PROBABILITY AND SCALABILITY: LEARNING AND COUNTING

Why More Data Helps

- Data:
  - All 5-grams that appear >= 40 times in a corpus of 1M English books
    - approx 80B words
    - 5-grams: 30Gb compressed, 250-300Gb uncompressed
    - Each 5-gram contains frequency distribution over *years*
  - Wrote code to compute
    - Pr(A,B,C,D,E | C=affect or C=effect)
    - Pr(any subset of A,…,E | any other fixed values of A,…,E with C=affect V effect)
- Observations [from playing with data]:
  - Mostly effect not affect
  - Most common word before affect is not
  - After not effect most common word is a
  - …

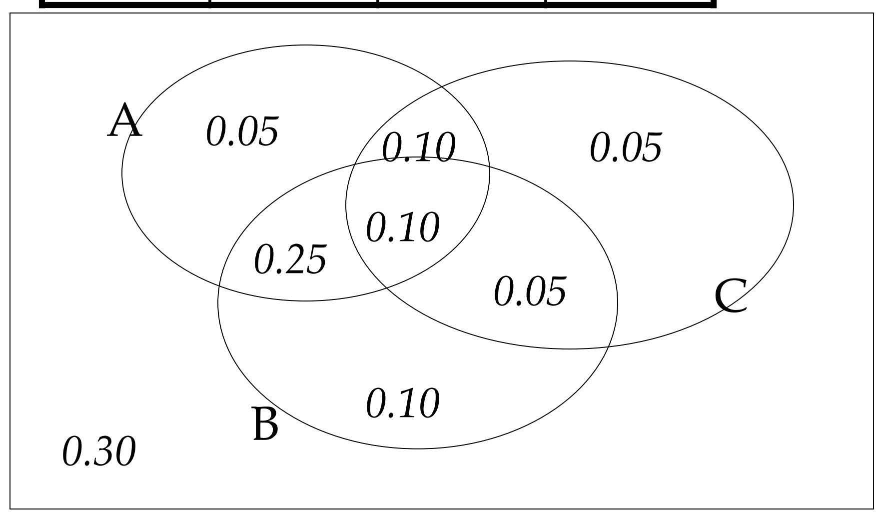
The Joint Distribution

Example: Boolean variables A, B, C

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



Some of the Joint Distribution

A	B	C	D	E	p
is	the	effect	of	the	0.00036
is	the	effect	of	a	0.00034
.	The	effect	of	this	0.00034
to	this	effect	:	“	0.00034
be	the	effect	of	the	…
	…	…	…	…	…
not	the	effect	of	any	0.00024
	…	…	…	…	…
does	not	affect	the	general	0.00020
does	not	affect	the	question	0.00020
any	manner	affect	the	principle	0.00018

An experiment: how useful is the brute-force joint classifier?

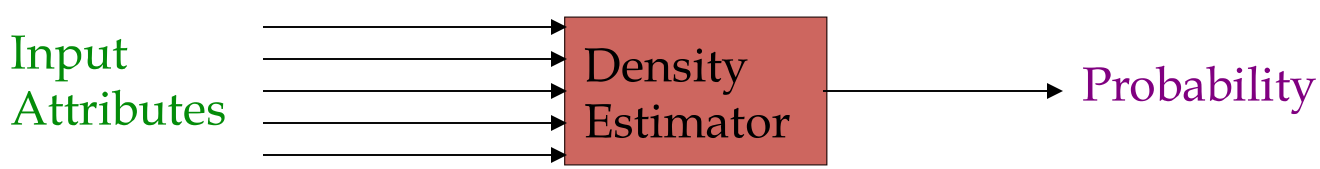
- Extracted all affect/effect 5-grams from an old Reuters corpus
  - about 20k documents
  - about 723 n-grams, 661 distinct
  - Financial news, not novels or textbooks
- Tried to predict center word with:
  - $Pr(C \mid A=a,B=b,D=d,E=e)$
  - then $P(C \mid A,B,D)$
  - then $P(C \mid B,D)$
  - then $P(C \mid B)$
  - then $P(C)$

EXAMPLES

- “The cumulative _ of the” → effect (1.0)
- “Go into _ on January” → effect (1.0)
- “From cumulative _ of accounting” not present in train data
  - Nor is ““From cumulative _ of _”
  - But “_ cumulative _ of _” → effect (1.0)
- “Would not _ Finance Minister” not present
  - But “_ not _ _ _” → affect (0.9625)

