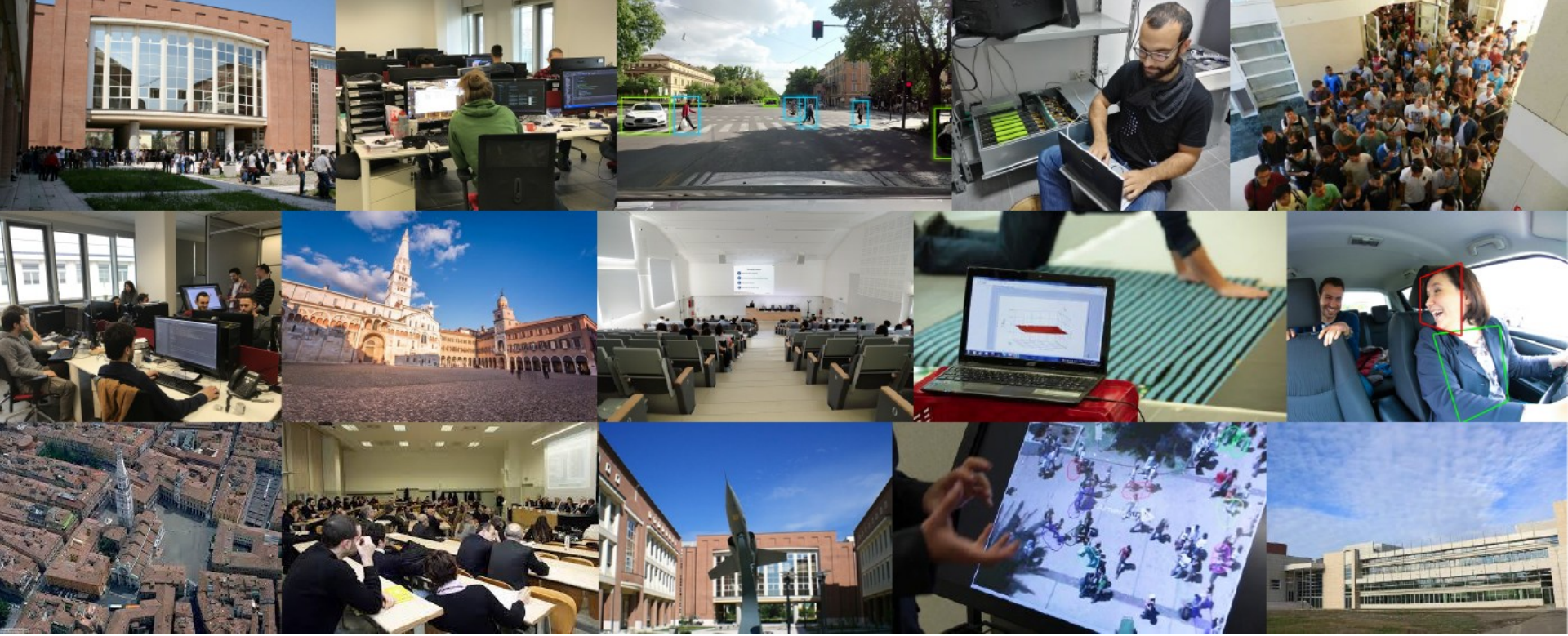


MACHINE LEARNING

- Metodi non parametrici -

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Notazione

x_j, $j = 1, \ldots, d$	$\rightarrow$	feature
$\mathbf{x} = [x_1, \ldots, x_d]$	$\rightarrow$	feature vector
y	$\rightarrow$	variabile target
$T = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$	$\rightarrow$	dataset di training

Metodi non parametrici

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Finora abbiamo visto solo metodi parametrici, sia per la regressione che per la classificazione

Oggi introdurremo una classe diversa di metodi, detti “non parametrici”, che possono essere usati sia per task di regressione che di classificazione

Tipicamente, questi metodi sono non lineari, nel senso che le funzioni di predizione risultanti dagli algoritmi che vedremo producono un output che è una funzione non lineare dell’input

Metodi parametrici e non parametrici

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Benchè una definizione precisa sia difficile, possiamo distinguerli così:

Parametrici:

- Si basano su una funzione ipotesi $h_\theta(x)$ la cui forma specifica dipende dai valori dei parametri (θ) che sono i coefficienti costanti di $h_\theta(x)$
- $h_\theta()$ è una funzione «analitica», dove, con un leggero abuso terminologico, intendiamo con funzione analitica una funzione composta da funzioni analitiche note (polinomi, logaritmo, esponenziale, ecc.)
- Il valore esatto di θ (ottenuto in fase di training) permette di scegliere $h()$ all'interno di una prefissata classe di funzioni (e. g., le funzioni lineari, o quelle gaussiane, ecc.)

