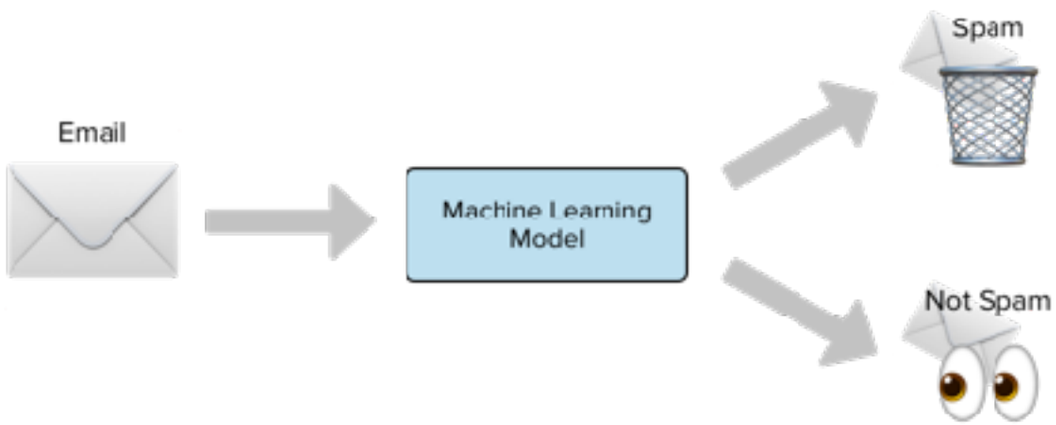
COS 484: Natural Language Processing

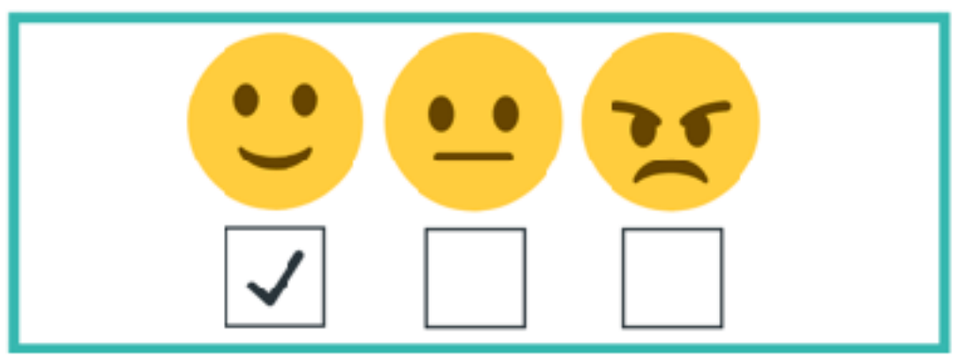
Text Classification

Fall 2019

Why classify?