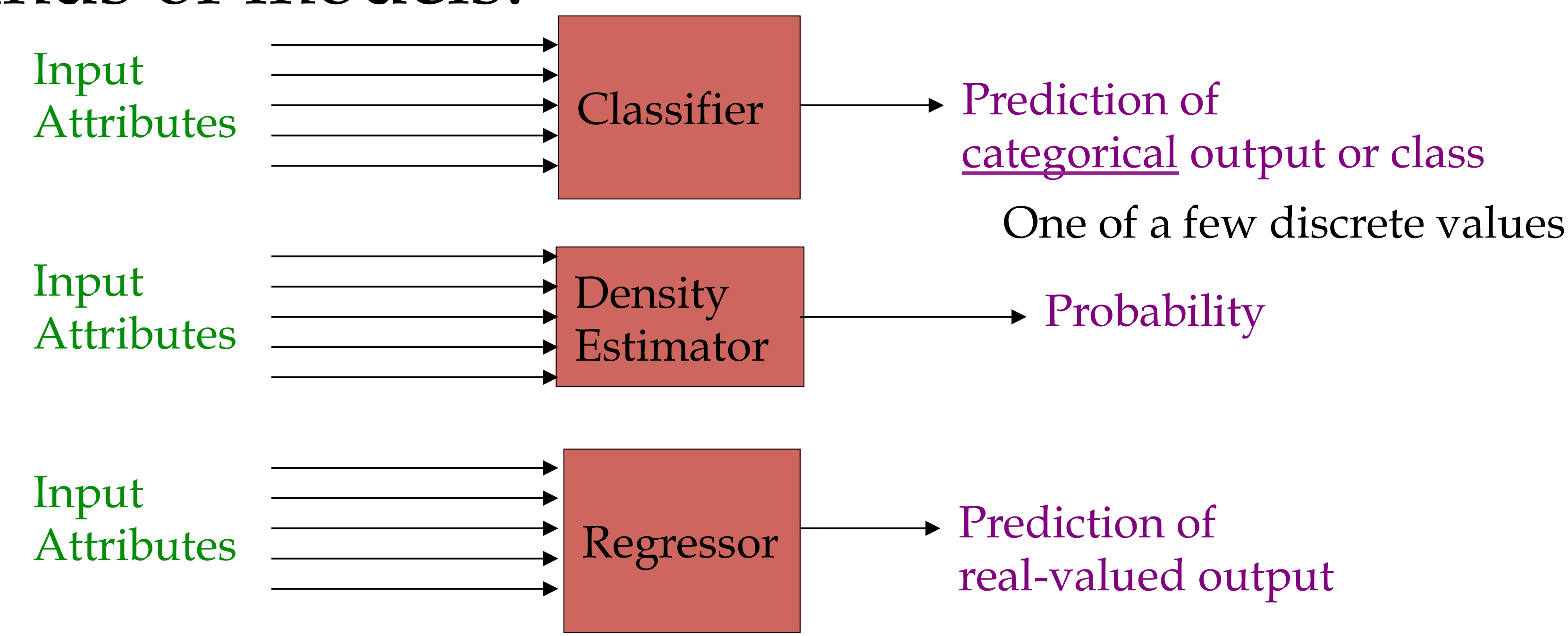
Density Estimation

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

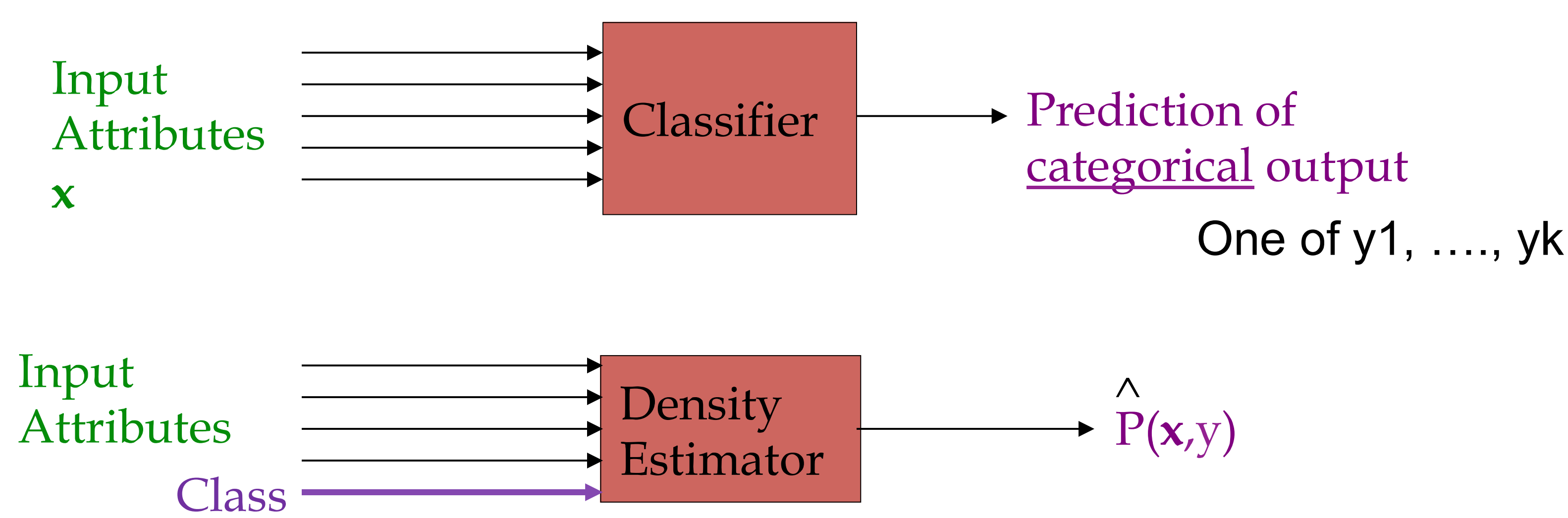


Density Estimation

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



Density Estimation → Classification



To classify **x**

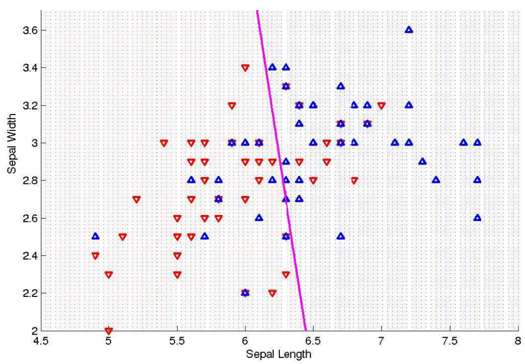
1. Use your estimator to compute $\hat{P}(\mathbf{x},y1)$, …., $\hat{P}(\mathbf{x},yk)$
2. Return the class y* with the highest predicted probability

Ideally is correct with $\hat{P}(x,y^*) = \hat{P}(x,y^*)/(\hat{P}(\mathbf{x},y1) + \ldots + \hat{P}(\mathbf{x},yk))$