Non parametrici:

- Si basano su algoritmi di ML più generici, in cui la predizione della variabile target in base all'input non dipende da una specifica classe di funzioni analitiche
- Spesso (ma non sempre!) i metodi non parametrici memorizzano esplicitamente tutti o alcuni degli esempi del training set
- La fase di training non usa il gradiente rispetto a θ (perché non c'è una funzione analitica dipendente da θ …) ma si basa su algoritmi specifici
- Attenzione, perché i «metodi non parametrici» possono avere i loro «parametri»… Sembra una contraddizione, ma il punto è che questi parametri non sono i coefficienti di una funzione analitica $h()$ bensì valori che definiscono le strutture dati dello specifico algoritmo

Metodi non parametrici

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Inizieremo studiando alcuni dei metodi non parametrici più comuni e li useremo sia per task di classificazione che per task di regressione:

- K-NN
- Alberi di decisione

Nelle prossime lezioni vedremo altri esempi di metodi non parametrici

Nearest Neighbor

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Nearest Neighbor - Classificazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Memorizzo tutto il dataset di training T
- Non esiste training…
- A inference time, adotto la seguente regola di decisione:

Dato $\boldsymbol{x}$, scelgo y_j tale che:

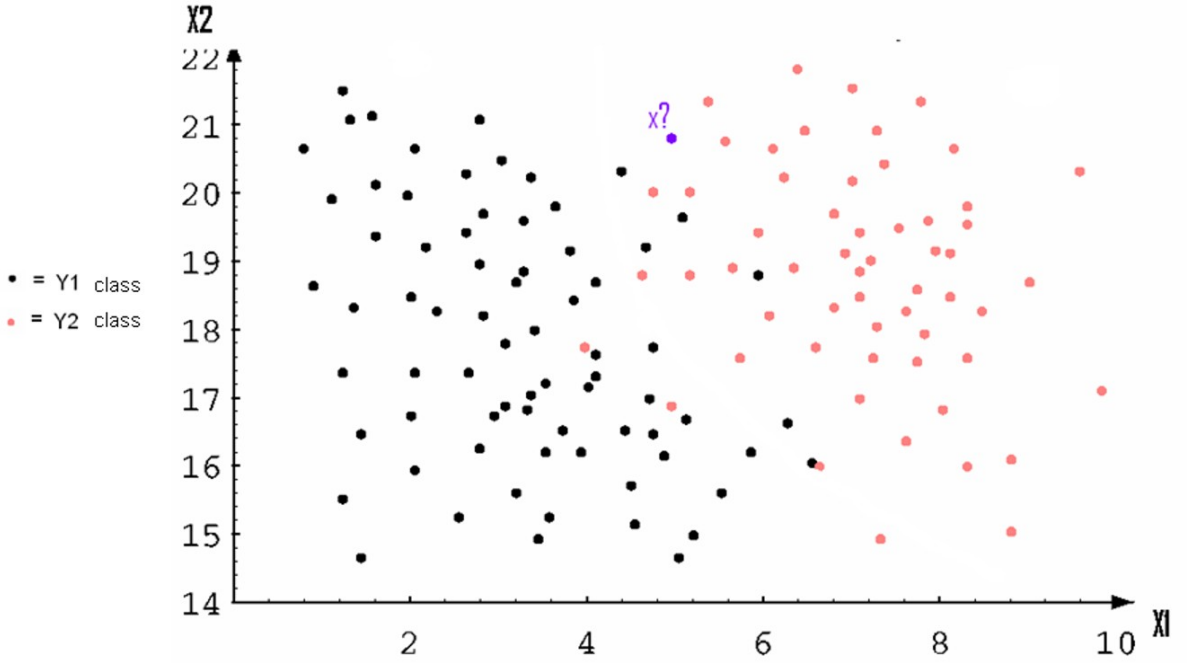
$(\boldsymbol{x}', y_j) \in T$ e

$Dist(\boldsymbol{x} - \boldsymbol{x}') <= Dist(\boldsymbol{x} - \boldsymbol{x}'')$ per ogni $(\boldsymbol{x}'', y_i) \in T$

che, se $Dist()$ è la distanza Euclidea, diventa:

$||\boldsymbol{x} - \boldsymbol{x}'|| <= ||\boldsymbol{x} - \boldsymbol{x}''||$ per ogni $(\boldsymbol{x}'', y_i) \in T$

