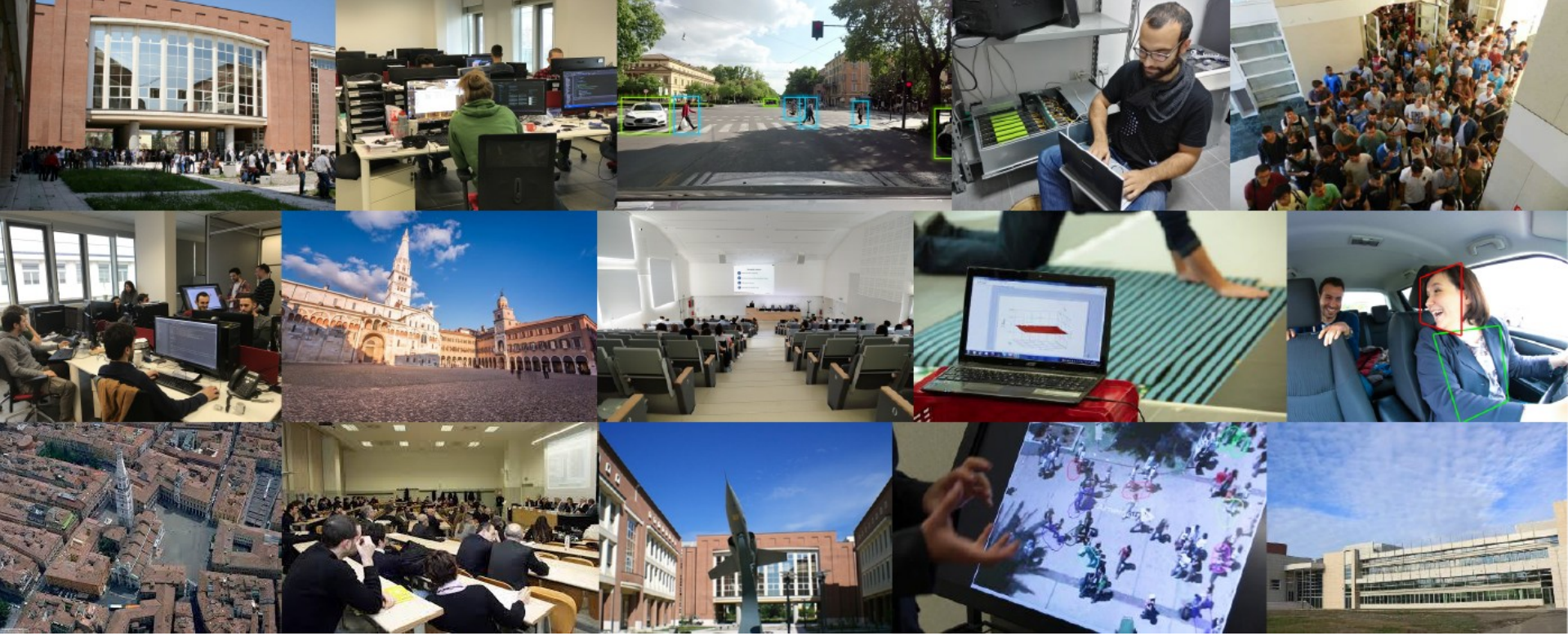


MACHINE LEARNING
Scalamento dei Dati e Pre-processing -

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Notazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

x_j, $j = 1, \ldots, d$ → variabili indipendenti (features)

$\mathbf{x} = [x_0, \ldots, x_{d-1}]$ → feature vector (x_0=1 per comodità)

y → variable target

$(\mathbf{x}^{(i)}, y^{(i)})$, $i = 1, \ldots, n$ → training samples

θ → vettore dei parametri

$\boldsymbol{\mu} = [\mu_1, \ldots, \mu_{d-1}]$ → vettore delle medie dei feature value prima dello scalamento

$\boldsymbol{\mu}' = [\mu'_1, \ldots, \mu'_{d-1}]$ → vettore delle medie dei feature value dopo lo scalamento

$\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_{d-1}]$ → vettore delle deviazioni standard prima dello scalamento

$\boldsymbol{\sigma}' = [\sigma'_1, \ldots, \sigma'_{d-1}]$ → vettore delle deviazioni standard dopo lo scalamento

