UVA CS 4774 : Machine Learning

S3: Lecture 17: Naïve Bayes Classifier for Text Classification

Module I

Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Review: Generative BC

$$c^* = argmax \;\; P(C = c_i|\mathbf{X} = \mathbf{x})$$
$$\propto P(\mathbf{X} = \mathbf{x}|C = c_i)P(C = c_i)$$
for $i = 1,2,\cdots,L$

$$P(\mathbf{X}|C),$$
$$C = c_1,\cdots,c_L, \mathbf{X} = (X_1,\cdots,X_p)$$

$P(\mathbf{x}|c_1)$ | $P(\mathbf{x}|c_2)$ | $P(\mathbf{x}|c_L)$

Generative Probabilistic Model for Class 1 | Generative Probabilistic Model for Class 2 | ••• | Generative Probabilistic Model for Class L

x_1 x_2 ••• x_p | x_1 x_2 ••• x_p | x_1 x_2 ••• x_p

$$\mathbf{x} = (x_1, x_2, \cdots, x_p)$$

Adapt from Prof. Ke Chen NB slides

Review: Naïve Bayes Classifier

$$\operatorname*{argmax}_{C} P(C \mid X) = \operatorname*{argmax}_{C} P(X, C) = \operatorname*{argmax}_{C} P(X \mid C) P(C)$$

Naïve Bayes Classifier

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

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

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

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

$$\underset{k}{\operatorname{argmax}} P(C\_k \mid X) = \underset{k}{\operatorname{argmax}} P(X,C) = \underset{k}{\operatorname{argmax}} P(X \mid C)P(C)$$

Generative Bayes Classifiers

Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Prob. models p(X|C) → EPE with 0-1 loss → MAP Rule → Many options → Prob. Models' Parameter

$$P(X_1, \cdots, X_p \mid C)$$

(EXTRA)

Gaussian Naive

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

Bernoulli Naïve

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

Multinomial

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

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 $\{c_1, \ldots, c_L\}$

- E.g.,
  - Spam filtering Task: Classify 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

→ $P(C=spam \mid D)$

Naive Bayes is Not So Naive

- Naive Bayes won 1st and 2nd 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.

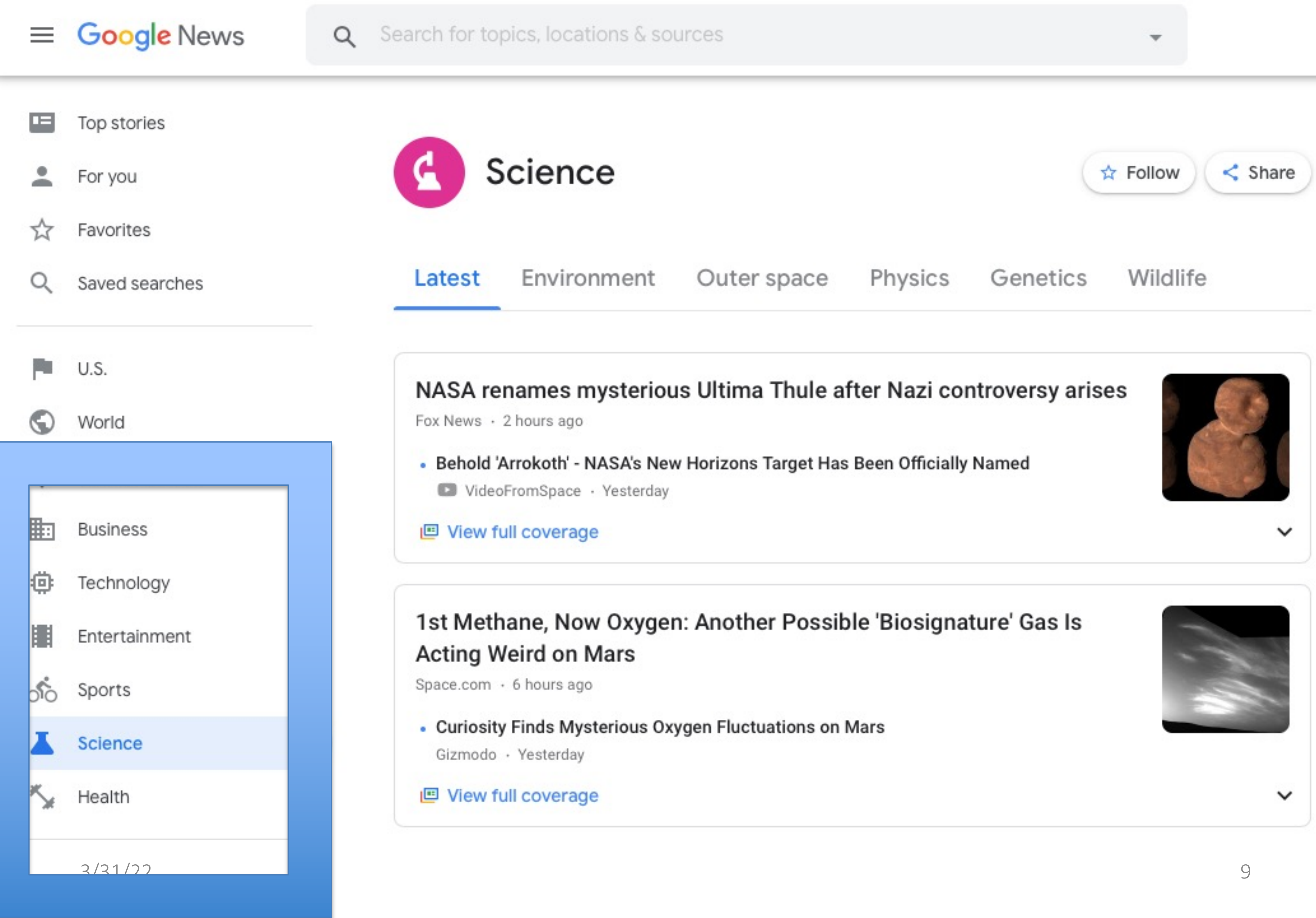
- A good dependable baseline for text classification (but not the best)!
- For most text categorization tasks, there are many relevant features and many irrelevant ones

Text classification Tasks

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

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 & sources
Top stories
For you
Favorites
Saved searches
U.S.
World
Business
Technology
Entertainment
Sports
Science
Health
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:

    $C = \{c_1, c_2, \ldots, c_J\}$

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(

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

)=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
...	.

