

UVA CS 4501 : Machine Learning

Lecture 19: Naïve Bayes Classifier for Text Classification

Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Where are we ? →

Three major sections for classification

- We can divide the large variety of classification approaches into **roughly three major types**

1. Discriminative

- directly estimate a decision rule/boundary
- e.g., support vector machine, decision tree

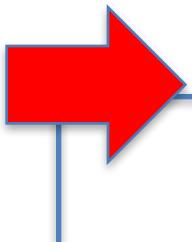
2. Generative:

- build a generative statistical model
- e.g., **naïve bayes classifier**, Bayesian networks

3. Instance based classifiers

- Use observation directly (no models)
- e.g. K nearest neighbors

Today : Naïve Bayes Classifier for Text

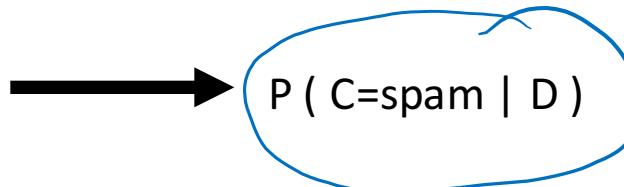
- 
- ✓ Dictionary based Vector space representation of text article
 - ✓ Multivariate Bernoulli vs. Multinomial
 - ✓ Multivariate Bernoulli
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - ✓ Multinomial naïve Bayes classifier
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models (**Extra**)

Text document classification, e.g. spam email filtering

- Input: document D
- Output: the predicted class C , c is from $\{c_1, \dots, c_L\}$
- Spam filtering Task: Classify **email** as ‘*Spam*’, ‘*Other*’.

TO BE REMOVED FROM FUTURE
MAILINGS, SIMPLY REPLY TO THIS
MESSAGE AND PUT "REMOVE" IN THE
SUBJECT.

99 MILLION EMAIL ADDRESSES
FOR ONLY \$99

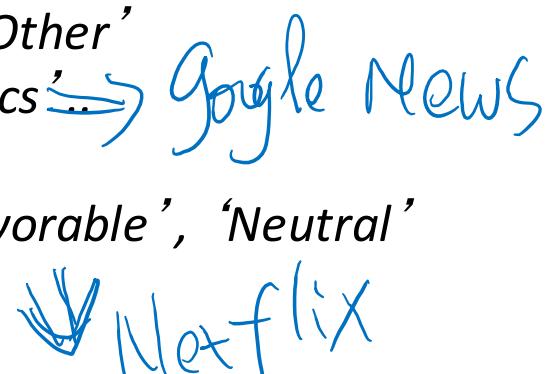


Text classification Tasks

- Input: document D
- Output: the predicted class C , c is from $\{c_1, \dots, c_L\}$

Text classification examples:

- Classify email as ‘Spam’, ‘Other’.
- Classify web pages as ‘Student’, ‘Faculty’, ‘Other’
- Classify news stories into topics ‘Sports’, ‘Politics’  Google News
- Classify business names by industry.
- Classify movie reviews as ‘Favorable’, ‘Unfavorable’, ‘Neutral’
- ... and many more.



Text Classification: Examples

- Classify shipment articles into one 93 categories.
- An example category ‘wheat’

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).

Maize Mar 48.0, total 48.0 (nil).

Sorghum nil (nil)

Oilseed export registrations were:

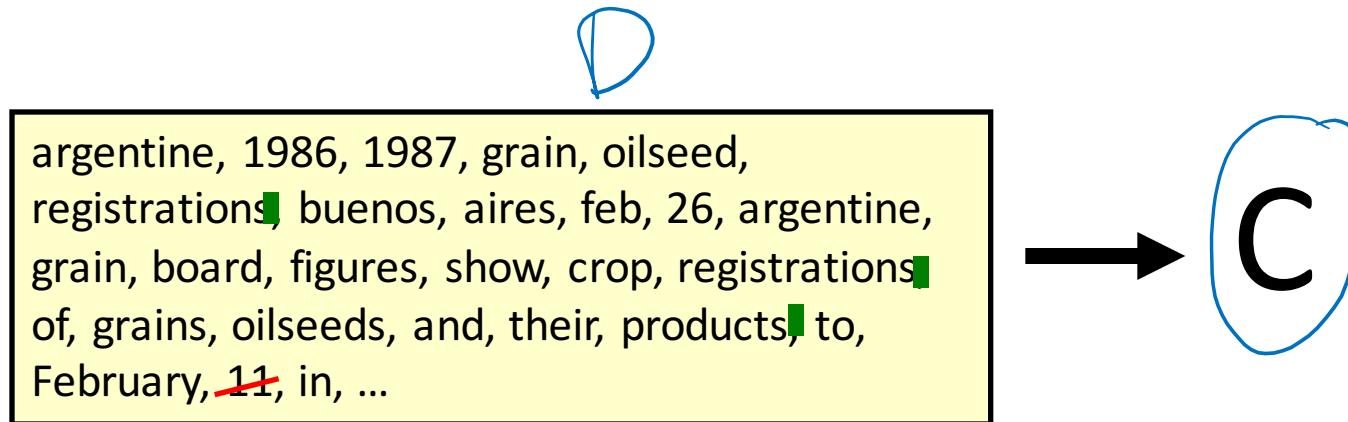
Sunflowerseed total 15.0 (7.9)

Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for subproducts, as follows....

$\{C_1, C_2, \dots, C_{93}\}$

Representing text: a list of words → Dictionary



'Bag of words' representation of text

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).

Maize Mar 48.0, total 48.0 (nil).

Sorghum nil (nil)

Oilseed export registrations were:

Sunflowerseed total 15.0 (7.9)

Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for sub-products, as follows....

Bag of word representation:

Represent text as a vector of word *frequencies*.



<i>word</i>	<i>frequency</i>
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

$$D = (w_1, w_2, \dots, w_k)$$

Another “Bag of words” representation of text → Each dictionary word as Boolean

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).

Maize Mar 48.0, total 48.0 (nil).

Sorghum nil (nil)

Oilseed export registrations were:

Sunflowerseed total 15.0 (7.9)

Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for sub-products, as follows....

<i>word</i>	<i>Boolean</i>
grain(s)	Yes
oilseed(s)	Yes
total	Yes
wheat	Yes
maize	Yes
Panda	No
Tiger	No
...	...

Bag of word representation:

Represent text as a vector of Boolean representing if a word *Exists or NOT.*

$$\mathbf{D} = (w_1, w_2, \dots, w_k)$$

Bag of words

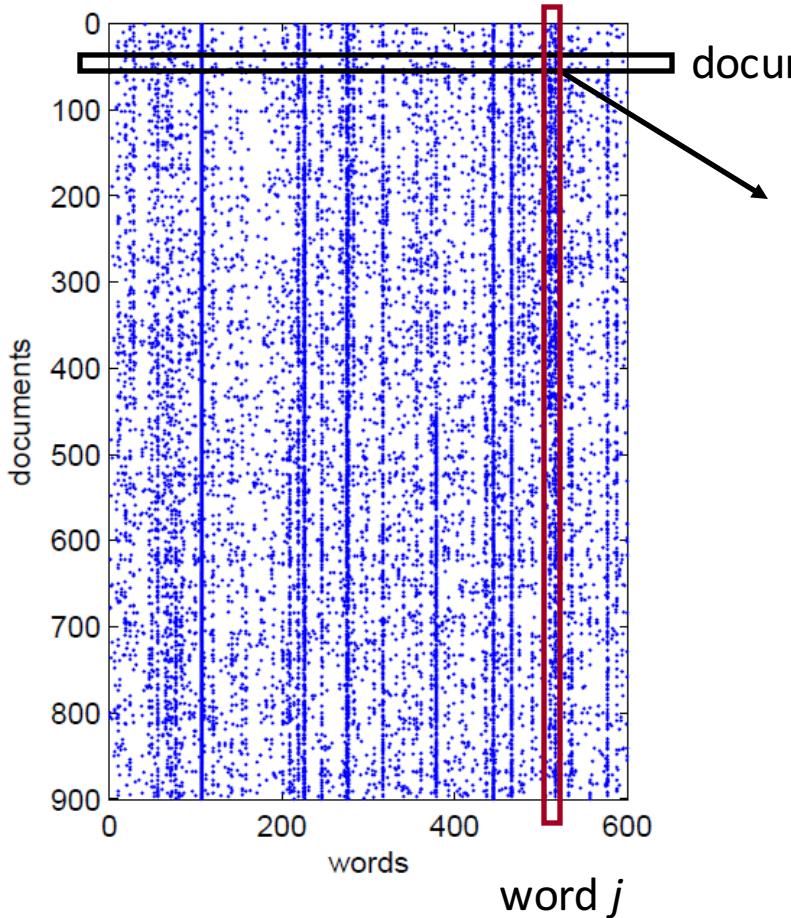
- What simplifying assumption are we taking?