Nearest Neighbor - Esempio

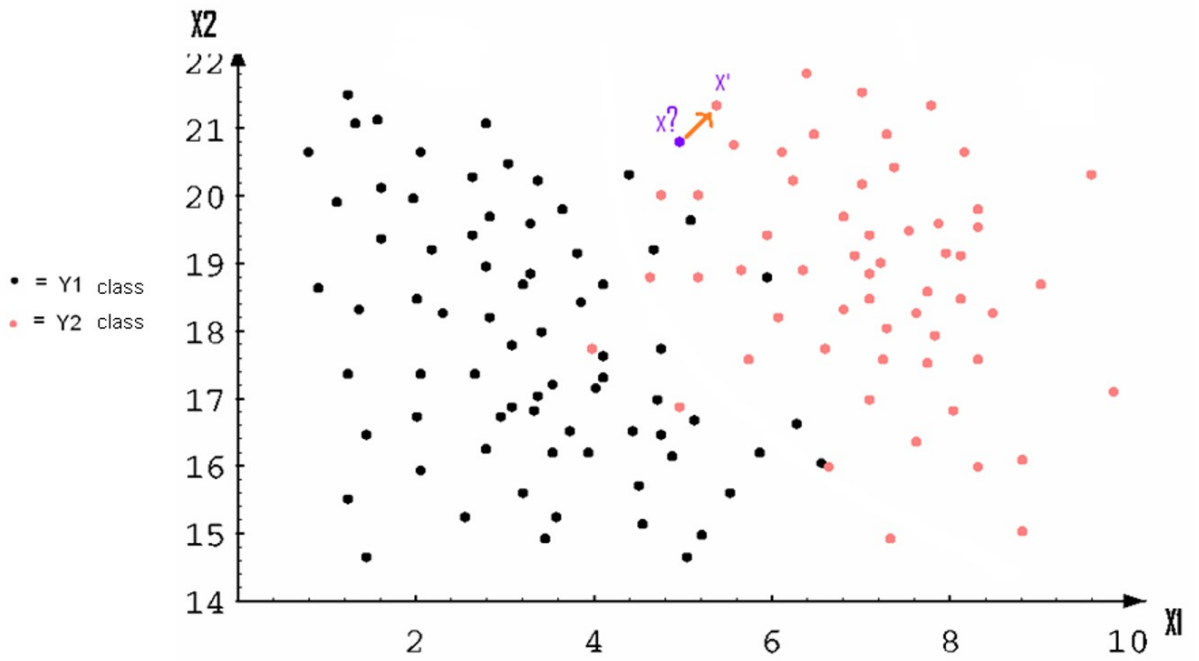


Dato $\boldsymbol{x}$, scelgo y_j tale che:

$(\boldsymbol{x}', y_j) \in T$ e

$||\boldsymbol{x} - \boldsymbol{x}'|| <= ||\boldsymbol{x} - \boldsymbol{x}''||$ per ogni $(\boldsymbol{x}'', y_i) \in T$

Nearest Neighbor - Esempio



Dato $\boldsymbol{x}$, scelgo y_j tale che:

$(\boldsymbol{x}', y_j) \in T$ e

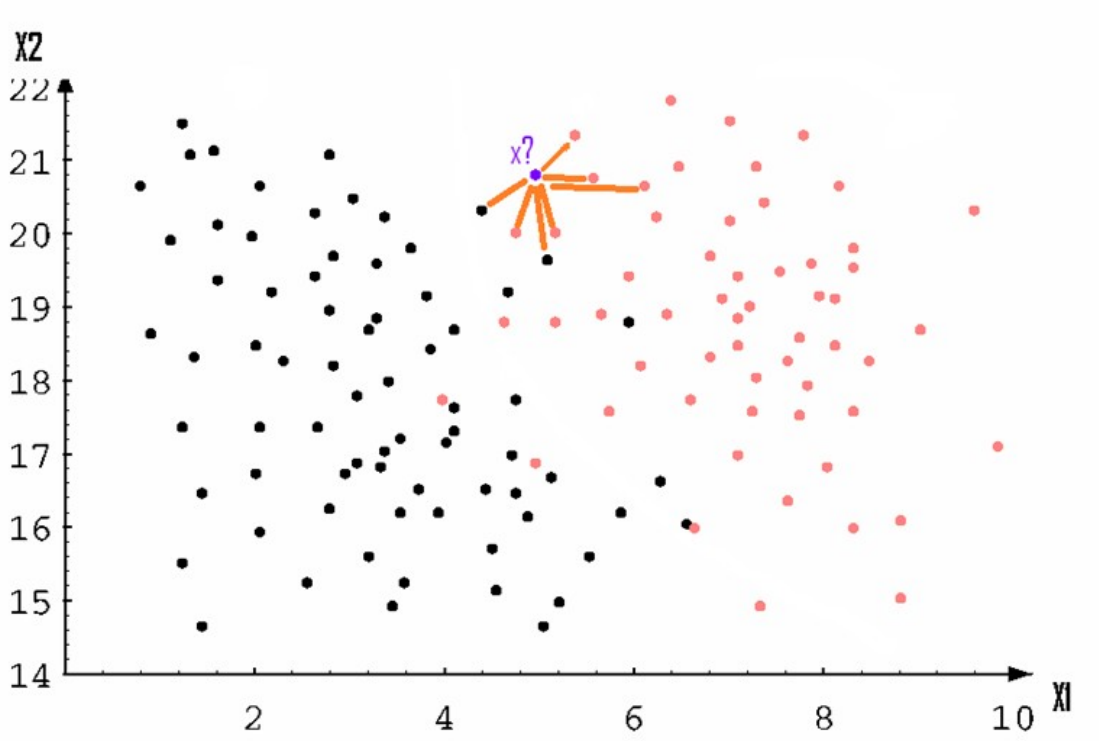
$||\boldsymbol{x} - \boldsymbol{x}'|| <= ||\boldsymbol{x} - \boldsymbol{x}''||$ per ogni $(\boldsymbol{x}'', y_i) \in T$

K-Nearest Neighbors (K-NN)- Classificazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

L'estensione al caso K-NN è semplice: considero *i k punti più vicini di* ***x***, con *k* numero prefissato, dopodiché scelgo la «moda», ovvero la classe «più votata»

K-Nearest Neighbors (K-NN)- Esempio (k = 7)



K-Nearest Neighbors - Regressione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

In un task di regressione l'output è dato dalla **media** (anziché dalla moda) delle *k* labels associate ai vicini di ***x***

