SFU NatLangLab

CMPT 413/713: Natural Language Processing

Classification

Spring 2024
2024-01-15

Adapted from slides from Danqi Chen, Karthik Narasimhan, and Anoop Sarkar

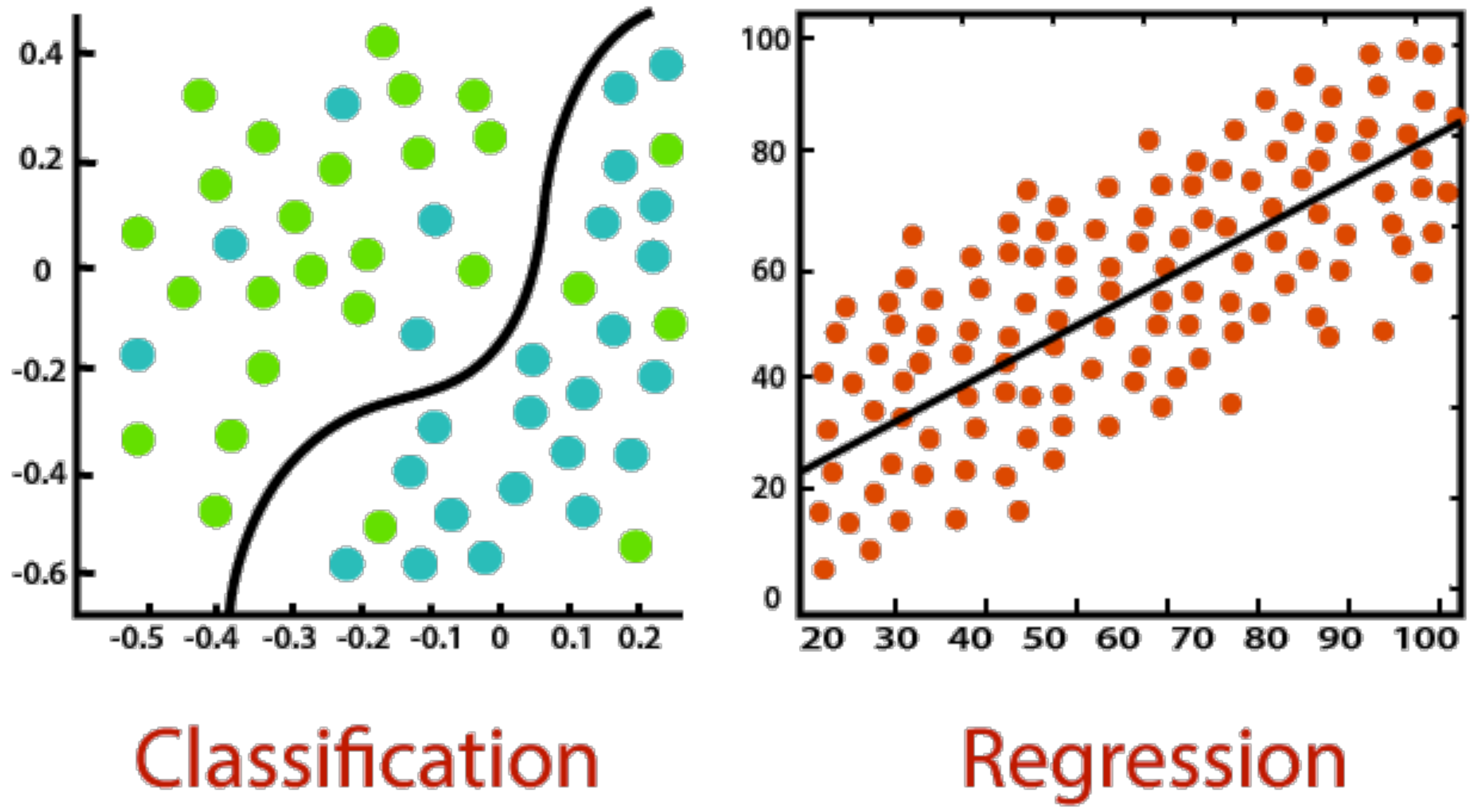
Review: Basic Machine Learning Terminology

