|  |
| --- |
| انواع طبقه بندی ها |
| پروژه ششم |
| علی رضاقلی زاده 610389126 |

**چکیده**

یکی از مراحل پردازش و تشخیص یک تصویر بعد از بدست آوردن بردار ویژگی ، طراحی یک طبقه بند است تا بتواند یک سیگنال را بدرستی تشخیص دهد . البته طبقه بند مشخصی وجود ندارد که بتوان با قطعیت یک سیگنال را طبقه بندی کرد حتی طبقه بندی نیز موجود نیست که برای همه مسایل دارای بازدهی معین باشد و به همین دلایل باید برای هر مسأله دنبال طبقه بند یا طبقه بند های مطلوب آن مسأله بود .

پس از به دست آوردن بردارهای ویژگی نوبت به طبقه­بندی داده­ها می رسد. به طور کلی انواع طبقه­بندی کننده­ها در ابتدا بر روی گروهی از بردارهای ویژگی تحت عنوان داده­های یادگیری، آموزش می­بینند و پس از تنظیم پارامترهای مربوط به آن­ها، مرحله ارزیابی آغاز می­گردد. در این مرحله عملکرد طبقه­بندی­کننده با طبقه­بندی گروهی از بردارهای ویژگی که پیش از این در آموزش شرکت نکرده‌اند، مورد ارزیابی قرار می­گیرد. در ادامه به بررسی 4 مورد از طبقه­بندی­کننده­ها روی داده های مختلف نظیر حلزونی ، فنوم و ست ایمیج می­پردازیم. این طبقه­بندی­کننده­ها عبارتند از طبقه­بندی­کننده نزدیکترین و چندمین نزدیک ترین همسایگی، طبقه­بندی­کننده بیز و طبقه­بندی­کننده پارزن .

**کلمات کلیدی:** طبقه بند ، داده آموزش ، داده تست

**مقدمه**

به طور کلی روش­های گوناگونی برای طبقه­بندی اطلاعات وجود دارد که از نظر سرعت انجام طبقه­بندی و نحوه آن با یکدیگر متفاوت می­باشند. طبقه­بندی­کننده­ها به دو دسته باسرپرست[[1]](#footnote-1) و بدون سرپرست[[2]](#footnote-2) تقسیم می­شوند. طبقه­بندی­کننده­های باسرپرست نیازمند بخشی از داده­ها که کلاس آنها مشخص شده است (داده­های آموزشی[[3]](#footnote-3)) می­باشند تا با کمک آنها بقیه داده­ها را طبقه­بندی و برچسب­گذاری نمایند، ولی روش­های بدون سرپرست تعداد ورودی­های کمتری لازم دارند و تنها داده­هایی که می­بایست طبقه­بندی شوند را به عنوان ورودی دریافت می­کنند. که ما در این گزارش طبقه بند های با سرپرست نظیر نزدیک ترین همسایه ، چندمین همسایه ، بیز و پارزن را مورد بررسی قرار داده و پیاده سازی می کنیم .

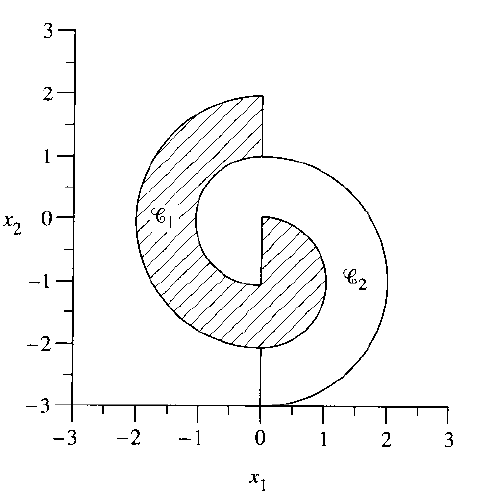
طبقه بندی با سرپرست به سه دسته تقسیم می شوند : دسته اول طبقه بندی ها ، بر اساس شباهت [[4]](#footnote-4) ؛ دسته دوم بر اساس روش های آماری[[5]](#footnote-5) و دسته سوم بر اساس روش شبکه های عصبی[[6]](#footnote-6) است . نزدیک ترین و چندمین نزدیک ترین همسایه در دسته اول و بیز و پارزن جزء دسته دوم قرار می گیرند .

نشان خواهیم داد که از لحاظ سرعت طبقه بندی ، نزدیک ترین و چندمین نزدیک ترین همسایه عملکرد بسیار ضعیف تری دارند .

داده حلزونی :

500 داده از هر کلاس انتخاب می کنیم که 200 داده از آن را برای آموزش و 300 داده را برای تست بر می داریم (از هر کلاس).

قابل ذکر است که کلاس داده های تستمان را داریم ولی طبقه بند ها این را نمی دانند و در آخر که روی آنها طبقه بند را زدیم برای بدست آوردن بازدهی آن طبقه بند مورد استفاده قرار می گیرد .



شکل 1 قسمت هاشور خورده کلاس 1 است .

**انواع طبقه بند ها روی داده ها و مقایسه آنها**

1. **نزدیک ترین و K امین نزدیک ترین همسایه**

روش KNN قادر است از میان داده­های گوناگون که هر یک با یک مجموعه از بردارهای ویژگی مشخص می­گردند، K داده که

به داده مورد بررسی نزدیکترند را انتخاب کرده و سپس با توجه به کلاس در برگیرنده اکثریت داده­های انتخاب شده، تصمیم نهایی برای طبقه­بندی بردار مورد بررسی را اتخاذ نماید. مقدار K در این روش همواره عددی انتخاب می شود که منجر به

بهترین نتیجه طبقه­بندی برای داده‌های آموزش می­گردد و سپس این مقدار برای طبقه­بندی داده­های آزمایش نیز مورد استفاده قرار می­گیرد. معیاری که برای سنجش فاصله بین دو بردار در این پایان­نامه به کار گرفته شده است، فاصله اقلیدسی می­باشد.



a

b

شکل 2 (a) و(b) به ترتیب 500 داده انتخابی از کلاس 1 و2 است .



شکل 4 داده های تست هستند که با طبقه بند K امین نزدیک ترین همسایه (K=1) طبقه بندی شده اند .از 600 داده تست ، 581 داده به درستی و 19 داده به اشتباه طبقه بندی شده اند (بازدهی 96.83 درصدی)؛ نقطه ها معرف دادهایی اند که به طور صحیح طبقه بندی شده اند وستاره معرف طبقه بندی اشتباه داده ها است .

شکل 3 (a) داده آموزش و (b) داده تست

a b

همان طور که دیده می شود در این روش ( اولین نزدیک ترین همسایه) احتمال خطا در داده های روی مرز خیلی بیشتر است حتی می توان گفت که در این روش تنها داده های نزدیک مرز ممکن است به اشتباه طبقه بندی شوند .



شکل 5 طبقه بندی داده ی شکل 3 (a) با طبقه بند (K=2) K امین نزدیک ترین همسایه ؛ در این طبقه بند از 600 داده ی تست ، 572 داده به درستی طبقه بندی شده اند و 28 داده به نادرستی (بازدهی 95.33 درصدی) که داده های درست در شکل با نقطه و داده های اشتباه طبقه بندی شده با ستاره نشان داده شده است .



شکل 6 طبقه بندی داده ی شکل 3 (a) با طبقه بند (K=3) K امین نزدیک ترین همسایه ؛ در این طبقه بند از 600 داده ی تست ، 680داده به درستی طبقه بندی شده اند و 20 داده به نادرستی (بازدهی 96.67درصدی) که داده های درست در شکل با نقطه و داده های اشتباه طبقه بندی شده با ستاره نشان داده شده است .

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | K |
| 96.33 | 96.00 | 96.00 | 95.50 | 96.50 | 95.83 | 96.67 | 95.33 | 96.83 | Performance |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | K |
| 95.50 | 94.83 | 95.33 | 94.83 | 95.50 | 95.83 | 96.00 | 96.67 | 96.17 | Performance |

**جدول 1**  بازدهی k امین نزدیک ترین همسایگی با تغییر k

که بین k های 1 تا 400 بررسی شده بازدهی k=1 بالا ترین محاسبه شده است .

پس می توان نتیجه گرفت که در طبقه بند k امین همسایه ی نزدیک احتمال خطا در داده های مرزی از بقیه داده ها بیشتر است همچنین سرعت محاسبه ی این طبقه بند به دلیل نیاز به داشتن فاصله از تمام داده های آموزش برای هر داده تست ، بسیار پایین ارزیابی می شود .

1. **طبقه بند بیز**

این طبقه بند بر اساس تحلیل آماری روی داده های آموزش در هر کلاس برای هر داده جدید (داده ی تست ) قضاوت می کند بدین گونه که احتمال تعلق داده ی جدیدنسبت به همه ی کلاس ها را بدست می آورد سپس این مقدار برای هر کلاس که بیشتر بود داده جدید را به آن کلاس برچسب گذاری می کنیم .



(2-1)

:احتمال تعلق داده  به کلاس  است .

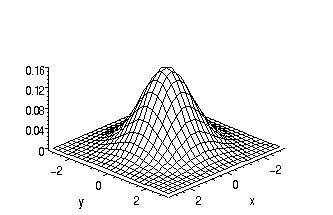
: احتمال داده  در کلاس  است .

: احتمال انتخاب کلاس  از میان بقیه ی کلاس ها .

: احنمال قرار گرفتن  در دامنه ی (فضای) کلاس ها .

که برای بدست آوردن  لازم است که تابع توزیع آن را داشته باشیم که این مستلزم آن است که ما کل فضای کلاس را داشته باشیم در حالی که ما جز تعدادی داده ی آموزش چیز دیگری از آن کلاس نداریم .

برای این مشکل ، می آیند تابع توزیع را تخمین می زنند که ما در اینجا با تابع گوسی این کار را انجام می دهیم .

 در شکل 7 نیز نمونه ای از یک تابع گوسی در دو بعد با میانگین (0 ,0) را نشان می دهد .

شکل 7 یک تابع گوسی در دو بعد را نشان می دهد .



شکل 8 (b) بعد از اعمال طبقه بندی بیس روی داده تست (a) است . داده های غلط طبقه بندی شده در (b) با ستاره (163 داده) و داده های صحیح (437 داده ) با نقطه نشان داده شده اند .

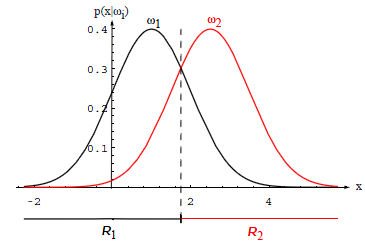
b

a

|  |  |  |
| --- | --- | --- |
| class 2 | Class 1 |  |
| 0.6817 | -0.6103 | Fiture 1 |
| -0.7632 | -0.2018 | fiture 2 |

**جدول 2** میانگین هر کلاس (داده های ما در حلزونی دو بعدی اند پس دارای دو ویژگی اند ) .

در این روش طبقه بندی داده هایی به طور اشتباه مورد قضاوت قرار می گیرند که داخل ناحیه ی تداخل تابع های توزیع تخمین زده از هر کلاس باشد یعنی مطابق شکل 9 داده ی مورد نظر در داخل بازه ی حدودا صفر تا چهار قرار گیرد در این صورت احتمال خطا تشخیص دادن توسط طبقه بند وجود دارد ؛ در مثال شکل 9 اگر داده ای که در واقع عضو کلاس 1 است در ناحیه ی دو تا چهار یا داده ای که در واقع عضو کلاس 2 است در ناحیه ی صفر تا دو قرار گیرد درآن صورت این داده ها به اشتباه به کلاس دیگری طبقه بندی می شود .

طبقه بند بیز روی این داده بازدهی 72.83 درصدی را داراست

شکل 9 دو تابع گوسی به عنوان تخمینی برای تابع توزیع دو کلاس w1 و w2

1. **طبقه بند پارزن**

روش پارزن روشي ناپارامتري براي تخمين تابع چگالي احتمال است. در اين روش كه با تعميم روش هيستوگرام بدست آمده است، به صورت محلي تابع چگالي احتمال را تقريب ميزنيم. اساس کار این روش مشابه هیستوگرام است با این تفاوت که ما در این روش خود را مستقیما با داده های با ابعاد بالا درگیر می کنیم. روش کار به صورت زیر است:

گام اول- یک ابر مکعب با طول ضلع را در نظر بگیرید. حجم این مکعب برابر با که بعد فضای داده ها (در این مساله برابر با 2) است.

گام دوم- برای هر نقطه  در تخمین pdf، مرکز این مکعب را بر روی داده مورد نظر قرار می دهیم.

گام سوم- تعداد نمونه های واقع شده در این مکعب را می شمریم. چنانچه این تعداد برابر با باشد، تخمین ما به صورت زیر در می آید:



که در آن  تعداد نمونه ها است. تابع  می تواند مثلثی، گوسی و یا هر تابع دلخواه دیگر باشد، مشروط بر آنکه مساحت زیر آن برابر با یک باشد.



a



b

شکل 10 (b) بعد از اعمال طبقه بندی پارزن روی داده تست (a) است . داده های غلط طبقه بندی شده در (b) با ستاره (277 داده) و داده های صحیح (323 داده ) با نقطه نشان داده شده اند

**4. ضمیمه**

**Code 1: K \_Nearest neighbor**

function ClassifiedSet=K\_NearestN(TrainSet,TestSet,K,NumberOfClass)

%TrainSet is Train point in each class (each point is in row)

%TestSet is n\*2 matrix that you must never confident on second column

%(first column is point(in each dimention)).

K\_SelectedPointSet=[]; %is k\*2 matrix that store point in first column and Label in second.

% DistanceFromK\_Selected=[]; %respectivly to K\_NearestPointSet / is k\*1 matrix

ClassifiedSet=[];

% LocalD=0;

% %LocalD is Max Distance of point from K selected as nearest point of Training Set

% pivot=[TestSet(1,1:end-1)];

% %pivot is one of the Test point . we want just find distance this piont

% %from other Training Set ,and use these to finde distance for other Test

% %points. BUT AT LAST WE DONT USE THIS THINKING BECOUSE OF MORE TIME

% %NESESSERY TO COMPUTE.

% DistanceFromPivot=[]; %is (number of TrainSet)\*3 that store distance of

% %correspond point(that store in first column)from Pivot in third column

% %and second column is label of correspond point in column 1.

% d=1;

%---------------Find K nearest point from PIVOT--------------

% for i=1:size(TrainSet,1)

% d=sqrt(sum((pivot(1,:)-TrainSet(i,1:end-1)).^2));

% if LocalD > d

% replaceNum=max(DistanceFromK\_Selected);

% b=(DistanceFromK\_Selected == replaceNum);

% DistanceFromK\_Selected(b)=d;

% index=transpose([1:K]);

% ind=index(b);

% K\_SelectedPointSet(ind(1,1),:)=TrainSet(i,:);

%

% LocalD=max(DistanceFromK\_Selected);

%

% end

% if i <= K

% DistanceFromK\_Selected=[DistanceFromK\_Selected;d];

% K\_SelectedPointSet=[K\_SelectedPointSet;TrainSet(i,:)];

% if i==K

% LocalD=max(DistanceFromK\_Selected);

% end

% end

% DistanceFromPivot=[DistanceFromPivot;TrainSet(i,:) d];

% end

% ClassifiedSet=LabelUnKnownPoint(pivot,K\_SelectedPointSet,ClassifiedSet,NumberOfClass);

DistanceFromThis=[]; %Distance of other Traning point From current

%Test point. DistanceFromThis is n\*3 that first column place Train point

%label and distance also place in second and third respectively

for j=1:size(TestSet,1)

point(1,:)=TestSet(j,1:end-1);

%---------------Find K nearest point--------------

for i=1:size(TrainSet,1)

d=sqrt(sum((point(1,:)-TrainSet(i,1:end-1)).^2));

DistanceFromThis=[DistanceFromThis;TrainSet(i,1:end) d];

end

dis=DistanceFromThis(:,end);

[B,IX]=sort(dis,1);

SortedDFT=zeros(size(DistanceFromThis,1),size(DistanceFromThis,2));

for l=1:size(DistanceFromThis,1)

SortedDFT(l,:)=DistanceFromThis(IX(l),:);

end

SortedDFT;

for h=1:K

K\_SelectedPointSet=[K\_SelectedPointSet;SortedDFT(h,1:end-1)];

end

%K\_SelectedPointSet

ClassifiedSet=LabelUnKnownPoint(point,K\_SelectedPointSet,ClassifiedSet,NumberOfClass);

%this function do labeling by means of K selected point's label

DistanceFromThis=[];

K\_SelectedPointSet=[];

end

end

function ClassifiedSet=LabelUnKnownPoint(point,K\_SelectedPointSet,ClassifiedSetIn,NumberOfClass)

%K\_SelectedPointSet is k\*2 matrix that store point in first column and

%Label in second.

%with assumption that label of each class are from 1 to NumberOfClass

%(Note in 1).

LabelArray=K\_SelectedPointSet(:,end);

NumberInEachClass=zeros(NumberOfClass,1); %is column matrix that stor

%number of piont in each class(WhatPoint is in K\_SelectedPointSet).

for i=1:NumberOfClass

NumberInEachClass(i,1)=sum(LabelArray==i);

end

ClassLabel=LaUnPoByK\_NNStrategy(NumberInEachClass);

ClassifiedSetIn=[ClassifiedSetIn;point ClassLabel];

ClassifiedSet=ClassifiedSetIn;

end

function ClassLabel=LaUnPoByK\_NNStrategy(NumberInEachClass)

%NumberInEachClass is single column that show number of point in

%each(1,2,..)class(correspond to its index).

%class

index=transpose([1:size(NumberInEachClass,1)]);

CL=index(NumberInEachClass==max(NumberInEachClass));

ClassLabel=CL(randi([1:size(CL,1)],1),1);

end

**Code 2: Bayes**

function ClassifiedSet=ClassifyByBayes(TrainSet,TestSet,NumberOfClass)

%-----------separate each class of Training Set-----------

Classes=cell(1,NumberOfClass);

for i=1:NumberOfClass

Classes{1,i}=cell(4,1);%Store Training point in each class

%correspond to its column index(for instance i's class store in

%classes{1,i}{1,1})and Mean of ech class(store in Classes{1,i}{2,1})and

%variance is stored in Classes{1,i}{3,1}.Each of these(mean and

%variance are column matrix.

end

for i=1:NumberOfClass

LCurrclass=(TrainSet(:,end)==i); %label of current class

Currclass=TrainSet(LCurrclass,1:end-1); %with out label

Classes{1,i}{1,1}=Currclass(:,1:end); %stor with out label

%a=Classes{1,i}{1,1}

end

%-----------Find mean of each class------------------------

Sum=0;

for i=1:NumberOfClass

Points=Classes{1,i}{1,1}; %each point is in row

Sum=sum(Points); %execute sum of each dimention

Mean=Sum./(size(Points,1)) %execute Mean of each dimention

%Mean is (dimention of point)\*1.

Classes{1,i}{2,1}=Mean;

MeanSet=[]; %at last became same size of Points matrix to excecute legaly

for j=1:size(Points,1)

MeanSet=[MeanSet;Mean];

end

Variance =sum((Points-MeanSet).^2);

Variance=(Variance./size(Points,1));

SD=Variance.^0.5; % SD is row matrix

Classes{1,i}{4,1}=SD;

Sum=0;

end

for i=1:NumberOfClass

Points=Classes{1,i}{1,1};

Cov=Covariancefunc(Points); %Covariancefunc:calculate covariance

%therough each column (dimention is number of column);Cov is

%(dim\*dim) matrix.

Classes{1,i}{3,1}=Cov;

end

%---------------------- classifying Test points-------------

index=[1:NumberOfClass];

ClassifiedSet=[];

for i=1:size(TestSet,1)

currPoint=transpose(TestSet(i,1:end-1)); %currPoint is column matrix

Posterior=zeros(1,NumberOfClass); %THIS IS ROW MATRIX THAT SRORES posterior of each class

for j=1:NumberOfClass

Cov=Classes{1,j}{3,1};

po=Classes{1,j}{1,1};

d=size(po,2);

m=transpose(Classes{1,j}{2,1}); %naw SD is column matrix

Pi=1/(NumberOfClass); %Pi is prior for i-th class

P=-(d/2)\*log(2\*pi)-0.5\*log(det(Cov))-0.5\*(transpose(currPoint-m))\*(inv(Cov)\*(currPoint-m))+log(Pi);

Posterior(1,j)=P;

end

c=index(Posterior == max(Posterior));

ClassifiedSet=[ClassifiedSet;transpose(currPoint) c];

end

end

**Code 3: Parzen**

function Classified=Parzen(TrainSet,TestSet,r,NumberOfClass)

ClassifiedSet=[];

for i=1:size(TestSet,1)

Point=TestSet(i,1:end-1);

NearestSet=[];

for j=1:size(TrainSet,1)

TrainPoint=TrainSet(j,1:end-1);

dis=sqrt(sum((Point-TrainPoint).^2));

if dis <= r

NearestSet=[NearestSet;TrainPoint TrainSet(j,end)];

end

end

ClassifiedSet=ParzenDecision(NearestSet,ClassifiedSet,Point,NumberOfClass);

end

Classified=ClassifiedSet;

end

function Classified=ParzenDecision(NearestSet,ClassifiedIn,Point,NumberOfClass)

%in this function we use Parzen method (we construct normal destribute

%for each point of one class and then we sum them.

%NearestSet consist of real label of each point at end column .

label=NearestSet(:,end);

Px=zeros(1,NumberOfClass);

for i=1:NumberOfClass

ThisClass=(label==i);

Class=NearestSet(ThisClass,1:end-1); % find each point of i's class.

%points in Class is with out label

Px(1,i)=Priority(Class,Point,NumberOfClass);

end

m=Px(1,1);

index=[];

for h=1:NumberOfClass

if Px(1,h) > m

ind=h;

index=[];

index=[ind];

m=Px(1,h);

end

if Px(1,h) == m

index=[index;h];

end

end

if size(index,1)==1

y=1;

else

y=randi([1,size(index,1)],1);

end

c=index(y,1);

ClassifiedIn=[ClassifiedIn;Point c];

Classified=ClassifiedIn;

end

function g=Priority(Class,Point,NumberOfClass)

%assume gousian function for each point in class and gain probablity of

%being 'Point' in this class

g=0;

Point=transpose(Point);

for i=1:size(Class,1)

Xi=Class(i,:);

m=transpose(Xi);

Cov=0.05.\*[0.5 0;0 0.5];

d=size(Xi,2);

Pi=1/(NumberOfClass);

g=g+(-(d/2)\*log(2\*pi)-0.5\*log(det(Cov))-0.5\*(transpose(Point-m))\*(inv(Cov)\*(Point-m))+log(Pi));

end

end

# References

### [1] [RO Duda, PE Hart, and DG Stork, Pattern Classification, New York](http://dl.acm.org/citation.cfm?id=1325635)

1. Supervised Classifier [↑](#footnote-ref-1)
2. Unsupervised Classifier [↑](#footnote-ref-2)
3. Train Data [↑](#footnote-ref-3)
4. Consept of similarity [↑](#footnote-ref-4)
5. Probabilistic approach [↑](#footnote-ref-5)
6. Neural Networks [↑](#footnote-ref-6)