



# MACHINE LEARNING

## - Modelli Generativi -

Probabilità Congiunta e Condizionata:

$$p(x|y) = p(x, y) / p(y)$$

$$p(x, y) = p(y, x)$$

$$p(x, y) = p(x|y)p(y)$$

Caso specifico di variabili *indipendenti*:

$$p(x|y) = p(x)$$

$$p(y|x) = p(y)$$

$$p(x, y) = p(x)p(y)$$

Probabilità congiunta applicata ad eventi multipli (*chain rule*):

$$\begin{aligned} p(x_1, \dots, x_n) &= p(x_1 \mid x_2, \dots, x_n) p(x_2, \dots, x_n) \\ &= p(x_1 \mid x_2, \dots, x_n) p(x_2 \mid x_3, \dots, x_n) p(x_3, \dots, x_n) \\ &= \dots \\ &= p(x_1 \mid x_2, \dots, x_n) p(x_2 \mid x_3, \dots, x_n) \cdots p(x_{n-1} \mid x_n) p(x_n) \end{aligned}$$

# Teorema di Bayes

$$p(x, y) = p(y, x)$$

$$p(x|y)p(y) = p(y|x)p(x)$$

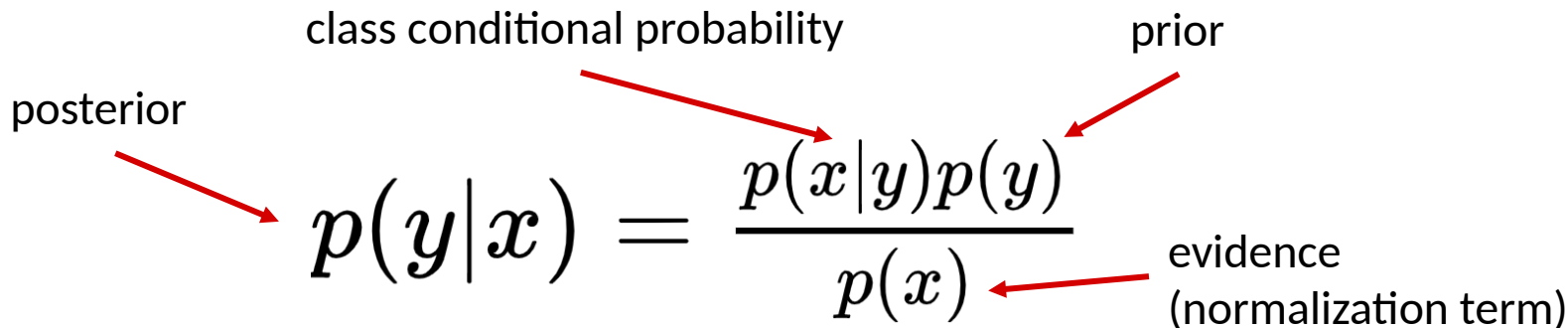
$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

posterior

class conditional probability

prior

evidence (normalization term)

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$


$x_j, j = 1, \dots, d$	→	feature (se categorica, $x_j \in \{1, \dots, m\}$ )
$\mathbf{x} = [x_1, \dots, x_d]$	→	feature vector
$y$	→	variable target (se categorica, $y \in \{1, \dots, k\}$ )
$(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \dots, n$	→	training sample
$\mathbf{1}\{\text{cond}\}$	→	funzione indicatore: restituisce <i>1</i> se <i>cond</i> = <i>true</i> , <i>0</i> se <i>cond</i> = <i>false</i>

# Modelli Generativi

## Modelli Discriminativi

Modellano direttamente  $p(y|\mathbf{x})$

## Modelli Generativi

Modellano  $p(\mathbf{x}|y)$  e  $p(y)$   
(e, a volte,  $p(\mathbf{x})$ )

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})}$$

I modelli generativi possono essere usati (anche) per *task generativi*, ovvero per generare nuovi dati (samples) sintetici seguendo una distribuzione ( $p(\mathbf{x})$  o  $p(\mathbf{x}|y)$ ) appresa tramite un dataset di training

Ad esempio, utilizzando tecniche di Deep Learning e modelli generativi, si possono generare artificialmente immagini, video, frasi di testo, brani musicali, ...

GPT è un esempio di modello generativo per la generazione di testo scritto

Purtroppo, in questo corso di introduzione al ML non avremo tempo per approfondire questo aspetto che oggi ha una grossa rilevanza pratica e che di solito viene chiamato «Generative AI»



È importante a questo punto chiarire la differenza tra modello e task. Nelle definizioni che seguono, sia le feature  $\mathbf{x}$  che la variabile target  $y$  possono essere sia numeriche che categoriche:

- Task discriminativo: predire il valore di  $y$  dato  $\mathbf{x}$  (e.g., classificazione, regressione)
- Task generativo: generare nuovi dati  $\mathbf{x}$  non appartenenti a  $T$  ma che seguono la distribuzione dei dati in  $T$
- Modello discriminativo: modella la distribuzione  $p(y|\mathbf{x})$
- Modello generativo: modella (alcune del-)le distribuzioni  $p(\mathbf{x})$ ,  $p(\mathbf{x}|y)$ ,  $p(y)$

Un modello generativo però può essere usato anche per task discriminativi (che sono l'interesse principale di questo corso)