Scalamento dei Dati

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Scalamento dei Dati

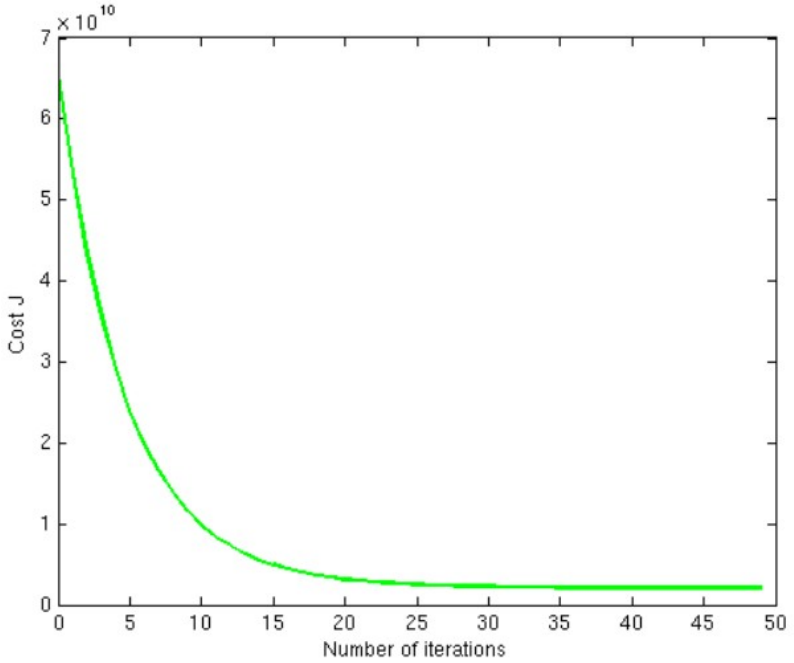
AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Lo scalamento dei dati (o «normalizzazione») è un’operazione di pre-processing del valore delle feature molto comune in ML

L’analizzeremo e la motiveremo soprattutto in funzione del Gradient Descent (GD) ma, nelle lezioni successive, vedremo che è utile anche con altri algoritmi di ML

Premessa (1): convergenza

In un algoritmo di ottimizzazione iterativo (come, e.g., il GD), col termine «convergenza» si intende il raggiungimento di uno stato oltre il quale la loss function non diminuisce più in maniera significativa



Premessa (2): sappiamo che...

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

I parametri nella **linear regression** possono essere determinati:

- Iterativamente (GD, SG

1. Inizializzo Θ in maniera random, *t:= 0*
2. Iterazione t-esima:

   a) Per ogni *j= 0, ..., d-1*:

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{n} [(y^{(i)} - h_\theta(\boldsymbol{x}^{(i)}))\boldsymbol{x}_j^{(i)}]$$

   b) If Condizione-di-Terminazione = true

   then return current Θ

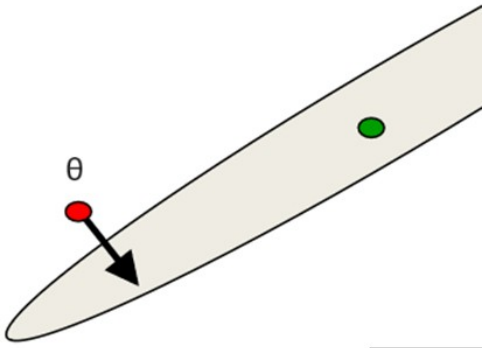
- Analiticamente (NE):

$$\theta = (X^T X)^{-1} X^T \mathbf{y}$$

Premessa (3): il gradiente non punta direttamente al minimo!

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Ricordiamoci che il gradiente, calcolato in un punto specifico (θ) punta nella direzione di massima crescita **locale** di J(θ), ma non «si accorge» di dove è realmente situato il minimo
Detto in termini più precisi, si può dimostrare che il gradiente è sempre perpendicolare alle curve di livello della loss function, come nell'esempio della figura accanto (dove il vettore corrisponde alla *sottrazione* del gradiente)
Come si vede dalla figura, sottrarre il gradiente non significa puntare direttamente verso il minimo locale
Al minimo locale ci si arriverà con una serie di aggiornamenti successivi di θ



Variabili definite su intervalli di valori diversi

Spesso un dataset contiene feature molto variabili in grandezza, unità di misura e intervallo di valori possibili.

