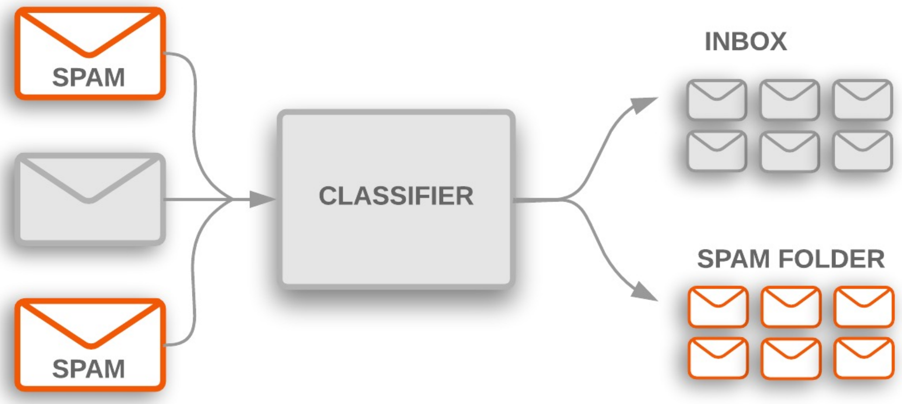
CSE 4392 Special Topics
Natural Language Processing

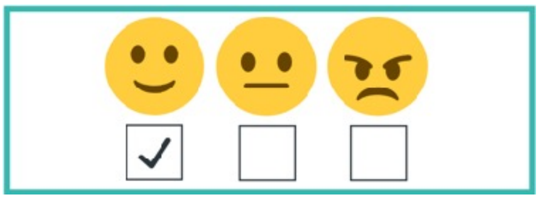
Text Classification

2024 Spring

Why Classify?



Spam detection

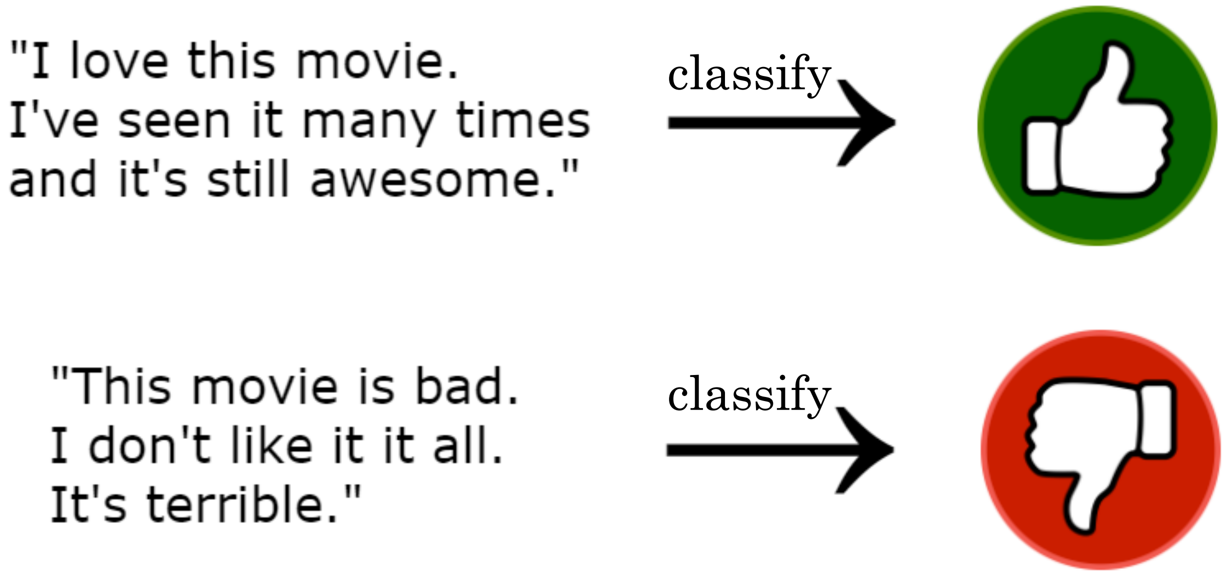


Sentiment analysis

- Authorship attribution
- Language detection
- News categorization

The Classification: The Task

- Inputs:
  - A document d
  - A set of classes $C = \{c_1, c_2, c_3, \dots, c_m\}$
- Output:
  - Predicted class c for document d



