\right) = 1$ per ogni $\underline{\boldsymbol{x}} \in \mathbb{R}^d$.

Osservazione 12.4 | Il kernel Gaussiano è universale

Il kernel Gaussiano è universale, per ogni $\gamma > 0$, per ogni funzione continua $f : \mathbb{R}^d \longrightarrow \mathbb{R}$, e per ogni $\varepsilon > 0$, esiste $g \in \mathbb{H}_\gamma$, con:

$$\mathbb{H}_\gamma = \left\{ \sum_{i=1}^{N} \alpha_i K_\gamma\left(\boldsymbol{x}_i, \cdot\right) : \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^d, \alpha_1, \ldots, \alpha_N \in \mathbb{R}, N \in \mathbb{N} \right\}$$

tale che h approssima f con errore al più ε.

TO BE CONTINUED...

13

Support Vector Machines

Definizione 13.1 | Support Vector Machine

La **Support Vector Machine** (SVM) è un algoritmo di apprendimento per classificatori lineari che, fissato un training set linearmente separabile $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m) \in \mathbb{R}^d \times \{-1, +1\}$, genera il classificatore lineare corrispondente all'unica soluzione $\underline{\boldsymbol{\omega}}^* \in \mathbb{R}^d$ del seguente problema di ottimizzazione convessa con vincoli lineari:

$$\begin{aligned} \min_{\boldsymbol{w} \in \mathbb{R}^d} \quad & \tfrac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & y_t \boldsymbol{w}^\top \boldsymbol{x}_t \geqslant 1 \quad t = 1, \ldots, m \end{aligned}$$

Geometricamente, $\underline{\boldsymbol{\omega}}^*$ rappresenta l'iperpiano separatore a margine massimo.

Dimostrazione 13.1 | $\underline{\boldsymbol{\omega}}^*$ rappresenta l'iperpiano separatore a margine massimo

Si noti che $\underline{\boldsymbol{u}}^*$ è identificato dalla soluzione del seguente problema di ottimizzazione:

$$\begin{aligned} & \max_{\gamma > 0} \gamma^2 \\ & \|\boldsymbol{u}\|_T^2 = 1 \\ & y_t \boldsymbol{u}^\top \boldsymbol{x}_t \geqslant \gamma \quad t = 1, \ldots, m \end{aligned}$$

Infatti $\underline{\boldsymbol{u}}$ che massimizza γ è lo stesso $\underline{\boldsymbol{u}}$ che massimizza γ^2 dato che la funzione $f(\gamma) = \gamma^2$ è monotona crescente per $\gamma > 0$. Dividendo per $\gamma > 0$ entrambi i membri di ciascun vincolo $y_t \boldsymbol{u}^\top \boldsymbol{x}_t \geqslant \gamma$ otteniamo il vincolo equivalente $y_t \left(\boldsymbol{u}^\top \boldsymbol{x}_t\right) / \gamma \geqslant 1$. Eseguendo il cambio di variabile $\underline{\boldsymbol{\omega}} = \frac{\underline{\boldsymbol{u}}}{\gamma}$ e notando che $\left\|\underline{\boldsymbol{\omega}}\right\|^2 = \frac{1}{\gamma^2}$ a causa del vincolo $\left\|\underline{\boldsymbol{u}}\right\|^2 = 1$, otteniamo il problema equivalente:

$$\begin{aligned} & \min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{w}\|^2 \\ & \gamma^2 \|\boldsymbol{w}\|^2 = 1 \\ & y_t \boldsymbol{w}^\top \boldsymbol{x}_t \geqslant 1 \quad t = 1, \ldots, m \end{aligned}$$

Si noti che il vincolo $\gamma^2 \|\boldsymbol{w}\|^2 = 1$ è superfluo in quanto, per ogni $\underline{\boldsymbol{\omega}} \in \mathbb{R}^d$ posso trovare $\gamma > 0$ tale che il vincolo è soddisfatto. Quindi lo possiamo eliminare e scalando la funzione obbiettivo per la costante $\frac{1}{2}$ otteniamo:

$$\begin{aligned} & \min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ & y_t \boldsymbol{w}^\top \boldsymbol{x}_t \geqslant 1 \quad t = 1, \ldots, m \end{aligned}$$