Ad esempio, se, nel descrivere una popolazione umana, la feature del peso è espressa in grammi e quella dell'altezza in metri, l'intervallo dei valori che le due feature possono assumere è molto diverso

Variabili definite su intervalli di valori diversi

Per la Normal Equation questo non è un problema poiché la soluzione viene calcolata in maniera analitica e porta sempre al minimo globale della loss function, indipendente dagli intervalli di valori delle feature

In sostanza, nella NE, la soluzione è ottenuta tramite operazioni di moltiplicazione, divisione e somma, la cui complessità computazionale è indipendente dalla grandezza degli operandi(1)

(1) A volte il calcolo della matrice inversa è a sua volta implementato con metodi iterativi. In tal caso la NE non è realmente una soluzione «diretta» e può essere anch'essa rallentata da feature con range di valori diversi. In questo corso di ML ignoreremo questo caso

Variabili definite su intervalli di valori diversi

Con il Gradient Descent avere feature definite su intervalli molto diversi però può però essere (un grosso) problema per la convergenza dell'algoritmo, che può essere rallentata

Vediamo perchè...

Gradient Descent e valori delle feature

Abbiamo sempre detto che il parameter space e il feature space sono concettualmente diversi

Però in pratica è anche vero che, in un modello lineare, c'è una corrispondenza tra il parametro θ_j e la feature x_j ($j > 0$):

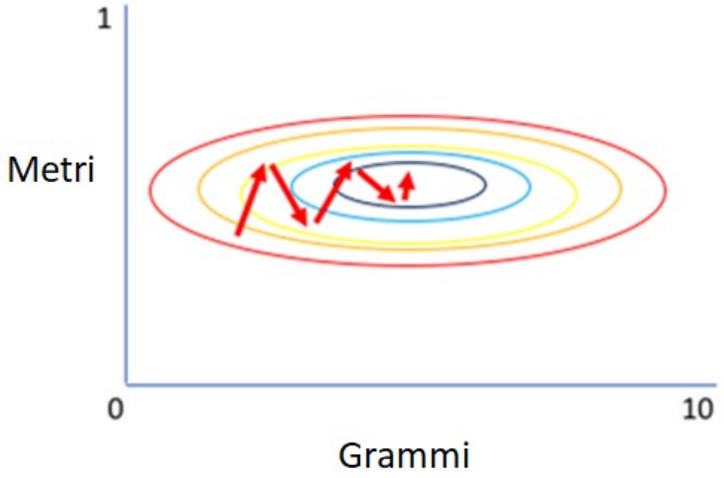
$$\mathbf{x} = [x_0 = 1, x_1, ..., x_{d-1}]$$
$$\theta = [\theta_0, \theta_1, ..., \theta_{d-1}]$$

Gradient Descent e valori delle feature

Sfruttando questa corrispondenza, nella figura accanto i due assi mostrano il parameter space *in cui ogni parametro è indicato usando il nome della feature corrispondente*

Verosimilmente una differenza di pochi grammi produrrà un effetto minimo in J() confrontato con una differenza di pochi metri

Quindi j() avrà una forma ellittica: molto pronunciata rispetto alla feature con un range ampio di valori (grammi),

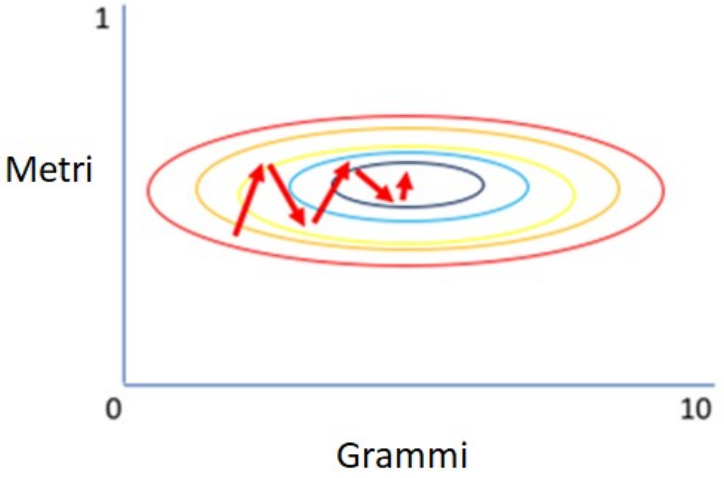


Gradient Descent e valori delle feature

Siccome il gradiente è perpendicolare alle curve di livello di J(), ciò porta all'effetto «oscillante» rappresentato nella figura

