

Natural Language Processing (AI615)

Prepared by Dr. Ashraf Elsayed

Lecture 5



Text Classification

Lecture 5

Prepared by
Dr. Ashraf Elsayed

The objectives of this lecture is to:

- Understand Text Classification
- Define and explain Text Classification and Naïve Bayes
- Understand the bag of words representation
- Understand Multinomial Naïve Bayes Classifier

Text Classification

- Assigning subject categories, topics, or genres
- Spam detection
- Authorship identification
- Age/gender identification
- Language Identification
- Sentiment analysis
- ...

Text Classification: definition

- *Input:*
 - a document d
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
- *Output:* a predicted class $c \in C$

Classification Methods: Hand-coded rules

- Rules based on combinations of words or other features
 - spam: black-list-address OR (“dollars” AND “have been selected”)
- Accuracy can be high
 - If rules carefully refined by expert
- But building and maintaining these rules is expensive

Classification Methods: Supervised Machine Learning

- *Input:*
 - a document d
 - a fixed set of classes $C = \{c_1, c_2, \dots, c_J\}$
 - A training set of m hand-labeled documents $(d_1, c_1), \dots, (d_m, c_m)$
- *Output:*
 - a learned classifier $\gamma: d \rightarrow c$

Classification Methods: Supervised Machine Learning

- Any kind of classifier
 - Naïve Bayes
 - Logistic regression
 - Support-vector machines
 - k-Nearest Neighbors
- ...

Text Classification and Naïve Bayes

Naïve Bayes (I)

Naïve Bayes Intuition

- Simple (“naïve”) classification method based on Bayes rule
- Relies on very simple representation of document
 - Bag of words