Rule-based Classification

- Combination of features on words in the document, and meta-data:
  - IF there exists word w in document d, and w is in {good, great, awesome, extraordinary, …}

    THEN output POSITIVE as class label
  - IF the email subject contains any words in {"casino", "weeds", "viagra", …}

    THEN output SPAM as class label
- Can be very accurate
- But hard and tedius to define (there can be many of them, some even unknown to us!)
- Not easily generalizable (may not apply in other domains or scenarios)

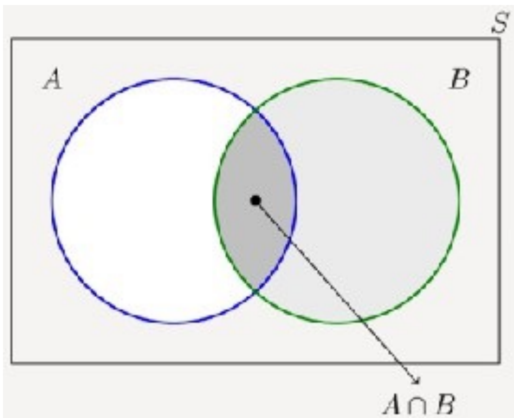
Supervised Learning: Use Statistics

- Data-driven approach
- Let the machine figure out the best patterns to use
- Inputs:
  - Set of *m* classes $C = \{c_1, c_2, \ldots, c_m\}$
  - Set of *n* 'labeled' documents: $\{(d_1, c_1), (d_2, c_2), \ldots, (d_n, c_n)\}$

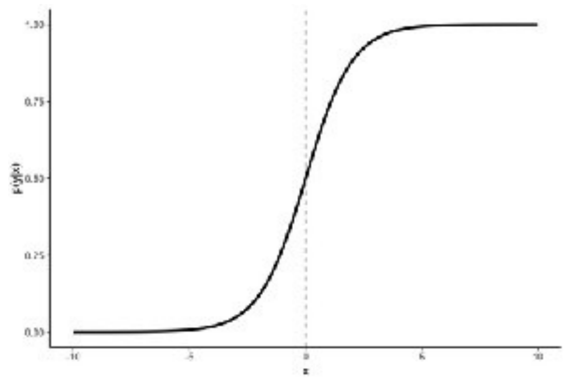
Key questions:
1) The form of F?
2) How to learn F?

- Output:
  - Trained classifier, $F : d \rightarrow c$

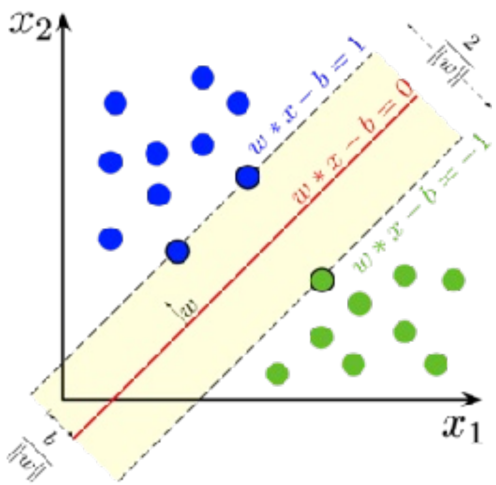
TYPES OF SUPERVISED CLASSIFIERS



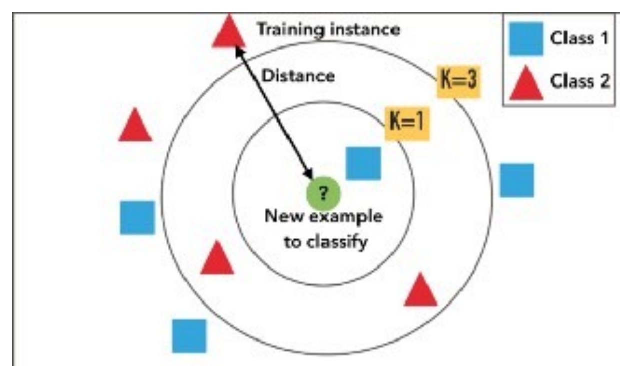
Naive Bayes



Logistic regression



Support vector machines



k-nearest neighbors

QUIZ

- Which of the four types of the classifiers has an inference cost proportional to the size of the training data?

a) Naïve Bayes
b) Logistic Regression
c) Support Vector Machine
d) K-Nearest Neighbors

MULTINOMIAL NAIVE BAYES

- Simple classification model making use of Bayes rule

class

- Bayes Rule:

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

docment

- Makes strong (naive) independence assumptions

Predicting a Class

- Best class:

$$C_{MAP} = \arg\max_{c \in C} P(c|d)$$
$$= \arg\max_{c} \frac{P(c)P(d|c)}{P(d)}$$
$$= \arg\max_{c} \; P(c)P(d|c)$$

d is a document
c is a class

- MAP = Maximum *a Posteriori*
- P(c) → Prior probability of class *c*
- P(d) → constant for *d*, so omitted

HOW TO COMPUTE P(D | C)?

- Option 1: represent the entire sequence of words
  - $P(w_1, w_2, w_3, \ldots, w_k \mid c)$ *(too many sequences!)*

- Option 2: Bag of words
  - Assume position of each word is irrelevant (both absolute and relative)
  - $P(w_1, w_2, w_3, \ldots, w_k \mid c) = P(w_1|c)\, P(w_2|c) \ldots P(w_k|c)$
  - Probability of each word is conditionally independent of each other given class c

BAG OF WORDS

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!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

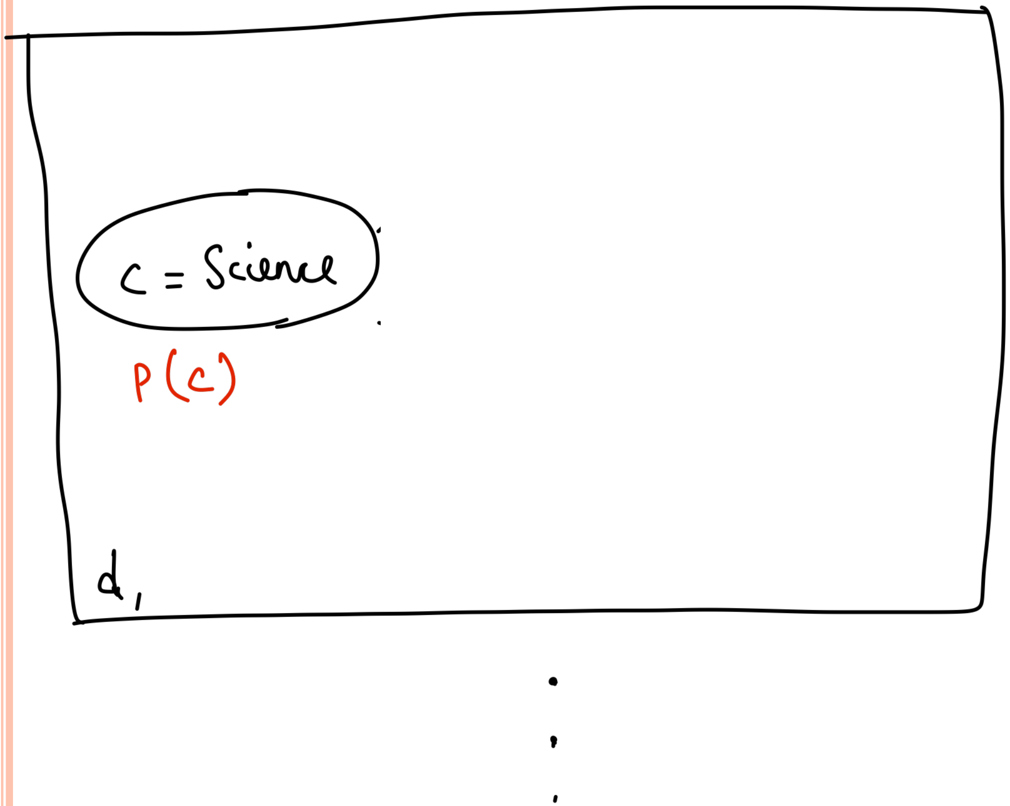
Predicting with Naive Bayes

- Mathematically, we now have:

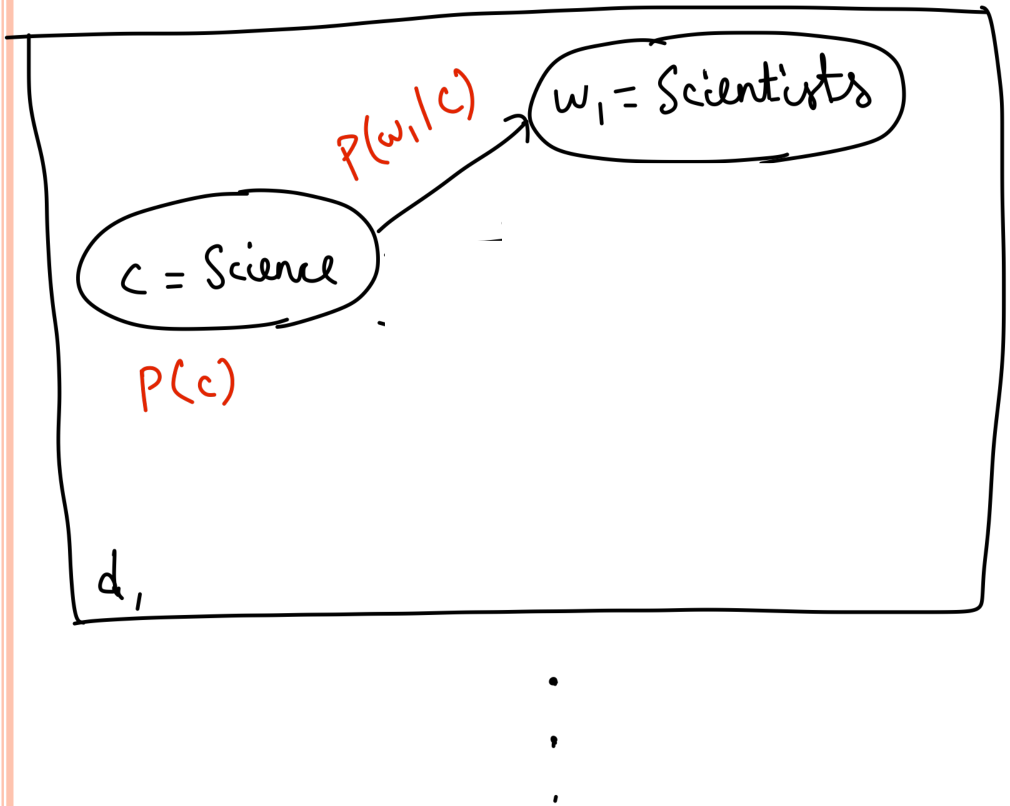
$$
\begin{aligned}
C_{MAP} &= \arg\max_{c} P(d|c)P(c) \\
&= \arg\max_{c} P(w_1, w_2, \ldots, w_k|c)P(c) \\
&= \arg\max_{c} P(c)\prod_{i=1}^{k} P(w_i|c)
\end{aligned}
$$

(Using the BOW assumption!)

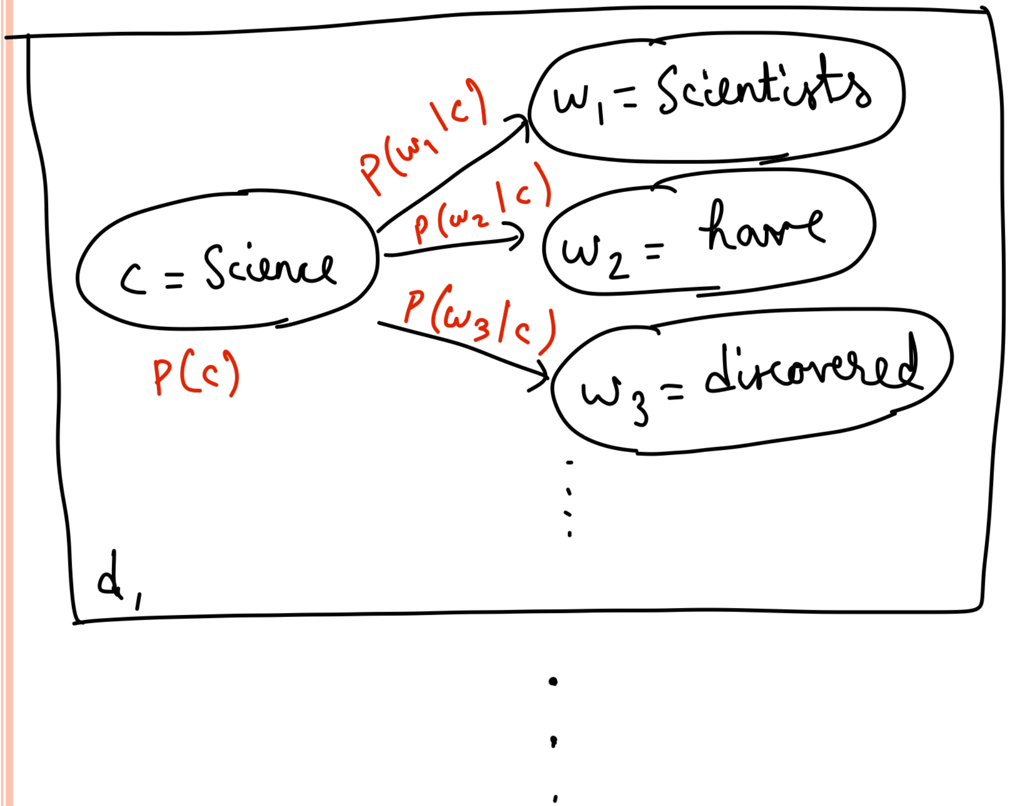
Naive Bayes as a Generative Model



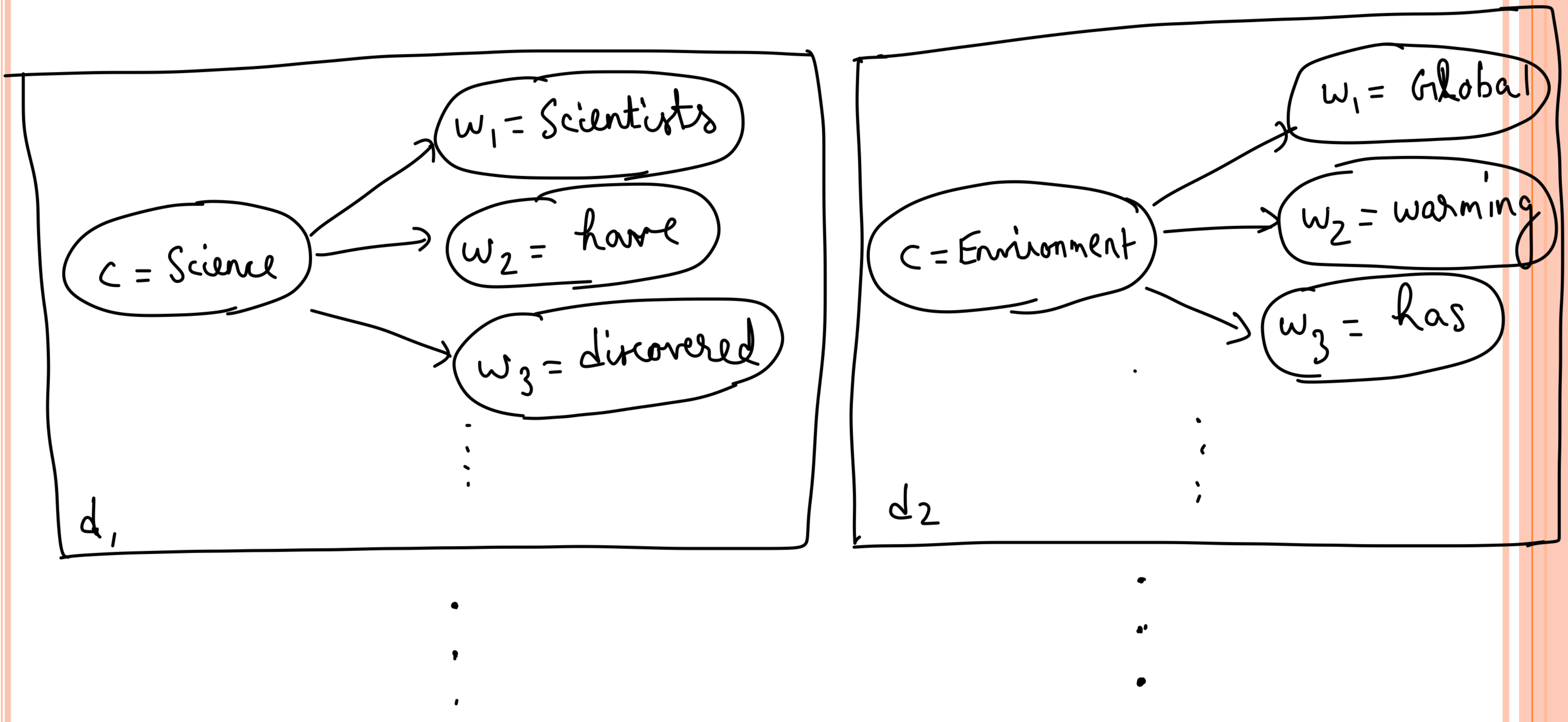
Naive Bayes as a Generative Model



Naive Bayes as a Generative Model



NAIVE BAYES AS A GENERATIVE MODEL



Generate the entire data set one document at a time

ESTIMATING PROBABILITIES

- Maximum likelihood estimates:

# of documents in class c_j

$$\hat{P}(c_j) = \frac{Count(class = c_j)}{\sum_c Count(class = c)}$$

Total # of documents

$$\hat{P}(w_i | c_j) = \frac{Count(w_i, c_j)}{\sum_w Count(w, c_j)}$$

Data sparsity

- What if count('amazing', *positive*) = *0*?
  - Implies P('amazing' | *positive*) = 0
- Given a review document, d = "…. most amazing movie ever …"

$$C_{MAP} = \arg\max_{c} \hat{P}(c) \prod_{i=1}^{k} P(w_i|c)$$

$$= \arg\max_{c} \hat{P}(c) * 0$$

Solution: Smoothing!

- Laplace smoothing:

$$\hat{P}(w_i|c) = \frac{Count(w_i, c) + \alpha}{\sum_w Count(w, c) + \alpha\ |V|}$$

Vocabulary Size

- Simple, easy to use

- Effective in practive

Overall Process

Input: Set of annotated documents $\{(d_i, c_i)\}_{i=1}^{n}$

1. Compute vocabulary set **V** of all words

2. Calculate

$$\hat{P}(c_j) = \frac{Count(\#docs\ in\ c_j)}{Total\ \#\ docs}$$

3. Calculate

$$\hat{P}(w_i|c) = \frac{Count(w_i, c) + \alpha}{\sum_{w \in V}[Count(w, c) + \alpha]}$$

4. (Prediction) Given document $d = (w_1, w_2, \ldots, w_k)$

$$C_{MAP} = \arg\max_{c} \hat{P}(c) \prod_{i=1}^{k} \hat{P}(w_i|c)$$

NAÏVE BAYES CLASSIFICATION EXAMPLE

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

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

	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:

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

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

Choosing a class:

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

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

Conditional Probabilities:

P(Chinese|*c*) = (5+1) / (8+6) = 6/14 = 3/7
P(Tokyo|*c*) = (0+1) / (8+6) = 1/14
P(Japan|*c*) = (0+1) / (8+6) = 1/14
P(Chinese|*j*) = (1+1) / (3+6) = 2/9
P(Tokyo|*j*) = (1+1) / (3+6) = 2/9
P(Japan|*j*) = (1+1) / (3+6) = 2/9

Quiz

	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	6	Macao Chinese Visit Tokyo Chinese	?

- Given the above training documents d1-d4, and their class labels, compute $P(c \mid d_6)$, after applying add-1 smoothing.