Per la precisione: se le feature hanno range di valori diversi, avrò difficoltà a trovare un learning rate (*alpha*) comune ad entrambe che non sia:

- nè troppo piccolo per entrambe
- nè troppo grande per le feature a range più piccolo, cosa che porterebbe ad un effetto oscillatorio e ad aumentare il numero di passi necessari prima di arrivare alla convergenza

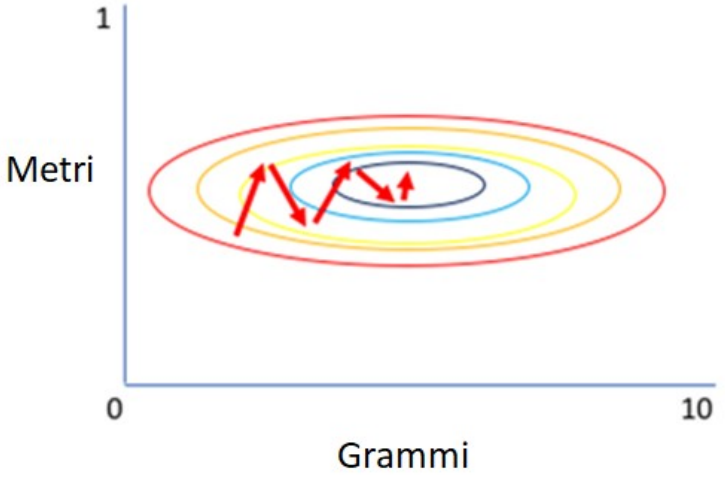


Normalizzazione delle feature

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

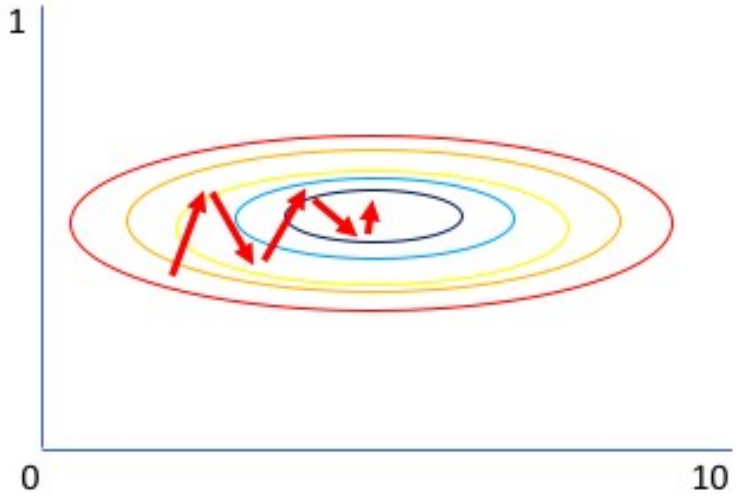
Un modo per alleviare efficacemente questo problema consiste nel pre-processare i dati e ricondurli (più o meno) tutti nello stesso range di valori

Ad esempio potrei trasformare ogni feature in modo che assuma valori in [0, 1] oppure in [-1, 1], ecc…

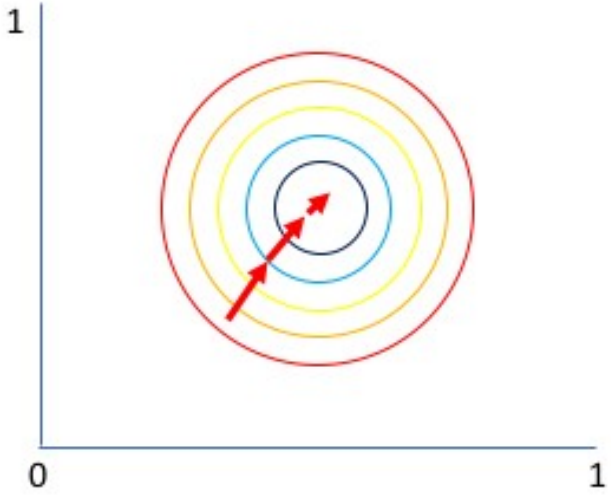


Esempio

Prima:



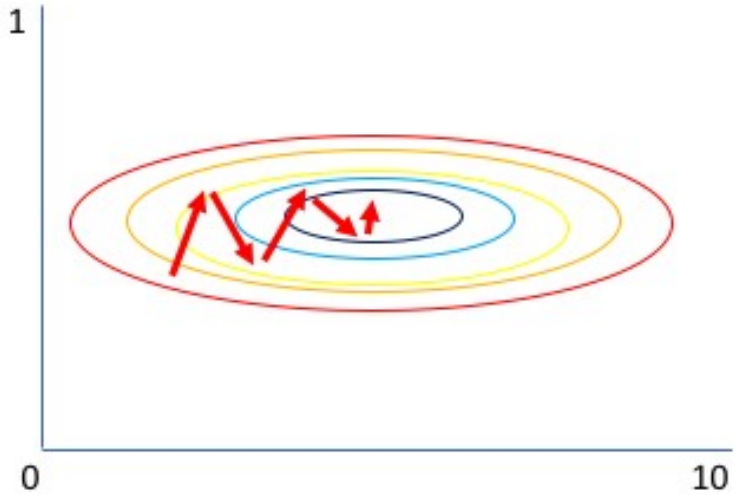
Dopo:



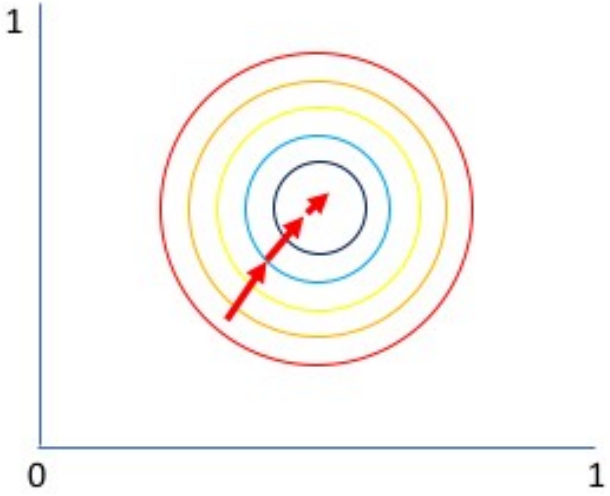
Se le feature sono normalizzate => J() ha una forma più sferica => i (negativi dei) gradienti, perpendicolari alle curve di livello, puntano più direttamente verso il minimo => dovrò fare meno iterazioni prima di giungere a convergenza

Esempio

Prima:



Dopo:



Esistono diversi metodi di scalamento che corrispondono ad una trasformazione delle feature, che deve essere fatta prima dell'applicazione del GD. Vediamo i più comuni

Feature transformations

- La **standardization** ridistribuisce i feature values in modo tale da avere un vettore delle medie (risultante *dopo* la trasformazione) **μ' = 0** e un vettore delle deviazioni standard **σ' = 1**.
  - Per far ciò, per ogni feature *j* (e *indipendentemente dalle altre feature*), si calcola la sua media (scalare) μ e la sua deviazione standard (scalare) σ
  - Dopodichè, sostituisco ogni feature value *x* della *j*-esima feature con *x'*:

$$x' = \frac{x - \mu}{\sigma}$$

Feature transformations

- Analogamente alla Standardization, altre tecniche usate sono:
- La **Mean Normalization** ridistribuisce i valori delle feature tra **-1** e **1** con **μ'=0**.

$$x' = \frac{x-\mu}{max(x)-min(x)}$$

- Il **Min-Max Scaling** porta i valori delle feature tra **0** e **1**.

$$x' = \frac{x-min(x)}{max(x)-min(x)}$$

Tra le varie tecniche, la Standardization è quella più comunemente usata

Inizializzazione dei parametri nel GD

Un altro vantaggio di avere feature tutte definite in un range di valori predefinito e uguale per tutte (e.g., con media **μ' = 0** e deviazione standard **σ' =1**) è che, nel GD, posso inizializzare i pesi utilizzando lo stesso intervallo

Ese.: `theta = np.random.randn(d)`

Fasi della normalizzazione

Attenzione: così come i parametri della funzione ipotesi devono essere calcolati con i soli dati di training, anche per calcolare i parametri della feature transform usata per scalare i dati *posso usare solo i dati di training* e non quelli di testing

Ad esempio, per la Standardizzazione, i parametri $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_{d-1}]$ e $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_{d-1}]$ devono essere calcolati usando solo la Design matrix X, e non i dati di testing

Questo perché i dati di testing devono simulare il comportamento del mio algoritmo di ML quando processerà nuovi samples (e.g., nella fase d'uso del regressore) e questi dati nuovi non sono disponibili quando decido il valore dei parametri da usare per lo

Fasi della normalizzazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

A inference time userò i parametri della feature transform calcolati in fase di training

In particolare, tali parametri vanno calcolati prima del training vero e proprio, per cui questa fase di pre-processing è preliminare al training ed avviene subito dopo l'EDA

Generalizzazione

Ragionamenti analoghi possono essere fatti nel caso della classificazione e/o usando funzioni ipotesi non lineari

In generale, se si usa il GD per ottimizzare i parametri di un modello parametrico, è sempre consigliabile operare uno scalamento delle feature

Vedremo in seguito che la normalizzazione delle feature è utile anche con algoritmi diversi dal GD (e.g., il Nearest Neighbor, ecc.)

Variabili categoriche

Feature categoriche

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Age	Employment Status	Job Category	Home Country	Job Salary
40	1	4	USA	LOW
35	0	6	ITALY	VERY LOW
50	1	3	ITALY	MODERATE
51	1	2	GERMANY	HIGH
22	1	2	FRANCE	HIGH

Quando ho delle feature categoriche, ovvero i cui valori non sono numeri, devo trasformarle in numeriche per poter usare la linear regression (e la gran maggioranza degli algoritmi di ML)
Il modo più semplice per farlo è associare un numero (ad ese., un intero) ad ogni possibile valore della feature
Ad ese., per Home Country, potrei associare:

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

Feature categoriche

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Age	Employment Status	Job Category	Home Country	Job Salary
40	1	4	USA	LOW
35	0	6	ITALY	VERY LOW
50	1	3	ITALY	MODERATE
51	1	2	GERMANY	HIGH
22	1	2	FRANCE	HIGH

Stessa cosa per Job Salary:

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

Feature categoriche

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Age	Employment Status	Job Category	Home Country	Job Salary
40	1	4	USA	LOW
35	0	6	ITALY	VERY LOW
50	1	3	ITALY	MODERATE
51	1	2	GERMANY	HIGH
22	1	2	FRANCE	HIGH

- Ho trasformato l'insieme di valori di ogni feature categorica in un insieme numerico (discreto)
- Questa trasformazione ha in sé un problema: il fatto di rendere «ordinato» un insieme di valori che in partenza non lo era
- Se per Job Salary non è un grosso problema (perché le categorie iniziali indicano già una quantità), per Home Country introduce un ordinamento arbitrario
- Tuttavia, solitamente questa è una soluzione accettabile
- Vedremo più in là che le feature categoriche in alcuni casi possono conservare la loro natura categorica (cioè «non ordinata»)
- Se invece ad essere categorica è la variabile dipendente, allora meglio usare un approccio di classificazione piuttosto che di regressione

Feature categoriche

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Age	Employment Status	Job Category	Home Country	Job Salary
40	1	4	USA	LOW
35	0	6	ITALY	VERY LOW
50	1	3	ITALY	MODERATE
51	1	2	GERMANY	HIGH
22	1	2	FRANCE	HIGH
35	0	2	?	?

- Durante l' EDA, se mancano dei valori, piuttosto che sostituirli con la media, meglio sostituirli con la "moda", cioè col valore (categorico/discreto) più frequente
- Nell'esempio di sopra, userei, rispettivamente, ITALY e HIGH

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.4]
- **Pattern Classification, second edition,** R. O. Duda, P. E. Hart, D. G. Stork, Wiley-Interscience, 2000
- **The elements of statistical learning**, Trevor Hastie, Robert Tibshirani, Jerome H. Friedman, New York: Springer series in statistics, 2001 [cap.3]
- **An introduction to statistical learning**, Gareth M. James, Daniela Witten, Trevor Hastie, Robert Tibshirani, New York: Springer, 2013 [cap. 3, 6]