Text classification with Naive Bayes

COMPSCI 485, Applications of Natural Language Processing

Katrin Erk
College of Information and Computer Sciences
University of Massachusetts Amherst

Roadmap

- Machine learning and text classification
- Classification method #1: Manually-defined rules and keywords
- Classification method #2: Supervised learning
  - Naive Bayes
  - Next week: Logistic regression

What is machine learning?

From Wikipedia:

Arthur Samuel:

gives “computers the ability to learn without being explicitly programmed”

Tom Mitchell:

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its perfor- mance at tasks in T, as measured by P, improves with experience E.”

Classification

- Classification: putting a label on each datapoint
  - Is this a noun, an adjective, a verb?
  - Is this a person’s name, a city name, a company name, or neither?
  - Does this text convey a positive or a negative opinion?
  - Is the domain of this text politics, financial, entertainment, sports?

Classification

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

Classification

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

Many NLP problems reduce 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**

Demo: Keyword count classifier

- Task: sentiment classification of movie reviews
- Can this be done with *manually defined* keyword lists?
    - For each category, define set of words
    - Predict a category if many of its words are used
- Let's try manually defined keywords!
    - Sending link on Piazza

f can be hand-designed rules

- if “won $10,000,000” in **x**, then **y** = **spam**
- if “CS485” in **x**, the **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)
CS485 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)
CS485 important update	not spam
ancient unicorns speaking english!!!	spam

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

Supervised learning: Training data, test data

- Classifier uses training data to learn, generalize to new example
- Test performance on new data, with respect to measure P. need **test data**:
  - ? ancient unicorns speaking English!!
  - We know the true label of the test data, but the machine does not.
  - We make the machine “predict” (guess) a label
  - We can then see how many of the machine’s predictions are correct
- **Why is it important to have held-out test data?**

Supervised learning: Training data, test data

- Train on training data
- Test performance on test data
- Possible third dataset: **development data.**
    - train on training data, test on development, calibrate training to do better, repeat. Then, at the end, test on completely unseen test data

Naive Bayes classifiers

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

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

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

Why is this representation convenient?
What do we lose
with this representation?

equivalent representation to:
actor i i the the love movie hate

A probabilistic classifier

- Collection of labels: $C = \{c_1, \ldots, c_n\}$
- Classifier predicts some $c \in C$
- True (gold) class: $\hat{c}$
- **Given a datapoint (document) d, we want to assign the likeliest label:** the label c that has the highest probability $P(c \mid d)$:

$c = \text{argmax}_{c' \in C} P(c' \mid d)$ “the c’ with the maximum P(c’|d)”

A probabilistic classifier

- Collection of labels: $C = \{c_1, \ldots, c_n\}$
- Classifier predicts some $c \in C$
- True (gold) class: $\hat{c}$
- **Given a datapoint (review) d, we want to assign the likeliest label:** the label c that has the highest probability $P(c \mid d)$:

We represent d as its bag of words

$c = \text{argmax}_{c' \in C} P(c' \mid d)$ "the c' with the maximum P(c'|d)"

A probabilistic classifier

- How do we compute the likeliest label for a review? $c = \text{argmax}_{c' \in C} P(c' \mid d)$
    - What factors make a label c likely?

A probabilistic classifier

- How do we compute the likeliest label for a review? $c = \text{argmax}_{c' \in C} P(c' \mid d)$
  - What factors make a label c likely? Same as before:
    - Words that appear often in positive reviews make the label “pos” likelier
    - Words that appear often in negative reviews make the label “neg” likelier
  - But additionally, we make use of overall label frequencies:
    - Do we mostly see negative reviews? Then we want to say “neg” is more likely a priori

A probabilistic classifier

Taking conditional probabilities apart: Bayes' rule

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

or in our case:

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

A probabilistic classifier

Taking conditional probabilities apart: Bayes' rule

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

We can drop P(d): We want to classify review d as either "pos" or "neg". In both P(pos | d) and P(neg | d), the denominator P(d) is exactly the same number.
So we get:

$$P(c \mid d) \sim P(d \mid c)P(c)$$

How likely are the words in the review to appear with this label?