# Modelli Generativi (usati per task discriminativi)

Concentriamoci sul task di classificazione e partiamo dalla decision rule (supponendo, cioè, di avere già in qualche modo addestrato dei modelli generativi)

Decision rule:

$$\begin{aligned}\arg \max_{y \in \{1, \dots, k\}} p(y|\mathbf{x}) &= \arg \max_{y \in \{1, \dots, k\}} \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \\ &= \arg \max_{y \in \{1, \dots, k\}} p(\mathbf{x}|y)p(y)\end{aligned}$$

Il significato delle formule qui sopra è: cerco il valore di  $y \in \{1, \dots, k\}$  che rende massima la (stima di) probabilità  $p(y|\mathbf{x})$

L'ultima equazione segue dal fatto che il denominatore ( $p(\mathbf{x})$ ) in questo caso è inutile (non dipende da  $y$ ) e quindi posso evitare di considerarlo

Per usare un modello generativo in un task discriminativo, quindi, avrò bisogno di modellare (separatamente)  $p(y|\mathbf{x})$  e  $p(y)$

Oltre alla distinzione tra modelli e task, che possono essere sia discriminativi che generativi, abbiamo già visto che esiste un'ulteriore distinzione è tra:

- *Metodi parametrici e metodi non parametrici*

Ciò vale anche per i modelli generativi, per cui posso avere varie combinazioni: ad esempio, un modello generativo parametrico e un modello discriminativo non parametrico, ecc.

Vedremo ora degli esempi di modelli generativi, sia parametrici che non parametrici (utilizzati per task discriminativi)

## Esempio: un modello generativo *non-parametrico*

Un esempio di modello generativo non parametrico è dato dalla modellazione di  $p(\mathbf{x}|y)$  e  $p(y)$  attraverso la *frequenza delle occorrenze* dei valori delle variabili  $\mathbf{x}$  e  $y$  nel dataset di training

Intuitivamente, si tratta di costruire degli istogrammi che «continuo» i vari valori delle variabili  $\mathbf{x}$  ed  $y$  negli esempi in  $T$  (tra un pò vedremo i dettagli)

E' un modello particolarmente indicato nel caso in cui le feature ( $\mathbf{x}$ ) siano categoriche ma può essere usato anche con feature numeriche

Ci concentreremo su un task di classificazione, assumendo quindi che anche  $y$  sia una variabile categorica

# Esempio: un modello generativo *non-parametrico*

Supponiamo che  $y \in \{1, \dots, k\}$  e di avere *una sola* variabile indipendente  $x$ , anch'essa discreta («categorica»):  $x \in \{1, \dots, m\}$ .

Nell'esempio a destra:

- $k = 4$  e  $m = 4$
- VERY LOW -> 1
- LOW -> 2
- MODERATE -> 3
- HIGH -> 4
- USA -> 1
- ITALY -> 2
- GERMANY -> 3
- FRANCE -> 4

Feature	Label
Home Country	Risk Label
USA	LOW
ITALY	VERY LOW
ITALY	MODERATE
GERMANY	HIGH
FRANCE	HIGH

# Modello generativo non-parametrico: training

Chiamo  $T$  il dataset di training ( $T = \{(x^{(i)}, y^{(i)})\}, i = 1, \dots, n$ )

Cominciamo col modellare  $p(y)$ . Ho bisogno di stimare:

- $p(y=1), \dots, p(y=k)$
- Posso farlo semplicemente contando le occorrenze (normalizzate) di ogni valore di  $y$  in  $T$ , ovvero la percentuale di esempi in  $T$  associati con ognuno dei  $k$  possibili valori di  $y$
- Implementativamente, posso usare un vettore  $o$  definito come segue:

$o[y] = \# \text{ occorrenze di esempi con label } y \text{ in } T / \text{cardinalità di } T$

Nell'esempio a destra ho che:

- $p(y=1) = o[1] = 1/5$
- $p(y=2) = o[2] = 1/5$
- $p(y=3) = o[3] = 1/5$
- $p(y=4) = o[4] = 2/5$

$p(y=1), \dots, p(y=k)$  è la distribuzione di probabilità *a priori* dei valori di  $y$

Feature	Label
Home Country	Risk Label
USA	LOW
ITALY	VERY LOW
ITALY	MODERATE
GERMANY	HIGH
FRANCE	HIGH

# Modello generativo *non-parametrico*: training

Alternativamente, e più semplicemente, posso assumere

- $p(y=1) = \dots = p(y=k) = 1/k$  (prior uniforme, ovvero classi bilanciate)

Per stimare  $p(x|y)$  uso  $T$  per calcolare le occorrenze di ogni possibile valore di  $x$  (ovvero  $x=1, x=2, \dots, x=m$ ) **dato**  $y$

In pratica, per ogni valore di  $y$ , posso usare un istogramma con  $m$  bins, un bin per ogni possibile valore della variabile  $x$

Quindi avrò  $k$  istogrammi diversi, ognuno con  $m$  bins, che posso rappresentare, ad esempio, con una matrice  $h$  di dimensioni  $m * k$



Ogni istogramma deve essere *normalizzato* perché deve rappresentare una distribuzione di probabilità (per cui «l'area» deve integrare ad 1), ovvero, *per ogni valore di  $y$* , devo avere che:  $h[1,y] + \dots + h[m,y] = 1$

