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

Sommario
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

Indice

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

Il dataset risulta essere sbilanciato, in quanto la classe 2 rappresenta il 23.93% del dataset (la classe 1 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].

[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 75.22% e 24.78%.

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 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:

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

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*.

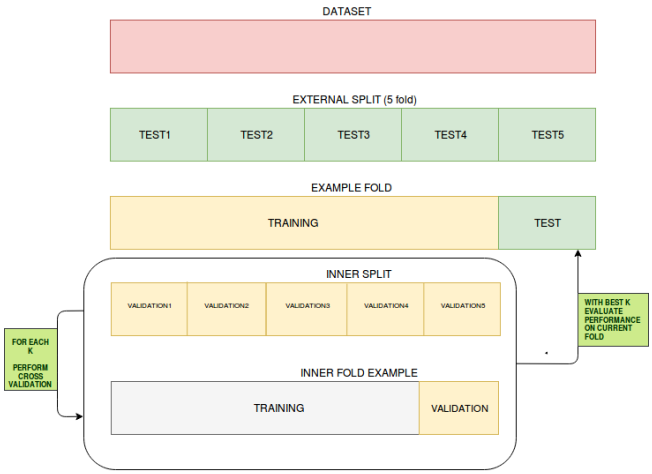


Figura 1. Il modo in cui è avvenuto lo split del dataset sia per la cross validazione esterna (split più esterno) sia per la cross validazione interna (split interno sui singoli training set di ogni rispettivo fold)

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

La sperimentazione è stata effettuata sia normalizzando i dati[4], sia in assenza di normalizzazione, questo per verificare l'impatto che questa procedura ha sulle performance dei classificatori generati.

La ricerca del parametro k è avvenuta su di un set di 26 possibili valori[5].

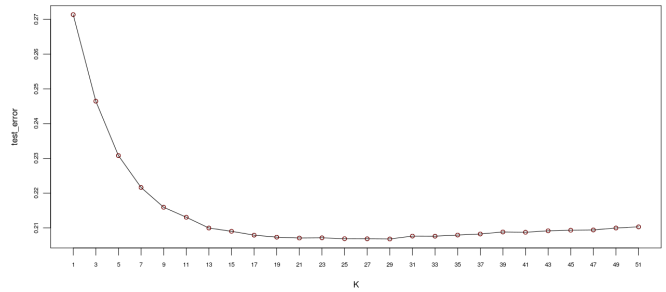
K-NN senza normalizzazione

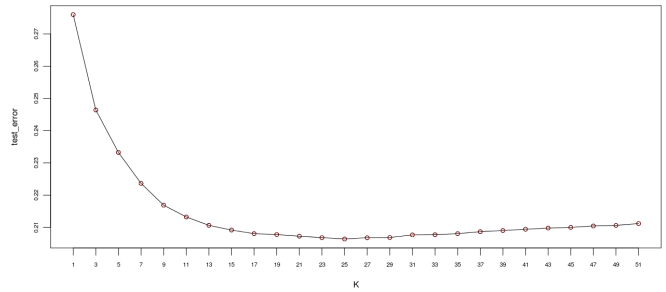
Le curve del test error al variare di k, valutate tramite cross-validation interna per ogni fold, hanno un andamento molto simile, come si evince dalla figura 2.

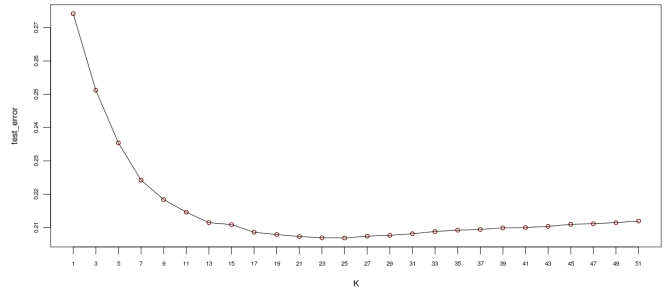
Il test error ha infatti il classico comportamento che si può osservare in K-NN, e cioè diminuisce progressivamente fino

