

# UVA CS 4774 : Machine Learning

## S3: Lecture 17: Naïve Bayes Classifier for Text Classification

Dr. Yanjun Qi

Module I

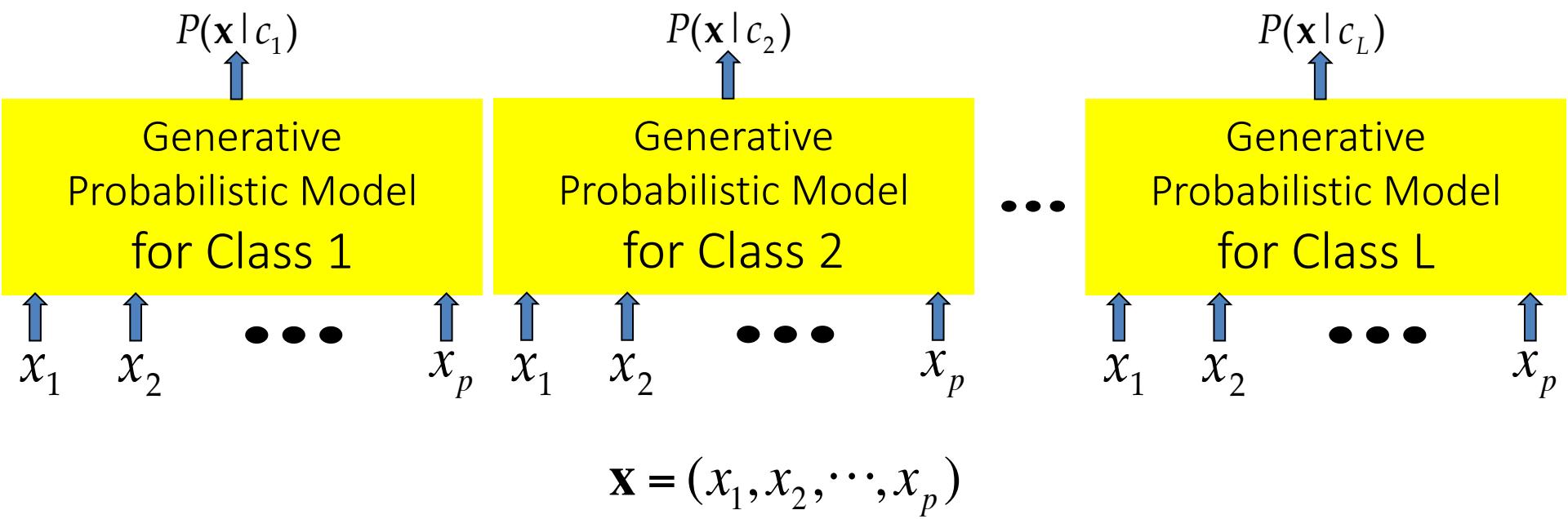
University of Virginia  
Department of Computer Science

# Review: Generative BC

$$\begin{aligned} c^* &= \operatorname{argmax} P(C = c_i | \mathbf{X} = \mathbf{x}) \\ &\propto P(\mathbf{X} = \mathbf{x} | C = c_i) P(C = c_i) \\ &\quad \text{for } i = 1, 2, \dots, L \end{aligned}$$

$$P(\mathbf{X}|C),$$

$$C = c_1, \dots, c_L, \mathbf{X} = (X_1, \dots, X_p)$$



# Review: Naïve Bayes Classifier

$$\operatorname{argmax}_C P(C|X) = \operatorname{argmax}_C P(X,C) = \operatorname{argmax}_C P(X|C)P(C)$$

Naïve  
Bayes  
Classifier

$$P(X_1, X_2, \dots, X_p | C) = P(X_1 | C)P(X_2 | C) \cdots P(X_p | C)$$

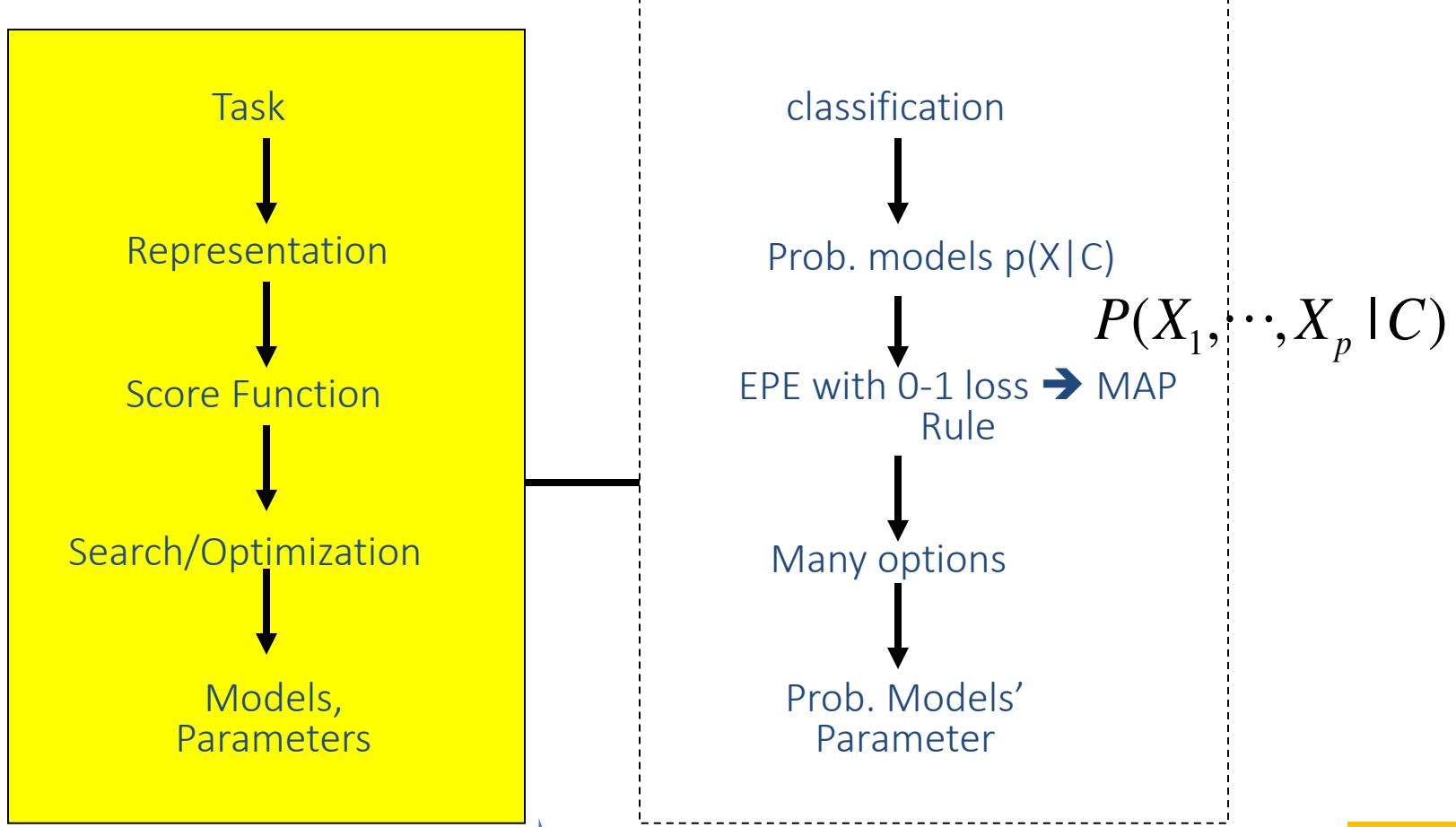
$$c^* = \operatorname{argmax} P(C = c_i | \mathbf{X} = \mathbf{x}) \propto P(\mathbf{X} = \mathbf{x} | C = c_i)P(C = c_i)$$

Assuming all input  
attributes are  
conditionally  
independent given a  
specific class label!

for  $i = 1, 2, \dots, L$

$$\underset{k}{\operatorname{argmax}} P(C_k | X) = \underset{k}{\operatorname{argmax}} P(X, C) = \underset{k}{\operatorname{argmax}} P(X | C)P(C)$$

## Generative Bayes Classifiers



$\rightarrow p(W_i = \text{true} | c_k) = p_{i,k}$

Bernoulli  
Naïve

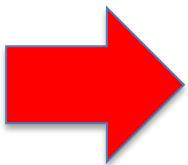
Gaussian  
Naïve

Multinomial

$$\hat{P}(X_j | C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right) \quad (\text{EXTRA})$$

$$P(W_1 = n_1, \dots, W_v = n_v | c_k) = \frac{N!}{n_{1k}! n_{2k}! \dots n_{vk}!} \theta_{1k}^{n_{1k}} \theta_{2k}^{n_{2k}} \dots \theta_{vk}^{n_{vk}}$$

# Today : Naïve Bayes Classifier for Text



- ✓ Dictionary based Vector space representation of text article
- ✓ Multivariate Bernoulli vs. Multinomial
- ✓ Multivariate Bernoulli naïve Bayes classifier
  - 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  $\{C_1, \dots, C_L\}$
- E.g.,
  - Spam filtering Task: Classify  $\text{email}$  as ‘Spam’, ‘Other’.

