IN550 Machine Learning

Preprocessamento di feature e dati
Ingegnerizzazione delle Feature

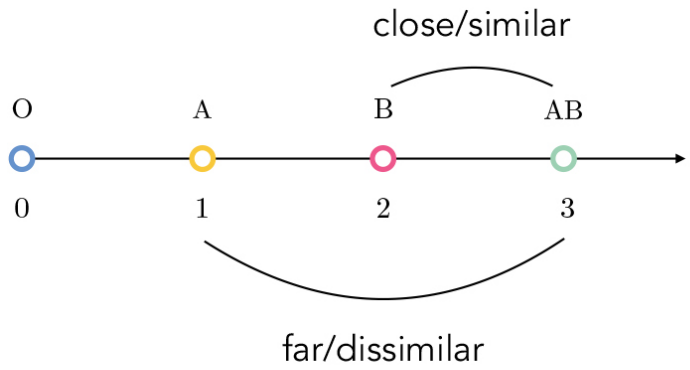
Vincenzo Bonifaci

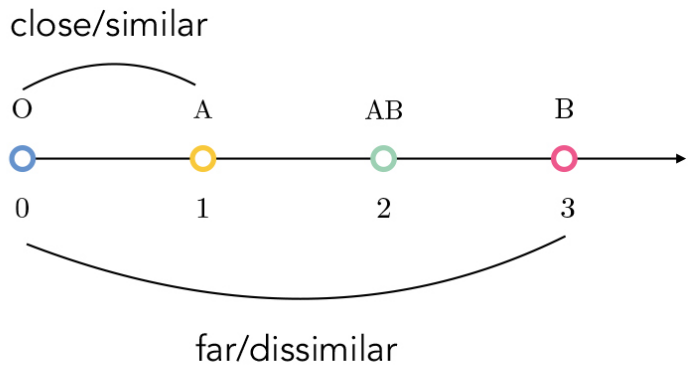
Preprocessamento di feature e dati

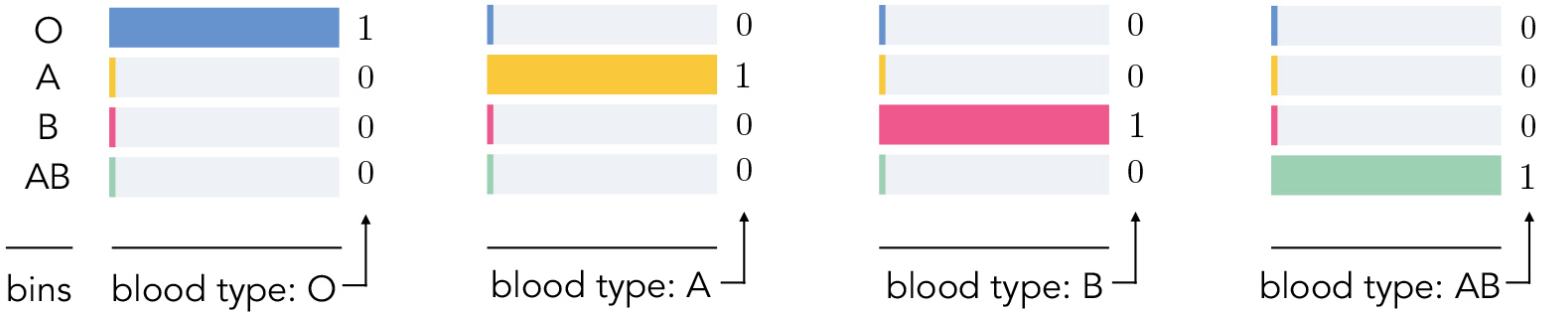
- Operazioni comuni di preprocessamento:
  - Creazione di feature basate su istogrammi
  - Scalatura/standardizzazione di feature
  - Discretizzazione (binning) di feature
  - Imputazione dati
  - Normalizzazione degli esempi

Istogrammi per dati categorici

Esempio: gruppo sanguigno (0, A, B, AB)
Codifica *ordinale* vs. istogramma *One-Hot Encoding*





