Data Mining and Machine learning

Naïve Bayes classifier

Edgar Acuna
Department of Mathematical Science
academic.uprm.edu/eacuna

How to build a Bayes Classifier

- Assume you want to predict output Y which has arity n_Y and values v_1, v_2, … v_{ny}.
- Assume there are m input attributes called X_1, X_2, … X_m
- Break dataset into n_Y smaller datasets called DS_1, DS_2, … DS_{ny}.
- Define DS_i = Records in which $Y=v_i$
- For each DS_i , learn Density Estimator M_i to model the input distribution among the $Y=v_i$ records.

How to build a Bayes Classifier

- Assume you want to predict output Y which has arity n_Y and values v_1, v_2, … v_{ny}.
- Assume there are m input attributes called X_1, X_2, … X_m
- Break dataset into n_Y smaller datasets called DS_1, DS_2, … DS_{ny}.
- Define DS_i = Records in which $Y=v_i$
- For each DS_i , learn Density Estimator M_i to model the input distribution among the $Y=v_i$ records.
- M_i estimates $P(X_1, X_2, \ldots X_m \mid Y=v_i)$

How to build a Bayes Classifier

- Assume you want to predict output Y which has arity n_Y and values $v_1, v_2, \ldots v_{ny}$.
- Assume there are m input attributes called $X_1, X_2, \ldots X_m$
- Break dataset into n_Y smaller datasets called $DS_1, DS_2, \ldots DS_{ny}$.
- Define DS_i = Records in which $Y=v_i$
- For each DS_i , learn Density Estimator M_i to model the input distribution among the $Y=v_i$ records.
- M_i estimates $P(X_1, X_2, \ldots X_m \mid Y=v_i)$

- Idea: When a new set of input values ($X_1 = u_1, X_2 = u_2, \ldots. X_m = u_m$) come along to be evaluated predict the value of Y that makes $P(X_1, X_2, \ldots X_m \mid Y=v_i)$ most likely

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)$$

Is this a good idea?

How to build a Bayes Classifier

- Assume you want to predict output Y which has arity n_Y and values $v_1, v_2, \ldots v_{ny}$.
- Assume there are m input attributes callec
- Break dataset into n_Y smaller datasets call
- Define DS_i = Records in which $Y=v_i$
- For each DS_i , learn Density Estimator M_i distribution among the $Y=v_i$ records.
- M_i estimates $P(X_1, X_2, \ldots X_m \mid Y=v_i)$

Much Better Idea

- Idea: When a new set of input values $(X_1 = u_1, X_2 = u_2, \ldots. X_m = u_m)$ come along to be evaluated predict the value of Y that makes $P(Y=v_i \mid X_1, X_2, \ldots X_m)$ most likely

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(Y = v \mid X_1 = u_1 \cdots X_m = u_m)$$

Is this a good idea?

Bayes Classifiers

1. Learn the distribution over inputs for each value Y.
2. This gives P(X_1, X_2, … X_m / $Y=v_i$).
3. Estimate P($Y=v_i$). as fraction of records with $Y=v_i$.
4. For a new prediction:

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(Y=v \mid X_1 = u_1 \cdots X_m = u_m)$$

$$= \underset{v}{\operatorname{argmax}}\, P(X_1 = u_1 \cdots X_m = u_m \mid Y=v) P(Y=v)$$

Bayes Classifiers

1. Learn the distribution over inputs for each value Y.
2. This gives P(X_1, X_2, … X_m / $Y=v_i$).
3. Estimate P($Y=v_i$). as fraction of records w… $Y=v_i$.
4. For a new prediction:

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(Y = v \mid X_1 = \cdots$$

$$= \underset{v}{\operatorname{argmax}}\, P(X_1 = u_1 \cdots X_m = u_m \cdots$$

We can use our favorite Density Estimator here.
But the easer and faster is
•Naïve Density Estimator

Naïve Bayes Classifier

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v) P(Y = v)$$

In the case of the naive Bayes Classifier this can be simplified:

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(Y = v) \prod_{j=1}^{n_Y} P(X_j = u_j \mid Y = v)$$

In here, the random variables X1, ….. Xm are considered independent

Naïve Bayes Classifier

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(X_1 = u_1 \cdots X_m = u_m \mid Y = v) P(Y = v)$$

In the case of the naive Bayes Classifier this can be simplified:

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(Y = v) \prod_{j=1}^{n_Y} P(X_j = u_j \mid Y = v)$$

If you have a lot of input attributes that product will underflow in floating point math. You should use logs:

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} \left(\log P(Y = v) + \sum_{j=1}^{n_Y} \log P(X_j = u_j \mid Y = v) \right)$$

Naïve Bayes classifier

Naïve Bayes classifiers can be applied when there are continuous attributes in the dataset. There are two options:

a) Discretize the continuous attributes using methods such as, Equal width discretization, Equal frequency discretization and, Entropy with MDLP discretization. Scikit learn has two function BernoulliNB and MultinomilaNB.

b) Assume a Gaussian distribution for the continuous predictors Scoit learn has a GaussianNB function.

The option a) is preferred by many.

Example

X1	X2	X3	Y
0	0	1	0
0	1	0	0
1	1	0	0
0	0	1	1
1	1	1	1
0	0	1	1
1	1	0	1

Example: Cont

To which class will be assigned the record (0,0,1)?

$$P(Y=0)=3/7 \qquad P(Y=1)=4/7$$

$$P(X_1=0,X_2=0,X_3=1/Y=0)=P(X_1=0/Y=0)P(X_2=0/Y=0)P(X_3=1/Y=0)=$$

$$(2/3)(1/3)(1/3)=2/27$$

$$P(X_1=0,X_2=0,X_3=1/Y=1)=P(X_1=0/Y=1)P(X_2=0/Y=1)P(X_3=1/Y=1)=$$

$$(2/4)(2/4)(3/4)=3/16$$

X1=0,X2=0, x3=1 will be assigned to class 1

Example 2. (continuous and discrete attributes)

X1	X2	X3	X4	Y
0	0	1	3.15	0
0	1	0	8.17	0
1	1	0	5.72	0
0	0	1	7.16	1
1	1	1	9.32	1
0	0	1	12.81	1
1	1	0	15.48	1

Example 2. (cont.)

- #Metodo 1. Discretizando la columna 4

```
> dnaiveeje2
     col1 col2 col3 col4 col5
[1,]    0    0    1    1    0
[2,]    0    1    0    1    0
[3,]    1    1    0    1    0
[4,]    0    0    1    1    1
[5,]    1    1    1    2    1
[6,]    0    0    1    2    1
[7,]    1    1    0    2    1
```

- #Metodo 2. Sin discretizar la columna 4
- Media y desviacion estandar de la col4 en cada clase

> mean(naiveeje2[naiveeje2[,5]==0,4])

- [1] 5.68
- > mean(naiveeje2[naiveeje2[,5]==1,4])
- [1] 11.1925

Example 2. (cont.)

- > sd(naiveeje2[naiveeje2[,5]==0,4])
- [1] 2.510239
- > sd(naiveeje2[naiveeje2[,5]==1,4])
- [1] 3.686293

```
> # a que clase sera asignado el vector(0,0,1,4.25)?
```

Hay que calcular P[X1=0/Y=0]P[X2=0/Y=0]P[x3=1/Y=0]f(x4=425/Y=0][Y=0]
y compararla con
P[X1=0/Y=1]P[X2=0/Y=1]P[x3=1/Y=1]f(x4=425/Y=1][Y=1]

```
> (2/27)*dnorm(4.25,5.68,2.5102)*3/7
[1] 0.009030122
> (3/16)*dnorm(4.25,11.1925,3.6862)*4/7
[1] 0.003195506
```

Luego el vector sera asignado a la clase 0.

Example 2. (cont.)

>pred=predict(a,naivebayes21[,-5],type="raw")

- pred1=max.col(pred)
- > pred1
- [1] 1 1 1 2 2 2 2

> table(pred1,naivebayes21[,5])

```
Pred1   0 1
      1 3 0
      2 0 4
```

Error=0

Naïve Bayes for Diabetes

Without Discretization

```
# Calculo de las probabilidades posteriores y de las clases predichas
clf = GaussianNB()
clf.fit(X1,y1)
proba=pd.DataFrame(clf.predict_proba(X1))
pred=clf.predict(X1)
print 'Accuracy by Resubstitution',(pred==y1).sum()/float(768)
Accuracy by Resubstitution 0.76171875
# Calculo de las probabilidades posteriores y de las clases predichas
```