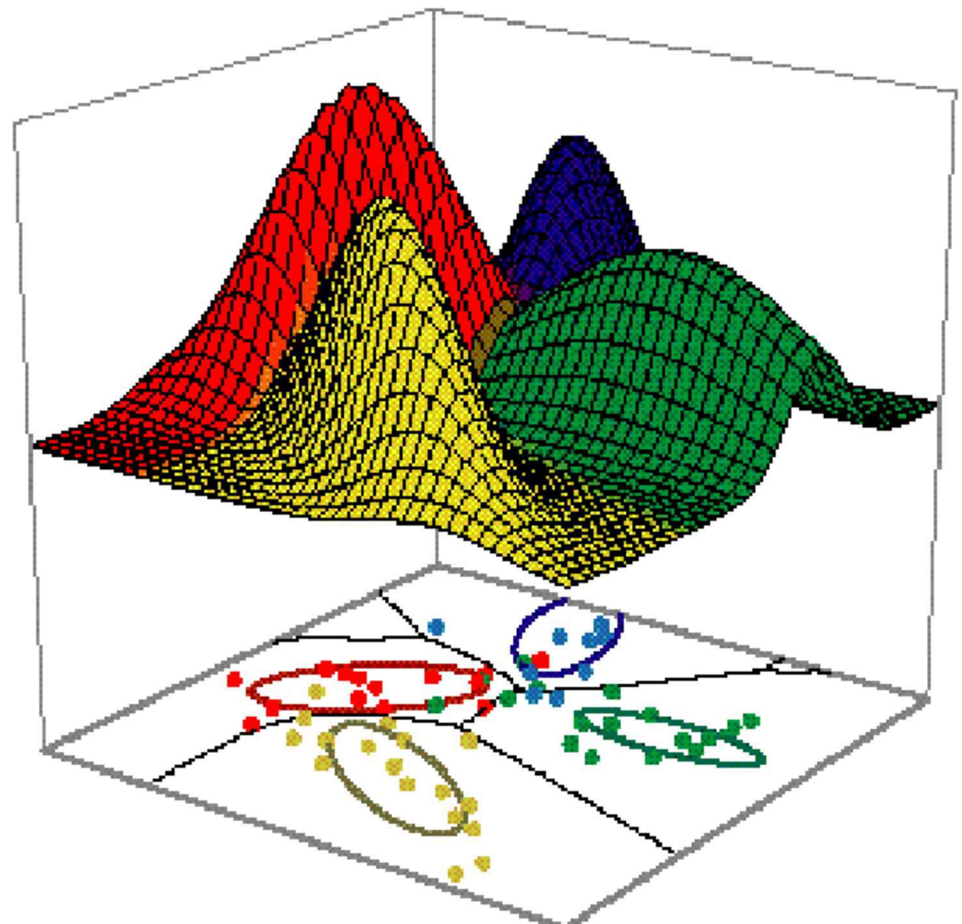
Binary case:
predict POS
if $\hat{P}(\mathbf{x})>0.5$

Classification vs Density Estimation

Classification



Density Estimation



PROBABILITY AND SCALABILITY: NAÏVE BAYES

Second most scalable learning method in the world?

Naïve Density Estimation

What's an alternative to the joint distribution?

The naïve model generalizes strongly:

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

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 $P(A \wedge \sim B \wedge C \wedge \sim D)$?

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 $P(A \wedge \sim B \wedge C \wedge \sim D)$?

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

Naïve Distribution General Case

- Suppose $X_1, X_2, \ldots, X_d$ are independently distributed.

$$\Pr(X_1 = x_1, \ldots, X_d = x_d) = \Pr(X_1 = x_1) \cdot \ldots \cdot \Pr(X_d = x_d)$$

- So if we have a Naïve Distribution we can construct any row of the implied Joint Distribution on demand.

- How do we learn this?

Learning a Naïve Density Estimator

$$P(X_i = x_i) = \frac{\#\text{records with } X_i = x_i}{\#\text{records}}$$ MLE

$$P(X_i = x_i) = \frac{\#\text{records with } X_i = x_i + mq}{\#\text{records} + m}$$ Dirichlet (MAP)

Another trivial learning algorithm!

Can we make this interesting? Yes!

- Key ideas:
  - Pick the class variable Y
  - Instead of estimating $P(X_1,\ldots,X_n,Y) = P(X_1)*\ldots*P(X_n)*Y$, estimate $P(X_1,\ldots,X_n | Y) = P(X_1 | Y)*\ldots*P(X_n | Y)$
  - Or, assume $P(X_i | Y)=Pr(X_i | X_1,\ldots,X_{i-1},X_{i+1},\ldots X_n,Y)$
  - Or, that X_i is conditionally independent of every X_j, $j!=i$, given Y.

  - How to estimate?

MLE or MAP

The Naïve Bayes classifier – v1

- Dataset: each example has
  - A unique id *id*
    - Why? For debugging the feature extractor
  - d attributes $X_1, \ldots, X_d$
    - Each X_i takes a discrete value in $dom(X_i)$
  - One class label Y in $dom(Y)$
- You have a *train* dataset and a *test* dataset
- Assume:
  - the dataset doesn't fit in memory
  - the model does

stream through it

The Naïve Bayes classifier – v0

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, y, $x_1,\ldots,x_d$ in *train:*
  - C("Y=ANY") ++;   C("Y=y") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X_j=x_j$") ++
- For each example *id*, y, $x_1,\ldots,x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\Pr(y',x_1,\ldots,x_d) = \left(\prod_{j=1}^{d}\Pr(X_j=x_j \mid Y=y')\right)\Pr(Y=y')$

$$= \left(\prod_{j=1}^{d}\frac{\Pr(X_j=x_j, Y=y')}{\Pr(Y=y')}\right)\Pr(Y=y')$$

  - Return the best y'

