ANLP

06 - binary classification (classf., part I)

David Schlangen
University of Potsdam, MSc Cognitive Systems
Winter 2019 / 2020

recap

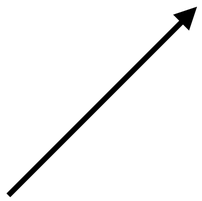
Process from our game

- Assume that X_t follows some given PD

$$P(X_t = w_t \mid X_1 = w_1, \ldots, X_{t-1} = w_{t-1})$$

- Then probability of the entire corpus (or sentence) $w = w_1 \ldots w_n$ is joint probability

$$P(w_1 \ldots w_n) = P(w_1) \cdot P(w_2 \mid w_1) \cdot P(w_3 \mid w_1, w_2) \cdot \ldots \cdot P(w_n \mid w_1, \ldots, w_{n-1})$$



How do we estimate these?

Independence assumptions

- Let's pretend that word at position t depends only on the words at positions t-1, t-2, …, t-k for some fixed k (*Markov assumption* of degree k).

- Then we get an n-gram model, with n = k+1:

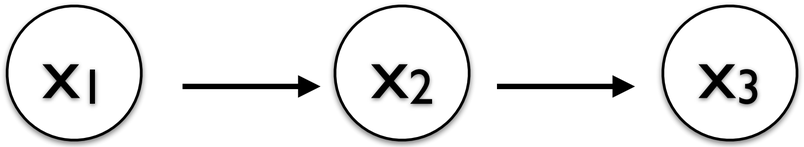
$$P(X_t \mid X_1, \ldots, X_{t-1}) = P(X_t \mid X_{t-k}, \ldots, X_{t-1})$$

  for all t.

- Special names for *unigram models* (n = 1), *bigram models* (n = 2), *trigram models* (n = 3).
  - Thus our second game was a bigram model.

graphical notation

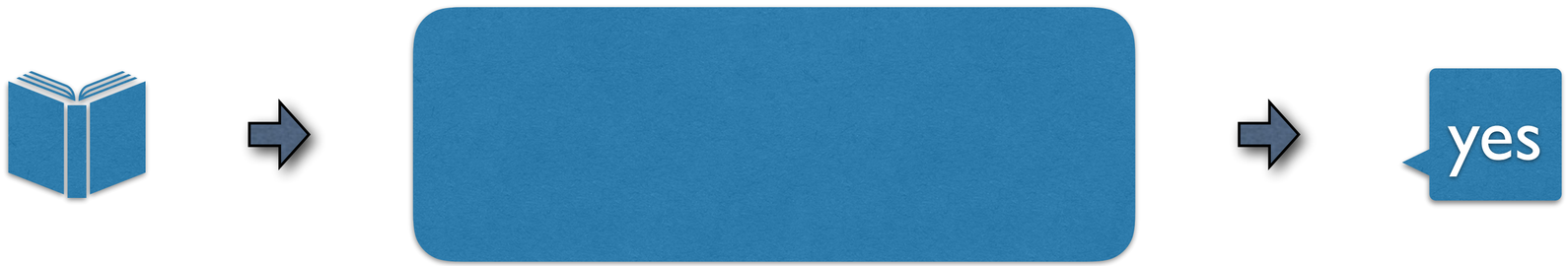
- it is common to draw the factorisation as follows (for bigram model)