Spam detection

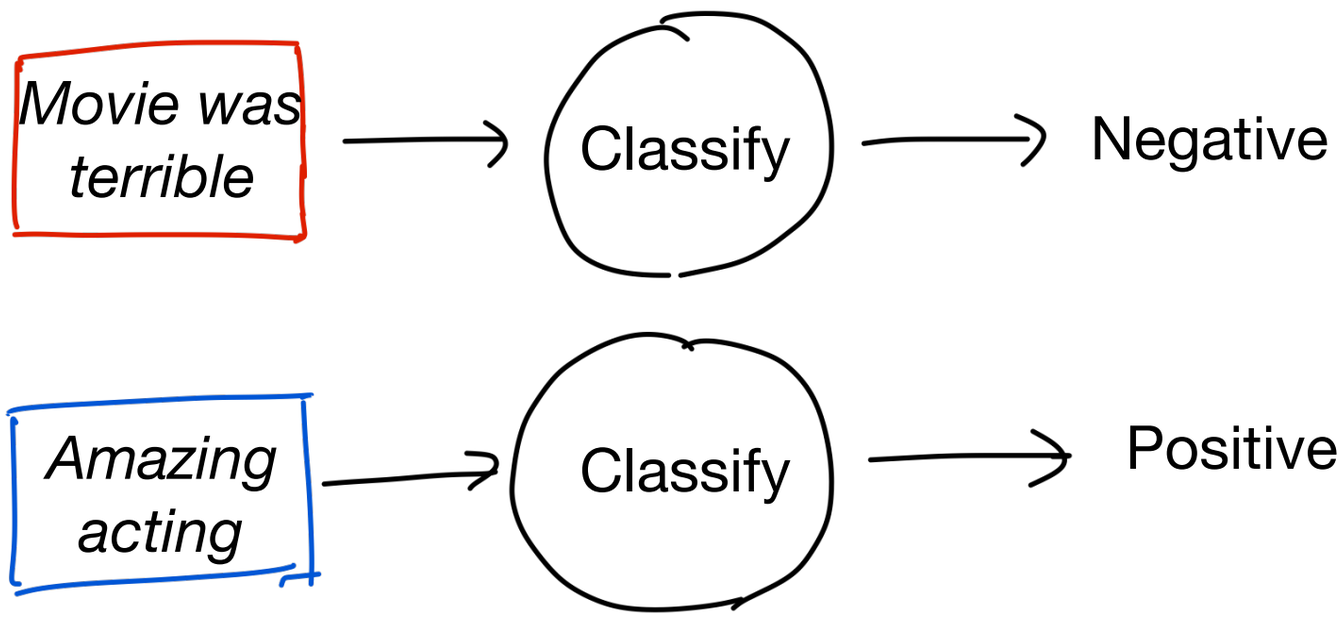


Sentiment analysis

- Authorship attribution
- Language detection
- News categorization

Classification: The Task

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



Rule-based classification

- Combinations of features on words in document, meta-data

  IF there exists word w in document d such that w in [good, great, extra-ordinary, …],
  THEN output Positive

  IF email address ends in [*ithelpdesk.com*, *makemoney.com*, *spinthewheel.com*, …]
  THEN output SPAM

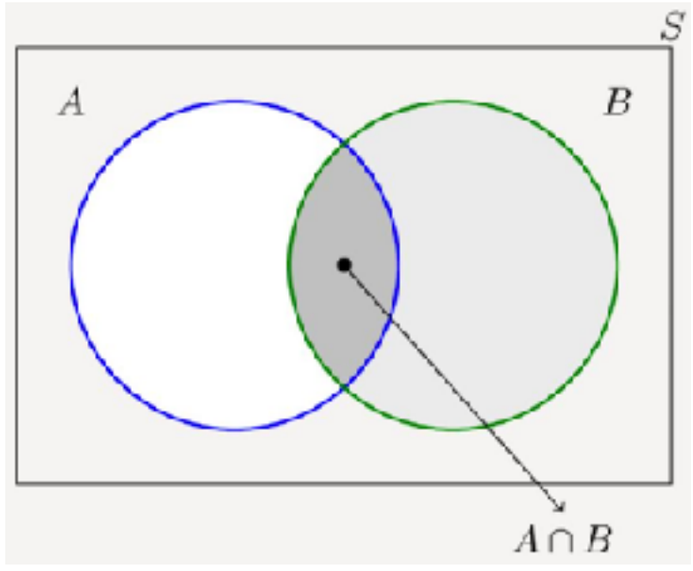
- Can be very accurate
- Rules may be hard to define (and some even unknown to us!)
- Expensive
- Not easily generalizable

Supervised Learning: Let's use statistics!

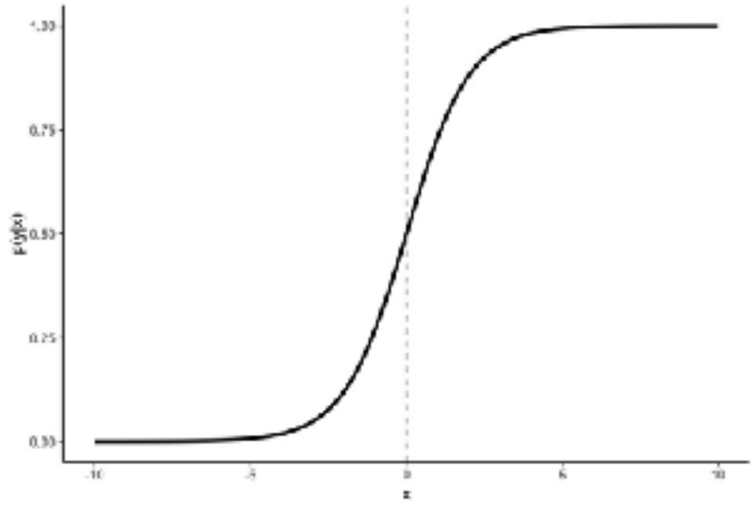
- Data-driven approach
- Let the machine figure out the best patterns to use
- Inputs:
  - Set of *m* classes C = {c_1, c_2, …, c_m}
  - Set of *n* 'labeled' documents: {(d_1, c_1), (d_2, c_2), …, (d_n, c_n)}
- Output:
  - Trained classifier, F : d -> c

KEY QUESTIONS:
a) Form of F?
b) How to learn F?