In generale avremo:  $h[x,y] = p(x|y) = p(x,y) / p(y)$

Quindi posso stimare i valori di  $h$  usando:

$$h[x,y] = \# \text{ occorrenze di } (x,y) \text{ in } T / \# \text{ occorrenze di } y \text{ in } T$$

# Esempio (1)

- $h[x,y] = p(x|y) = p(x,y) / p(y)$
- Posso stimare i valori di  $h$  usando:  
 $h[x,y] = \# \text{ occorrenze di } (x,y) \text{ in } T / \# \text{ occorrenze di } y \text{ in } T$

		$p(x y=1)$	$p(x y=2)$	$p(x y=3)$	$p(x y=4)$
USA	1	0	1	0	0
Italy	2	1	0	1	0
Germany	3	0	0	0	1/2
France	4	0	0	0	1/2
		1	2	3	4
		very low	low	moderate	high

Feature	Label
Home Country	Risk Label
USA	LOW
ITALY	VERY LOW
ITALY	MODERATE
GERMANY	HIGH
FRANCE	HIGH

## Esempio (2)

- $h[x,y] = p(x|y) = p(x,y) / p(y)$
- Posso stimare i valori di  $h$  usando:  
 $h[x,y] = \# \text{ occorrenze di } (x,y) \text{ in } T / \# \text{ occorrenze di } y \text{ in } T$

		$p(x y=1)$	$p(x y=2)$	$p(x y=3)$	$p(x y=4)$
USA	1	0	1	0	0
Italy	2	1	0	1	0
Germany	3	0	0	0	1/3
France	4	0	0	0	2/3
		1	2	3	4
		very low	low	moderate	high

Feature	Label
Home Country	Risk Label
USA	LOW
ITALY	VERY LOW
ITALY	MODERATE
GERMANY	HIGH
FRANCE	HIGH
FRANCE	HIGH

In questo caso  $T$  contiene un «doppione», cosa che non è così strana, se si considera che sto rappresentando ogni entità del dominio (ovvero ogni persona) con una sola feature...

$o[y] = \# \text{ occorrenze di esempi con label } y \text{ in } T / \text{cardinalità di } T$

$h[x,y] = \# \text{ occorrenze di } (x,y) \text{ in } T / \# \text{ occorrenze di } y \text{ in } T$

Scritti in maniera più formale, se  $T = \{(x^{(i)}, y^{(i)})\}, (i = 1, \dots, n)$ :

$$o[b] = P(y = b) = \frac{\sum_{i=1}^n 1\{y^{(i)} = b\}}{n}$$

$$h[a, b] = P(x = a | y = b) = \frac{\sum_{i=1}^n 1\{x^{(i)} = a \wedge y^{(i)} = b\}}{\sum_{i=1}^n 1\{y^{(i)} = b\}}$$

# Modello generativo *non-parametrico*: *inference time*

A questo punto la predizione, a *inference time*, è semplice: per un dato valore di testing  $x$ , usando  $o[y]$  e  $h[x,y]$  posso calcolare:

$$\arg \max_{y \in \{1, \dots, k\}} p(x|y)p(y) = \arg \max_{y \in \{1, \dots, k\}} h[x, y]o[y]$$

# Esempio

$$\arg \max_{y \in \{1, \dots, k\}} p(x|y)p(y) = \arg \max_{y \in \{1, \dots, k\}} h[x, y]o[y]$$

$$o[1] = 1/6$$

$$o[2] = 1/6$$

$$o[3] = 1/6$$

$$o[4] = 1/2$$

Supponendo di avere un  
sample di testing  $x = 4$   
(FRANCE), avrò che:

- $h[4, 1] * o[1] = 0$
- $h[4, 2] * o[2] = 0$
- $h[4, 3] * o[3] = 0$
- $h[4, 4] * o[4] = 2/3 * 1/2$   
 $= 2/6 = 1/3$

Quindi scelgo  $y = 4$ , ovvero  
*High*

USA 1	0	1	0	0
Italy 2	1	0	1	0
Germany 3	0	0	0	1/3
France 4	0	0	0	2/3
	1	2	3	4
	very low	low	moderate	high

$h[x, y]$

Feature		Label	
Home Country		Risk Label	
USA		LOW	
ITALY		VERY LOW	
ITALY		MODERATE	
GERMANY		HIGH	
FRANCE		HIGH	
FRANCE		HIGH	

# Esempio

$$\arg \max_{y \in \{1, \dots, k\}} p(x|y)p(y) = \arg \max_{y \in \{1, \dots, k\}} h[x, y]o[y]$$

$$o[1] = 1/6$$

$$o[2] = 1/6$$

$$o[3] = 1/6$$

$$o[4] = 1/2$$

**Osservazione:** se sommo i valori di  $h[x, y]$  rispetto alle righe (i.e., tenendo fisso  $x$ ) non ottengo 1: perché?