The bag of words representation

Y
(

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

Y
(

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: using a subset of words

Y
(

```
x love xxxxxxxxxxxxxxxxxxxx sweet
xxxxxxxx satirical xxxxxxxxxxxx
xxxxxxxxxxxx great xxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxx fun xxxx
xxxxxxxxxxxxxxxx whimsical xxxx
romantic xxxx laughing
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxx recommend xxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx several xxxxxxxxxxxxxxxxxxxx
xxxxx happy xxxxxxxxxx again
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

) = C



The bag of words representation

Y
(

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

) = C



Bag of words for document classification

Test document

parser
language
label
translation
...

Machine Learning

learning
training
algorithm
shrinkage
network...

NLP

parser
tag
training
translation
language...

Garbage Collection

garbage
collection
memory
optimization
region...

Planning

planning
temporal
reasoning
plan
language...

GUI

...

?

Text Classification and Naïve Bayes

Formalizing the Naïve Bayes Classifier

Bayes' Rule Applied to Documents and Classes

- For a document d and a class c

$$P(c | d) = \frac{P(d | c)P(c)}{P(d)}$$

Naïve Bayes Classifier (I)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(c | d)$$

MAP is “maximum a posteriori” = most likely class

$$= \operatorname{argmax}_{c \in C} \frac{P(d | c)P(c)}{P(d)}$$

Bayes Rule

$$= \operatorname{argmax}_{c \in C} P(d | c)P(c)$$

Dropping the denominator

Likelihood

Prior probability

Naïve Bayes Classifier (II)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(d \mid c)P(c)$$

$$= \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n \mid c)P(c)$$

Document d
represented as
features x1..xn

Naïve Bayes Classifier (III)

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n \mid c)P(c)$$

$O(|X|^n \cdot |C|)$ parameters

Could only be estimated if a very, very large number of training examples was available.

How often does this class occur?

We can just count the relative frequencies in a corpus

Multinomial Naïve Bayes Independence Assumptions

$$P(x_1, x_2, \dots, x_n \mid c)$$

- **Bag of Words assumption:** Assume position doesn't matter
- **Conditional Independence:** Assume the feature probabilities $P(x_i \mid c_j)$ are independent given the class c .

$$P(x_1, \dots, x_n \mid c) = P(x_1 \mid c) \square P(x_2 \mid c) \square P(x_3 \mid c) \square \dots \square P(x_n \mid c)$$

Multinomial Naïve Bayes Classifier

$$c_{MAP} = \operatorname{argmax}_{c \in C} P(x_1, x_2, \dots, x_n \mid c) P(c)$$

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{x \in X} P(x \mid c)$$

Applying Multinomial Naive Bayes Classifiers to Text Classification

positions $\leftarrow$ all word positions in test document

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

Problems with multiplying lots of probs

- There's a problem with this:

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

- Multiplying lots of probabilities can result in floating-point underflow!
- $.0006 * .0007 * .0009 * .01 * .5 * .000008....$
- Idea: Use logs, because $\log(ab) = \log(a) + \log(b)$
- We'll sum logs of probabilities instead of multiplying probabilities!

We actually do everything in log space

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

This: $c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \left[\log P(c_j) + \sum_{i \in \text{positions}} \log P(x_i | c_j) \right]$

Notes:

1) Taking log doesn't change the ranking of classes!

The class with highest probability also has highest log probability!

2) It's a linear model:

Just a max of a sum of weights: a **linear** function of the inputs

So naive bayes is a **linear classifier**

Text Classification and Naïve Bayes

Naïve Bayes: Learning

Learning the Multinomial Naïve Bayes Model

- First attempt: maximum likelihood estimates
 - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{\text{doccount}(C = c_j)}{N_{doc}}$$

$$\hat{P}(w_i | c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

Parameter estimation

$$\hat{P}(w_i | c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

fraction of times word w_i appears
among all words in documents of topic c_j

- Create mega-document for topic j by concatenating all docs in this topic
 - Use frequency of w in mega-document

