text classification with naive Bayes

CS 490A, Fall 2020

Applications of Natural Language Processing

https://people.cs.umass.edu/~brenocon/cs490a_f20/

Brendan O’Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

- Thanks for your exercises!
    - Type/token ratio def'n
- HW1 released today: Naive Bayes text classification!
    - Due Friday, 9/18
- Python tutorial in Tomas' OH
    - Wed 11:15am to 12:45pm
    - See Piazza "logistics" post, as always
- Schedule: https://people.cs.umass.edu/~brenocon/cs490a_f20/schedule.html

text classification

- input: some text **x** (e.g., sentence, document)
- output: a label **y** (from a finite label set)
- goal: learn a mapping function *f* from **x** to **y**

text classification

- input: some text **x** (e.g., sentence, document)
- output: a label **y** (from a finite label set)
- goal: learn a mapping function *f* from **x** to **y**

fyi: basically every NLP problem reduces to learning a mapping function with various definitions of **x** and **y**!

problem	**x**	**y**
sentiment analysis	text from reviews (e.g., IMDB)	{positive, negative}
topic identification	documents	{sports, news, health, …}
author identification	books	{Tolkien, Shakespeare, …}
spam identification	emails	{spam, not spam}

… many more!

input **x**:

From European Union <info@eu.org>
Subject
Reply to [redacted]

Please confirm to us that you are the owner of this very email address with your copy of identity card as proof.

YOU EMAIL ID HAS WON $10,000,000.00 ON THE ONGOING EUROPEAN UNION COMPENSATION FOR SCAM VICTIMS. CONTACT OUR EMAIL:
CONTACT US NOW VIA EMAIL: [redacted] NOW TO CLAIM YOUR COMPENSATION

label **y**: **spam** or **not spam**

we'd like to learn a mapping *f* such that
f(**x**) = **spam**

f can be hand-designed rules

- if "won $10,000,000" in **x**, **y** = **spam**
- if "CS490A Fall 2020" in **x**, **y** = **not spam**

what are the drawbacks of this method?

f can be **learned** from data

- given **training data** (already-labeled **x,y** pairs) learn *f* by maximizing the likelihood of the training data

- this is known as **supervised learning**

training data:

x (email text)	**y** (spam or not spam)
learn how to fly in 2 minutes	spam
send me your bank info	spam
CS585 Gradescope consent poll	not spam
click here for trillions of $$$	spam

… ideally many more examples!

heldout data:

x (email text)	**y** (spam or not spam)
CS585 important update	not spam
ancient unicorns speaking english!!!	spam

training data:

x (email text)	**y** (spam or not spam)
learn how to fly in 2 minutes	spam
send me your bank info	spam
CS585 Gradescope consent poll	not spam
click here for trillions of $$$	spam

… ideally many more examples!

heldout data:

x (email text)	**y** (spam or not spam)
CS585 important update	not spam
ancient unicorns speaking english!!!	spam

vs. preds

learn mapping function on training data, measure its accuracy on heldout data

probability review

- random variable X takes value x with probability $p(X = x)$; shorthand $p(x)$
- joint probability: $p(X = x, Y = y)$
- conditional probability: $p(X = x \mid Y = y)$

$$= \frac{p(X = x, Y = y)}{p(Y = y)}$$

- when does $p(X = x, Y = y) = p(X = x) \cdot p(Y = y)$?

probability of some input text

- goal: assign a probability to a sentence
  - sentence: sequence of *tokens*
    $p(w_1, w_2, w_3, \ldots, w_n)$
    p(the cat sleeps) > p(cat sleeps the)
  - $w_i \in V$ where V is the **vocabulary** (*types*)
- some constraints:

non-negativity	for any $w \in V$, $p(w) \geq 0$
probability distribution, sums to 1	$\sum_{w \in V} p(w) = 1$

how to estimate p(sentence)?

$$p(w_1, w_2, w_3, \ldots, w_n)$$

we could count all occurrences of the sequence

$$w_1, w_2, w_3, \ldots, w_n$$

in some large dataset and normalize by the number of sequences of length *n* in that dataset

