Αναγνώριση Προτύπων

# Εργαστηριακή Άσκηση 3

Ο ΑΛΓΟΡΙΘΜΟΣ Κ-MEANS ΚΑΙ ΤΟ ΚΡΙΤΗΡΙΟ ΤΑΞΙΝΟΜΗΣΗΣ ΤΩΝ Κ-ΠΛΗΣΙΕΣΤΕΡΩΝ ΠΡΩΤΟΤΥΠΩΝ

Ιωάννης Δασούλας – 1053711

1. ∆ιάβασµα παραδειγµάτων ανίχνευσης τοπικής συµπεριφοράς της Ιονόσφαιρας

Mε τη συνάρτηση [x,c] = ReadIonosphere(Tot), αποθηκεύονται στον πίνακα x όλα τα δεδομένα ανά σήμα και στον πίνακα c όλες οι αντιστοιχίες σε κλάση ( σε μορφή 1 και 2, όπου bad = 1 και good =2 ).

[x,c] = ReadIonosphere(351);

2. Υπολογίστε τρία εικονικά πρωτότυπα από τα παραδείγµατα

Χρησιμοποιώντας τη συνάρτηση [Rc,Ru,Prot] =

ClassKMDistEucl(x,c,KN)θέτοντας ΚΝ = 3 για τον υπολογισμών 3 εικονικών πρωτοτύπων.

function [Rc,Ru,Prot] = ClassKMDistEucl(x,c,KN)

%#

%# [Rc,Ru,Rep] = ClassKMDistEucl(x,c,KN)

%# Pattern Recognition:

%# Distance measure: Euclidian

%# Prototypes: K-prototypes K-MEANS

%# Classification rule: Minimum Distance

%#

%# Input

%# x: Pattern Vectors

%# c: Classes

%# KN: Number of protpotypes

%# Output

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

%# Ru: Correct classification rate using the U-method

%# Rep: Pattern vectors on each class

%#

NumOfClass = max(c) ;

NumOfPatterns = columns(x) ;

Rep = zeros(NumOfClass,1) ;

Prot = zeros(4,0) ;

%#

%# C-Error

%#

for j = 1:NumOfPatterns

k = c(j) ;

Rep(k) = Rep(k) + 1 ;

end

for i=1:NumOfClass

Pat = zeros(4,0) ;

for j = 1:NumOfPatterns

k = c(j) ;

if ( k == i )

Pat = [ Pat, x(:,j) ] ;

end

end

Prot = [ Prot, KMeans(Pat,KN,0.0001) ] ;

end

NoProt = columns(Prot) ;

Rc = zeros(NumOfClass,1) ;

for i = 1:NumOfPatterns

for j = 1:NoProt

Dist(j) = (x(:,i) - Prot(:,j))' \* ( x(:,i) - Prot(:,j) ) ;

end

Rec = ceil(ArgMin(Dist)/KN) ;

if (Rec == c(i))

Rc(Rec) = Rc(Rec) + 1 ;

end

end

%#

%# U-Error

%#

Ru = zeros(NumOfClass,1) ;

OrgProt = Prot ;

for i = 1:NumOfPatterns

One = c(i) ;

Pat = zeros(4,0) ;

for j = 1:NumOfPatterns

k = c(j) ;

if ( k == One && j ~= i )

Pat = [ Pat, x(:,j) ] ;

end

end

NewProt = KMeans(Pat,KN,0.0001) ;

if ( One == 1 )

Prot = [ NewProt, OrgProt(:,One\*KN+1:NumOfClass\*KN) ] ;

elseif ( One == NumOfClass )

Prot = [ OrgProt(:,1:(One-1)\*KN), NewProt ] ;

else

Prot = [ OrgProt(:,1:(One-1)\*KN), NewProt, OrgProt(:,One\*KN+1:NumOfClass\*KN) ] ;

end

for j = 1:NoProt

Dist(j) = (x(:,i) - Prot(:,j))' \* ( x(:,i) - Prot(:,j) ) ;

end

Rec = ceil(ArgMin(Dist)/KN) ;

if (Rec == One )