① ②

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

love
loves
loving
→ love

Representing text:

a list of words → Dictionary

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

1 2 3

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

→ [NLTK]

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.

$$D = (W_1, W_2, \cdots, W_K)$$

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.

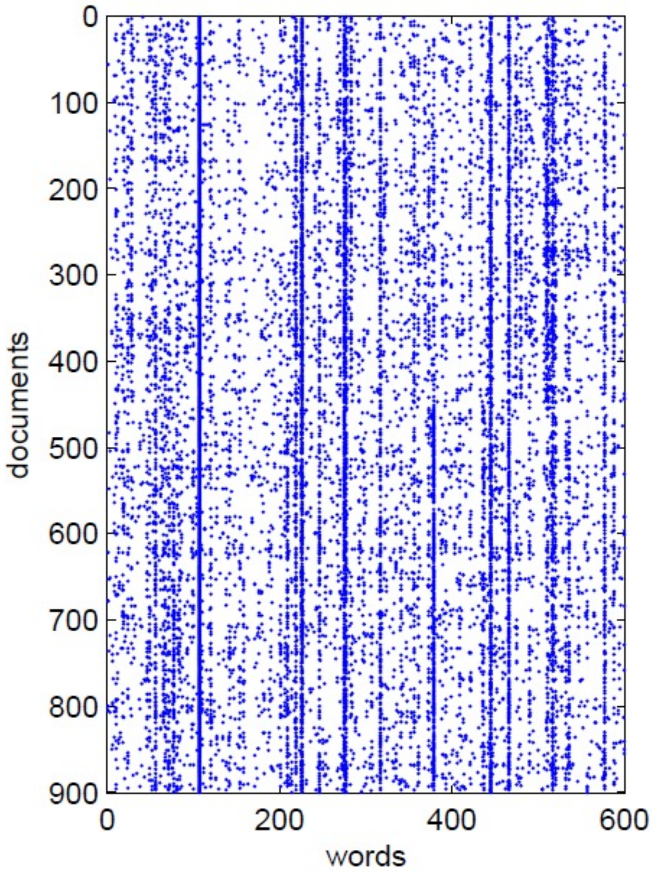
$D = (w_1, w_2, \cdots, w_K)$

Bag of words

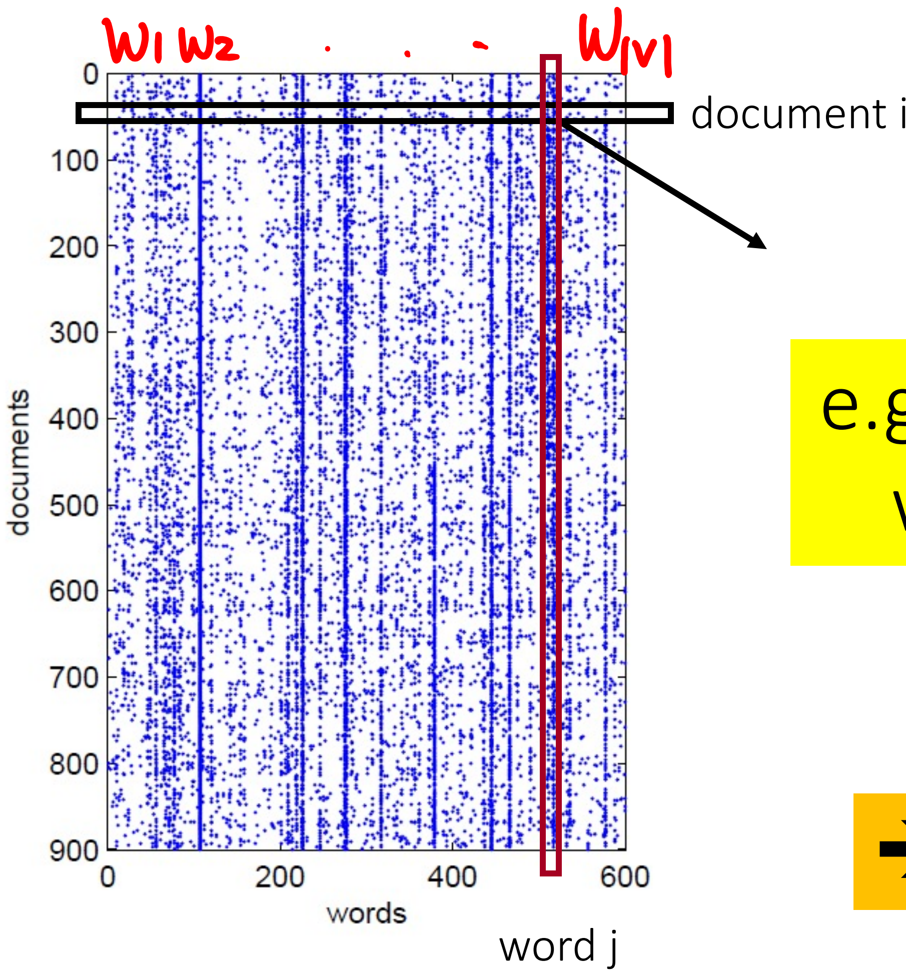
- What simplifying assumption are we taking?

We assumed word order is not important.