labeled training data

- Supervised vs Unsupervised learning

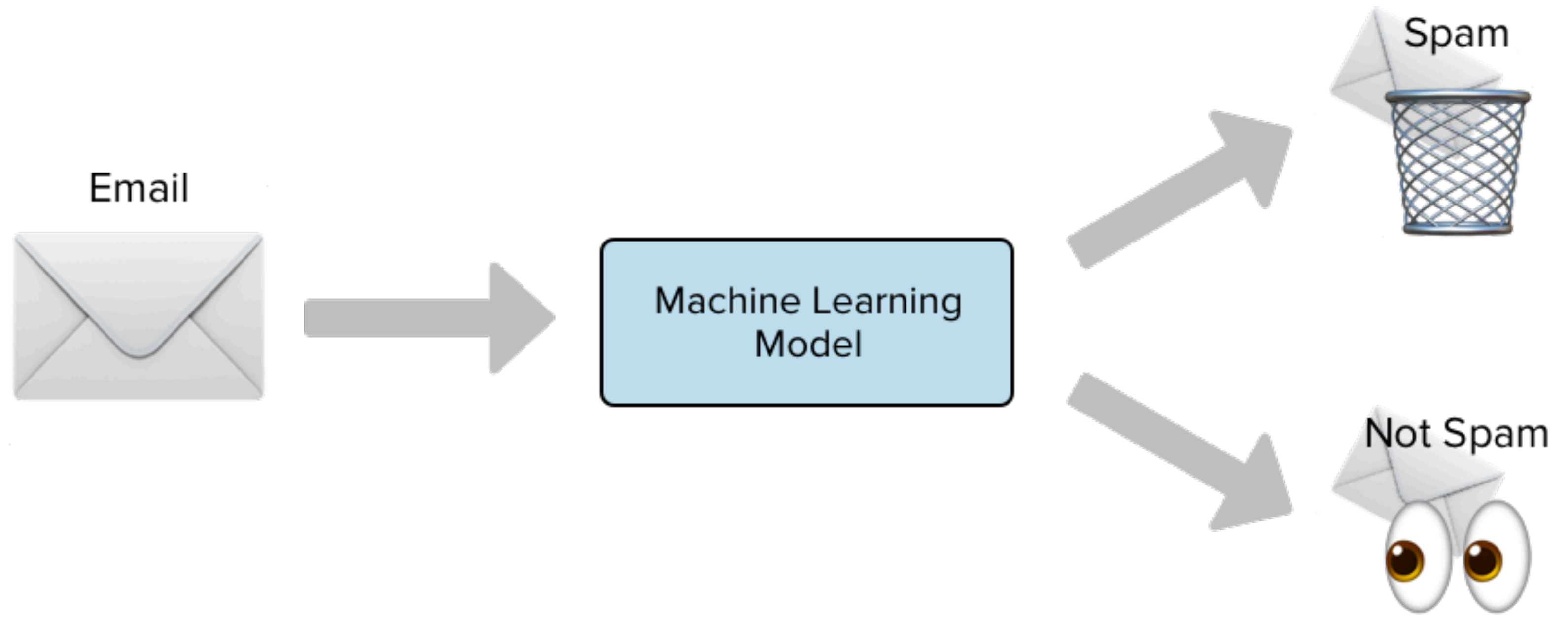
- Classification vs Regression

- Discriminative vs Generative models

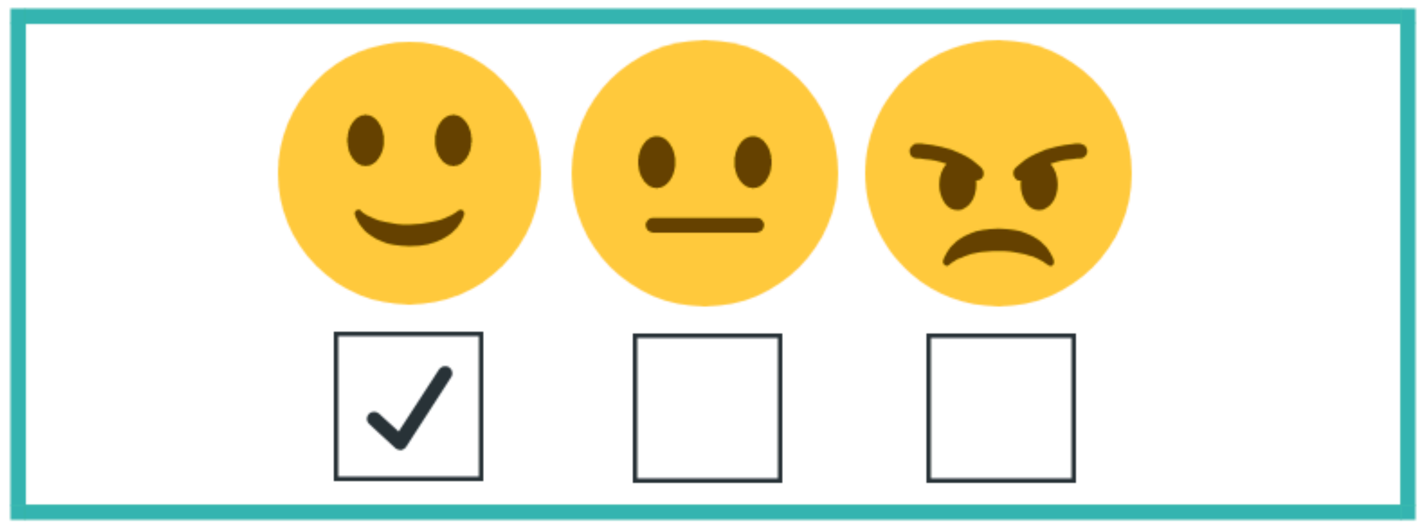


- We will do **Supervised Text Classification**

Why classify?



Spam detection



Sentiment analysis

- Authorship attribution
- Language detection
- News categorization

Movie Reviews

neg: unbelievably disappointing

pos: Full of zany characters and richly applied satire, and some great plot twists

pos: this is the greatest screwball comedy ever filmed

neg: It was pathetic. The worst part about it was the boxing scenes.

Classification as a subtask in NLP

- NLP is all (mostly) about classification
  - Text classification: Spam/Not Spam, Sentiment Analysis
  - Generating sentences: select word to generate at each step (classification over vocabulary!)
  - Building dialog system (identifying intent)
  - Parsing (identifying word to attach to)

Classification as a subtask in NLP

Intent detection

ADDR_CHANGE: I just moved and want to change my address.

ADDR_CHANGE: Please help me update my address

FILE_CLAIM: I just got into a terrible accident and I want to file a claim

CLOSE_ACCOUNT: I'm moving and I want to disconnect my service

Prepositional phrase attachment

noun attach: *I bought **the shirt** **with pockets***

verb attach: *I **bought** the shirt **with my credit card***

noun attach: *I washed **the shirt** **with mud***

verb attach: *I **washed** the shirt **with soap***

Text classification: the task

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

sequence of words
sentence

Multiple classes: m
Binary: m=2

Movie was terrible → Classify → Negative

Amazing acting → Classify → Positive

Rule-based classification

- Look for patterns, and 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

- Simple, can be very accurate
- But: 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 $C = \{c_1, c_2, \ldots, c_m\}$
  - Set of n 'labeled' documents: $\{(d_1, c_1), (d_2, c_2), \ldots, (d_n, c_n)\}$

- Output: Trained classifier, $F : d \rightarrow c$
  - What form should F take?
  - How to learn F?

Designing machine learning models

general recipe

Building the model

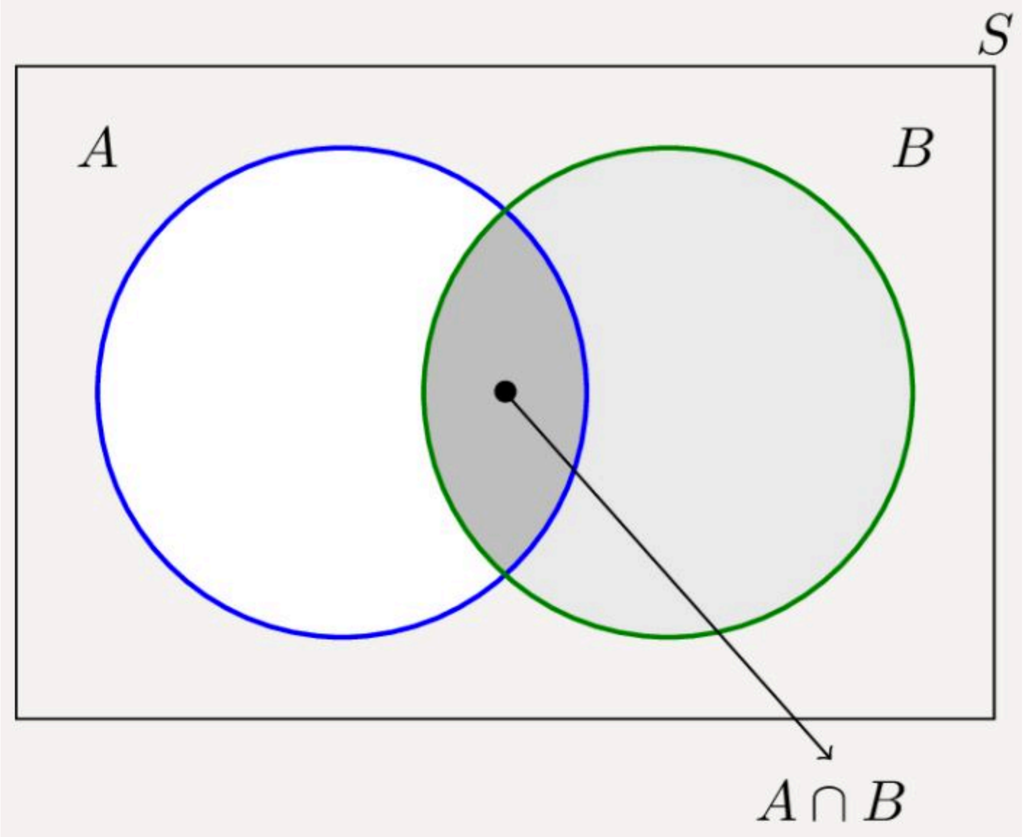
- Input **features**: $f(x) \rightarrow [f_1, f_2, \ldots, f_m]$
  - Need to determine features
- Output: estimate $P(y \mid x)$ for each class c
  - Need to model $P(y \mid x)$ with a **family of functions**
- Train phase: Learn **parameters** of model to minimize loss function
  - Need **training objective** and **optimization** algorithm
- Test phase: Apply parameters to predict class given a new input