Ru(Rec) = Ru(Rec) + 1 ;

end

end

Στον πίνακα Prot αποθηκεύονται, στο πρώτο μισό (στήλες 1-3) τα εικονικά πρωτότυπα της κλάσης bad και στο δεύτερο μισό (στήλες 4-6) τα εικονικά πρωτότυπα της κλάσης good. Έπειτα, με μία επανάληψη διαχωρίζονται τα πρότυπα των δύο κλάσεων σε δύο πίνακες 34 Χ 3 bad και good, όπως φαίνονται παρακάτω.

n = 351;

[x,c] = ReadIonosphere(n);

prots = 3;

[Rc,Ru,Prot] = ClassKMDistEucl(x,c,prots);

bad\_prots = zeros(34,3);

good\_prots = zeros(34,3);

[rows, cols] = size(Prot);

for i=1:rows

bad\_prots(i,1)=Prot(i,1);

bad\_prots(i,2)=Prot(i,2);

bad\_prots(i,3)=Prot(i,3);

good\_prots(i,1) = Prot(i,1 + prots);

good\_prots(i,2) = Prot(i,2 + prots);

good\_prots(i,3) = Prot(i,3 + prots);

end

Τα τρία εικονικά πρωτότυπα της bad κλάσης:

0.6786 0.5238 0.7532

0 0 0

0.5944 0.6617 0.0887

-0.3977 0.3973 -0.0125

0.7571 0.8963 -0.1225

0.1095 -0.2516 0.0684

0.6974 0.5335 0.0165

-0.5372 0.4989 0.0186

0.6747 0.4483 0.1435

0.0112 0.1854 0.1147

0.7948 0.5743 0.1270

-0.0924 0.3262 0.0255

0.6729 0.5730 0.0081

0.0970 -0.1776 -0.0469

0.7628 0.3735 -0.1114

-0.2748 0.4643 -0.0592

0.7655 0.3981 0.1206

0.0065 -0.2925 -0.0580

0.7034 0.4180 0.0577

-0.0600 0.2589 -0.1285

0.6305 0.0770 0.0083

0.3893 0.3079 -0.0803

0.7371 0.6476 -0.1209

-0.2072 0.0521 -0.0399

0.5365 0.5688 0.0603

-0.0341 -0.3480 -0.0109

0.8290 0.9503 0.4509

-0.2219 0.6650 -0.2658

0.6661 0.3777 -0.0404

-0.2473 0.3972 -0.0597

0.7064 0.1557 -0.0899

-0.1700 0.2571 0.0259

0.7390 0.1776 -0.0435

-0.2503 0.4312 0.0628

Τα τρία εικονικά πρωτότυπα της good κλάσης:

1.0000 1.0000 1.0000

0 0 0

0.8277 0.8245 0.8380

0.4067 0.2042 -0.0167

0.6092 0.8021 0.8492

0.6332 0.3401 0.0199

0.2857 0.6885 0.8258

0.8395 0.4627 -0.0084

-0.1140 0.5303 0.8213

0.8487 0.5853 0.0020

-0.4242 0.4229 0.8082

0.7044 0.6729 0.0032

-0.6482 0.2456 0.8090

0.4801 0.6995 -0.0162

-0.7782 0.1285 0.8063

0.1890 0.6780 -0.0046

-0.6718 -0.0130 0.7746

-0.1779 0.6682 -0.0307

-0.5382 -0.1519 0.7582

-0.4148 0.6019 -0.0333

-0.3284 -0.2377 0.7568

-0.5964 0.5166 -0.0099

-0.0677 -0.3096 0.7340

-0.5563 0.4257 -0.0288

0.2044 -0.3784 0.7131

-0.5098 0.3182 -0.0402

0.3711 -0.3907 0.7060

-0.3697 0.2244 -0.0294

0.4558 -0.3998 0.6687

-0.1692 0.1348 -0.0281

0.4030 -0.3765 0.6658

-0.0193 0.0411 -0.0286

0.3377 -0.3521 0.6373

0.1613 -0.0340 -0.0439

3. Υπολογίστε το ελάχιστο σφάλµα του συστήµατος ταξινόµησης

Με την ίδια κλήση της συνάρτησης αποθηκεύονται στον 2Χ1 πίνακα Rc ο αριθμός των παραδειγμάτων που ταξινομούνται σωστά για την κλάση bad και good αντίστοιχα. Το ποσοστό σωστής ταξινόμησης είναι το άθροισμα αυτών των 2 διά το συνολικό αριθμό παραδειγμάτων (351). Οπότε το σφάλμα ισούται με τη μονάδα μείον το ποσοστό αυτό.

Rc

sorted = Rc(1,1)+ Rc(2,1);

Cerror = 1 - sorted/351

Αποτελέσματα:

Rc = 109

200

C-error = 0.1197

4. Επιλέξατε τρία παραδείγµατα σαν πρωτότυπα

Αρχικά υπολογίζοντας τoν αριθμό των bad και good παραδειγμάτων, χωρίζεται ο πίνακας x σε 2 αντίστοιχους πίνακες (bad\_array, good\_array). Έπειτα, για κάθε πρότυπο υπολογίζεται η απόσταση από τα 3 εικονικά και κρατείται η ελάχιστη από αυτές τις 3 στον πίνακα bad\_min\_dist και good\_min\_dist. Στη συνέχεια, με 3 επαναλήψεις υπολογίζονται 3 πρότυπα με την μικρότερη απόσταση (τετράγωνο της ευκλείδιας) και αποθηκεύονται στους πίνακες best\_of\_bad και best\_of\_good.

bad\_counter = 0;

good\_counter = 0;

for i=1:n;

if(c(i)==1);

bad\_counter = bad\_counter + 1;

else

good\_counter = good\_counter + 1;

end

end

bad\_array = zeros(34, bad\_counter);

good\_array = zeros(34, good\_counter);

bad\_index = 1;

good\_index = 1;

for i=1:n;

if(c(i)==1);

for j=1:34;

bad\_array(j, bad\_index) = x(j, i);

end

bad\_index = bad\_index + 1;

else

for j=1:34;

good\_array(j, good\_index) = x(j, i);

end

good\_index = good\_index + 1;

end

end

for i=1:bad\_counter

bad\_dist(i,1) = (bad\_array(:,i) - bad\_prots(:,1))' \* ( bad\_array(:,i) - bad\_prots(:,1) ) ;

bad\_dist(i,2) = (bad\_array(:,i) - bad\_prots(:,2))' \* ( bad\_array(:,i) - bad\_prots(:,2) ) ;

bad\_dist(i,3) = (bad\_array(:,i) - bad\_prots(:,3))' \* ( bad\_array(:,i) - bad\_prots(:,3) ) ;

bad\_min\_dist(i)=min(bad\_dist(i,:));

end

bad\_min1 = 10e7;

for i=1:bad\_counter

if bad\_min1 > bad\_min\_dist(i)

bad\_min1 = bad\_min\_dist(i);

bad\_min1\_pos = i;

end

end

bad\_min2 = 10e7;

for i=1:bad\_counter

if i~=bad\_min1\_pos

if bad\_min2 > bad\_min\_dist(i)

bad\_min2 = bad\_min\_dist(i);

bad\_min2\_pos = i;

end

end

end

bad\_min3 = 10e7;

for i=1:bad\_counter

if (i~=bad\_min1\_pos && i~=bad\_min2\_pos);

if bad\_min3 > bad\_min\_dist(i)

bad\_min3 = bad\_min\_dist(i);

bad\_min3\_pos = i;

end

end

end

best\_of\_bad(:,1) = bad\_array(:, bad\_min1\_pos);