$\boxed{\text{how many } \textit{parameters} \text{ would this require?}}$

chain rule

$$p(w_1, w_2, w_3, \ldots w_n \mid y{=}k)$$
$$= p(w_1 \mid y{=}k)\ p(w_2\ , w_1 \mid y{=}k)\ p(w_3 \mid w_1, w_2) \ldots$$

naive Bayes' conditional indepedence assumption:
the probability of generating a word is independent of all other words
(conditional on doc class)

=

this is called the **unigram probability.**
what are its limitations?

toy sentiment example

- vocabulary V: {i, hate, love, the, movie, actor}
- training data (movie reviews):
  - i hate the movie
  - i love the movie
  - i hate the actor
  - the movie i love
  - i love love love love love the movie
  - hate movie
  - i hate the actor i love the movie

labels:
positive
negative

bag-of-words representation

i hate the actor i love the movie

bag-of-words representation

i hate the actor i love the movie

word	count
i	2
hate	1
love	1
the	2
movie	1
actor	1

bag-of-words representation

i hate the actor i love the movie

word	count
i	2
hate	1
love	1
the	2
movie	1
actor	1

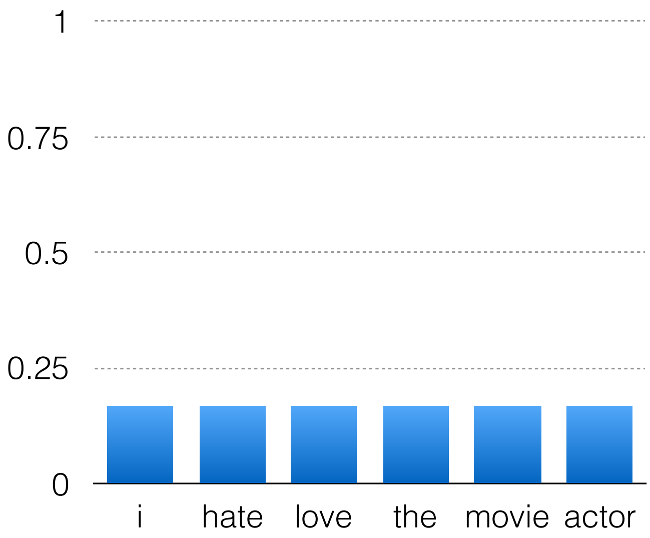
equivalent representation to:

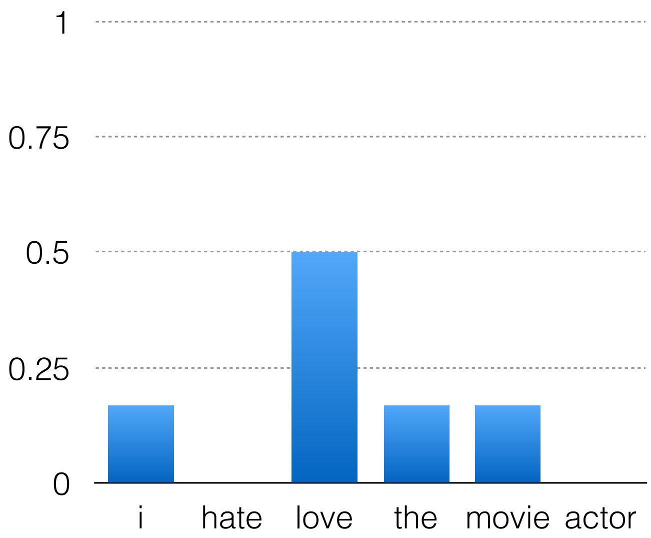
actor i i the the love movie hate

naive Bayes

- represents input text as a bag of words
- assumption: each word is independent of all other words
- given labeled data, we can use naive Bayes to estimate probabilities for unlabeled data
- goal: infer probability distribution that generated the labeled data for each label

which of the below distributions most likely generated the positive reviews?