Tale media può anche essere pesata, ad esempio con il valore inverso della distanza tra ***x*** e ognuno dei *k* vicini ($1/ Dist(\boldsymbol{x} - \boldsymbol{x}')$)

K-Nearest Neighbors

Nello scegliere l'iperparametro *k* bisogna considerare che:

1. Diminuendo il valore di *k* le previsioni diventano meno stabili e tendenzialmente più soggette ad overfitting
2. Nella classificazione, in cui si considera un voto a maggioranza tra le label (moda), di solito si fa sì che *k* sia un numero dispari per evitare parità nel «voto» (oppure si pesano i vicini usando l'inverso della loro distanza da ***x***)

Scalamento dei Dati

Anche nel K-NN è utile scalare le feature
Infatti, se *Dist*(***x*** – ***x***’) è la distanza Euclidea, feature con intervalli di valori possibili diversi (e.g., perché espresse in unità di misura diverse, come grammi e metri…) hanno un’influenza diversa nel determinare la distanza rispetto ad ***x***:

$$||\boldsymbol{x} - \boldsymbol{x}'|| = \sqrt{\sum_{j=1}^{d} (\boldsymbol{x}_j - \boldsymbol{x}'_j)^2}$$

K-Nearest Neighbors

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Vantaggi

- L'algoritmo è semplice e facile da implementare
- Non c'è bisogno di ottimizzare parametri (e.g., usando il gradient descent o altro...)
- E' un modello non lineare, quindi con una grossa capacità espressiva

Svantaggi

- L'algoritmo diventa significativamente più lento con l'aumentare del numero di sample e/o di feature
- Bisogna memorizzare tutto *T*
- Se *k* non è scelto bene può portare ad overfitting

Alberi di Decisione

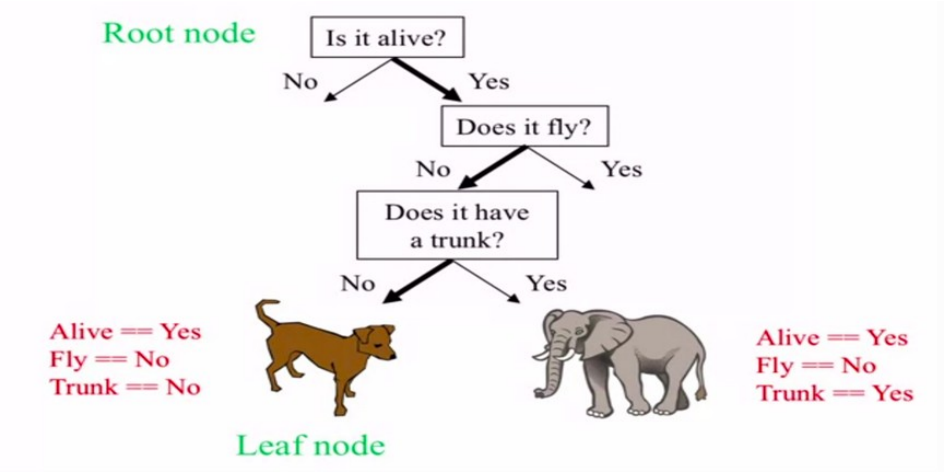
Alberi di Decisione

Un albero di decisione è una struttura ad albero (tipicamente binario) che può essere usata per task di classificazione o di regressione

Ad ogni nodo interno è associato un test

A inference time la decisione viene presa seguendo il risultato dei test dalla radice ad una foglia

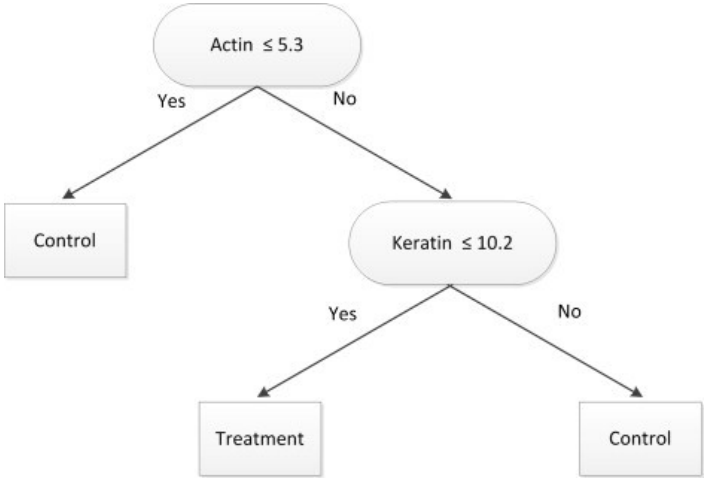
L'algoritmo di training consiste nel costruire (automaticamente) l'albero e lo vedremo tra poco



Alberi di Decisione

In manira più precisa, un albero di decisione può essere definito come segue:

- **Nodo interno** → E' associato ad un *test*, tipicamente semplice e basato su una sola feature (x_i) e un valore di soglia (t). Ese.: «x_i <= t ?»
- **Rami** → Corrispondono al risultato del test. Ese.: seguo il ramo sinistro se x_i <= t , il ramo destro se x_i > t
- **Foglia** → E' un nodo terminale al quale è associato un valore di output per la variabile target