With Discretization; Equal width

```
from sklearn.preprocessing import KBinsDiscretizer
est = KBinsDiscretizer(n_bins=10, encode='ordinal', strategy='uniform')
est.fit(X)
Xt = est.transform(X)
clf=MultinomialNB()
clf.fit(Xt,y1)
scores = cross_val_score(clf, X1, y1, cv=10)
print("Accuracy by cross validation: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std()))
Accuracy by cross validation: 0.654 (+/- 0.048)
```

The Auto-mpg dataset

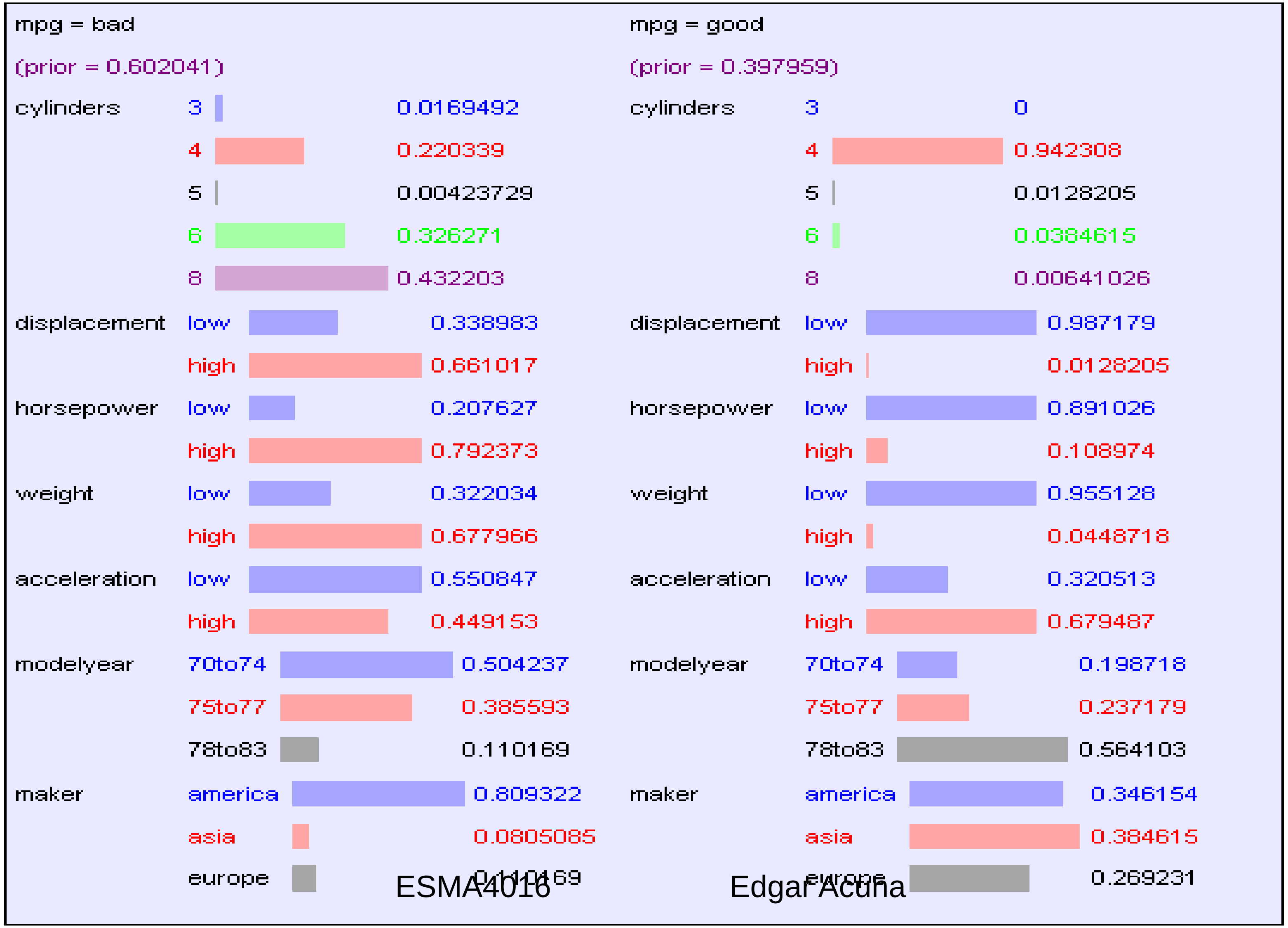
Donor: Quinlan,R. (1993)
Number of Instances: 398 minus 6 missing=392 (training: 196 test: 196):
Number of Attributes: 9 including the class attribute7. Attribute Information:
1. mpg: continuous (discretizado bad<=25,good>25)
2. cylinders: multi-valued discrete
3. displacement: continuous (discretizado low<=200, high>200)
4. horsepower: continuous ((discretizado low<=90, high>90)
5. weight: continuous (discretizado low<=3000, high>3000)
6. acceleration: continuous (discretizado low<=15, high>15)
7. model year: multi-valued discrete (discretizado 70-74,75-77,78-82
8. origin: multi-valued discrete
9. car name: string (unique for each instance)
Note: horsepower has 6 missing values