The Naïve Bayes classifier – v0

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, $y, x_1, \ldots, x_d$ in *train:*
  - C("Y=ANY") ++;   C("$Y=y$") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X_j=x_j$") ++
- For each example *id*, $y, x_1, \ldots, x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\Pr(y', x_1, \ldots, x_d) = \left(\prod_{j=1}^{d} \Pr(X_j = x_j \mid Y = y')\right) \Pr(Y = y')$

$$= \left(\prod_{j=1}^{d} \frac{C(X_j = x_j \wedge Y = y')}{C(Y = y')}\right) \frac{C(Y = y')}{C(Y = ANY)}$$

  - Return the best y'

The Naïve Bayes classifier – v0

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, y, $x_1,\ldots,x_d$ in *train:*
  - C("Y=ANY") ++;   C("Y=y") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X_j=x_j$") ++
- For each example *id*, y, $x_1,\ldots,x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\Pr(y',x_1,\ldots,x_d) = \left(\prod_{j=1}^{d} \Pr(X_j = x_j \mid Y = y')\right)\Pr(Y = y')$

$$= \left(\prod_{j=1}^{d} \frac{C(X_j = x_j \wedge Y = y')}{C(Y = y')}\right)\frac{C(Y = y')}{C(Y = ANY)}$$

This may overfit, so …

  - Return the best y'

The Naïve Bayes classifier – v1

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, $y, x_1, \ldots, x_d$ in *train:*
  - C("Y=ANY") ++; C("Y=y") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X_j=x_j$") ++
- For each example *id*, $y, x_1, \ldots, x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\Pr(y', x_1, \ldots, x_d) = \left(\prod_{j=1}^{d} \Pr(X_j = x_j \mid Y = y')\right) \Pr(Y = y')$

$$= \left(\prod_{j=1}^{d} \frac{C(X_j = x_j \wedge Y = y') + mq_x}{C(Y = y') + m}\right) \frac{C(Y = y') + mq_y}{C(Y = ANY) + m}$$

where:
$q_j = 1/|dom(X_j)|$
$q_y = 1/|dom(Y)|$
$mq_x = 1$

  - Return the best y'

This may underflow, so …

The Naïve Bayes classifier – v1

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, $y, x_1, \ldots, x_d$ in *train:*
  - C("Y=ANY") ++;   C("$Y=y$") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X_j=x_j$") ++
- For each example *id*, $y, x_1, \ldots, x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\log \Pr(y', x_1, \ldots, x_d)$ =

$$= \left(\sum_j \log \frac{C(X_j = x_j \wedge Y = y') + mq_j}{C(Y = y') + m} \right) + \log \frac{C(Y = y') + mq_j}{C(Y = ANY) + m}$$

where:
$q_j = 1/|dom(X_j)|$
$q_y = 1/|dom(Y)|$
$mq_x = 1$

  - Return the best y'

The Naïve Bayes classifier – v2

- For text documents, what features do you use?
- One common choice:
  - X_1 = first word in the document
  - X_2 = second word in the document
  - X_3 = third …
  - X_4 = …
  - …
- But: $\Pr(X_{13}=\textit{hockey} \mid Y=\textit{sports})$ is probably not that different from $\Pr(X_{11}=\textit{hockey} \mid Y=\textit{sports})$…so instead of treating them as different variables, treat them as different copies of the same variable

The Naïve Bayes classifier – v1

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, $y, x_1, \ldots, x_d$ in *train:*
  - C("Y=ANY") ++;   C("$Y=y$") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X_j=x_j$") ++
- For each example *id*, $y, x_1, \ldots, x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\Pr(y', x_1, \ldots, x_d) = \left(\prod_{j=1}^{d} \Pr(X_j = x_j \mid Y = y')\right) \Pr(Y = y')$

$$= \left(\prod_{j=1}^{d} \frac{\Pr(X_j = x_j, Y = y')}{\Pr(Y = y')}\right) \Pr(Y = y')$$

  - Return the best y'