From: "" <takworlld@hotmail.com>  
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down  
Stop paying rent TODAY !

Change your life NOW by taking a simple course!  
Click Below to order:  
<http://www.wholesaledaily.com/sales/nmd.htm>

$$\rightarrow P(C=\text{spam} | D)$$

# Naive Bayes is Not So Naive

- Naive Bayes won 1<sup>st</sup> and 2<sup>nd</sup> place in KDD-CUP 97 competition out of 16 systems

Goal: Financial services industry direct mail response prediction: Predict if the recipient of mail will actually respond to the advertisement – 750,000 records.
- A good dependable baseline for text classification (but not the best)!
- For most text categorization tasks, there are many relevant features and many irrelevant ones

# 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’ ..
- Classify **movie reviews** as ‘Favorable’, ‘Unfavorable’, ‘Neutral’
- ... and many more.

[Google News](#)

Search for topics, locations &amp; sources

[Top stories](#)[For you](#)[Favorites](#)[Saved searches](#)[U.S.](#)[World](#)[Business](#)[Technology](#)[Entertainment](#)[Sports](#)[Science](#)[Health](#)

10/23/20



## Science

[Follow](#)[Share](#)[Latest](#)[Environment](#)[Outer space](#)[Physics](#)[Genetics](#)[Wildlife](#)

### NASA renames mysterious Ultima Thule after Nazi controversy arises

Fox News · 2 hours ago



- Behold 'Arrokoth' - NASA's New Horizons Target Has Been Officially Named

VideoFromSpace · Yesterday

[View full coverage](#)

### 1st Methane, Now Oxygen: Another Possible 'Biosignature' Gas Is Acting Weird on Mars

Space.com · 6 hours ago



- Curiosity Finds Mysterious Oxygen Fluctuations on Mars

Gizmodo · Yesterday

[View full coverage](#)

# Text Categorization/Classification

- Given:
  - A representation of a text document  $d$ 
    - Issue: how to represent text documents.
    - Usually some type of high-dimensional space – bag of words
  - A fixed set of output classes:
$$C = \{c_1, c_2, \dots, c_J\}$$

# The bag of words representation

f(

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.

) = C

# The bag of words representation

$$f(\begin{array}{|c|c|} \hline \text{great} & 2 \\ \hline \text{love} & 2 \\ \hline \text{recommend} & 1 \\ \hline \text{laugh} & 1 \\ \hline \text{happy} & 1 \\ \hline \dots & \dots \\ \hline \end{array}) = c$$

# Representing text:

a list of words →

## Dictionary

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

Common refinements: **remove stopwords**, **stemming**, collapsing multiple occurrences of words into one....

↙ love ↗  
loves  
loving ↗ love

# Representing text:

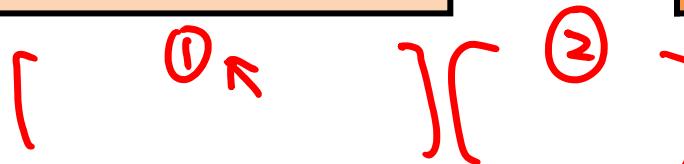
a list of words →

Dictionary

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.



Common refinements: remove stopwords, stemming, collapsing multiple occurrences of words into one....

love → love  
loves → love  
loving → love  
loved → love

# Representing text:

a list of words →

## Dictionary



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

Common refinements: [remove stopwords, stemming, collapsing multiple occurrences of words into one....]

→ [NLTK]

① [ ] ② [ ] ③ [ ]  
collapsing multiple occurrences of words into one....  
  
love  
loves  
loving → love

# ‘Bag of words’ representation of text

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

Bag of word representation:

Represent text as a vector of word frequencies.

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

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

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	Boolean
great	Yes
love	Yes
recommend	Yes
laugh	Yes
happy	Yes
hate	No
...	.

Bag of word representation:

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

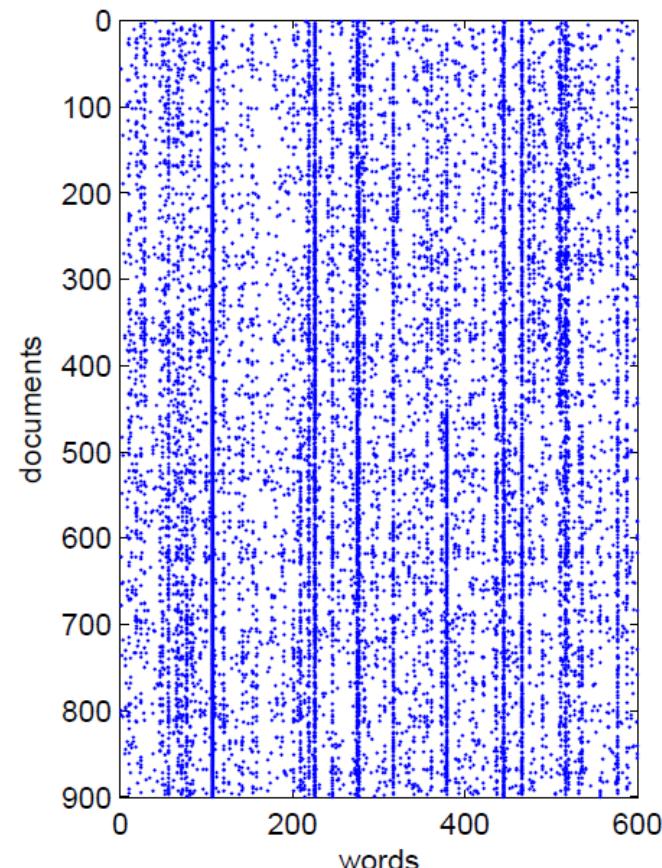
$$D = (w_1, w_2, \dots, w_k)$$

# Bag of words

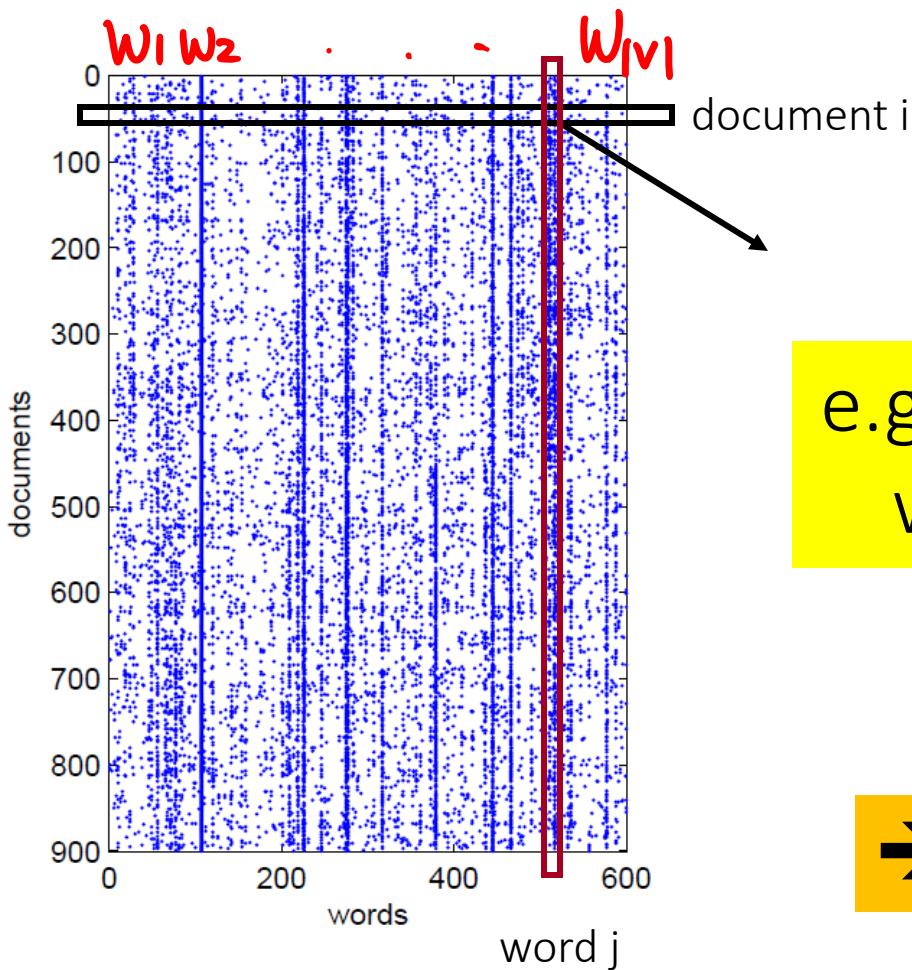
- What simplifying assumption are we taking?

We assumed word order  
is not important.

