Αναγνώριση Προτύπων

Άσκηση 5

Δασούλας Ιωάννης - 1053711

**1. ∆ιάβασµα προτύπων από ιατρικές εργαστηριακές εξετάσεις για την διάγνωση ζακχαρώδη διαβήτη σε γυναίκες**

Αρχικά διαβάζονται οι τιμές του αρχείου pima-indians-diabetes.data

μέσω της συνάρτησης [x,c] = ReadDiabetes(n), όπου total ο αριθμός των δειγμάτων (768). Το αρχείο περιέχει 8 μετρήσεις και μία κατηγορία, 0 ή 1. Αν η κατηγορία είναι 0, ο εξεταζόμενος θεωρείται υγιής, ενώ αν είναι 1 θεωρείται ασθενής. Με τη συνάρτηση αποθηκεύονται στον πίνακα x οι 8 μετρήσεις σε έναν πίνακα 8x768 και στον c όλες οι κατηγορίες σε ένα διάνυσμα 1x768.

Έπειτα, με μία επανάληψη, χωρίζονται σε 2 πίνακες οι μετρήσεις των υγιών εξεταζόμενων (πίνακα H) και των ασθενών (πίνακας U).

n = 768;

[x,c] = ReadDiabetes(n);

H\_index = 1; %H -> Healty

U\_index = 1; %U -> Unhealty

for i=1:n

if(c(i) == 1)

H(:,H\_index) = x(:,i);

H\_index = H\_index + 1;

elseif (c(i) == 2)

U(:,U\_index) = x(:,i);

U\_index = U\_index + 1;

end

end

**2. Υπολογίστε τους συντελεστές των γραµµικών συναρτήσεων απόφασης µε τον αλγόριθµο Ho-Kashyap.**

Αρχικά, για να λειτουργήσει σωστά στη Matlab η συνάρτηση HoKa(), πρέπει να γίνουν κάποιες αλλαγές σε αυτή, καθώς και στις NoGreatValMat(), AbsMat(), GreatValMat(). Επειδή δεν διατίθεται η συνάρτηση rows(), ο υπολογισμός των γραμμών και στηλών των πινάκων γίνεται με την συνάρτηση size(). Επίσης οι εντολές μορφής a++, αντικαταστάθηκαν από εντολές της μορφής a = a + 1, ενώ μπήκαν σε σχόλιο οι εντολές printf(). Οι συναρτήσεις όπως διαμορφώθηκαν είναι:

NoGreatValMat():

function a = GreatValMat( b, v )

%# usage: a = GreatValMat( b, v )

%#

%# This function returns 1 if one element of the matrix b

%# is less than v otherwise returns 0

%#

[r,c] = size(b);

for i=1:r

for j = 1:c

if ( b(i,j) < v )

a = 0 ;

return ;

end

end

end

a = 1 ;

AbsMat():

function a = AbsMat( b )

%# usage: a = AbsMat( b )

%#

%# This function returns the absolute values of the array elements |b(i,j)|

%#

[r,c] = size(b);

for i=1:r

for j = 1:c

if ( b(i,j) < 0.0 )

a(i,j) = - b(i,j) ;

else

a(i,j) = b(i,j) ;

end ;

end ;

end ;

NoGreatValMat():

function a = NoGreatValMat( b,v )

a = 0 ;

[r,c] = size(b);

for i=1:r

for j = 1:c

if ( b(i,j) >= v )

a = a + 1 ;

end

end

end

HoKa():

function [Rc,Rep,w ] = HoKa(x1,x2,Lr,MaxRep)

%#

%# [Rc,Rep] = HoKa(x1,x2,Lr,MaxRep)

%#

%# Input

%# x1: Pattern Vectors for the first class

%# x2: Pattern Vectors for the second class

%# Lr: Learning rate

%# MaxRep: Maximum repetitions

%# Output

%# Rc: Correct classification rate using the C-method

%# Rep: Pattern vectors on each class

%#

[x1\_rows, x1\_cols] = size(x1);