We assumed *word order* is not important.

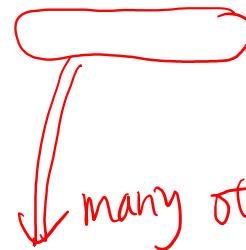
$$D = (w_1, w_2, \dots, w_k)$$

Bag of words representation

BoW - many Variations



$X(i,j) = \text{Frequency of word } j \text{ in document } i$

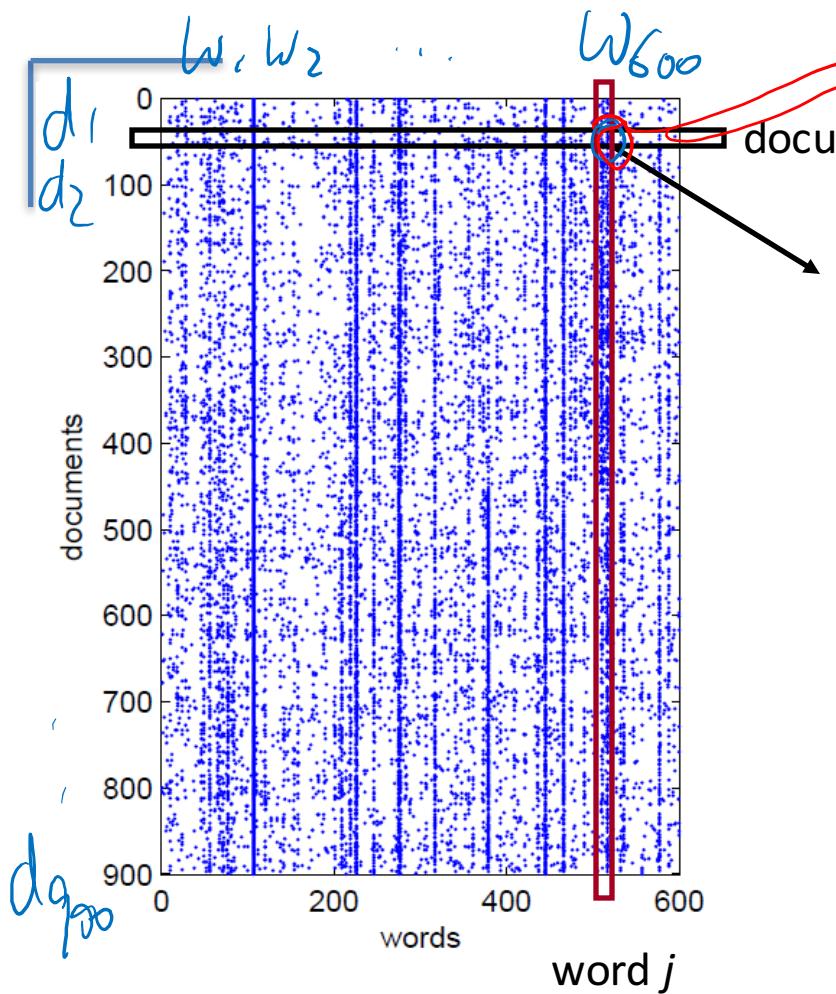


many other choices, e.g.
BM25 / TF-IDF / ...

A collection
of
documents

	X_1	X_2	X_3	C
s_1				
s_2				
s_3				
s_4				
s_5				
s_6				11

Bag of words representation



$$X(d_i, w_j)$$

$X(i,j) = \text{Frequency of word } j \text{ in document } i$

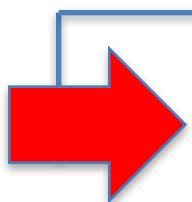
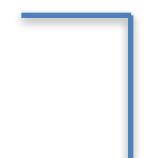
A collection
of
documents

	X_1	X_2	X_3	C
s_1				
s_2				
s_3				
s_4				
s_5				
s_6			12	

Unknown Words

- How to handle words in the **test** corpus that did not occur in the training data, i.e. ***out of vocabulary*** (OOV) words?
- Train a model that includes an **explicit** symbol for an unknown word (<UNK>).
 - Choose a vocabulary in advance and replace **other** (i.e. not in vocabulary) words in the corpus with <UNK>.
 - Many cases <UNK> also used to replace **rare** words

Today : Naïve Bayes Classifier for Text

- 
- ✓ Dictionary based Vector space representation of text article
 - ✓ Multivariate Bernoulli vs. Multinomial
 - ✓ Multivariate Bernoulli
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - ✓ Multinomial naïve Bayes classifier
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models (**Extra**)
- 

'Bag of words' representation of text

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).

Maize Mar 48.0, total 48.0 (nil).

Sorghum nil (nil)

Oilseed export registrations were:

Sunflowerseed total 15.0 (7.9)

Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for sub-products, as follows....

$$D = (w_1, w_2, \dots, w_k)$$

$$\Pr(D | C = c)$$

?

$$\operatorname{argmax}_i P(C_i)$$

$P(d | C_i)$

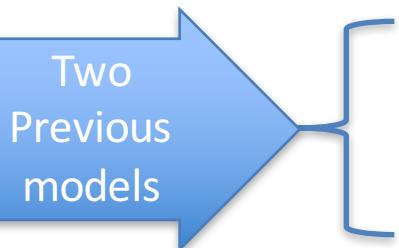
word	frequency
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

‘Bag of words’ representation of text

$$\Pr(D \mid C = c)$$

?

$$D = (w_1, w_2, \dots, w_k)$$

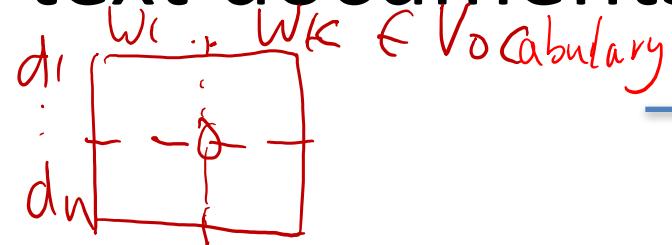


$$\Pr(W_1 = \text{true}, W_2 = \text{false}, \dots, W_k = \text{true} \mid C = c)$$