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



# Bag of words representation

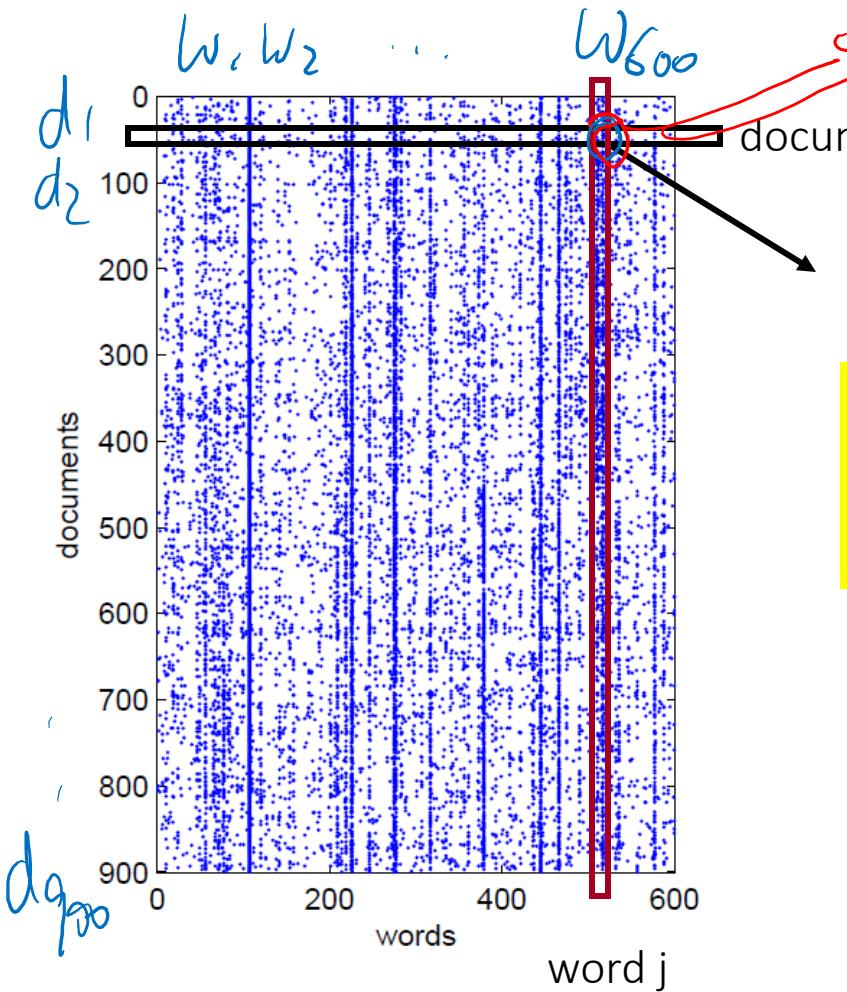


e.g.,  $X(i,j)$  = Frequency of word  $j$  in document  $i$

→ Tabular

	X <sub>1</sub>	X <sub>2</sub>	X <sub>3</sub>	C
S <sub>1</sub>				
S <sub>2</sub>				
S <sub>3</sub>				
S <sub>4</sub>				
S <sub>5</sub>				
S <sub>6</sub>				19

# Bag of words representation



e.g.,  $X(i,j) =$  Frequency of word  $j$  in document  $i$

A collection  
of documents

→ Tabular

	$X_1$	$X_2$	$X_3$	C
$s_1$				
$s_2$				
$s_3$				
$s_4$				
$s_5$				
$s_6$				20

# 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>.
  - Very often, <UNK> also used to replace **rare** words



# Thank You

Thank you

# UVA CS 4774 : Machine Learning

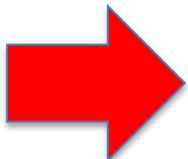
## Lecture 17: Naïve Bayes Classifier for Text Classification

Dr. Yanjun Qi

Module II

University of Virginia  
Department of Computer Science

# Today : Naïve Bayes Classifier for Text

- 
- ✓ Dictionary based Vector space representation of text article
  - ✓ Multivariate Bernoulli vs. Multinomial
  - ✓ Multivariate Bernoulli naïve Bayes classifier
    - 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’ → what probability model?

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	
great	.
love	.
recommend	.
laugh	.
happy	.
...	.

$$\Pr(D = d \mid C = c_i)$$

?

$$c^* = \operatorname{argmax} P(D = d \mid C = c_i) P(C = c_i)$$

# 'Bag of words' → what probability model?

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	
great	.
love	.
recommend	.
laugh	.
happy	.
...	.

$$\underset{i \in \{1, 2, \dots, L\}}{\operatorname{argmax}} P(C_i) \boxed{P(d | C_i)}$$

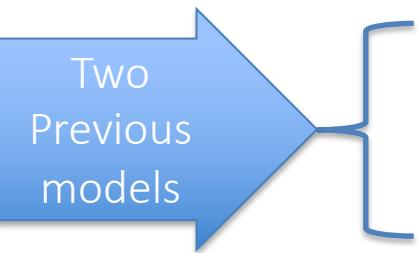
$$D = (w_1, w_2, \dots, w_k)$$

$$\Pr(D = d | C = c_i)$$

$$c^* = \operatorname{argmax} P(D = d | C = c_i) P(C = c_i)$$

‘Bag of words’ → what probability model?

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



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

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

27

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

# Naïve Probabilistic Models of text documents



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

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

Multivariate Bernoulli Distribution

Two  
Previous  
models

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

Multinomial Distribution

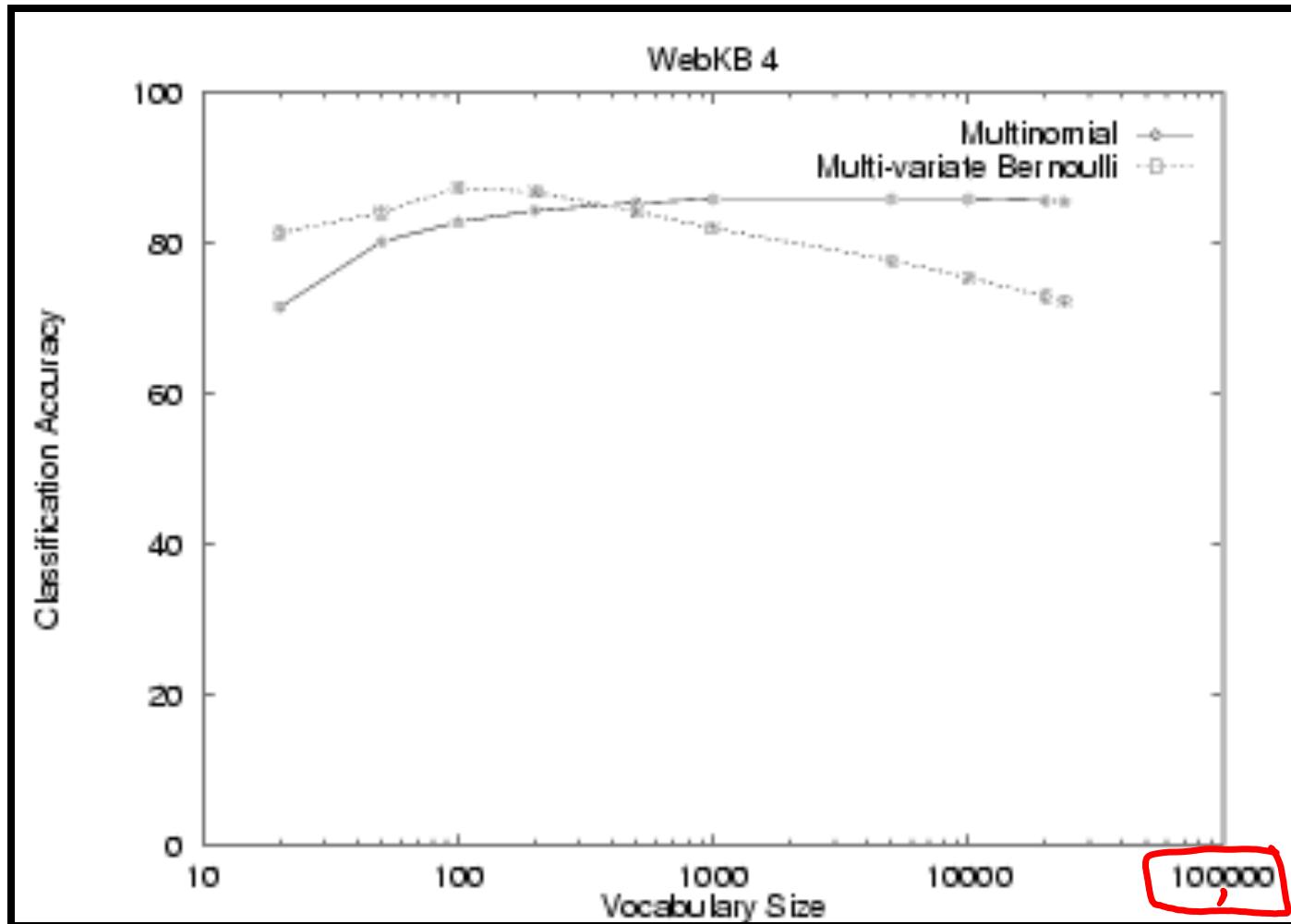
# Text Classification with Naïve Bayes Classifier

- 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\_staff}
- Train on ~5,000 hand-labeled web pages
  - Cornell, Washington, U.Texas, Wisconsin
- Crawl and classify a new site (CMU)