Esempio:

$$h[4, 1] + h[4, 2] + h[4, 3] + h[4, 4] = 2/3$$

USA	1	0	1	0	0
Italy	2	1	0	1	0
Germany	3	0	0	0	1/3
France	4	0	0	0	2/3
		1	2	3	4
		very low	low	moderate	high

$h[x, y]$

Feature		Label	
Home Country		Risk Label	
USA		LOW	
ITALY		VERY LOW	
ITALY		MODERATE	
GERMANY		HIGH	
FRANCE		HIGH	
FRANCE		HIGH	

# Modello generativo non parametrico basato sulle occorrenze

Il modello appena visto è «non-parametrico» perché rappresenta l'informazione del training set in una forma non parametrica, attraverso le statistiche delle occorrenze e co-occorrenze nel training set, e non usa una funzione analitica dipendente da parametri

Uno dei vantaggi di questo modello è che posso facilmente usarlo con variabili categoriche dato che non dipende dall'ordinamento (arbitrario) che ho usato quando all'inizio ho rappresentato  $x$  e  $y$  con valori numerici



# Modello generativo non parametrico basato sulle occorrenze

Detto in altri termini, se, invece di usare:

- USA -> 1
- ITALY -> 2
- GERMANY -> 3
- FRANCE -> 4

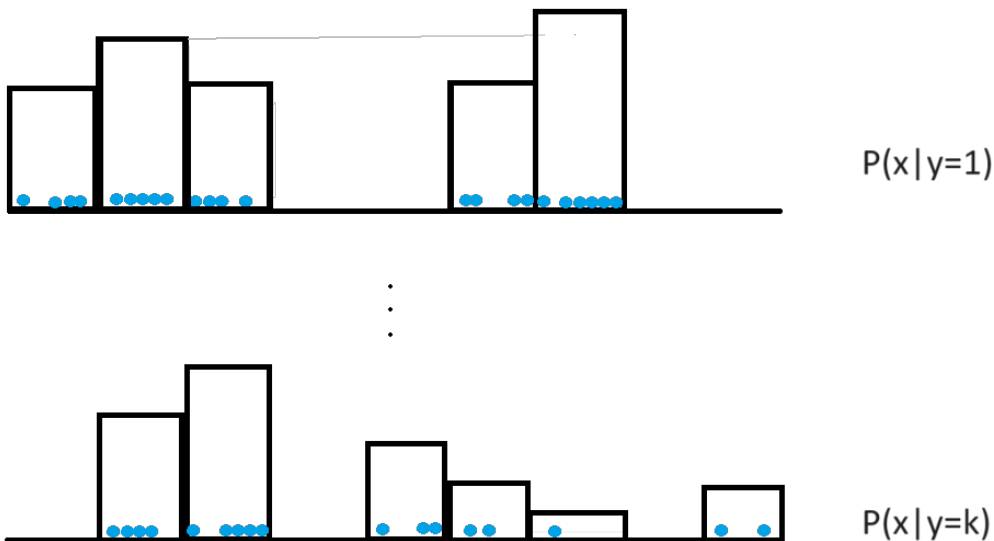
avessi usato, e.g.:

- USA -> 2
- ITALY -> 3
- GERMANY -> 4
- FRANCE -> 1

l'unica differenza sarebbe stata una permutazione delle righe di  $h$

# Modello generativo non parametrico con *feature* (x) numerica

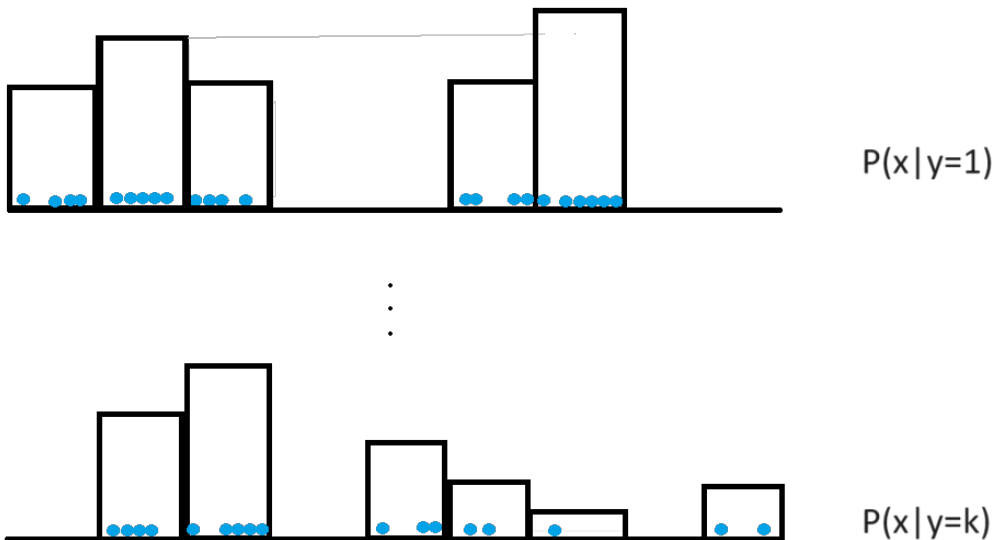
Tale modello può essere facilmente adattato a trattare il caso in cui  $x$  sia una variabile numerica: basta definire i bin dei vari istogrammi utilizzando dei sotto-intervalli (e.g., ottenuti con una partizione uniforme) dell'intervallo dei possibili valori di  $x$ , come nei grafici qui sotto:



# Modello generativo non parametrico con *feature* (x) numerica

Come sempre, devo ricordarmi che il mio obiettivo è ottenere  $k$  distribuzioni di probabilità  $p(x|y=1)$ , ...  $p(x|y=k)$ , per cui ogni istogramma dovrà essere normalizzato dividendo per la sua «area», cioè per la somma dei valori di tutti i suoi bin

Esattamente come prima, per la classe  $i$ -esima, «l'area» corrisponde al numero di occorrenze di  $y=i$  in  $T$



## Esempio: un modello generativo *parametrico*

Vediamo ora un esempio di modello generativo *parametrico* continuando, per semplicità, ad assumere di avere una sola feature  $x$  e limitiamoci a modellare la dipendenza statistica più importante, ovvero  $p(x|y)$

Supponiamo che  $x$  sia una feature numerica e  $y$  categorica (task di classificazione)

L'aspetto più importante è che, invece di usare istogrammi (struttura dati non parametrica), devo modellare le  $k$  distribuzioni di probabilità  $p(x|y=1), \dots, p(x|y=k)$  con  $k$  corrispondenti funzioni analitiche dipendenti da parametri

# Esempio: un modello generativo *parametrico*

Per la precisione, dovrò usare  $k$  funzioni del tipo:

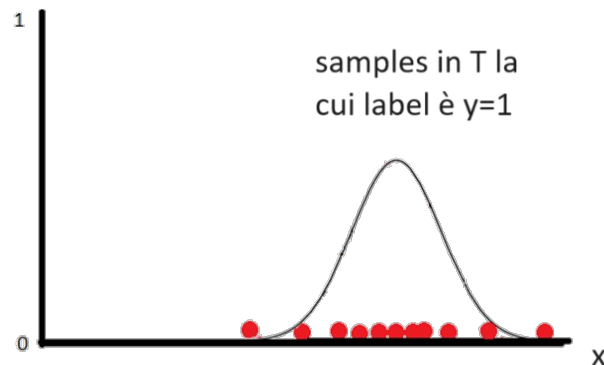
- $g_1(x; \theta_1) = p(x|y=1),$
- ...
- $g_i(x; \theta_i) = p(x|y=i),$
- ...
- $g_k(x; \theta_k) = p(x|y=k)$

Per farlo, anzitutto devo scegliere la *classe* di queste funzioni analitiche  
Per semplicità, useremo la stessa classe di funzioni per tutte e  $k$  le distribuzioni

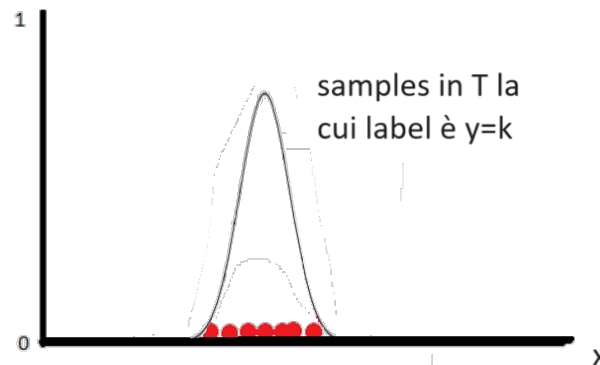
# Esempio: un modello generativo *parametrico*

Una scelta comune è la classe delle distribuzioni Gaussiane

*Questa scelta è valida se so che, per ogni valore di  $y$  (e.g.,  $y=i$ ), gli esempi  $(x, y=i)$  in  $T$  sono distribuiti in maniera Gaussiana come nel grafico a fianco*

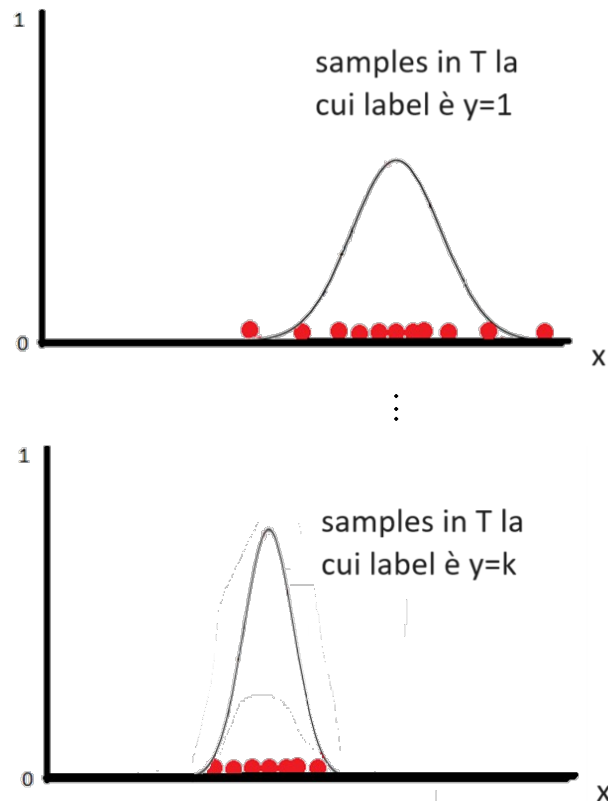


⋮



# Esempio: un modello generativo *parametrico*

Dato che  $g_i(x; \theta_i)$  è una Gaussiana, allora:  $\theta_i = [\mu_i, \sigma_i]$   
Quindi, per la  $i$ -esima funzione parametrica, dovrò calcolarmi la media e la deviazione standard della Gaussiana corrispondente, e lo farò utilizzando il sottoinsieme di  $T$  corrispondente a  $T_i = \{(x, y) \mid y=i\}$   
Posso farlo in forma diretta (closed form solution), e il calcolo è banale: per ogni  $i$ , devo semplicemente applicare le formule della media e della deviazione standard per ottenere  $[\mu_i, \sigma_i]$