$$\Pr(W_1 = n_1, W_2 = n_2, \dots, W_k = n_k \mid C = c)$$

Probabilistic Models of text documents

$$\Pr(D | C = c) \quad ?$$



$$D = (w_1, w_2, \dots, w_k)$$

$$\Pr(W_1 = \text{true}, W_2 = \text{false}, \dots, W_k = \text{true} | C = c)$$

Multivariate Bernoulli Distribution

$$\Pr(W_1 = n_1, W_2 = n_2, \dots, W_k = n_k | C = c)$$

Multinomial Distribution

Two
Previous
models

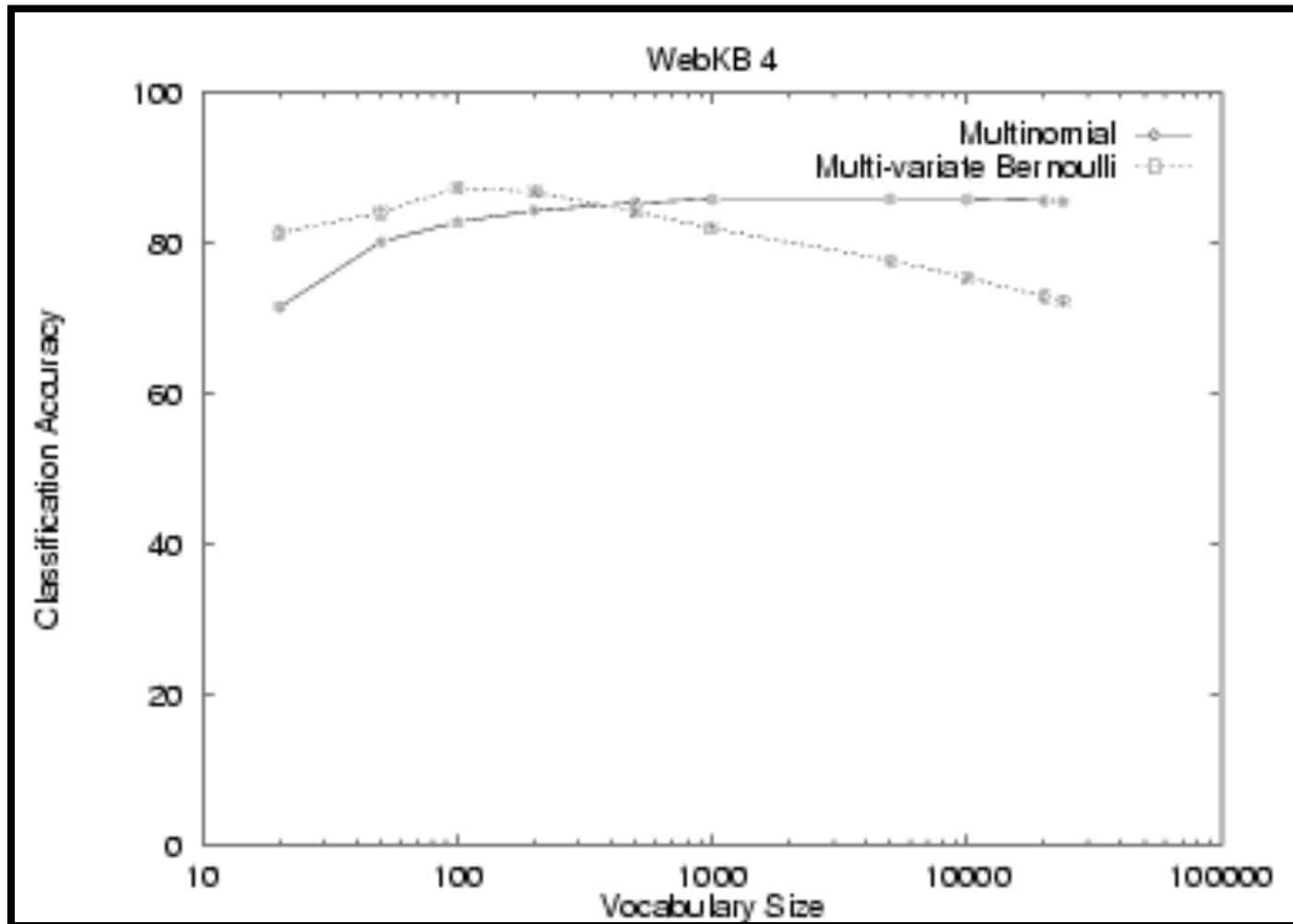
Text Classification with Naïve Bayes Classifier

- Multinomial vs Multivariate Bernoulli?
- Multinomial model is almost always more effective in text applications!

Experiment: Multinomial vs multivariate Bernoulli

- M&N (1998) did some experiments to see which is better
- Determine if a university web page is {student, faculty, other_stuff}
- Train on ~5,000 hand-labeled web pages
 - Cornell, Washington, U.Texas, Wisconsin
- Crawl and classify a new site (CMU)

Multinomial vs. multivariate Bernoulli



Today : Naïve Bayes Classifier for Text

- ✓ Dictionary based Vector space representation of text article
- ✓ Multivariate Bernoulli vs. Multinomial
- ✓ Multivariate Bernoulli
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
- ✓ Multinomial naïve Bayes classifier
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models (**Extra**)

Model 1: Multivariate Bernoulli

- *Model 1: Multivariate Bernoulli*
 - One feature X_w for each word in dictionary
 - $X_w = \text{true}$ in document d if w appears in d
OR not \Rightarrow Binary

Model 1: Multivariate Bernoulli

- Model 1: Multivariate Bernoulli
 - One feature X_w for each word in dictionary
 - $X_w = \text{true}$ in document d if w appears in d
 OR not \Rightarrow Binary
 - Naive Bayes assumption:
 - Given the document's class label, C_j , appearance of one word in the document tells us nothing about chances that another word appears
- $p(d|C)$
- $$= \Pr(W_1 = \text{true}, W_2 = \text{false}, \dots, W_k = \text{true} | C = c)$$
- Dictionary of words²³

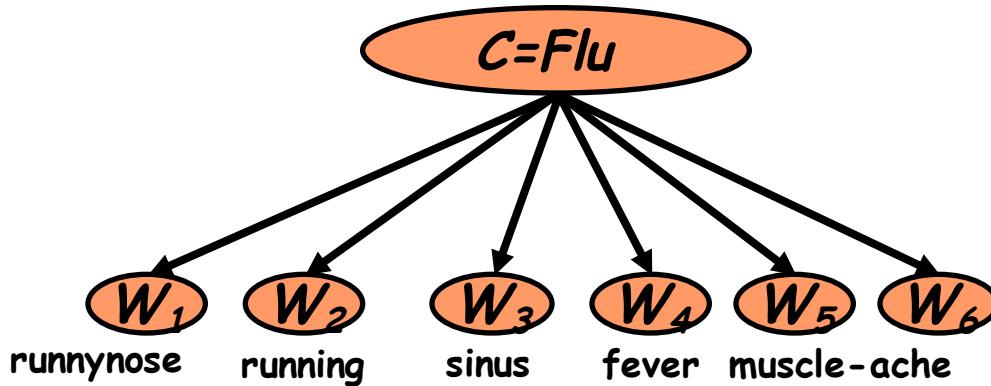
Model 1: Multivariate Bernoulli Naïve Bayes Classifier

$$P(w_1, w_2, \dots, w_k | c) = P(w_1 | c) P(w_2 | c) \dots P(w_k | c)$$

word	True/false
grain(s)	True
oilseed(s)	True
total	True
wheat	True
chemical	False
...	...

- **Conditional Independence Assumption:** Features (word presence) are *independent* of each other given the class variable:
- Multivariate Bernoulli model is appropriate for **binary feature variables**

Model 1: Multivariate Bernoulli



this is
naïve

$$\Pr(W_1 = \text{true}, W_2 = \text{false}, \dots, W_k = \text{true} | C = c) \\ = P(W_1 = \text{true} | C) \cdot P(W_2 = \text{false} | C) \cdot \dots \cdot P(W_k = \text{true} | C)$$

Bernoulli
parameter

Review: Bernoulli Distribution

e.g. Coin Flips

- You flip a coin
 - Head with probability p
 - Binary random variable
 - Bernoulli trial with success probability p

$$\Pr(W_i = \text{true} | C = c_j) = p_{w_i, j}$$

Review: Bernoulli Distribution

e.g. Coin Flips

- You flip a coin
 - Head with probability p
 - Binary random variable
 - Bernoulli trial with success probability p

$$d_{ts} = \dots \{W_1 = \text{true}, W_2 = \text{false}, W_3 = \text{true}\}$$

$$\Pr(W_i = \text{true} | C = c_j) = P_{Wij}.$$

$$P(d_{ts} | c_j) = P_{W1j} (1 - P_{W2j}) P_{W3j}$$

$$P(c_j)$$

Underflow Prevention: log space

- Multiplying lots of probabilities, which are between 0 and 1, can result in **floating-point underflow**.
- Since $\log(xy) = \log(x) + \log(y)$, it is better to perform all computations *by summing logs of probabilities rather than multiplying probabilities*.
- Class with highest final un-normalized log probability score is still the most probable.