Alberi di Decisione

L'algoritmo di apprendimento, *ovvero di costruzione (automatica) dell'albero*, deve scegliere, per ogni nodo, il test da associargli, ovvero:

1. La feature x_i su cui si basa il test, scelta tra le d possibili feature (x_1, ..., x_i, ... x_d)

2. Il valore di soglia t, scelto nell'intervallo dei valori possibili per quella feature x_i

Alberi di Decisione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Nella costruzione di un albero di decisione si segue un procedimento di tipo *Divide et Impera*:

- Selezionare un test per il nodo radice e creare un ramo per ogni possibile risultato del test. Ese.: «x_i <= t ?»
- Dividere le istanze di T in due sottoinsiemi, uno per ramo (T_s e T_d), a seconda del risultato del test scelto per quel nodo
- Ese.: T_s conterrà gli elementi di T per cui «x_i <= t ?» è vero, e T_d conterrà gli elementi di T per cui «x_i <= t ?» è falso
- T_s sarà quindi «ereditato» dal nodo corrispondente al ramo sinistro, mentre T_d verrà associato al nodo destro (splitting)
- Il test deve essere scelto in modo da massimizzare *l'omogeneità* interna a T_s e T_d. Ad ese., se il task è una classificazione binaria, vorrei che in T_s ci fossero prevalentemente samples di una classe e in T_d samples dell'altra
- Ripetere ricorsivamente i passi di sopra per ogni sottoinsieme creato
- Fermare la ricorsione per un nodo (e creare una foglia) quando: (a) le istanze del suo sottoinsieme sono poche oppure (b) le istanze del suo sottoinsieme sono sufficientemente *omogenee* tra loro
- Un nodo è omogeneo quando tutti i sample del sottoinsieme ereditato hanno lo stesso valore di y (classificazione) o un valore di y simile (regressione)

Alberi di Decisione

Il valore di output per la variabile target (*y*) da associare ad ogni foglia è scelto:

- **Per un task di classificazione:** selezionando la label presente a maggioranza (relativa) tra i samples di quel nodo foglia (moda)
- **Per un task di regressione:** calcolando la media delle label presenti in quel nodo foglia

Decision rule: a inference time la variabile target di un nuovo sample (***x***) è ottenuta sottoponendo ***x*** ad una serie di test, partendo dalla radice e scegliendo i rami in base all'esito di ogni test, e infine scegliendo il valore di *y* associato alla foglia risultante

Alberi di Decisione: algoritmo di costruzione

Come già accennato in precedenza, siccome la decisione finale viene presa nelle foglie, che devono essere il più omogenee possibile rispetto alla variabile target y, ad ogni passo di splitting selezioniamo la feature e il valore di soglia che porta alla maggiore «omogeneità» nei due sottoinsiemi di dati da associare ai nodi riusultanti

Quindi, se il nodo attuale è associato all'insieme T_v, devo trovare un test (i.e., una coppia (feature, soglia)) che mi permetta di aumentare l'omogeneità nei sottoinsiemi risultanti T_s e T_d

Alberi di Decisione - Classificazione

- Per un task di classificazione ($y \in \{1, \ldots, k\}$), l'omogeneità è misurata attraverso misure di *diseguaglianza della distribuzione di probabilità* delle *k* classi nei sottoinsiemi T_s e T_d

Tipicamente, per misurare il livello di diseguaglianza di una distribuzione di probabilità si può usare:

- *L'indice di Gini* (non approfondiremo quest'argomento)

Oppure

- Il disordine statistico (*Entropia*). L'Entropia dell'informazione è, tra l'altro, un concetto molto più generale, usato in vari ambiti del ML (e non solo…), per cui vediamolo più in dettaglio

Shannon Entropy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

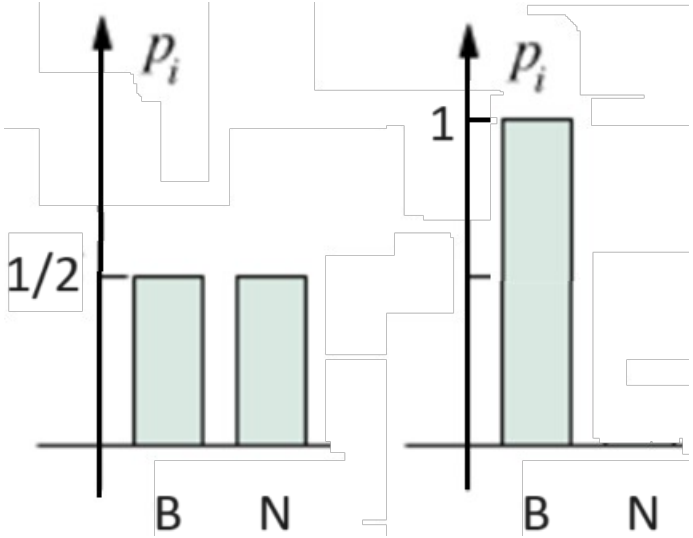
L'entropia dell'informazione misura il *disordine* (disomogeneità) di una distribuzione di probabilità, inteso come il livello di *incertezza* contenuto in quella distribuzione