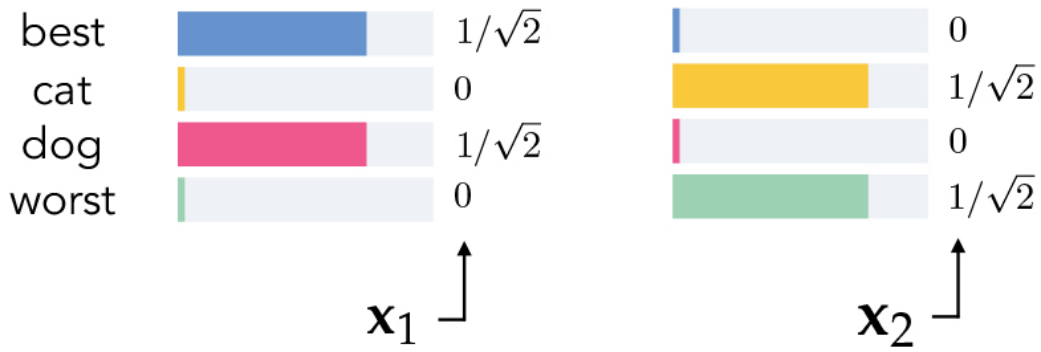


Istogrammi per dati testuali

Istogramma *Bag of Words (BoW)*
Basato sul conteggio delle parole nel testo

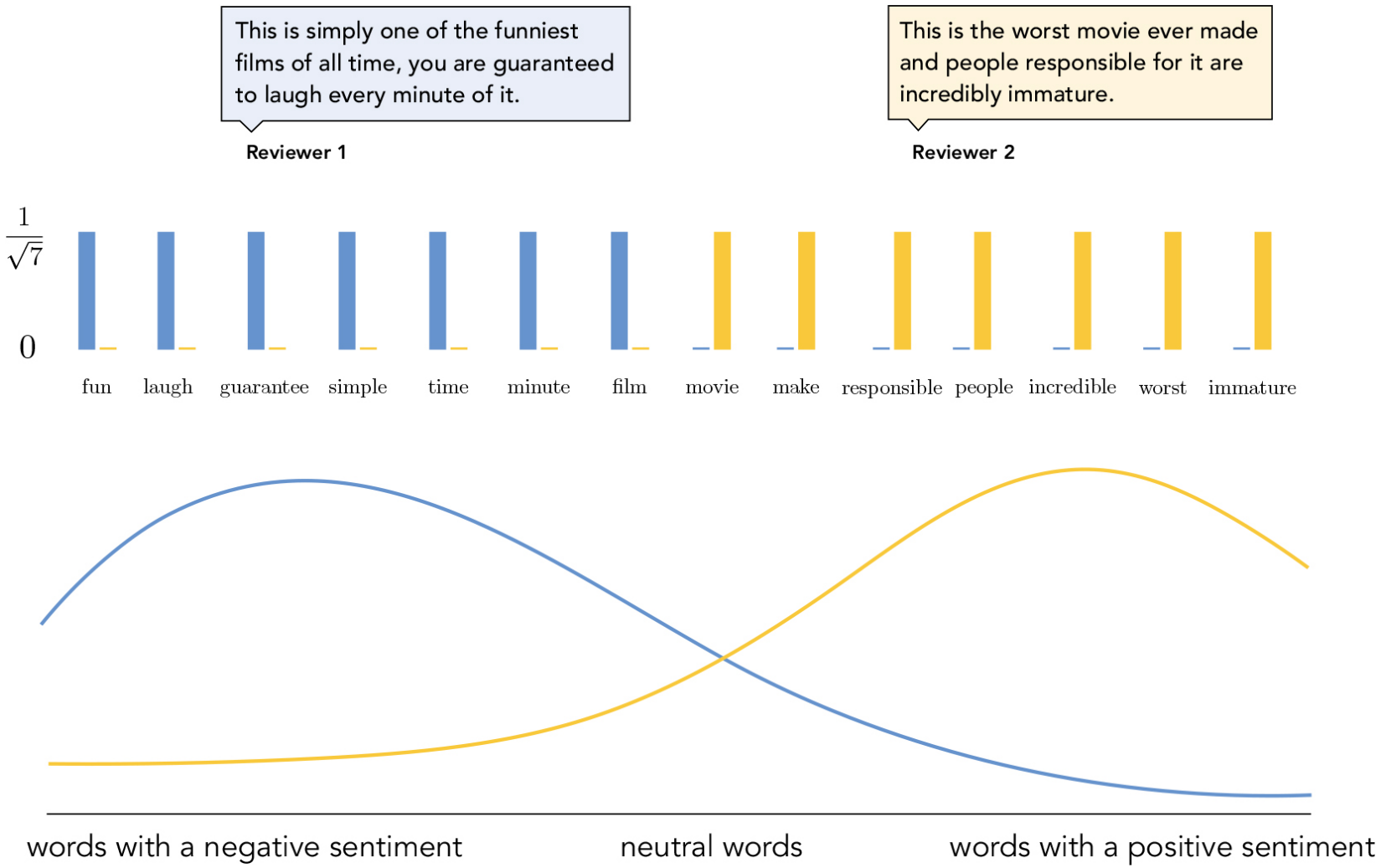
Esempio:

1. `dogs are the best`
2. `cats are the worst`



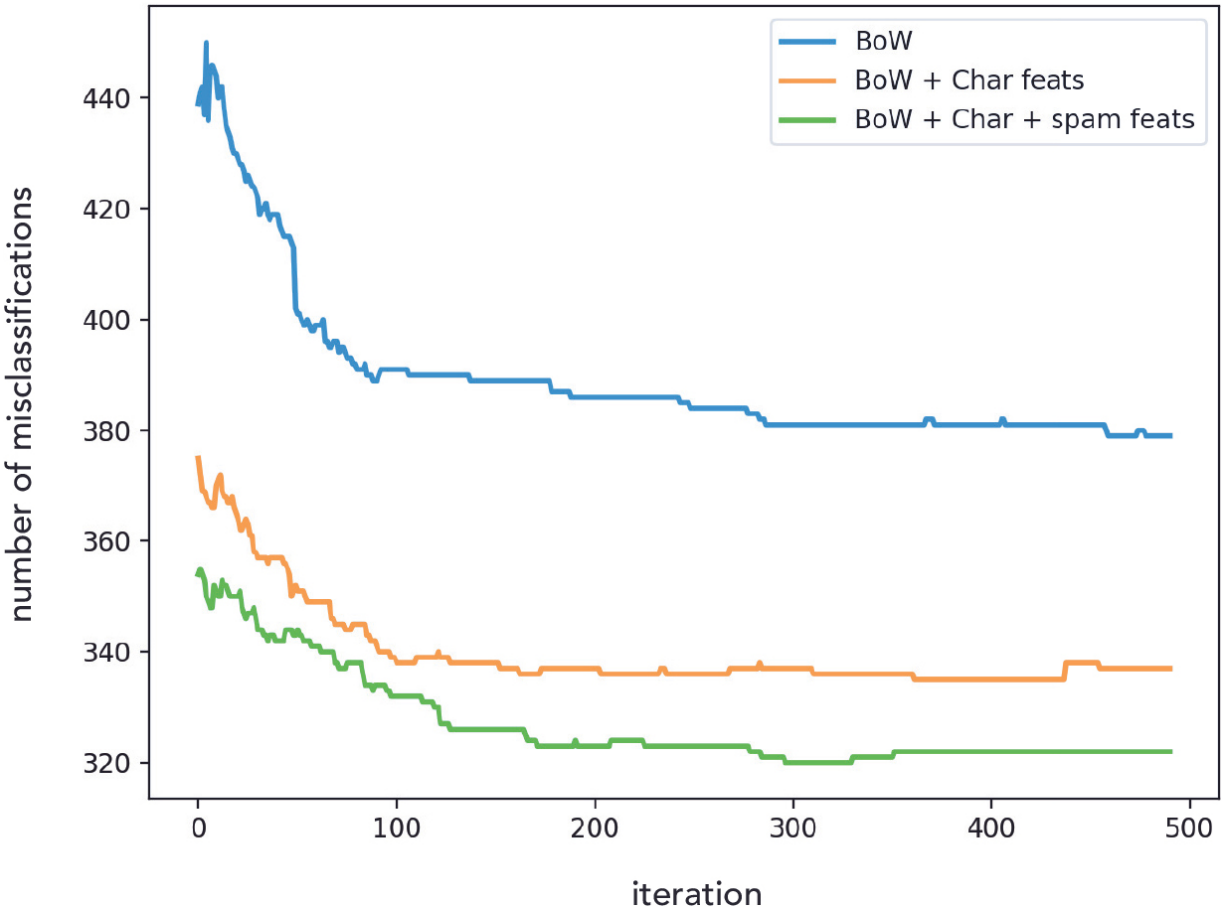
In questa figura gli esempi sono stati anche normalizzati in modo da avere norma ℓ_2 unitaria (vedi Normalizzazione più avanti)