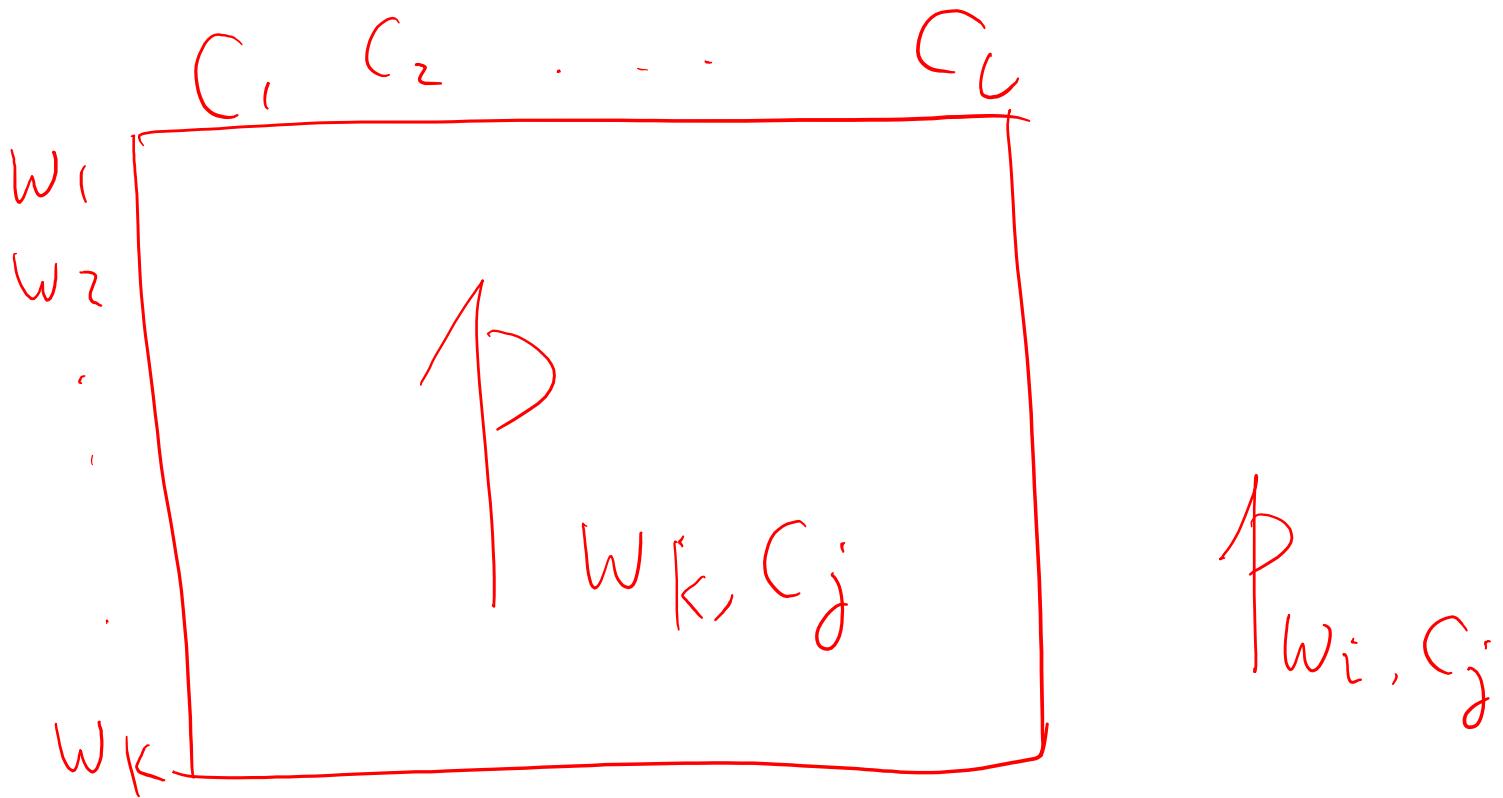
$$c_{NB} = \operatorname{argmax}_{c_j \in C} \log P(c_j) + \sum_{i \in \text{dictionary}} \log P(x_i | c_j)$$

- Note that model is now just **max of sum of weights...**

Today : Naïve Bayes Classifier for Text

- ✓ Dictionary based Vector space representation of text article
- ✓ Multivariate Bernoulli vs. Multinomial
- ✓ Multivariate Bernoulli
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
- ✓ Multinomial naïve Bayes classifier
 - Testing
 - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models
 - Training With Maximum Likelihood Estimation for estimating parameters

ϕ estimated
from data



Review: Maximum Likelihood Estimation

A general Statement

Consider a sample set $T=(X_1 \dots X_n)$ which is drawn from a probability distribution $P(X|\theta)$ where θ are parameters.

If the X s are independent with probability density function $P(X_i|\theta)$, the joint probability of the whole set is

$$P(X_1 \dots X_n | \theta) = \prod_{i=1}^n P(X_i | \theta)$$

this may be maximised with respect to θ to give the maximum likelihood estimates.

The idea is to

- ✓ assume a particular model with unknown parameters, θ
- ✓ we can then define the probability of observing a given event conditional on a particular set of parameters. $P(X_i | \theta)$
- ✓ We have observed a set of outcomes in the real world.
- ✓ It is then possible to choose a set of parameters which are most likely to have produced the observed results.

$$\hat{\theta} = \operatorname{argmax}_{\theta} P(X_1 \dots X_n | \theta)$$

This is maximum likelihood. In most cases it is **both consistent and efficient**. It provides a standard to compare other estimation techniques.

$$\log(L(\theta)) = \sum_{i=1}^n \log(P(X_i | \theta))$$

It is often convenient to work with the Log of the likelihood function.

Review: Bernoulli Distribution

e.g. Coin Flips

- You flip n coins
 - How many heads would you expect
 - Head with probability p
 - Number of heads X out of n trial
 - Each Trial following Bernoulli distribution with parameters p

N trials, e.g. $\{H, H, T, H, H, T, H, T, \dots, H\}$
 $x_1, x_2, x_3, x_4, \dots, x_n$

Review: Defining Likelihood for Bernoulli

- Likelihood = $p(\text{data} \mid \text{parameter})$

→ e.g., for n independent tosses of coins, with **unknown** Bernoulli parameter p

Observed data →
 x heads-up from n trials

function of x_i
 \downarrow
 PMF:

$$f(x_i \mid p) = p^{x_i} (1-p)^{1-x_i}$$

$$x = \sum_{i=1}^n x_i$$

LIKELIHOOD:

$$L(p) = \prod_{i=1}^n p^{x_i} (1-p)^{1-x_i} = p^x (1-p)^{n-x}$$

\uparrow
 function of p

Review: Deriving the Maximum Likelihood Estimate for Bernoulli

$$-l(p) = -\log(L(p)) = -\log[p^x(1-p)^{n-x}]$$

Minimize the negative log-likelihood

$$= -\log(p^x) - \log((1-p)^{n-x})$$

$$= -x \log(p) - (n-x) \log(1-p)$$

$$\hat{p} = \frac{x}{n}$$

i.e. Relative frequency of a binary event

Parameter estimation

- Multivariate Bernoulli model:

$$\hat{P}(w_i = \text{true} | c_j) = \frac{\text{fraction of documents of label } c_j}{\text{in which word } w_i \text{ appears}}$$

- Smoothing to Avoid Overfitting

Today : Naïve Bayes Classifier for Text

- ✓ Dictionary based Vector space representation of text article
- ✓ Multivariate Bernoulli vs. Multinomial
- ✓ Multivariate Bernoulli
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
- ✓ Multinomial naïve Bayes classifier
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models (**Extra**)

Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ representation of text

<i>word</i>	<i>frequency</i>
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

$$\Pr(W_1 = n_1, \dots, W_k = n_k \mid C = c)$$

Can be represented as a multinomial distribution.

In a document class of ‘wheat’, “grain” is more likely, where as in a “hard drive” shipment class the parameter for ‘grain’ is going to be smaller.

Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ representation of text

<i>word</i>	<i>frequency</i>
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

$$\Pr(W_1 = n_1, \dots, W_k = n_k \mid C = c)$$

Can be represented as a multinomial distribution.

Words = like colored balls, there are K possible type of them (i.e. from a dictionary of K words)

In a document class of ‘wheat’ , “grain” is more likely.
where as in a “hard drive” shipment class the parameter for ‘grain’ is going to be smaller.

Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ representation of text

<i>word</i>	<i>frequency</i>
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

$$\Pr(W_1 = n_1, \dots, W_k = n_k \mid C = c)$$

Can be represented as a multinomial distribution.

Words = like colored balls, there are K possible type of them (i.e. from a dictionary of K words)

Document = contains N words, each word occurs n_i times (like a bag of N colored balls)

In a document class of ‘wheat’, “grain” is more likely, where as in a “hard drive” shipment class the parameter for ‘grain’ is going to be smaller.