$D = (w_1, w_2, \cdots, w_K)$



Bag of words representation

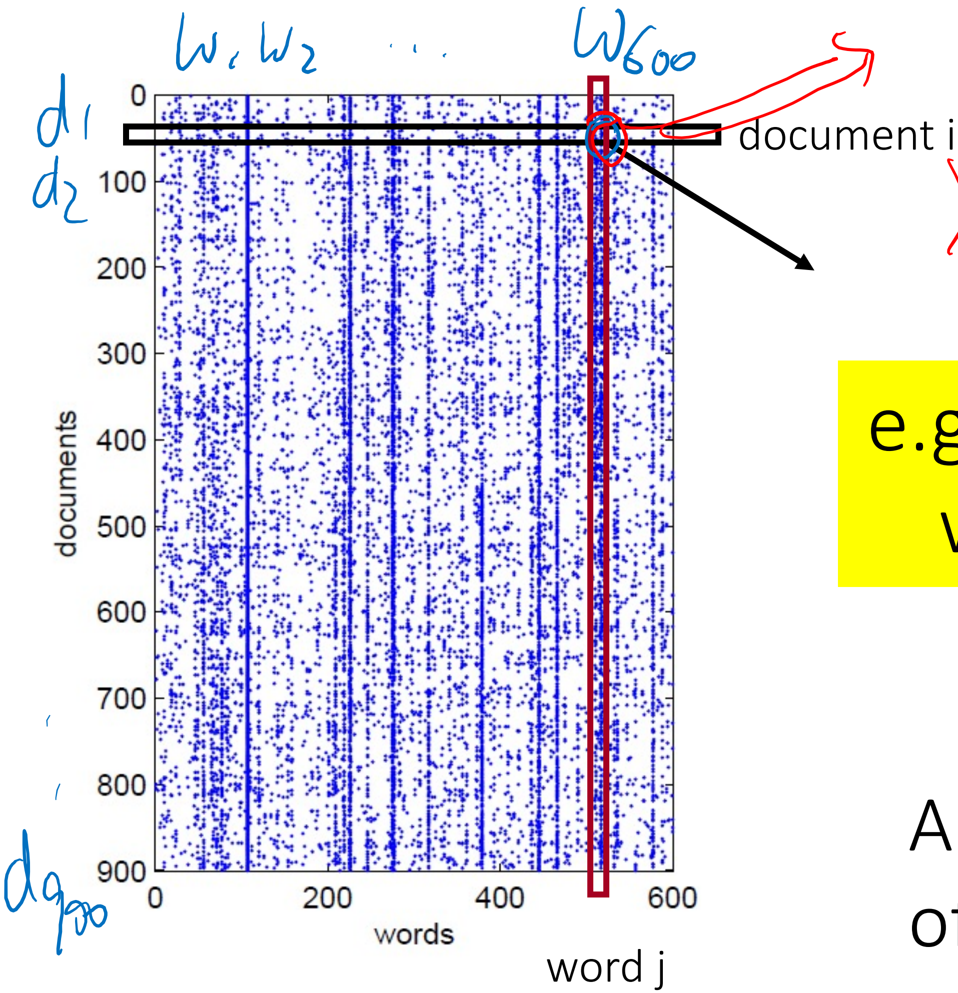


e.g., X (i,j) = Frequency of word j in document i

➔ Tabular

	X_1	X_2	X_3	C
S_1				
S_2				
S_3				
S_4				
S_5				
S_6				

Bag of words representation



$X(d_i, w_j)$

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				

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

Module II

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 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	.
...	.

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

?

$$c^* = argmaxP(D = d|C = c_i)P(C = c_i)$$

'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	.
...	.

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

$$D = (w_1, w_2, \cdots, w_K)$$

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

?

$$c^* = argmaxP(D = d|C = c_i)P(C = c_i)$$

'Bag of words' ➔ what probability model?

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

Two Previous models

$$\Pr(W_1 = true, W_2 = false \ldots, W_k = true \mid C = c)$$

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

$D = (W_1, W_2, \cdots, W_K)$

Naïve Probabilistic Models of text documents

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

Two Previous models

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

Multivariate Bernoulli Distribution

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

Multinomial Distribution

Text Classification with Naïve Bayes Classifier

- Multinomial vs Multivariate Bernoulli?

- Multinomial model is almost always more effective in text applications!