L'idea intuitiva è che una distribuzione è tanto più «ordinata» quanto maggiore è la sicurezza che ho nello «scommettere» sul valore della variabile aleatoria associata a quella distribuzione

Shannon Entropy: Esempio

Ad ese., se in un'urna c'è un 50% di palline bianche e un 50% di palline nere ho il massimo disordine perché non ho nessuna informazione che mi permetta di preferire una classe di palline rispetto ad un'altra

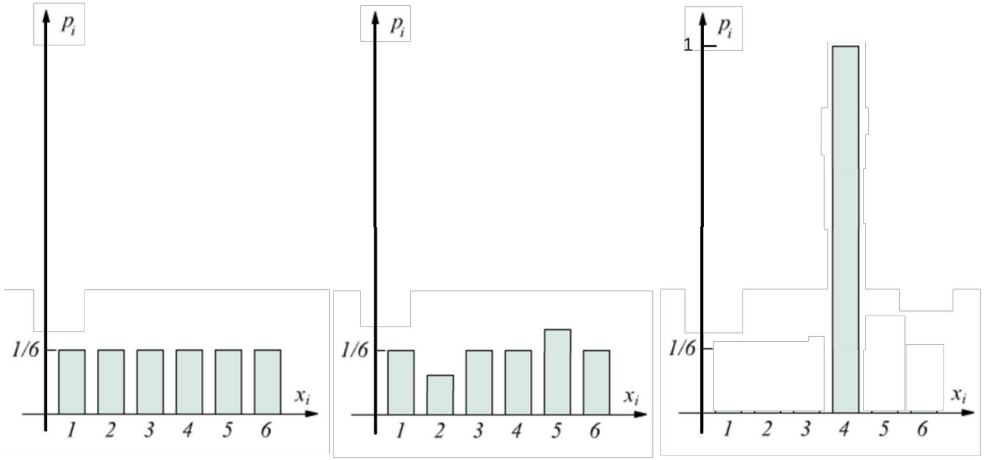
Viceversa, se so che nell'urna ci sono solo palline bianche, allora ho la sicurezza totale che posso scommettere sull'estrazione di una pallina bianca…



Shannon Entropy: Esempio

In maniera analoga, le tre distribuzioni di probabilità nella figura a fianco si riferiscono a 3 dadi diversi:

1. Dado standard. Distribuzione uniforme: massimo disordine
2. Dado truccato in cui il numero 5 ha una probabilità maggiore rispetto agli altri: l'ordine è aumentato
3. Dado truccato in cui esce sempre e solo il numero 4: ordine massimo



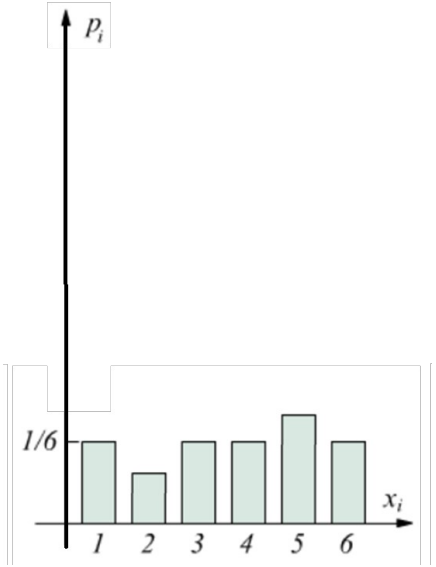
Shannon Entropy

Formalmente, dato un insieme $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ la Shannon Entropy (H) è definita come segue:

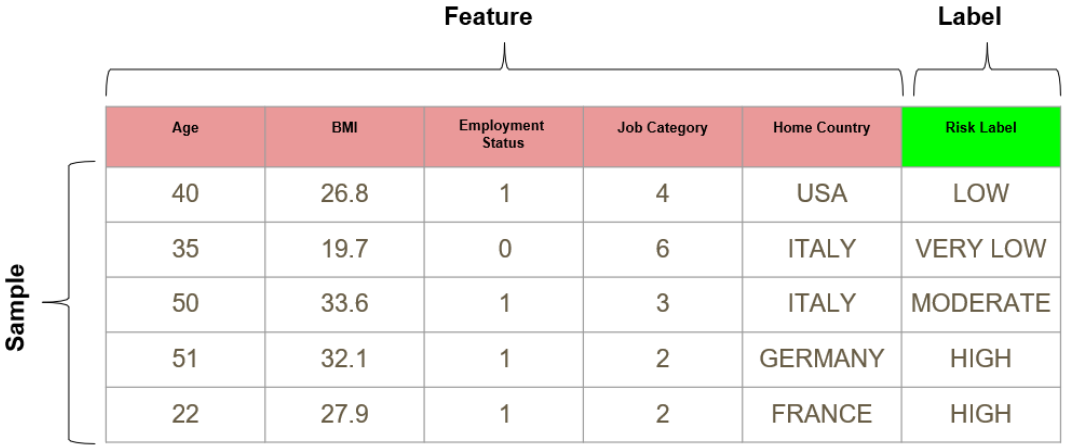
$$H(T) = -\sum_{i=1}^{k} p_i log(p_i)$$

dove $p_i = P(Y = i)$ è la probabilità di ogni classe

Ogni p_i può essere calcolato usando la percentuale di esempi di T con classe $Y = i$