Multinomial distribution

- The multinomial distribution is a generalization of the binomial distribution.
- The binomial distribution counts successes of an event (for example, heads in coin tosses). K=2
- The parameters:
 - N (number of trials)
 - p (the probability of success of the event)

flip N times of the same coin \Rightarrow

e.g. $N_{\text{Head}} + N_{\text{Tail}} = N$

$$P_{\text{head}} = p$$

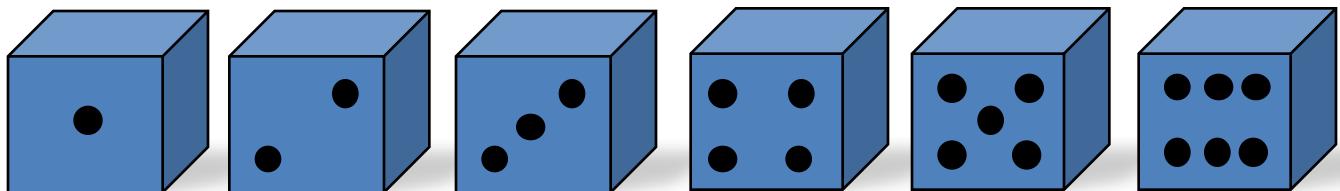
$$P_{\text{tail}} = 1 - p$$



A binomial distribution is the multinomial distribution with $k=2$ and $\theta_1 = p$, $\theta_2 = 1 - \theta_1$

Multinomial distribution

- The multinomial distribution is a generalization of the binomial distribution.
- The binomial distribution counts successes of an event (for example, heads in coin tosses). K=2
- The parameters:
 - N (number of trials)
 - p (the probability of success of the event)
- The multinomial counts the number of a set of events (for example, how many times each side of a die comes up in a set of rolls).
 - The parameters:
 - N (number of trials)
$$N_1 + N_2 + \dots + N_k = N$$
 - $\theta_1 \dots \theta_k$ (the probability of success for each category)K=6



Multinomial Distribution for Text Classification

- W_1, W_2, \dots, W_k are variables

Number of possible orderings of N balls

$$P(W_1 = n_1, \dots, W_k = n_k | c, N, \theta_{1,c}, \dots, \theta_{k,c}) = \frac{N!}{n_1! n_2! \dots n_k!} \theta_{1,c}^{n_1} \theta_{2,c}^{n_2} \dots \theta_{k,c}^{n_k}$$

Multinomial Distribution for Text Classification

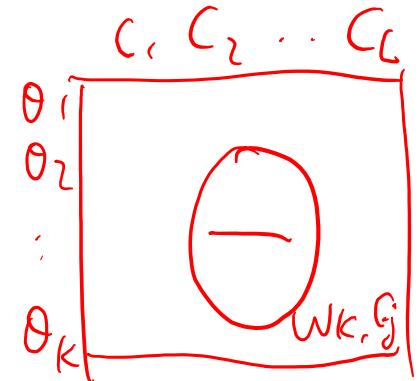
- W_1, W_2, \dots, W_k are variables

$$P(W_1 = n_1, \dots, W_k = n_k | c, N, \theta_{1,c}, \dots, \theta_{k,c}) = \frac{N!}{n_1! n_2! \dots n_k!} \theta_{1,c}^{n_1} \theta_{2,c}^{n_2} \dots \theta_{k,c}^{n_k}$$

Number of possible orderings of N balls

$$\sum_{i=1}^k n_i = N \quad \sum_{i=1}^k \theta_{i,c} = 1$$

Label invariant



Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ – TESTING Stage

<i>word</i>	<i>frequency</i>
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

$$\arg \max_c P(W_1 = n_1, \dots, W_k = n_k, c)$$

$$= \arg \max_c \{ p(c) * \frac{N!}{n_1! n_2! \dots n_k!} \theta_{1,c}^{n_1} \theta_{2,c}^{n_2} \dots \theta_{k,c}^{n_k} \}$$

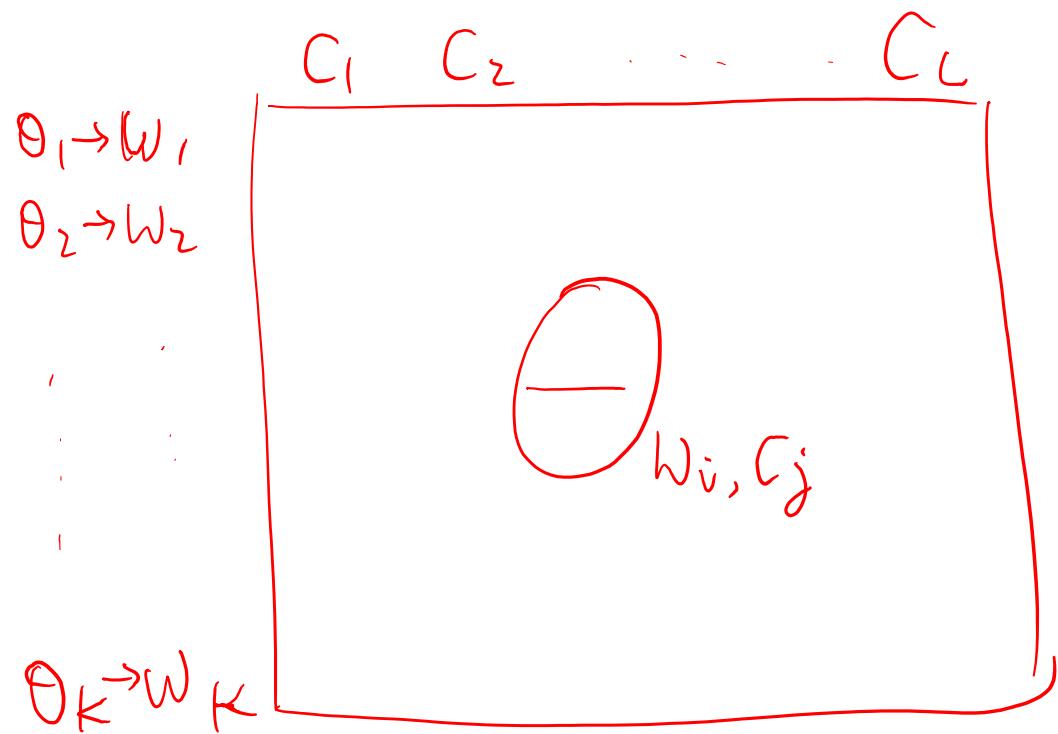
~~$\frac{N!}{n_1! n_2! \dots n_k!}$~~

multinomial coefficient,
invariant with the label c ,
therefore left out

Today : Naïve Bayes Classifier for Text

- ✓ Dictionary based Vector space representation of text article
- ✓ Multivariate Bernoulli vs. Multinomial
- ✓ Multivariate Bernoulli
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
- ✓ Multinomial naïve Bayes classifier
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models (**Extra**)

estimate $\Theta_{K \times L}$ from training data



Deriving the Maximum Likelihood Estimate for multinomial distribution

LIKELIHOOD:

$$\arg \max_{\theta_1, \dots, \theta_k} P(d_1, \dots, d_T | \theta_1, \dots, \theta_k)$$

function of θ

θ vector

$$= \arg \max_{\theta_1, \dots, \theta_k} \prod_{t=1}^T P(d_t | \theta_1, \dots, \theta_k)$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \prod_{t=1}^T \frac{N_{d_t}!}{n_{1,d_t}! n_{2,d_t}! \dots n_{k,d_t}!} \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} \dots \theta_k^{n_{k,d_t}}$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \prod_{t=1}^T \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} \dots \theta_k^{n_{k,d_t}}$$

$$s.t. \sum_{i=1}^k \theta_i = 1$$

Deriving the Maximum Likelihood Estimate for multinomial distribution

$$\arg \max_{\theta_1, \dots, \theta_k} \log(L(\theta))$$

Constrained optimization

$$s.t. \sum_{i=1}^k \theta_i = 1$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \log\left(\prod_{t=1}^T \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} \dots \theta_k^{n_{k,d_t}}\right)$$

Deriving the Maximum Likelihood Estimate for multinomial distribution

$$\arg \max_{\theta_1, \dots, \theta_k} \log(L(\theta))$$

Constrained optimization

$$s.t. \sum_{i=1}^k \theta_i = 1$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \log\left(\prod_{t=1}^T \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} \dots \theta_k^{n_{k,d_t}}\right)$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \sum_{t=1, \dots, T} n_{1,d_t} \log(\theta_1) + \sum_{t=1, \dots, T} n_{2,d_t} \log(\theta_2) + \dots + \sum_{t=1, \dots, T} n_{k,d_t} \log(\theta_k)$$