Teorema 13.1 | $\underline{\boldsymbol{\omega}}^*$ rappresenta l'iperpiano separatore a margine massimo

Per ogni $(\underline{\boldsymbol{x}}_1, y_1), \ldots, (\underline{\boldsymbol{x}}_m, y_m) \in \mathbb{R}^d \times \{-1, +1\}$ linearmente separabile, il vettore $\underline{\boldsymbol{u}}^*$ che realizza il margine massimo

$$\gamma^* = \max_{\boldsymbol{u} : \|\boldsymbol{u}\| = 1} \min_{t = 1, \ldots, m} y_t \boldsymbol{u}^\top \boldsymbol{x}_t$$

soddisfa $\boldsymbol{u}^* = \gamma^* \boldsymbol{w}^*$, dove $\underline{\boldsymbol{\omega}}^*$ è soluzione del problema di ottimizzazione convessa di una SVM.

Lemma 13.1 | Condizione di ottimalità di Fritz John

Si consideri il problema:

$$\begin{aligned} \min_{\boldsymbol{w} \in \mathbb{R}^d} \quad & f(\boldsymbol{w}) \\ \text{s.t.} \quad & g_t(\boldsymbol{w}) \leqslant 0 \quad t = 1, \ldots, m \end{aligned}$$

dove le funzioni $f, g_1, \ldots, g_m$ sono differenziabili. Se $\underline{\boldsymbol{\omega}}_0$ è una soluzione ottima, allora esiste un vettore $\boldsymbol{\alpha} \in \mathbb{R}^m$ tale che:

$$\nabla f(\boldsymbol{w}_0) + \sum_{t \in I} \alpha_t \nabla g_t(\boldsymbol{w}_0) = \boldsymbol{0}$$

dove $I = \{1 \leqslant t \leqslant m : g_t(\boldsymbol{w}_0) = 0\}$

Osservazione 13.1 | Forma della soluzione ottima e vettori di supporto

Applicando la condizione di Fritz John alla funzione obbiettivo SVM, con $f(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2$ e $g_t(\underline{\boldsymbol{\omega}}) = 1 - y_t \boldsymbol{w}^\top \boldsymbol{x}_t$ otteniamo che:

$$\boldsymbol{w}^* - \sum_{t \in I} \alpha_t y_t \boldsymbol{x}_t = \boldsymbol{0}$$

Quindi la soluzione ha forma:

$$\boldsymbol{w}^* = \sum_{t \in I} \alpha_t y_t \boldsymbol{x}_t$$

dove I denota l'insieme degli esempi di training $(\boldsymbol{x}_t, y_t)$ tali che $y_t (\boldsymbol{w}^*)^\top \boldsymbol{x}_t = 1$. Questi $\underline{\boldsymbol{x}}_t$ sono i cosiddetti **vettori di supporto**, ovvero quelle istanze di training sulle quali $\underline{\boldsymbol{\omega}}^*$ ha margine esattamente pari a 1.

Se andassi a rimuovere dal training set tutti gli esempi tranne quelli di supporto la soluzione SVM non cambierebbe.

Definizione 13.2 | SVM su training set non linearmente separabile

Nel caso

14

Reti neurali e deep learning

Definizione 14.1 | Sigmoide

La funzione non lineare $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ è chiamata sigmoide ed è utilizzata come funzione di attivazione:

$$\sigma(z) = \frac{1 - e^{-z}}{1 + e^{-z}} \in [-1, +1]$$

Definizione 14.2 | Rete neurale feedforward

Una rete neurale feedforward è la tipologia di rete neurale più semplice, ottenuta combinando predittori della forma $g(\boldsymbol{x}) = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x}\right)$.

Una rete neurale feedforward calcola una funzione $\underline{\boldsymbol{f}} : \mathbb{R}^d \longrightarrow \mathbb{R}^n$ rappresenta come un grafo diretto aciclico $G = (V, E)$.

Ad ogni arco $(i, j) \in E$ è associato un parametro, chiamato peso, $w_{ij} \in \mathbb{R}$. Indichiamo con W la matrice $|V| \times |V|$ dei pesi, dove $w_{ij} = 0$ se $(i, j) \notin E$.

Il grafo G, la matrice dei pesi W e la funzione di attivazione σ definiscono la funzione $\underline{\boldsymbol{f}} = \underline{\boldsymbol{f}}_{G,W,\sigma}$ calcolata dalla rete.