# Esempio: un modello generativo *parametrico*

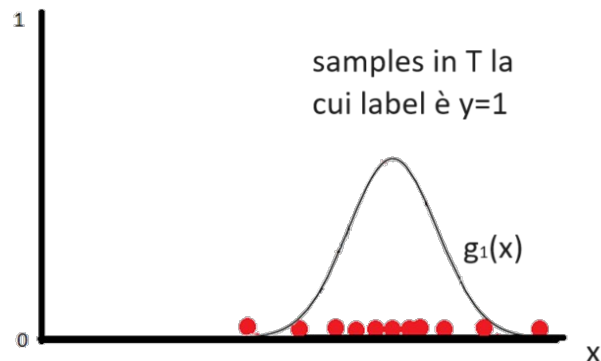
Per cui avrò che, per ogni  $i$ :

$$p(x|y=i) = g_i(x; [\mu_i, \sigma_i]) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp -\frac{(x - \mu_i)^2}{2\sigma_i^2}$$

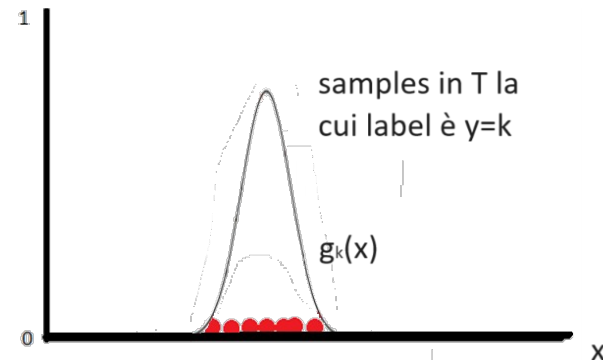
La decision rule rimane la stessa. Ad esempio, se assumo che  $p(y)$  sia una distribuzione uniforme (i.e.,  $p(y=1) = \dots = p(y=k) = 1/k$ ), allora ho:

$$\arg \max_{y \in \{1, \dots, k\}} p(x|y)p(y) = \arg \max_{y \in \{1, \dots, k\}} p(x|y)1/k = \arg \max_{y \in \{1, \dots, k\}} g_y(x)$$

dove, nell'ultima equazione,  $1/k$  scompare perché è un termine indipendente da  $y$



$\vdots$





# Modelli generativi multidimensionali

Estendiamo ora quanto visto finora al caso (più realistico) in cui ho  $d$  feature:

$$\mathbf{x} = [x_1, \dots, x_d]$$

La difficoltà principale sarà che ora ogni class-conditional probability  $p(\mathbf{x}|y=i)$  è una distribuzione multidimensionale, cioè è definita in uno spazio a  $d$  dimensioni

Nel caso parametrico (per ogni  $i$ ) potrei usare una Gaussiana multidimensionale (non lo vedremo ma è un'estensione semplice del caso precedente)

Approfondiamo, invece, il caso non parametrico, assumendo che tutte e  $d$  le feature siano di natura categorica

# Caso multidimensionale

Se ho  $d$  variabili categoriche, ognuna col suo specifico insieme di valori possibili, la situazione è la seguente

$$x_1 \in \{1, \dots, m_1\}$$

...

$$x_d \in \{1, \dots, m_d\}$$

Feature		Label
Job Category	Home Country	Risk Label
B	USA	LOW
C	ITALY	VERY LOW
B	ITALY	MODERATE
A	GERMANY	HIGH
A	FRANCE	HIGH
B	FRANCE	HIGH

In tal caso, a inference time, devo riuscire a calcolare:

$$\arg \max_{y \in \{1, \dots, k\}} p(x_1, \dots, x_d | y) p(y)$$

Mentre  $p(y)$  rimane invariata, ora la class conditional probability  $p(x_1, \dots, x_d | y)$  è una *distribuzione congiunta* (condizionata da  $y$ ) di  $d$  variabili diverse

# Caso multidimensionale: esempio

In quest'esempio, per ogni valore di  $y$ ,  
dovrei costruire un istogramma  
(normalizzato) a 2 dimensioni che  
rappresenti tutte le possibili  
combinazioni di valori per Job Category  
( $x_1$ ) e Home Country ( $x_2$ )

Feature		Label
Job Category	Home Country	Risk Label
B	USA	LOW
C	ITALY	VERY LOW
B	ITALY	MODERATE
A	GERMANY	HIGH
A	FRANCE	HIGH
B	FRANCE	HIGH

# Caso multidimensionale: esempio

$P(x_1, x_2 | y=1)$

1 (USA)	0	0	0
2 (ITALY)	0	0	1
3 (GERMANY)	0	0	0
4 (FRANCE)	0	0	0
	1 (A)	2(B)	3(C)

$Y = 1$  (VERY LOW)

$P(x_1, x_2, | y=4)$

1 (USA)	0	0	0
2 (ITALY)	0	0	0
3 (GERMANY)	1/3	0	0
4 (FRANCE)	1/3	1/3	0
	1 (A)	2(B)	3(C)

$Y = 4$  (HIGH)

...