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive** (***thumbs-up***)?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{\text{count}(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} \text{count}(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$c_{MAP} = \operatorname{argmax}_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing for Naïve Bayes

$$\hat{P}(w_i | c) = \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} (\text{count}(w, c) + 1)}$$

$$= \frac{\text{count}(w_i, c) + 1}{\sum_{w \in V} \text{count}(w, c) + |V|}$$

Multinomial Naïve Bayes: Learning

- From training corpus, extract *Vocabulary*
- Calculate $P(c_j)$ terms
 - For each c_j in C do
 - $docs_j \leftarrow$ all docs with class $= c_j$
- Calculate $P(w_k | c_j)$ terms
 - $Text_j \leftarrow$ single doc containing all $docs_j$
 - For each word w_k in *Vocabulary*
 - $n_k \leftarrow$ # of occurrences of w_k in $Text_j$

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

$$P(w_k | c_j) \propto \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

Number of
words in $Text_j$

Unknown words

- What about unknown words
 - that appear in our test data
 - but not in our training data or vocabulary?
- We **ignore** them
 - Remove them from the test document!
 - Pretend they weren't there!
 - Don't include any probability for them at all!

Stop words

- Some systems ignore stop words
 - **Stop words:** very frequent words like *the* and *a*.
 - Sort the vocabulary by word frequency in training set
 - Call the top 10 or 50 words the **stopword list**.
 - Remove all stop words from both training and test sets
 - As if they were never there!
- But removing stop words doesn't usually help
 - So in practice most NB algorithms use **all** words and **don't** use stopwords lists

Text Classification and Naïve Bayes

Multinomial Naïve Bayes: A Worked Example

$$\hat{P}(c) = \frac{N_c}{N}$$

$$\hat{P}(w | c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|}$$

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Priors:

$$P(c) = \frac{3}{4}$$

$$P(j) = \frac{1}{4}$$

Conditional Probabilities:

$$P(\text{Chinese} | c) = (5+1) / (8+6) = 6/14 = 3/7$$

$$P(\text{Tokyo} | c) = (0+1) / (8+6) = 1/14$$

$$P(\text{Japan} | c) = (0+1) / (8+6) = 1/14$$

$$P(\text{Chinese} | j) = (1+1) / (3+6) = 2/9$$

$$P(\text{Tokyo} | j) = (1+1) / (3+6) = 2/9$$

$$P(\text{Japan} | j) = (1+1) / (3+6) = 2/9$$

Choosing a class:

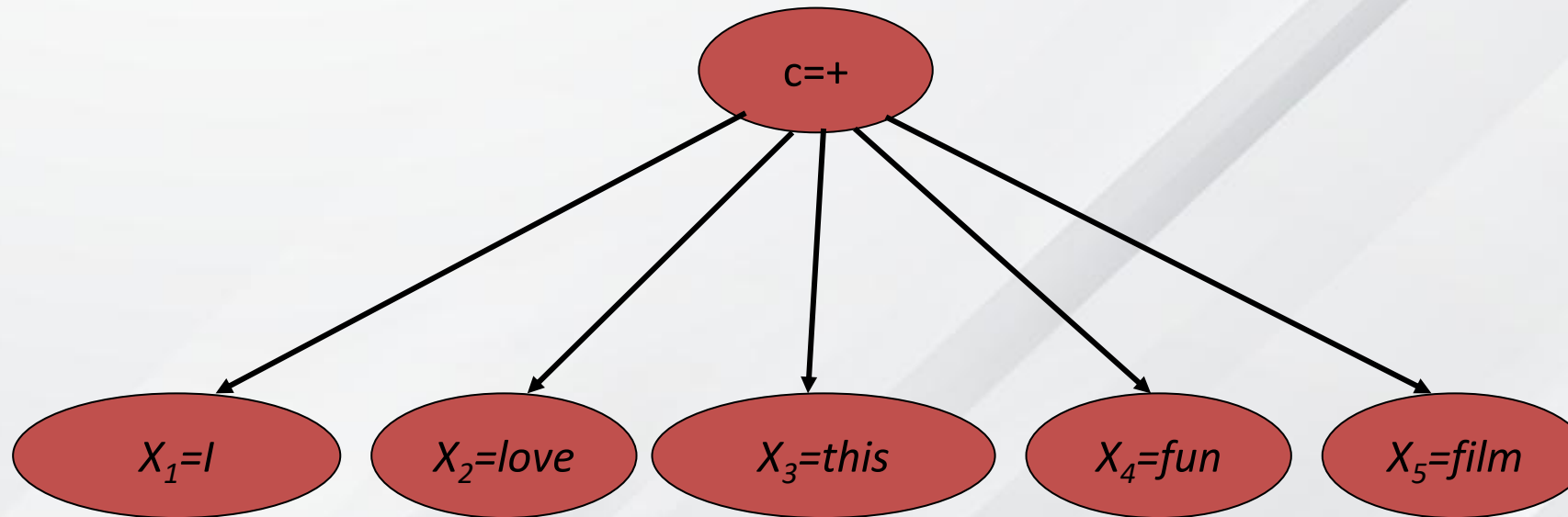
$$P(c | d5) \propto 3/4 * (3/7)^3 * 1/14 * 1/14 \\ \approx 0.0003$$

$$P(j | d5) \propto 1/4 * (2/9)^3 * 2/9 * 2/9 \\ \approx 0.0001$$

Text Classification and Naïve Bayes

Naïve Bayes: Relationship to Language
Modeling

Generative Model for Multinomial Naïve Bayes



Naïve Bayes and Language Modeling

- Naïve bayes classifiers can use any sort of feature
 - URL, email address, dictionaries, network features
- But if, as in the previous slides
 - We use **only** word features
 - we use **all** of the words in the text (not a subset)
- Then
 - Naïve bayes has an important similarity to language modeling.

Each class = a unigram language model

- Assigning each word: $P(\text{word} \mid c)$
- Assigning each sentence: $P(s \mid c) = \prod P(\text{word} \mid c)$

Class *pos*

0.1	I	<u>I</u>	<u>love</u>	<u>this</u>	<u>fun</u>	<u>film</u>
0.1	love	0.1	0.1	0.01	.05	0.1
0.01	this					
0.05	fun					
0.1	film					
...						

$$P(s \mid \text{pos}) = 0.00000005$$

Naïve Bayes as a Language Model

- Which class assigns the higher probability to s?

Model pos		Model neg	
0.1	I	0.2	I
0.1	love	0.001	love
0.01	this	0.01	this
0.05	fun	0.005	fun
0.1	film	0.1	film

I	love	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

$$P(s|\text{pos}) > P(s|\text{neg})$$

Example

Consider the following three sentences:

1. "I love cats."
2. "Dogs are adorable."
3. "Cats and dogs are good companions."

To create the BoW representation, we first construct a vocabulary consisting of all unique words across the documents: "I," "love," "cats," "dogs," "are," "adorable," "and," "good," "companions."

Next, we create a feature vector for each document, where each element represents the count or presence of a word in the vocabulary. For instance, the BoW representation of the first sentence would be $[1, 1, 1, 0, 0, 0, 0, 0, 0]$ since it contains “I,” “love,” and “cats,” but not the other words.

Applying Naive Bayes on Bag of Words

To classify a new document using the Naive Bayes approach, we calculate the posterior probability for each class and assign the document to the class with the highest probability. To compute the likelihood of a document given a class, we make the assumption of feature independence, which simplifies the calculation.

