Università degli Studi di Urbino Carlo Bo

Implementazione di Regressione Lineare e K-Nearest Neighbors in Haskell e Prolog

Corso: Programmazione Logica e Funzionale

31 luglio 2023

Docente
Prof. Marco Bernardo

Corsista
Andrea De Lorenzis, 308024

Indice

1 Specifica del problema

Linear regression e K-Nearest neighbors (KNN) sono due algoritmi che trovano applicazione in diversi campi, tra cui statistica, analisi di dati e intelligenza artificiale. La prima consente di prevedere il valore di una variabile sconosciuta mediante un modello basato su un'equazione lineare. La seconda è una tecnica utilizzata per la classificazione di oggetti basandosi sulle caratteristiche degli oggetti vicini a quello considerato. L'obiettivo di questo progetto è quello di implementare le due tecniche nei linguaggi di programmazione Haskell e Prolog.

2 Analisi del problema

2.1 Dati di ingresso del problema

I dati di ingresso del problema sono inseriti dall'utente e consistono in un insieme di punti bidimensionali per allenare e valutare i modelli. L'utente inizialmente deve fornire un valore numerico per scegliere l'operazione da svolgere (0 - Linear regression, 1 - KNN, 2 - Terminare il programma). Una volta scelta una delle prime due operazioni, all'utente è richiesto di inserire un insieme di punti bidimensionali per allenare il modello. Successivamente, potrà inserire un ulteriore punto da valutare sul modello allenato, ottenendo poi il risultato. In particolare, i dati di ingresso per la linear regression sono:

- Un insieme di punti 2D separati da spazio, ognuno dei quali è costituito da una coordinata x e una coordinata y

$$x_1 \ y_1 \ x_2 \ y_2 \ \dots$$

- Un punto 2D da valutare sul modello lineare allenato.

I dati di ingresso per il KNN sono:

- Un insieme di punti 2D etichettati e separati da andata a capo, ognuno dei quali è costituito da una coordinata x, una coordinata y e una classe o etichetta numerica che rappresenta la categoria di appartenenza del punto

$$x_1 \ y_1 \ \text{label1}$$
$$x_2 \ y_2 \ \text{label2}$$
$$\dots$$

- Il valore k di vicini da valutare per il punto.
- Il punto 2D di test, senza etichetta, da classificare tramite KNN.

2.2 Dati di uscita del problema

I dati di uscita per la linear regression sono:

- Coefficienti della retta di regressione, ossia il valore della pendenza (m) e dell'intercetta (b), i quali definiscono l'equazione della retta

$$y = mx + b$$

- Risultati predittivi stimando nuovi valori di y per determinati valori di x sulla base del modello di regressione lineare ottenuto.

Invece, l'operazione di KNN produce in uscita:

- Classificazione del punto di test, al quale verrà assegnato una classe in base alle etichette della maggioranza dei suoi k punti più vicini.
- Etichette dei k punti più vicini al punto di test, informazioni utile a fini di comprendere le ragioni della classificazione assegnata al punto di test.

2.3 Relazioni intercorrenti

Nella linear regression si cerca di modellare la relazione tra la variabile indipendente x e la variabile dipendente y attraverso una retta. Pertanto la relazione che intercorre tra i dati di ingresso ed uscita per questa operazione è:

$$y = mx + b$$

dove:

- y è la variabile dipendente (o di output)
- x è la variabile indipendente (o di input)
- m è il coefficiente di pendenza, che rappresenta il cambiamento in y rispetto a una variazione unitaria in x
- b è l'intercetta, che rappresenta il valore di y quando x è uguale a zero

Per il calcolo dei coefficienti m e b vengono utilizzate le seguenti formule:

$$m = \frac{Cov(x, y)}{Var(x)}$$
$$b = \bar{y} - m \cdot \bar{x}$$

dove:

- $Cov(c, y)$ rappresenta la covarianza tra x e y, calcolata come media delle deviazioni dei punti (x, y) rispetto alle loro medie. Si calcola usando questa formula:

$$\text{cov}(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

- $Var(x)$ rappresenta la varianza di x, data da:

$$\text{var}(x) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

- $\bar{x}$ e $\bar{y}$ sono le medie di x e y, rispettivamente.

Dopo aver calcolato i coefficienti m e b, possiamo utilizzare l'equazione della retta per effettuare delle previsioni per dei nuovi punti

$$\hat{y} = mx + b$$

Dove $\hat{x}$ è il valore predetto di y per il nuovo valore di x.

Per quanto riguarda il KNN, questo si basa su una misura di distanza tra i punti nel dataset. Useremo la distanza euclidea tra due punti (x_1, y_2) e (x_2, y_2), data da:

$$Dist(x_1, y_2, x_2, y_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Il parametro k influisce sulla classificazione dei punti e rappresenta il numero di vicini che verranno considerati. Un valore di k più piccolo comporta una classificazione più influenzata dai punti vicini, mentre una valore più grande porta ad una classificazione più generale basata su una maggiore diversità di vicini. Il punto di test viene poi assegnato alla classe di maggioranza presa dai suoi k punti più vicini. Vengono cioè conteggiate le etichette dei punti vicini, e si assegna al punto l'etichetta avente il maggior numero di occorrenze.