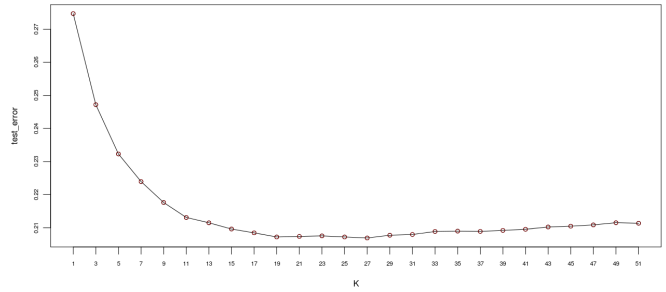
[4] Ogni specificazione delle feature è stata trasformata in una specificazione di una normale standard, sottraendo ad ogni valore la media campionaria e dividendo per la deviazione standard campionaria

[5] K={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51}









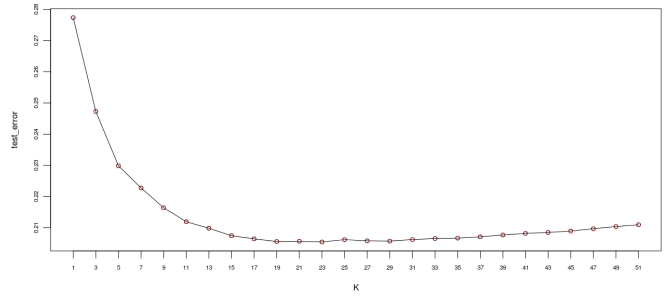


Figura 2. Test error (della cross validazione interna al fold) al variare di k per ogni singolo fold.

fold	k
1	29
2	25
3	25
4	27
5	25

Tabella 1. I k ottimi generati per ogni fold

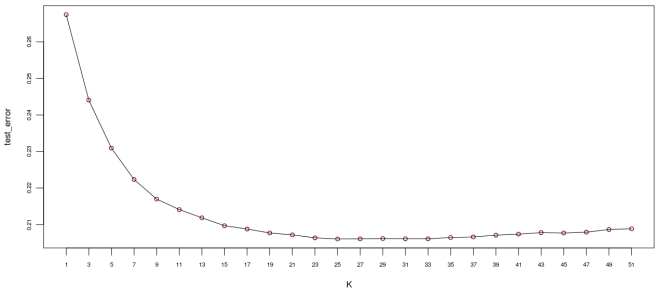


Figura 3. Test error al variare di k per la cross validazione interna finale.

ad arrivare ad un k^* ottimo in cui l'errore è minimo, per poi ritornare a salire nuovamente. I k^* ottimi ottenuti risultano essere poi relativamente vicini per ogni fold (così come il test error), come si evince dalla tabella 1.

La performance finale del classificatore con selezione automatica del k^* ottimo, valutata con cross validazione esterna, si attesta sul 79% di accuratezza. Infine è stato selezionato il k finale del modello, con una nuova cross validation interna, questa volta su tutto il dataset, che è risultato essere pari a 25. L'andamento del test error al variare di k sulla cross-validation interna finale, è mostrata nella figura 3.

Nelle tabelle 2 e 3 vi sono un po' di statistiche dettagliate su questa specifica sperimentazione.

K-NN con normalizzazione dei dati

TO- DO

3. Lavori futuri

- Verifica delle performance a seguito dello shuffling dei dati.
- Verifica delle performance su diversi sub-set di feature.

class	precision
1	0.7921+-0.0036
2	0.8206+-0.0136

Tabella 2. La precision finale per le due classi

time	accuracy	finalK
81524.295	0.7940+-0.0029	25

Tabella 3. Alcune statistiche della sperimentazione. Il tempo totale per effettuare la nested cross-validation, espresso in secondi, l'accuratezza finale +- deviazione standard e il k finale selezionato.

Riferimenti bibliografici

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