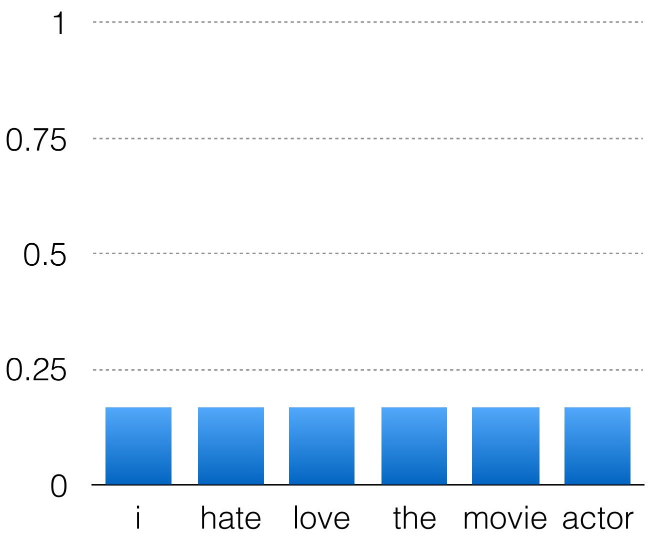




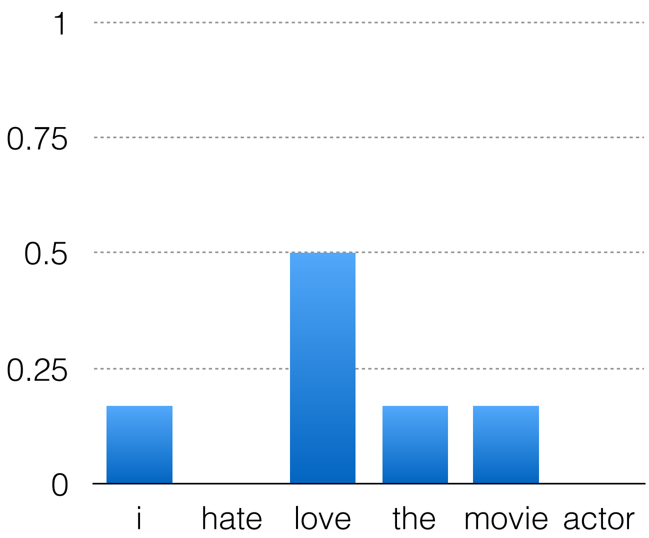
… back to our reviews

$$p(\text{i love love love love love the movie})$$
$$= p(\text{i}) \cdot p(\text{love})^5 \cdot p(\text{the}) \cdot p(\text{movie})$$