Istogrammi per dati testuali

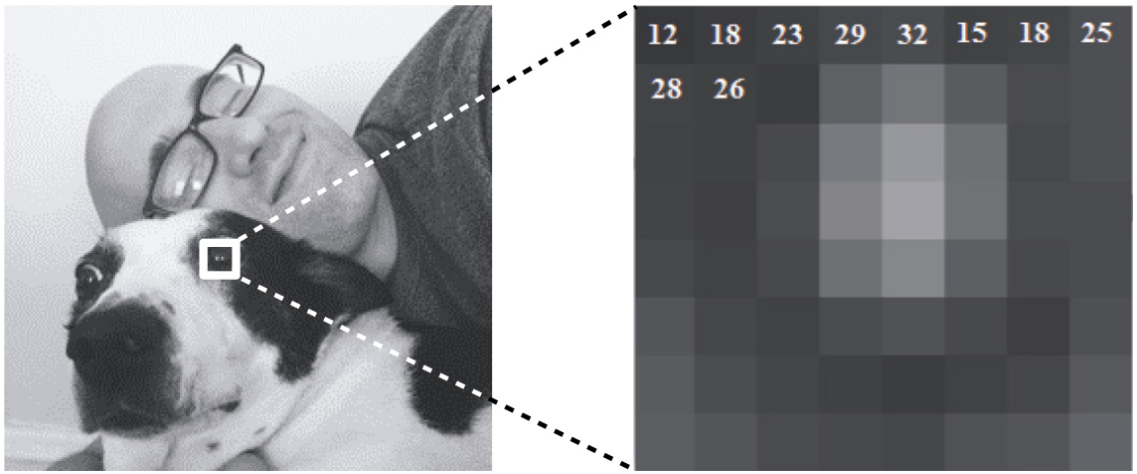


Istogrammi per dati testuali

Performance per un classificatore di spam basato su BoW

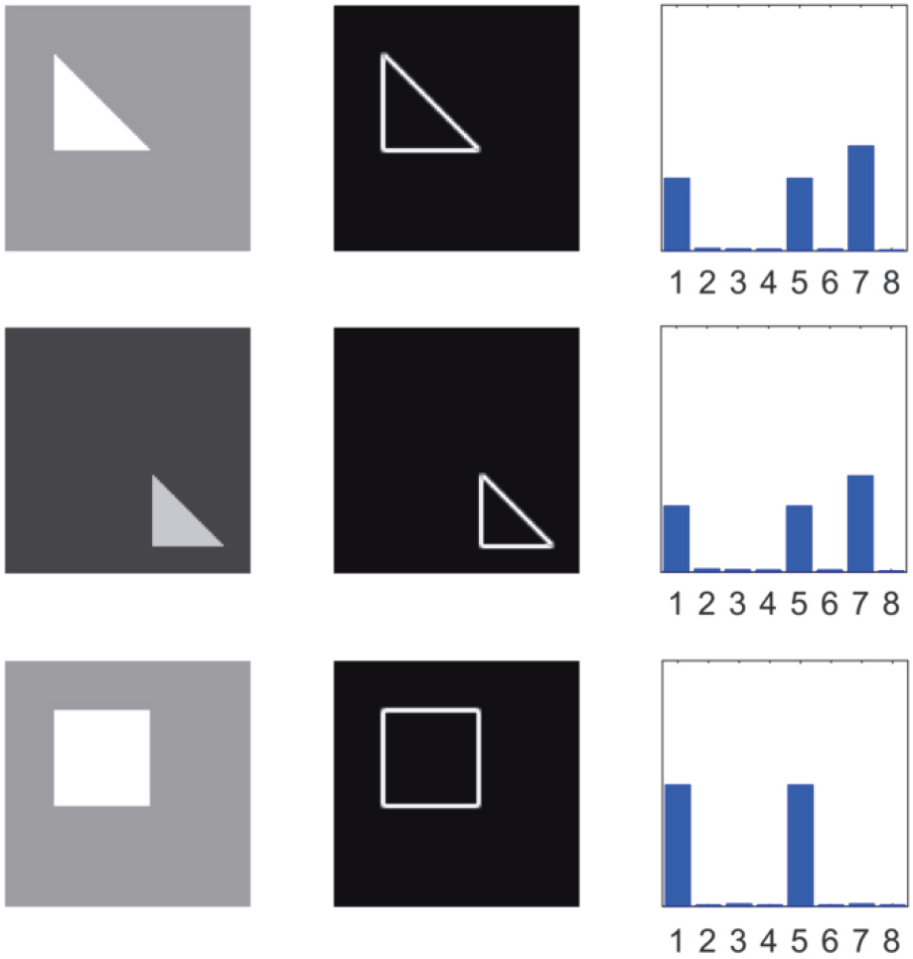


Istogrammi per immagini



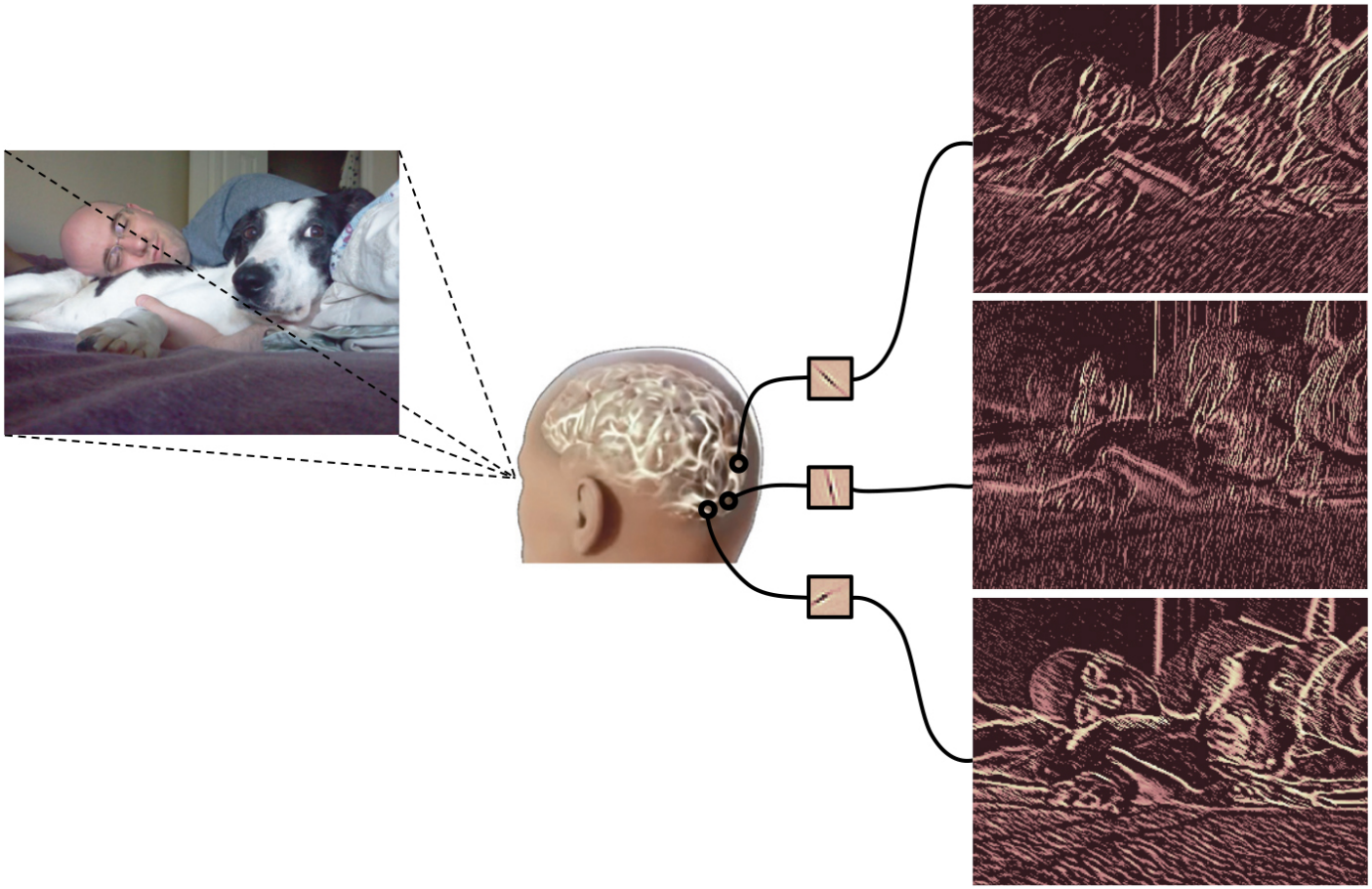
Istogrammi per immagini

Istogrammi basati sui bordi [edge] in varie direzioni



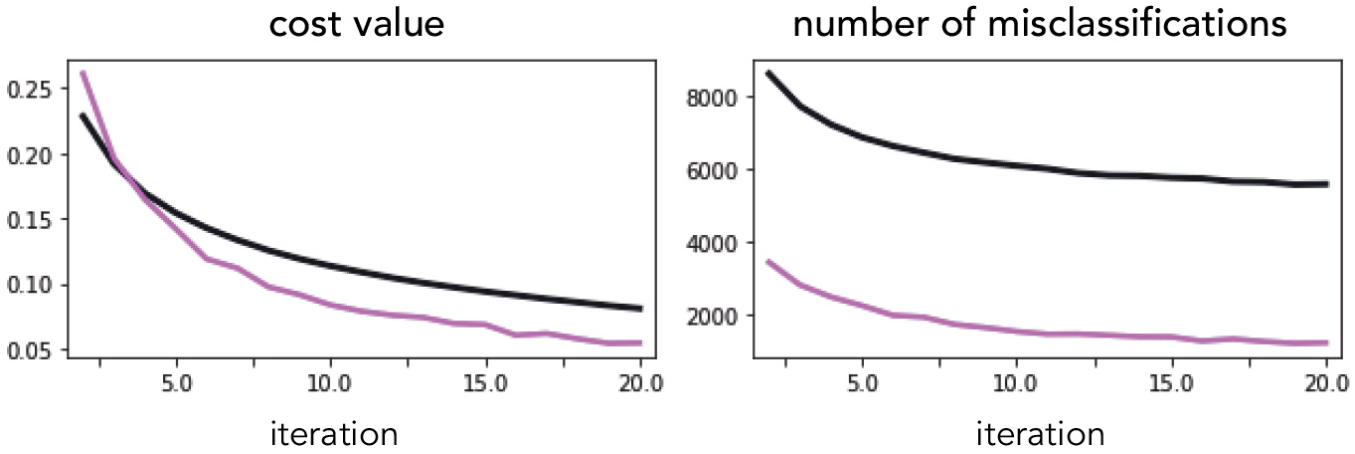
Istogrammi per immagini