Adapt From Manning' textCat tutorial

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

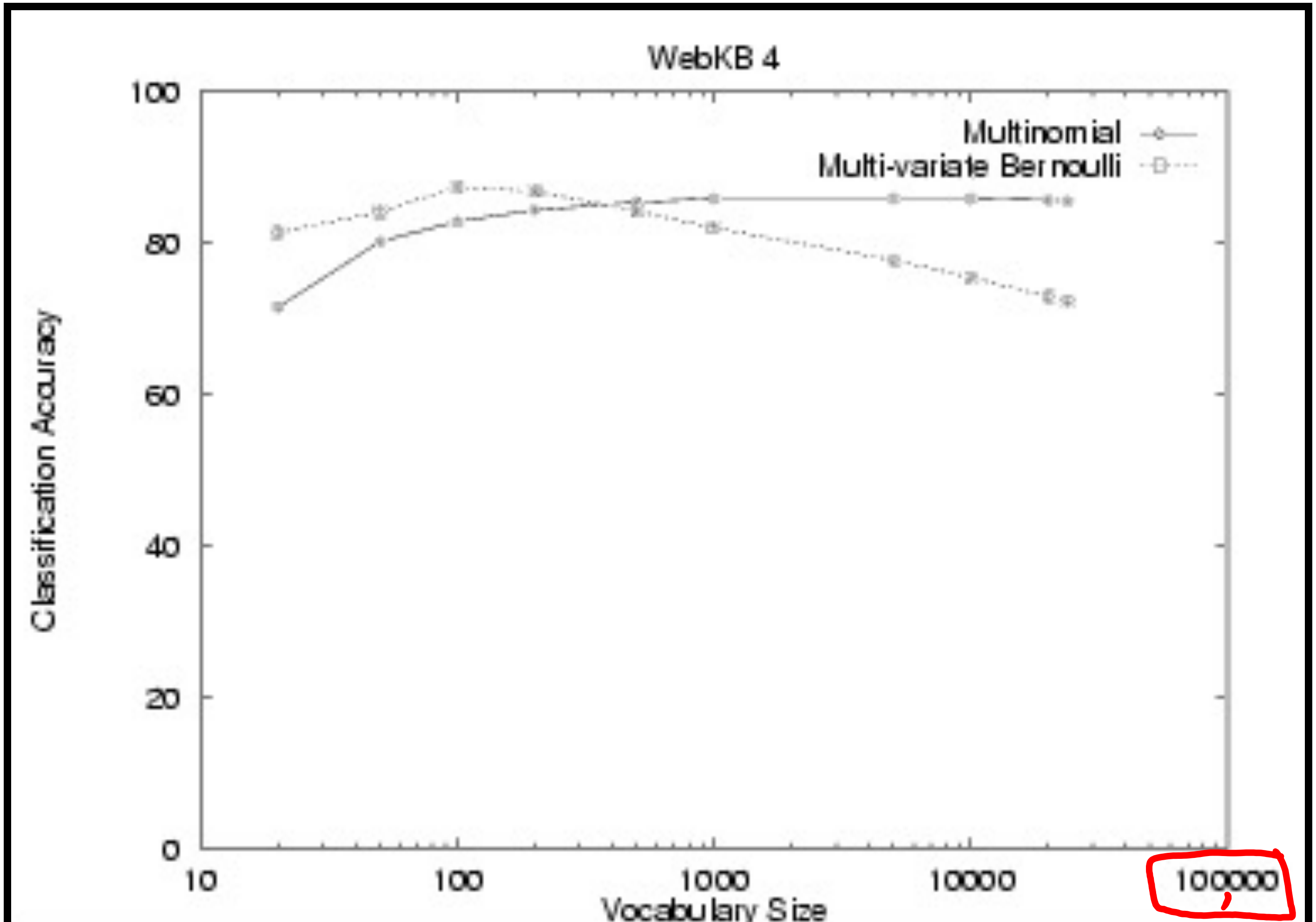
train

- Crawl and classify a new site (CMU)