Esempio



Age	BMI	Employment Status	Job Category	Home Country	Risk Label
40	26.8	1	4	USA	LOW
35	19.7	0	6	ITALY	VERY LOW
50	33.6	1	3	ITALY	MODERATE
51	32.1	1	2	GERMANY	HIGH
22	27.9	1	2	FRANCE	HIGH

- VERY LOW -> 1
- LOW -> 2
- MODERATE -> 3
- HIGH -> 4

Nell'esempio sopra ho che:

- $p_1 = P(Y = 1) = 1/5$
- $p_2 = P(Y = 2) = 1/5$
- $p_3 = P(Y = 3) = 1/5$
- $p_4 = P(Y = 4) = 2/5$

Proprietà dell'Entropia

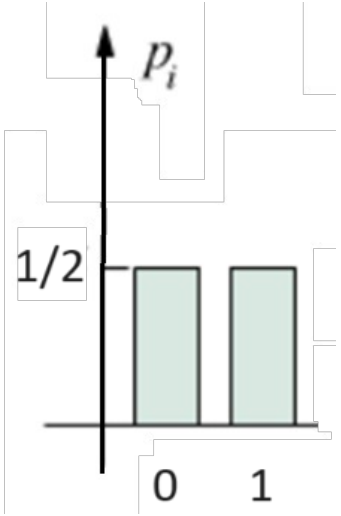
$$H(T) = -\sum_{i=1}^{k} p_i log(p_i)$$

- Tipicamente il logaritmo è in base 2
- Per convenzione si assume $0\ log(0) = 0$ (sfruttando il fatto che $lim_{p->0+}\ p\ log(p) = 0$)

Entropia: esempi

- In un task di classificazione binaria, se le due classi ($Y = 0$ e $Y = 1$) sono presenti in maniera perfettamente bilanciata in T, allora ho che $p_0 = p_1 = ½$

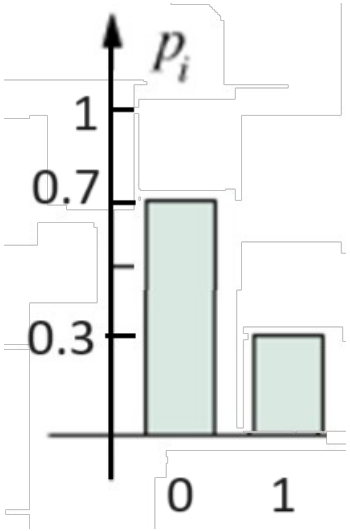
$$
\begin{aligned}
H(T) &= -\sum_{i=0}^{1} p_i log(p_i) \\
&= -\sum_{i=0}^{1} \frac{1}{2} \log_2 \frac{1}{2} \\
&= -\sum_{i=0}^{1} \frac{1}{2} \cdot (-1) = 1
\end{aligned}
$$



Entropia: esempi

- Nel caso in cui, invece, ad esempio, *P(Y = 0)* = 0.7 e *P(Y = 1)* = 0.3, allora avrei:

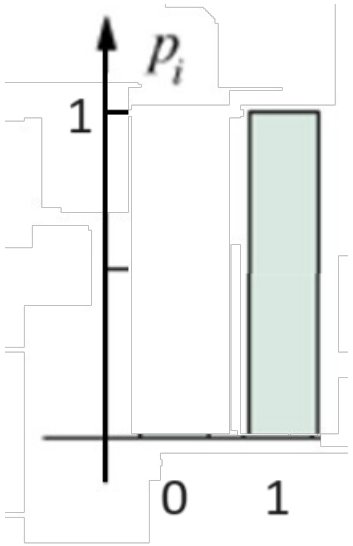
$$
\begin{aligned}
H(T) &= -0.7\log_2(0.7) - 0.3\log_2(0.3) \\
&\approx -0.7 \cdot (-0.515) - 0.3 \cdot (-1.737) \\
&= 0.8816 < 1
\end{aligned}
$$



Entropia: esempi

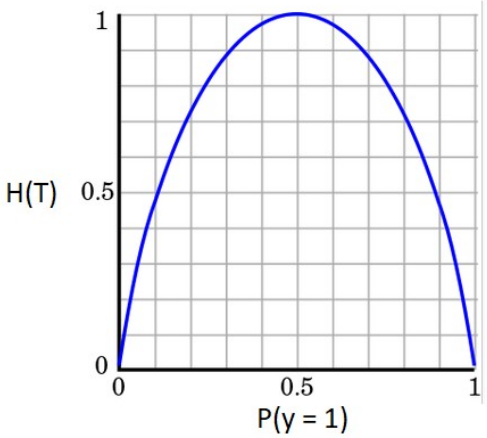
- Se *P(Y = 0)* = 0 e *P(Y = 1)* = 1, allora:

$$H(T) = -0\log_2(0) - 1\log_2(1)$$
$$= -0 - 0 = 0$$



Entropia: esempi

- Nel caso di due classi il grafico dell'entropia è quello qui sotto



- Più in generale, con *k* classi, l'entropia massima si ha con la distribuzione uniforme (*P(Y = 1)* = … *P(Y = k)*) e l'entropia minima si ha quando *T* contiene solo una classe (è totalmente "omogeneo")

Alberi di Decisione - Classificazione

L'obiettivo è trovare degli split che diminuiscano l'entropia negli insiemi associati ai due nodi risultanti.