best\_of\_bad(:,2)=bad\_array(:, bad\_min2\_pos);

best\_of\_bad(:,3)=bad\_array(:, bad\_min3\_pos);

for i=1:good\_counter

good\_dist(i,1) = (good\_array(:,i) - good\_prots(:,1))' \* ( good\_array(:,i) - good\_prots(:,1) ) ;

good\_dist(i,2) = (good\_array(:,i) - good\_prots(:,2))' \* ( good\_array(:,i) - good\_prots(:,2) ) ;

good\_dist(i,3) = (good\_array(:,i) - good\_prots(:,3))' \* ( good\_array(:,i) - good\_prots(:,3) ) ;

good\_min\_dist(i)=min(good\_dist(i,:));

end

good\_min1 = 10e7;

for i=1:good\_counter

if good\_min1 > good\_min\_dist(i)

good\_min1 = good\_min\_dist(i);

good\_min1\_pos = i;

end

end

good\_min2 = 10e7;

for i=1:good\_counter

if i~=good\_min1\_pos

if good\_min2 > good\_min\_dist(i)

good\_min2 = good\_min\_dist(i);

good\_min2\_pos = i;

end

end

end

good\_min3 = 10e7;

for i=1:good\_counter

if (i~=good\_min1\_pos && i~=good\_min2\_pos);

if good\_min3 > good\_min\_dist(i)

good\_min3 = good\_min\_dist(i);

good\_min3\_pos = i;

end

end

end

best\_of\_good(:,1) = good\_array(:, good\_min1\_pos);

best\_of\_good(:,2)=good\_array(:, good\_min2\_pos);

best\_of\_good(:,3)=good\_array(:, good\_min3\_pos);

Αποτελέσματα:

Πρωτότυπα της bad κλάσης:

1.0000 1.0000 1.0000

0 0 0

0.0640 0.1859 0.0198

-0.1527 -0.1667 0.0070

-0.0443 0 0.0409

0.0591 0 -0.0085

0.0837 0 0.0212

-0.0246 0 0.0113

-0.0148 0 0.0113

0.1872 0 0.0437

0.0640 0 0.0028

0 0 0.0014

0.1231 0.1154 0.0198

-0.0985 -0.1907 -0.0310

0.0591 0 -0.0198

0 0 0.0607

0.0197 0 -0.0409

-0.0296 0 0.0268

-0.1281 0 -0.0240

-0.2069 0 -0.0042

0.0690 0 0.0437

0.0148 0 -0.0254

0.0690 -0.0513 0.0183

0.0296 -0.0657 0

0.0788 0.0785 0

0.1626 0.0897 -0.0127

0.2808 0.1731 0.0183

-0.0493 -0.1090 -0.0113

-0.0591 0.1250 0.0056

-0.0936 0.0961 -0.0155

0.0443 0.0256 -0.0169

0.0542 -0.0481 -0.0240

0.0739 0.1683 0.0070

-0.1084 0.1955 0

Πρωτότυπα της good κλάσης:

1.0000 1.0000 1.0000

0 0 0

0.7605 0.6832 0.7241

0.0109 0.0537 -0.0108

0.8633 0.8480 0.7970

0.0026 0.0020 0.0108

0.8582 0.8434 0.8000

0.0038 0.0030 0.0020

0.7999 0.8430 0.7902

0.0230 0.0990 0.0108

0.8150 0.7581 0.7842

0.1207 0.0410 -0.0098

0.8310 0.8189 0.8335

0.0074 0.0059 0.0325

0.8182 0.8074 0.8512

0.0085 0.0067 0.0168

0.8278 0.8062 0.8010

-0.0697 -0.1245 -0.0079

0.7653 0.7793 0.7911

0.0388 -0.0354 -0.0296

0.7698 0.7637 0.7596

0.0115 0.0091 0.0335

0.7507 0.7463 0.7478

0.0123 0.0098 0.0552

0.7714 0.7963 0.7261

-0.0030 -0.0424 -0.0148

0.7089 0.7082 0.7804

0.0138 0.0110 0.0061

0.6616 0.6223 0.7409

0.0085 0.1160 -0.0503

0.6630 0.6662 0.8296

0.0148 0.0119 0.0296

0.6389 0.6441 0.7902

0.0152 0.0123 0.0079

5. Υπολογίστε το ελάχιστο σφάλµα του συστήµατος ταξινόµησης

Αρχικά οι 2 πίνακες ενώνονται σε έναν πίνακα 34x6 ονόματι best.