Similitudine con elaborazione visiva nel cervello umano



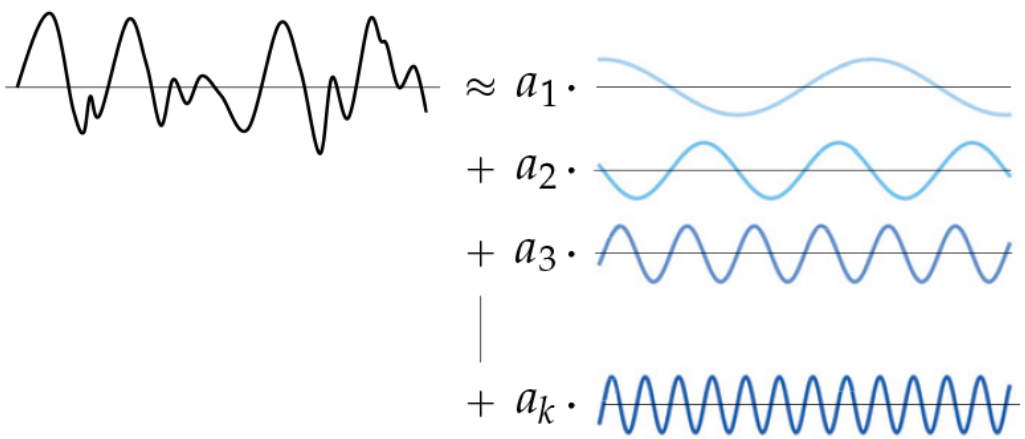
Istogrammi per immagini

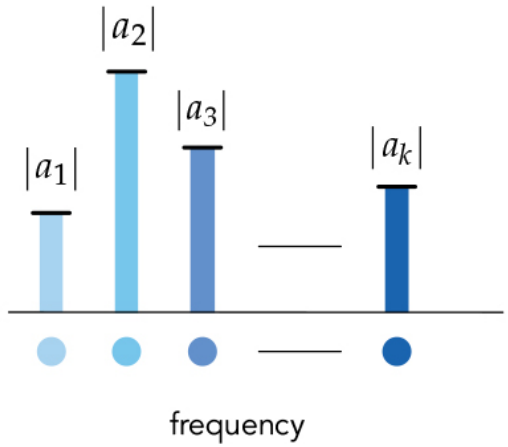
Performance senza e con dati dei bordi nel dataset MNIST
(regressione logistica multinomiale, mini-batch SGD con $\eta = 0.01$)



Istogrammi per dati audio

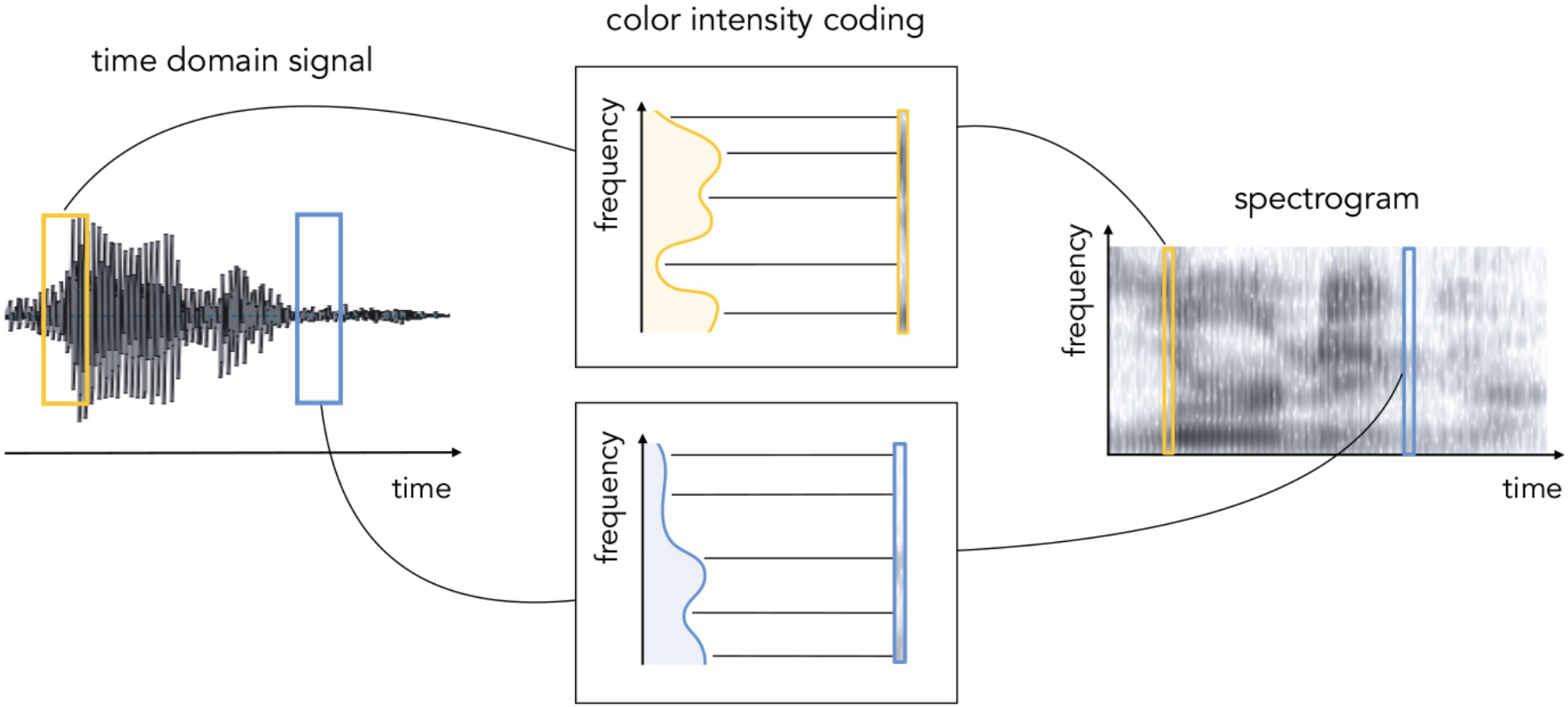
Decomposizione in base di Fourier (frequenze fondamentali)





Istogrammi per dati audio

Spettrogramma



Usati ad es. per identificare brani musicali da servizi come Shazam e AcoustID

Scalatura di una feature

Una comune operazione di preprocessamento è *scalare* una variabile dividendola per il suo massimo valore possibile

Esempio: l'intensità di un pixel di un'immagine è un intero tra 0 e 255
$x_j \leftarrow x_j/255$ in modo che il nuovo x_j sia in $[0, 1]$