# Multinomial vs. multivariate Bernoulli



```

from sklearn.pipeline import Pipeline
text_clf = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])
text_clf.fit(twenty_train.data, twenty_train.target)

Pipeline(memory=None,
         steps=[('vect',
                  CountVectorizer(analyzer='word', binary=False,
                                  decode_error='strict',
                                  dtype=<class 'numpy.int64'>, encoding='utf-8',
                                  input='content', lowercase=True, max_df=1.0,
                                  max_features=None, min_df=1,
                                  ngram_range=(1, 1), preprocessor=None,
                                  stop_words=None, strip_accents=None,
                                  token_pattern='(?u)\\b\\w\\w+\\b',
                                  tokenizer=None, vocabulary=None)),
            ('tfidf',
                  TfidfTransformer(norm='l2', smooth_idf=True,
                                   sublinear_tf=False, use_idf=True)),
            ('clf',
                  MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True,
                                 verbose=False))
]

```

```

import numpy as np
twenty_test = fetch_20newsgroups(subset='test',
                                 categories=categories, shuffle=True, random_state=42)
docs_test = twenty_test.data
predicted = text_clf.predict(docs_test)
np.mean(predicted == twenty_test.target)

```

0.834886817576564

```

from sklearn import metrics
print(metrics.classification_report(twenty_test.target, predicted,
                                     target_names=twenty_test.target_names))
metrics.confusion_matrix(twenty_test.target, predicted)

```

	precision	recall	f1-score	support
alt.atheism	0.95	0.80	0.87	319
comp.graphics	0.87	0.98	0.92	389
sci.med	0.94	0.89	0.91	396
soc.religion.christian	0.90	0.95	0.93	398
accuracy			0.91	1502
macro avg	0.91	0.91	0.91	1502
weighted avg	0.91	0.91	0.91	1502

```

array([[256, 11, 16, 36],
       [ 4, 380,  3, 21],
       [ 5, 35, 353,  3],
       [ 5, 11,  4, 378]])

```

10/23

I will code run the following notebook I adapted from scikit-learn

[https://colab.research.google.com/drive/1BKmJ4S4QKxlo9leJy0QokjdRSJuz\\_YXx?usp=sharing](https://colab.research.google.com/drive/1BKmJ4S4QKxlo9leJy0QokjdRSJuz_YXx?usp=sharing)

```

[ ] from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(twenty_train.data)
X_train_counts.shape
(2257, 35788)

```

```

[ ] from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
(2257, 35788)

```

```

▶ from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB().fit(X_train_tfidf, twenty_train.target)

```

```

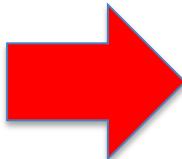
[ ] docs_new = ['Book is love', 'Deep Neural Nets on the GPU is fast']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)

for doc, category in zip(docs_new, predicted):
    print('%r => %s' % (doc, twenty_train.target_names[category]))
'Book is love' => soc.religion.christian
'Deep Neural Nets on the GPU is fast' => comp.graphics

```

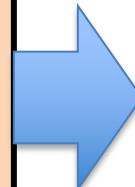
# Today : Naïve Bayes Classifier for Text

- 
- ✓ 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
  - For each word in a dictionary, feature  $x_w$
  - $x_w = \text{true}$  in document  $d$  if  $w$  appears in  $d$

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet.



word	Boolean
great	Yes
love	Yes
recommend	Yes
laugh	Yes
happy	Yes
hate	No

# 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  $d$
  - Naive Bayes assumption:
    - Given the document's class label, appearance of one word in the document tells us nothing about chances that another word appears

$p(w_1|c)p(w_2|c) \dots$

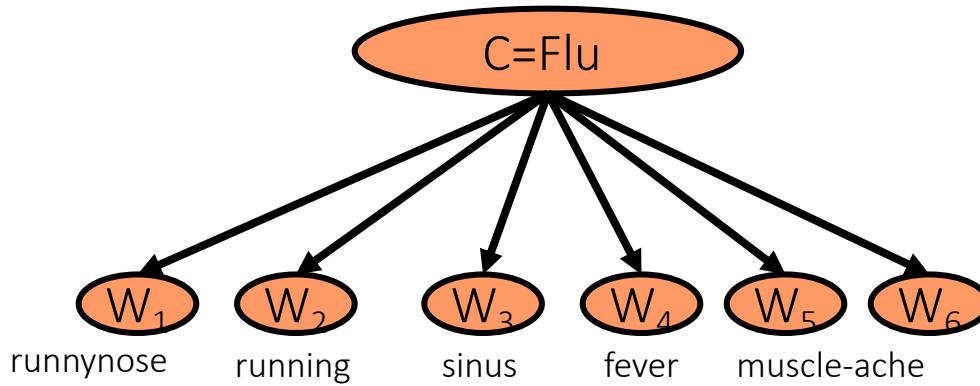
$$\Pr(W_1 = \text{true}, W_2 = \text{false}, \dots, W_k = \text{true} | C = c)$$

# Model 1: Multivariate Bernoulli Naïve Bayes Classifier

word	True/false
great	Yes
love	Yes
recommend	Yes
laugh	Yes
happy	Yes
hate	No
...	.

- 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



$$P(W_i | C_j)$$

$i = 1, \dots, N$   
 $j = 1, \dots, L$

this is  
naïve

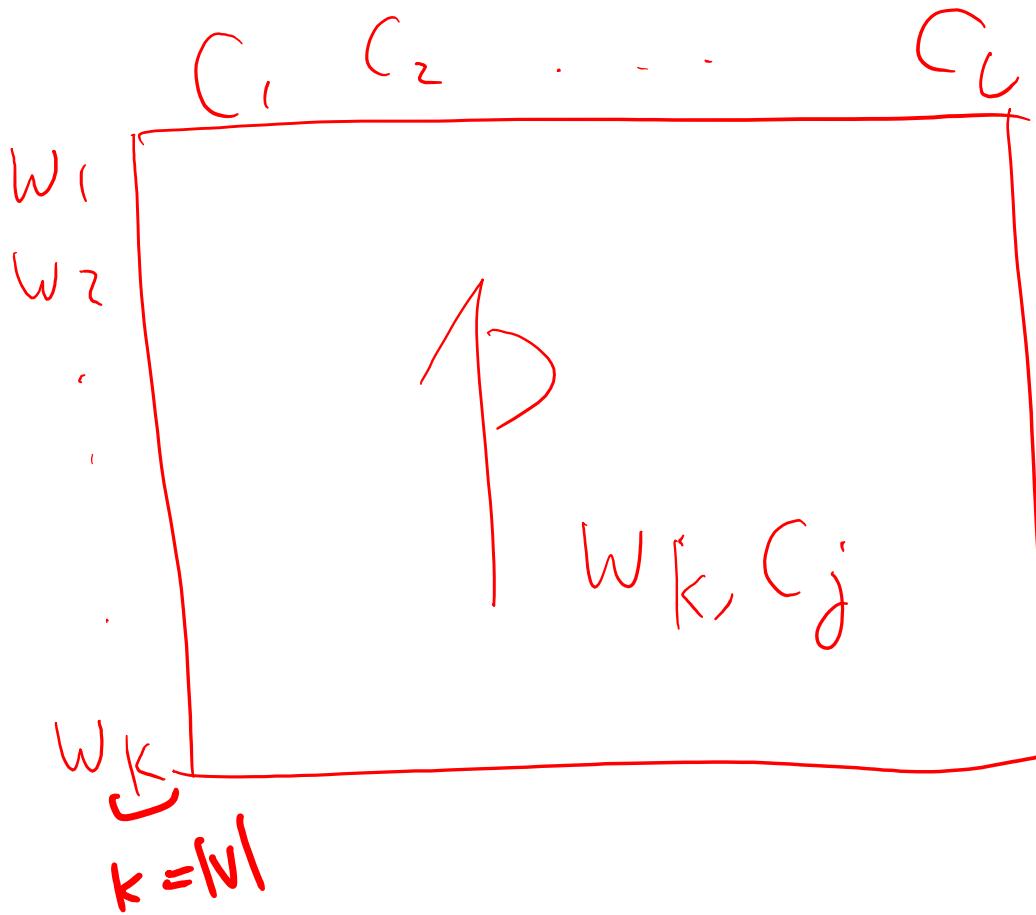
$$\begin{aligned} & \Pr(W_1 = \text{true}, W_2 = \text{false}, \dots, W_k = \text{true} | C = c) \\ &= P(W_1 = \text{true} | C) \cdot P(W_2 = \text{false} | C) \cdot \dots \cdot P(W_k = \text{true} | C) \end{aligned}$$

# Review: Bernoulli Distribution

## e.g. Coin Flips

- You flip a coin
  - Head with probability  $p$
  - Binary random variable
  - Bernoulli trial with success probability  $p$ 
$$\Pr(W_i = \text{true} | C = c)$$
  - How many heads would you expect
  - Number of heads  $X$ : discrete random variable
  - Binomial distribution with parameters  $k$  and  $p$

estimated  
from data



# 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  $p$

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

# Review: Calculating Likelihood

Given:  $\{x_1, x_2, \dots, x_n\}$



$\{H, H, T, \dots, H\}$

$\downarrow$  reformulate

$\{1, 1, 0, \dots, 1\}$

$$p(x_i | \theta) = p^{x_i} (1-p)^{1-x_i} \quad (\text{Here } x_i \in \{0, 1\})$$