[x2\_rows, x2\_cols] = size(x2);

NumOfP1 = x1\_cols ;

x1 = [x1;ones(1,NumOfP1)] ;

NumOfP2 = x2\_cols ;

x2 = [x2;ones(1,NumOfP2)] ;

Rep = [NumOfP1,NumOfP2] ;

TotPat = sum(Rep) ;

Rc = zeros(2,1) ;

if ( x1\_rows ~= x2\_rows )

printf( 'Error in vectors x1, x2\n' ) ;

end

z = [x1,-x2]';

[z\_rows, z\_cols] = size(z);

Nv = z\_cols ;

Np = z\_rows ;

b = 0.1 \* rand( Np, 1 ) ;

piz = inv( z' \* z ) \* z' ;

w = piz \* b ;

b1 = z \* w ;

e = b1 - b ;

i = 0 ;

while( i < MaxRep )

%printf( 'Step %d\n', i ) ;

[b1\_rows, b1\_cols] = size(b1);

%printf( 'Classification Score: %7.4f%%\n', ( 100 \* NoGreatValMat( b1, 0.0 ) ) / ( b1\_rows \* b1\_cols) ) ;

if ( GreatValMat(b1,0.0) == 1 )

%printf( 'Linear Separation of classes in %d repetitions\n', i ) ;

Rc = Rep ;

return ;

end

ea = AbsMat(e) ;

b = b + Lr \* ( e + ea ) ;

w = piz \* b ;

b1 = z \* w ;

e = b1 - b ;

i = i+1 ;

end

%display(sprintf( 'Original Ho-Kasyap not convergence in %d repetitions\n', i )) ;

[b1\_rows, b1\_cols] = size(b1);

%display(sprintf( 'Classification Score: %7.4f%%\n', ( 100 \* NoGreatValMat( b1, 0.0 ) ) / ( b1\_rows \* b1\_cols) )) ;

Rc(1) = NoGreatValMat( b1(1:NumOfP1), 0.0 ) ;

Rc(2) = NoGreatValMat( b1(NumOfP1+1:TotPat), 0.0 ) ;

Θέτοντας το learning rate ίσο με 0.1 και τον μέγιστο αριθμό επαναλήψεων ίσο με 1000, δίνονται τα παρακάτω αποτελέσματα:

Lr = 0.1;

MaxRep = 1000;

[Rc, Rep, w] = HoKa(H,U,Lr,MaxRep);

w

w =

-0.0031

-0.0008

0.0003

-0.0000

0.0000

-0.0023

-0.0212

-0.0005

0.1999

**3. Υπολογίστε το σφάλµα του συστήµατος ταξινόµησης.**

Το σφάλμα υπολογίζεται από τη σχέση: Cerror = 1 - ((Rc(1) + Rc(2)) / n)

Αποτέλεσμα:  
 Cerror = 0.2161

**4. Μελετήστε την επίδραση που έχει ο συντελεστής εκπαίδευσης στον αλγόριθµο Ho-kashyap.**

Για την μέτρηση της αξιοπιστίας, δημιουργήθηκε ένας πίνακας που για δείχνει το ποσοστό αξιοπιστίας για κάθε συντελεστή από 0.05 εώς 1 με βήμα 0.05 και στο τέλος γίνεται το αντίστοιχο plot. Στο διάνυσμα position αποθηκεύονται όλες οι τιμές των συντελεστών και στο correct\_table το ποσοστό αξιοπιστίας για τον συντελεστή αυτό και δεδομένο μέγιστο αριθμό επαναλήψεων.

index = 1;

for i=0.05:0.05:1

position(index) = i;

[Rc, Rep, w] = HoKa(H,U,i,MaxRep);

correct\_table(index) =((Rc(1) + Rc(2))/n);

index = index + 1;

end

plot(position, correct\_table)

Αποτέλεσμα για μέγιστο αριθμό επαναλήψεων ίσο με 1000:

Εικόνα που περιέχει κείμενο, χάρτης, πίνακας, καθιστός

Περιγραφή που δημιουργήθηκε αυτόματα

Για μέγιστο αριθμό επαναλήψεων ίσο με 10000, τα αποτελέσματα ήταν ανάλογα.

