Credit Card Approval

Comparazione classificatori per approvazione di carte di credito

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

Motivo dell'esperimento

Con questo esperimento abbiamo voluto confrontare i vari classificatori generativi e discriminativi affrontati nel corso di Metodi Quantitativi per l'informatica.
I vari classificatori sono :

Generativi

- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Diagonal Discriminant Analysis

Discriminativi

- Linear Logistic Regression
- Logistic Regression (Boundary quadratico)
- Logistic Regression (Boundary lineare e regolarizzazione)
- Logistic Regression (Boundary quadratico e regolarizzazione)

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

IL DATASET

Il Dataset contiene i dati offuscati (per motivi di privacy) riguardanti informazioni su alcuni clienti di banca che hanno richiesto una carta di credito. I possibili risultati sono due : *Idoneo*, *non idoneo*.

Sono presenti quasi 700 campioni, 37 dei quali con dati mancanti. Per semplicità abbiamo deciso di eliminarli

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

IL LINGUAGGIO

È stato deciso di utilizzare MATLAB insieme al pacchetto fornito dal corso (pmtk3) e sfruttare alcune librerie già presenti.

Insieme a quelle librerie abbiamo sviluppato funzioni per effettuare la maggior parte delle operazioni.

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

STANDARDIZZAZIONE

Per poter standardizzare la Design Matrix abbiamo usato la nostra funzione LoadData(), la quale carica la design matrix ed effettua la standardizzazione per le feature continue *standardize(X)* ed il one hot encoding per quelle categoriche con la funzione *oneHotEncoding(V)*

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

MANIPOLAZIONE DATI

Dopo la standardizzazione abbiamo usato la funzione *divide(X,Y, p)* che come prima cosa effettua la permutazione di tutte le righe della Design Matrix e divide quest'ultima (e quella delle Y) in Test e Training in base alla percentuale p passata come parametro

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

divide(X, Y, p)

- X = Design Matrix
- Y = Class Labels
- P = percentuale

Divide le matrici X ed Y in Test set(p%) e Training set(1-p%), dopo aver permutato tutte le righe.

PCA

Abbiamo deciso di effettuare la PCA, per eliminare le feature con poca importanza, e da 46 siamo passati a 32.
In questo caso abbiamo utilizzato una funzione già implementata in MATLAB

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

PCA(X)

- X = Design Matrix

Effettua la PCA su X, e ritorna una lis

PROCEDIMENTO CALCOLO ERR-RATE

Una volta divisa la Design Matrix con *divide*, si procede con il fitting dei vari modelli (7 in totale). Ogni modello è fittato 10 volte ed il modello trovato nell'iterazione i-sima, viene applicato alla matrice di test ed i risultati (percentuale di misclassification) vanno nella matrice *errRates*.

I valori predetti, invece, sono aggiunti alla matrice del Bias (*biasMatrix*), usati per poi calcolare sia il B*ias* che la V*ariance*

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

I MODELLI

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

GDA

L'implementazione degli algoritmi di fitting per la GDA è stata affidata alle librerie standard di MATLAB.
In particolare, è stata usata la funzione *fitcdiscr*, che allena un classificatore tramite la GDA. Specificando alcuni parametri è possibile scegliere quale tipo di GDA usare:

- LDA
- QDA
- DLDA

Per ogni modello è stata creata una funzione apposita.

fitcdiscr(X, Y, …)

- X = Design Matrix
- Y= Class Labels

opzionali :

- Prior = imposta il prior per ogni classe di Y
- DiscrimType = imposta il tipo di Discrimanante (LDA, QDA, …)

Ritorna un discriminant analysis classifier in base ai parametri.

crossval(Obj)

- Obj = Discriminant analysis classifier, prodotto usando fitcdiscr

Crea un modello partizionato da obj, discriminant analysis classifier già fittato.

LDA(X, Y, testX)

- X = Design Matrix
- Y = Class Labels
- testX=partizione del dataset utilizzata per il testare il modello computato.