Scalatura min-max della feature x_j

1. $x_j \leftarrow \frac{x_j - m_j}{M_j - m_j}$ dove m_j e M_j sono rispettivamente il minimo e il massimo di x_j

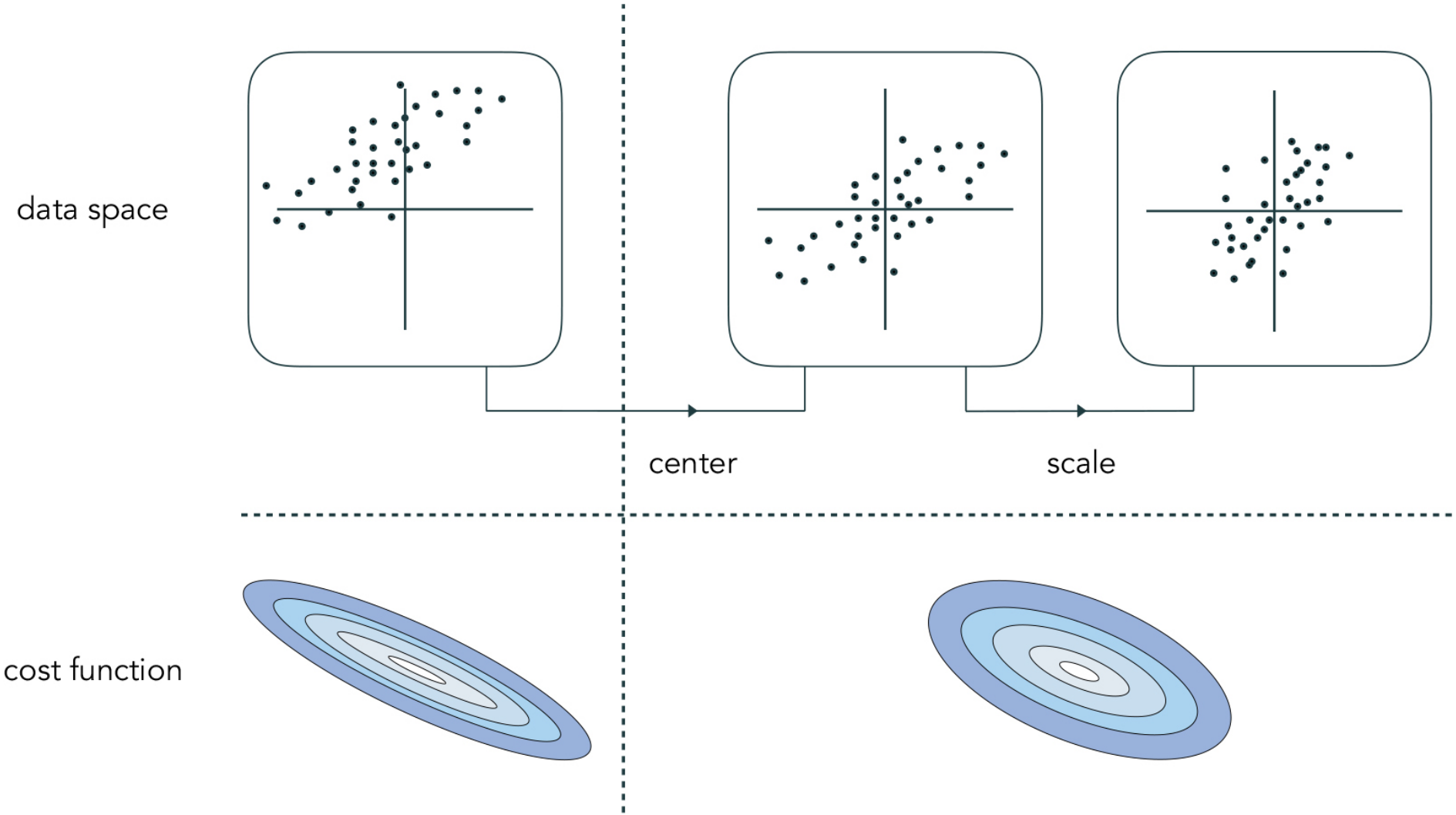
Standardizzazione di una feature

Un'operazione alternativa è *standardizzare* una feature

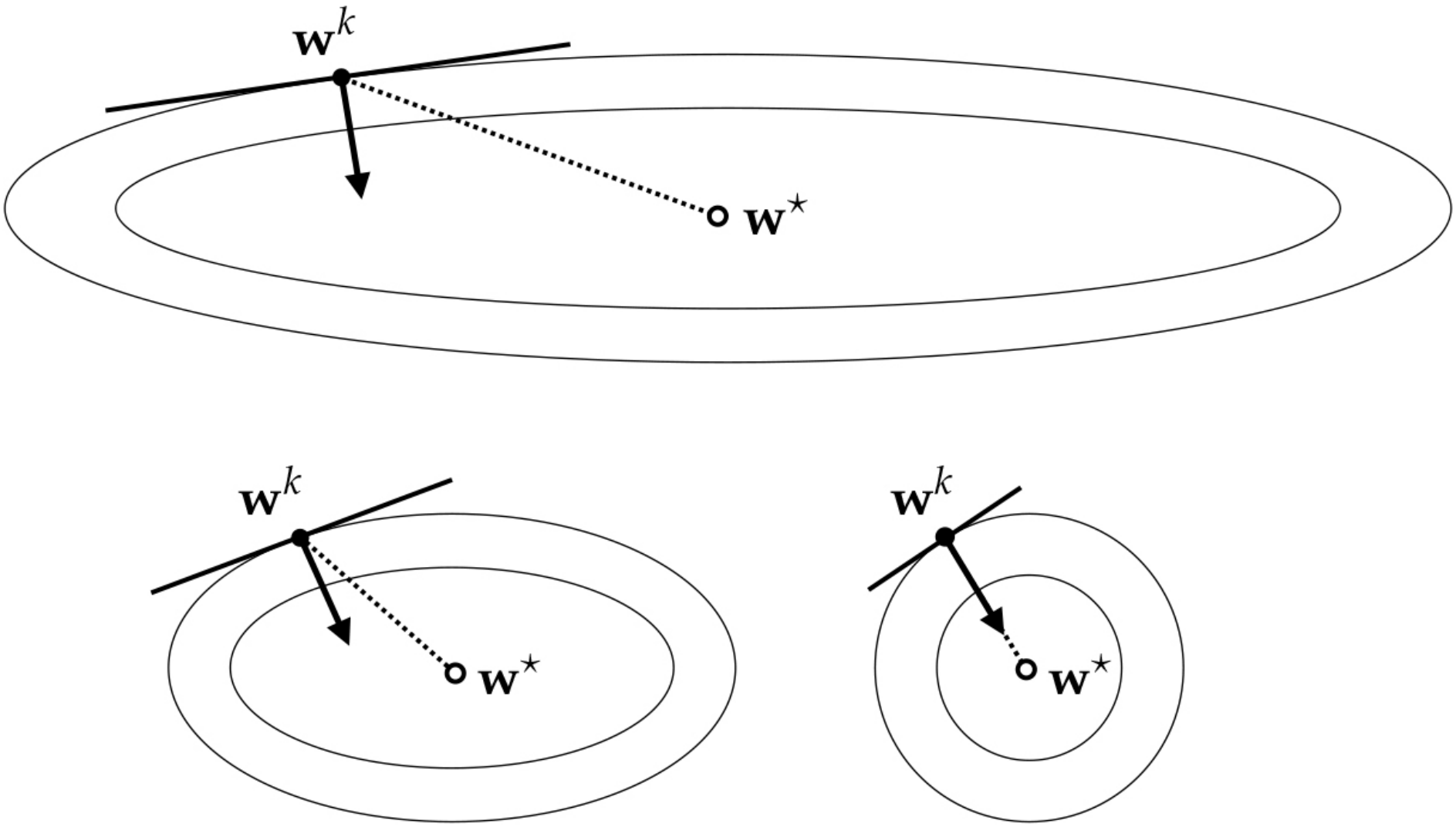
Standardizzazione della feature x_j

1. *Centratura*: $x_j \leftarrow x_j - \mu_j$ dove μ_j è la media della variabile x_j
2. *Scalatura*: $x_j \leftarrow x_j / \sigma_j$ dove $\sigma_j^2 = (1/m) \sum_i x_j^2$ è la varianza della variabile x_j (σ_j è la sua deviazione standard)

