Preprocessing

Corso di AA, anno 2018/19, Padova

Fabio Aiolli

28 Novembre 2018

Pipeline di Apprendimento Supervisionato

- Analisi del problema
- Raccolta, analisi e pulizia dei dati
- Preprocessing e valori mancanti
- Studio delle correlazioni tra variabili
- Feature Selection/Weighting/Learning
- Scelta del predittore e Model Selection
- Test

Oggetti

- Vettori: p.e. valori di pressione del sangue, battito cardiaco a riposo, altezza peso di una persona, utili ad una società assicurativa per determinare la sua speranza di vita
- Stringhe: p.e. le parole di un documento testuale in NER, o la struttura del DNA
- Insiemi e Bag: p.e. l'insieme dei termini in un documento, o magari anche la loro frequenza?
- Tensori: p.e. Immagini (2D) e Video (3D)
- Alberi e grafi: p.e. la struttura di un documento XML, o di una molecola in chimica
- ...
- Strutture composte: p.e. una pagina web può contenere immagini, testo, video, tabelle, ecc.

Oggetti

- Feature categoriche o simboliche
  - Nominali [Nessun ordine]
    p.e. per un auto: paese di origine, fabbrica, anno di uscita in commercio, colore, tipo, ecc.
  - Ordinali [Non preservano distanze]
    p.e. gradi militari dell'esercito: soldato, caporale, sergente, maresciallo, tenente, capitano)
- Feature quantitative o numeriche
  - Intervalli [Enumerabili]
    p.e. livello di apprezzamento di un prodotto da 0 a 10
  - Ratio [Reali]
    p.e. il peso di una persona

Codifica variabili categoriche o simboliche

OneHot Encoding

Le variabili categoriche si possono rappresentare in un vettore con tante componenti quanti sono i possibili valori della variabile.

Es. Possibili valori delle variabili:

- Marca: `Fiat [c0], Toyota [c1], Ford[c2]`
- Colore: `Bianca [c3], Nera [c4], Rossa [c5]`
- Tipo: `Economica [c6], Sportiva [c7]`

```
(Toyota, Rossa, Economica) -> [0,1,0,0,0,1,1,0]
```

Codifica OneHot in sklearn

La codifica OneHot si può ottenere facilmente in sklearn usando i metodi fit e transform della classe OneHotEncoder nel modo seguente:

```
>>> enc = preprocessing.OneHotEncoder()
>>> enc.fit([[0,0,0], [1,1,1], [2,2,1]])
OneHotEncoder(categorical_features='all', dtype=<...
'numpy.float64'>,handle_unknown='error', n_values='auto',
parse=True)
>>> enc.transform([[1,2,0]]).toarray()
array([[ 0., 1., 0., 0., 0., 1., 1., 0.]])
```

Nota che la codifica del valore della variabile categorica utilizza una rappresentazione intera $0 \leq v \leq m-1$ dove m è il numero di possibili valori della variabile

Codifica delle variabili continue

In questo caso è decisamente più difficoltoso trovare un buon mapping

Tipicamente, le feature vengono trasformate per ottenere valori "comparabili" con le altre feature

- Standardization (centering e/o variance scaling)
- Scaling in un range
- Normalizzazione

Sia $\hat{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$ e $\sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \hat{x}_j)^2}$

- Centering: $c(x_{ij}) = x_{ij} - \hat{x}_j$
- Standardizzazione: $s(x_{ij}) = \frac{c(x_{ij})}{\sigma_j}$
- Scaling: $h(x_{ij}) = \frac{x_{ij} - x_{min,j}}{x_{max,j} - x_{min,j}}$
- Normalizzazione: $g(\mathbf{x}) = \frac{\mathbf{x}}{||\mathbf{x}||}$

Preprocessing variabili continue in sklearn: StandardScaler

```
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> scaler
StandardScaler(copy=True, with_mean=True, with_std=True)
>>> scaler.mean_
array([ 1.  ..., 0.  ..., 0.33...])
>>> scaler.scale_
array([ 0.81..., 0.81..., 1.24...])
>>> scaler.transform(X_train)
array([[ 0.  ..., -1.22..., 1.33...],
 [ 1.22..., 0.  ..., -0.26...],
 [-1.22..., 1.22..., -1.06...]])
```

Preprocessing variabili continue in sklearn: MinMaxScaler

```
>>> min_max_scaler = preprocessing.MinMaxScaler()
>>> X_train_minmax = min_max_scaler.fit_transform(X_train)
>>> X_train_minmax
array([[ 0.5 , 0.  , 1.  ],
 [ 1.  , 0.5 , 0.33333333],
 [ 0.  , 1.  , 0.  ]])
array([ 0.81..., 0.81..., 1.24...])
>>> scaler.transform(X_train)
array([[ 0.  ..., -1.22..., 1.33...],
 [ 1.22..., 0.  ..., -0.26...],
 [-1.22..., 1.22..., -1.06...]])
```

Preprocessing variabili continue in sklearn: Normalizer

Il prodotto scalare (e le distanze) tra vettori sono influenzati dalla norma degli stessi, oltre che dall'angolo tra i vettori: $\langle \mathbf{x}, \mathbf{z} \rangle = ||\mathbf{x}|| \cdot ||\mathbf{z}|| \cdot \cos(\theta)$. Quando vogliamo considerare SOLO l'angolo formato tra i vettori possiamo ricorrere alla normalizzazione.