Posso stimare la diminuzione di entropia tramite l'*information gain G*, che calcola quanto diminuisce l'entropia di due nodi figli rispetto all'entropia del nodo genitore

Alberi di Decisione - Classificazione

Algoritmo di costruzione, che chiamo con *Build(T)*:

Algorithm 1 $Build(T_v)$

1: Creo il nodo v
2: **if** $|T_v|$ è piccolo o tutti i suoi elementi sono associati alla stessa classe **then**
3: Mi fermo e creo un nodo foglia al quale associo la label maggioritaria in T_v
4: **end if**
5: **for** $i \in 1, ..., d$ **do**
6: Per la feature x_i seleziono un insieme di valori di soglia candidati: $C = \{t_1, ..., t_m\}$
7: **for** $t_j \in C$ **do**
8: Partiziono T_v nell'insieme di "sinistra" T_s e nell'insieme di "destra" T_d, basandomi sul test: $x_i \leq t_j$
9: Calcolo il *guadagno* $G_{i,j}$:

$$G_{i,j} = H(T_v) - \sum_{z \in \{s,d\}} \frac{|T_z|}{|T_v|} H(T_z)$$

10: **end for**
11: **end for**
12: Scelgo la coppia (x_i, t_j) che massimizza $G_{i,j}$ e associo il corrispondente test a v
13: Usando tale test creo due nodi figli di v, che ereditano, rispettivamente, T_s e T_d
14: Chiamo ricorsivamente $Build(T_s)$ e $Build(T_d)$

Al passo 2, $|T|$ indica la cardinalità dell'insieme *T*

Al passo 6, le soglie in *C* possono essere scelte, ad ese., in maniera uniforme o in maniera random rispetto all'intervallo di valori possibili per x_i

Alberi di Decisione - Regressione

In un task di regressione l'aumento dell'omogeneità che decide gli split ad ogni nodo può essere misurata, ad esempio, attraverso il **Residual Sum of Squares**, calcolato tra tutte le coppie di esempi di quel nodo:

$$RSS(T) = \sum_{y \in T} \sum_{y' \in T, y' \neq y} (y - y')^2$$

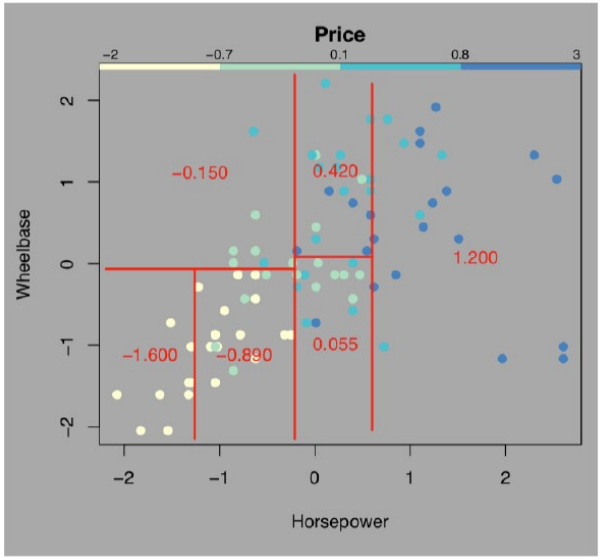
Usando la formula di sopra, se T_v, T_s e T_d sono i dataset associati, rispettivamente, al nodo radice, al nodo di sinistra e a quello di destra, allora lo split migliore è quello che massimizza la differenza (pesata) tra RSS(T_v) e RSS(T_s) + RSS(T_d)

Alberi di Decisione - Regressione

Il criterio di fermata si basa sulla varianza del nodo attuale T_v: quando *Var[Y]* in T_v è minore di una cera soglia, smetto di suddividere T_v

Ad ogni nodo foglia associo il valore corrispondente alla media calcolata sui valori di *Y* di quel nodo

Nella figura a destra, un esempio con 2 feature (cavalli vapore e interasse) usato per predire il prezzo di un'auto (*Y*)



Scalamento dei Dati

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Nel caso degli alberi di decisione non è necessario scalare il valore delle feature

Questo perché ogni test si basa su una sola feature, quindi le feature sono sempre usate in maniera indipendente l'una dalle altre

Alberi di Decisione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Vantaggi:

- Gli alberi di piccole dimensioni possono essere «interpretati», nel senso che, ispezionando i test associati ad ogni nodo, si può cercare di capire in base a cosa vengono prese le decisioni
- E' un modello non lineare, quindi con una grossa capacità espressiva
- Effettua implicitamente una feature selection

Svantaggi:

- Tende all'overfitting e piccoli cambiamenti nei dati di training possono risultare in grossi cambiamenti nella struttura dell'albero
- La profondità dell'albero e in genere la sua topologia non può essere ottimizzata efficientemente

Riferimenti

- **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. 5]
- **Pattern Classification, second edition,** R. O. Duda, P. E. Hart, D. G. Stork, Wiley-Interscience, 2000
- **An introduction to statistical learning**, Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani, New York: Springer, 2013 [cap. 4]