test

Adapt From Manning' textCat tutorial

Multinomial vs. multivariate Bernoulli



n~5k

p~100k

I will code run the following notebook I adapted from scikit-learn https://colab.research.google.com/drive/1BKmJ4S4QKxIo9leJy0QokjdRSJuz_YXx?usp=sharing

```
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)
```

```
0.8348868175765646
```

```
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,   2],
       [  5,  35, 353,   3],
       [  5,  11,   4, 378]])
```

```
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 = 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 = 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

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

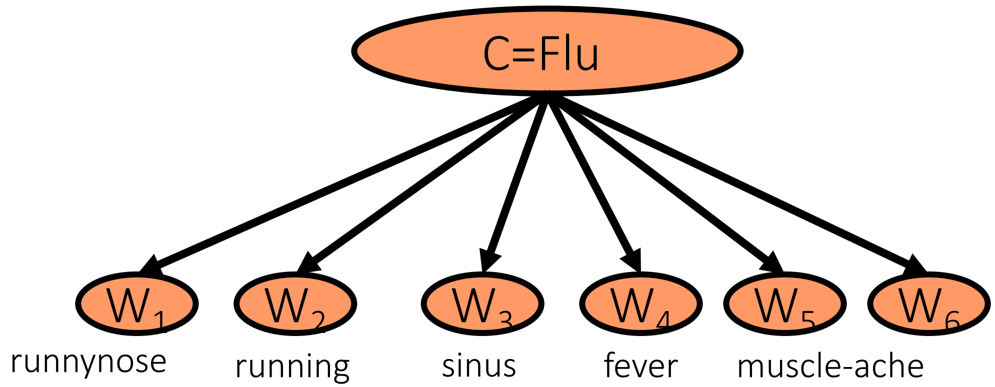
$$\Pr(W_1 = true, W_2 = false \ldots, W_k = true \mid C = c)$$

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



$p(w_i | c_j)$

$i = 1, \cdots, N$

$j = 1, \cdots, L$

this is naïve

$$\Pr(W_1 = true, W_2 = false, \ldots, W_k = true \mid C = c)$$
$$= P(W_1 = true \mid C) \bullet P(W_2 = false \mid C) \bullet \cdots \bullet P(W_k = true \mid C)$$

Review: Bernoulli Distribution e.g. Coin Flips

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

$$\Pr(W_i = true \mid C = c)$$

  - How many heads would you expect
  - Number of heads X: discrete random variable
  - Binomial distribution with parameters *k* and *p*

estimated from data

$$
\begin{array}{c|ccccc|}
 & c_1 & c_2 & \cdots & & c_L \\
\hline
w_1 & & & & & \\
w_2 & & & & & \\
\vdots & & & P_{w_k, c_j} & & \\
w_K & & & & & \\
\hline
\end{array}
$$

$\underbrace{}_{K = |V|}$

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. $\begin{cases} H & H & T & H & H & T & H & T & \cdots & H \\ X_1 & X_2 & X_3 & X_4 & \cdots & & & & & X_n \end{cases}$

Review: Calculating Likelihood

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

$\Downarrow$

$\{H, H, T, \ldots H\}$

$\Downarrow$ reformulate

$\{1, 1, 0, \ldots, 1\}$

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

Review: Defining Likelihood for Bernoulli

- Likelihood = p(data | 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: $f(x_i \mid p) = p^{x_i}(1-p)^{1-x_i}$

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

$\Rightarrow p^{\Sigma x_i} (1-p)^{\Sigma(1-x_i)}$

LIKELIHOOD:

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

function of p

Review: Deriving the Maximum Likelihood Estimate for Bernoulli

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

Minimize the negative log-likelihood

➔ MLE parameter estimation

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

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

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

i.e. Relative frequency of a binary event

Review: Deriving the Maximum Likelihood Estimate for Bernoulli

$$\arg\min_p \{-l(p)\} = \arg\min_p \{-x\log(p)-(n-x)\log(1-p)\}$$

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

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

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

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

$$0 = -x + pn$$

Minimize the negative log-likelihood

→ MLE parameter estimation

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

i.e. Relative frequency of a binary event

Parameter estimation

- Multivariate Bernoulli model:

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

- Smoothing to Avoid Overfitting

Adapt From Manning' textCat tutorial

Testing Stage: (Look Up Operations)

$$d_{ts} = \{w_1 = \text{true},\ w_2 = \text{false},\ w_3 = \text{true}\}$$

$$p(d_{ts} | c_j) = p_{w1,j}(1 - p_{w2,j})\, p_{w3,j}$$

$$p(c_j)$$

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.

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

- Note that model is now just max of sum of weights…

Adapt From Manning' textCat tutorial

Thank You

Thank you

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
...	.

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

Can be represented as a multinomial distribution.

Model 2: Multinomial Naïve Bayes - 'Bag of words' representation of text

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

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

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

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

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)

