**Recunoasterea Formelor si Inteligenta Artificiala**

**1 - Nearest Neighbours**

**PALM**

Dumitru Teodor

Cuprins:

3 ... 1) Breviar Teoretic

5 ... 2) Implementare cu explicatii pe parcurs

8 ... 3) Rezultate Experimentale si Concluzii

10 ... 4) Pro si Contra

10 ... 5) Aplicatii

11 ... 6) Codul utilizat

12 ... 7) Reprezentarea bazei de date PALM

13 ... 8) Bibliografie

1. **Breviar teoretic**

K-NN este una dintre cele mai fundamentale si simple metode de clasificare. Clasificarea k-NN a fost dezvoltata din nevoia de a efectua o analiza discriminatorie atunci cand estimarile parametrice ale densitatilor de probabilitate sunt necunoscute sau dificil de determinat.

Algoritmul traditional al celui mai apropiat vecin (Nearest Neighbour - NN) este un clasificator neparametric, care aloca vectorul de intrare de clasficat X, acelei clase care corespunde celui mai apropiat vecin al lui X din lotul vectorilor de referinta.

Formularea algoritmului este urmatoarea:

Fie setul de N vectori de referinta etichetati

Cu etichetele de apartenenta la una din cele M clase corespunzatoare

Unde

Daca se aplica la intrare un vector X care trebuie clasificat, se determina distanta minima de la X la vectorii de referinta etichetati. Presupunem ca:

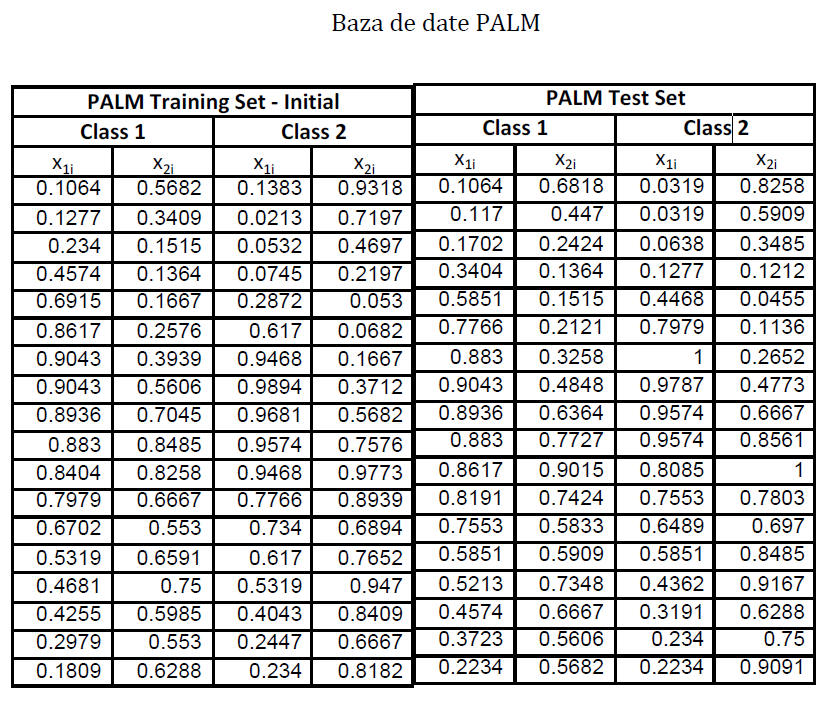
ceea ce inseamna ca este cel mai apropiat vecin al lui X. Daca

atunci decidem ca

,

Altfel spus, se aloca vectorului X clasa celui mai apropiat vecin din lotul de vectori etichetati.

Aceasta idee poate fi extinsa la cei mai apropiati k vectori vecini etichetati ai lui X, astfel incat X este atribuit clasei care este reprezentata printr-o majoritate a celor mai apropiati k vecini. Se obtine astfel algoritmul de clasificare k-NN (al celor mai apropiati k vecini). Pentru k>1 si M=2 clase, se alege k impar.



1. **Implementare cu explicatii pe parcurs**

**Pasul 1:** Se introduc loturile de antrenare si de test (din baza de date PALM) in functie de clase. Fiecare matrice are dimensiunea de 18x2.

Training\_Class1 = [0.1064 0.5682; 0.1277 0.3409; 0.234 0.1515; 0.4574 0.1364; 0.6915 0.16; 0.8617 0.2576; 0.9043 0.3939; 0.9043 0.5606; 0.8936 0.7045; 0.883 0.8485; 0.8404 0.8258; 0.7979 0.6667; 0.6702 0.553; 0.5319 0.6591; 0.4681 0.75; 0.4255 0.5985; 0.2979 0.553; 0.1809 0.6288];

Training\_Class2 = [0.1383 0.9318; 0.0213 0.7197; 0.0532 0.4697; 0.0745 0.2197; 0.2872 0.053; 0.617 0.0682; 0.9468 0.1667; 0.9894 0.3712; 0.9681 0.5682; 0.9574 0.7576; 0.9468 0.9773; 0.7766 0.8939; 0.734 0.6894; 0.617 0.7652; 0.5319 0.947 ; 0.4043 0.8409; 0.2447 0.6667; 0.234 0.8182];

Test\_Class1 =[0.1064 0.6818; 0.117 0.447; 0.1702 0.2424; 0.3404 0.1364; 0.5851 0.1515; 0.7766 0.2121; 0.883 0.3258; 0.9043 0.4848; 0.8936 0.6364; 0.883 0.7727; 0.8617 0.9015; 0.8191 0.7424; 0.7553 0.5833; 0.5851 0.5909; 0.5213 0.7348; 0.4574 0.6667; 0.3723 0.5606; 0.2234 0.5682];

Test\_Class2 = [0.0319 0.8258; 0.0319 0.5909; 0.0638 0.3485; 0.1277 0.1212; 0.4468 0.0455; 0.7979 0.1136 ;1 0.2652 ; 0.9787 0.4773 ; 0.9574 0.6667; 0.9574 0.8561; 0.8085 1 ; 0.7553 0.7803; 0.6489 0.697; 0.5851 0.8485 ; 0.4362 0.9167; 0.3191 0.6288; 0.234 0.75; 0.2234 0.9091];

**Pasul 2:** Pentru simplitate dorim doar 2 matrici, deoarece este mai usor de lucrat cu 2 matrici asa ca vom folosi 2 matrici, Train si Test, de dimensiunea fiecareia 36x2.

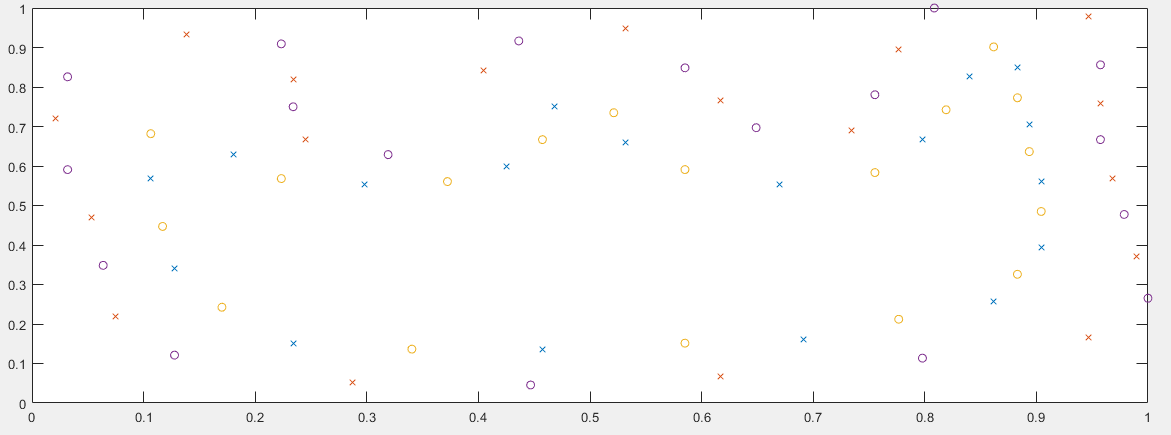
Training = [Training\_Class1;Training\_Class2];

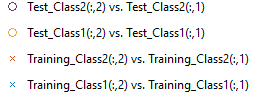
Test = [Test\_Class1;Test\_Class2];

**Pasul 3:** Vom folosi functia plot pentru a vizualiza datele.

plot(Training\_Class1, Training\_Class2, 'x', Test\_Class1, Test\_Class2, 'o');

legend('Train Cls 1','Train Cls 2','Test Cls 1','Test Cls 2')





Cu ‚o’ sunt reprezentati vectorii din clasele de Test, adica vectorii care trebuie clasificati.

Cu ‚x’ sunt reprezentati vectorii din clasele de Antrenare.

Avem si o evidenta pe culori in functie de fiecare clasa.

**Pasul 4:** Definim o functie pentru calculul distantei euclidiene intre 2 puncte. Distanta euclidiana este foarte importanta pentru metoda noastra de clasificare.

Stim formula pentru distanta euclidiana:

// pt acest exemplu X este vectorul de test, iar Z este vectorul de antrenare.

function [ dist ] = distanta( el1,el2,el3,el4 )

dist=sqrt((el3-el1)^2+(el4-el2)^2);

end

**Pasul 5:** Calculam distantele euclidiene pentru fiecare vector de test.

Adica pentru primul vector din matricea de Test [0.1064 , 0.6818] se va calcula distanta euclidiana cu fiecare vector din matricea noastra Training. In total o sa fie 36 de distante euclidiene. Dupa se trece la urmatorul vector din matricea de Test.

Urmand ca la pasul urmator sa se extraga minimul din toate aceste 36 de valori, care corespund fiecarui vector din matricea de Test.

for j=1:36

for i=1:36

v(i)=distanta(Training(i,1),Training(i,2),Test(j,1),Test(j,2));

end

**Pasul 6:** Determinarea minimului.

[X,I]=min(v);

Unde X reprezinta valoarea minimului, iar I coloana la care se gaseste valoarea respectiva.

**Pasul 7:** Clasificarea vectorului de test.

Vom folosi valoarea lui I; stim ca la inceputul programului am facut acele 2 concatenari, acuma vom pune conditia pentru clasificare.

Noi aveam 4 matrici in total, fiecare de dimensiunea 18x2, cand am concatenat 2 cate 2 (Train Class 1 + Train Class 2 si Test Class 1 + Test Class) am obtinut cele 2 matrici Train si Test fiecare avand 36x2. Deci daca minimul este gasit in prima jumatate a vectorului v, atunci vectorul clasificat apartine clasei 1, in caz contrar vectorul apartine clasei 2.

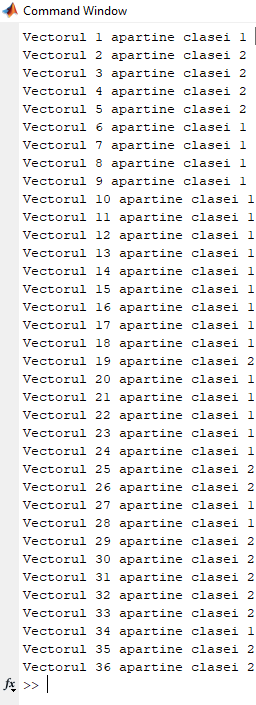
if I>18 fprintf('Vectorul %d apartine clasei 2 \n',j);

else fprintf('Vectorul %d apartine clasei 1 \n',j);

end

1. **Rezultate experimentale si concluzii**

La rularea programului avem urmatoarea clasificare:



Din baza de date PALM stim ca primii 18 vectori apartin clasei 1, iar urmatorii 18 (de la 19 la 36 inclusiv) apartin clasei 2.

Din rezultatele noastre observam ca avem 12 erori din 36 posibile, adica avem 24 clasificari corecte.

Ca sa scadem numarul erorilor ar trebui sa introducem mai multi vectori in loturile de antrenare, in momentul de fata sunt 36 de vectori sau o alta metoda de reducere a erorilor este de a modifica K-ul.

In opinia noastra 1-NN este un clasificator bun, dar depinde pentru ce este utilizat si care este scopul unui proiect, in functie de cerinta alegi un anume clasificator.

Algoritmul functioneaza corect, imi clasifica vectorul un functie de cel mai apropiat vecin.

**Exemplu:** Vectorul 2, adica [0.117 , 0.447], a fost clasificat in clasa 2 folosind 1-NN, dar acesta teoretic(adica din baza de date PALM) apartine clasei 1.

Daca cautam minimele se observa de pe grafic ca cel mai apropiat punct este [0.0532 , 0.4697], care apartine clasei 2.

Calc distanta euclidiana: sqrt((0.117 – 0.0532)^2 + (0.447 - 0.4697)^2) = 0.0677180

Urmatoarele 2 puncte cele mai apropiate sunt 2 vectori care apartin clasei 1, acestia fiind: [0.1064 , 0.5682] si [0.1277 , 0.3409], si vom calcula distanta euclidiana si pt acestia.

sqrt((0.117 - 0.1064)^2 + (0.447 - 0.5682)^2) = 0.121663

sqrt((0.117 - 0.1277)^2 + (0.447 - 0.3409)^2) = 0.106638

Din toate aceste rezultate observam ca daca folosim 1-NN, vectorul 2 va fi clasificat in clasa 2, dar daca folosim 3-NN acesta va fi clasificat in clasa1.

In concluzie pentru a imbunatati clasificarea se poate mari K-ul si sa se extinda lotul de antrenare.

1. **Pro si contra**

**PRO**

* Simplu și puternic. Nu este nevoie de adaugarea parametrilor complexi pt a construi un model.
* Usor de explicat

**CON**

* Necesita multa memorie si predictiile pot dura mult timp
* Exista algoritmi mai buni

1. **Aplicatii**

* Clasificarea caracterelor scrise de mana folosind NN cu o baza de date mare. (Smith, S.J et. al.; IEEE PAMI, 2004. Classify handwriten characters into numbers)
* Utilizarea clasificatorului k-NN pentru detectarea intruziunilor (Yihua Liao, V.Rao Vemuri; Computers and Security Journal, 2002 Classify program behavior as normal or intrusive.)
* Defectarea defectiunilor utilizand k-NN pentru procesul de fabricare a semiconductorilor (He, Q.P., Jin Wang; IEEE Transactons in Semiconductor Manufacturing, 2007 Early fault detecton in industrial systems.)
* Recuperarea rapida a imaginilor pe baza de continut folosind K-NN (z. Lu, H. Burkhardt, S. Boehmer; LNCS, 2006. CBIR (Content based image retrieval), return the closest neighbors as the relevant items to a query.)

1. **Codul utilizat**

Training\_Class1 = [0.1064 0.5682; 0.1277 0.3409; 0.234 0.1515; 0.4574 0.1364; 0.6915 0.16; 0.8617 0.2576; 0.9043 0.3939; 0.9043 0.5606; 0.8936 0.7045; 0.883 0.8485; 0.8404 0.8258; 0.7979 0.6667; 0.6702 0.553; 0.5319 0.6591; 0.4681 0.75; 0.4255 0.5985; 0.2979 0.553; 0.1809 0.6288];

Training\_Class2 = [0.1383 0.9318; 0.0213 0.7197; 0.0532 0.4697; 0.0745 0.2197; 0.2872 0.053; 0.617 0.0682; 0.9468 0.1667; 0.9894 0.3712; 0.9681 0.5682; 0.9574 0.7576; 0.9468 0.9773; 0.7766 0.8939; 0.734 0.6894; 0.617 0.7652; 0.5319 0.947 ; 0.4043 0.8409; 0.2447 0.6667; 0.234 0.8182];

Test\_Class1 =[0.1064 0.6818; 0.117 0.447; 0.1702 0.2424; 0.3404 0.1364; 0.5851 0.1515; 0.7766 0.2121; 0.883 0.3258; 0.9043 0.4848; 0.8936 0.6364; 0.883 0.7727; 0.8617 0.9015; 0.8191 0.7424; 0.7553 0.5833; 0.5851 0.5909; 0.5213 0.7348; 0.4574 0.6667; 0.3723 0.5606; 0.2234 0.5682];

Test\_Class2 = [0.0319 0.8258; 0.0319 0.5909; 0.0638 0.3485; 0.1277 0.1212; 0.4468 0.0455; 0.7979 0.1136 ;1 0.2652 ; 0.9787 0.4773 ; 0.9574 0.6667; 0.9574 0.8561; 0.8085 1 ; 0.7553 0.7803; 0.6489 0.697; 0.5851 0.8485 ; 0.4362 0.9167; 0.3191 0.6288; 0.234 0.75; 0.2234 0.9091];

Training = [Training\_Class1;Training\_Class2];

Test = [Test\_Class1;Test\_Class2];

plot(Training\_Class1, Training\_Class2, 'x', Test\_Class1, Test\_Class2, 'o');

legend('Train Cls 1','Train Cls 2','Test Cls 1','Test Cls 2')

for j=1:36

for i=1:36

v(i)=distanta(Training(i,1),Training(i,2),Test(j,1),Test(j,2));

end

[X,I]=min(v);

if I>18 fprintf('Vectorul %d apartine clasei 2 \n',j);

else fprintf('Vectorul %d apartine clasei 1 \n',j);

end

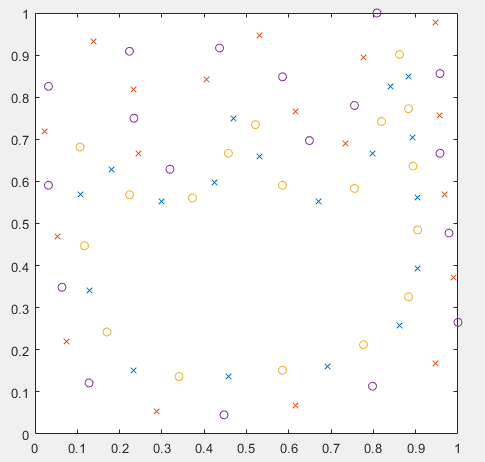
end

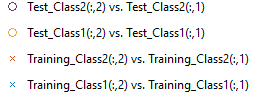
function [ dist ] = distanta( el1,el2,el3,el4 )

dist=sqrt((el3-el1)^2+(el4-el2)^2);

end

1. **Reprezentarea grafica a bazei de date PALM**





1. **Bibliografie**

<https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec06.pdf>

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0002-introduction-to-computational-thinking-and-data-science-fall-2016/lecture-slides-and-files/MIT6\_0002F16\_lec13.pdf

<https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>

<http://www.victorneagoe.com/university/prai/lab.html>

cartea: Retele Neurale pentru explorarea datelor de Victor Neagoe

<https://www.coursera.org/learn/ml-clustering-and-retrieval> (Week 2)

<https://d3c33hcgiwev3.cloudfront.net/_887cfef01c46924f8cd2a38cc099fa07_retrieval-intro-annotated.pdf?Expires=1548892800&Signature=BJWZb9vNPzlK9ZK6QukmWMkclvbXR45JCbkSx5Y0HisewmUFVGlfqhwWGohPfPuAxf~t-sA6Mft9Wzpggg-P-dAZPfQX4pF4wcCrCNmjsIjka1BuINCcEOJ-GCT1oJV8Lcxrk-tqX8BJPhrVt6NZSncHD1mWiq8jdHG8L4Cp7WI_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A>

(de pe coursera Week 2)