Mε ανάλογο τρόπο κατασκευάστηκε και η γραφική συνάρτηση του ριθμού επαναλήψεων σε σχέση με τον συντελεστή εκπαίδευσης.

index = 1;

for i=0.05:0.05:1

position(index) = i;

[Rc, Rep, w, i] = HoKa(H,U,i,MaxRep);

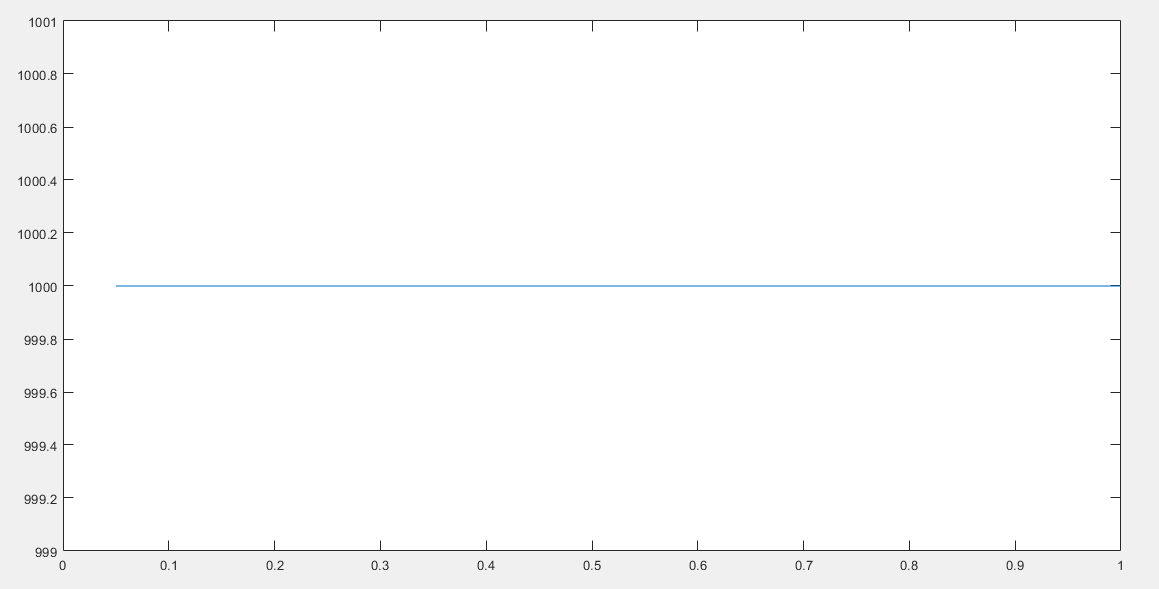
correct\_table(index) = i;

index = index + 1;

end

plot(position, correct\_table)

Αποτέλεσμα:



Παρατηρείται ότι ανεξαρτήτως του συντελεστή εκπαίδευσης, ο αριθμός επαναλήψεων ισούται με τον μέγιστο που έχει τεθεί ίσος με 1000. Παρόλο που ο συντελεστής εκπαίδευσης είναι μεταξύ 0 και 1, ο αλγόριθμος δεν συγκλίνει λόγω των πολλών διαστάσεων των δεδομένων.

**5. Τι παρατηρείτε από τα αποτελέσµατα των µετρήσεων**

Ο αλγόριθμος έχει πολύ καλή απόδοση και το ελάχιστο σφάλμα που δίνει είναι πολύ ικανοποιητικό, ειδικά σε σχέση με τον Perceptron.