Journal, Vol. XXI, No. 1, 1-5, 2013
Additional note

Progetto di Metodi statistici per l'apprendimento: K-Nearest Neighbors

Studente: Marco Odore - **Matricola:** 868906

Abstract
Implementzione in R e C++ dell'algoritmo K-nearest neighbors e sua applicazione ad un problema di classificazione binaria, nella sua versione classica e online.

Keywords
Apprendimento Supervisionato — K-NN — Classificazione binaria

Contents

Introduzione

Si è scelto di implementare in R l'algoritmo di apprendimento supervisionato K-NN e di applicarlo ad un problema di classificazione binario, dove si vuole associare ad una persona il suo possibile reddito annuale (discretizzato, tramite una soglia, a due valori possibili), date alcune sue informazioni di base.

Il dataset

Il dataset utilizzato [1] contiene 48842 istanze, ognuna delle quali è caratterizzata da 14 feature:

- **Età** - Tipo continuo
- **Workclass** - Tipo nominale (8 valori possibili)
- **Fnlwgt** - Tipo continuo
- **Education** - Tipo nominale (16 valori possibili)
- **Education-num** - Tipo continuo (trasformazione di education in tipo continuo)
- **Marital-status** - Tipo nominale (7 valori possibili)
- **Occupation** - Tipo nominale (14 valori possibili)
- **Relationship** - Tipo nominale (6 valori possibili)
- **Race** - Tipo nominale (6 valori possibili)
- **Sex** - Tipo nominale (2 valori possibili)
- **Capital-gain** - Tipo continuo
- **Capital-loss** - Tipo continuo
- **Hours-per-week** - Tipo continuo
- **Native-country** - Tipo nominale (41 valori possibili)

Per rendere l'input gestibile da K-NN, si sono trasformate le feature di tipo nominale in una serie di feature binarie, e cioè utilizzando delle variabili dummy per ogni possibile valore che la feature può assumere[1].
Per quanto riguarda l'etichettatura, ognuna delle istanze può assumere due possibili valori, e cioè

- **classe 1**: $> 50k$
- **classe 2**: $\leq 50k$

Il dataset risulta essere sbilanciato, in quanto la classe 1 rappresenta il 23.93% del dataset (la classe 2 il 76.07%). Inoltre al suo interno vi sono alcune istanze che per alcune feature non possiede specificazioni. Per questa motivazione si è deciso di escluderle dal processo di valutazione ed apprendimento, riducendo così la cardinalità complessiva del dataset a 45222 istanze[2].

Ottimizzazione delle performance

Data la lentezza dell'algoritmo K-NN, soprattutto con grandi dataset, si è deciso di implementare la funzione di ricerca per i k più vicini in C++, che rispetto ad R risulta molto più veloce. Per quanto riguarda il task di *cross-validation* invece, si è

[1] Con l'introduzione delle variabili dummy si è passati a 88 feature complessive.

[2] Senza le istanze con valori sconosciuti le percentuali delle classi 1 e 2 si attestano rispettivamente su 24.78% e 75.22%.

deciso di parallelizzarlo grazie ad una funzione fornita da una libreria di R[3], che ha permesso di suddividere agevolmente il problema.

1. Metodi e valutazione dei modelli

Per il problema trattato si sono utilizzate due versioni di K-NN, e cioè quella classica e quella online.

K-NN

Questo algoritmo permette di generare un classificatore h_{k-NN} memorizzando l'intero training set fornito in input e calibrando un suo parametro k, il quale permette di gestire l'underfitting e l'overfitting sul problema.

L'assegnazione dell'etichetta ad un'istanza x avviene nella seguente maniera:

$h_{k-NN}(x) =$la maggioranza delle etichette y_t delle k istanze x_t più vicine a x

K-NN online

Si tratta di una variante di K-NN che in fase di training opera nella seguente maniera:

- Prova a classificare l'istanza x_t usando K-NN sull'insieme S.
- Se $h_{K-NN}(x_t) \neq y_t$ aggiungi x_t all'insieme S.

L'algoritmo inizia da un insieme S vuoto, aggiungendovi man mano esempi quando sbaglia, lasciandolo inalterato invece quando non commette errori.

Nested Cross-Validation

Per la stima delle performance dei classificatori generati dalla versione base di K-NN e quella online si è deciso di utilizzare la *cross-validation esterna* a 5 fold, sfruttando come metrica il *test error*, calcolato come segue:

$$\tilde{er}(h_{k-NN}) = \frac{1}{n}\sum_{i=1}^{n} l(y_t, h_{k-NN}(x_t))$$

Dove l è la funzione di perdita *zero-uno*, che vale 1 nel caso in cui il predittore commette un errore.

Per selezionare invece il k ottimale automaticamente, si è effettuata una cross-validation interna, sempre a 5 fold, sia nella fase di selezione finale (dove viene scelto il k finale del modello), sia per la selezione del k ottimale di ogni singolo fold generato dalla cross validazione esterna. Nella figura 1 sono mostrati i passi della *nested cross-validation* (cross validazione interna + cross validazione esterna)

[3] *parLapply*, nella libreria di R *parallel*.

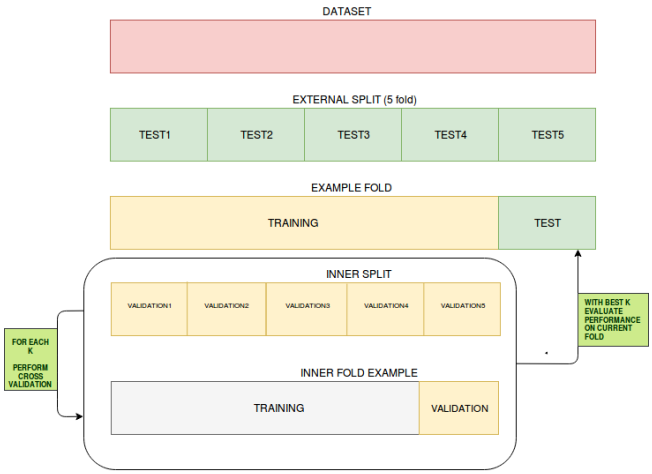

Figure 1. Il modo in cui è avvenuto lo split del dataset sia per la cross validazione esterna (split più esterno) sia per la cross validazione interna (split interno sui singoli training set di ogni rispettivo fold)

Rischio sequenziale

Nel caso specifico di K-NN online si è deciso inoltre di valutare l'andamento del rischio sequenziale sull'intero dataset, calcolato come segue:

$$er_{seq}(t) = \frac{1}{t}\sum_{i=1}^{t} l(y_t, h_{k-NN}(x_t))$$

ad ogni passo t, e cioè alla t-esima istanza fornita in pasto all'algoritmo.

2. Sperimentazione

References

[1] M. Lichman. UCI machine learning repository. 2013. https://archive.ics.uci.edu/ml/datasets/adult