A probabilistic classifier

Taking conditional probabilities apart: Bayes' rule

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

We can drop P(d): We want to classify review d as either "pos" or "neg". In both P(pos | d) and P(neg | d), the denominator P(d) is exactly the same number.
So we get:

$$P(c \mid d) \sim P(d \mid c)P(c)$$

How likely, in general, is this label?

The probability of a review given label “pos”

- We need, for example, P(“I love this movie” | pos)
- Goal: assign a probability to a sentence
    - Sentence: sequence of *tokens* $\langle$ "*I*", "*love*", "*this*", "*movie*"$\rangle$
- Every word w is from a set V, the vocabulary
- How do we compute the probability of a word sequence?

Word probabilities

- The “Bayes” part of Naive Bayes: Bayes’ rule, taking apart the probability $P(c \mid d)$
- The “naive” part: The probability of any word w is independent of all other words in the document. (That’s quite naive, but it works.)
  Say d is the word sequence $d = \langle w_1, \ldots, w_n \rangle$. Then this assumption gives us:

  $$P(d \mid c) \approx \sum_i P(w_i \mid c)$$

Word probabilities

- The “Bayes” part of Naive Bayes: Bayes’ rule, taking apart the probability $P(c \mid d)$
- The “naive” part: The probability of any word w is independent of all other words in the document. (That’s quite naive, but it works.)
  Say d is the word sequence $d = \langle w_1, \ldots, w_n \rangle$. Then this assumption gives us:

$$P(d \mid c) \approx \sum_i P(w_i \mid c)$$

Independence of w1, w2: Then
P(w1 and w2 | c) = P(w1|c) P(w2|c)

Word probabilities

- The “Bayes” part of Naive Bayes: Bayes’ rule, taking apart the probability $P(c \mid d)$
- The “naive” part: The probability of any word w is independent of all other words in the document. (That’s quite naive, but it works.)
  Say d is the word sequence $d = \langle w_1, \ldots, w_n \rangle$. Then this assumption gives us:

$$P(d \mid c) \approx \sum_i P(w_i \mid c)$$

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

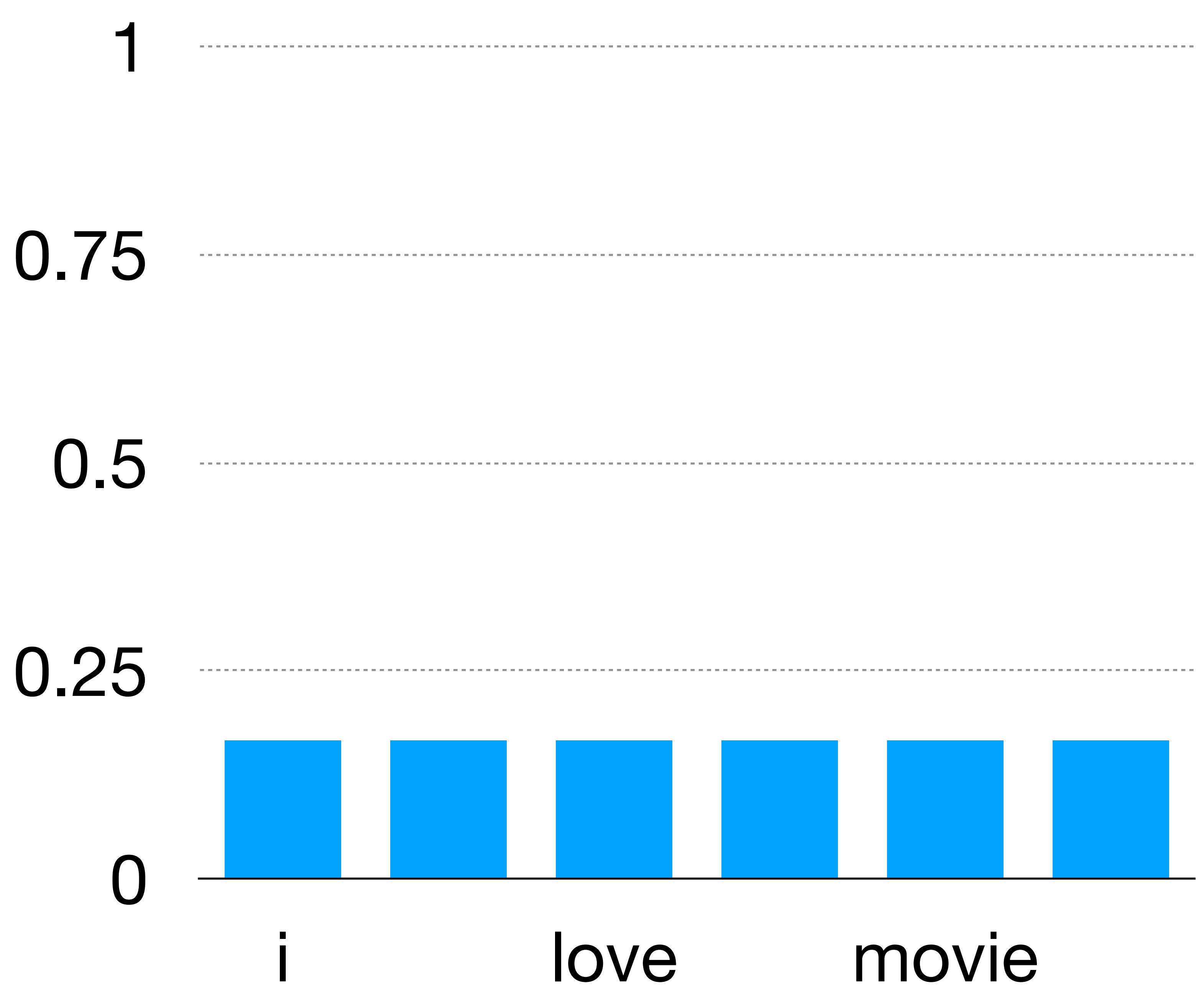
negative

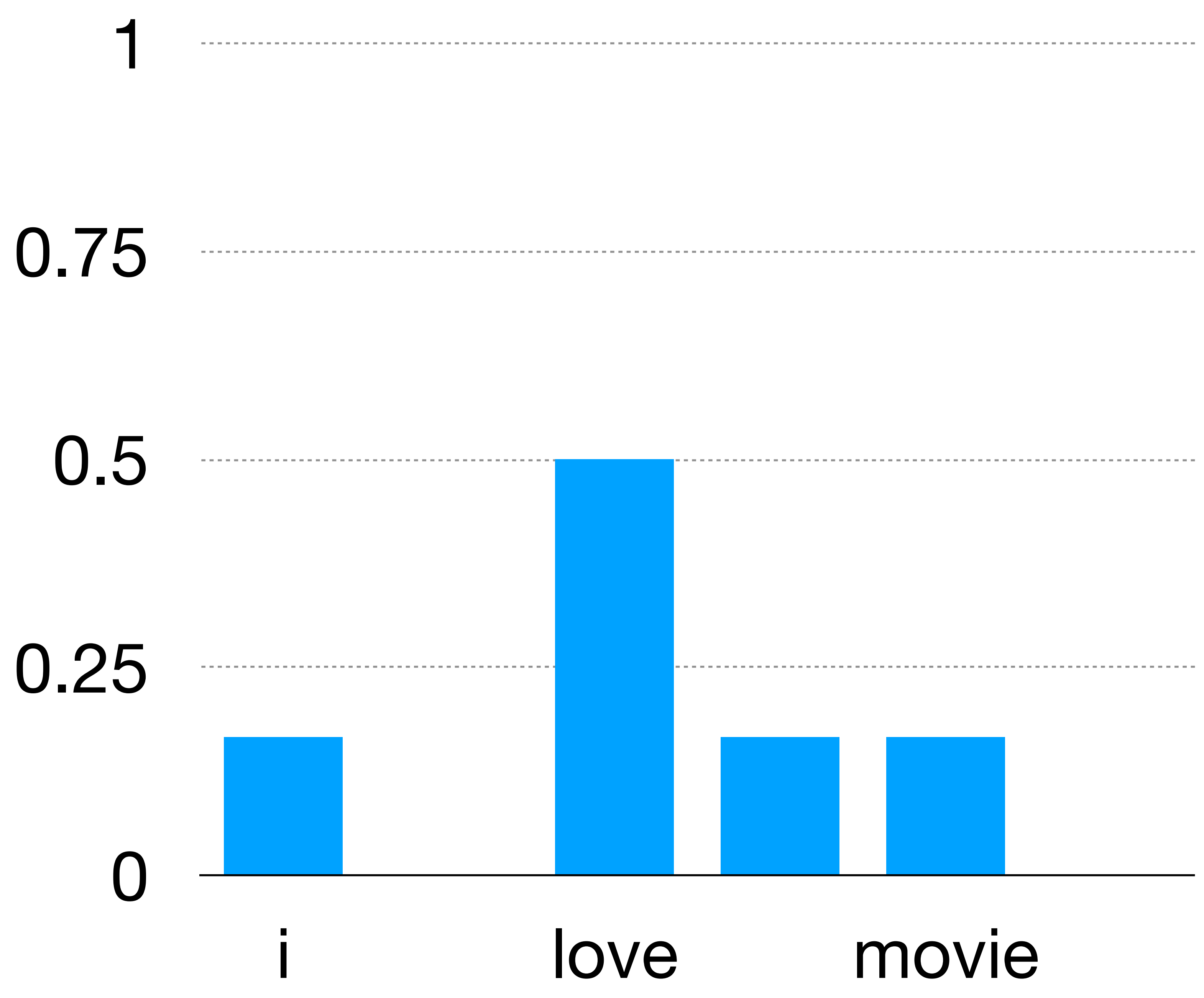
Naive Bayes, again

- Assumption: each word is independent of all other words, conditional on document label

- 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 word distributions looks like one found in positive reviews?