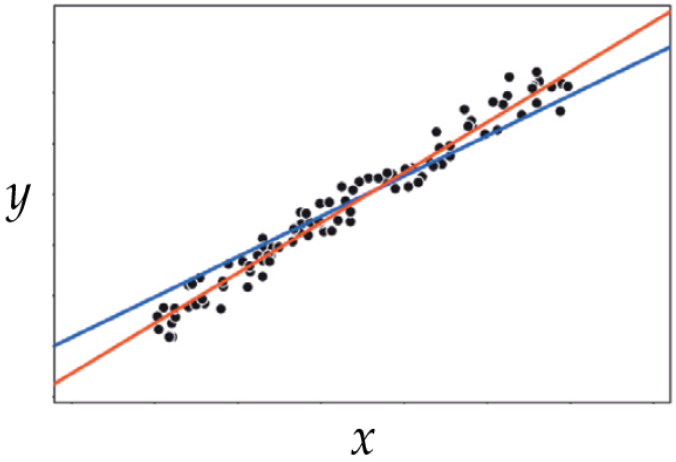
Effetti della standardizzazione

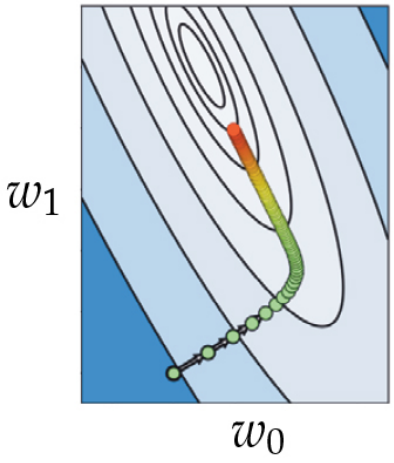


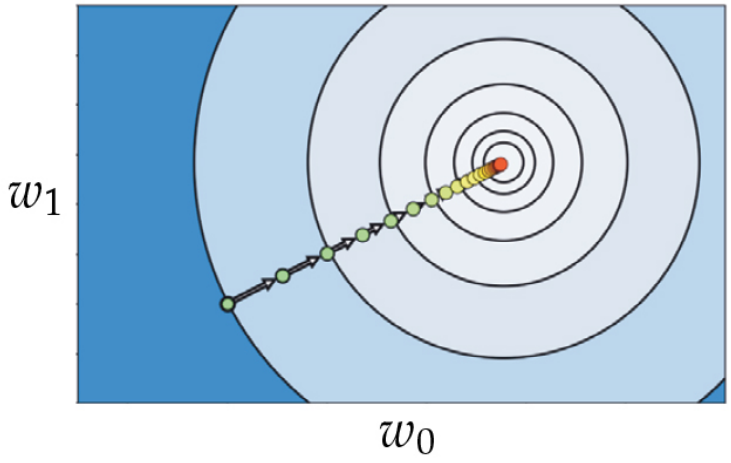
Effetti della standardizzazione

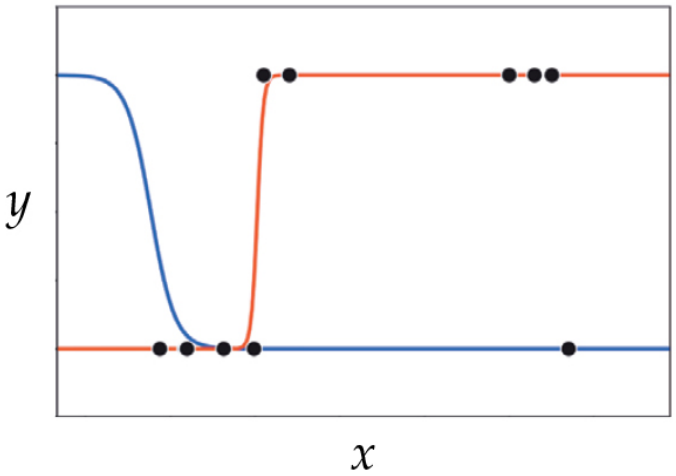


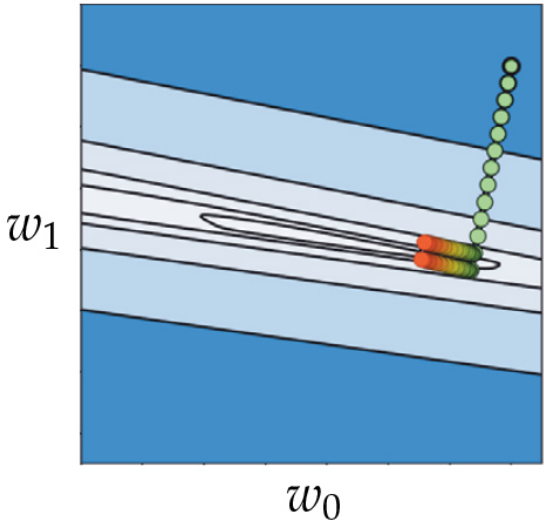
Effetti della standardizzazione

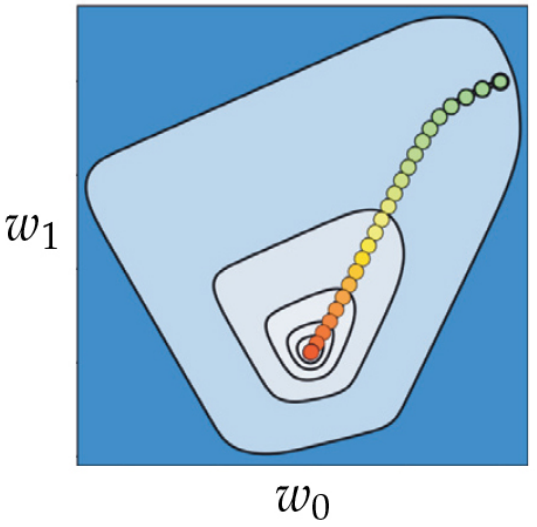












Discretizzazione (quantizzazione) di dati numerici

A volte è utile convertire una variabile numerica in variabile ordinale o categorica:

- Può aumentare l'espressività del modello (trasformazione non-lineare)
- Mantiene l'interpretabilità

Si usano intervalli (detti *bin*) specificati manualmente o impostandone il numero K (`n_bins` in `scikit-learn`) e calcolandoli dai dati

Esempio con $K = 3$:

$$X = \begin{bmatrix} 0 \\ -3 \\ 6 \end{bmatrix} \quad \rightarrow \quad X_{\text{ord}} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} \text{ oppure } X_{\text{one-hot}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

I bin calcolati in questo caso sono: $[-\infty, -1)$, $[-1, 2)$, $[2, +\infty]$

Come trasformare separatamente le feature?

Colonne (feature) diverse possono richiedere trasformazioni diverse

City	Title	Expert rating	User rating
London	His Last Bow	5	4
London	How Watson Learned the Trick	3	5
Paris	A Moveable Feast	4	4
...	...	...	...

Possiamo usare `ColumnTransformer()` per separare le trasformazioni:

```
column_trans = ColumnTransformer(
    [('nome1', OneHotEncoder(), ['city']),
    ('nome2', CountVectorizer(), 'title')],
    remainder=MinMaxScaler()
)
column_trans.fit_transform(X)
```

Dati mancanti (imputazione dati)

Dataset reali spesso contengono esempi con dati mancanti

Il processo di completamento di dati mancanti è detto *imputazione*

Strategie più comuni per l'imputazione:

- Variabili numeriche: usare la media degli altri esempi (`strategy='mean'` in `scikit-learn`)