Feature		Label
Job Category	Home Country	Risk Label
B	USA	LOW
C	ITALY	VERY LOW
B	ITALY	MODERATE
A	GERMANY	HIGH
A	FRANCE	HIGH
B	FRANCE	HIGH

# Caso multidimensionale: caso generale

Supponiamo, per semplicità, che  $m_1 = \dots m_d = m$

In tal caso, *per ogni valore di  $y$* , dovrei costruire un istogramma a  $d$  dimensioni con un totale di  $m^d$  bins...

Ciò comporterebbe due problemi:

1. La complessità spaziale è intrattabile con  $d$  grande. Avremmo bisogno di un tensore  $h$  della dimensione  $O(k m^d)$
2. Overfitting: avrei bisogno di un dataset di training con  $O(k m^d)$  samples per «riempire» tutti gli elementi di  $h$  (curse of dimensionality)

Una possibile soluzione è data da un'approssimazione chiamata naive Bayes assumption

# Naïve Bayes Classifier



Osservazione 1: usando la definizione di probabilità condizionata, la class conditional probability  $p(x_1, \dots, x_d|y)$  che voglio modellare può essere espressa come:

$$p(x_1, \dots, x_d|y) = p(x_1, \dots, x_d, y) / p(y) \quad (1)$$

Osservazione 2: Posso applicare la chain rule al numeratore del secondo termine dell'equazione (1) e ottenere:

$$p(x_1, \dots, x_d, y) = p(x_1 | x_2, \dots, x_d, y) p(x_2 | x_3, \dots, x_d, y) \dots p(x_d | y) p(y) \quad (2)$$

Combinando le due formule e semplificando  $p(y)$ , ottengo:

$$p(x_1, \dots, x_d|y) = p(x_1 | x_2, \dots, x_d, y) p(x_2 | x_3, \dots, x_d, y) \dots p(x_d | y) \quad (3)$$

La **Naïve Bayes Assumption** assume che le feature  $x_j$  siano tra loro *condizionalmente indipendenti*, dato  $y$ , ovvero che, *per un dato valore della variabile  $y$* , il valore che assume, e.g., la variabile  $x_i$  è indipendente dal valore che assume la variabile  $x_j$

Formalmente, ciò può essere espresso come:

$$p(x_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_d, y) = p(x_i | y) \text{ (per ogni } 1 \leq i \leq d) \quad (4)$$

Ovvero,  $x_i$  dipende solo da  $y$  ma non dalle altre variabili

Quindi, come caso particolare, abbiamo anche:

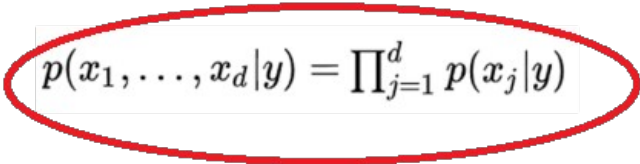
$$p(x_i | x_{i+1}, \dots, x_d, y) = p(x_i | y) \text{ (per ogni } 1 \leq i \leq d) \quad (5)$$

Usando la **Naïve Bayes Assumption** (equazione (5)), l'equazione (3) diventa:

$$p(x_1, \dots, x_d | y) = p(x_1 | y) p(x_2 | y) \dots p(x_d | y) = \prod_{j=1}^d p(x_j | y)$$

$$\arg \max_{y \in \{1, \dots, k\}} p(x_1, \dots, x_d | y) p(y) = \arg \max_{y \in \{1, \dots, k\}} \prod_{j=1}^d p(x_j | y) p(y)$$




$$p(x_1, \dots, x_d | y) = \prod_{j=1}^d p(x_j | y)$$

L'ipotesi Naïve Bayes ci permette di ricondurre la stima di una distribuzione  $d$ -dimensionale al prodotto di  $d$  distribuzioni *mono-dimensionali*

Quest'ultime possono essere stimate *indipendentemente l'una dalle altre* usando un modello mono-dimensionale, *parametrico oppure non-parametrico*, come, ad esempio, quelli visti in precedenza

$$p(x_1, \dots, x_d | y) = \prod_{j=1}^d p(x_j | y)$$

In sostanza, l'ipotesi di (presunta) indipendenza delle feature, mi permette di evitare di doverle modellare in maniera *congiunta*.