In generale possiamo partizionare i nodi in tre sotto-insiemi: $V = V_{\text{in}} \cup V_{\text{hid}} \cup V_{\text{out}}$, dove:

V_{in} con $|V_{\text{in}}| = d$ sono i nodi dello strato di input, cioè i nodi che non ricevono nessun arco in ingresso.

V_{out} con $|V_{\text{out}}| = n$ sono i nodi nello strato di output, ovvero i nodi che non emettono nessun arco in uscita.

V_{hid} sono i nodi nello strato nascosto.

Osservazione 14.1 | Come viene calcolata la funzione della rete?

Per calcolare la funzione della rete , ad ogni nodo $i \in V$ è associato un valore $v_i \in \mathbb{R}$ nel modo spiegato in seguito:

Dato $j \in V \setminus V_{\text{in}}$ indichiamo con:

1. $\underline{\boldsymbol{w}}(j)$ il vettore le cui componenti sono i pesi w_{ij} per ogni $(i, j) \in E$
2. $\underline{\boldsymbol{v}}(j)$ il vettore le cui componenti sono i valori v_i per ogni $(i, j) \in E$

Il valore $\boldsymbol{f}(\boldsymbol{x}) = (f_1, \ldots, f_n)$ viene calcolato come segue:

1. Ogni nodo $i \in V_{\text{in}}$ assume valore $v_i = x_i$
2. Ogni nodo $j \in V \setminus V_{\text{in}}$ assume valore $v_j = \sigma\left(\boldsymbol{w}(j)^\top \boldsymbol{v}(j)\right)$
3. $f_k = v_k$, dove k è il k-esimo nodo di output.

Definizione 14.3 | $\mathbb{F}_{G,\sigma}$

Fissati d, la dimensionalità dell'input, e n, la dimensionalità la classe $\mathbb{F}_{G,\sigma}$ dei predittori $\boldsymbol{f} : \mathbb{R}^d \to \mathbb{R}^n$ tali che $\boldsymbol{f} = \boldsymbol{f}_{G,W,\sigma}$ per una qualche matrice dei pesi W, ovvero predittori rappresentabili da una rete neurale feedforward con grafo G, matrice dei pesi W e funzione di attivazione σ.

Teorema 14.1 | Rete come funzione universale

Per ogni d, esiste una rete neurale $G = (V, E)$ con $d + 1$ nodi di input, uno strato di nodi nascosti e un nodo di output, tale che $\mathbb{F}_{G,\text{sgn}}$ contiene tutte le funzioni della forma $f : \{-1, +1\}^d \longrightarrow \{-1, +1\}$.

Dimostrazione 14.1 | Rete come funzione universale

Fissiamo una rete con $2^d + 1$ nodi nascosti e connettività completa dello strato di input con lo strato nascosto e dello strato nascosto con lo strato di output.

Numeriamo i nodi come segue: 0 è il nodo di output, $1, \ldots, 2^d + 1$ sono i $2^d + 1$ nodi nascosti e i seguenti sono i $d + 1$ nodi di input.

Dato una qualunque funzione $f : \{-1, +1\}^d \longrightarrow \{-1, +1\}$ dimostriamo che esiste una matrice dei pesi W tale che $f_{G,W,\text{sgn}} = f$. Siano $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, con $N \leqslant 2^d$, tutte e sole le istanze $\boldsymbol{x}_i \in \{-1, +1\}^d$ tali che $f\left(\underline{\boldsymbol{x}}_i\right) = 1$.

Si osservi che per ogni $\boldsymbol{x} \in \{-1, +1\}^d$ e per ogni $i = 1, \ldots, N$:

$$\boldsymbol{x}^\top \boldsymbol{x}_i \leqslant d - 2 \text{ se } \boldsymbol{x} \neq \boldsymbol{x}_i$$
$$\boldsymbol{x}^\top \boldsymbol{x}_i = d \text{ altrimenti.}$$

Quindi $g_i(\boldsymbol{x}) = \text{sgn}\left(\boldsymbol{x}^\top \boldsymbol{x}_i - d + 1\right) = 1$ se e solo se $\underline{\boldsymbol{x}} = \underline{\boldsymbol{x}}_i$. Ogni funzione g_i può essere implementata dal nodo nascosto i nel modo seguente:

Si trasforma ogni istanza $\boldsymbol{x} = (x_1, \ldots, x_d)$ nell'istanza $\tilde{\boldsymbol{x}} = (\boldsymbol{x}, 1) = (x_1, \ldots, x_d, 1)$ e si assegnano i pesi $\boldsymbol{w}(i) = (\boldsymbol{x}_i, -d + 1)$. Questo implica che: $\text{sgn}\left(\boldsymbol{w}(i)^\top \tilde{\boldsymbol{x}}\right) = g_i(\boldsymbol{x})$ Poi assegnamo $\boldsymbol{w}(N + 1) = (0, \ldots, 0, 1)$ in modo che $\text{sgn}\left(\boldsymbol{w}(i)^\top \tilde{\boldsymbol{x}}\right) = 1$ per ogni $\underline{\boldsymbol{x}}$.

I rimanenti $\underline{\boldsymbol{w}}(i)$ per $i = N + 2, \ldots, 2^d + 1$ possono essere assegnati arbitrariamente.

Infine f può essere rappresentata come $f(\boldsymbol{x}) = g_1(\boldsymbol{x}) \vee \cdots \vee g_N(\boldsymbol{x})$ il che corrisponde ad assegnare i pesi:

$$\boldsymbol{w}(0) = \left(\underbrace{1, \ldots, 1}_{N \text{ volte}}, N - 1, \underbrace{0, \ldots, 0}_{2^d - N \text{ volte}} \right)$$

dove 0 è il nodo di output.

Osservazione 14.2 |

La dimostrazione del teorema precedente utilizza una rete con un singolo strato nascosto di dimensione almeno pari al numero massimo di istanze positive per le funzioni f da apprendere. Questo numero è tipicamente esponenziale nella dimensione d.

Teorema 14.2 | Nodi di una rete universale

Per ogni intero d sia $s(d)$ il più piccolo intero tale che esiste $G=(V,E)$ con $|V|=s(d)$ per cui $\mathbb{F}_{G,\text{sgn}}$ contiene tutte le funzioni della forma $f:\{-1,+1\}^d \longrightarrow \{-1,+1\}$. Allora

$$|V| = \Omega\left(2^{d/3}\right)$$

Un risultato analogo vale quando sgn è rimpiazzata dalla funzione sigmoidale.

Il teorema stabilisce che se vogliamo una rete in grado di apprendere tutte le funzioni da $\{-1,+1\}^d$ a $\{-1,+1\}$, allora tale rete deve avere un numero complessivo di nodi esponenziale in d.

Definizione 14.4 | Funzione Lipshitziana

Una funzione $f:[-1,+1]^d \to [-1,+1]$ è Lipshitziana se f è tale che esiste $L<\infty$ che soddisfa:

$$\left|f(\boldsymbol{x})-f\left(\boldsymbol{x}'\right)\right| \leqslant L\left\|\boldsymbol{x}-\boldsymbol{x}'\right\|$$

per ogni $\boldsymbol{x},\boldsymbol{x}' \in [-1,+1]^d$

Definizione 14.5 | Regola di derivazione delle funzioni composte

$$\frac{df(g(x))}{dx} = \frac{df(g)}{dg}\frac{dg(x)}{dx}$$

Osservazione 14.3 | Approssimazione di funzioni Lipshitziana

Le reti feedforward con un singolo strato nascosto non solo implementano qualsiasi funzione booleana, ma sono anche in grado di approssimare qualsiasi funzione $f:[-1,+1]^d \to [-1,+1]$ che sia Lipshitziana quando la funzione di attivazione utilizzata è la sigmoidale.

Ovvero per ogni $\varepsilon>0$ e per ogni $f:[-1,+1]^d \to [-1,+1]$ Lipshitziana esistono G (con un singolo strato nascosto) e W tali che $\left|f_{G,W,\sigma}(\boldsymbol{x})-f(\boldsymbol{x})\right| \leqslant \varepsilon$ per ogni $\boldsymbol{x} \in [-1,+1]^d$.

Come nel caso delle funzioni booleane, il numero di nodi nello strato nascosto necessario per rappresentare la funzione f con un grado di approssimazione fissato $\varepsilon>0$ può essere esponenziale in d.

Osservazione 14.4 | Reti profonde vanno meglio di reti larghe

Il teorema appena presentato ci dice che se vogliamo imparare un numero elevato di funzioni con una rete G, allora G deve avere un numero esponenziale di nodi comunque esso venga scelto.

L'esperienza sembra però indicare che una rete con un grande numero di strati nascosti ciascuno contenente relativamente pochi nodi, una **deep neural network**, è in grado di rappresentare più funzioni rispetto ad una rete con pochi strati nascosti ciascuno contenente tanti nodi.

In altre parole, aggiungere a G uno strato nascosto con pochi nodi aumenta $\mathbb{F}_{G,\sigma}$ tanto quanto aumenterebbe aggiungendo un gran numero di nodi ad uno strato esistente.

I dettagli precisi di questo fenomeno sono ancora un tema attivo di ricerca.

Definizione 14.6 | Algoritmo di discesa del gradiente stocastico

Le reti feedforward vengono tipicamente addestrate usando l'algoritmo di discesa del gradiente stocastico:

$$w_{i,j} \leftarrow w_{i,j} - \eta_t \frac{\partial \ell_{Z_t}(W)}{\partial w_{i,j}}$$

dove Z_t è l'indice di un esempio estratto a caso dal training set e $\ell_t(W)=\ell\left(f_{G,W,\sigma}(\boldsymbol{x}_t),y_t\right)$ è una funzione di perdita tale che $\ell\left(\cdot,y\right)$ è convessa.

Definizione 14.7 | Mini-batched stochastic gradient

Per accelerare la convergenza, viene usata spesso la versione **mini-batched** del gradiente stocastico:

$$w_{i,j} \leftarrow w_{i,j} - \eta_t \frac{1}{|S_t|}\sum_{s\in S_t}\frac{\partial \ell_s(W)}{\partial w_{i,j}}$$

dove S_t è un sottoinsieme di cardinalità fissata estratto a caso dal training set.

Osservazione 14.5 | $\ell_t(W)$ non è convessa

È importante osservare che $\ell_t(W)$ è una funzione non convessa dei pesi W anche quando $\ell\left(\cdot,y\right)$ è convessa per ogni y. Questo ha due conseguenze importanti:

1. Il problema di determinare i pesi W che minimizzano, anche in modo approssimato, l'errore sul training set è considerato computazionalmente intrattabile.
2. Il metodo della discesa del gradiente converge tipicamente a un minimo locale del training error.

Definizione 14.8 | Algoritmo di retropropagazione del gradiente

L'applicazione della discesa del gradiente per l'addestramento di una rete neuronale feedforward prende il nome di algoritmo di retropropagazione del gradiente.

Per comodità, supponiamo che 0 sia l'indice del nodo di output. Quindi $f_{G,W,\sigma}(\boldsymbol{x}_t) = v_0 = \sigma(s_0)$, dove $s_0 = \boldsymbol{w}(0)^\top \boldsymbol{v}(0)$ Per un qualunque i nel primo strato nascosto, ovvero tale che $(i,0) \in E$, applicando due volte la regola di derivazione delle funzioni composte:

$$\begin{aligned}\frac{\partial \ell_t(W)}{\partial w_{i,0}} &= \frac{\partial \ell(v_0, y_t)}{\partial w_{i,0}} \\ &= \frac{d\ell(v_0, y_t)}{dv_0}\frac{dv_0}{ds_0}\frac{\partial s_0}{\partial w_{i,0}} \\ &= \ell'(v_0)\frac{d\sigma(s_0)}{ds_0}\frac{\partial s_0}{\partial w_{i,0}}\end{aligned}$$

La derivata $\ell'(v_0) = \frac{d\ell(v_0,y_t)}{dv_0}$ è calcolabile direttamente a partire dalla definizione della funzione di perdita.

In modo simile, la derivata $\frac{d\sigma(s_0)}{ds_0} = \sigma'(s_0)$ è calcolabile direttamente a partire dalla funzione di attivazione.

Infine, ricordando che $s_0 = \boldsymbol{w}(0)^\top \boldsymbol{v}(0)$, abbiamo:

$$\frac{\partial s_0}{\partial w_{i,0}} = v_i = \sigma\left(\boldsymbol{w}(i)^\top \boldsymbol{v}(i)\right)$$

Possiamo quindi scrivere:

$$\frac{\partial \ell_t(W)}{\partial w_{i,0}} = \ell'(v_0)\,\sigma'(s_0)\,v_i$$

Calcoliamo ora $\frac{\partial \ell_t(W)}{\partial w_{i,j}}$ per i nodi i nel secondo strato nascosto. Si noti che:

$$\begin{aligned}v_0 &= \sigma\left(\boldsymbol{w}(0)^\top \boldsymbol{v}(0)\right) \\ &= \sigma\left(\sum_{j:(j,0)\in E} w_{j,0}\,\sigma\left(\boldsymbol{w}(j)^\top \boldsymbol{v}(j)\right)\right) \\ &= \sigma\left(\sum_{j:(j,0)\in E} w_{j,0}\,\sigma\left(\sum_{i:(i,j)\in E} w_{i,j} v_i\right)\right)\end{aligned}$$

Quindi, per ogni nodo i nel secondo strato nascosto, denotando $s_i = \boldsymbol{w}(i)^\top \boldsymbol{v}(i)$ possiamo scrivere:

$$\begin{aligned}\frac{\partial \ell_t(W)}{\partial w_{i,j}} &= \underbrace{\frac{d\ell(v_0, y_t)}{ds_0}}_{\ell'(v_0)\sigma'(s_0)}\underbrace{\frac{\partial s_0}{\partial v_j}}_{w_{j,0}}\underbrace{\frac{dv_j}{ds_j}}_{\sigma'(s_j)}\underbrace{\frac{\partial s_j}{\partial w_{i,j}}}_{v_i} \\ &= \ell'(v_0)\,\sigma'(s_0)\,w_{j,0}\sigma'(s_j)\,v_i\end{aligned}$$

A partire dai nodi i del terzo strato nascosto, il calcolo di $\frac{\partial \ell_t(W)}{\partial s_i} = \frac{\partial \ell(v_0,y_t)}{\partial s_i}$ è un po' più complesso in quanto v_0 dipende da s_i attraverso tutti i nodi del secondo strato nascosto che sono connessi ad i.

Ovvero $v_0 = g(s_{j_1}, \ldots, s_{j_r})$, dove g è la funzione che calcola l'output v_0 a partire dai valori s_{jk} dei nodi $j_1, \ldots, j_r$ del secondo strato nascosto connessi a i e $s_{j_k} = h_k(s_i)$ dove h_k è la funzione che descrive come s_{jk} dipende da s_i. La regola per la derivazione delle funzioni multidimensionali composte ci dice che se $v_0 = g(h_1(s_i), \ldots, h_r(s_i))$ allora:

$$\frac{dg}{ds_i} = \sum_{j=1}^{r} \frac{\partial g}{\partial h_j}\frac{dh_j}{ds_i}$$

Nel nostro caso abbiamo quindi:

$$\begin{aligned}\frac{\partial \ell(v_0, y_t)}{\partial s_i} &= \sum_{j:(i,j)\in E} \frac{\partial \ell(v_0, y_t)}{\partial s_j}\frac{\partial s_j}{\partial s_i} \\ &= \sum_{j:(i,j)\in E} \frac{\partial \ell(v_0, y_t)}{\partial s_j}\frac{\partial s_j}{\partial v_i}\frac{dv_i}{s_i} \\ &= \sum_{j:(i,j)\in E} \frac{\partial \ell(v_0, y_t)}{\partial s_j} w_{i,j}\sigma'(s_i)\end{aligned}$$

Quindi, introducendo la definizione ricorsiva:

$$\delta_i = \frac{\partial \ell_t(W)}{\partial s_i} = \begin{cases} \ell'(v_0)\,\sigma'(s_0) & \text{se } i = 0 \\ \sigma'(s_i)\sum_{j:(i,j)\in E} \delta_j w_{i,j} & \text{altrimenti} \end{cases}$$

possiamo infine scrivere la derivata parziale per un qualunque $(i,j) \in E$ come:

$$\frac{\partial \ell_t(W)}{\partial w_{i,j}} = \frac{\partial \ell_t(W)}{\partial s_j}\frac{\partial s_j}{\partial w_{i,j}} = \delta_j v_i$$