Available at: https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data

The auto-mpg dataset

18.0 8 307.0 130.0 3504. 12.0 70 1 "chevrolet chevelle malibu"
15.0 8 350.0 165.0 3693. 11.5 70 1 "buick skylark 320"
18.0 8 318.0 150.0 3436. 11.0 70 1 "plymouth satellite"
16.0 8 304.0 150.0 3433. 12.0 70 1 "amc rebel sst"
17.0 8 302.0 140.0 3449. 10.5 70 1 "ford torino"
……………………………………………………………….
27.0 4 140.0 86.00 2790. 15.6 82 1 "ford mustang gl"
44.0 4 97.00 52.00 2130. 24.6 82 2 "vw pickup"
32.0 4 135.0 84.00 2295. 11.6 82 1 "dodge rampage"
28.0 4 120.0 79.00 2625. 18.6 82 1 "ford ranger"
31.0 4 119.0 82.00 2720. 19.4 82 1 "chevy s-10"

Resultados del clasificador NB para "MPG": 392 records



NB results for "mpg"

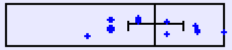
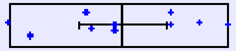
- #without discretization:
- > b=naiveBayes(mpg~.,data=autompg)

```
> pred=predict(b,autompg[,-1],type="raw")
> pred1=max.col(pred)
> table(pred1,autompg[,1])
pred1   1   2
    1 180   8
    2  56 148
> error=64/392
> error
[1] 0.1632
>#with manual discretization
> b=naiveBayes(mpg~.,data=autompg2)
> pred=predict(b,autompg2[,-1])
> table(pred,autompg2[,1])
pred   1   2
   1 182   7
   2  54 149
> 61/392
[1] 0.1556122
```

BC Results: “MPG”: 392 records

mpg = bad			mpg = good		
(prior = 0.602041)			(prior = 0.397959)		
cylinders	3	0.0169492	cylinders	3	0
	4	0.220339		4	0.942308
	5	0.00423729		5	0.0128205
	6	0.326271		6	0.0384615
	8	0.432203		8	0.00641026
displacement	low	0.338983	displacement	low	0.987179
	high	0.661017		high	0.0128205
horsepower	low	0.207627	horsepower	low	0.891026
	high	0.792373		high	0.108974
weight	low	0.322034	weight	low	0.955128
	high	0.677966		high	0.0448718
acceleration	low	0.550847	acceleration	low	0.320513
	high	0.449153		high	0.679487
modelyear	70to74	0.504237	modelyear	70to74	0.198718
	75to77	0.385593		75to77	0.237179
	78to83	0.110169		78to83	0.564103
maker	america	0.809322	maker	america	0.346154
	asia	0.0805085		asia	0.384615
	europe	0.110169		europe	0.269231

The Classifier learned by “Naive BC”

Name	Model	Parameters	FracRight			
Model1	bayesclass	density=joint submodel=gauss gausstype=general	0.885256	+/-	0.0247796	
Model2	bayesclass	density=naive submodel=gauss gausstype=general	0.852372	+/-	0.0400495	

Advantages/Disadvantages of NB

Probabilities zeros can affect the NB classifier. A correction factor is apply to to the computation of probabilities to solve this problem.

- The discretization process seems to affect the performance of the classifier.
- Naïve Bayes is very cheap. It does not have problem working with 10,000 attributes.
- Naïve Bayes is a particular case of Bayesian networks