$= 5.95374181\text{e-}7$



$= 1.4467592\text{e-}4$

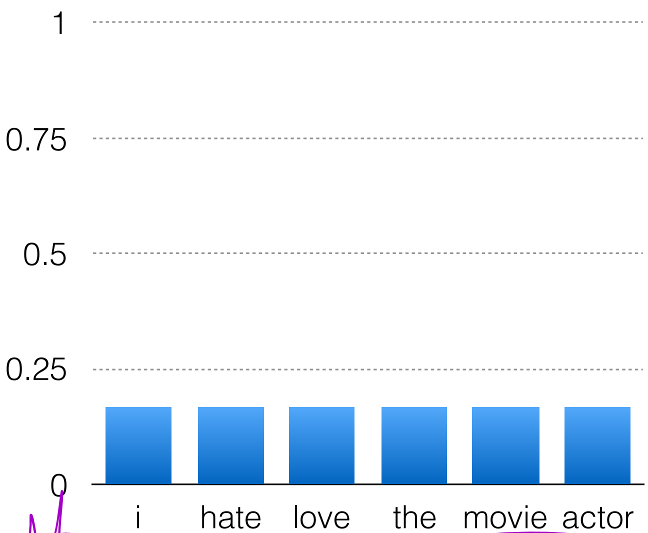


… back to our reviews

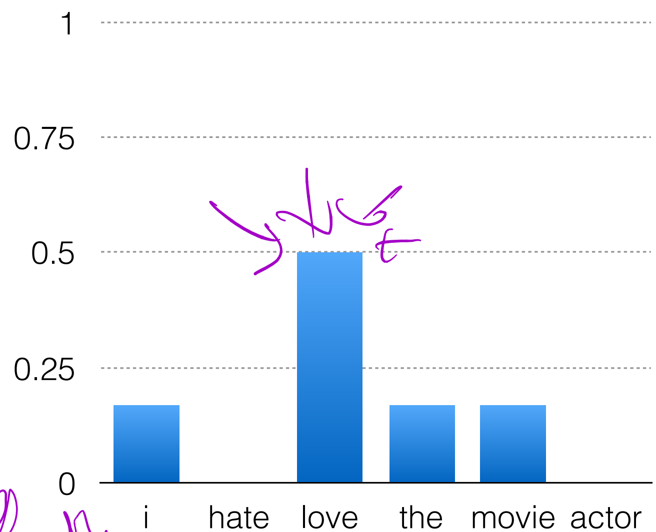
$$p(\text{i love love love love love the movie})$$

$$= p(\text{i}) \cdot p(\text{love})^5 \cdot p(\text{the}) \cdot p(\text{movie})$$

= 5.95374181e-7



= 1.4467592e-4



logs to avoid underflow

$p(w_1) \cdot p(w_2) \cdot p(w3) \ \ldots \cdot p(w_n)$ $= \prod_i p(w_i)$

can get really small esp. with large *n*

$\log\left(\prod_i p(w_i)\right) = \sum_i \log p(w_i)$

logs to avoid underflow

$p \in [0,1]$

$p(w_1) \cdot p(w_2) \cdot p(w3) \ \ldots \cdot p(w_n)$
can get really small esp. with large *n*

$\log 1 = 0$

$$\log \prod p(w_i) = \sum \log p(w_i)$$

$p(\text{i}) \cdot p(\text{love})^5 \cdot p(\text{the}) \cdot p(\text{movie}) \ = 5.95374181\text{e-}7$

$\log p(\text{i}) + 5 \log p(\text{love}) + \log p(\text{the}) + \log p(\text{movie})$

$= -14.3340757538$

class conditional probabilities

Bayes rule (ex: x = sentence, y = label in {pos, neg})

posterior　　prior　　likelihood

$$p(y \mid x) = \frac{\overbrace{p(y)}^{\text{prior}} \cdot \overbrace{P(x \mid y)}^{\text{likelihood}}}{\underbrace{p(x)}_{\text{Normalizer}}}$$

our predicted label is the one with the highest posterior probability

$$\underbrace{\hat{y}}_{\text{PREDICTION}} = \left(\arg\max_y p(y|x)\right) = \arg\max_y \underbrace{p(y)\, p(x|y)}_{\text{"unnorm posterior"}}$$

remember the independence assumption!

maximum a posteriori (MAP) class

$$\hat{y} = \arg\max_{y \in Y} p(y) \cdot P(x \mid y)$$

$$= \arg\max_{y} p(y) \cdot \prod_i P(w_i \mid y)$$

$$= \arg\max_{y} \log p(y) + \sum_i \log P(w_i \mid y)$$

e.g. $\Rightarrow \hat{y} = \text{POS}$

computing the prior…

- i hate the movie
- i love the movie
- i hate the actor
- the movie i love
- i love love love love love the movie
- hate movie
- i hate the actor i love the movie

$p(y)$ lets us encode inductive bias about the labels

we can estimate it from the data by simply counting…

label y	count	p(Y=y)	log(p(Y=y))
positive	3	0.43	-0.84
negative	4	0.57	-0.56

computing the likelihood…

Learned from training data

p(X | y=positive)

word	training count	p(w | y)
i	3	0.19
hate	0	0.00
love	7	0.44
the	3	0.19
movie	3	0.19
actor	0	0.00
total	**16**	

0

p(X | y=negative)

word	training count	p(w | y)
i	4	0.22
hate	4	0.22
love	1	0.06
the	4	0.22
movie	3	0.17
actor	2	0.11
total	**18**	

6 0000 3

p(X | y=positive)

word	count	p(w l y)
i	3	0.19
hate	0	0.00
love	7	0.44
the	3	0.19
movie	3	0.19
actor	0	0.00
total	**16**	

p(X | y=negative)

word	count	p(w l y)
i	4	0.22
hate	4	0.22
love	1	0.06
the	4	0.22
movie	3	0.17
actor	2	0.11
total	**18**	

new review X_{new}: love love the movie

$$\log p(X_{\text{new}} \mid \text{positive}) = \sum_{w \in X_{\text{new}}} \log p(w \mid \text{positive}) = \boxed{-4.96}$$

$$\log p(X_{\text{new}} \mid \text{negative}) = \boxed{-8.91}$$

posterior probs for X_{new}

"log-propor-to"
"proportional to"

$$\log p(\text{positive} \mid X_{\text{new}}) \propto \overbrace{\log P(\text{positive})}^{\log(.4)} + \log p(X_{\text{new}} \mid \text{positive})$$

$$= -0.84 - 4.96 = -5.80$$

$$\log p(\text{negative} \mid X_{\text{new}}) \propto -0.56 - 8.91 = -9.47$$

What does NB predict? $\hat{y}$ = POS

what if we see no positive training documents containing the word "awesome"?

$$p(\text{awesome} \mid \text{positive}) = 0$$

Add-1 (Laplace) smoothing

= "dog"

$$\text{unsmoothed } P(w_i \mid y) = \frac{\text{count}(w_i, y)}{\sum_{w \in V} \text{count}(w, y)}$$

"Max. Lik. Estimator"

"Relative Freq."

= total toks under class y

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

+ 6000

what happens if we do add-α smoothing as α increases?

α = 100

α = 400000

Example

Proceedings of the Conference on Empirical Methods in Natural
Language Processing (EMNLP), Philadelphia, July 2002, pp. 79-86.
Association for Computational Linguistics.

Thumbs up? Sentiment Classification using Machine Learning Techniques

Bo Pang and **Lillian Lee**
Department of Computer Science
Cornell University
Ithaca, NY 14853 USA
{pabo,llee}@cs.cornell.edu

Shivakumar Vaithyanathan
IBM Almaden Research Center
650 Harry Rd.
San Jose, CA 95120 USA
shiv@almaden.ibm.com

[Pang et al., 2002]

490A:
Acc = 68.1%

	Proposed word lists	Accuracy	Ties
Human 1	positive: *dazzling, brilliant, phenomenal, excellent, fantastic* negative: *suck, terrible, awful, unwatchable, hideous*	58%	75%
Human 2	positive: *gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting* negative: *bad, cliched, sucks, boring, stupid, slow*	64%	39%

Figure 1: Baseline results for human word lists. Data: 700 positive and 700 negative reviews.

	Features	# of features	frequency or presence?	NB	ME	SVM
(1)	unigrams	16165	freq.	**78.7**	N/A	72.8
(2)	unigrams	”	pres.	81.0	80.4	**82.9**
(3)	unigrams+bigrams	32330	pres.	80.6	80.8	**82.7**
(4)	bigrams	16165	pres.	77.3	**77.4**	77.1
(5)	unigrams+POS	16695	pres.	81.5	80.4	**81.9**
(6)	adjectives	2633	pres.	77.0	**77.7**	75.1
(7)	top 2633 unigrams	2633	pres.	80.3	81.0	**81.4**
(8)	unigrams+position	22430	pres.	81.0	80.1	**81.6**

Figure 3: Average three-fold cross-validation accuracies, in percent. Boldface: best performance for a given setting (row). Recall that our baseline results ranged from 50% to 69%.

Why did NB win?

- NB/ML sees how words actually used
- Tone or other moderate signals
- Negations??

- Human-Machine Cooperation?

word log-likelihood ratios

- NB's log-posterior is *weighted* word counting

word
i
hate
love
the
movie
actor

p(X | y=positive)

