Fondamenti di Machine Learning

Laurea Triennale in Ingegneria delle Comunicazioni

5: Classificazione (regressione logistica e Naive Bayes)

Lecturer: S. Scardapane

SAPIENZA
UNIVERSITÀ DI ROMA

Modelli per la classificazione

Formulazione del problema

Passiamo ora ad affrontare più in dettaglio il problema della **classificazione**, nel quale la predizione $\hat{y}$ del modello può assumere solo un insieme finito di valori $\{1, \ldots, C\}$, corrispondenti alle *classi* da predire (mutuamente esclusive). Ad esempio:

1. Classificazione di immagini mediche (es., diagnostica).
2. Predizione in un modello di generazione del testo (es., GPT).
3. Classificazione di un cliente in un determinato segmento di mercato (es., possibile consumatore ricorrente).
4. Predizione di quali clienti potrebbero lasciare l'azienda (**churn prediction**).

Abbiamo già visto come il k-NN possa essere usato per la classificazione (sostituendo la media con un voto di maggioranza), con tutti i limiti già menzionati. In questa lezione introduciamo invece due altri approcci:

- Una estensione del least-squares a problemi di classificazione, la **regressione logistica** (o **logistic regression**).
- Come confronto, un semplice metodo probabilistico basato su una assunzione di indipendenza tra feature, il **Naive Bayes**.

Modelli per la classificazione

Least-squares regression

Supponiamo di voler direttamente estendere il least-squares per predire la classe più probabile (**least-squares classification**). In questo caso, denotiamo come $\mathbf{y}$ il one-hot encoding della classe y (es., se $y = 2$ e $c = 3$, $\mathbf{y} = [0 \;\; 1 \;\; 0]^\top$).

Possiamo costruire un modello lineare con c output come lo stack di c diversi modelli unidimensionali:

$$f(\mathbf{x}) = \begin{bmatrix} \mathbf{w}_1^\top \\ \vdots \\ \mathbf{w}_c^\top \end{bmatrix} \mathbf{x} + \begin{bmatrix} b_1 \\ \vdots \\ b_c \end{bmatrix} = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{1}$$

dove $f(\mathbf{x}) \in \mathbb{R}^c$, $\mathbf{W} \in \mathbb{R}^{c \times d}$ e $\mathbf{b} \in \mathbb{R}^c$.

Possiamo allenare il modello minimizzando l'errore quadratico medio (in questo caso parliamo di **Brier score**); in fase di inferenza, la classe predetta diventa:

$$\text{Classe di } \mathbf{x} = \underset{k=1,\dots,c}{\arg\max}\left\{f_k(\mathbf{x})\right\}.$$

Questo approccio è semplice da implementare ma ha diverse problematiche:

1. Abbiamo perso una interpretazione probabilistica (i valori di f possono essere negativi o maggiori di 1).
2. Per $c > 2$, è possibile che una o più classi vengano mascherate.

Esempio di least squares classification

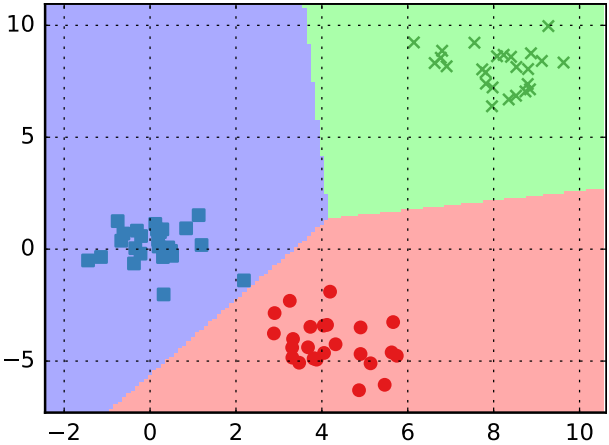


Figure 1: Con 3 classi perfettamente separabili, le **decision boundaries** trovate sono corrette.

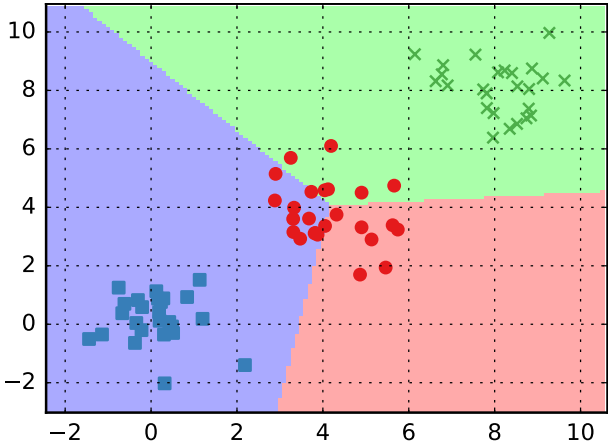


Figure 2: In questo caso, nonostante le tre classi siano perfettamente separabili, una delle tre viene completamente mascherata dalle altre due.

Modelli per la classificazione

Regressione logistica

Riepilogando, possiamo rappresentare l'output in due modi: ad esempio, con tre classi:

- $y = 1$, classe 1 (es., gatto).
- $y = 2$, classe 2 (es., cane).
- $y = 3$, classe 3 (es., altro).

Oppure, con un one-hot encoding delle stesse classi:

$$\text{Gatto: } \mathbf{y} = [1, 0, 0] \quad \text{Cane: } \mathbf{y} = [0, 1, 0] \quad \text{Altro: } \mathbf{y} = [0, 0, 1]\,. \tag{2}$$

Vorremmo un modello che permetta di predire una *distribuzione di probabilità* categorica sulle varie classi:

$$p(\mathbf{y}|f(\mathbf{x})) = \prod_i f_i(\mathbf{x})^{y_i} \tag{3}$$

Per fare questo, l'output del modello deve rispettare queste due proprietà:

$$f_i(\mathbf{x}) \geq 0 \text{ (le probabilità sono positive)} \tag{4}$$

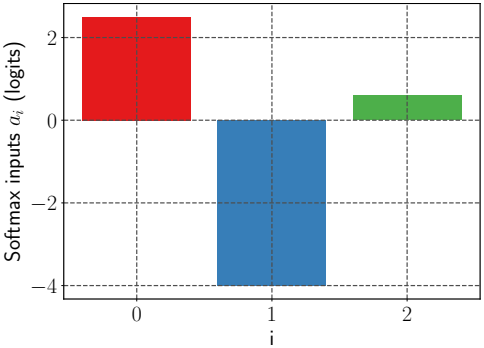
$$\sum_i f_i(\mathbf{x}) = 1 \text{ (le probabilità sommano ad 1)} \ . \tag{5}$$

La funzione **softmax** è definita come:

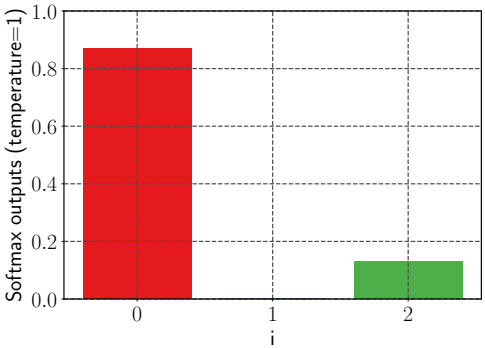
$$\text{softmax}_i(\mathbf{a}) = \frac{\exp(a_i)}{\sum_j \exp(a_j)} \tag{6}$$

Un qualsiasi vettore softmax($\mathbf{x}$) rispetta le due proprietà viste prima: positività (grazie al numeratore) e somma ad 1 (grazie al denominatore).

Visualizzazione della softmax



(a) Softmax inputs (logits)



(b) Softmax outputs

Un modello lineare per la classificazione si ottiene combinando un classico modello lineare con la softmax:

$$\underset{(c)}{f(\mathbf{x})} = \text{softmax}(\underset{(c,d)}{\mathbf{W}} \underset{(d)}{\mathbf{x}}) \tag{7}$$

I valori $\mathbf{Wx}$ vengono chiamati i **logits** del modello. Possiamo anche aggiungere esplicitamente i bias $\underset{(c)}{\mathbf{b}}$, ottenendo:

$$f(\mathbf{x}) = \text{softmax}(\mathbf{Wx} + \mathbf{b})\,. \tag{8}$$

Per derivare una loss function applichiamo il principio di maximum likelihood, data la classe desiderata (one-hot encoded) $\mathbf{y}$ e le predizioni del modello $\hat{\mathbf{y}}$:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\log(p(\mathbf{y} \mid \hat{\mathbf{y}})) = -\sum_{i=1}^{c} y_i \log(\hat{y}_i) \tag{9}$$

Questa viene detta la **cross-entropy** loss.

La cross-entropy loss ha una semplice interpretazione se ricordiamo che $\mathbf{y}$ è un vettore one-hot encoded. Denotando con t l'indice della classe da predire, $t = \arg\max \mathbf{y}$, possiamo semplificare la CE notando che un solo termine è diverso da 0:

$$\mathrm{CE}(\mathbf{y}, \hat{\mathbf{y}}) = -\log(\hat{y}_t)\,. \tag{10}$$

Quindi, minimizzare la CE porta semplicemente a massimizzare la probabilità predetta corrispondente alla classe desiderata.

Combinando il tutto, la **regressione logistica** è un modello lineare $f(\mathbf{x}) = \text{softmax}(\mathbf{W}\mathbf{x})$ allenato minimizzando la cross-entropy:

$$\text{LR}(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^{n} \text{CE}\left(\mathbf{y}_i, f(\mathbf{x}_i)\right) . \tag{11}$$

Non è più possibile risolvere questo problema in forma chiusa, ed è necessario ricorrere a metodi iterativi (es., discesa al gradiente). Un modello di questo tipo ha dc parametri (o $dc + c$ considerando il bias).

Modelli per la classificazione

Classificazione binaria

La **binary classification**, $c = 2$, permette una serie di semplificazioni. In effetti, possiamo predire un unico valore $f(\mathbf{x}) \in [0, 1]$:

$$f(\mathbf{x}) \quad \text{probabilità della classe 1} \ , \tag{12}$$

$$1 - f(\mathbf{x}) \quad \text{probabilità della classe 2} \ . \tag{13}$$

In questo caso la softmax diventa la funzione **sigmoide**:

La sigmoide (o funzione sigmoidea) $\sigma(s) \in [0, 1]$ è definita come:

$$\sigma(s) = \frac{1}{1 + \exp(-s)} \ . \tag{14}$$

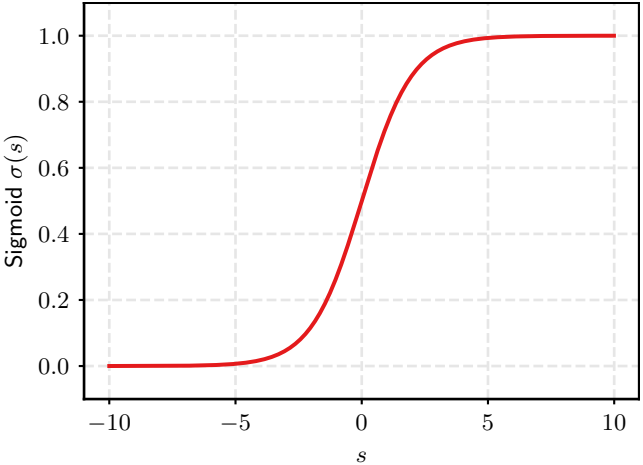


Figure 3: Visualizzazione della funzione sigmoidea. Si noti come 0 e 1 vengono raggiunti solo asintoticamente.

Combinando i due otteniamo un modello di regressione logistica valido solo nel caso binario:

$$\text{BIN-LR}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\Big[\underbrace{-y_i \log\big(\sigma(\mathbf{w}^\top \mathbf{x})\big)}_{\text{Classe 1}} \underbrace{-(1-y_i)\log\big(1-\sigma(\mathbf{w}^\top \mathbf{x})\big)}_{\text{Classe 2}}\Big] \tag{15}$$

In questo caso, la classe predetta dal modello diventa:

$$\text{classe} = \begin{cases} 1 & \text{se } \sigma(\mathbf{w}^\top \mathbf{x}) > 0.5\,, \\ 0 & \text{altrimenti}\ . \end{cases} \tag{16}$$

Consideriamo il gradiente della sigmoide:

$$\sigma'(s) = \sigma(s)(1 - \sigma(s))\,. \tag{17}$$

Inserendolo nel calcolo del gradiente della regressione logistica binaria otteniamo:

$$\nabla \text{BIN-LR}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(\sigma(\mathbf{w}^\top \mathbf{x}_i) - y_i)\mathbf{x}_i\,. \tag{18}$$

Si noti la similarità con il caso della regressione.

In effetti, possiamo riscrivere il modello come:

$$\underbrace{\mathbf{w}^\top \mathbf{x} + b}_{\text{logits}} = \underbrace{\log\left(\frac{y}{1-y}\right)}_{\sigma^{-1}(y)} \tag{19}$$

Questo chiarisce perché ci riferiamo alla regressione logistica come ad un modello lineare: è equivalente ad un modello lineare allenato su una trasformazione non-lineare dell'output. Questa classe di modelli viene detta generalized linear models.

Modelli per la classificazione

Calibrazione

Un malinteso comune nella classificazione è che $f(x)_i$ possa essere immediatamente interpretato come *la probabilità che il pattern x appartenga alla classe i*.

Tuttavia, questo è vero solo quando il modello addestrato soddisfa:

$$p(y = i \mid x) = f_i(\mathbf{x}) \,. \tag{20}$$

In questo caso diciamo che il modello è ben **calibrato**, ma questo deve essere verificato manualmente.

Per misurare la calibrazione di un modello, manteniamo un set di validazione separato e dividiamo l'intervallo $[0, 1]$ in m intervalli equispaziati (ognuno di dimensione $1/m$). Definiamo:

- B_m il numero di campioni dal set di validazione, la cui confidenza prevista rientra nel bin m.
- p_m la confidenza media della rete per quel bin.
- a_m l'accuratezza media della rete per questi elementi.

L'**errore di calibrazione atteso** (ECE) è dato da:

$$\mathrm{ECE} = \sum_m \frac{B_m}{n} |a_m - p_m| \,. \tag{21}$$

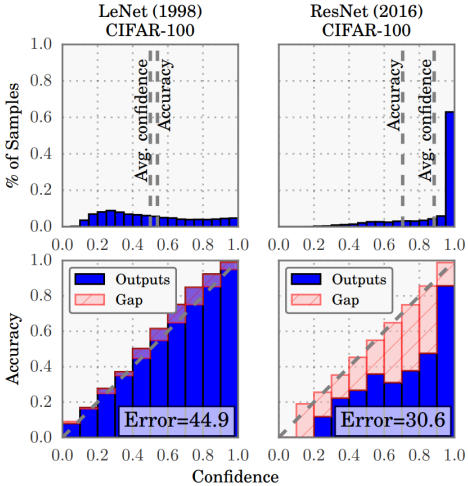


Figure 4: Tracciando a_m contro p_m per ogni bin otteniamo un grafico di affidabilità (da Guo et al., 2017).

Modelli per la classificazione

Naive Bayes

Concludiamo con una semplice baseline per la classificazione, chiamata **Naive Bayes**. Ricordiamo il teorema di Bayes:

$$p(y \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y)p(y)}{p(\mathbf{x})} \,. \tag{22}$$

Si noti come $p(y)$ (detta la **prior distribution** sulle classi) è relativamente semplice da stimare con metodi frequentisti. Inoltre, nel caso ci interessi solo il massimo di $p(y \mid \mathbf{x})$, possiamo ignorare il denominatore (costante per ogni classe).

Per rendere il problema trattabile, l'assunzione di base del **Naive Bayes** è che ogni feature è indipendente data la classe:

$$p(\mathbf{x} \mid y) = \prod_{i=1}^{d} p(x_i \mid y) \tag{23}$$

Sotto questa assunzione, ogni termine $p(x_i \mid y)$ è una semplice distribuzione univariata che può essere stimata a partire dai dati.

Ad esempio, supponiamo che x_i sia una feature categorica con n_i classi. In questo caso, $p(x_i \mid y)$ diventa una distribuzione categorica, e:

$$p(x_i = j \mid y) = \frac{|\{(\mathbf{x}, y') \in \mathcal{S} \mid x_i = j \text{ e } y' = y\}|}{|\{(\mathbf{x}, y') \in \mathcal{S} \mid y' = y\}|} \tag{24}$$

dove n è la dimensione del dataset. Questo non richiede altro che contare in maniera opportuna gli elementi del dataset.

Nel caso di una feature continua, possiamo stimare media e varianza di una Gaussiana univariata (condizionale alla classe).

Riscrivendo nel dominio logaritmico, scegliamo quindi la classe che massimizza la quantità:

$$y^* = \arg\max_y \left\{ \log p(y) + \sum_i \log p(x_i \mid y) \right\} \tag{25}$$

Si può estendere Naive Bayes al caso di regressione trasformando l'output y con un binning in un numero predeterminato di intervalli (ma è meno comune).

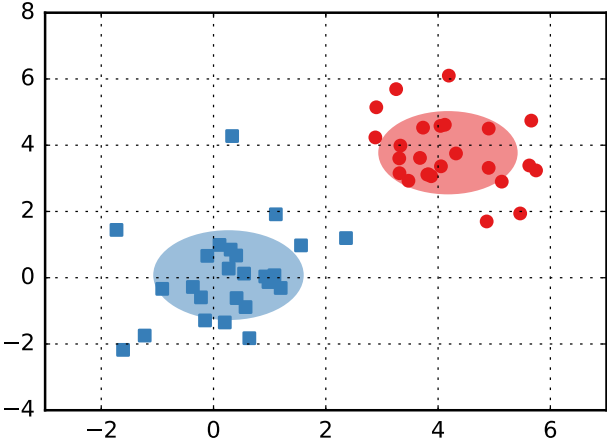


Figure 5: Esempio di Naive Bayes su un semplice dataset 2D con due distribuzioni Gaussiane univariate.