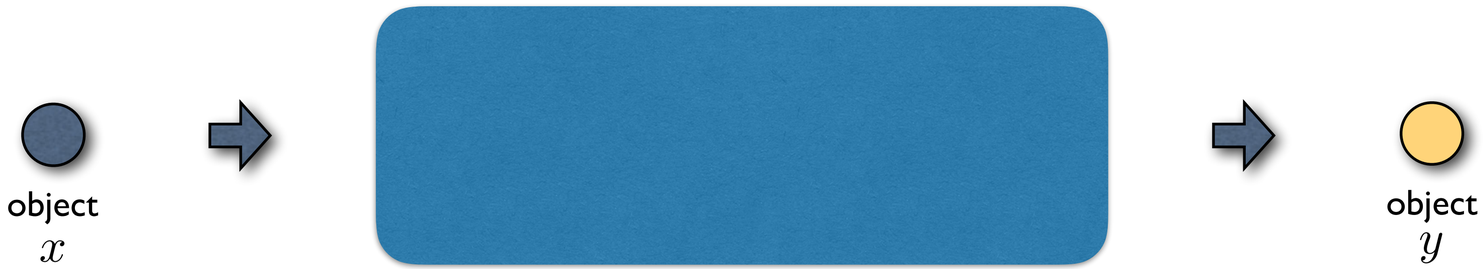
today

- classification: taking linguistic object and assigning a label to it
- today: binary classification, label from 0/1, no/yes, -/+, …