FEATURES

- In general, Naive Bayes can use any set of features, not just words
  - URLs, email addresses, Capitalization, …
  - Domain knowledge crucial to performance

Rank	Category	Feature	Rank	Category	Feature
1	Subject	Number of capitalized words	1	Subject	Min of the compression ratio for the bz2 compressor
2	Subject	Sum of all the character lengths of words	2	Subject	Min of the compression ratio for the zlib compressor
3	Subject	Number of words containing letters and numbers	3	Subject	Min of character diversity of each word
4	Subject	Max of ratio of digit characters to all characters of each word	4	Subject	Min of the compression ratio for the lzw compressor
5	Header	Hour of day when email was sent	5	Subject	Max of the character lengths of words
		(a)			(b)
		Spam URLs Features			
1	URL	The number of all URLs in an email	1	Header	Day of week when email was sent
2	URL	The number of unique URLs in an email	2	Payload	Number of characters
3	Payload	Number of words containing letters and numbers	3	Payload	Sum of all the character lengths of words
4	Payload	Min of the compression ratio for the bz2 compressor	4	Header	Minute of hour when email was sent
5	Payload	Number of words containing only letters	5	Header	Hour of day when email was sent

Top features for Spam detection

Naive Bayes and Language Models

- If features = bag of words, each class is a unigram language model!

- For class c, assigning each word: $P(w|c)$
  assigning sentence: $P(S|c) = \prod_{w \in S} P(w|c)$

Class *positive*

0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film
…	

I	love	this	fun	film
0.1	0.1	.05	0.01	0.1

P(s | pos) = 0.0000005

Naïve Bayes as a Language Model

- Which class assigns the higher probability to s?

Model pos

0.1	I
0.1	love
0.01	this
0.05	fun
0.1	film

Model neg

0.2	I
0.001	love
0.01	this
0.005	fun
0.1	film

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

$$P(s|pos) > P(s|neg)$$

Evaluation

- Consider binary classification

- Table of predictions

Confusion Matrix

Truth

Predicted

	Positive	Negative
Positive	100	5
Negative	45	100

false positives

false negatives

- Ideally, we want:

	Positive	Negative
Positive	145	0
Negative	0	105

Evaluation Metrics

		Truth	
		Positive	Negative
Predicted	Positive	100	5
	Negative	45	100

- True positive: Predicted + and actual +
- True negative: Predicted - and actual -
- False positive: Predicted + and actual -
- False negative: Predicted - and actual +

$$Accuracy = \frac{TP + TN}{Total} = \frac{200}{250} = 80\%$$

Evaluation Metrics

		Truth	
		Positive	Negative
Predicted	Positive	100	5
	Negative	45	100

	Positive	Negative
Positive	10	45
Negative	5	190

- True positive: Predicted + and actual +
- True negative: Predicted - and actual -
- False positive: Predicted + and actual -
- False negative: Predicted - and actual +

$$Accuracy = \frac{TP + TN}{Total} = \frac{200}{250} = 80\%$$

Still the same result!
Accuracy is a coarse-grain measure.
Also not suitable for retrieval (finding true positives).

Precision and Recall

- Precision: % of selected classes that are correct

$$\textbf{Precision}(+) = \frac{TP}{TP + FP} \qquad \textbf{Precision}(-) = \frac{TN}{TN + FN}$$

- Recall: % of correct items selected

$$\textbf{Recall}(+) = \frac{TP}{TP + FN} \qquad \textbf{Recall}(-) = \frac{TN}{TN + FP}$$

F-Score

- Combined measure
- Harmonic mean of Precision and Recall

$$F_1 = \frac{2 \cdot \textbf{Precision} \cdot \textbf{Recall}}{\textbf{Precision} + \textbf{Recall}}$$

- Or more generally,

$$F_\beta = \frac{(1 + \beta^2) \cdot \textbf{Precision} \cdot \textbf{Recall}}{\beta^2 \cdot \textbf{Precision} + \textbf{Recall}}$$

Choosing Beta

		Truth	
		Positive	Negative
Predicted	Positive	200	100
	Negative	50	100

$$F_\beta = \frac{(1+\beta^2)\cdot \textbf{Precision} \cdot \textbf{Recall}}{\beta^2 \cdot \textbf{Precision} + \textbf{Recall}}$$

- Which value of Beta maximizes F_β for recall?