Deriving the Maximum Likelihood Estimate for multinomial distribution

$$\arg \max_{\theta_1, \dots, \theta_k} \log(L(\theta))$$

Constrained optimization

$$s.t. \sum_{i=1}^k \theta_i = 1$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \log\left(\prod_{t=1}^T \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} \dots \theta_k^{n_{k,d_t}}\right)$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \sum_{t=1, \dots, T} n_{1,d_t} \log(\theta_1) + \sum_{t=1, \dots, T} n_{2,d_t} \log(\theta_2) + \dots + \sum_{t=1, \dots, T} n_{k,d_t} \log(\theta_k)$$

$$\theta_i = \frac{\sum_{t=1, \dots, T} n_{i,d_t}}{\sum_{t=1, \dots, T} n_{1,d_t} + \sum_{t=1, \dots, T} n_{2,d_t} + \dots + \sum_{t=1, \dots, T} n_{k,d_t}} = \frac{\sum_{t=1, \dots, T} n_{i,d_t}}{\sum_{t=1, \dots, T} N_{d_t}}$$

Constrained optimization
MLE estimator

Deriving the Maximum Likelihood Estimate for multinomial distribution

$$\arg \max_{\theta_1, \dots, \theta_k} \log(L(\theta))$$

Constrained optimization

$$s.t. \sum_{i=1}^k \theta_i = 1$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \log\left(\prod_{t=1}^T \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} \dots \theta_k^{n_{k,d_t}}\right)$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \sum_{t=1, \dots, T} n_{1,d_t} \log(\theta_1) + \sum_{t=1, \dots, T} n_{2,d_t} \log(\theta_2) + \dots + \sum_{t=1, \dots, T} n_{k,d_t} \log(\theta_k)$$

$$\theta_i = \frac{\sum_{t=1, \dots, T} n_{i,d_t}}{\sum_{t=1, \dots, T} n_{1,d_t} + \sum_{t=1, \dots, T} n_{2,d_t} + \dots + \sum_{t=1, \dots, T} n_{k,d_t}} = \frac{\sum_{t=1, \dots, T} n_{i,d_t}}{\sum_{t=1, \dots, T} N_{d_t}}$$

Constrained optimization
MLE estimator

- i.e. We can create a mega-document by concatenating all documents d_1 to d_T
- Use relative frequency of w in mega-document

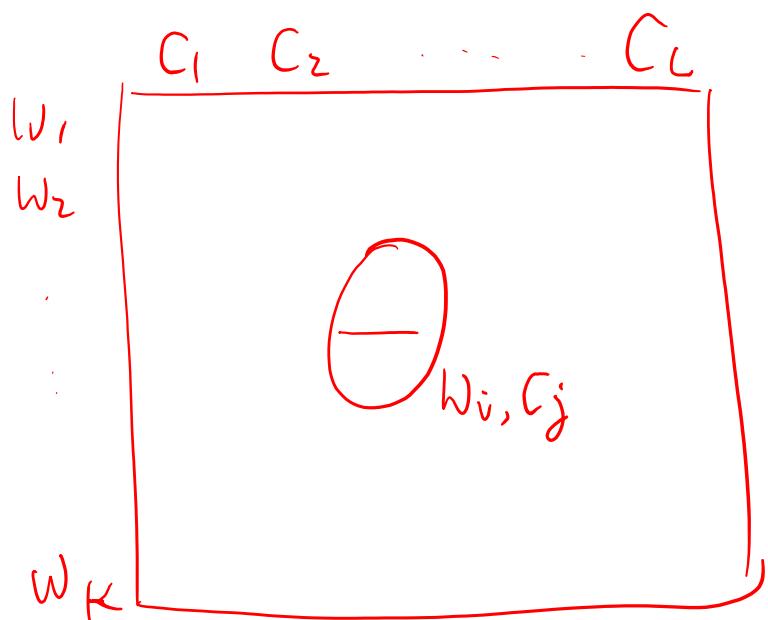
Deriving the Maximum Likelihood Estimate for multinomial Bayes Classifier

LIKELIHOOD:

$$\arg \max_{\theta_1, \dots, \theta_k} P(d_1, \dots, d_T | \theta_1, \dots, \theta_k, C=j)$$

↑
train T documents
 w_1, \dots, w_K

estimate $\theta_{K \times L}$ from training data



Deriving the Maximum Likelihood Estimate for multinomial distribution

$$\arg \max_{\theta_1, \dots, \theta_k} \log(L(\theta))$$

Constrained optimization

$$\text{s.t. } \sum_{i=1}^k \theta_i = 1$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \log\left(\prod_{t=1}^T \theta_1^{n_{1,d_t}} \theta_2^{n_{2,d_t}} \dots \theta_k^{n_{k,d_t}}\right)$$

$$= \arg \max_{\theta_1, \dots, \theta_k} \sum_{t=1, \dots, T} n_{1,d_t} \log(\theta_1) + \sum_{t=1, \dots, T} n_{2,d_t} \log(\theta_2) + \dots + \sum_{t=1, \dots, T} n_{k,d_t} \log(\theta_k)$$

Constrained optimization
MLE estimator

$$\theta_i = \frac{\sum_{t=1, \dots, T} n_{i,d_t}}{\sum_{t=1, \dots, T} n_{1,d_t} + \sum_{t=1, \dots, T} n_{2,d_t} + \dots + \sum_{t=1, \dots, T} n_{k,d_t}} = \frac{\sum_{t=1, \dots, T} n_{i,d_t}}{\sum_{t=1, \dots, T} N_{d_t}}$$

→ i.e. We can create a mega-document by concatenating all documents d_1 to d_T .

→ Use relative frequency of w in mega-document

Parameter estimation

Multinomial model:

$$\hat{P}(X_i = w \mid c_j) = \begin{array}{l} \text{fraction of times in which} \\ \text{word } w \text{ appears} \\ \text{across all documents of topic } c_j \end{array}$$

Can create a mega-document for topic j by concatenating all documents on this topic
Use frequency of w in mega-document

Multinomial : Learning Algorithm for parameter estimation with MLE

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(w_k | c_j)$ terms
 - For each c_j in C do
 - $docs_j \leftarrow$ subset of documents for which the target class is c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

Multinomial : Learning Algorithm for parameter estimation with MLE

- From training corpus, extract *Vocabulary*
- Calculate required $P(c_j)$ and $P(w_k | c_j)$ terms
 - For each c_j in C do
 - $docs_j \leftarrow$ subset of documents for which the target class is c_j

$$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- $Text_j \leftarrow$ is length n_j and is a single document containing all $docs_j$ for class c_j
- for each word w_k in *Vocabulary*

- $n_{k,j} \leftarrow$ number of occurrences of w_k in $Text_j$; n_j is length of $Text_j$

- $P(w_k | c_j) \leftarrow \frac{n_{k,j} + \alpha}{n_j + \alpha |\text{Vocabulary}|}$ e.g., $\alpha = 1$ (Smoothing)

Relative frequency of word w_k appears across all documents of class c_j

Naive Bayes is Not So Naive

- Naïve Bayes: First and Second place in KDD-CUP 97 competition, among 16 (then) state of the art algorithms

Goal: Financial services industry direct mail response prediction model: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.

- Robust to Irrelevant Features

Irrelevant Features cancel each other without affecting results
Instead Decision Trees can **heavily** suffer from this.