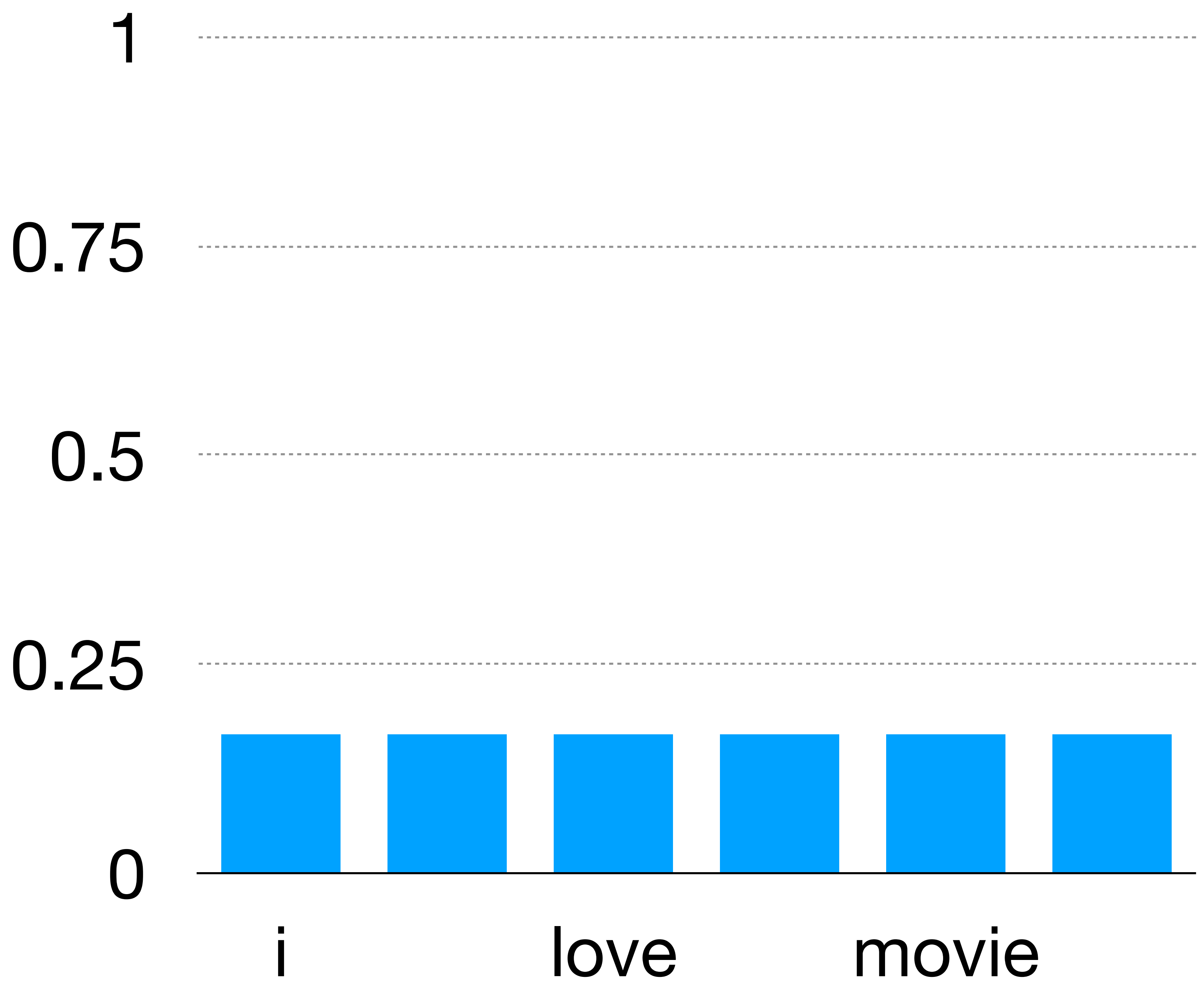


… back to our reviews

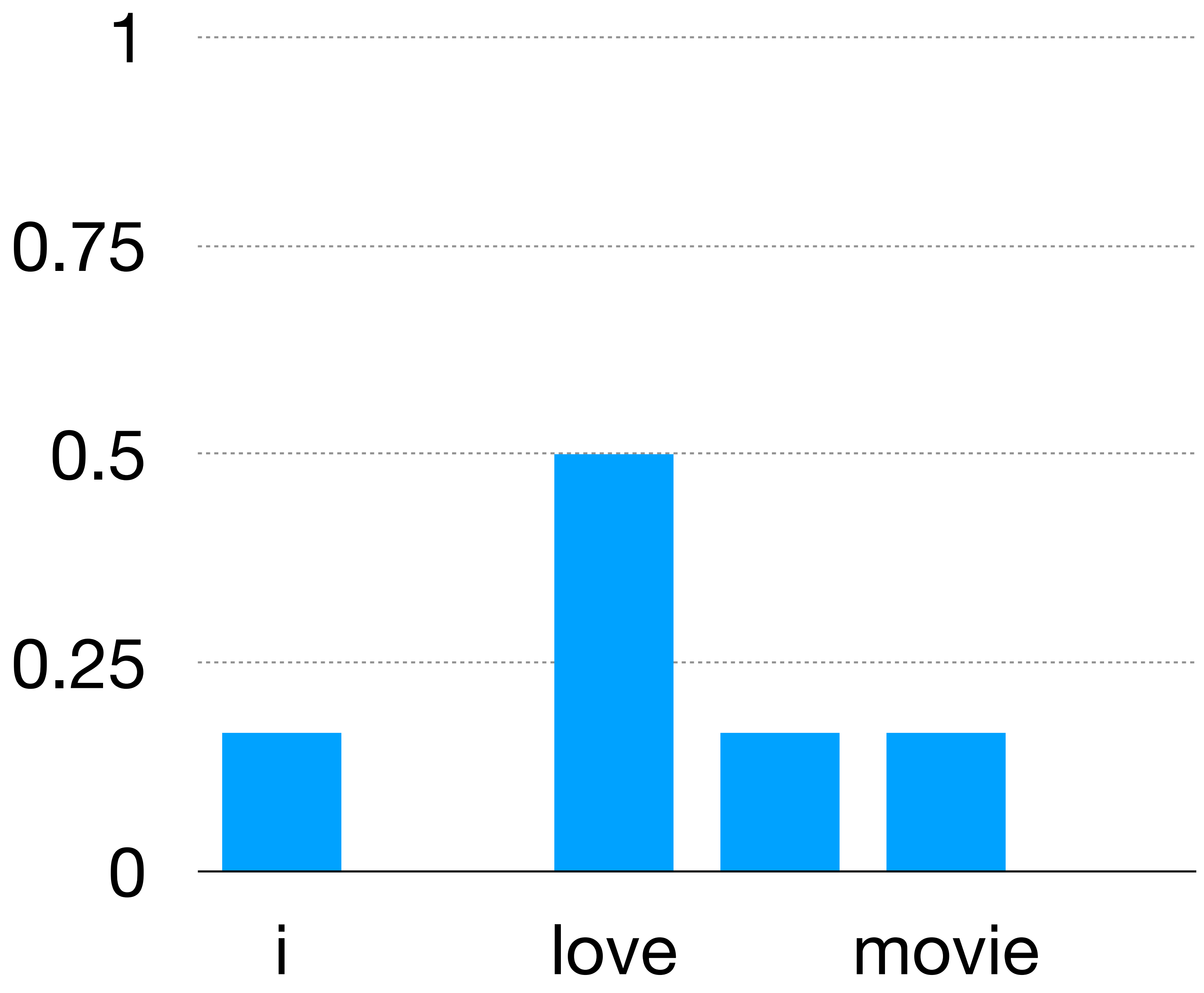
$$p(\text{i love love love love love the movie})$$