# Review: Defining Likelihood for Bernoulli

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

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

Observed data →  
 $x$  heads-up from  $n$  trials

function of  $x_i$

↓

PMF:

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

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

LIKELIHOOD:

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

↑

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

# Review: Deriving the Maximum Likelihood Estimate for Bernoulli

$$\underset{p}{\operatorname{arg\!min}} \{-l(p)\} = \underset{p}{\operatorname{arg\!min}} \left\{ -x \log(p) - (n-x) \log(1-p) \right\}$$

$$\frac{dl(p)}{dp} = -\frac{x}{p} - \frac{(n-x)}{1-p} = 0$$
$$0 = -x + pn$$

$$0 = -\frac{x}{p} + \frac{n-x}{1-p}$$

Minimize the negative log-likelihood

→ MLE parameter estimation

$$0 = \frac{-x(1-p) + p(n-x)}{p(1-p)}$$

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

i.e. Relative frequency of a binary event

$$0 = -x + px + pn - px$$

# 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

# Testing Stage: (Look Up Operations)

$$d_{ts} = \{ w_1 = \text{true}, w_2 = \text{false}, w_3 = \text{true} \}$$
$$P(d_{ts} | c_j) = P_{w1,j} (\neg P_{w2,j}) P_{w3,j}$$
$$P(c_j)$$

# 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.

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

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



Thank You

Thank you

# EXTRA

# UVA CS 4774 : Machine Learning

## Lecture 17: Naïve Bayes Classifier for Text Classification

Module III  
Extra

Dr. Yanjun Qi  
University of Virginia  
Department of Computer Science

# 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)

# 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

# Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ representation of text

word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

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

Can be represented as a multinomial distribution.

# Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ representation of text

word              frequency

word	frequency
great	2
love	2
recommend	1
laugh	1
happy	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 )

# Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ representation of text

word              frequency

word	frequency
great	2
love	2
recommend	1
laugh	1
happy	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 )

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

# Multinomial distribution

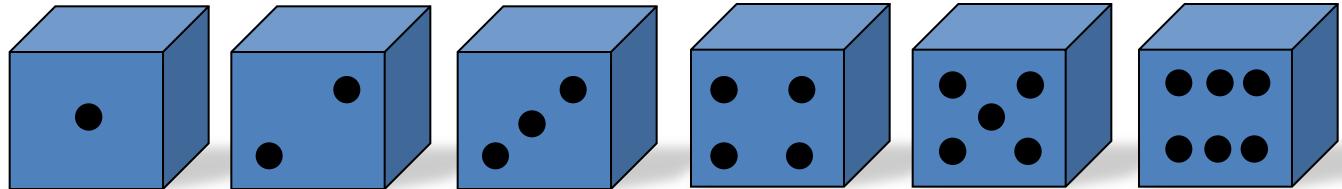
- The **multinomial distribution** is a generalization of the binomial distribution.
- The **binomial distribution** counts successes of an event (for example, heads in N coin tosses).
- The parameters:
  - $N$  (number of trials)
  - $p$  (the probability of success of the event)

$$\theta_1 = p$$

$$\theta_2 = 1 - \theta_1$$



$$\theta_1 \dots \theta_k$$



# 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).
- The parameters:
  - $N$  (number of trials)
  - $p$  (the probability of success of the event)



$$\{H \ H\bar{T} \ H \dots \ H\}_N$$

$$X = \text{Num}_N(H)$$

$$\{0, 1, \dots, N\}$$

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).
- The parameters:
  - $N$  (number of trials)
  - $p$  (the probability of success of the event)



flip  $N$  times of the same Coin  $\Rightarrow$   
 e.g.  $N_{\text{Head}} + N_{\text{Tail}} = N$

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

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

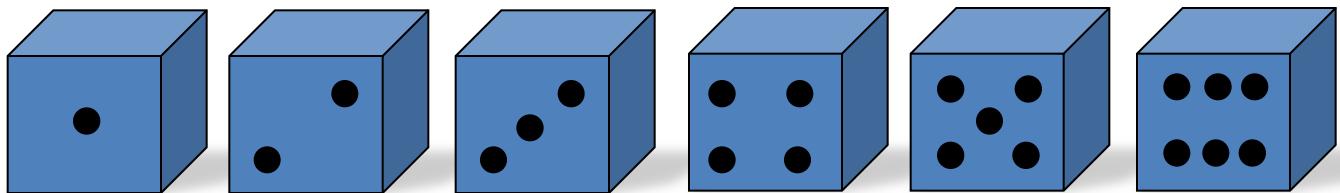
A binomial distribution is the multinomial distribution with  $k=2$  and  $\theta_1 = p$   
 $\theta_2 = 1 - \theta_1$

# 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).
- 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)
    - $\theta_1 \dots \theta_k$  (the probability of success for each category)

$$\textcircled{1} \quad N_1 + N_2 + \dots + N_k = N$$

$$\textcircled{2} \quad \theta_1 + \theta_2 + \dots + \theta_k = 1$$



# 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

$$\textcircled{H} = \{N, \theta_1, \theta_2, \dots, \theta_k | c\}$$

$\downarrow$

$$d_i$$

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

# Multinomial Distribution for Text Classification

- $W_1, W_2, \dots, W_k$  are variables

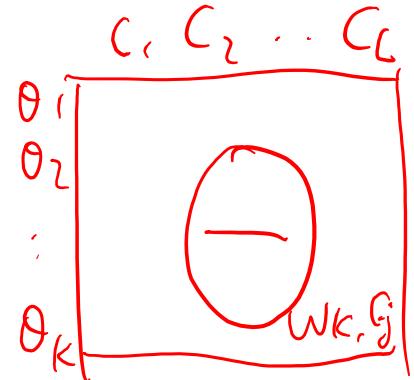
$$P(W_1 = n_1, \dots, W_k = n_k | c, N, \theta_{1,c}, \dots, \theta_{k,c}) = \frac{N!}{n_1! n_2! \dots n_k!} \theta_{1,c}^{n_1} \theta_{2,c}^{n_2} \dots \theta_{k,c}^{n_k}$$

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

Number of possible orderings of N balls

$\frac{N!}{n_1! n_2! \dots n_k!} \theta_{1,c}^{n_1} \theta_{2,c}^{n_2} \dots \theta_{k,c}^{n_k}$

Label invariant



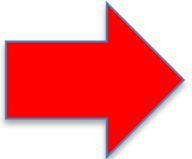
# Model 2: Multinomial Naïve Bayes

- ‘Bag of words’ – TESTING Stage

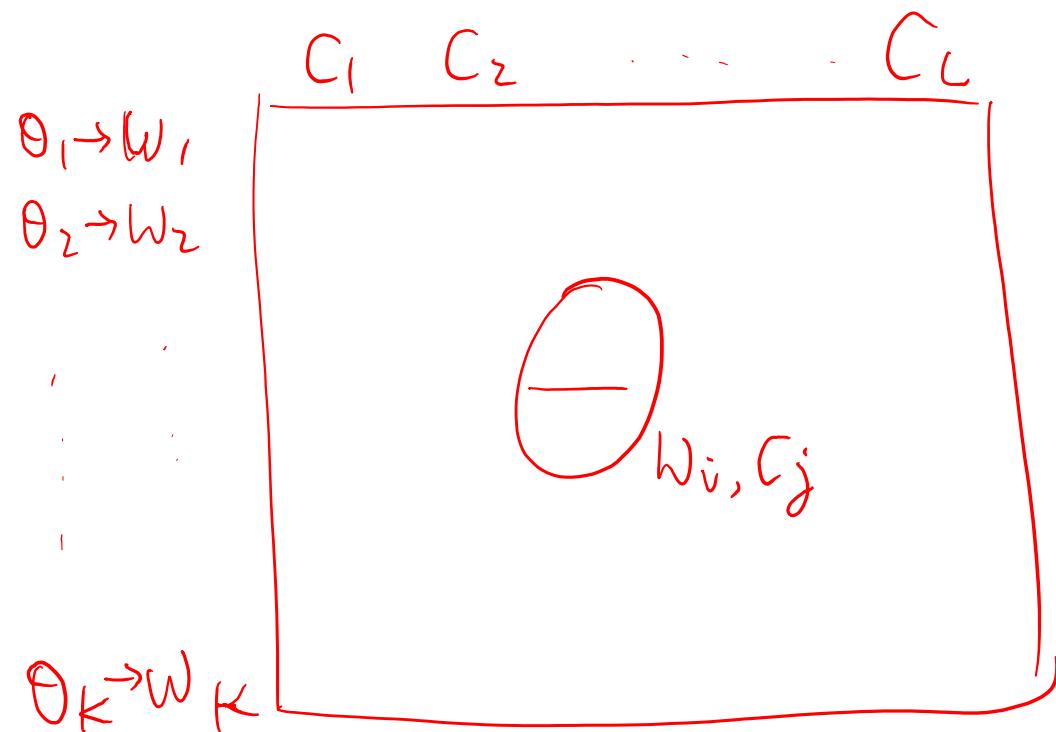
word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

$$\begin{aligned}
 & \underset{c}{\operatorname{argmax}} P(W_1 = n_1, \dots, W_k = n_k, c) \\
 &= \underset{c}{\operatorname{argmax}} \{ p(c) * \theta_{1,c}^{n_1} \theta_{2,c}^{n_2} \dots \theta_{k,c}^{n_k} \}
 \end{aligned}$$