Effettua la Linear Discriminant Analysis usando fitcdiscr con parametro DiscrimType = ‘pseudoLinear’, che inverte la matrice di covarianza usando la pseudo-inversa, effettua la cross validation e ritorna l’errore medio di tutte le partizioni (kFold = 5).

QDA(X, Y, testX)

- X = Design Matrix
- Y = Class Labels
- testX=partizione del dataset utilizzata per il testare il modello computato.

Effettua la Quadratic Discriminant Analysis usando fitcdiscr con parametro DiscrimType = ‘pseudoQuadratic’, che inverte la matrice di covarianza usando la pseudo-inversa, effettua la cross validation e ritorna l’errore medio di tutte le partizioni (kFold = 5).

DLDA(X, Y, testX)

- X = Design Matrix
- Y = Class Labels
- testX=partizione del dataset utilizzata per il testare il modello computato.

Effettua la Diagonal Linear Discriminant Analysis usando la fitcdiscr con parametro DiscrimType = 'diagLinear', effettua la cross validation e ritorna l'errore medio di tutte le partizioni (kFold = 5).

RISULTATI CLASSIFICATORI GENERATIVI

Tutti e tre i modelli, nonostante la loro complessità danno risultati simili.
Supponiamo sia dovuto all'ottimizzazione delle librerie standard.

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

LOGISTIC REGRESSION

Il modello discriminativo utilizzato è la Logistic Regression. Abbiamo optato per una boundary lineare ed uno quadratico per il fitting. A queste curve, abbiamo poi associato la regolarizzazione L2, per limitare le oscillazioni di queste ultime, e la tecnica si è rivelata particolarmente efficace per la Logistic Regression con boundary quadratico, con una diminuzione dell'errore medio del 6% !

Per la regolarizzazione abbiamo implementato una funzione che trova il parametro *lambda* (tra 0 e 20), che minimizza gli error-rate complessivi. Una volta trovato lambda, abbiamo trovato il modello imponendo quel preciso valore di lambda. C'è stato un miglioramento dell'1% (da 11,7 a 10,9).

Valerio Colitta, Daniele Cominu, Alessio Fiorenza

LLogReg(X, Y,testX)

- X = Design Matrix
- Y = Class Labels
- testX=partizione del dataset utilizzata per il testare il modello computato.

Effettua la Logistic Regression con Boundary Lineare e cross validation ed infine ritorna l'errore medio di tutte le partizioni (kFold = 5).

QLogReg(X, Y,testX)

- X = Design Matrix
- Y = Class Labels
- testX=partizione del dataset utilizzata per il testare il modello computato.

Effettua la Logistic Regression con Boundary Quadratico e cross validation ed infine ritorna l'errore medio di tutte le partizioni (kFold = 5).

LLogRegReg(X, Y,testX)

- X = Design Matrix
- Y = Class Labels
- testX=partizione del dataset utilizzata per il testare il modello computato.

Effettua la Logistic Regression con Boundary Lineare con regolarizzazione. Esegue una cross validation per determinare il miglior iper-parametro ed infine ritorna l'errore medio di tutte le partizioni (kFold = 5).

QLogRegReg(X, Y,testX)

- X = Design Matrix
- Y = Class Labels
- testX=partizione del dataset utilizzata per il testare il modello computato.

Effettua la Logistic Regression con Boundary Quadratico con regolarizzazione. Esegue una cross validation per determinare il miglior iper-parametro ed infine ritorna l'errore medio di tutte le partizioni (kFold = 5).

RISULTATI CLASSIFICATORI DISCRIMINATIVI

In generale, tra tutti e quattro i modelli, i due con complessità minore (LB e LBR) hanno dato risultati migliori. In particolare la Logistic Regression con boundary lineare e regolarizzazione (LBR) totalizza gli errori più bassi tra tutti gli errori minimi e medi.

I modelli con parametri più elevati, e quindi più predisposti ad overfitting tendono a non generalizzare a fronte di nuovi dati di input. Tuttavia, con la LR con Boundary Quadratico e Regolarizzazione, si riesce a limitare la variazione del polinomio, avendo così varianza minore e risultati migliori.