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

= 5.95374181e-7



= 1.4467592e-4



Logarithms to avoid underflow

$$p(w_1) \cdot p(w_2) \cdot p(w3) \ \ldots \cdot p(w_n)$$

can get really small esp. with large *n*

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

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

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

$$= -14.3340757538$$

This implementation trick is very common in ML and NLP

Estimating word probabilities from counts

- Our goal: infer probability distribution that generated the labeled data for each label
- One way to estimate P(word | pos) and P(word | neg): use counts in a text
- Probability estimated as relative frequency: “maximum likelihood estimation”, use the distribution that maximizes the likelihood of the training data

Estimating word probabilities from counts

$p(X \mid y{=}POS)$

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**	

$p(X \mid y{=}NEG)$

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**	

$p(X \mid y=\text{POS})$

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**	

$p(X \mid y=\text{NEG})$

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

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

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

How likely are positive reviews in general?

- Reminder: We are estimating the probability of a label c (pos, neg) given a datapoint d as $P(c \mid d) \sim P(d \mid c)P(c)$
- We approximated P(d|c) by the product of the probabilities of individual words
- P(c) is the general probability of seeing a positive or negative review
- How do we estimate that probability?

How likely are positive reviews in general?

- Reminder: We are estimating the probability of a label c (pos, neg) given a datapoint d as $P(c \mid d) \sim P(d \mid c)P(c)$
- We approximated P(d|c) by the product of the probabilities of individual words
- P(c) is the general probability of seeing a positive or negative review
  - This lets us encode the inductive bias about the labels
- How do we estimate that probability?
  - The same was as for P(word | c): through relative frequencies

Computing the prior probability of “pos”, “neg”

- 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

This gives us the following counts and probability estimates:

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

Posterior probabilities for X_{new}

$$\begin{aligned}\log p(\text{POS} \mid X_{\text{new}}) &\propto \log P(\text{POS}) + \log p(X_{\text{new}} \mid \text{POS}) \\ &= -0.84 - 4.96 = -5.80\end{aligned}$$

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

What does NB predict?

Naive Bayes summary

- Assumptions

Naive Bayes summary

- Assumptions:
  - Independence between features, in our case between words in a text: A text is just a bag of words
  - For the test data, assume the probability distribution that makes the training data most likely. (Are there alternatives?)

Naive Bayes summary

- Steps to use

  1. Training: learn p(c) and p(w|c) parameters for all classes and words, based on their counts in labeled training data

  2. Prediction: For a new document, use the learned parameters to predict the (non-normalized) posterior probability of class labels, choose the more likely one

     (Non-normalized: we dropped the denominator P(d))

What if we see no positive training documents containing the word “awesome”?

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

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

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

any review that contains "awesome" will have zero probability for the positive class!

Add-α (pseudocount) smoothing

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

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

what happens if we do

add-α smoothing as α increases?

Example: Training

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Example: Prediction

Model Parameters

New doc x =

P(+) =

P(–) =

w	P(w|+)	P(w|-)
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...	...	...

Other details

- Binarization
  - Issue: overcounting word repetitions
  - Solution:
- Negation handling
  - Issue: “not fun” as bag of words: we lose information of what is negated
  - Solution: