Data Mining

Bayesian Classifier

A Quick Review of Probability

- The Axioms of Probability
  - $0 <= P(A) <= 1$
  - $P(True) = 1$
  - $P(False) = 0$
  - $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$

Theorems from the Axioms

- $0 <= P(A) <= 1$, $P(True) = 1$, $P(False) = 0$
- $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$

From these we can prove:

$$P(\text{not } A) = P(\sim A) = 1 - P(A)$$

$$P(A) = P(A \text{ and } B) + P(A \text{ and } \sim B)$$

Multivalued Random Variables

- Suppose A can take on more than 2 values
- A is a *random variable with arity k* if it can take on exactly one value out of *{v_1, v_2, .. v_k}*
- Thus…

$$P(A = v_i \ \ and\ A = v_j) = 0 \text{ if } i \neq j$$

$$P(A = v_1 \ \text{ or } \ A = v_2 \ \cdots \text{ or } \ A = v_k) = 1$$

An easy fact about Multivalued Random Variables

- Using the axioms of probability…

  0 <= P(A) <= 1, P(True) = 1, P(False) = 0

  P(A or B) = P(A) + P(B) - P(A and B)

- And assuming that A obeys…

$$P(A = v_i \wedge A = v_j) = 0 \text{ if } i \neq j$$

$$P(A = v_1 \vee A = v_2 \vee \ldots \vee A = v_k) = 1$$

- It's easy to prove that

$$P(A = v_1 \vee A = v_2 \vee \ldots \vee A = v_i) = \sum_{j=1}^{i} P(A = v_j)$$

- And thus we can prove

$$\sum_{j=1}^{k} P(A = v_j) = 1$$

Another fact about Multivalued Random Variables:

- Using the axioms of probability…

  0 <= P(A) <= 1, P(True) = 1, P(False) = 0

  P(A or B) = P(A) + P(B) - P(A and B)

- And assuming that A obeys…

$$P(A = v_i \text{ and } A = v_j) = 0 \text{ if } i \neq j$$

$$P(A = v_1 \text{ or } A = v_2 \text{ or } A = v_k) = 1$$

- It can be proved that

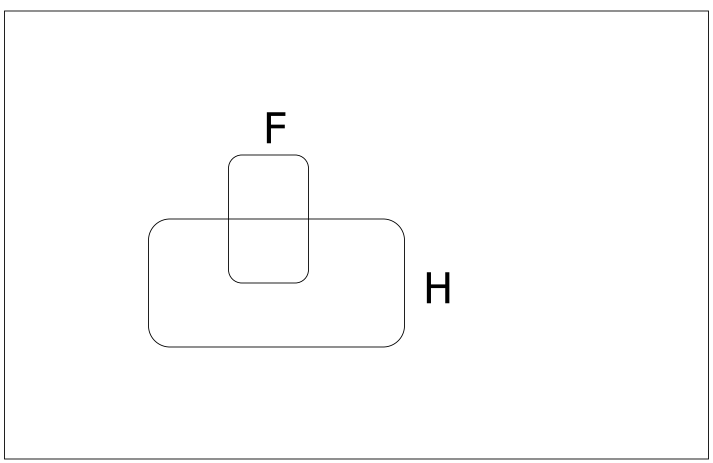
$$P(B \text{ and } [A = v_1 \text{ or } A = v_2 \text{ or } A = v_i]) = \sum_{j=1}^{i} P(B \text{ and } (A = v_j))$$

- And thus we can prove

$$P(B) = \sum_{j=1}^{k} P(B \text{ and } A = v_j)$$

Conditional Probability

- P(A|B) = Fraction of worlds in which B is true that also have A true

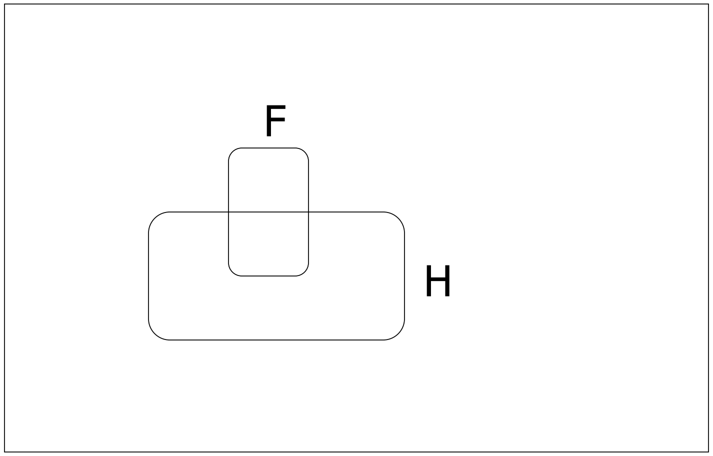


H = “Have a headache”
F = “Coming down with Flu”

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

“Headaches are rare and flu is rarer, but if you’re coming down with flu there’s a 50-50 chance you’ll have a headache.”

Conditional Probability



H = “Have a headache”
F = “Coming down with Flu”

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

P(H|F) = Fraction of flu-inflicted worlds in which you have a headache

$$= \frac{\text{\#worlds with flu and headache}}{\text{\#worlds with flu}}$$

$$= \frac{\text{Area of “H and F” region}}{\text{Area of “F” region}}$$

$$= \frac{P(H \text{ and } F)}{P(F)}$$

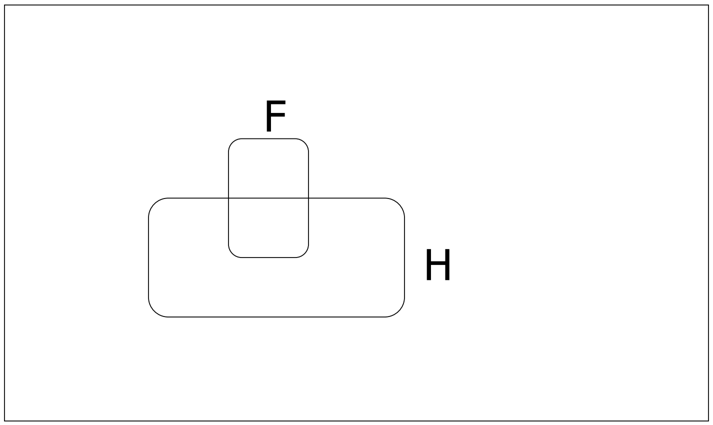
Definition of Conditional Probability

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)}$$

Corollary: The Chain Rule

$$P(A \text{ and } B) = P(A|B)\,P(B)$$

Probabilistic Inference



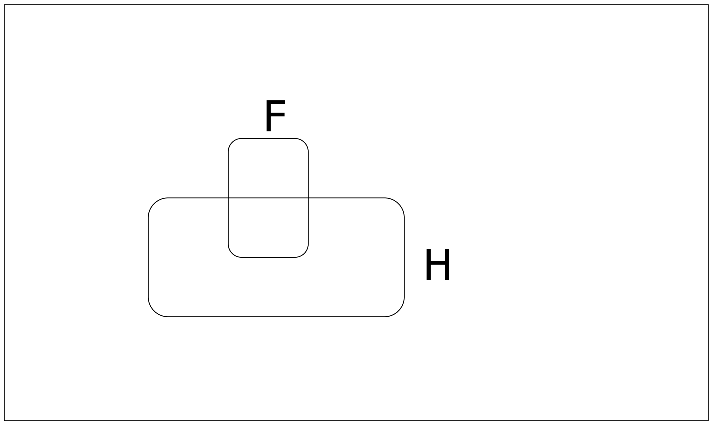
H = "Have a headache"
F = "Coming down with Flu"

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

One day you wake up with a headache. You think: "Drat! 50% of flus are associated with headaches so I must have a 50-50 chance of coming down with flu"

Is this reasoning correct?

Probabilistic Inference



H = "Have a headache"
F = "Coming down with Flu"

P(H) = 1/10
P(F) = 1/40
P(H|F) = 1/2

P(F and H) = …

P(F|H) = …

What we just did…is the Bayes Rule

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)} = \frac{P(B|A)\,P(A)}{P(B)}$$

Bayes, Thomas (1763) An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London,* **53:370-418**

More General Forms of Bayes Rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\sim A)P(\sim A)}$$

$$P(A = v_i|B) = \frac{P(B|A = v_i)P(A = v_i)}{\sum_{k=1}^{n_A} P(B|A = v_k)P(A = v_k)}$$

Useful Easy-to-prove facts

$$P(A \mid B) + P(\neg A \mid B) = 1$$

$$\sum_{k=1}^{n_A} P(A = v_k \mid B) = 1$$

The Joint Distribution

Recipe for making a joint distribution of M variables:

Example: Boolean variables A, B, C

The Joint Distribution

Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values of your variables (if there are M Boolean variables then the table will have 2^M rows).

Example: Boolean variables A, B, C

A	**B**	**C**
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

The Joint Distribution

Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values of your variables (if there are M Boolean variables then the table will have 2^M rows).
2. For each combination of values, say how probable it is.

Example: Boolean variables A, B, C

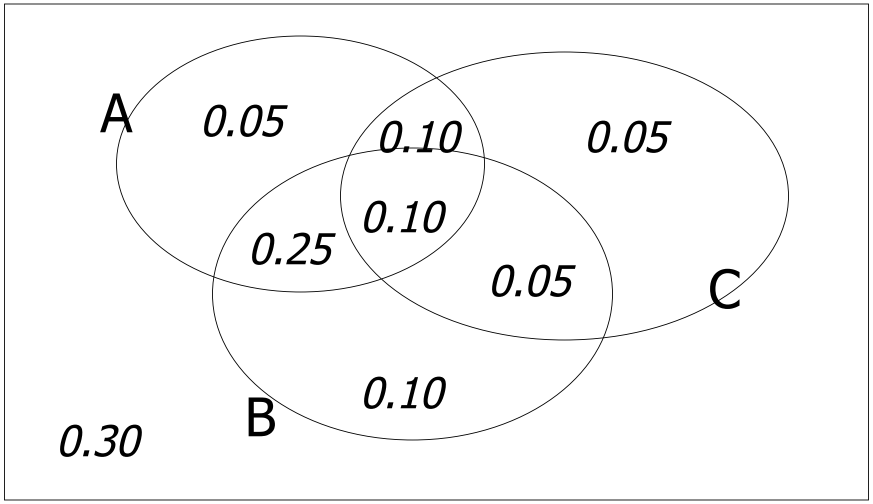
A	**B**	**C**	**Prob**
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

The Joint Distribution

Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values of your variables (if there are M Boolean variables then the table will have 2^M rows).
2. For each combination of values, say how probable it is.
3. If you subscribe to the axioms of probability, those numbers must sum to 1.

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10



Using the Joint

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

One you have the JD you can ask for the probability of any logical expression involving your attribute

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

Using the Joint

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

P(Poor Male) = 0.4654

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

Using the Joint

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

P(Poor) = 0.7604

$$P(E) = \sum_{\text{rows matching } E} P(\text{row})$$

Inference with the Joint

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

$$P(E_1 \mid E_2) = \frac{P(E_1 \; and \; E_2)}{P(E_2)} = \frac{\sum_{\text{rows matching } E_1 \text{ and } E_2} P(\text{row})}{\sum_{\text{rows matching } E_2} P(\text{row})}$$

Inference with the Joint

gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

$$P(E_1 \mid E_2) = \frac{P(E_1 \text{ and } E_2)}{P(E_2)} = \frac{\sum_{\text{rows matching } E_1 \text{ and } E_2} P(\text{row})}{\sum_{\text{rows matching } E_2} P(\text{row})}$$

P(Male | Poor) = 0.4654 / 0.7604 = 0.612

Inference is a big deal

- I' ve got this evidence. What' s the chance that this conclusion is true?
  - I' ve got a sore neck: how likely am I to have meningitis?
  - I see my lights are out and it' s 9pm. What' s the chance my spouse is already asleep?

- There' s a thriving set of industries growing based around Bayesian Inference. Highlights are: Medicine, Pharma, Help Desk Support, Engine Fault Diagnosis

Where do Joint Distributions come from?

- Idea One: Expert Humans
- Idea Two: Simpler probabilistic facts and some algebra

Example: Suppose you knew

P(A) = 0.7	P(C|A and B) = 0.1
	P(C|A and ~B) = 0.2
P(B|A) = 0.2	P(C|~A and B) = 0.3
P(B|~A) = 0.1	P(C|~A and ~B) = 0.1

Then you can automatically compute the JD using the chain rule

P(A=x and B=y and C=z) = P(C=z|A=x and B=y) P(B=y|A=x) P(A=x)

What is P(A, B, ~C)?

Where do Joint Distributions come from?

- Idea Three: Learn them from data!

Prepare to see one of the most impressive learning algorithms you' ll come across in the entire course….

Learning a joint distribution

Build a JD table for your attributes in which the probabilities are unspecified

A	B	C	Prob
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

The fill in each row with

$$\hat{P}(\text{row}) = \frac{\text{records matching row}}{\text{total number of records}}$$

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	**0.25**
1	1	1	0.10

Fraction of all records in which A and B are True but C is False

Example of Learning a Joint

- This Joint was obtained by learning from three attributes in the UCI "Adult" Census Database [Kohavi 1995]

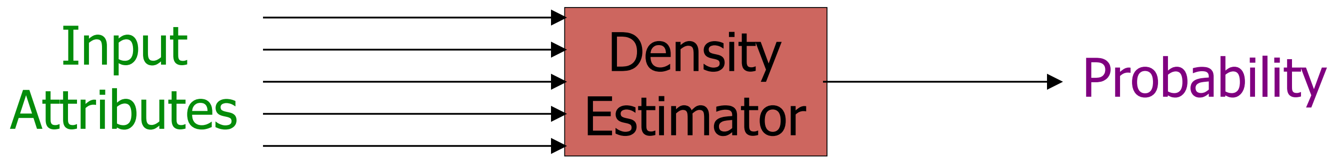
gender	hours_worked	wealth	
Female	v0:40.5-	poor	0.253122
		rich	0.0245895
	v1:40.5+	poor	0.0421768
		rich	0.0116293
Male	v0:40.5-	poor	0.331313
		rich	0.0971295
	v1:40.5+	poor	0.134106
		rich	0.105933

Where are we?

- We have recalled the fundamentals of probability
- We have become content with what JDs are and how to use them
- We know how to learn JDs from data.

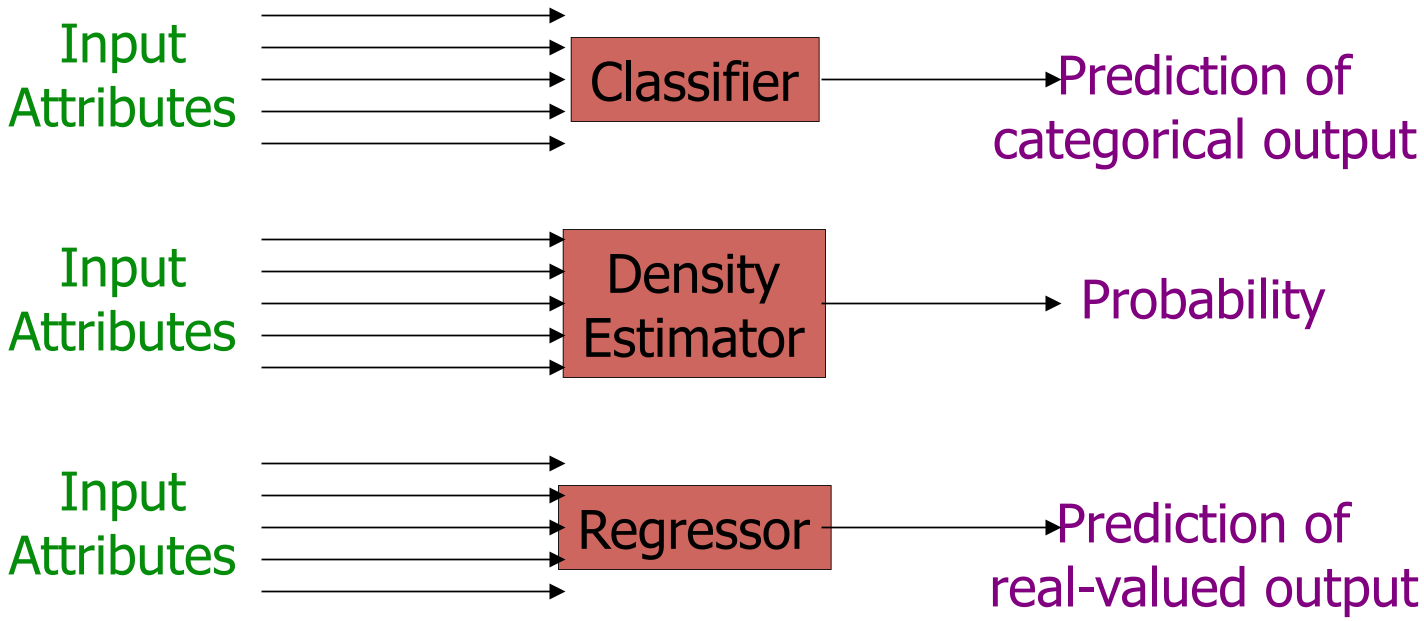
Density Estimation

- Our Joint Distribution learner is our first example of something called Density Estimation
- A Density Estimator learns a mapping from a set of attributes to a Probability

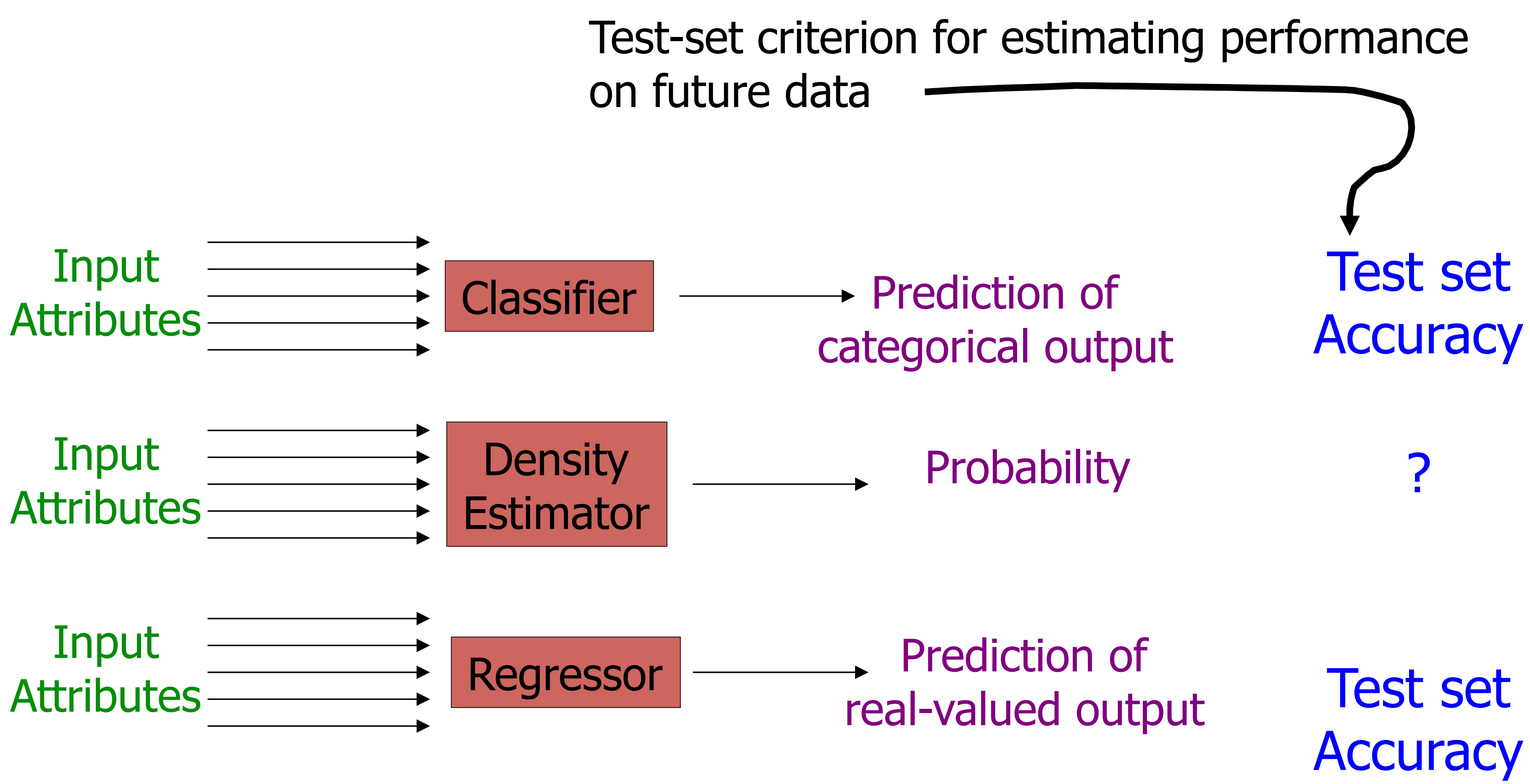


Density Estimation

- Compare it against the two other major kinds of models:



Evaluating Density Estimation



Evaluating a density estimator

- Given a record **x**, a density estimator *M* can tell you how likely the record is:

$$\hat{P}(\mathbf{x}|M)$$

- Given a dataset with *R* records, a density estimator can tell you how likely the dataset is:

(Under the assumption that all records were independently generated from the Density Estimator's JD)

$$\hat{P}(\text{dataset} \mid M) = \hat{P}(\mathbf{x}_1 \text{ and } \mathbf{x}_2 \ldots \text{ and } \mathbf{x}_R \mid M) = \prod_{k=1}^{R} \hat{P}(\mathbf{x}_k \mid M)$$

Revisit the Miles Per Gallon dataset

192
Training
Set
Records

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	europe
bad	70to74	america
bad	70to74	america
bad	70to74	asia
bad	70to74	asia
bad	75to78	america
:	:	:
:	:	:
:	:	:
bad	70to74	america
good	79to83	america
bad	75to78	america
good	79to83	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe

the Miles Per Gallon dataset

192 Training Set Records

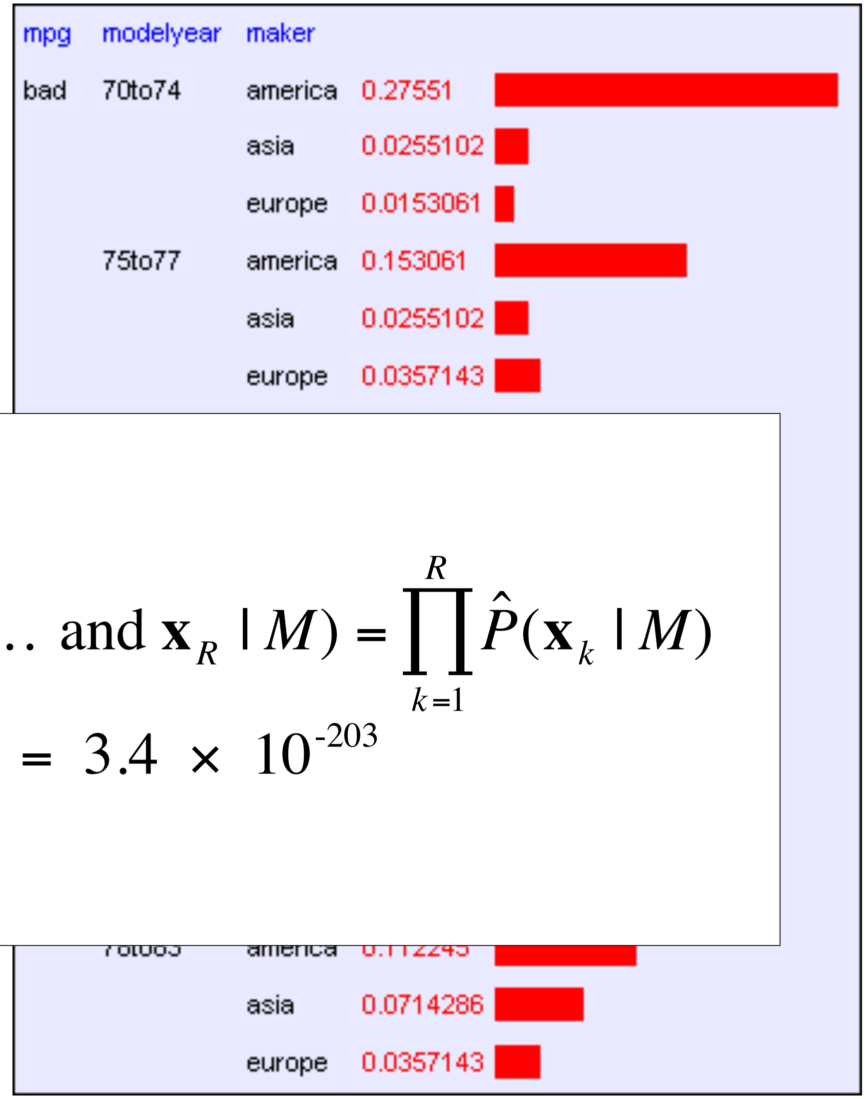
mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	europe
bad	70to74	america
bad	70to74	america
bad	70to74	asia
bad	70to74	asia
bad	75to78	america
:	:	:
:	:	:
:	:	:
bad	70to74	america
good	79to83	america
bad	75to78	america
good	79to83	america
bad	75to78	america
good	79to83	america
good	79to83	america
bad	70to74	america
good	75to78	europe
bad	75to78	europe

mpg	modelyear	maker	
bad	70to74	america	0.27551
		asia	0.0255102
		europe	0.0153061
	75to77	america	0.153061
		asia	0.0255102
		europe	0.0357143
	78to83	america	0.0561224
		asia	Never
		europe	Never
good	70to74	america	0.0102041
		asia	0.0306122
		europe	0.0459184
	75to77	america	0.0306122
		asia	0.0408163
		europe	0.0357143
	78to83	america	0.112245
		asia	0.0714286
		europe	0.0357143

the Miles Per Gallon dataset

192 Training Set Records

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	europe
bad	70to74	america
bad	70to74	america
bad	70to74	asia
bad	70to74	asia



$$\hat{P}(\text{dataset} \mid M) = \hat{P}(\mathbf{x}_1 \text{ and } \mathbf{x}_2 \ldots \text{ and } \mathbf{x}_R \mid M) = \prod_{k=1}^{R} \hat{P}(\mathbf{x}_k \mid M)$$

$$= \text{(in this case)} = 3.4 \times 10^{-203}$$

Log Probabilities

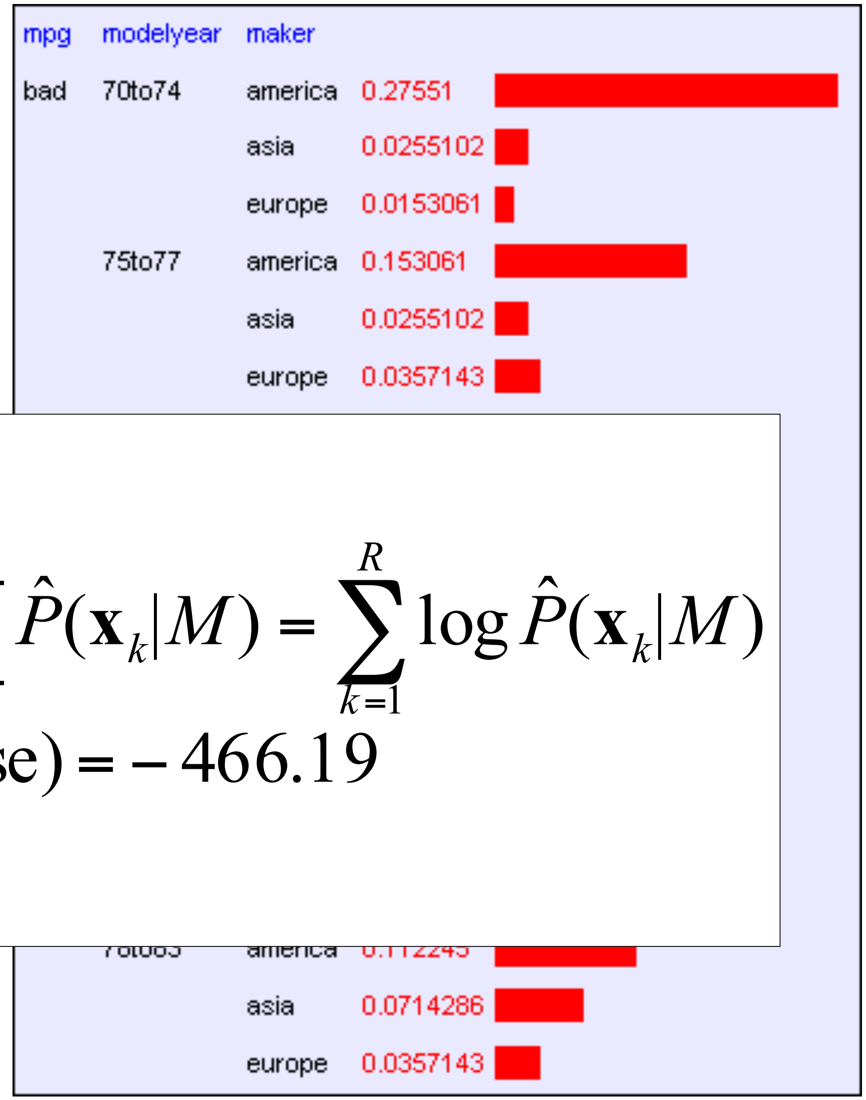
Since probabilities of datasets get so small we usually use log probabilities

$$\log \hat{P}(\text{dataset}|M) = \log \prod_{k=1}^{R} \hat{P}(\mathbf{x}_k|M) = \sum_{k=1}^{R} \log \hat{P}(\mathbf{x}_k|M)$$

the Miles Per Gallon dataset

192 Training Set

mpg	modelyear	maker
good	75to78	asia
bad	70to74	america
bad	75to78	europe
bad	70to74	america
bad	70to74	america
bad	70to74	asia
bad	70to74	asia



$$\log \hat{P}(\text{dataset}|M) = \log \prod_{k=1}^{R} \hat{P}(\mathbf{x}_k|M) = \sum_{k=1}^{R} \log \hat{P}(\mathbf{x}_k|M)$$

$$= (\text{in this case}) = -466.19$$

Summary: The Good News

- We have a way to learn a Density Estimator from data.
- Density estimators can do many good things…
  - Can sort the records by probability, and thus spot weird records (anomaly detection)
  - Can do inference: P(E1|E2)

    Automatic Doctor / Help Desk etc
  - Can perform classification, e.g., $p(C_k|A_1, A_2, \dots A_n)$
  - Ingredient for Bayes Classifiers (see later)

Summary: The Bad News

- Density estimation by directly learning the joint is trivial, mindless and dangerous

Using a test set

	Set Size	Log likelihood
Training Set	196	-466.1905
Test Set	196	-614.6157

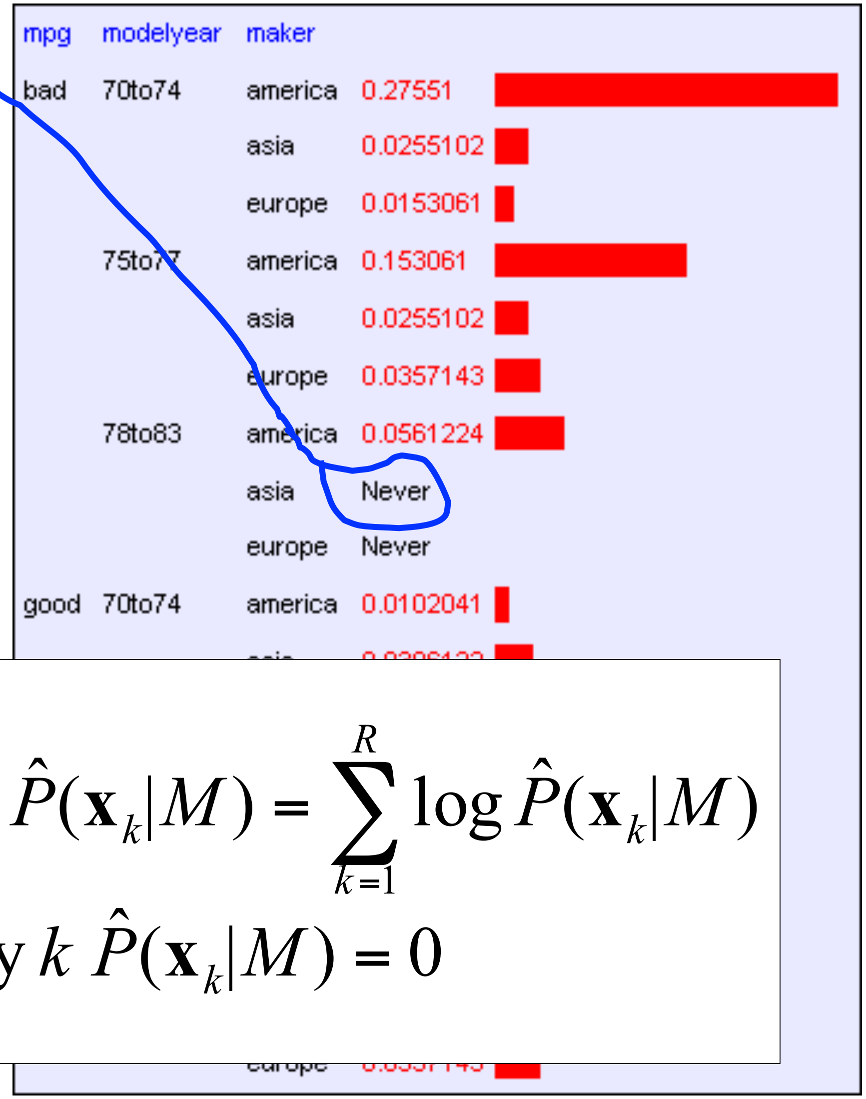
An independent test set with 196 cars has a worse log likelihood

(actually it's a billion quintillion quintillion quintillion quintillion times less likely)

….Density estimators can overfit. And the full joint density estimator is the overfittiest of them all!

Overfitting Density Estimators

If this ever happens, it means there are certain combinations that we learn are impossible



mpg	modelyear	maker	
bad	70to74	america	0.27551
		asia	0.0255102
		europe	0.0153061
	75to77	america	0.153061
		asia	0.0255102
		europe	0.0357143
	78to83	america	0.0561224
		asia	Never
		europe	Never
good	70to74	america	0.0102041

$$\log \hat{P}(\text{testset}|M) = \log \prod_{k=1}^{R} \hat{P}(\mathbf{x}_k|M) = \sum_{k=1}^{R} \log \hat{P}(\mathbf{x}_k|M)$$

$$= -\infty \text{ if for any } k \; \hat{P}(\mathbf{x}_k|M) = 0$$

Using a test set

	Set Size	Log likelihood
Training Set	196	-466.1905
Test Set	196	-614.6157

The only reason that our test set didn't score -infinity is that the code is hard-wired to always predict a probability of at least one in 10^{20}

We need Density Estimators that are less prone to overfitting

Naïve Density Estimation

The problem with the Joint Estimator is that it just mirrors the training data.

We need something which generalizes more usefully.

The naïve model generalizes strongly:

Assume that each attribute is distributed independently of any of the other attributes.

Independently Distributed Data

- Let *x[i]* denote the *i'* th field of record *x*.
- The independent distribution assumption says that for any *i,v,* $u_1\ u_2 \ldots\ u_{i-1}\ u_{i+1} \ldots\ u_M$

$$P(x[i] = v \mid x[1] = u_1, x[2] = u_2, \ldots x[i-1] = u_{i-1}, x[i+1] = u_{i+1}, \ldots x[M] = u_M) = P(x[i] = v)$$

- Or in other words, *x[i]* is independent of *{x[1],x[2],..x[i-1], x[i+1],…x[M]}*
- This is often written as

$$x[i] \perp \{x[1], x[2], \ldots x[i-1], x[i+1], \ldots x[M]\}$$

A note about independence

- Assume A and B are Boolean Random Variables. Then

  "A and B are independent" if and only if

  $$P(A|B) = P(A)$$

- "A and B are independent" is often notated as

  $$A \perp B$$

Independence Theorems

- Assume $P(A|B) = P(A)$

  Then

  $$P(A \text{ and } B) = P(A)\,P(B)$$

- Assume $P(A|B) = P(A)$

  Then

  $$P(\sim A|B) = P(\sim A)$$

- Assume $P(A|B) = P(A)$

  Then

  $$P(B|A) = P(B)$$

- Assume $P(A|B) = P(A)$

  Then

  $$P(A|\sim B) = P(A)$$

Multivalued Independence

For multivalued Random Variables A and B,

$$A \perp B$$

if and only if

$$\forall u, v : P(A = u \mid B = v) = P(A = u)$$

from which you can then prove things like…

$$\forall u,v : P(A = u \;\text{ and } B = v) = P(A = u)P(B = v)$$

$$\forall u, v : P(B = v \mid A = v) = P(B = v)$$

Multivalued Independence

- Example:
  – Suppose that each record is generated by randomly shaking a green dice and a red dice
    - Dataset 1: A = red dice value, B = green dice value
    - Dataset 2: A = red dice value, B = sum of two dice values
    - Dataset 3: A = sum of the two dice values, B = difference of the two dice values
  – Which of these datasets violates the naïve assumption?

Using the Naïve Distribution

- Once you have a Naïve Distribution you can easily compute any row of the joint distribution.
- Suppose A, B, C and D are independently distributed,

  What is P(A and ~B and C and ~D)?

Using the Naïve Distribution

- Once you have a Naïve Distribution you can easily compute any row of the joint distribution.
- Suppose A, B, C and D are independently distributed. What is P(A and ~B and C and ~D)?

$= P(A | \sim B \wedge C \wedge \sim D)\, P(\sim B \wedge C \wedge \sim D)$

$= P(A)\, P(\sim B \wedge C \wedge \sim D)$

$= P(A)\, P(\sim B | C \wedge \sim D)\, P(C \wedge \sim D)$

$= P(A)\, P(\sim B)\, P(C \wedge \sim D)$

$= P(A)\, P(\sim B)\, P(C | \sim D)\, P(\sim D)$

$= P(A)\, P(\sim B)\, P(C)\, P(\sim D)$

Using the Naïve Distribution

- Suppose A, B, C and D are independently distributed. What is P(A and ~B and C and ~D)?

Naïve Distribution General Case

- Suppose *x[1], x[2], … x[M]* are independently distributed.

$$P(x[1]=u_1, x[2]=u_2, \ldots x[M]=u_M) = \prod_{k=1}^{M} P(x[k]=u_k)$$

- So if we have a Naïve Distribution we can construct any row of the implied Joint Distribution on demand.
- So we can do any inference
- But how do we learn a Naïve Density Estimator?

Learning a Naïve Density Estimator

$$\hat{P}(x[i]=u) = \frac{\#\text{records in which } x[i]=u}{\text{total number of records}}$$

Another trivial learning algorithm!

Contrast

Joint DE	Naïve DE
Can model anything	Can model only very boring distributions
No problem to model "C is a noisy copy of A"	Outside Naïve's scope
Given 100 records and more than 6 Boolean attributes will screw up badly	Given 100 records and 10,000 multivalued attributes will be fine

Empirical Results: "Hopeless"

The "hopeless" dataset consists of 40,000 records and 21 Boolean attributes called a,b,c, … u. Each attribute in each record is generated 50-50 randomly as 0 or 1.

Name	Model	Parameters	LogLike		
Model1	joint	submodel=gauss gausstype=general	-272625	+/-	301.109
Model2	naive	submodel=gauss gausstype=general	-58225.6	+/-	0.554747

Average test set log probability during 10 folds cross-validation

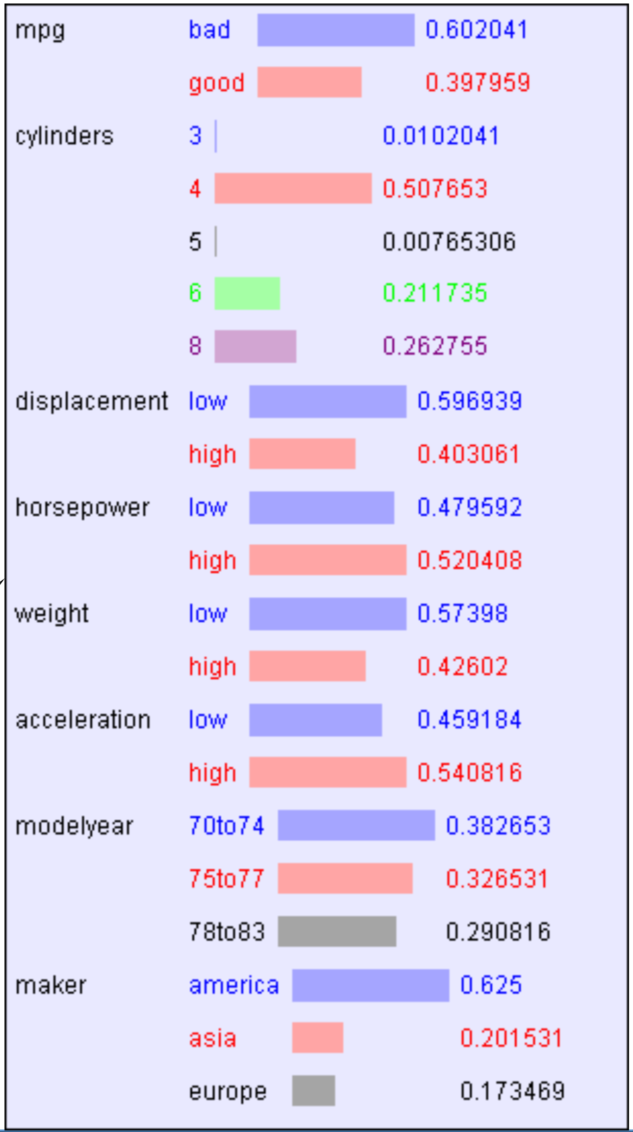
Despite the vast amount of data, "Joint" overfits hopelessly and does much worse

Empirical Results: “MPG”

The “MPG” dataset consists of 392 records and 8 attributes

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	
bad	3	low	low	low	low	70to74	america	Never
							asia	0.00255102
							europe	Never
						75to77	Never	
						78to83	Never	
					high	Never		
				high	Never			
			high	low	low	70to74	america	Never
							asia	0.00255102
							europe	Never
						75to77	america	Never
							asia	0.00255102
							europe	Never
						78to83	america	Never
							asia	0.00255102
							europe	Never
					high	Never		
				high	Never			
		high	Never					
	4	low	low	low	low	70to74	america	0.00255102
							asia	Never
							europe	0.00255102
						75to77	Never	
						78to83	Never	
					high	70to74	america	0.0204082
							asia	0.00255102

A tiny part of the DE learned by “Joint”

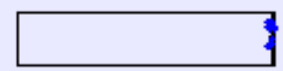


The DE learned by “Naive”

Empirical Results: “MPG”

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker	
bad	3	low	low	low	low	70to74	america	Never
							asia	0.00255102
							europe	Never
						75to77	Never	
						78to83	Never	
					high	Never		
		high	Never					
	4	low	low	low	low	70to74	america	0.00255102
							asia	Never
							europe	0.00255102
						75to77	Never	
						78to83	Never	
					high	70to74	america	0.0204082
							asia	0.00255102

A tiny part of

Name	Model	Parameters	LogLike			
Model1	joint	submodel=gauss gausstype=general	-472.486	+/-	77.2184	
Model2	naive	submodel=gauss gausstype=general	-257.212	+/-	3.02246	

mpg	bad	0.602041
	good	0.397959
cylinders	3	0.0102041
	4	0.507653
	5	0.00765306
acceleration	low	0.459184
	high	0.540816
modelyear	70to74	0.382653
	75to77	0.326531
	78to83	0.290816
maker	america	0.625
	asia	0.201531
	europe	0.173469

The DE learned by “Naive”

Empirical Results: "Weight vs. MPG"

mpg	weight	
bad	low	0.193878
	high	0.408163
good	low	0.380102
	high	0.0178571

The DE learned by "Joint"

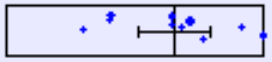
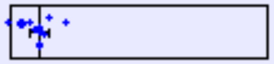
mpg	bad	0.602041
	good	0.397959
weight	low	0.57398
	high	0.42602

The DE learned by "Naive"

Empirical Results: "Weight vs. MPG"

mpg	weight		
bad	low	0.193878	
	high	0.408163	
good			

mpg		
bad		0.602041
good		0.397959

Name	Model	Parameters	LogLike			
Model1	joint	submodel=gauss gausstype=general	-44.3562	+/-	2.27547	
Model2	naive	submodel=gauss gausstype=general	-53.2231	+/-	0.610411	

learned by "Joint"

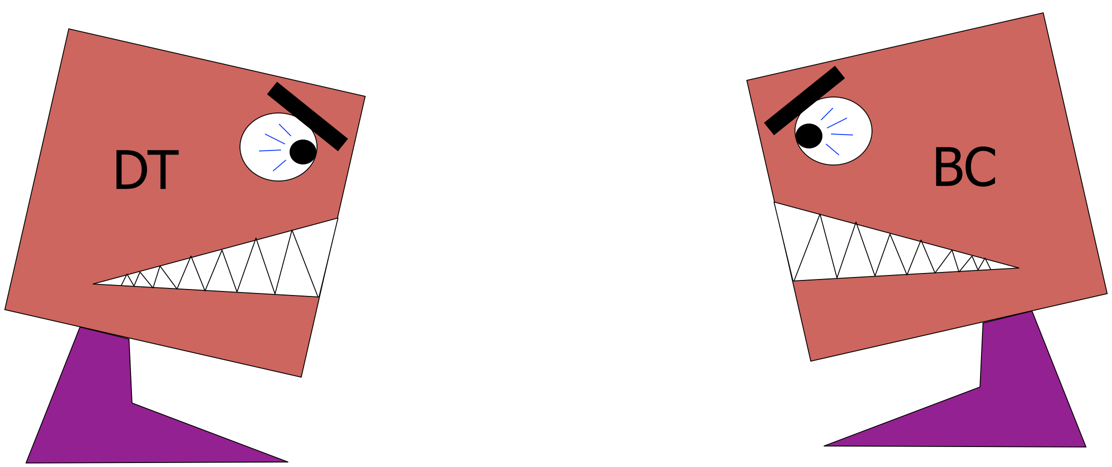
The DE learned by "Naive"

Reminder: The Good News

- We have two ways to learn a Density Estimator from data.
- Other, vastly more impressive Density Estimators developed
  - Mixture Models, Bayesian Networks, Density Trees, Kernel Densities and many more
- Density estimators can do many good things…
  - Anomaly detection
  - Can do inference: P(E1|E2) Automatic Doctor / Help Desk etc
  - Ingredient for Bayes Classifiers

Bayes Classifiers

- A formidable and sworn enemy of decision trees



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

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(X_1 = u_1, X_2 = u_2, \cdots X_m = u_m \mid Y = v)$$

Is this a good idea?

How to build a ~~Bayes~~ Classifier

- Assume you want to predict output … v_{ny}.
- Assume there are m input attribute
- Break dataset into n_Y smaller datase
- Define DS_i = Records in which $Y=v_i$
- For each DS_i , learn Density Estimat … among the $Y=v_i$ records.
- M_i estimates $P(X_1, X_2, \ldots X_m \mid Y=v_i)$

This is a Maximum Likelihood classifier.

It can get silly if some Ys are very unlikely

- 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

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}}\, P(X_1 = u_1, X_2 = u_2, \cdots X_m = u_m \mid Y = v)$$

Is this a good idea?

How to build a Bayes Classifier

- Assume you want to predict output Y which h
- Assume there are m input attributes called X_1
- Break dataset into n_Y smaller datasets called L
- Define DS_i = Records in which $Y=v_i$
- For each DS_i , learn Density Estimator M_i to m ne $Y=v_i$ 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

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(Y = v \mid X_1 = u_1, X_2 = u_2, \cdots X_m = u_m)$$

Terminology

- MLE (Maximum Likelihood Estimator):

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

- MAP (Maximum A-Posteriori Estimator):

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

Computing a posterior probability

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

$$P(Y = v \mid X_1 = u_1 \cdots X_m = u_m)$$

$$= \frac{P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)P(Y = v)}{P(X_1 = u_1 \cdots X_m = u_m)}$$

$$= \frac{P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)P(Y = v)}{\sum_{j=1}^{n_Y} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v_j)P(Y = v_j)}$$

Bayes Classifiers in a nutshell

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:

$$
\begin{aligned}
Y^{\text{predict}} &= \operatorname*{argmax}_{v} P(Y = v \mid X_1 = u_1 \cdots X_m = u_m) \\
&= \operatorname*{argmax}_{v} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v) P(Y = v)
\end{aligned}
$$

Bayes Classifiers in a nutshell

- Step 1. Learn the distribution over inputs for each value Y.
- Step 2. This gives $P(X_1, X_2, \ldots$
- Step 3. Estimate $P(Y=v_i)$ as with $Y=v_i$.
- Step 4. For a new prediction

We can use our favorite Density Estimator here.

Right now we have two options:

•Joint Density Estimator
•Naïve Density Estimator

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

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

Joint Density 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 joint Bayes Classifier this degenerates to a very simple rule:

- $Y^{predict}$ = the most common value of Y among records in which $X_1 = u_1, X_2 = u_2, \ldots. X_m = u_m$.

- Note that if no records have the exact set of inputs $X_1 = u_1, X_2 = u_2, \ldots. X_m = u_m$, then P($X_1$, X_2, … X_m / $Y=v_i$) = 0 for all values of Y.

- In that case we just have to guess Y' s value

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:

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

An Example

Day	Outlook	Temperature	Humidity	Wind	PlayGolf
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D3	overcast	hot	high	weak	yes
D4	rain	mild	high	weak	yes
D5	rain	cool	normal	weak	yes
D6	rain	cool	normal	strong	no
D7	overcast	cool	normal	strong	yes
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D10	rain	mild	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes
D14	rain	mild	high	strong	no

To Learn a Naïve Bayes Classifier from this data

Two classes: $y=v_1$: play golf=no

$y=v_2$: play golf=yes

four attributes:

x_1: three values (sunny, overcast, rain)

x_2: three values (hot, mild, cool)

x_3: two values (high, normal)

x_4: two values (weak, strong)

Which probabilities do we need to compute?

- P(class1 = yes)

P(class2=no)

P(a1=sunny|y=yes)
P(a1=overcast|y=yes)
P(a1=rain|y=yes)

P(a2=hot|y=yes)
P(a2=mild|y=yes)
P(a2=cool|y=yes)

P(a3=high|y=yes)
P(a3=normal|y=yes)

P(a4=weak|y=yes)
P(a4=strong|y=yes)

P(a1=sunny|y=no)
P(a1=overcast|y=no)
P(a1=rain|y=no)

P(a2=hot|y=no)
P(a2=mild|y=no)
P(a2=cool|y=no)

P(a3=high|y=no)
P(a3=normal|y=no)

P(a4=weak|y=no)
P(a4=strong|y=no)

Reorder according to class label

Day	Outlook	Temperature	Humidity	Wind	Play Golf
D1	sunny	hot	high	weak	no
D2	sunny	hot	high	strong	no
D3	overcast	hot	high	weak	yes
D4	rain	mild	high	weak	yes
D5	rain	cool	normal	weak	yes
D6	rain	cool	normal	strong	no
D7	overcast	cool	normal	strong	yes
D8	sunny	mild	high	weak	no
D9	sunny	cool	normal	weak	yes
D10	rain	mild	normal	weak	yes
D11	sunny	mild	normal	strong	yes
D12	overcast	mild	high	strong	yes
D13	overcast	hot	normal	weak	yes
D14	rain	mild	high	strong	no

Classification Step

Given a new case/object:

outlook=sunny,

temperature=cool,

humid=high,

wind = strong

Question: whether to play or not to play golf?

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:

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

Technical Hint:
If you have 10,000 input attributes that product will underflow in floating point math. You should use logs:

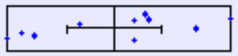
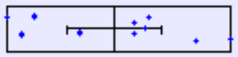
$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} \left(\log P(Y = v) + \sum_{j=1}^{n_Y} \log P(X_j = u_j \mid Y = v) \right)$$

Naive BC Results: "Logical"

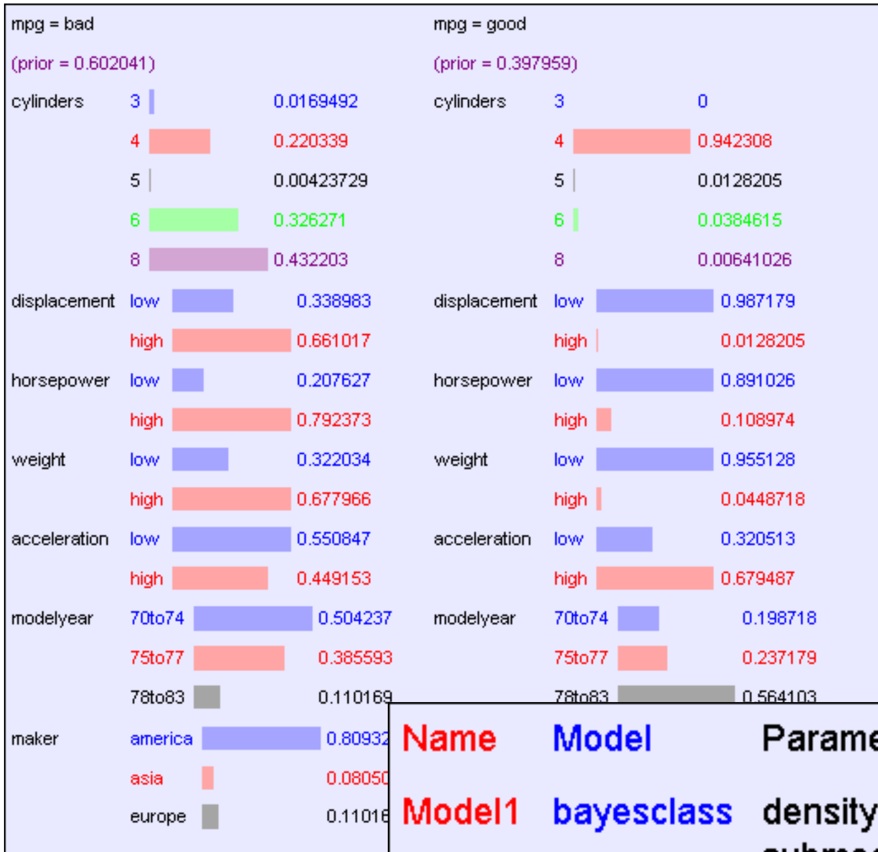
The "logical" dataset consists of 40,000 records and 4 Boolean attributes called a,b,c,d where a,b,c are generated 50-50 randomly as 0 or 1. d = a and ~c, except that in 10% of records it is flipped

		d = 0 (prior = 0.69945)			d = 1 (prior = 0.30055)
a	0	0.644935	a	0	0.163783
	1	0.355065		1	0.836217
b	0	0.501287	b	0	0.498503
	1	0.498713		1	0.501497
c	0	0.358818	c	0	0.834054
	1	0.641182		1	0.165946

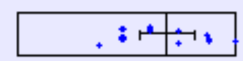
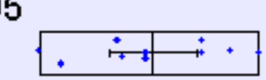
The Classifier learned by "Naive BC"

Name	Model	Parameters	FracRight			
Model1	bayesclass	density=joint submodel=gauss gausstype=general	0.90065	+/-	0.00301897	
Model2	bayesclass	density=naive submodel=gauss gausstype=general	0.90065	+/-	0.00301897	

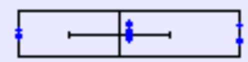
BC Results: “MPG”: 392 records



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	

BC Results: “MPG”: 40 records

Name	Model	Parameters	FracRight			
Model1	bayesclass	density=joint submodel=gauss gausstype=general	0.725	+/-	0.114333	
Model2	bayesclass	density=naive submodel=gauss gausstype=general	0.8	+/-	0.122227	

Classify text with naïve Bayes classifier

- Why?
  - Learn which news articles are of interest
  - Learn to classify web pages by topic
  - Spam control…
- Naïve Bayes is among the most effective algorithms

What attributes shall we use to represent text documents?

Text Classification – data formulation

- Class label:

  Target concept Interesting?

  Document → {class1=yes, class2=no}

- represent each document by vector of words (one attribute per word position in document)
  - Remove stopwords, numbers, tags, single letters, …
  - Change all words to lower case
  - Stemming (only retain roots)
  - Remove words appeared only once

Naïve Bayes Classifier for Text Classification

- Build classifier: estimate

  P(class1=yes), P(class2=no),

  P(doc|class1=yes), P(doc|class2=no)

conditional independence assumption:

$$P(doc \mid class_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k \mid class_j)$$

Probability word in position i is w_k for $class_j$

Naïve Bayes Classifier for Text Classification

- Additional assumption: positional independence assumption

drop word positioning

$P(a_i{=}w_k|class_j) = P(a_m{=}w_k|class_j)$, for all i, m

Therefore,

$$P(doc \mid class_j) = \prod_{i=1}^{length(doc)} P(a_i = w_k \mid class_j)$$

$$= \prod_{i}^{length(doc)} P(w_i \mid class_j)$$

Steps in Learning Naïve Bayes Text Classifier

- Collect all words and other tokens that occur in examples
- Vocabulary = all distinct words and other tokens in the examples
- Calculate $P(class_j)$ and $P(w_k|class_j)$ for each target value $class_j$:
  - doc_j = subset of document examples for which the target value is $class_j$
  - $P(class_j) = |doc_j| / |\text{all document examples}|$
  - $text_j \leftarrow$ a single document created by concatenating all members of doc_j

Steps in Learning Naïve Bayes Text Classifier

– n = total number of words in $Text_j$ (counting duplicate words multiple times)

– for each word w_k in Vocabulary

n_k = number of times word w_k occurs in $Text_j$

$$P(w_k \mid class_j) = \frac{(n_k + 1)}{n + |vocabulary|}$$

Steps in Classifying a Document using the Naïve Test Classifier

- Positions = all word positions in the document that contain tokens found in Vocabulary
- Return v_{NB}, where

$$v_{NB} = \operatorname*{argmax}_{j} \; P(class_j) \prod_{i \in positions} P(w_i \mid class_j)$$

Example Application: Classify newsgroup documents

Given 1000 training documents from each group, learn to classify new documents according to which newsgroup it came from:

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.hockey
….	

Result: Naïve Bayes obtained 89% classification accuracy