# 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:

function of  $\theta$

$\theta$  vector

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

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

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

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

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

# 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

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

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

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

Constrained optimization  
MLE estimator

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

# Deriving the Maximum Likelihood Estimate for multinomial distribution

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

Constrained optimization

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

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

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

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  $w_i$  in mega-document

# Deriving the Maximum Likelihood Estimate for multinomial distribution

Constrained  
optimization  
MLE estimator

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

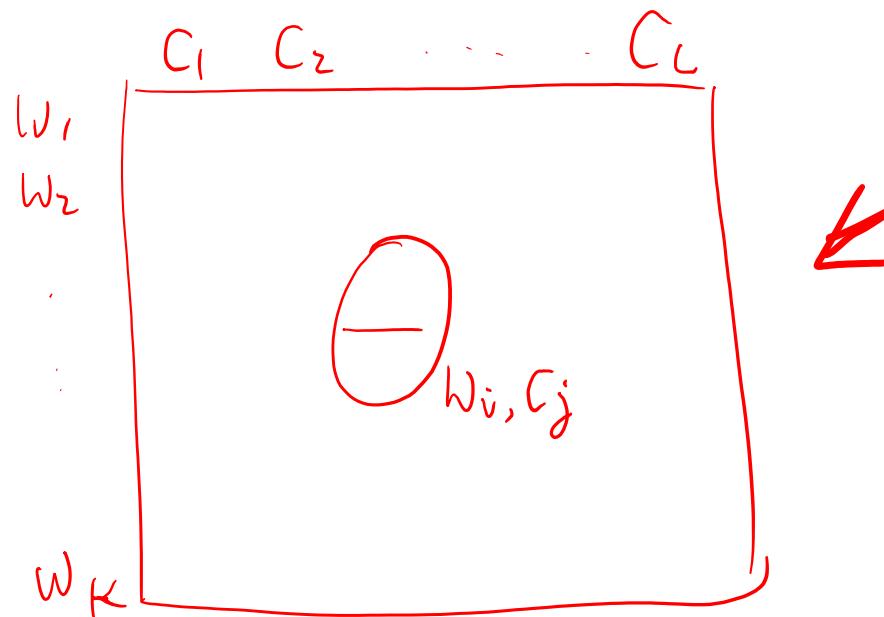
- i.e. We can create a mega-document by concatenating all documents  $d_1$  to  $d_T$
- Use relative frequency of a specific  $w$  in the mega-document

# Deriving the Maximum Likelihood Estimate for multinomial Bayes Classifier

LIKELIHOOD:

$$\underset{\theta_{1,Cj}, \dots, \theta_{k,Cj}}{\operatorname{argmax}} P(d_1, \dots, d_T | \Theta)$$

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



# Parameter estimation

## Multinomial model:

$$\hat{P}(X_{\underline{i}} = w_{\textcolor{red}{i}} \mid c_j) = \textcolor{red}{\theta}_{w_i, c_j}$$

fraction of times in which each dictionary word  $w$  appears across all documents of class  $c_j$

- Can create a mega-document for class  $j$  by concatenating all documents on this class,
- 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
    - $\text{docs}_j \leftarrow$  subset of documents for which the target class is  $c_j$

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

- $\text{Text}_j \leftarrow$  is length  $n_j$  and is a single document containing all  $\text{docs}_j$  for class  $c_j$
- for each word  $w_k$  in Vocabulary
  - $n_{k,j} \leftarrow$  number of occurrences of  $w_k$  in  $\text{Text}_j$ ;  $n_j$  is length of  $\text{Text}_j$
  - $P(w_k | c_j) \leftarrow \frac{n_{k,j} + \alpha}{n_j + \alpha |\text{Vocabulary}|}$  e.g.,  $\alpha = 1$  (Smoothing)

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

# Multinomial Bayes: Time Complexity

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

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

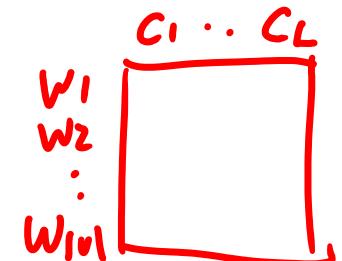
- Assumes V and all  $D_i$ ,  $n_i$ , and  $n_{k,j}$  pre-computed in  $O(T * L_d)$  time during one pass through all of the data.
- $|C| |V|$  = Complexity of computing all probability values (loop over words and classes)
- Generally just  $O(T * L_d)$  since usually  $|C| |V| < T * L_d$

- **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



# Recap: Multinomial Naïve Bayes

- ‘Bag of words’ – TESTING Stage

word	frequency
great	2
love	2
recommend	1
laugh	1
happy	1
...	.

$$\begin{aligned} & \underset{c}{\operatorname{argmax}} P(W_1 = n_1, \dots, W_k = n_k, c) \\ &= \underset{c}{\operatorname{argmax}} \{ p(c) * \theta_{1,c}^{n_1} \theta_{2,c}^{n_2} \dots \theta_{k,c}^{n_k} \} \end{aligned}$$

① BOW

$\{w_i, w_j, \dots, w_{Ld}\}$  ↪  
 very very good  
 (2)  $p(c) p(w_{p_1}) p(w_{p_2}) \dots p(w_{p_{Ld}})$   
 $p(\text{very}) p(\text{very}) p(\text{good})$

# Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$$\begin{aligned} c_{NB} &= \operatorname{argmax}_{c_j \in C} P(c_j) \prod_i P(x_i | c_j) \\ &= \operatorname{argmax}_{c_j \in C} P(c_j) P(x_1 = \text{"the"} | c_j) \cdots P(x_n = \text{"the"} | c_j) \end{aligned}$$

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

Low Storage! Since we don't need to save the BOW version of dataset at all

# Multinomial Naïve Bayes: Classifying Step

- Positions ← all word positions in current document which contain tokens found in Vocabulary

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

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

Equal to,  
(without the coefficient)

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

# An Example: Model conditional probability of generating a word string from two possible classes (models)

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

→ d is more likely to be from class C2

Model C1

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

10/23/20

Model C2

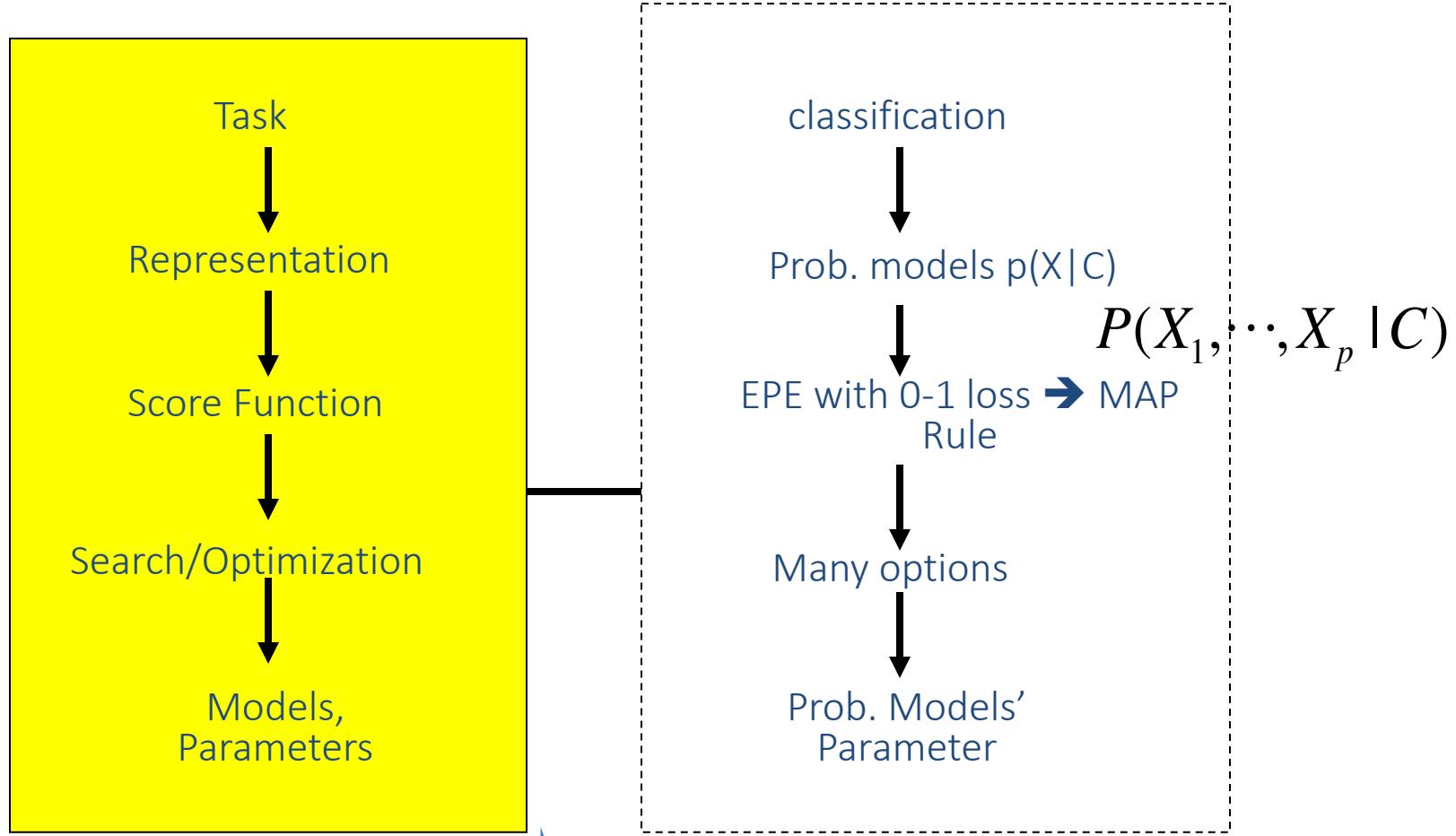
0.2	the
0.0001	boy
0.03	said
0.02	likes
0.1	black
0.01	dog
0.0001	garden