Let's illustrate this with a practical example of classifying movie reviews as positive or negative. We have a dataset of labeled movie reviews, and we want to build a classifier using the Naive Bayes / Bag of Words approach.

Example

Suppose we have the following training data:

Review	Class
"The movie was amazing!"	Positive
"I didn't like the film."	Negative
"Great acting and an engaging plot."	Positive
"Terrible movie, I would not recommend it to anyone."	Negative

Review	Class
"The movie was amazing!"	Positive
"I didn't like the film."	Negative
"Great acting and an engaging plot."	Positive
"Terrible movie, I would not recommend it to anyone."	Negative

We preprocess the text by tokenizing it into words and creating a vocabulary: ["the," "movie," "was," "amazing," "I," "didn't," "like," "film," "great," "acting," "and," "an," "engaging," "plot," "terrible," "would," "not," "recommend," "it," "to," "anyone"].

Next, we construct the BoW representation for each review:

Review	BoW Representation	Class
"The movie was amazing!"	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	Positive
"I didn't like the film."	[0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	Negative
"Great acting and an engaging plot."	[0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]	Positive
"Terrible movie, I would not recommend it to anyone."	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1]	Negative

Now, let's assume we have a new movie review: "The acting was superb!" We can convert this review into its BoW representation: [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0].

To classify this review, we calculate the posterior probability for each class using Naive Bayes. We estimate the class prior probabilities based on the training data and calculate the likelihood of the BoW representation given each class. Finally, we normalize the probabilities to obtain the posterior probabilities.

Let's assume we obtain the following probabilities:

Class	Prior Probability	Likelihood	Posterior Probability
Positive	0.5	0.004	0.002
Negative	0.5	0.00000000000002	0.00000000000001

Based on these calculations, we can conclude that the new review is more likely to belong to the positive class. Thus, we classify it as a positive review.

Example

	Document ID	Keywords in the document	Class h
Training Set	1	Love Happy Joy Joy Happy	Yes
	2	Happy Love Kick Joy Happy	Yes
	3	Love Move Joy Good	Yes
	4	Love Happy Joy Love Pain	Yes
	5	Joy Love Pain Kick Pain	No
	6	Pain Pain Love kick	No
Testing Set	7	Love Pain Joy Love Kick	?

- The prior probabilities of a document being classified using the six documents are,

$$P(h) = \frac{4}{6} = \frac{2}{3}$$

and

$$p(-h) = \frac{2}{6} = \frac{1}{3}$$

- That is there is $\frac{2}{3}$ prior probability that a document will be classified as h and $\frac{1}{3}$ probability of not h.

Testing Example:

Love Pain Joy Love Kick = ?

Class h	Class $-h$
$P(\text{Love} \mid h) = 5/19$	$P(\text{Love} \mid -h) = 2/9$
$P(\text{Pain} \mid h) = 1/19$	$P(\text{Pain} \mid -h) = 4/9$
$P(\text{Joy} \mid h) = 5/19$	$P(\text{Joy} \mid -h) = 1/9$
$P(\text{Kick} \mid h) = 1/19$	$P(\text{Kick} \mid -h) = 2/9$

Testing Example:

Love Pain Joy Love Kick = ?

Class h	Class $-h$
$P(\text{Love} h) = 5/19$	$P(\text{Love} -h) = 2/9$
$P(\text{Pain} h) = 1/19$	$P(\text{Pain} -h) = 4/9$
$P(\text{Joy} h) = 5/19$	$P(\text{Joy} -h) = 1/9$
$P(\text{Kick} h) = 1/19$	$P(\text{Kick} -h) = 2/9$

$$\begin{aligned}
 P(h|d_7) &= P(h) * (P(\text{Love}|h) * (P(\text{Love}|h) * P(\text{Pain}|h) * P(\text{Joy}|h) * P(\text{Kick}|h) \\
 &= (2/3) * (5/19) * (5/19) * (1/19) * (5/19) * (1/19) = \mathbf{0.0000067}
 \end{aligned}$$

$$\begin{aligned}
 p(-h|d_7) &= p(-h) * P(\text{Love} | -h) * P(\text{Love} | -h) * P(\text{Pain} | -h) * P(\text{Joy} | -h) * P(\text{Kick} | -h) \\
 &= (1/3) * (2/9) * (2/9) * (4/9) * (1/9) * (2/9) = \mathbf{0.00018}
 \end{aligned}$$

Class label for testing example is: **No**

Summary: Naive Bayes is Not So Naive

- Very Fast, low storage requirements
- Robust to Irrelevant Features
 - Irrelevant Features cancel each other without affecting results
- Very good in domains with many equally important features
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for problem
- A good dependable baseline for text classification
 - **But we will see other classifiers that give better accuracy**



Thank You