- Very good in domains with many equally important features

Decision Trees suffer from *fragmentation* in such cases – especially if little data

- A good dependable baseline for text classification (but not the best)!
- Optimal if the Independence Assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- Very Fast: Learning with one pass of counting over the data; testing linear in the number of attributes, and document collection size
- Low Storage requirements

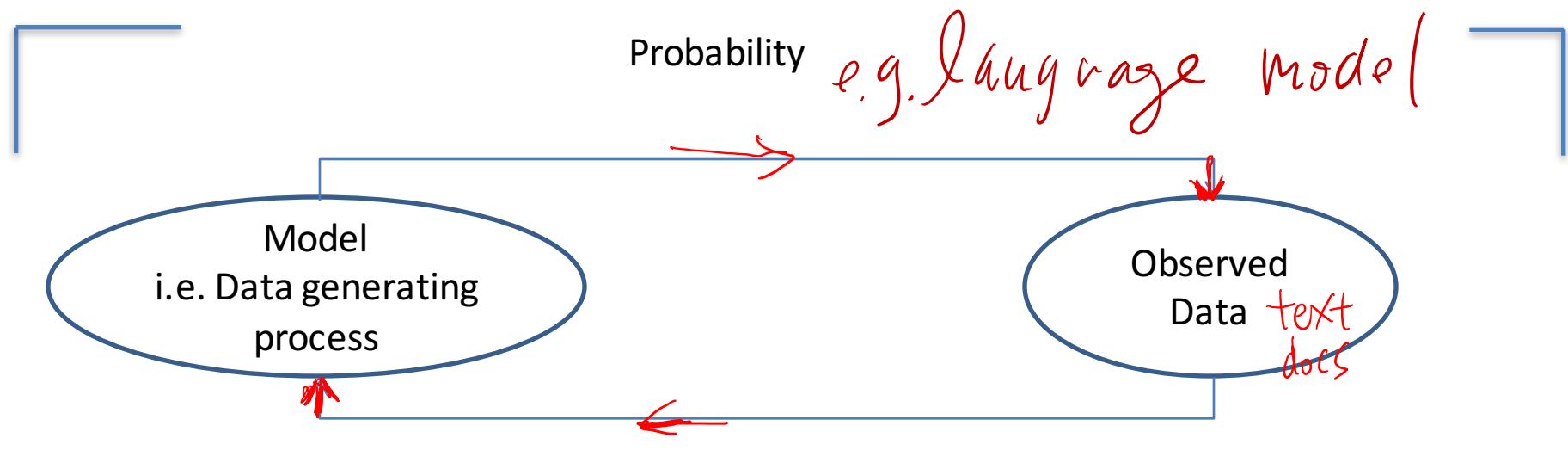
References

- ❑ Prof. Andrew Moore's review tutorial
- ❑ Prof. Ke Chen NB slides
- ❑ Prof. Carlos Guestrin recitation slides
- ❑ Prof. Raymond J. Mooney and Jimmy Lin's slides about language model
- ❑ Prof. Manning's textCat tutorial

Today : Naïve Bayes Classifier for Text

- ✓ Dictionary based Vector space representation of text article
- ✓ Multivariate Bernoulli vs. Multinomial
- ✓ Multivariate Bernoulli
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
- ✓ Multinomial naïve Bayes classifier
 - Testing
 - Training With Maximum Likelihood Estimation for estimating parameters
 - Multinomial naïve Bayes classifier as Conditional Stochastic Language Models (Extra)

The Big Picture



But how to specify a model?

Build a *generative model* that approximates how data is produced.

Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ representation of text

word	frequency
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

WHY is this naïve ???

$$\Pr(W_1 = n_1, \dots, W_k = n_k | C = c)$$

Can be represented as a multinomial distribution.

Words = like colored balls, there are K possible type of them (i.e. from a dictionary of K words)

Document = contains N words, each word occurs n_i times (like a bag of N colored balls)

multinomial coefficient,
normally can leave out
in practical calculations.

$$P(W_1 = n_1, \dots, W_k = n_k | c, N, \theta_1, \dots, \theta_k) = \frac{N!}{n_1! n_2! \dots n_k!} \theta_1^{n_1} \theta_2^{n_2} \dots \theta_k^{n_k}$$

Main Question:

WHY MULTINOMIAL ON TEXT IS NAÏVE PROB. MODELING ?

Multinomial Naïve Bayes as → a generative model that approximates how a text string is produced

- **Stochastic Language Models:**

- Model *probability* of generating strings (each word in turn following the sequential ordering in the string) in the language (commonly all strings over dictionary Σ).
- E.g., unigram model

Model C_1

0.2	the	→ θ_1
0.1	a	→ θ_2
0.01	boy	→ θ_3
0.01	dog	
0.03	said	
0.02	likes	
4/12/18	...	
		→ θ_K

Multinomial Naïve Bayes as → a generative model that approximates how a text string is produced

- Stochastic Language Models:**

- Model *probability* of generating strings (each word in turn following the sequential ordering in the string) in the language (commonly all strings over dictionary Σ).
- E.g., unigram model

Model C_1

0.2	the
0.1	a
0.01	boy
0.01	dog
0.03	said
0.02	likes
	...

$$P(d | C_1) = P(\text{the boy likes the dog} | C_1)$$

Multiply all five terms

$$P(d | C_1) = 0.00000008$$

Multinomial Naïve Bayes as Conditional Stochastic Language Models

- Model conditional *probability* of generating any string from two possible models

Model C1

0.2	the
0.01	boy
0.0001	said
0.0001	likes
0.0001	black
0.0005	dog
0.01	garden

Model C2

0.2	the
0.0001	boy
0.03	said
0.02	likes
0.1	black
0.01	dog
0.0001	garden

$$P(C2) = P(C1) \xrightarrow{\text{from training}}$$

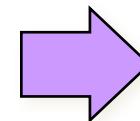
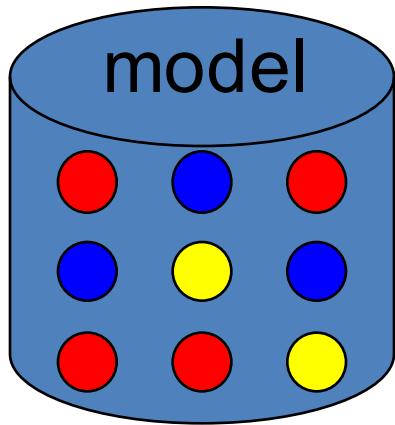
the	boy	likes	black	dog
—	—	—	—	—
0.2	0.01	0.0001	0.0001	0.0005
C1	C1	C1	C1	C1
0.2	0.0001	0.02	0.1	0.01
C2	C2	C2	C2	C2

$$P(d|C2) P(C2) > P(d|C1) P(C1)$$

→ d is more likely to be from class C2

A Physical Metaphor

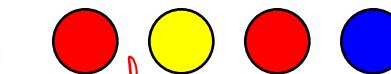
- Colored balls are randomly drawn from (with replacement)



$K=3$

$\{w_1, w_2, w_3\}$ red
 blue yellow
 $\theta_1, \theta_2, \theta_3$

A string of words



dts

{ red 2
yellow 1
blue 1 }

$\{\theta_1, \theta_2, \theta_3\}$ } model
 } paraS

$$P(\text{red yellow red blue}) = P(w_1) P(w_3) P(w_1) P(w_2)$$

$$= \theta_1^2 \theta_2^1 \theta_3^1$$

[Multinomial Distri]

Unigram language model → More general: Generating language string from a probabilistic model

$$\begin{aligned} & [P(\bullet \bullet \bullet \bullet)] \\ & = [P(\bullet | B_1) P(\bullet | B_2 | B_1) P(\bullet | B_3 | B_1 B_2) P(\bullet | B_4 | B_1 B_2 B_3)] \end{aligned}$$

Chain rule

- Unigram Language Models

$$\Rightarrow P(\bullet | B_1) P(\bullet | B_2) P(\bullet | B_3) P(\bullet | B_4)$$

• Easy.
• Effective!

NAÏVE : conditional independent on each position of the string

Unigram model : each position is independent from other positions in the text

Unigram language model → More general: Generating language string from a probabilistic model

$$\begin{aligned} & \boxed{P(\bullet \bullet \bullet \bullet)} \\ & \xrightarrow{\text{Chain rule}} \\ & = \boxed{P(\bullet | B_1) P(\bullet | B_2 | B_1) P(\bullet | B_3 | B_1 B_2) P(\bullet | B_4 | B_1 B_2 B_3)} \end{aligned}$$

- Unigram Language Models

$$\Rightarrow P(\bullet | B_1) P(\bullet | B_2) P(\bullet | B_3) P(\bullet | B_4)$$

• Easy.
• Effective!