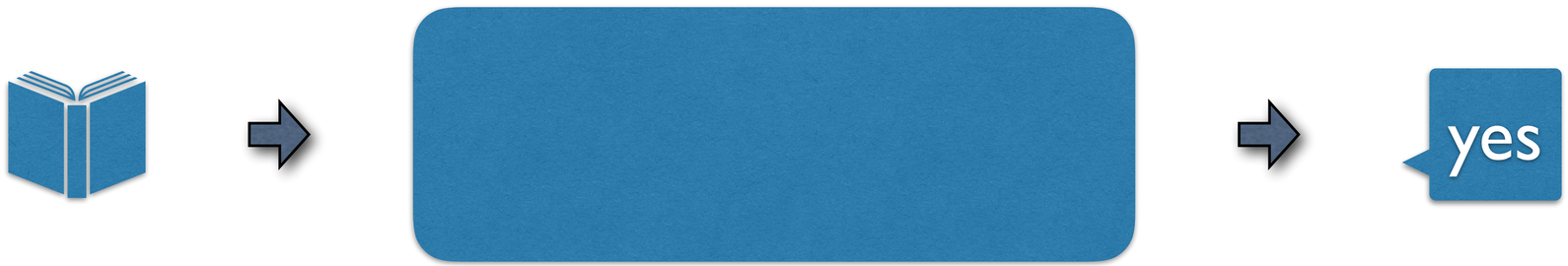
classification



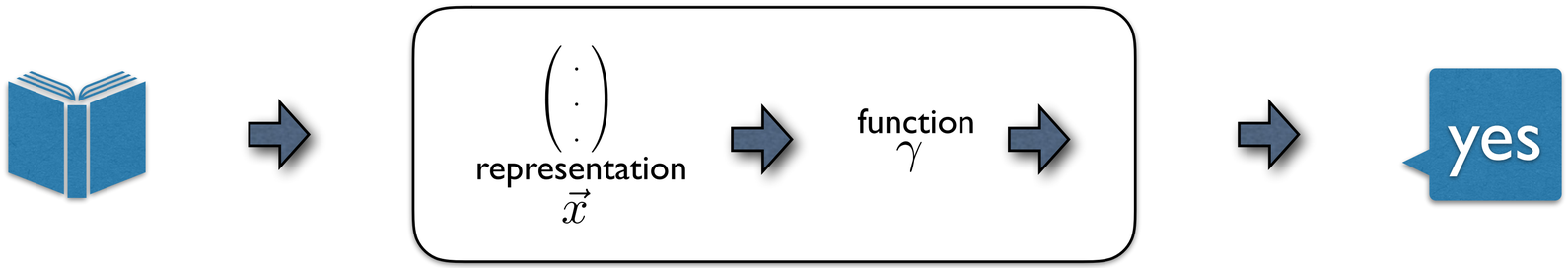
classification



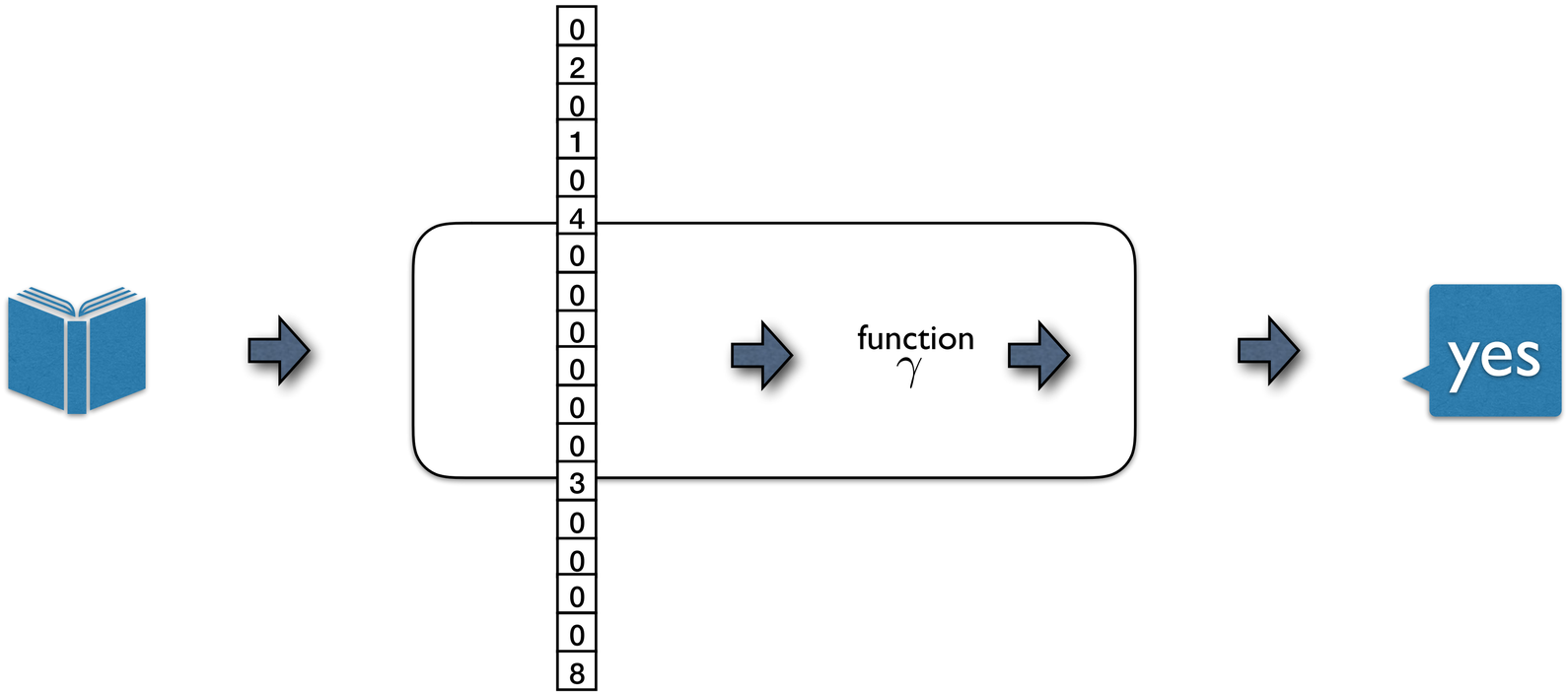
classification



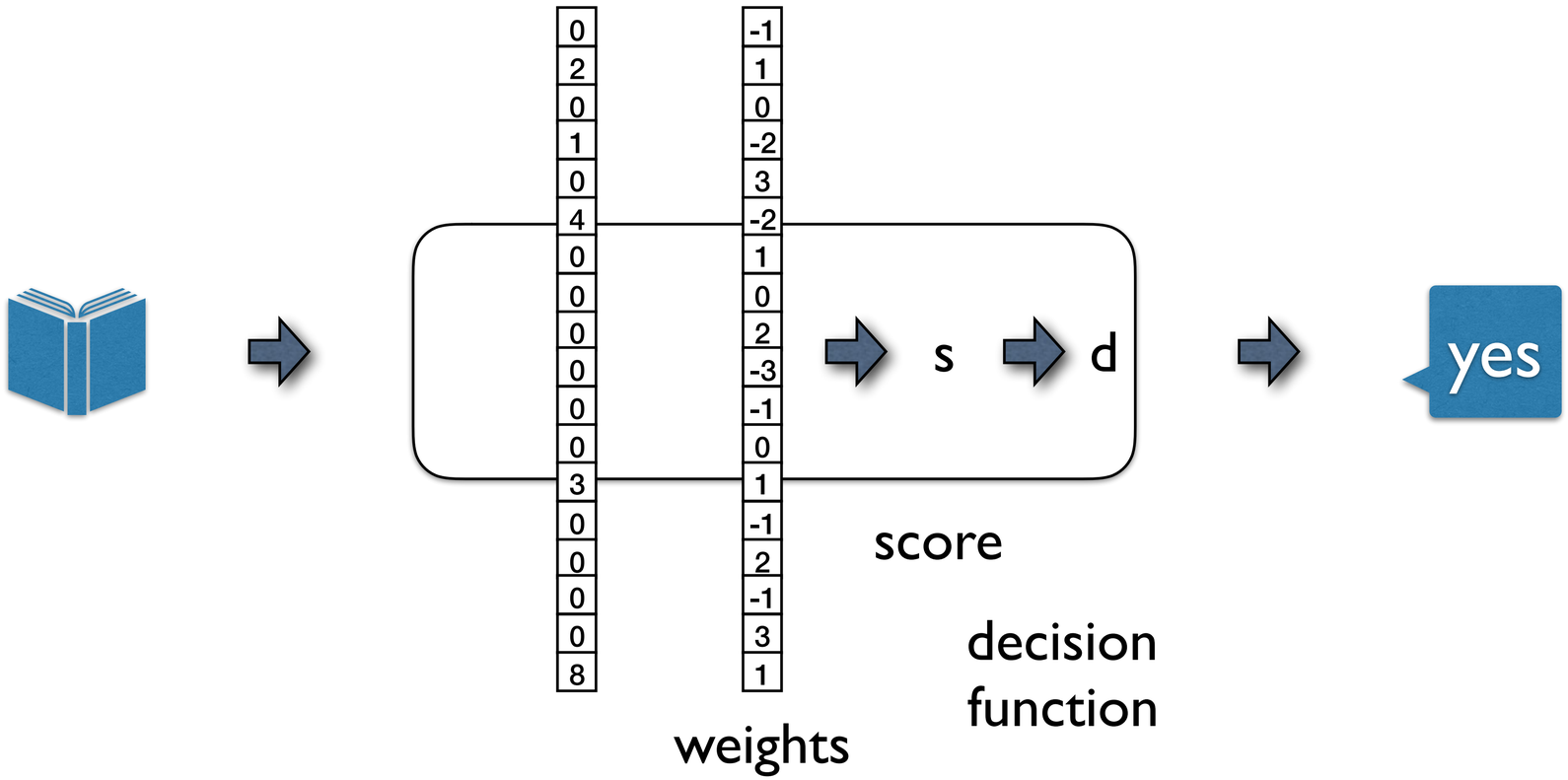
classification



classification



classification



terminology

- labelled data $\{(\mathbf{x}^{(i)}, y^{(i)})\}, 1 \leq i \leq N \quad y^{(i)} \in \mathcal{Y}, K = |\mathcal{Y}|$
- representation of instance; feature function $\boldsymbol{f}(\mathbf{x}, y)$
- scoring function $\Psi(\mathbf{x}, y; \theta) \quad \Psi(\boldsymbol{f}(\mathbf{x}, y), y; \theta)$
- decision function: typically, argmax (= the label that gets the best score)

Classification

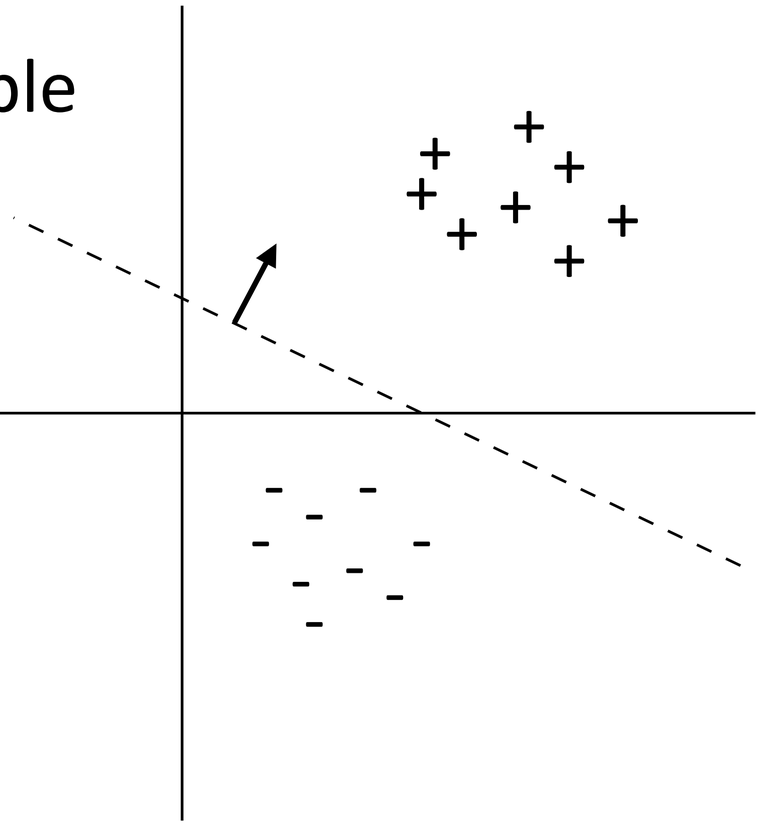
Linear Classification

- Datapoint x with label $y \in \{0, 1\}$

- Embed datapoint in a feature space $f(x) \in \mathbb{R}^n$
  but in this lecture $f(x)$ and x are interchangeable

- Linear decision rule: $w^\top f(x) + b > 0$

$$w^\top f(x) > 0$$



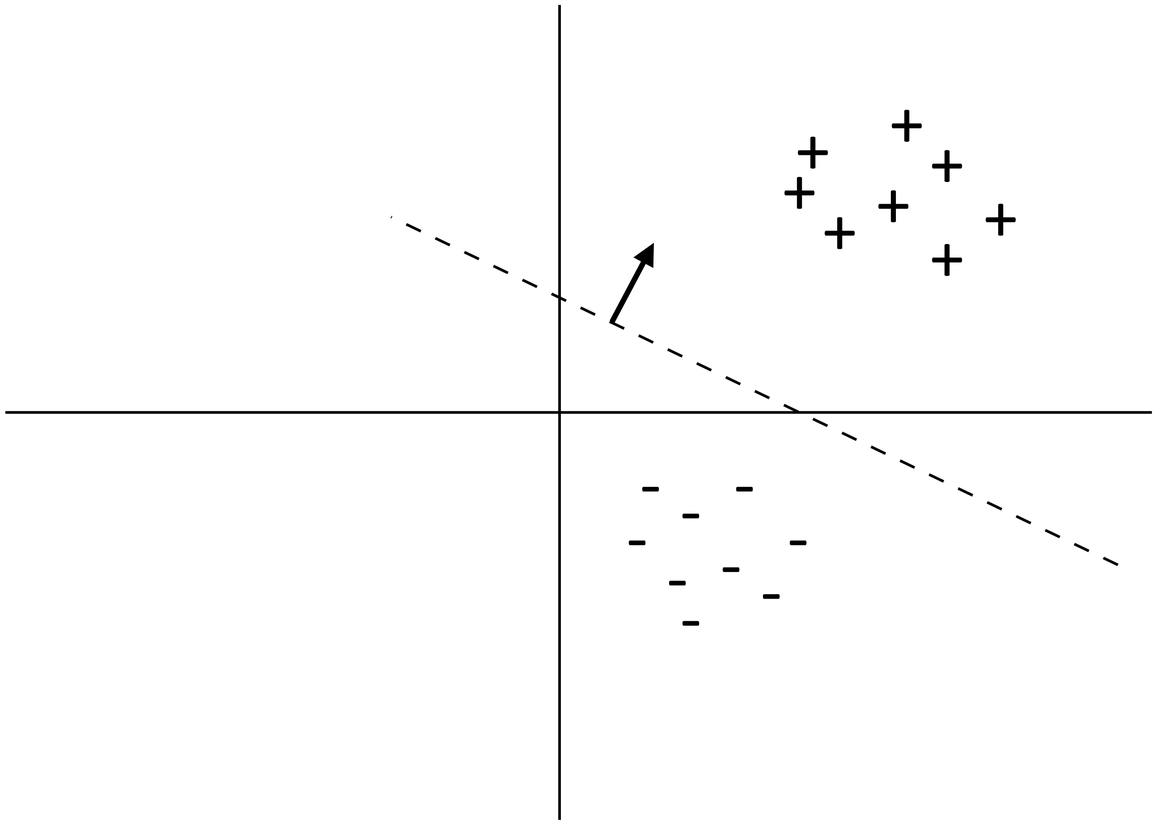
- Can delete bias if we augment feature space:

$f(x)$ = [0.5, 1.6, 0.3]

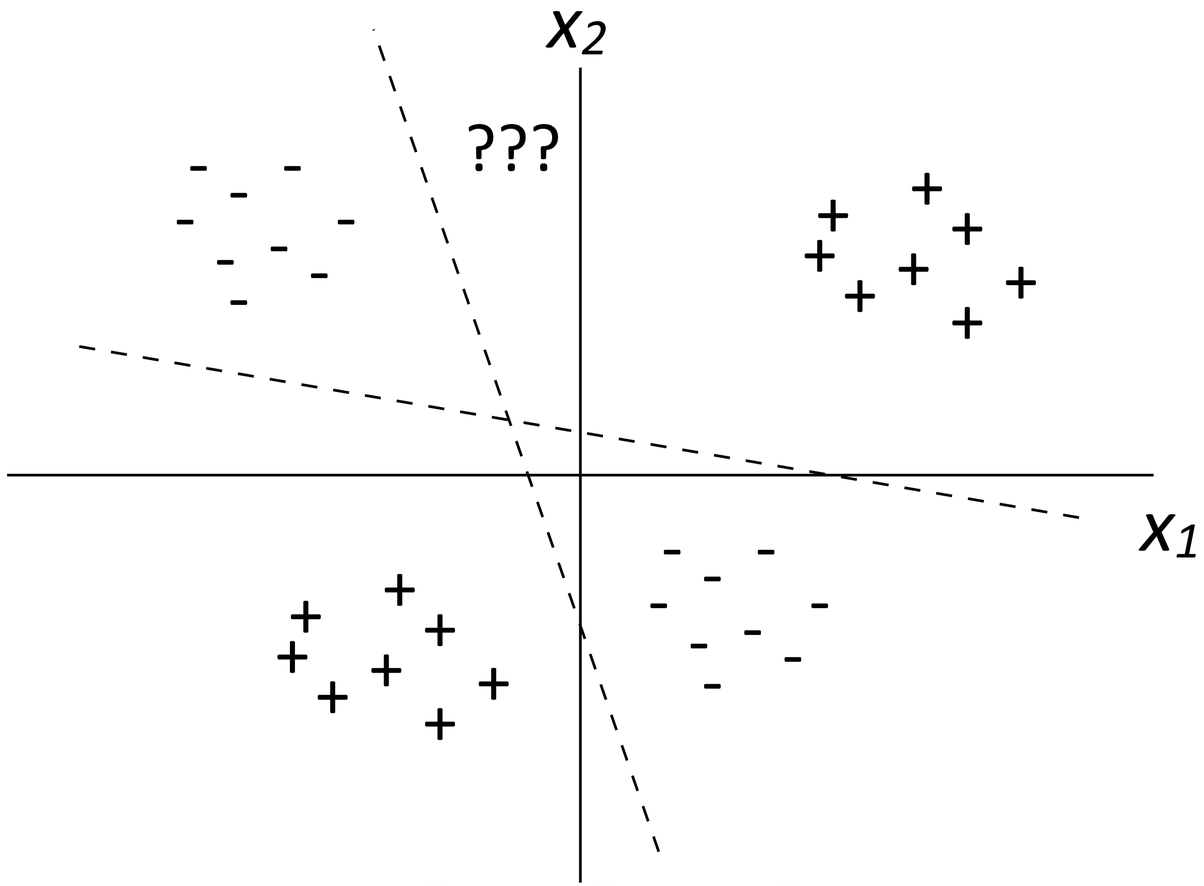
↓

[0.5, 1.6, 0.3, **1**]