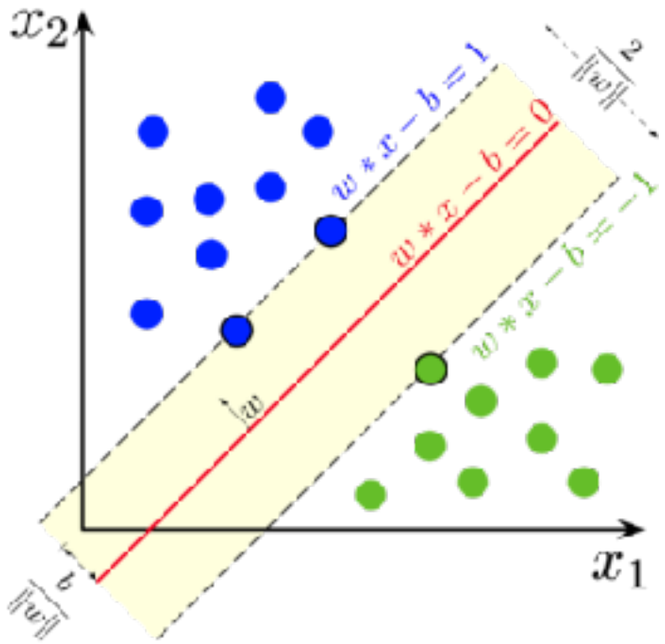
Types of supervised classifiers



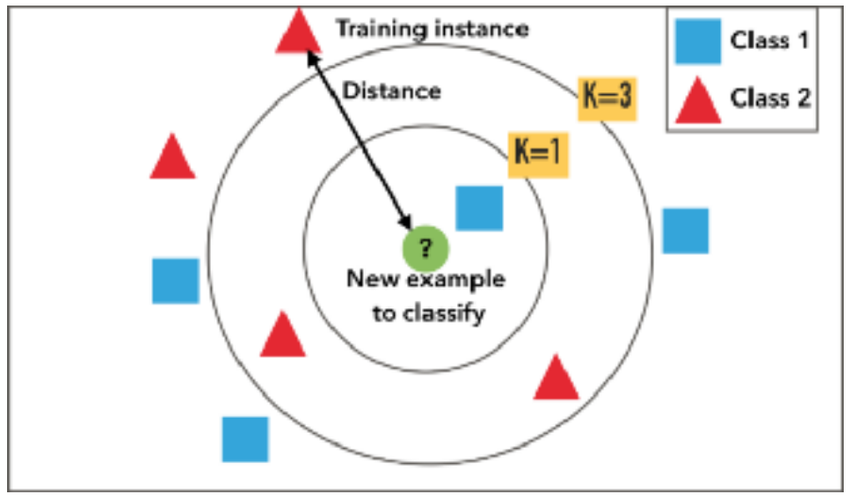
Naive Bayes



Logistic regression



Support vector machines



k-nearest neighbors

Multinomial Naive Bayes

- Simple classification model making use of Bayes rule
  - Bayes Rule:

d – document
c – class

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

- Makes strong (naive) independence assumptions

Predicting a class

d – document
c – class

- Best class, $c_{MAP} = \underset{c \in C}{\operatorname{argmax}} \; P(c|d)$

$$= \underset{c}{\operatorname{argmax}} \; \frac{P(c)\,P(d|c)}{P(d)}$$

$$= \underset{c}{\operatorname{argmax}} \; P(c)\,P(d|c)$$

$P(c)$ ⟶ Prior probability of class $\underline{c}$

$P(d|c)$ ⟶ Conditional probability of generating document $\underline{d}$ from class $\underline{c}$.

How to represent 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* 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

- We now have:

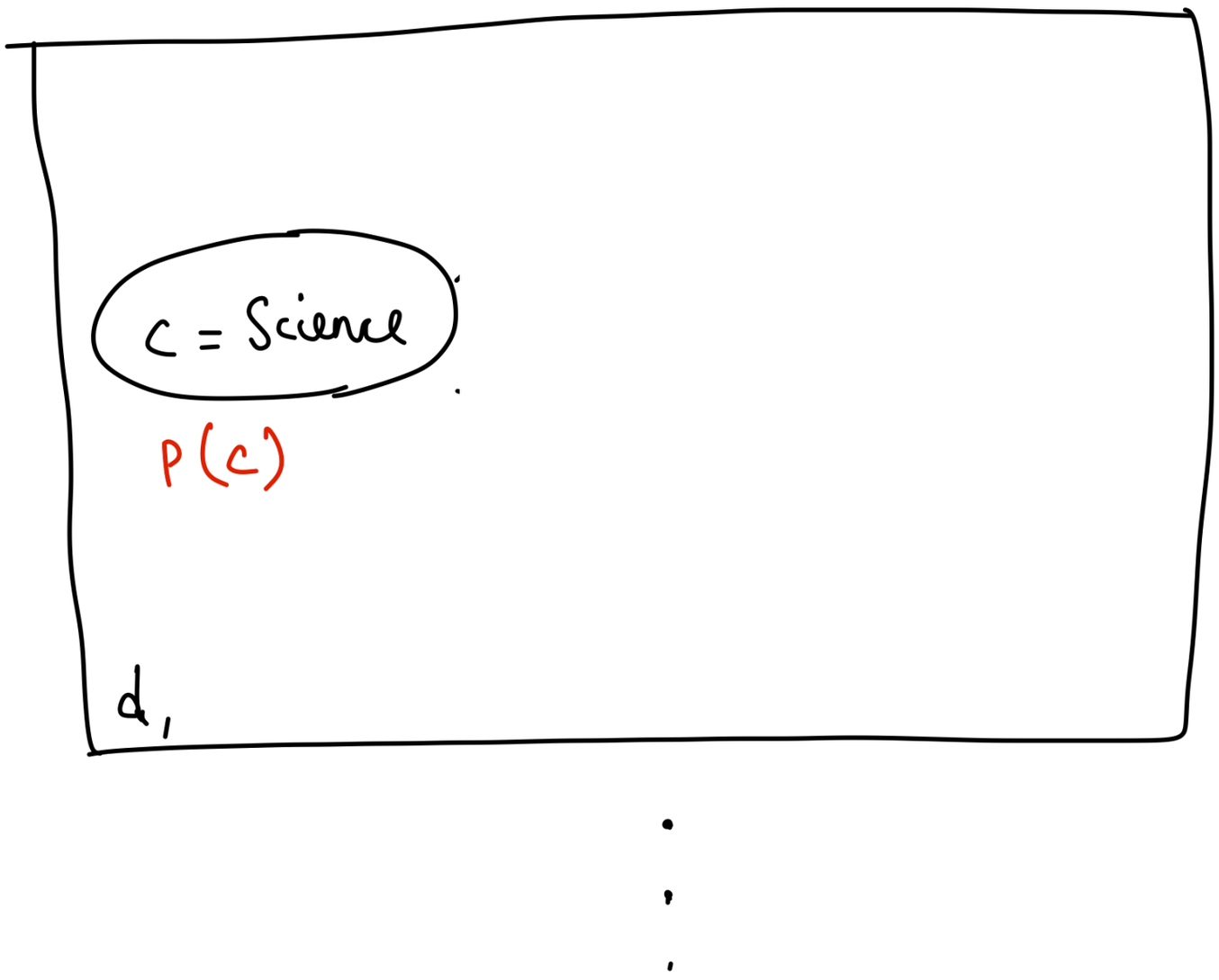
$$c_{MAP} = \underset{c}{\operatorname{argmax}} \; P(d \mid c) \; P(c)$$

$$= \underset{c}{\operatorname{argmax}} \; P(w_1, w_2, \ldots w_k \mid c) \; P(c)$$

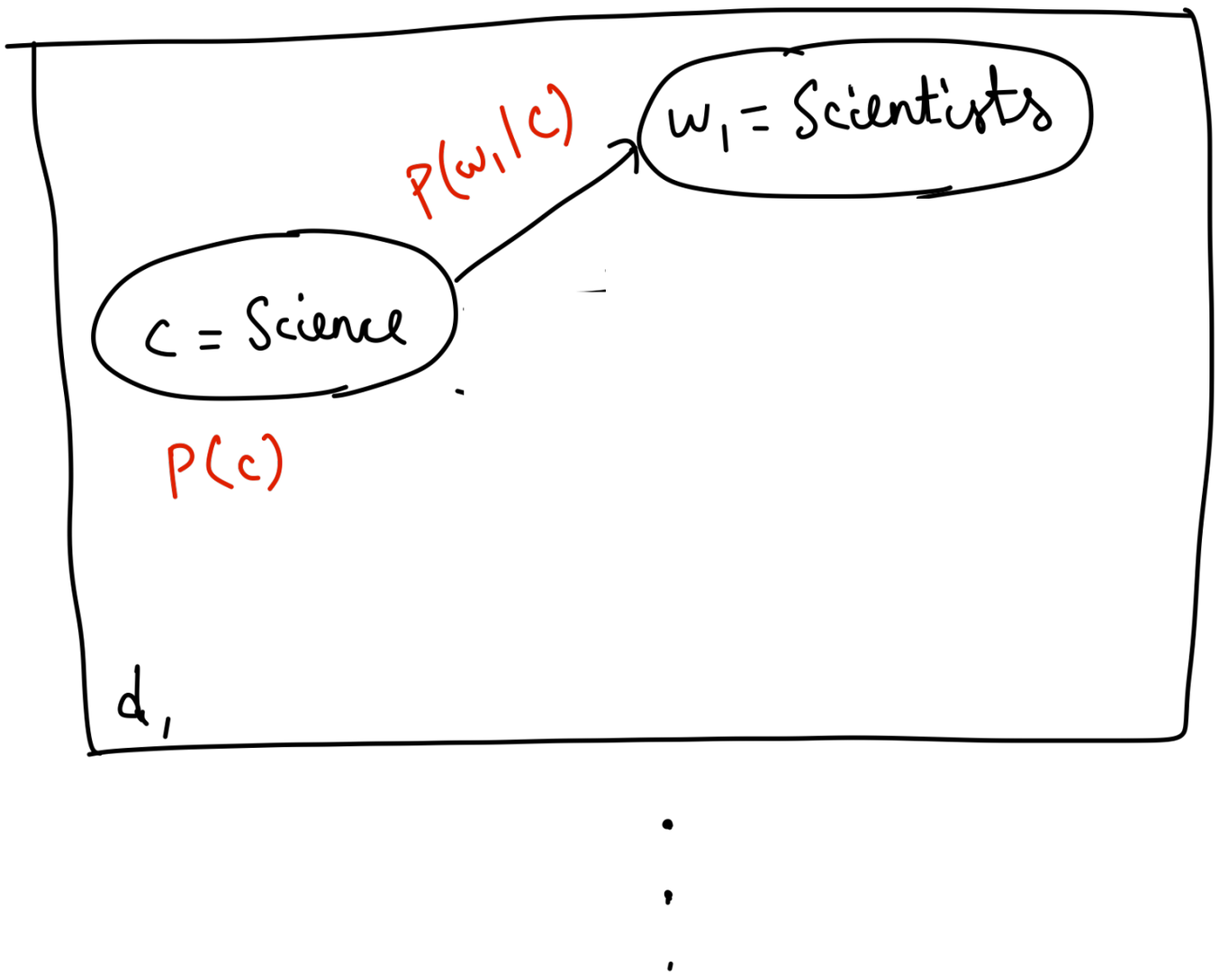
$$= \underset{c}{\operatorname{argmax}} \; P(c) \prod_{i=1}^{k} P(w_i \mid c)$$

(using BOW assumption)

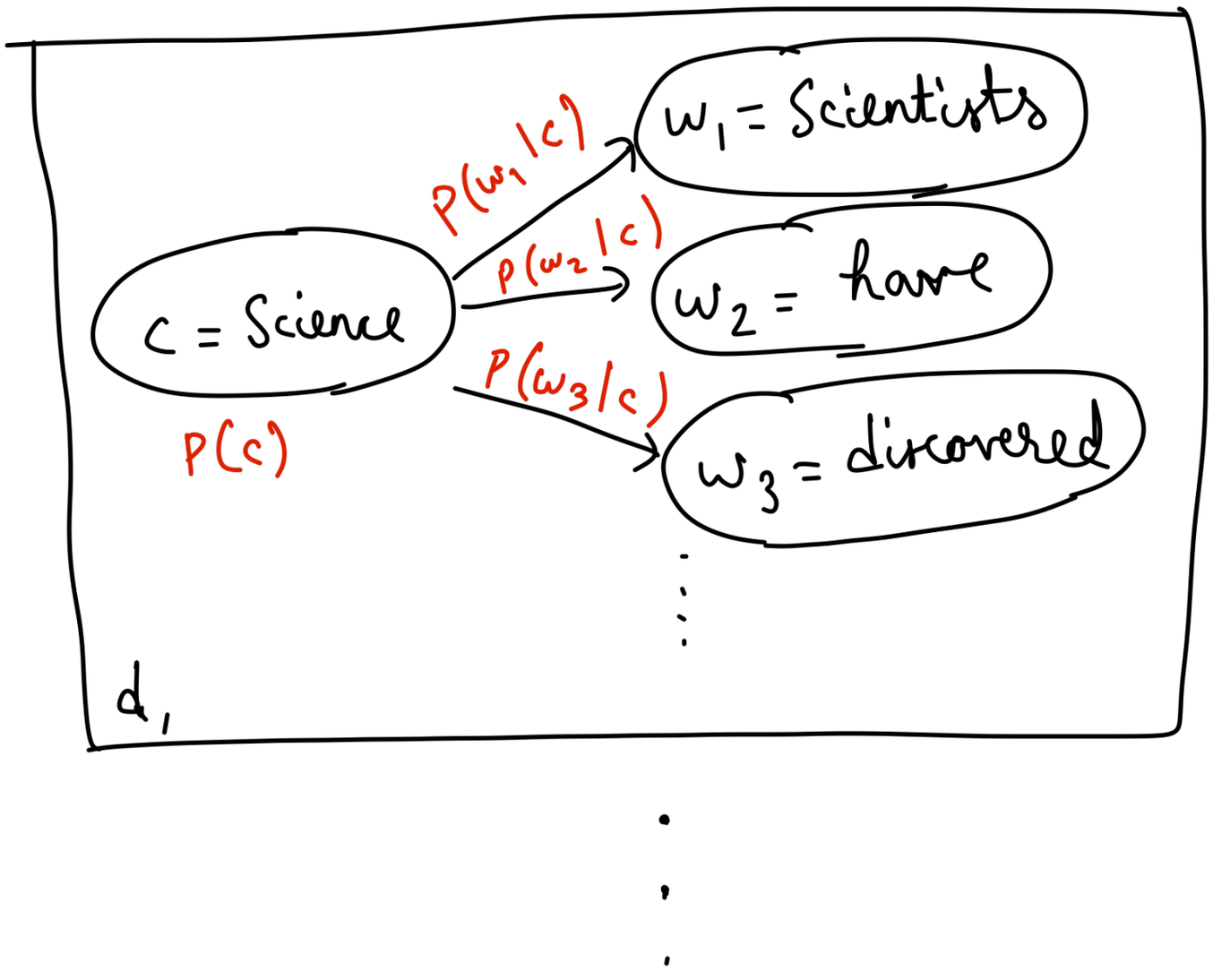
Naive Bayes as a generative classifier



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:
-

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

← Total # of documents

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

Data sparsity

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

$$c_{MAP} = \underset{c}{\operatorname{argmax}} \; \hat{P}(c) \prod_{i=1}^{k} P(\omega_i \mid c)$$

$$= \underset{c}{\operatorname{argmax}} \; \hat{P}(c) \cdot 0$$

Solution: Smoothing!

- Laplace smoothing:

$$\hat{P}(w_i|c) = \frac{\text{Count}(w_i, c) + \alpha}{\left[\sum_w \text{count}(w, c)\right] + \alpha|V|}$$

↖ Vocabulary size

- Simple, easy to use

- Effective in practice

Overall process

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

  A. Compute vocabulary **V** of all words

  B. Calculate $\hat{P}(c_j) = \frac{\text{Count}(c_j)}{\text{Total \# docs}}$

  C. Calculate $\hat{P}(w_i \mid c_j) = \frac{\text{Count}(w_i, c_j) + \alpha}{\sum_{w \in V} [\text{Count}(w, c_j) + \alpha]}$

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

$$c_{MAP} = \arg\max_{c} \hat{P}(c) \cdot \prod_{i=1}^{k} \hat{P}(w_i \mid c)$$

Naive Bayes 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:

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

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

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

Choosing a class:

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

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

Features

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

Top features for Spam detection

Rank	Category	Feature
1	Subject	Number of capitalized words
2	Subject	Sum of all the character lengths of words
3	Subject	Number of words containing letters and numbers
4	Subject	Max of ratio of digit characters to all characters of each word
5	Header	Hour of day when email was sent

(a)

Rank	Category	Feature
1	Subject	Min of the compression ratio for the bz2 compressor
2	Subject	Min of the compression ratio for the zlib compressor
3	Subject	Min of character diversity of each word
4	Subject	Min of the compression ratio for the lzw compressor
5	Subject	Max of the character lengths of words

(b)

Spam URLs Features

Rank	Category	Feature
1	URL	The number of all URLs in an email
2	URL	The number of unique URLs in an email
3	Payload	Number of words containing letters and numbers
4	Payload	Min of the compression ratio for the bz2 compressor
5	Payload	Number of words containing only letters

Rank	Category	Feature
1	Header	Day of week when email was sent
2	Payload	Number of characters
3	Payload	Sum of all the character lengths of words
4	Header	Minute of hour when email was sent
5	Header	Hour of day when email was sent

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 *pos*

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

Naive 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	this	fun	film
0.1	0.1	0.01	0.05	0.1
0.2	0.001	0.01	0.005	0.1

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

Evaluation

- Consider binary classification
- Table of predictions

Truth

Predicted	Positive	Negative
Positive	100	5
Negative	45	100

Confusion Matrix

- Ideally, we want:

	Positive	Negative
Positive	145	0
Negative	0	105

Evaluation Metrics

Truth

Predicted		Positive	Negative
	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 +

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

Evaluation Metrics

Truth

Predicted		Positive	Negative
	Positive	100	5
	Negative	45	100

	Positive	Negative
Positive	100	25
Negative	25	100

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

Coarse metric

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

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

Predicted	Positive	Negative
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 positive class?

A. $\beta = 0.5$

B. $\beta = 1$

C. $\beta = 2$

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

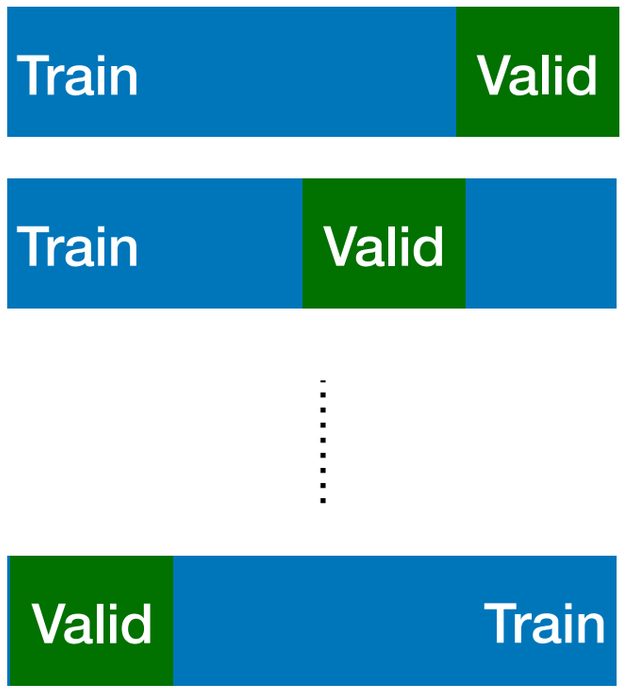
Micro Ave. Table

	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 for metric on Validation (aka Development) set
- Finally evaluate on ‘unseen’ test set
- Cross-validation:
  - Repeatedly sample several train-val splits
  - Reduces bias due to sampling errors



Advantages of Naive 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 problem
- A good dependable baseline for text classification
  - **But we will see other classifiers that give better accuracy**

Practical Naive 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
  - Data skew causes NB to prefer one class over the other
- Solution: Complement Naive Bayes (Rennie et al., 2003)

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

→ Count # times wrd occurs in classes other than 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”

$$P(\text{class} = CA \mid \text{document}) > P(\text{class} = MA \mid \text{document})$$

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