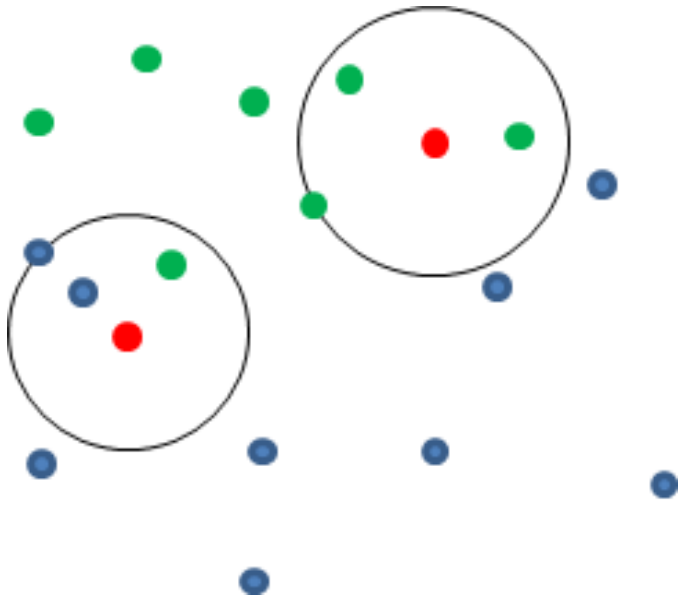
```
>>> normalizer = preprocessing.Normalizer().fit()
>>> normalizer
Normalizer(copy=True, norm='l2')
>>> normalizer.transform(X)
array([[ 0.40..., -0.40..., 0.81...],
 [ 1.  ..., 0.  ..., 0.  ...],
 [ 0.  ..., 0.70..., -0.70...]])
```

Preprocessing variabili continue in `sklearn`: Altre opzioni

- `Binarizer` per la binarizzazione delle features (basata su soglia)
- `Imputer` per la imputation di valori mancanti (basati su media, mediana, valore più frequente nella riga o nella colonna dello stesso)
- `FunctionTransformer` per la definizione custom del preprocessing

K-Nearest Neighbors

K-Nearest Neighbors è un semplice algoritmo di classificazione in cui un esempio di test è classificato come la classe di maggioranza dei suoi k-vicini nel training set

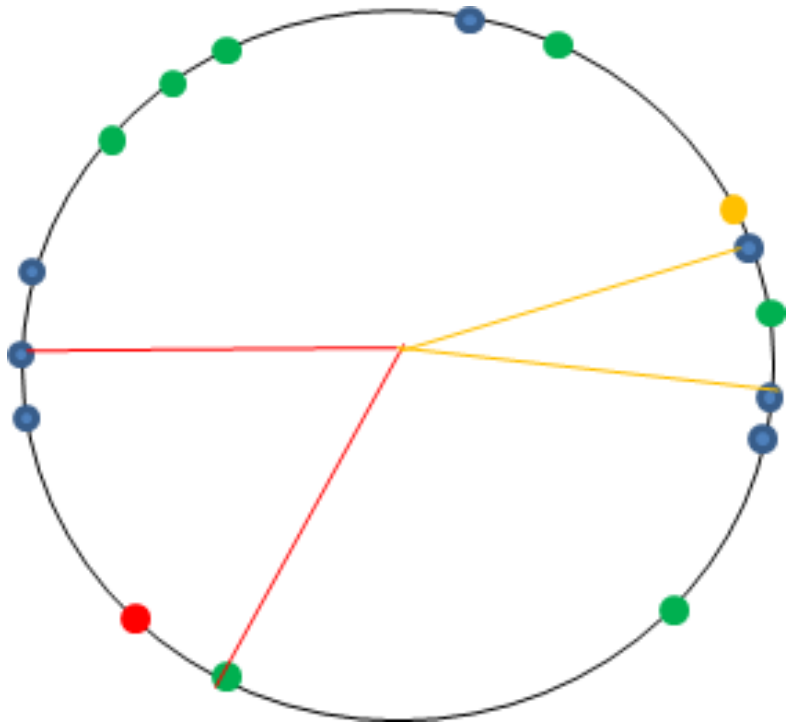


Esempio 3-NN in 2D usando la distanza Euclidea

K-Nearest Neighbors con Normalizzazione esempi

K-Nearest Neighbors quando gli esempi stanno tutti in una palla di raggio unitario. La distanza diventa equivalente al prodotto scalare.
Infatti,

$$||\mathbf{x} - \mathbf{z}||^2 = ||\mathbf{x}||^2 + ||\mathbf{z}||^2 - 2\langle \mathbf{x}, \mathbf{z} \rangle = \text{const} - 2\langle \mathbf{x}, \mathbf{z} \rangle$$



Esempio 3-NN in 2D con norma degli esempi unitaria

Ricapitolando

Abbiamo visto:

- Codifica delle variabili categoriche
- Codifica delle variabili continue
- Preprocessing

Esperienze:

- Ragionare su come potrebbero essere rappresentate variabili di tipo intervallo. È ragionevole rappresentarle come variabili numeriche qualsiasi o è meglio avere una codifica ad-hoc?
- Provare a preprocessare il dataset TITANIC disponibile su kaggle. Fissato un modello predittivo (per esempio SVM, k-nn, ecc.) verificare se ci sono variazioni significative nell'accuracy in predizione.