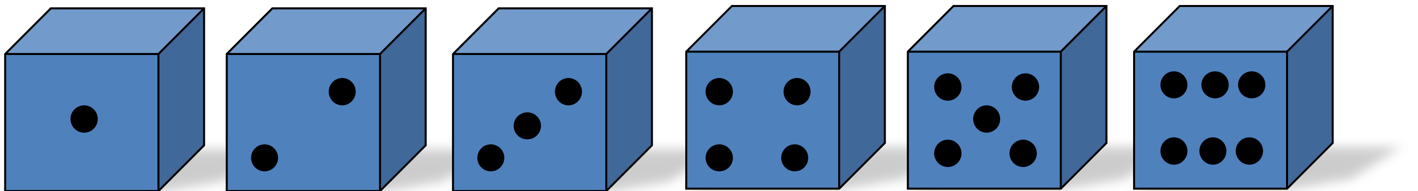
Multinomial distribution

- 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)

$$\theta_1 = p$$
$$\theta_2 = 1 - \theta_1$$

$$\theta_1 .. \theta_k$$



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)

- $\{H\ H\ T\ H\ \ldots\ H\}_N$

$X = Num_N(H)$

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

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 ⇒

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

$p_{head} = p$

$p_{tail} = 1 - p$

A binomial distribution is the multinomial distribution with k=2 and

$$\theta_1 = p$$

$$\theta_2 = 1 - \theta_1$$

Multinomial distribution

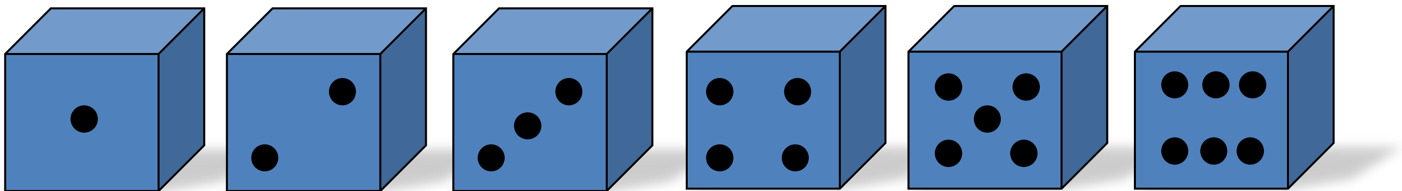
- The multinomial distribution is a generalization of the binomial distribution.

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

① $N_1+N_2+\cdots+N_k=N$

② $\theta_1+\theta_2\cdots+\theta_k=1$



Multinomial Distribution for Text Classification

- $W_1, W_2, ..W_k$ are variables

Number of possible orderings of N balls

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

Multinomial Distribution for Text Classification

- $W_1, W_2, ..W_k$ are variables

Number of possible orderings of N balls

$$P(W_1 = n_1, \ldots, W_k = n_k \mid c, N, \theta_{1,c}, \ldots, \theta_{k,c}) = \cancel{\frac{N!}{n_1! n_2! \ldots n_k!}} \underline{\theta_{1,c}}^{n_1} \underline{\theta_{2,c}}^{n_2} \ldots \underline{\theta_{k,c}}^{n_k}$$

$$H = \{N, \theta_1, \theta_2, \ldots, \theta_k \mid C\}$$

$N \rightarrow d_i$

$$\underset{C}{\operatorname{argmax}}\; p(W_1 = n_1, \ldots, W_k = n_k \mid C)\, p(C)$$

Multinomial Distribution for Text Classification

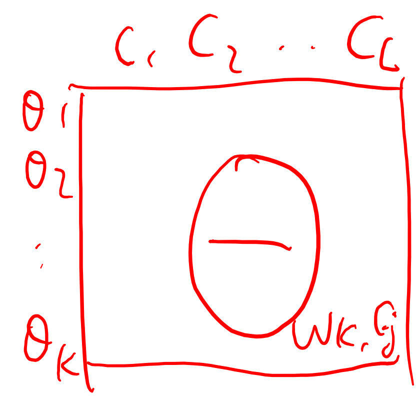
- $W_1, W_2, ..W_k$ are variables

Number of possible orderings of N balls

$$P(W_1 = n_1, \ldots, W_k = n_k \mid c, N, \theta_{1,c}, .., \theta_{k,c}) = \cancel{\frac{N!}{n_1! n_2! .. n_k!}} \theta_{1,c}^{\ n_1} \theta_{2,c}^{\ n_2} .. \theta_{k,c}^{\ n_k}$$

Label invariant

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



Multinomial : Training Algorithm (parameter estimation with MLE)

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

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

Training: Parameter estimation

Multinomial model:

θ_{w_i, c_j}

$$\hat{P}(X_i = w_i \mid c_j) =$$

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

Adapt From Manning' textCat tutorial

Model 2: Multinomial Naïve Bayes - 'Bag of words' – TESTING Stage

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

$$\arg\max_{c} P(W_1 = n_1, \ldots, W_k = n_k, c)$$

$$= \arg\max_{c} \{ p(c) * \theta_{1,c}{}^{n_1} \theta_{2,c}{}^{n_2} .. \theta_{k,c}{}^{n_k} \}$$

EXTRA

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

$$
\begin{array}{c|ccccc|}
 & c_1 & c_2 & \cdots & & c_L \\
\hline
\theta_1 \to w_1 & & & & & \\
\theta_2 \to w_2 & & & & & \\
\vdots & & & \Theta_{w_i, c_j} & & \\
\theta_K \to w_K & & & & & \\
\hline
\end{array}
$$

Deriving the Maximum Likelihood Estimate for multinomial distribution

LIKELIHOOD:

$$\arg\max_{\theta_1,..,\theta_k} P(d_1,\ldots,d_T \mid \theta_1,..,\theta_k)$$

$w_1 = n_{1,t},\ w_2 = n_{2,t},\ \cdot\ ;\ w_k = n_{k,t}$

function of θ

θ vector

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

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

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

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

Deriving the Maximum Likelihood Estimate for multinomial distribution

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

Constrained optimization

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

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

Deriving the Maximum Likelihood Estimate for multinomial distribution

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

Constrained optimization

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

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

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

Deriving the Maximum Likelihood Estimate for multinomial distribution

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

Constrained optimization

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

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

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

Constrained optimization MLE estimator

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