- Variabili categoriche: usare la moda degli altri esempi (`strategy='most_frequent'` in `scikit-learn`)

In alternativa, si devono scartare le feature o gli esempi che presentano dati mancanti (in quanto quasi nessun modello li supporta)

Normalizzazione degli esempi

Un altro preprocessamento comune consiste nel *normalizzare* gli esempi:

Normalizzazione

$x^{(i)} \leftarrow \frac{x^{(i)}}{\left\|x^{(i)}\right\|}$ dove $\|\cdot\|$ è una norma, per $i = 1, 2, \ldots, m$

Si noti che è una trasformazione applicata agli esempi, non a una feature

Esempio di normalizzazione in norma ℓ_1:

$$X = \begin{bmatrix} 0 & 1 \\ -3 & 0 \\ 6 & 1 \end{bmatrix} \quad \rightarrow \quad X_{\text{norm}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \\ 6/7 & 1/7 \end{bmatrix}$$

Concatenazione di trasformazioni e/o del modello

I vari passi di trasformazione e il modello andranno concatenati a cascata

In `scikit-learn`, la prassi migliore è esplicitare la concatenazione attraverso l'interfaccia `Pipeline`

Ad es., per standardizzare le feature e applicare una regressione logistica regolarizzata:

```
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('reglogistica', LogisticRegression(penalty='l2')),
])
pipe.fit(X_train, y_train)
```

Preprocessamento di feature in `scikit-learn`

Moduli: `sklearn.preprocessing`, `sklearn.feature_extraction.text`

	Interfaccia `scikit-learn`
Codifica ordinale	`OrdinalEncoder()`
Codifica one-hot	`OneHotEncoder()`
Bag of words	`CountVectorizer()`
Scalatura min-max	`MinMaxScaler()`
Standardizzazione	`StandardScaler()`
Discretizzazione	`KBinsDiscretizer(n_bins)`
Trasformazioni separate	`ColumnTransformer()`

Altri tipi di preprocessamento in `scikit-learn`

Moduli: `sklearn.imputer`, `sklearn.preprocessing`, `sklearn.pipeline`

	Interfaccia `scikit-learn`
Imputazione	`SimpleImputer(strategy)`
Normalizzazione	`Normalizer(norm)`
Concatenazione	`Pipeline()`