A. $\beta = 0.5$

B. $\beta = 1$

C. $\boxed{\beta = 2}$

Aggregating Scores

- We have Precision, Recall, F1 for each class
- How to combine them for an overall score?
  - Macro-average: Compute for each class, then average
  - Micro-average: Collect predictions for all classes and jointly evaluate

Macro vs Micro average

Class 1

	Truth: yes	Truth: no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth: yes	Truth: no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

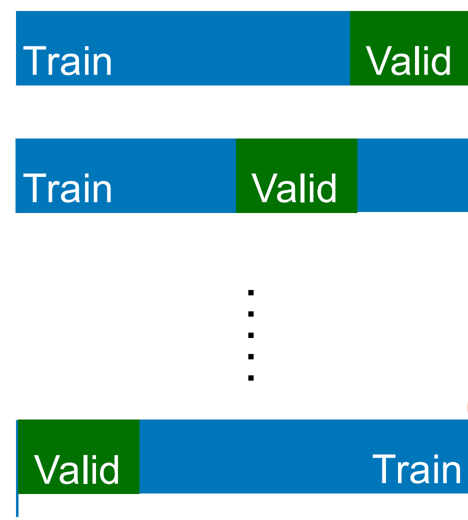
	Truth: yes	Truth: no
Classifier: yes	100	20
Classifier: no	20	1860

- Macroaveraged precision: (0.5 + 0.9)/2 = 0.7
- Microaveraged precision: 100/120 = .83
- Microaveraged score is dominated by score on common classes

Validation

- Choose a metric: Precision/Recall/F1
- Optimize params (α) for metric on Validation (aka Development) set
- Finally evaluate on 'unseen' Test set

- Cross-validation (no need test set):
  - Repeatedly sample several train-val splits (4:1 → 5-fold cross-validation)
  - Reduces bias due to sampling errors



Advantages of Naïve Bayes

- 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
  - Decision Trees suffer from *fragmentation* in such cases – especially if little data
- Optimal if the independence assumptions hold: If assumed independence is correct, then it is the Bayes Optimal Classifier for the problem
- A good dependable baseline for text classification
  - **But we will see other classifiers that give better accuracy**

Practical Naïve Bayes

- Small data sizes:
  - Naive Bayes is great! (high bias)
  - Rule-based classifiers might work well, too
- Medium size datasets:
  - More advanced classifiers might perform better (e.g., SVM, logistic regression)
- Large datasets:
  - Naive Bayes becomes competitive again (although most classifiers work well)

Failings of Naive Bayes (1)

- Independence assumptions are too strong

x1	x2	Class: x_1 XOR x_2
1	1	0
0	1	1
1	0	1
0	0	0

  - XOR problem: Naive Bayes cannot learn a decision boundary
  - Both variables are jointly required to predict class

Failings of Naive Bayes (2)

- Class imbalance:
  - One or more classes have more instances than others in the training data
  - Data skew causes NB to prefer one class over the other
  - Solution: Complement Naive Bayes (Rennie et al., 2003)

Count # times word w_i occurs in classes other than c

$$\hat{P}(w_i|\tilde{c}_j) = \frac{\sum_{c \neq c_j} Count(w_i, c)}{\sum_{c \neq c_j} \sum_w Count(w, c)}$$

Failings of Naive Bayes (3)

- Weight magnitude errors:
  - Classes with larger weights are preferred
  - 10 documents with class=MA and "Boston" occurring once each
  - 10 documents with class=CA and "San Francisco" occurring once each
  - New document: "Boston Boston Boston San Francisco San Francisco"

$$P(class = CA|\ doc) > P(class = MA|doc)$$

(because model treats "San" and "Francisco" as independent words!)

Practical text classification

- Domain knowledge is crucial to selecting good features

- Handle class imbalance by re-weighting classes

- Use log scale operations instead of multiplying probabilities
  - Since $\log(xy) = \log(x) + \log(y)$
  - Better to sum logs of probabilities instead of multiplying probabilities.
  - Class with highest un-normalized log probability score is still most probable.

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

  - Model is now just max of sum of weights