Deriving the Maximum Likelihood Estimate for multinomial distribution

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

Constrained optimization

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

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

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

Constrained optimization MLE estimator

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

➔ 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:

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

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

$$\begin{array}{c|cccc} & c_1 & c_2 & \cdots & c_L \\ \hline w_1 & & & & \\ w_2 & & & & \\ \vdots & & \Theta_{w_i,c_j} & & \\ w_K & & & & \end{array}$$

Parameter estimation

Multinomial model:

$$\theta_{w_i, c_j}$$

$$\hat{P}(X_i = w_i \mid c_j) =$$

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

Adapt From Manning' textCat tutorial

Multinomial : Learning Algorithm for parameter estimation with MLE

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

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

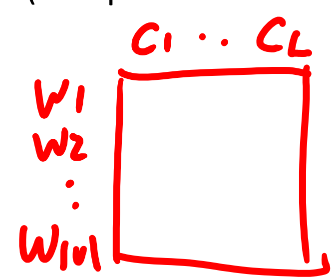
- $Text_j \leftarrow$ is length n_j and is a single document containing all $docs_j$ for class c_j
- for each word w_k in Vocabulary
  - $n_{k,j} \leftarrow$ number of occurrences of w_k in $Text_j$; n_j is length of $Text_j$
  - $$P(w_k \mid c_j) \leftarrow \frac{n_{k,j} + \alpha}{n_j + \alpha |Vocabulary|}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

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

Multinomial Bayes: Time Complexity

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

- **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

Adapt From Manning' textCat tutorial

Recap: Multinomial Naïve Bayes - 'Bag of words' – TESTING Stage

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

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

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

$\{W_i, W_j, \ldots W_{Ld}\}$ ⇒

very very good

① BOW

② $p(c)\,p(W_{p_1})\,p(W_{p_2}) \ldots p(W_{p_{Ld}})$

$p(\text{very})\,p(\text{very})\,p(\text{good})$

Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(x_1 = \text{"the"} \mid c_j) \cdots P(x_n = \text{"the"} \mid c_j)$$

- 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 !!!

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

Equal to,
(without the coefficient)

$$\Pr(W_1 = n_1, \ldots, W_k = n_k \mid C = c_j)$$

Adapt From Manning' textCat tutorial

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

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
C1	0.2	0.01	0.0001	0.0001	0.0005
C2	0.2	0.0001	0.02	0.1	0.01

$$\underset{k}{\operatorname{argmax}} P(C\_k \mid X) = \underset{k}{\operatorname{argmax}} P(X, C) = \underset{k}{\operatorname{argmax}} P(X \mid C) P(C)$$

Generative Bayes Classifier

Task → Representation → Score Function → Search/Optimization → Models, Parameters

classification → Prob. models p(X|C) → EPE with 0-1 loss → MAP Rule → Many options → Prob. Models' Parameter

$$P(X_1, \cdots, X_p \mid C)$$

Gaussian Naive

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

Bernoulli Naïve

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

Multinomial

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

Thank You

UVA CS 4774 : Machine Learning

Lecture 17: Naïve Bayes Classifier for Text Classification

Module IV Extra

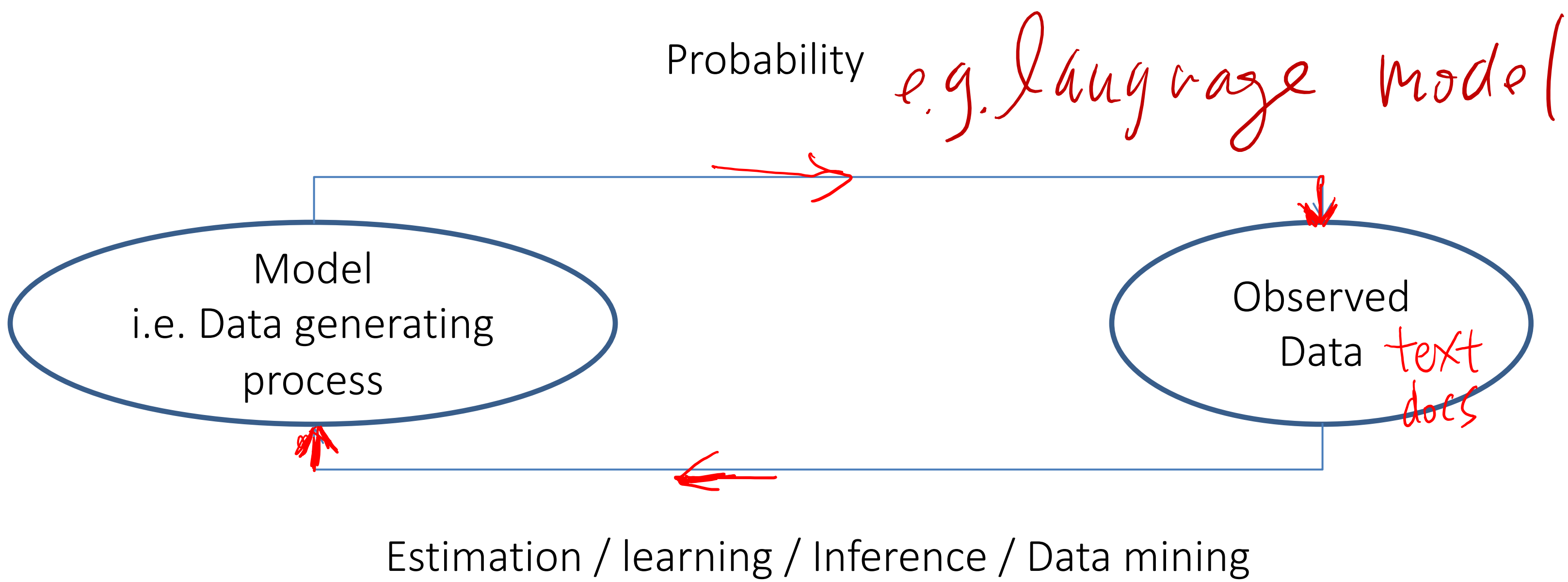
Dr. Yanjun Qi

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



e.g. MLE

But how to specify a model?

Build a generative model that approximates how data is produced.

GBC Models	$\lvert x_i \rvert = K$ $1, \ldots, p$	$p(c_j)$ $j = 1, \ldots, L$	$p(X_1 X_2 X \ldots X_p \mid c_j)$ #
GBC discrete	$\lvert x_i \rvert = K$	# $O(L)$	$K^p \times L$
NBC discrete naive	$\lvert x_i \rvert = K$	$O(L)$	$Kp \times L$
Naive Gaussian	$N(\mu_i, \Lambda_i)$ μ_i: $p \times 1$, Λ_i: $p \times p$	$O(L)$	$2p \times L$
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	$\theta_1, \ldots, \theta_k \mid C$	$O(L)$	$\lvert V \rvert \times L$

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
...	...

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

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)

WHY is this naïve ???

multinomial coefficient, normally can leave out in practical calculations.