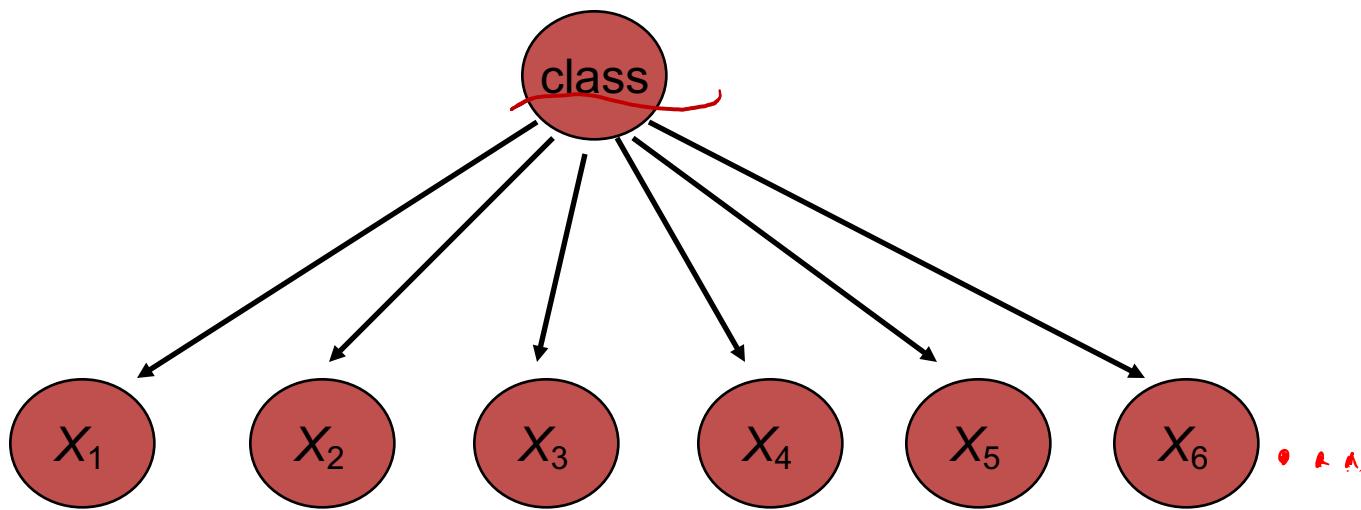
NAÏVE : conditional independent on each position of the string

- Also could be bigram (or generally, n -gram) Language Models

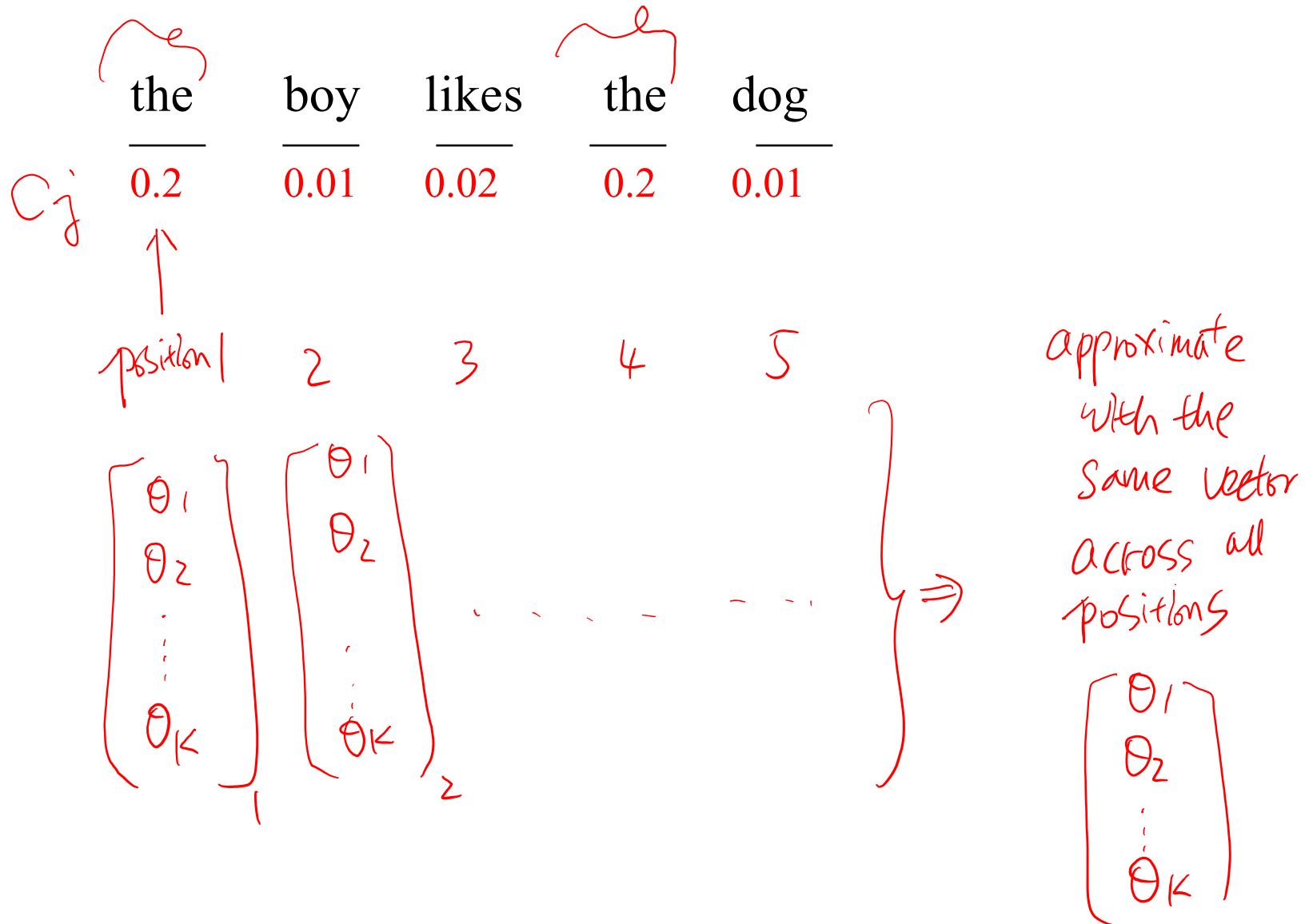
$$P(\bullet | B_1) P(\bullet | B_2 | B_1) P(\bullet | B_3 | B_2) P(\bullet | B_4 | B_3) \dots P(\bullet | B_j | B_{j-1})$$

Whi.
 of
 Virg.
 of

Multinomial Naïve Bayes *Classifier* a class conditional unigram language model



- Think of X_i as the word on the i^{th} position in the document string
- Effectively, the probability of each class is done as a class-specific unigram language model



Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$$\Rightarrow \arg\max P(c_j | X)$$

$$c_{NB} = \arg\max_{c_j \in C} P(c_j) \prod_i P(x_i | c_j)$$

$$= \arg\max_{c_j \in C} P(c_j) P(x_1 = "the" | c_j) \dots P(x_n = "the" | c_j)$$

the boy like the dog

■ Still too many possibilities

- Use same parameters for a word across positions
- Result is bag of words model (over word tokens)

Multinomial Naïve Bayes:

Classifying Step

testing

- Positions \leftarrow all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB} , where

Easy to implement, no need to construct bag-of-words vector explicitly !!!

$P(WK|C_j)$
at position i

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

the	boy	likes	black	dog
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$P(s|C2) P(C2) > P(s|C1) P(C1)$

Multinomial Naïve Bayes: Classifying Step

- Positions \leftarrow all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB} , where

Easy to implement, no need to construct bag-of-words vector explicitly !!!

$$c_{NB} = \operatorname{argmax}_{c_j \in C} P(c_j) \prod_{i \in \text{positions}} P(x_i | c_j)$$

$P(w_k | c_j)$

Equal to, (leaving out of multinomial coefficient)

$$\Pr(W_1 = n_1, \dots, W_k = n_k | C = c_j)$$

the	boy	likes	black	dog
—	—	—	—	—
0.2	0.01	0.0001	0.0001	0.0005

the	boy	likes	black	dog
—	—	—	—	—
0.2	0.0001	0.02	0.1	0.01

$P(s|C2) P(C2) > P(s|C1) P(C1)$

Multinomial Bayes: Time Complexity

- **Training Time:** $O(T^*L_d + |C||V|)$

where L_d is the average length of a document in D .

 - Assumes V and all D_i , n_i , and $n_{k,j}$ pre-computed in $O(T^*L_d)$ time during one pass through all of the data.
 - $|C||V|$ = Complexity of computing all probability values (loop over words and classes)
 - Generally just $O(T^*L_d)$ since usually $|C||V| < T^*L_d$
- **Test Time:** $O(|C| L_t)$

where L_t is the average length of a test document.

 - **Very efficient overall**, linearly proportional to the time needed to just read in all the words.
 - Plus, **robust** in practice

T: num. doc
 $|V|$ dict size
 $|C|$ class size