Ad esempio, usando un modello non parametrico basato sulle occorrenze, con quest'ipotesi ho bisogno di  $d$  matrici del tipo:

- $h_1[x, y] = p(X_1 = x | Y = y)$  (è specifica per la feature  $x_1$ )
- $h_2[x, y] = p(X_2 = x | Y = y)$  (è specifica per la feature  $x_2$ )
- ...
- $h_d[x, y] = p(X_d = x | Y = y)$  (è specifica per la feature  $x_d$ )

- $h_1[x, y] = p(X_1 = x \mid Y = y)$  (è specifica per la feature  $x_1$ )
- $h_2[x, y] = p(X_2 = x \mid Y = y)$  (è specifica per la feature  $x_2$ )
- ...
- $h_d[x, y] = p(X_d = x \mid Y = y)$  (è specifica per la feature  $x_d$ )

Posso concisamente rappresentare queste  $d$  matrici feature-specific con un unico tensore  $H$  di dimensioni  $d * m * k$  tale che:  $H[j, x, y] = h_j[x, y] = P(X_j = x \mid Y = y)$

La prima dimensione di  $H$  serve ad indicizzare la  $j$ -esima feature. La seconda rappresenta il valore che quella feature assume. La terza rappresenta il valore della variabile target

Ese.:  $H[2, 3, 2]$  rappresenta  $p(x_2 = 3 \mid y = 2) = p(x_2 = \text{GERMANY} \mid y = \text{LOW})$

Ho ricondotto il problema di modellazione da  $O(k m^d)$  ad  $O(d k m)$  !

A inference time, avendo pre-calcolato  $H$ , dato un vettore di feature (di testing)  $[x_1, \dots, x_d]$ , la decision rule diventa:

$$\arg \max_{y \in \{1, \dots, k\}} p(x_1, \dots, x_d | y) p(y) = \arg \max_{y \in \{1, \dots, k\}} \prod_{j=1}^d p(x_j | y) p(y) = \arg \max_{y \in \{1, \dots, k\}} \prod_{j=1}^d H[j, x_j, y] o[y]$$

# Naïve Bayes Classifier: vantaggi e svantaggi

L'assunzione Naive Bayes è un'approssimazione della realtà. Se le feature  $x_1, \dots, x_d$  sono solo debolmente dipendenti, tale assunzione introduce un errore tutto sommato trascurabile

Quest'errore di modellazione è compensato dal fatto che ho ricondotto la stima di una probabilità condizionale  $p(\mathbf{x}|y)$  multidimensionale al prodotto di probabilità mono-dimensionali:

$$p(x_1, \dots, x_d|y) = \prod_{j=1}^d p(x_j|y)$$

Ciò porta a ridurre enormemente il problema di overfitting, perché ora ho bisogno di  $O(d k m)$  training samples anziché  $O(k m^d)$



Intuitivamente, l'assunzione Naive Bayes porta ad uno «scambio» tra due tipi di errori: anzichè rischiare un severo overfitting (errore di generalizzazione), preferisco introdurre un errore nella modellazione congiunta delle feature

Ma ho risolto il problema dell'overfitting completamente?

Ovviamente no. L'overfitting è un problema trasversale a tutti i metodi di ML: chi più, chi meno, tutti ne sono soggetti e non è possibile avere un metodo che sia garantito esserne completamente immune

Ad esempio, se ho pochi samples in  $T$ , alcuni dei bin degli istogrammi dei modelli non-parametrici monodimensionali potrebbero essere stati stimati con pochi esempi di training, quindi essere inaffidabili

# Naïve Bayes Classifier: overfitting

A peggiorar le cose, nel caso specifico del modello generativo non parametrico discusso finora, alcuni bin potrebbero avere valore 0

Ad esempio, potrei avere che, per un qualche valore di  $y = b$ , feature  $x_j$  e feature value  $x_j = a$ ,  $H[j, a, b] = 0$  semplicemente perchè  $x_j = a$  non è mai stato incontrato nel training set insieme ad  $y = b$

Nell'esempio a fianco ciò accade, e.g., con  $y = b = 1$  ed  $x_2 = a = 1$

Il problema è che, se esiste anche un solo elemento  $H[j, a, b] = 0$ , tutto il prodotto si annulla:

$$o[b] * H[1, a_1, b] * \dots * H[j, a_j, b] * \dots * H[d, a_d, b] = 0$$

		Feature				Label
		1	2	3	4	
USA	1	0	1	0	0	
Italy	2	1	0	1	0	
Germany	3	0	0	0	1/3	
France	4	0	0	0	2/3	
		1	2	3	4	
		very low	low	moderate	high	

Home Country	Risk Label
USA	LOW
ITALY	VERY LOW
ITALY	MODERATE
GERMANY	HIGH
FRANCE	HIGH
FRANCE	HIGH

$$H[2, x, y]$$

$$\arg \max_{y \in \{1, \dots, k\}} \prod_{j=1}^d H[j, x_j, y] o[y]$$

Il «Laplace Smoothing» è una *regolarizzazione* che permette di risolvere il problema di stime uguali a 0 per mancanza di osservazioni nel training set.

Se  $y \in \{1, \dots, k\}$  e  $x_j \in \{1, \dots, m\}$ :

$$H[j, a, b] = \frac{1 + \sum_{i=1}^n 1\{x_j^{(i)} = a \wedge y^{(i)} = b\}}{m + \sum_{i=1}^n 1\{y^{(i)} = b\}} \quad (j= 1, \dots, d)$$

- <https://see.stanford.edu/materials/aimlcs229/cs229-notes2.pdf>
- **A tu per tu col Machine Learning. L'incredibile viaggio di un developer nel favoloso mondo della Data Science**, Alessandro Cucci, The Dot Company, 2017 [cap.6]
- **An introduction to statistical learning**, Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani, New York: Springer, 2013 [cap. 4]
- **Pattern Classification, second edition**, R. O. Duda, P. E. Hart, D. G. Stork, Wiley-Interscience, 2000