The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id, y,* $x_1,\ldots,x_d$ in *train:*
  - C("Y=ANY") ++; C("$Y=y$") ++
  - For j in $1..d$:
    - C("$Y=y \wedge \cancel{X_j}=x_j$") ++
- For each example *id, y,* $x_1,\ldots,x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\Pr(y', x_1,\ldots,x_d) = \left(\prod_{j=1}^{d} \Pr(\cancel{X_j} = x_j \mid Y = y')\right)\Pr(Y = y')$

$$= \left(\prod_{j=1}^{d} \frac{\Pr(\cancel{X_j} = x_j, Y = y')}{\Pr(Y = y')}\right)\Pr(Y = y')$$

  - Return the best y'

The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, $y, x_1, \ldots, x_d$ in *train:*
  - C("Y=ANY") ++;   C("Y=y") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X=x_j$") ++
- For each example *id*, $y, x_1, \ldots, x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\Pr(y', x_1, \ldots, x_d) = \left(\prod_{j=1}^{d} \Pr(X = x_j \mid Y = y') \right) \Pr(Y = y')$

$$= \left(\prod_{j=1}^{d} \frac{\Pr(X = x_j, Y = y')}{\Pr(Y = y')} \right) \Pr(Y = y')$$

  - Return the best y'

The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id, y,* $x_1,\ldots,x_d$ in *train:*
  - C("*Y*=ANY") ++; C("*Y*=*y*") ++
  - For *j* in *1..d*:
    - C("*Y*=*y* ^ X=x_j") ++
- For each example *id, y,* $x_1,\ldots,x_d$ in *test:*
  - For each *y′* in *dom(Y):*
    - Compute log Pr($y', x_1,\ldots,x_d$) =

$$= \left(\sum_j \log \frac{C(X = x_j \wedge Y = y') + mq_x}{C(X = ANY \wedge Y = y') + m}\right) + \log \frac{C(Y = y') + mq_y}{C(Y = ANY) + m}$$

where:
$q_j = 1/|V|$
$q_y = 1/|dom(Y)|$
$mq_x=1$

  - Return the best *y′*

The Naïve Bayes classifier – v2

- You have a *train* dataset and a *test* dataset
- To classify documents, these might be:
  - http://wcohen.com academic,FacultyHome William W. Cohen Research Professor Machine Learning Department Carnegie Mellon University Member of the Language Technology Institute the joint CMU-Pitt Program in Computational Biology the Lane Center for Computational Biology and the Center for Bioimage Informatics Director of the Undergraduate Minor in Machine Learning Bio Teaching Projects Publications recent all Software Datasets Talks Students Colleagues Blog Contact Info Other Stuff …
  - http://google.com commercial Search Images Videos ….
  - …
- How about for n-grams?

Assume hashtable holding all counts fits in memory

Complexity of Naïve Bayes

- You have a *train* dataset and a *test* dataset
- Initialize an "event counter" (hashtable) C
- For each example *id*, $y, x_1, \ldots, x_d$ in *train:*
  - C("Y=ANY") ++; C("$Y=y$") ++
  - For j in $1..d$:
    - C("$Y=y \wedge X=x_j$") ++
- For each example *id*, $y, x_1, \ldots, x_d$ in *test:*
  - For each y' in $dom(Y)$:
    - Compute $\log \Pr(y', x_1, \ldots, x_d)$ =

$$= \left(\sum_j \log \frac{C(X = x_j \wedge Y = y') + mq_x}{C(X = ANY \wedge Y = y') + m} \right) + \log \frac{C(Y = y') + mq_y}{C(Y = ANY) + m}$$

  - Return the best y'

Sequential reads

Complexity: O(n), n=size of *train*

where:
$q_j = 1/|V|$
$q_y = 1/|dom(Y)|$
$mq_x = 1$

Sequential reads

Complexity: O($|dom(Y)| * n'$), n'=size of *test*

Naïve Bayes v2

- This is one example of a *streaming classifier*
  - Each example is only read only once
  - You can create a classifier and perform classifications at any point
  - Memory is minimal (<< O(n))
    - Ideally it would be constant
    - Traditionally less than O(sqrt(N))
  - Order doesn't matter
    - Nice because we may not control the order of examples in real life
    - This is a hard one to get a learning system to have!
- There are few competitive learning methods that as stream-y as naïve Bayes…