3 Progettazione dell'algoritmo

3.1 Scelte di progetto

Gli input per entrambi i programmi sono liste di punti bidimensionali, rappresentati mediante coppie di valori reali nel caso della regressione lineare, e triple di valori nel caso del KNN (due valori reali e un carattere). La lista di coppie reali deve essere inserita nel seguente formato:

$$[(x_1, y_1), \ldots, (x_n, y_n)]$$

I punti del KNN invece sono acquisiti in questo formato, contenente anche un carattere per rappresentare la classe:

$$[(x_1, y_1, ' < classe >'), \ldots, (x_n, y_n, ' < class >')]$$

All'avvio del programma, si è deciso di presentare all'utente un menu principale in cui poter scegliere le diverse operazioni (1: regressione lineare, 2: KNN, 3: uscita). Una volta scelta una delle due operazioni, all'utente viene chiesto di inserire il dataset corrispondente nel corretto modo. Una volta inserito il dataset e svolta l'operazione, l'utente può continuamente valutare nuovi punti sul modello creato all'interno del ciclo. E' il programma che, al termine di ogni valutazione, chiede all'utente se ha intenzione di procedere con un'ulteriore operazione o se preferisce tornale al menu per svolgere un'altra operazione. Il programma termina quando si sceglie l'opzione numero 3.

3.2 Passi dell'algoritmo

I passi dell'algoritmo per l'intero programma sono:

- Presentare il menu di scelta operazioni:
  - Caso base: se l'utente sceglie l'opzione 3 (chiusura del menu).
  - Caso generale: se viene scelta un'altra opzione valida, viene eseguita l'operazione corrispondente:
    1. Regressione lineare:
       * Lettura del dataset di punti bidimensionali, nel formato descritto.
       * Estrazione delle coordinate x e y dalla lista di punti in due liste separate.
       * Calcolo della varianza sui valori di x.
       * Calcolo della covarianza tra le liste di valori di x e y.
       * Calcolo della media su entrambe le liste di valori di x e y.
       * Applicazione dei risultati parziali nelle formule descritte sopra per trovare pendenza e intercetta della retta.
       * Stampa del risultato, ossia dei coefficienti della retta trovata.
       * Lettura del valore x di test da valutare sulla retta interpolatrice.
       * Stampa del risultato, ossia del corrispondente valore y trovato.
       * Richiesta se continuare o meno con altre valutazioni:
         · Caso base: se la scelta dell'utente è 'no'.
         · Caso generale: se la scelta dell'utente è 'si', viene svolta una nuova valutazione (lettura del valore e stampa del risultato).
    2. K-nearest neighbors:
       * Lettura del dataset di punti etichettati, nel formato descritto.
       * Lettura del numero di vicini da valutare (k).
       * Lettura del punto di test da valutare (senza etichetta).
       * Ricerca dei k punti più vicini al punto di test.
       * Ricerca della classe di maggioranza tra i vicini del punto, cioè quella avente il maggior numero di occorrenze tra i vicini.
       * Stampa della classe trovata per il punto
       * Richiesta se continuare o meno con altre valutazioni:
         · Caso base: se la scelta dell'utente è 'no'.
         · Caso generale: se la scelta dell'utente è 'si', viene svolta una nuova valutazione (lettura del punto e stampa del risultato)
    3. Uscita dal menu: stampa del messaggio di chiusura.

4 Implementazione dell'algoritmo

5 Testing del programma

Il programma emette un messaggio di errore e chiede di rieseguire l'inserimento nei seguenti casi:

- All'interno del menu viene scelta un'opzione non presente o un valore che non sia un numero intero.
- All'atto della scelta se continuare o meno con la valutazione dei valori, l'utente inserisce un valore diverso da 's' o 'n'.
- L'utente inserisce il dataset in un formato incorretto, oppure tale per cui i punti al suo interno non siano numeri reali o siano scritti in un formato incorretto.
- L'utente inserisce un dataset contenente meno di due punti.
- L'utente inserisce un valore/punto di valutazione in un formato incorretto o di un tipo diverso da quello richiesto.

5.1 Haskell

5.1.1 Linear regression

Test Haskell 1
Dataset: $[(1,2),(2,4),(3,6),(4,8),(5,10)]$
Valore x: 6.0
Retta: $y = 2.00x + 0.00$
Valore y: 12.00

Test Haskell 2
Dataset: $[(1,2),(2,3),(3,7),(4,6),(5,8),(6,9)]$
Valore x: 7.0
Retta: $y = 1.50x + 0.50$
Valore y: 11.50

Test Haskell 3 (Pendenza negativa)
Dataset: $[(-1,3),(0,1),(1,-1),(2,-3),(3,-5)]$
Valore x: 4.0
Retta: $y = -2.00x + 1.00$
Valore y: -7.00

Test Haskell 4 (Retta verticale)
Dataset: $[(2,1),(2,2),(2,3),(2,4),(2,5)]$
Valore x: 2.0
Retta: $y = NaNx + NaN$
Valore y: NaN