for i=1:3

best(:,i)=best\_of\_bad(:,i);

end

for i=4:6

best(:,i)=best\_of\_good(:,i-3);

end

Έπειτα, δημιουργήθηκε μία νέα συνάρτηση Rc\_best = BestCError(x,c,KN,best) βάση της [Rc,Ru,Prot] = ClassKMDistEucl(x,c,KN)

για τον υπολογισμό του νέου σφάλματος C για τα πρωτότυπα παραδείγματα που υπολογίστηκαν.

function Rc\_best = ClassKMDistEucl(x,c,KN,best)

%#

%# [Rc,Ru,Rep] = ClassKMDistEucl(x,c,KN)

%# Pattern Recognition:

%# Distance measure: Euclidian

%# Prototypes: K-prototypes K-MEANS

%# Classification rule: Minimum Distance

%#

%# Input

%# x: Pattern Vectors

%# c: Classes

%# KN: Number of protpotypes

%# best: Table of prototypes

%# Output

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

%#

NumOfClass = max(c) ;

NumOfPatterns = columns(x) ;

Rep = zeros(NumOfClass,1) ;

%#

%# C-Error

%#

for j = 1:NumOfPatterns

k = c(j) ;

Rep(k) = Rep(k) + 1 ;

end

NoProt = columns(best) ;

Rc\_best = zeros(NumOfClass,1) ;

for i = 1:NumOfPatterns

for j = 1:NoProt

Dist\_best(j) = (x(:,i) - best(:,j))' \* ( x(:,i) - best(:,j) ) ;

end

Rec\_best = ceil(ArgMin(Dist\_best)/KN) ;

if (Rec\_best == c(i))

Rc\_best(Rec\_best) = Rc\_best(Rec\_best) + 1 ;

end

end

Ουσιαστικά, χρησιμοποιείται ο πίνακας best αντί για τον Prot και υπολογίζεται το ποσοστό σωστής ταξινόμησης που χρησιμοποιείται στην συνέχεια ως εξής:

Rc\_new = BestCError(x,c,prots,best);

sorted\_best = Rc\_new(1,1)+ Rc\_new(2,1);

Cerror\_best = 1 - sorted\_best/351

Εκτελεί τα ίδια βήματα που περιεγράφηκαν στο ερώτημα 3.

Αποτέλεσμα:

C-error = 0.2877

6. Τι παρατηρείτε από τα αποτελέσματα των μετρήσεων

Όπως αναμένονταν, το σφάλμα των εικονικών πρωτοτύπων είναι μικρότερο από αυτό των πρωτοτύπων από τα υπάρχοντα παραδείγματα, όπως και στην προηγούμενη εργαστηριακή άσκηση που η ταξινόμηση γίνονταν βάση του τετραγώνου της ευκλείδειας απόστασης. Επίσης, παρατηρείται ότι με τον αλγόριθμο k-means τα σφάλματα είναι πολύ μικρότερα σε σχέση με την προηγούμενη εργαστηριακή άσκηση, αλλά επαληθεύεται και το μειονέκτημά του αλγόριθμου ότι η σύγκλιση επιτυγχάνεται σε ένα τοπικό ελάχιστο του αθροίσματος των αποστάσεων των προτύπων από τα πρωτότυπα, καθώς πολλές φορές τα αποτελέσματα των προγραμμάτων διέφεραν σε ελάχιστα.