$P(C2) = P(C1)$  ↗ [from training]

the	boy	likes	black	dog
—	—	—	—	—
$C_1 0.2$	0.01	0.0001	0.0001	0.0005
$C_2 0.2$	0.0001	0.02	0.1	0.01

$$\underset{k}{\operatorname{argmax}} P(C_k | X) = \underset{k}{\operatorname{argmax}} P(X, C) = \underset{k}{\operatorname{argmax}} P(X | C)P(C)$$

## Generative Bayes Classifier



Bernoulli  
Naïve

$p(W_i = \text{true} | c_k) = p_{i,k}$

Gaussian  
Naïve

Multinomial

$$\hat{P}(X_j | C = c_k) = \frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp\left(-\frac{(X_j - \mu_{jk})^2}{2\sigma_{jk}^2}\right)$$

$$P(W_1 = n_1, \dots, W_v = n_v | c_k) = \frac{N!}{n_{1k}! n_{2k}! \dots n_{vk}!} \theta_{1k}^{n_{1k}} \theta_{2k}^{n_{2k}} \dots \theta_{vk}^{n_{vk}}$$



# Thank You

Thank you

# UVA CS 4774 : Machine Learning

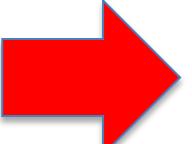
## Lecture 17: Naïve Bayes Classifier for Text Classification

Module IV  
Extra

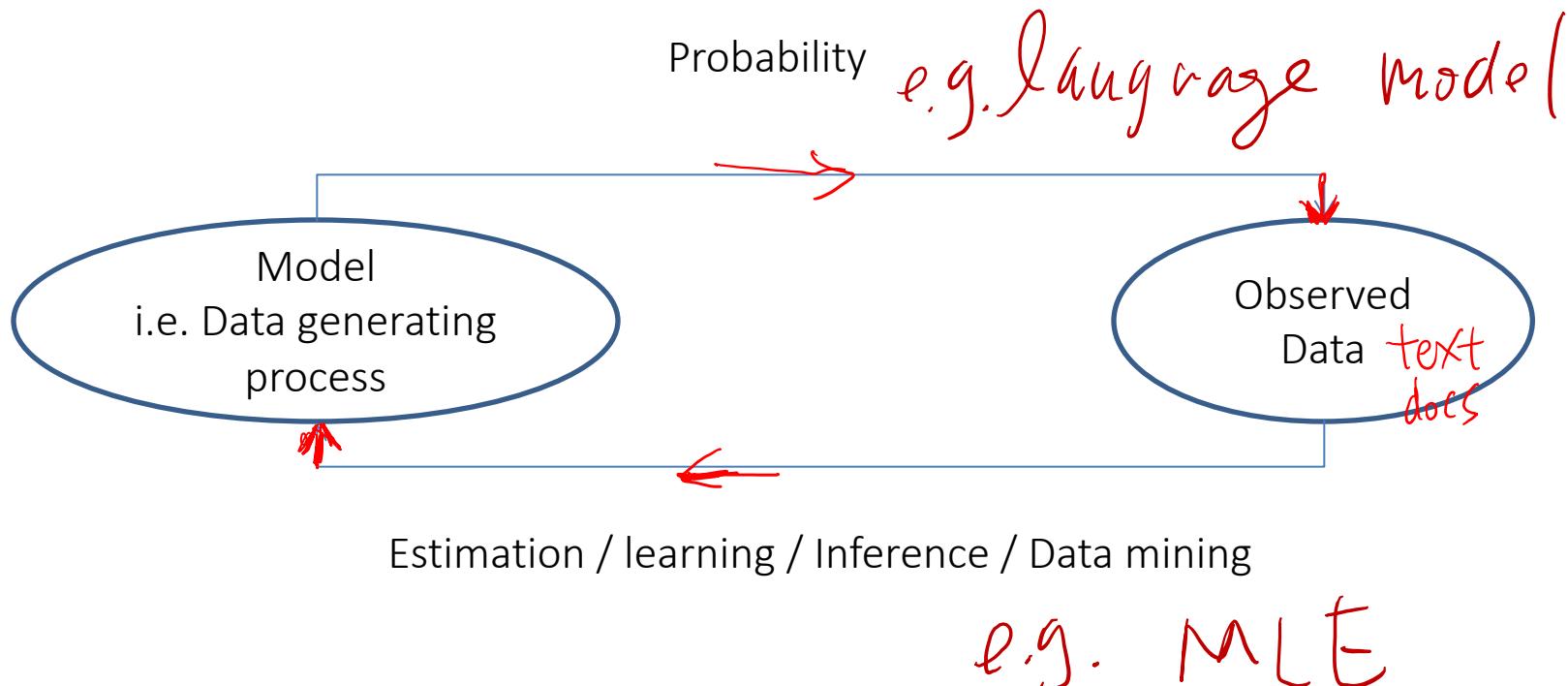
Dr. Yanjun Qi

University of Virginia  
Department of Computer Science

# 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.

GBC Models	$ x_i  = k$ $i, \dots, p$	$P(c_j)$ $j=1, \dots, L$	$P(x_1 x_2 \dots x_p   c_j)$ #
GBC discrete	$ x_i  = k$	# $O(L)$	$k^p \times L$
NBC discrete naive	$ x_i  = k$	$O(L)$	$k^p \times L$
Naive Gaussian	$N(\mu_i, \Lambda_i)$ $\downarrow$ $p \times 1$ $p \times p$	$O(L)$	$2p \times L$
LDA	$N(\mu_i, \Sigma)$	$O(L)$	$p \times L + p^2/2$
QDA	$N(\mu_i, \Sigma_i)$	$O(L)$	$(p+p^2) \times L$
Multinomial BC	$\theta_1, \dots, \theta_k   c$	$O(L)$	$ V  \times L$

# 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 ???

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

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

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.

Why  
naïve

???

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  $\Sigma$ ).
- E.g., unigram model

Model C\_1

0.2	the	→ $\theta_1$
0.1	a	→ $\theta_2$
0.01	boy	→ $\theta_3$
0.01	dog	
0.03	said	
0.02	likes	
10/23/20	...	

→  $\theta_K$

$\Sigma_a$

# 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  $\Sigma$ ).
- 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

10/23/20

Model C2	
0.2	the
0.0001	boy
0.03	said
0.02	likes
0.1	black
0.01	dog
0.0001	garden

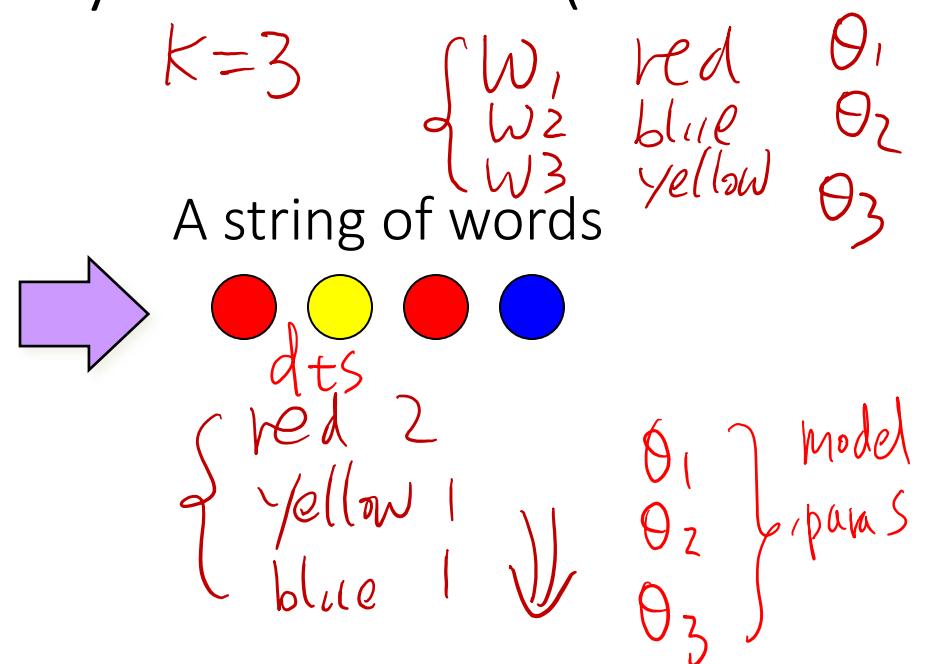
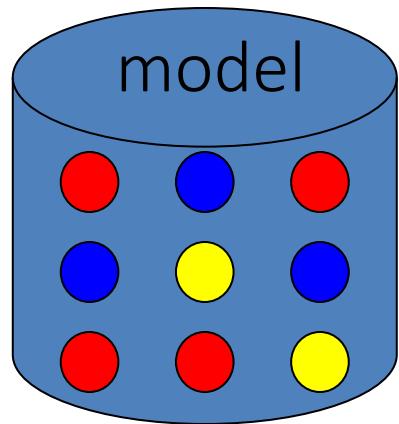
P(C2) = P(C1)	→ [from training]
the	boy
—	—
0.2	0.01
C1	0.0001
—	0.0001
0.2	0.02
C2	0.0001
—	0.1
0.0001	0.0005
—	0.01
0.0001	0.0001
—	0.0001
0.0001	0.01
—	0.0001