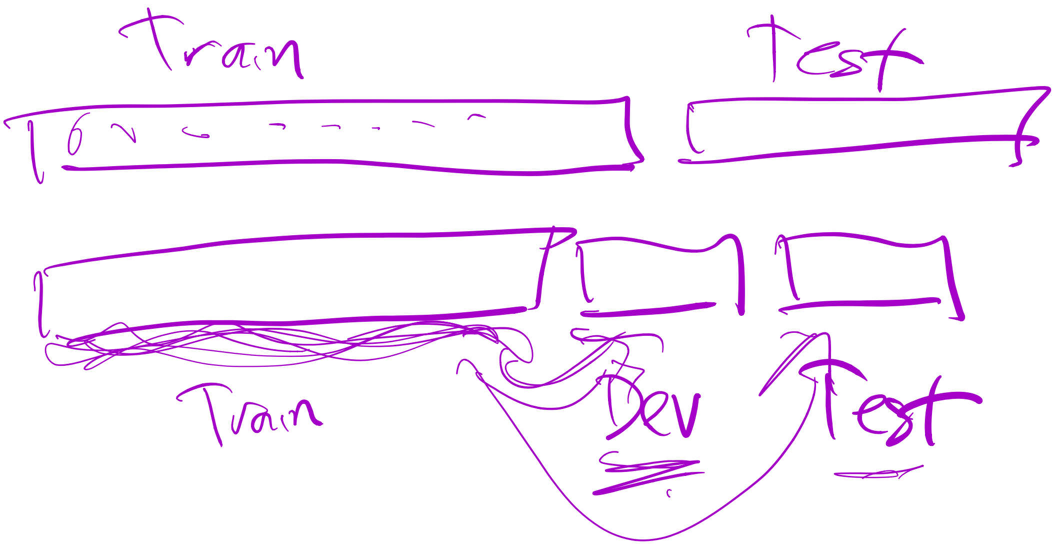
word	count	p(w I y)
i	3	0.19
hate	0	0.00
love	7	0.44
the	3	0.19
movie	3	0.19
actor	0	0.00
total	**16**	

p(X | y=negative)

word	count	p(w I y)
i	4	0.22
hate	4	0.22
love	1	0.06
the	4	0.22
movie	3	0.17
actor	2	0.11
total	**18**	

Data splits for evaluation

- Training vs. Test sets
- Training vs. Development/Tuning vs. Test set



Cross-validation

- Compared to fixed train/dev/test, more useful for small datasets

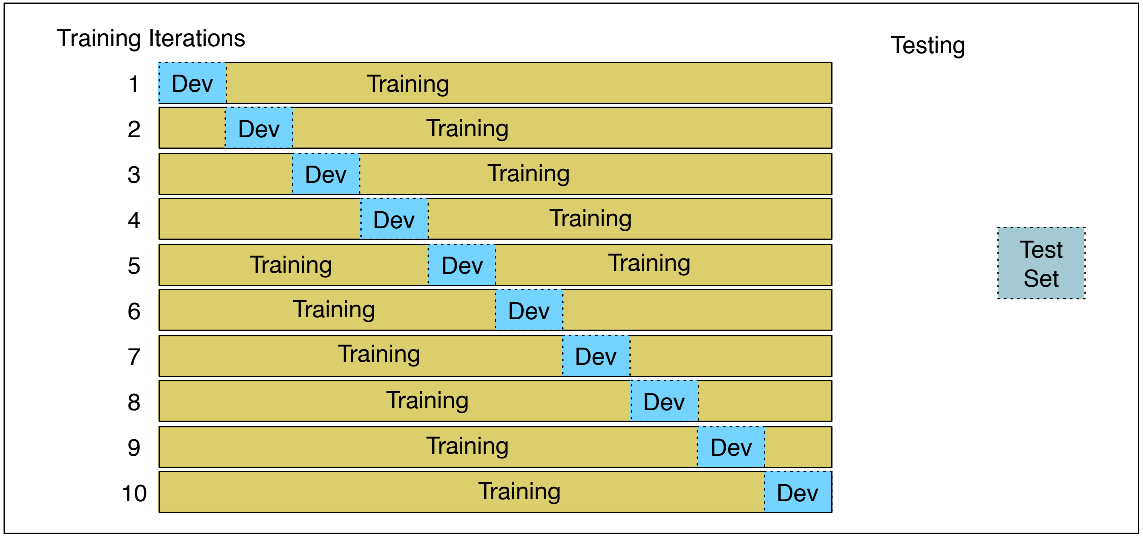


Figure 4.7 10-fold cross-validation