Test Haskell 5 (Numeri decimali)
Dataset: $[(0.1,0.3),(0.2,0.6),(0.3,0.9),(0.4,1.2),(0.5,1.5)]$
Valore x: 0.6
Retta: $y = 3.00x + 0.00$
Valore y: 1.80

5.1.2 K-Nearest neighbors

Test Haskell 6
Dataset: $[(1, 2,' A'), (2, 3,' A'), (2, 4,' A'), (3, 5,' B'), (4, 6,' B'), (5, 7,' B'), (6, 8,' C'), (7, 9,' C'), (8, 10,' C')]$
K: 3
Punto di test: $(3, 4)$
Classe prevista: 'A'

Test Haskell 7
Dataset: $[(1, 2,' A'), (2, 3,' A'), (2, 4,' A'), (3, 5,' B'), (4, 6,' B'), (5, 7,' B'), (6, 8,' C'), (7, 9,' C'), (8, 10,' C')]$
K: 2
Punto di test: $(6, 5)$
Classe prevista: 'B'

Test Haskell 8 (Singolo vicino)
Dataset: $[(1, 2,' A'), (2, 3,' A'), (2, 4,' A'), (3, 5,' B'), (4, 6,' B'), (5, 7,' B'), (6, 8,' C'), (7, 9,' C'), (8, 10,' C')]$
K: 1
Punto di test: $(3.5, 4.5)$
Classe prevista: 'B'

Test Haskell 9 (Punto di test uguale ad un vicino)
Dataset: $[(1, 2,' A'), (2, 3,' A'), (2, 4,' A'), (3, 5,' B'), (4, 6,' B'), (5, 7,' B'), (6, 8,' C'), (7, 9,' C'), (8, 10,' C')]$
K: 1
Punto di test: $(2, 4)$
Classe prevista: 'A'

Test Haskell 10 (Punto di test vicino a 3 punti della stessa classe 'A')
Dataset: $[(1, 2,' A'), (2, 3,' A'), (3, 4,' A'), (4, 5,' B'), (5, 6,' B')]$
K: 5
Punto di test: $(3.5, 4.5)$
Classe prevista: 'A'

5.2 Prolog

5.2.1 Linear regression

Test Prolog 1
Dataset: $[(1, 2), (2, 4), (3, 6), (4, 8), (5, 10)]$
Valore x: 6.0
Retta: $y = 2.00x + 0.00$
Valore y: 12.00

Test Prolog 2
Dataset: $[(1, 2), (2, 3), (3, 7), (4, 6), (5, 8), (6, 9)]$
Valore x: 7.0
Retta: $y = 1.50x + 0.50$
Valore y: 11.50

Test Prolog 3 (Pendenza negativa)
Dataset: $[(-1, 3), (0, 1), (1, -1), (2, -3), (3, -5)]$
Valore x: 4.0

Retta: $y = -2.00x + 1.00$
Valore y: -7.00

Test Prolog 4 (Retta verticale)
Dataset: $[(2,1),(2,2),(2,3),(2,4),(2,5)]$
Valore x: 2.0
Retta: $y = NaNx + NaN$
Valore y: NaN

Test Prolog 5 (Numeri decimali)
Dataset: $[(0.1,0.3),(0.2,0.6),(0.3,0.9),(0.4,1.2),(0.5,1.5)]$
Valore x: 0.6
Retta: $y = 3.00x + 0.00$
Valore y: 1.80

5.2.2 K-Nearest neighbors

Test Prolog 6
Dataset: $[(1,2,'A'),(2,3,'A'),(2,4,'A'),(3,5,'B'),(4,6,'B'),(5,7,'B'),(6,8,'C'),(7,9,'C'),(8,10,'C')]$
K: 3
Punto di test: $(3,4)$
Classe prevista: 'A'

Test Prolog 7
Dataset: $[(1,2,'A'),(2,3,'A'),(2,4,'A'),(3,5,'B'),(4,6,'B'),(5,7,'B'),(6,8,'C'),(7,9,'C'),(8,10,'C')]$
K: 2
Punto di test: $(6,5)$
Classe prevista: 'C'

Test Prolog 8
Dataset: $[(1,2,'A'),(2,3,'A'),(2,4,'A'),(3,5,'B'),(4,6,'B'),(5,7,'B'),(6,8,'C'),(7,9,'C'),(8,10,'C')]$
K: 2
Punto di test: $(6,5)$
Classe prevista: 'C'

Test Prolog 9 (Singolo vicino)
Dataset: $[(1,2,'A'),(2,3,'A'),(2,4,'A'),(3,5,'B'),(4,6,'B'),(5,7,'B'),(6,8,'C'),(7,9,'C'),(8,10,'C')]$
K: 1
Punto di test: $(3.5,4.5)$
Classe prevista: 'B'

Test Prolog 10 (Punto di test uguale ad un vicino)
Dataset: $[(1,2,'A'),(2,3,'A'),(2,4,'A'),(3,5,'B'),(4,6,'B'),(5,7,'B'),(6,8,'C'),(7,9,'C'),(8,10,'C')]$
K: 1
Punto di test: $(2,4)$
Classe prevista: 'A'