General guidelines for model building

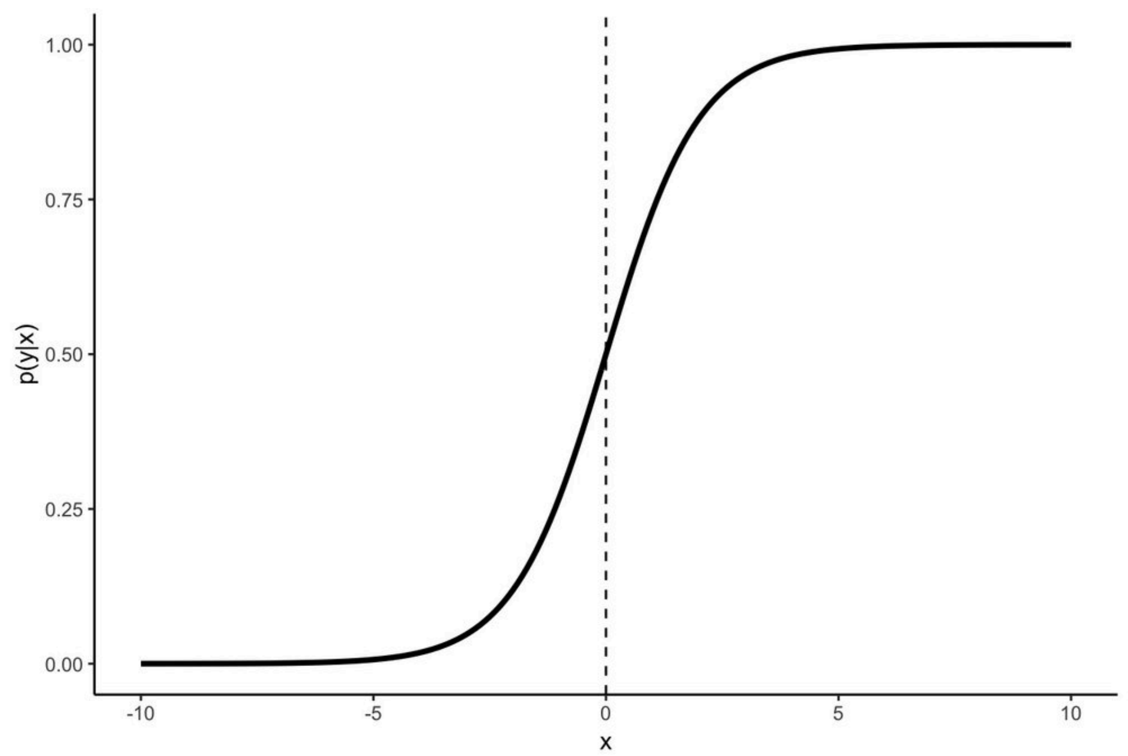
Two steps to building a probability model:

1. Define the model
    - What form should F take?
    - What independence assumptions do we make?
    - What are the model parameters (probability values)?
2. Estimate the model parameters (training/learning)
    - How to learn F? What to optimize? What is the training objective?

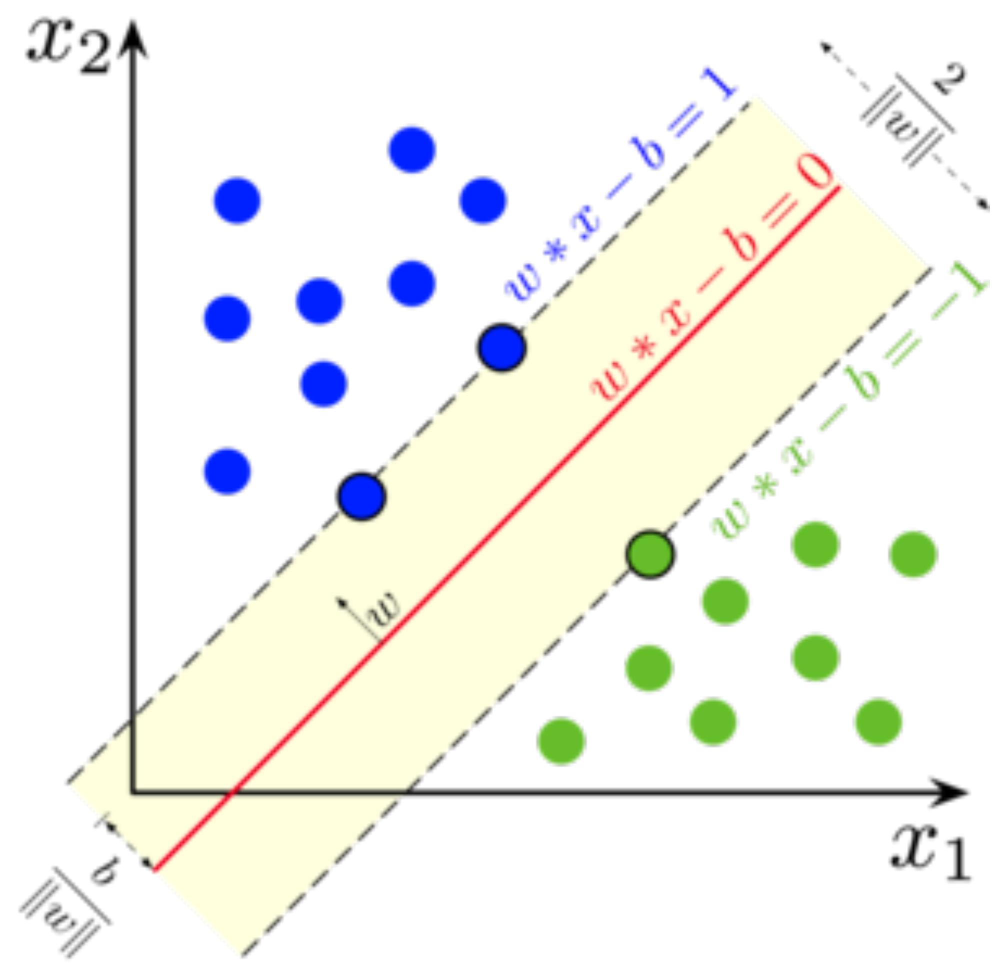
Types of supervised classifiers



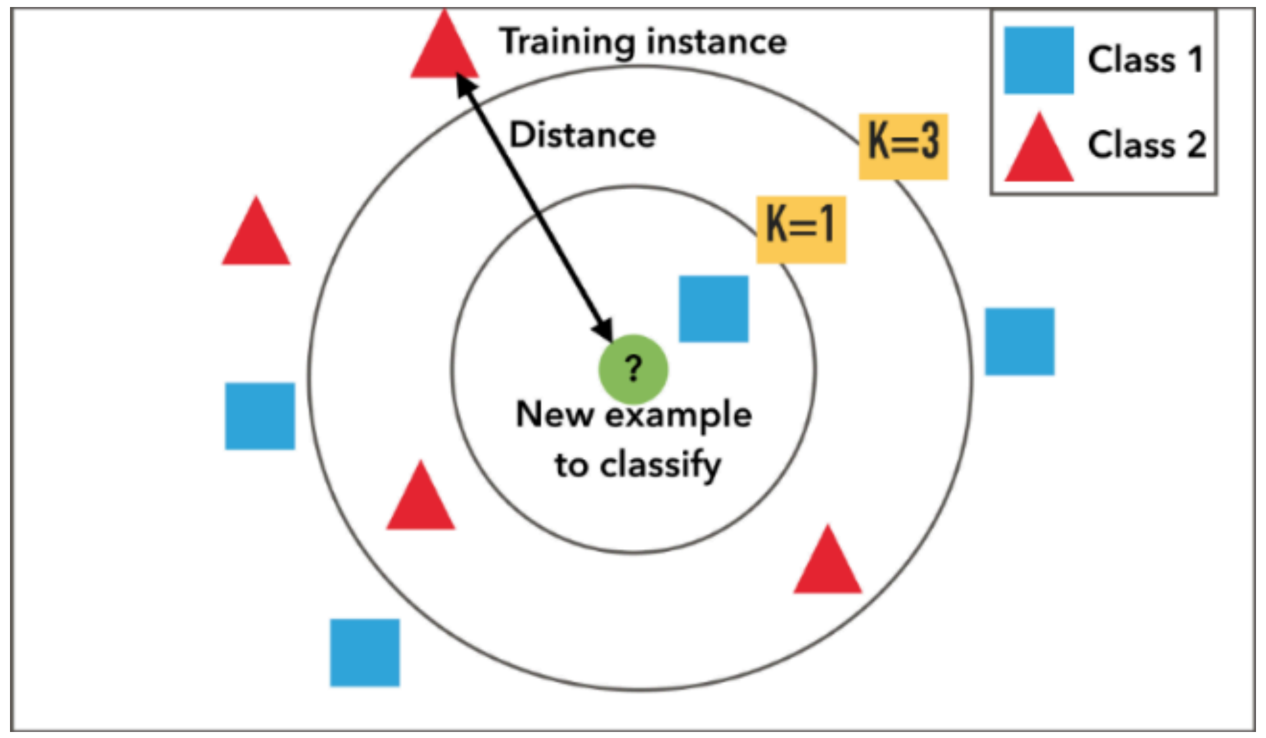
Naive Bayes



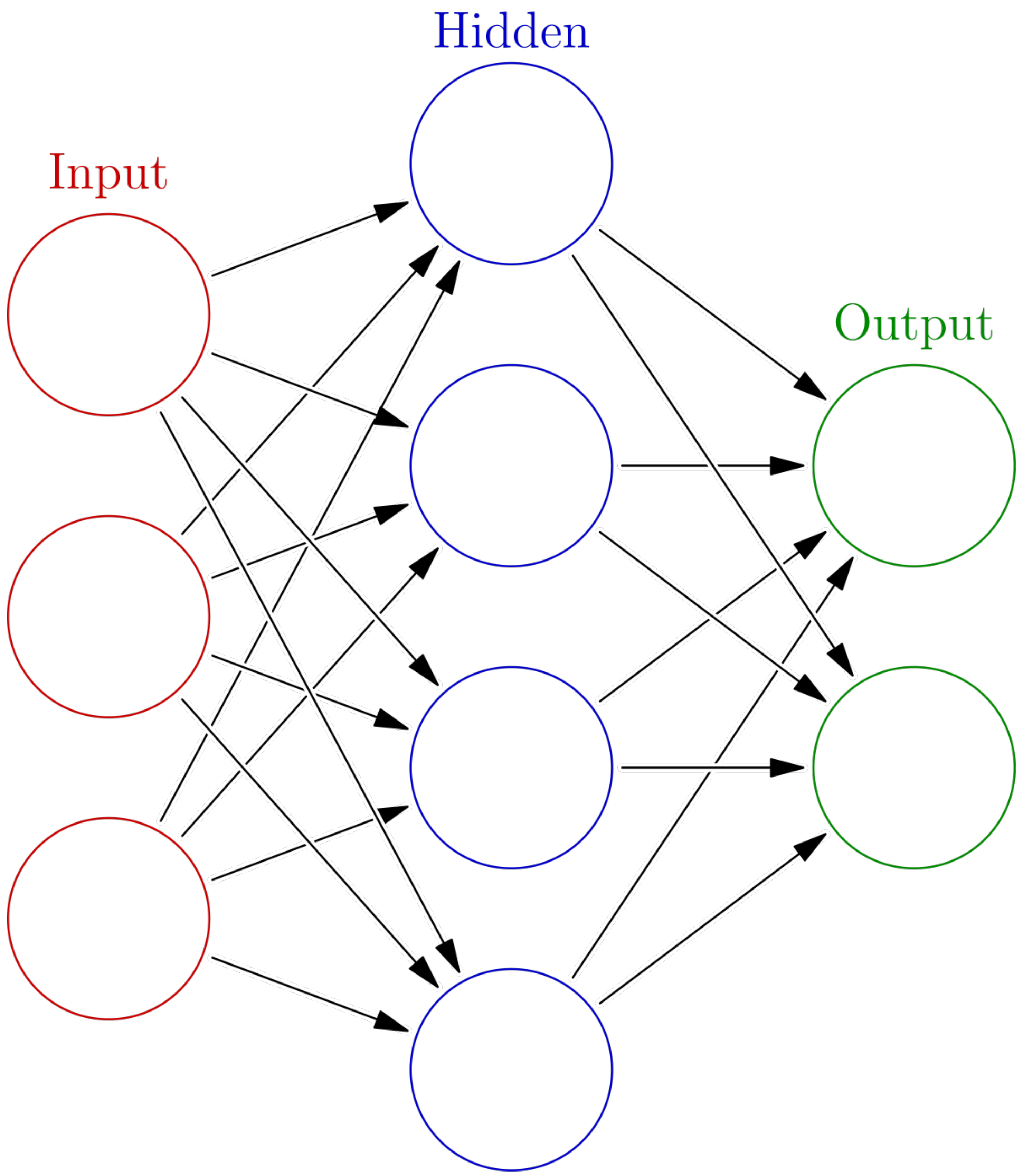
Logistic regression



Support vector machines



k-nearest neighbors



Neural networks

Naive Bayes

Naive Bayes Classifier
General setting

- Let the input x be represented as r features: $f_j,\ 1 \leq j \leq r$

- Let y be the output classification

- We can have a simple classification model using Bayes rule

$$\underbrace{P(y|x)}_{\text{Posterior}} = \frac{\overbrace{P(y)}^{\text{Prior}} \cdot \overbrace{P(x|y)}^{\text{Likelihood}}}{\underbrace{P(x)}_{\text{Evidence}}}$$

- Make strong (naive) conditional independence assumptions

$$P(x|y) = \prod_{j=1}^{r} P(f_j|y) \xrightarrow{\text{Bayes rule}} P(y|x) \propto P(y) \cdot \prod_{j=1}^{r} P(f_j|y)$$

Naive Bayes classifier for text classification

- For text classification: input x is document $d = (w_1, \ldots, w_k)$
- Use as our features the words w_j, $1 \leq j \leq |V|$ where V is our vocabulary
- c is the output classification
- Predicting the best class:

$$c_{MAP} = \arg\max_{c \in C} P(c \mid d)$$

maximum a posteriori (MAP) estimate

$$= \arg\max_{c \in C} \frac{P(c)P(d \mid c)}{P(d)}$$

$$= \arg\max_{c \in C} P(c)P(d \mid c)$$

$P(d \mid c) \rightarrow$ Conditional probability of generating document d from class c

$P(c) \rightarrow$ Prior probability of class c

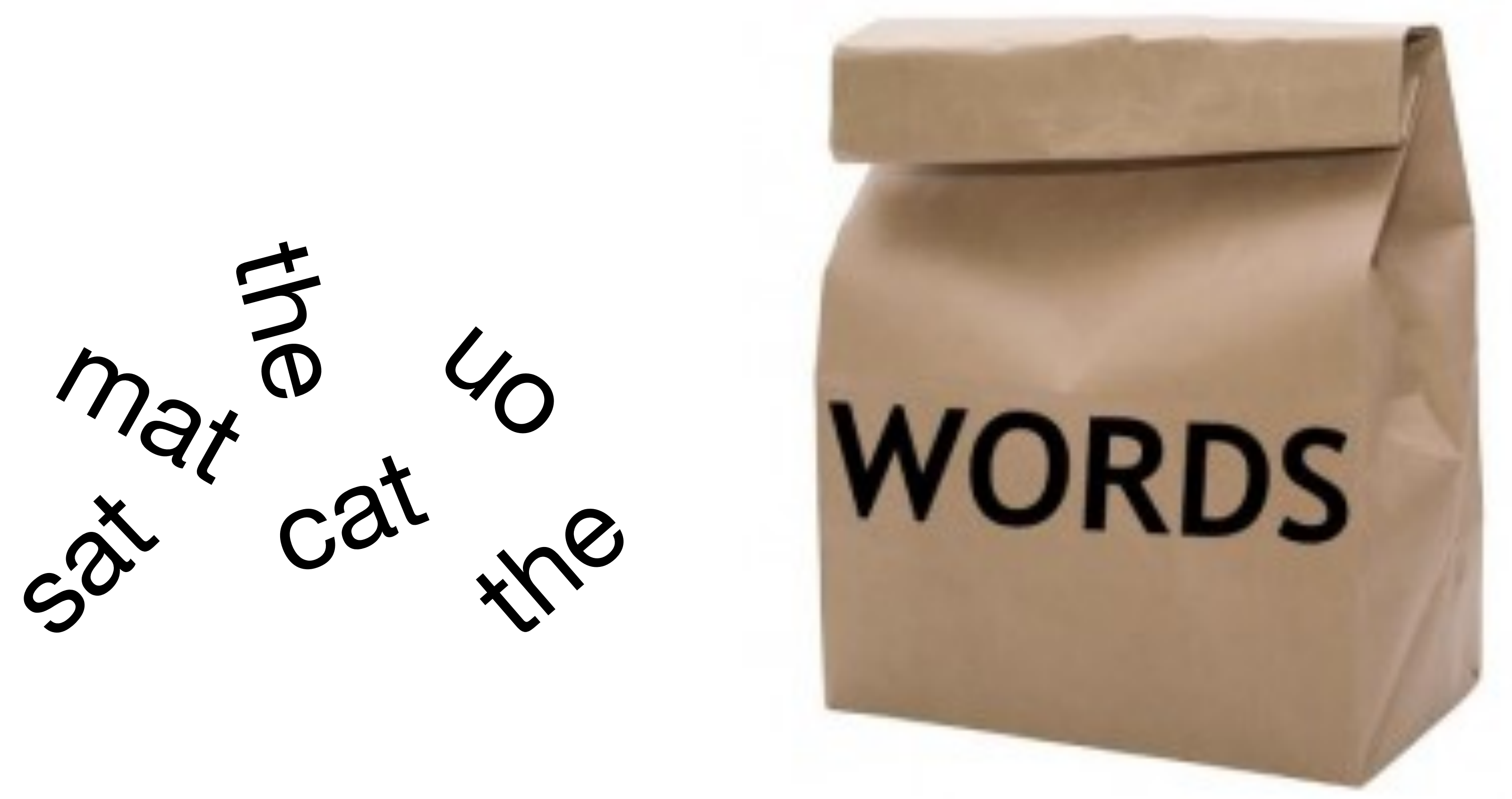
Represent P(d | c) as Bag of Words model

- Assume position of each word is irrelevant (both absolute and relative)

Order doesn't matter

- $P(w_1, w_2, w_3, \ldots, w_k | c) = P(w_1 | c)P(w_2 | c)\ldots P(w_k | c)$

- Probability of each word is *conditionally independent* given class c



Predicting with Naive Bayes

- Once we assume that the position of each word is irrelevant and that the words are conditionally independent given class c, we have:

$$P(d \mid c) = P(w_1, w_2, w_3, \ldots, w_k \mid c) = P(w_1 \mid c)P(w_2 \mid c)\ldots P(w_k \mid c)$$

- The maximum a posteriori (MAP) estimate is now:

$\hat{P}$ is used to indicate the estimated probability

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

Note that k is the number of tokens (words) in the document.

The index i is the position of the token.

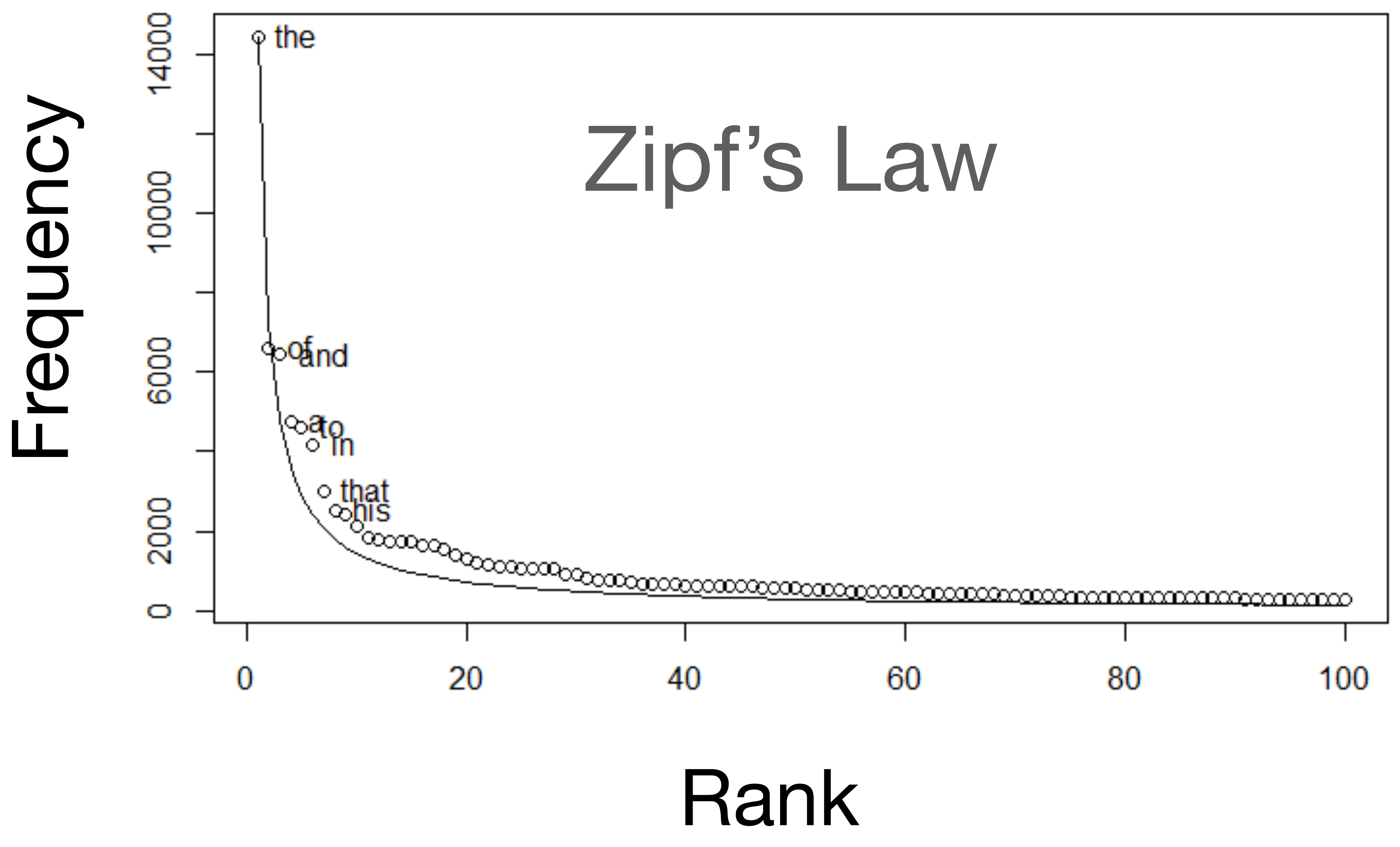
Maximum likelihood estimate

- Count and take average:

$$\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}$$

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

Can suffer from sparsity issues!



Solution: Smoothing!

- Maximum likelihood estimate

$$\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}$$

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

- Smoothing

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

Laplace smoothing

- 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 $$\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}$$

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

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

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

Variants

Name based on the distribution of the features

$$P(f_i \,|\, y) \rightarrow P(w_i \,|\, c)$$

Multinomial Naive Bayes
Normal counts (0,1,2,…) for each document

$$\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}$$

$\boxed{\text{Binary Multinomial NB} \quad \text{Binarized counts (0/1) for each document}}$

Some work show this works better than full counts or the Multivariate Bernoulli NB

Multivariate Bernoulli NB
Estimate *P(w|c)* as fraction of documents of class *c* with word *w*

- Explicitly model *P(!w|c) = 1 - P(w|c)*

Variants

Name based on the distribution of the features $P(f_i | y) \rightarrow P(w_i | c)$

Multinomial Naive Bayes
Normal counts (0,1,2,…) for each document $\hat{P}(c_j) = \frac{\text{Count}(c_j)}{n}$

Binary Multinomial NB
Binarized counts (0/1) for each document

Multivariate Bernoulli NB
Estimate *P(w|c)* as fraction of documents of class *c* with word *w*

- Explicitly model *P(!w|c) = 1 - P(w|c)*

Naive Bayes Example

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

Smoothing with $\alpha = 1$

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

	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:

$P(c)=$

$P(j)=$

Choosing a class:

P(c|d5) $\propto$

P(j|d5) $\propto$

Conditional Probabilities:

P(Chinese|c) =

P(Tokyo|c) =

P(Japan|c) =

P(Chinese|j) =

P(Tokyo|j) =

P(Japan|j) =

(Credits: Dan Jurafsky)

Naive Bayes Example

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

Smoothing with $\alpha = 1$

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

	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	?

- Let's compute the priors: what is $\hat{P}(\mathrm{c})$ and $\hat{P}(\mathrm{j})$?

$$\hat{P}(\mathrm{c}) = \frac{3}{4}, \hat{P}(\mathrm{j}) = \frac{1}{4}$$

- Let's compute $\hat{P}(\mathrm{Japan} \mid \mathrm{c})$:

$$count(\mathrm{Japan}, \mathrm{c}) = 0 \qquad count(\mathrm{c}) = \sum_{w \in V} count(w, \mathrm{c}) = 8 \qquad |V| = 6$$

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

(Credits: Dan Jurafsky)

Naive Bayes Example

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

Smoothing with $\alpha = 1$

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

	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
≈ 0.0003

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

(Credits: Dan Jurafsky)

Some details

- Vocabulary is important
- Tokenization matters: it can affect your vocabulary
  - Tokenization = how you break your sentence up into tokens / words
  - Make sure you are consistent with your tokenization!
    - *aren't* aren't
      arent
      are n't
      aren t
    - Emails, URLs, phone numbers, dates, emoticons
  - Special multi-word tokens: NOT_happy
  - Modern NLP system use subword tokens (e.g. byte pair encoding)

Some details

- Vocabulary is important
- Tokenization matters: it can affect your vocabulary
  - Tokenization = how you break your sentence up into tokens / words
  - Make sure you are consistent with your tokenization!
- Handling unknown words in test not in your training vocabulary?
  - Remove them from your test document! Just ignore them.
- Handling stop words (common words like *a*, *the* that may not be useful)
  - Remove them from the training data!
  - In practice not that helpful, so use all words!

Better to use

- Modified counts (tf-idf) that down weighs frequent, unimportant words
- Better models!

Features

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

Top features for Spam detection

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

Properties of Naive Bayes

+ Simple baseline method
+ Works well for small data sizes
+ Optimal if the independence assumptions hold: if the assumed independence is correct, then it is the Bayes Optimal Classifier for the problem

- But not if the independence assumption is broken
- Does not handle rare classes well - will favour more common class
- Also need to design features

• Modern NLP: use pretrained word embeddings with neural networks

Generative vs Discriminative Models

- Naive Bayes is a Generative Model: It models $p(y|x) \propto p(y)p(x|y)$
- It models how the document is generated from words
- You can use this model to sample documents
- Next: Logistic Regression, a Discriminative model that models $p(y|x)$ directly.

Evaluation

Evaluation Metrics

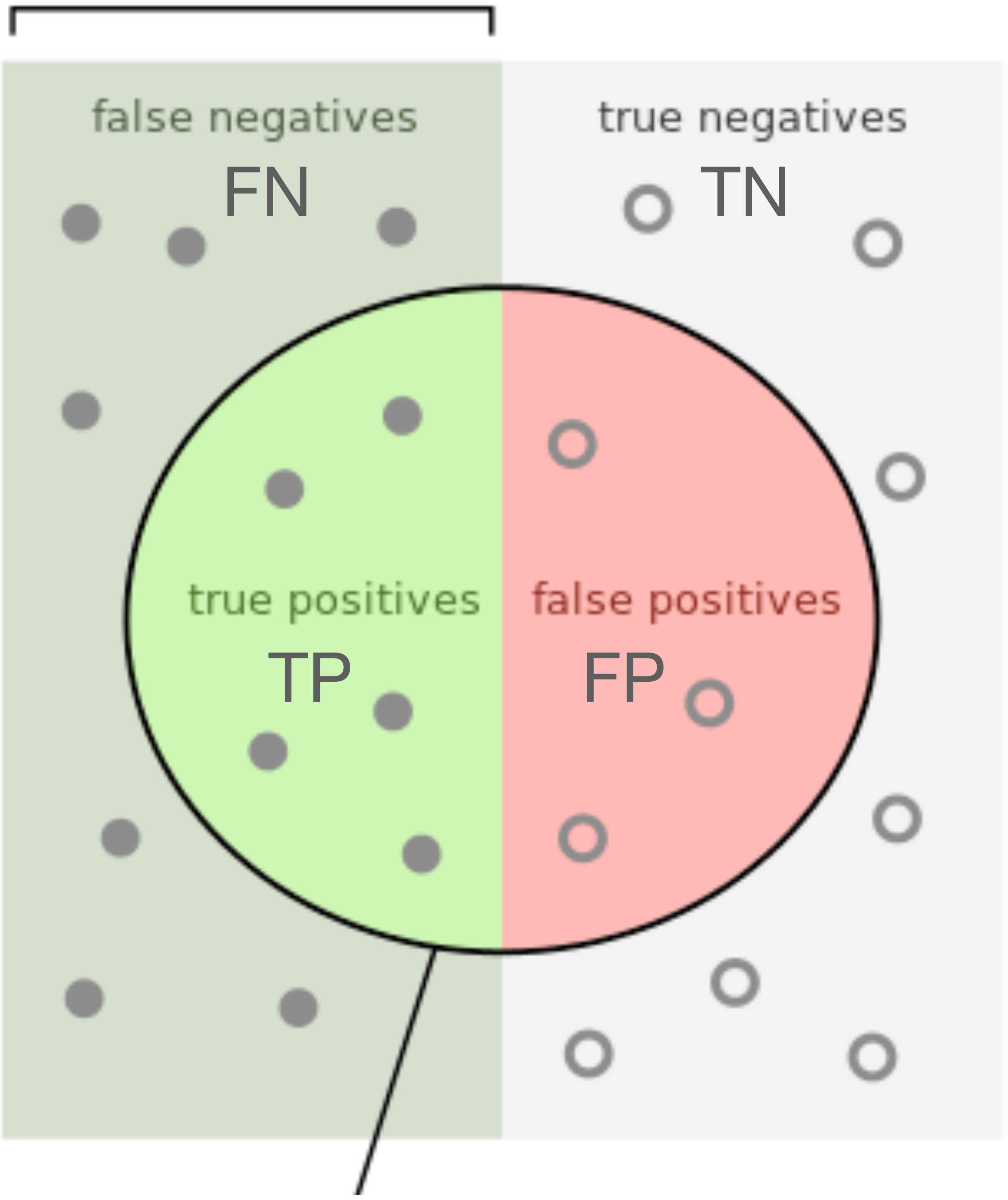
Confusion matrix

Truth

Predicted		Positive	Negative
	Positive	100 TP	5 FP
	Negative	45 FN	100 TN

- True positive (TP): Predicted + and actual +
- True negative (TN): Predicted - and actual -
- False positive (FP): Predicted + and actual -
- False negative (FN): Predicted - and actual +

Actual positives



Predicted positives

(image credit: wikipedia)

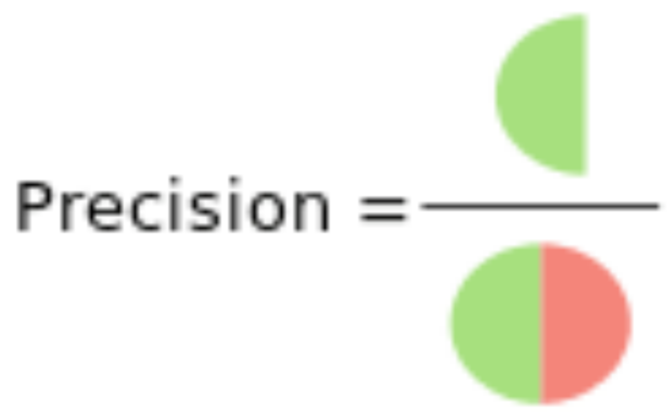
$$\text{Accuracy} = \frac{TP + TN}{Total} = \frac{200}{250} = 80\,\%$$

Coarse metric

Precision and Recall

- Precision: % of selected classes that are correct

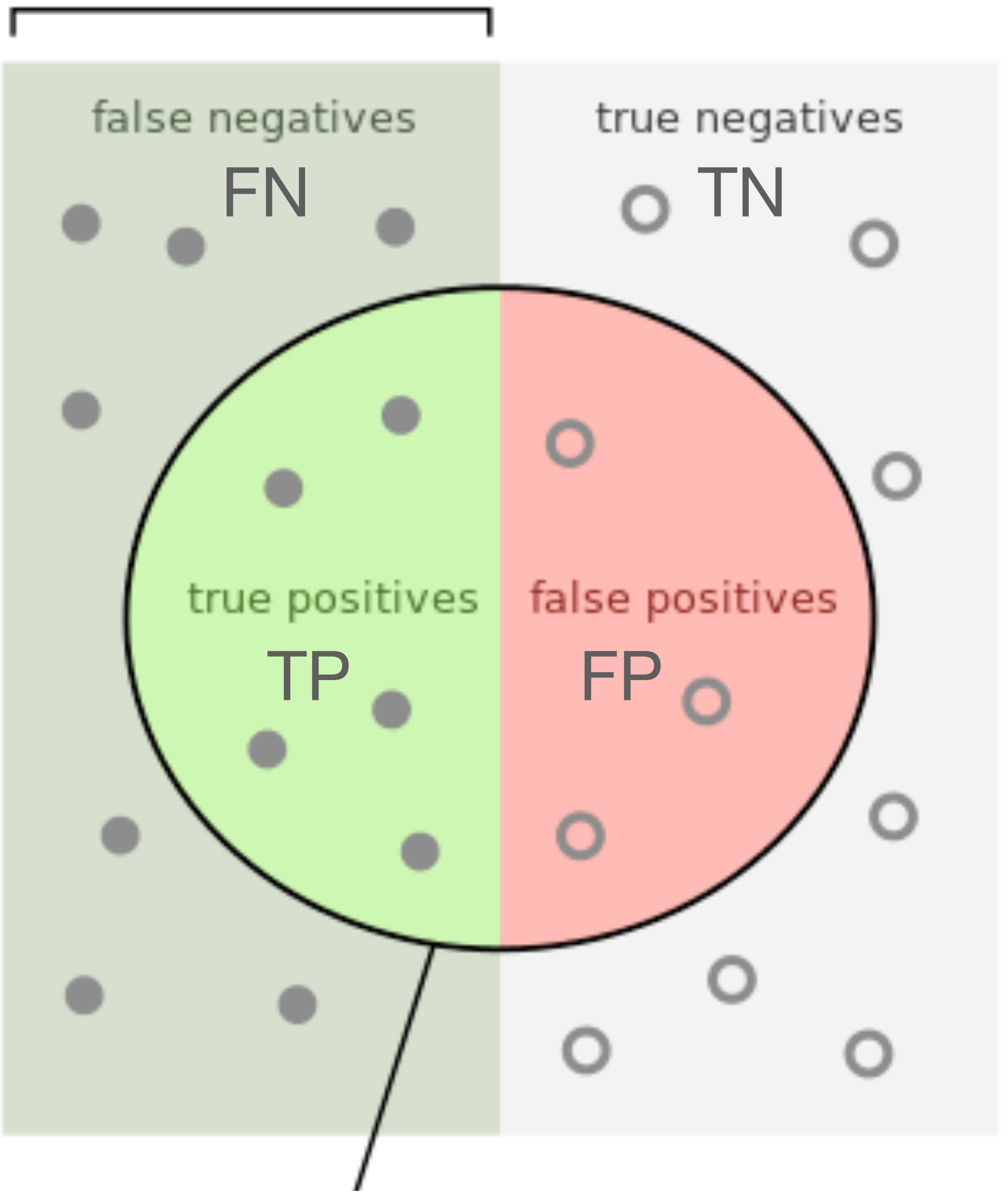
$$\text{Precision}(+) = \frac{TP}{TP + FP}$$



- Recall: % of correct items selected

$$\text{Recall}(+) = \frac{TP}{TP + FN}$$

Actual positives (relevant)



Predicted positives (selected/retrieved)

(image credit: wikipedia)

F-Score

- Combined measure

- Harmonic mean of Precision and Recall

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

- Or more generally,

Use β to control importance of Precision vs Recall

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

Aggregating scores

- How to handle more than 2 classes?

- We have Precision, Recall, F1 for each class

		gold labels urgent	normal	spam	
system output	urgent	8	10	1	**precision**$_u = \frac{8}{8+10+1}$
	normal	5	60	50	**precision**$_n = \frac{60}{5+60+50}$
	spam	3	30	200	**precision**$_s = \frac{200}{3+30+200}$
		recall$_u = \frac{8}{8+5+3}$	**recall**$_n = \frac{60}{10+60+30}$	**recall**$_s = \frac{200}{1+50+200}$	

(Credits: Dan Jurafsky)

Aggregating scores

- How to handle more than 2 classes?
- 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

- Micro-averaged score is dominated by score on common classes

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

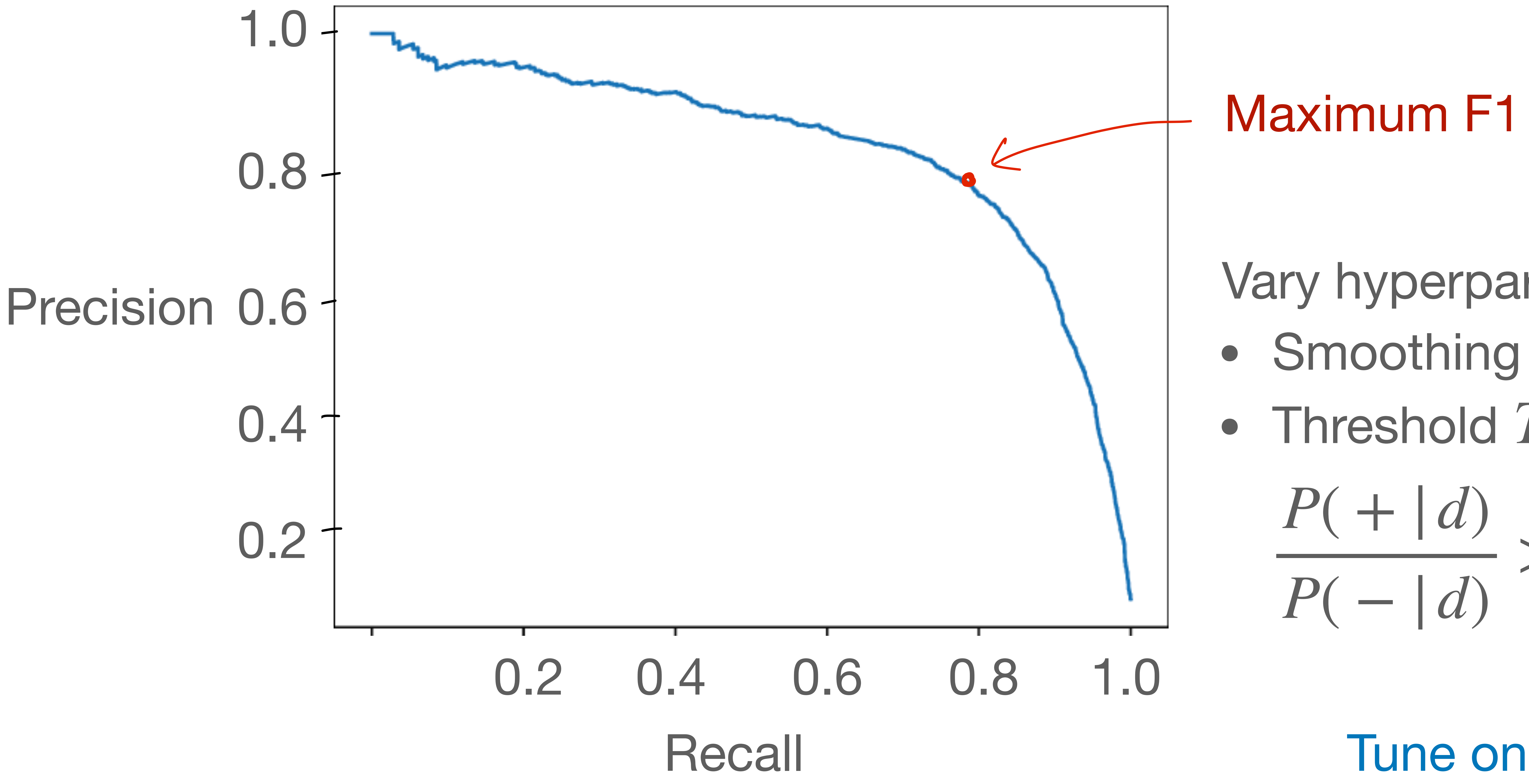
	true yes	true no
system yes	268	99
system no	99	635

$$\begin{array}{c}\text{microaverage}\\ \text{precision}\end{array} = \frac{268}{268+99} = \mathbf{.73}$$

$$\begin{array}{c}\text{macroaverage}\\ \text{precision}\end{array} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

(Credits: Dan Jurafsky)

Precision Recall tradeoff



Vary hyperparameters

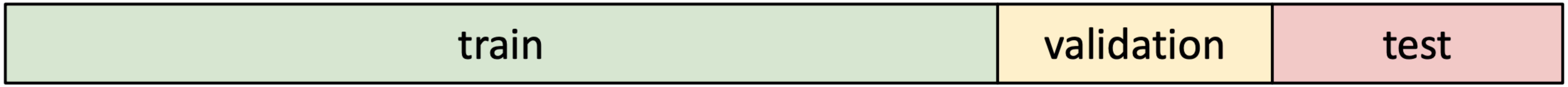
- Smoothing α
- Threshold T

$$\frac{P(+\,|\,d)}{P(-\,|\,d)} > T$$

Tune on validation set

Train, val, test split

- Train model on training set
- Tune hyperparameters on validation set
- Evaluate performance on unseen test set



Why do we do this?

Summary

- Evaluation Metrics
  - Accuracy - coarse metric
  - Precision, Recall, F1 for each class
- Aggregated scores
  - Macro-average: Compute for each class, then average
  - Micro-average: Collect predictions for all classes and jointly evaluate (dominated by common classes)
- Precision-Recall curve: pick threshold for maximum F1
  - Use validation set to tune hyperparameters, test set should remain “unseen”