why naive ???

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

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 ∑).
  - 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
…	

Σ_α

θ_1

θ_2

θ_3

θ_K

Adapt From Manning' textCat tutorial

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 ∑).
  - E.g., unigram model

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

Model C_1

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

the	boy	likes	the	dog
0.2	0.01	0.02	0.2	0.01

Multiply all five terms

P(d | C_1) = 0.00000008

Adapt From Manning' textCat tutorial

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) → [from training]

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

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

[testing]

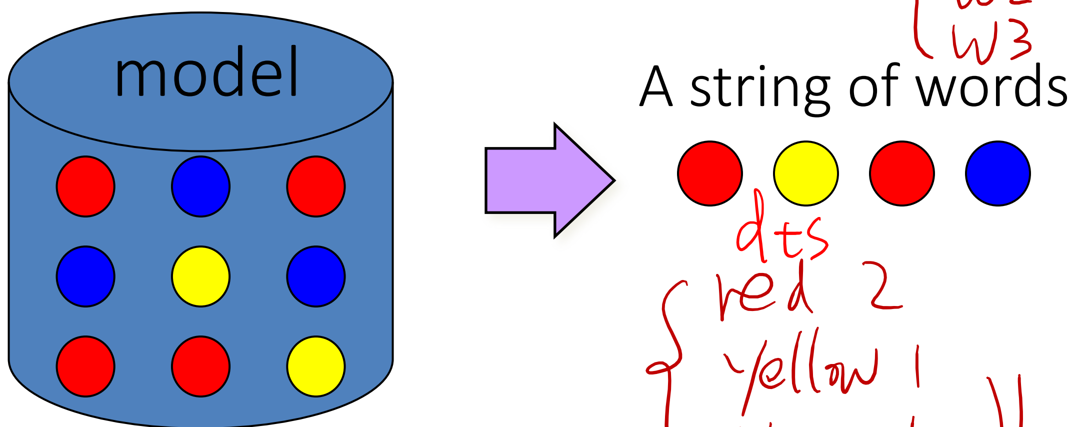
→ d is more likely to be from class C2

A Physical Metaphor

- Colored balls are randomly drawn from (with replacement)

$K=3$

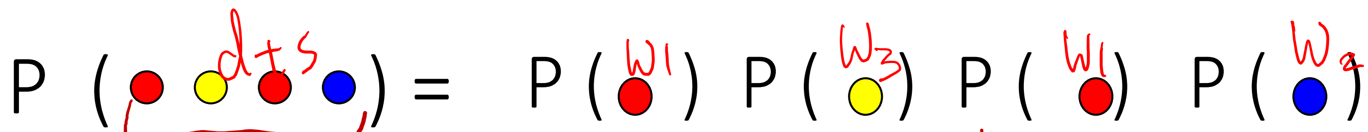
$\begin{cases} w_1 & \text{red} & \theta_1 \\ w_2 & \text{blue} & \theta_2 \\ w_3 & \text{yellow} & \theta_3 \end{cases}$



dts

$\begin{cases} \text{red} & 2 \\ \text{yellow} & 1 \\ \text{blue} & 1 \end{cases}$

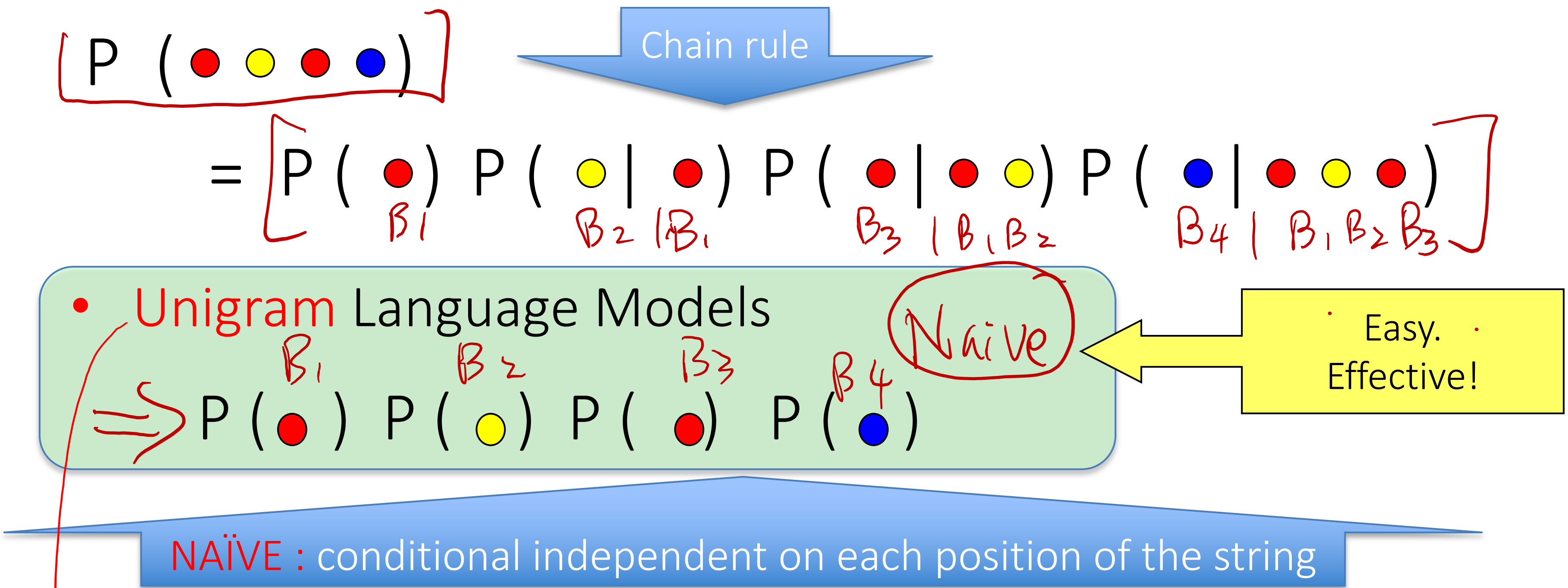
$\left.\begin{matrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{matrix}\right\}$ model paras



$= \theta_1^2 \, \theta_2^1 \, \theta_3^1$ [multinomial Distri]

Unigram language model ➔ More general:

Generating language string from a probabilistic model



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

Adapt From Manning' textCat tutorial

Unigram language model ➔ More general:

Generating language string from a probabilistic model

$\boxed{P(\bullet \bullet \bullet \bullet)}$

Chain rule

$= \Big[\underset{B_1}{P(\bullet)} \; \underset{B_2 \mid B_1}{P(\bullet \mid \bullet)} \; \underset{B_3 \mid B_1 B_2}{P(\bullet \mid \bullet \bullet)} \; \underset{B_4 \mid B_1 B_2 B_3}{P(\bullet \mid \bullet \bullet \bullet)} \Big]$

- Unigram Language Models

Naive

$\Rightarrow \underset{B_1}{P(\bullet)} \; \underset{B_2}{P(\bullet)} \; \underset{B_3}{P(\bullet)} \; \underset{B_4}{P(\bullet)}$

Easy.
Effective!

NAÏVE : conditional independent on each position of the string

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

$\underset{B_1}{P(\bullet)} \; \underset{B_2 \mid B_1}{P(\bullet \mid \bullet)} \; \underset{B_3 \mid B_2}{P(\bullet \mid \bullet)} \; \underset{B_4 \mid B_3}{P(\bullet \mid \bullet)}$

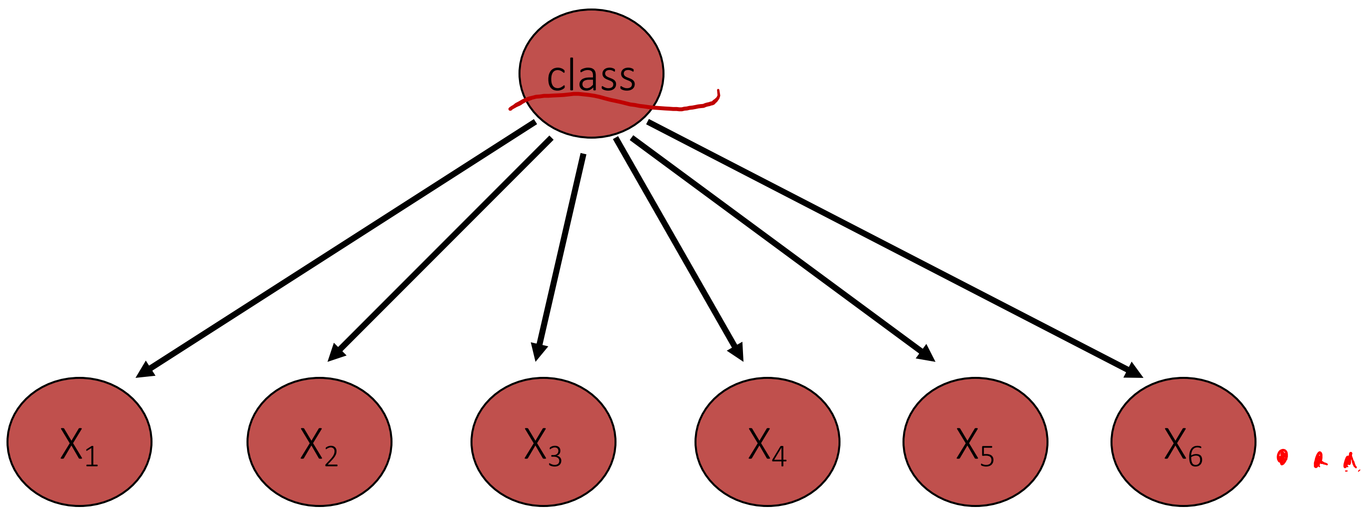
$B_j \mid B_{j-1}$

univ. of | univ. virg. | of

Adapt From Manning' textCat tutorial

Multinomial Naïve Bayes = a class conditional unigram language model

Classifier



- 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

Adapt From Manning' textCat tutorial

	the	boy	likes	the	dog
C_j	0.2	0.01	0.02	0.2	0.01
position	1	2	3	4	5

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_K\end{bmatrix}_1 \quad \begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_K\end{bmatrix}_2 \quad \cdots \quad \Rightarrow$$

approximate with the same vector across all positions

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_K\end{bmatrix}$$

Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- Attributes are text positions, values are words.

$\Rightarrow \operatorname{argmax} P(C_j \mid X)$

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j)$$

the boy like the dog

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \underbrace{P(x_1 = \text{"the"} \mid c_j)}_{?} \cdots \underbrace{P(x_n = \text{"the"} \mid c_j)}_{?}$$

- 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 ← 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 !!!

$P(w_k|c_j)$ at position i

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

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)

Adapt From Manning' textCat tutorial

Multinomial Naïve Bayes: Classifying Step

- Positions ← 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 !!!

$P(w_k|c_j)$

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}}\, P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Equal to, (leaving out of multinomial coefficient)

$$\Pr(W_1 = n_1, \ldots, W_k = n_k \mid C = c_j)$$

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)

Adapt From Manning' textCat tutorial

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' textCat tutorial