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

→ d is more likely to be from class C2

# A Physical Metaphor

- Colored balls are randomly drawn from (with replacement)



$$P(\underbrace{\bullet \bullet dts}_{\text{dts}}) = P(w_1) P(w_3) P(w_1) P(w_2)$$

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

[Multinomial Distri]

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

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

- 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

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

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

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

- Unigram Language Models

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

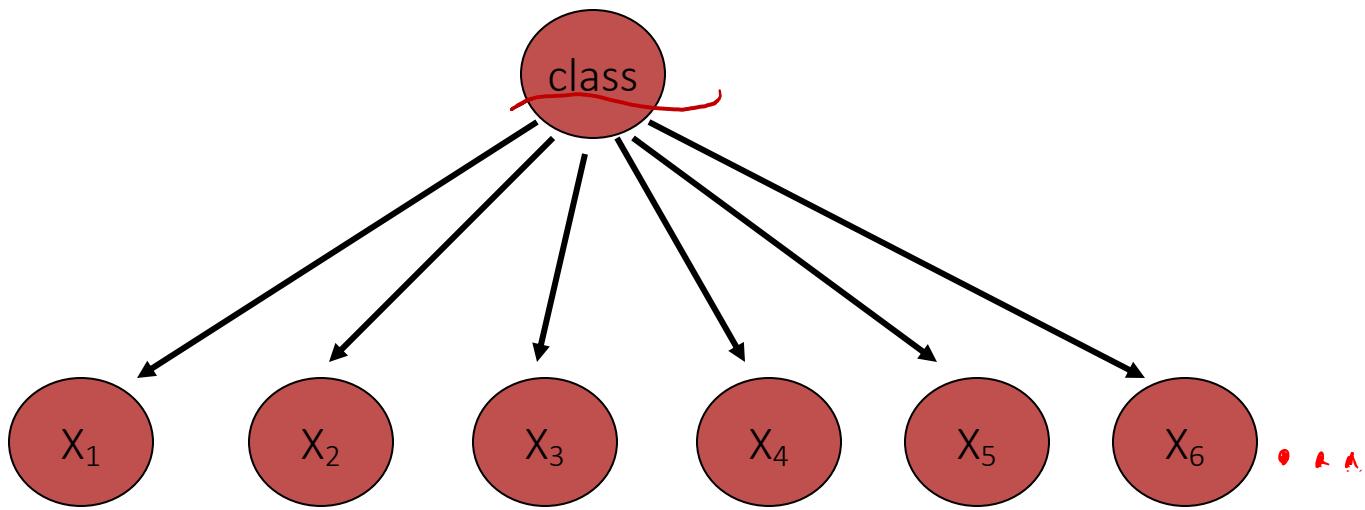
• Easy.  
• Effective!

NAÏVE : conditional independent on each position of the string

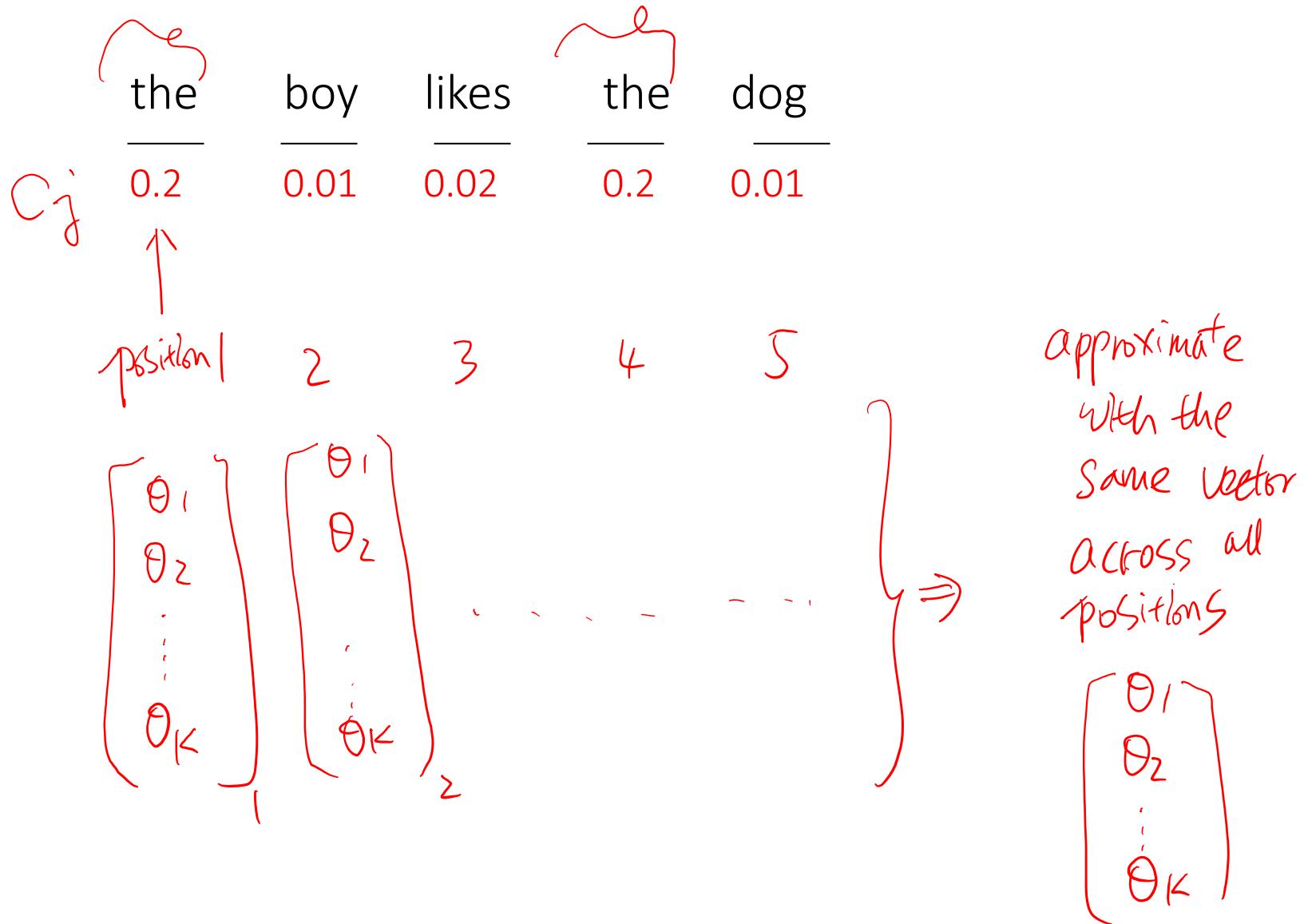
- Also could be bigram (or generally,  $n$ -gram) Language Models

$$\begin{aligned} & B_1 \quad B_2 | B_1 \quad B_3 | B_2 \quad B_4 | B_3 \quad B_j | B_{j-1} \\ & P(\bullet | B_1) P(\bullet | B_2 | B_1) P(\bullet | B_3 | B_2) P(\bullet | B_4 | B_3) \\ & \text{whi.v.} \quad \text{of} | \text{univ.} \quad \text{Virg.| of} \end{aligned}$$

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



- Think of  $X_i$  as the word on the  $i^{\text{th}}$  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.

$$\Rightarrow \text{argmax } P(c_j | X)$$

$$\begin{aligned}
 c_{NB} &= \underset{c_j \in C}{\text{argmax}} P(c_j) \prod_i P(x_i | c_j) \\
 &= \underset{c_j \in C}{\text{argmax}} P(c_j) P(x_1 = "the" | c_j) \cdots P(x_n = "the" | c_j)
 \end{aligned}$$

- 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 !!!

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

*P(WK|C<sub>j</sub>) at position i*

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

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

# 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 !!!

$$P(w_k | c_j)$$

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

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
0.2	0.0001	0.02	0.1	0.01

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

# 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