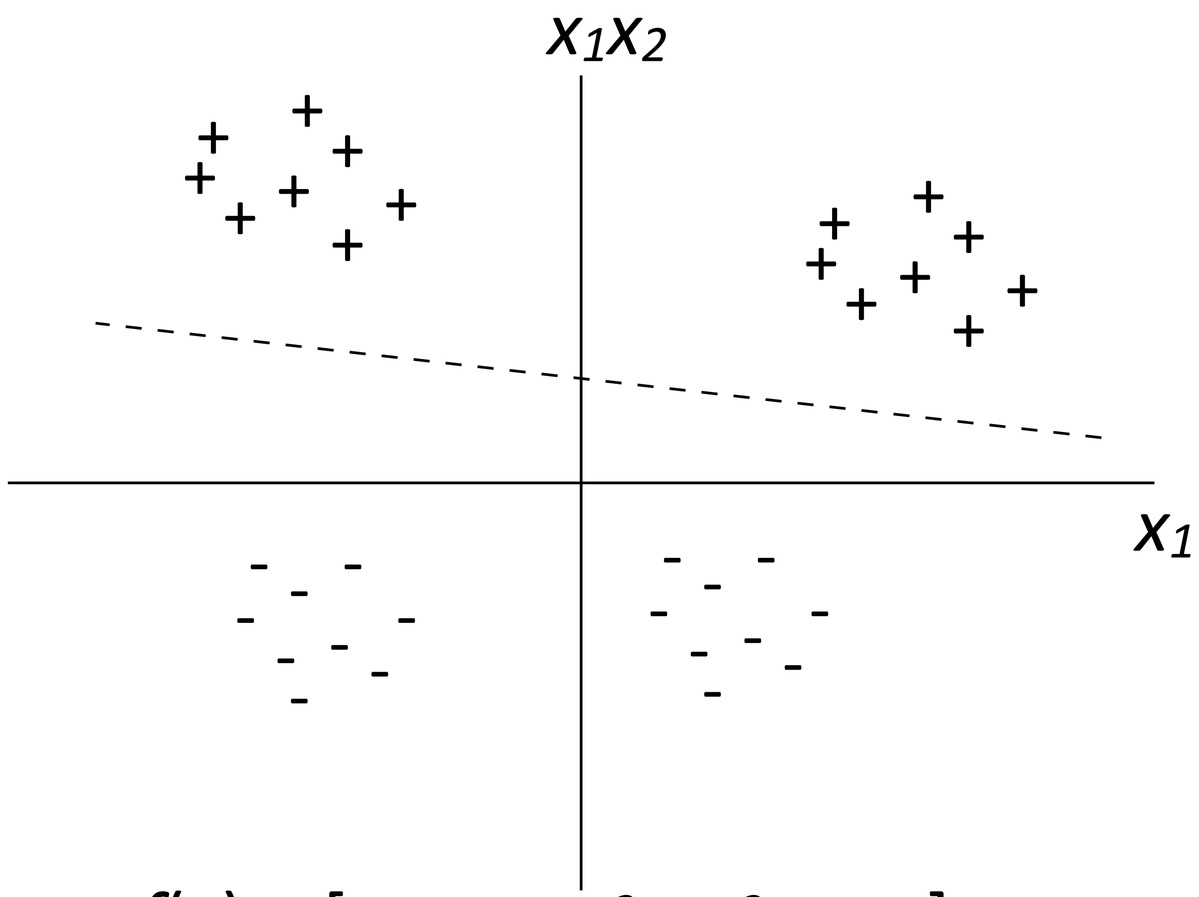
Linearly separable



Linearly separable



$f(x) = [x_1, x_2]$



$f(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2]$

- "Kernel trick" does this for "free," but is too expensive to use in NLP applications, training is $O(n^2)$ instead of $O(n \cdot (\text{num feats}))$

Classification: Sentiment Analysis

this movie was great! would watch again Positive

that film was awful, I'll never watch again Negative

- Surface cues can basically tell you what's going on here: presence or absence of certain words (*great*, *awful*)
- Steps to classification:
  - Turn examples like this into feature vectors
  - Pick a model / learning algorithm
  - Train weights on data to get our classifier

Feature Representation

this movie was great! would watch again Positive

- Convert this example to a vector using *bag-of-words features*

[contains *the*]	[contains *a*]	[contains *was*]	[contains *movie*]	[contains *film*]	…
position 0	position 1	position 2	position 3	position 4	
$f(x)$ = [0	0	1	1	0	…

- Very large vector space (size of vocabulary), sparse features (how many?)
- Requires *indexing* the features (mapping them to axes)
- More sophisticated feature mappings possible (tf-idf), as well as lots of other features: n-grams, character n-grams, parts of speech, lemmas, …

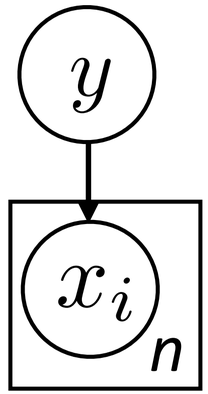
Generative vs. Discriminative Modeling

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Generative models: probabilistic models of P(x,y)
  - Compute $P(y|x)$, predict $\text{argmax}_y P(y|x)$ to classify
  - Examples: Naive Bayes (comes next), Hidden Markov Models
- Discriminative models model P(y|x) directly, compute $\text{argmax}_y P(y|x)$
  - Examples: logistic regression
  - Cannot draw samples of *x*, but typically better classifiers

Naive Bayes, the generative story

- to create document (bag of words), first
  - sample label y (= decide on positive / negative)
  - then sample the word counts from P(x | y)

- inference: for which value of y does this story fit the observation?



Naive Bayes

- Data point $x = (x_1, ..., x_n)$, label $y \in \{0, 1\}$
- Formulate a probabilistic model that places a distribution $P(x, y)$
- Compute $P(y|x)$, predict $\text{argmax}_y P(y|x)$ to classify

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

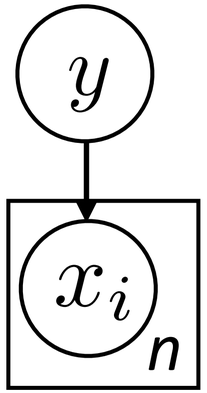
Bayes' Rule

constant: irrelevant for finding the max

$$\propto P(y)P(x|y)$$

"Naive" assumption:

$$= P(y)\prod_{i=1}^{n} P(x_i|y)$$



linear model!

$$\text{argmax}_y P(y|x) = \text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y)\right]$$

Naive Bayes Example

it was great $\longrightarrow$ $P(y|x) \propto \left[\quad \right]$

Need to know prior P(y)

and likelihood of word, given the class

$$P(y|x) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

$$\text{argmax}_y \log P(y|x) = \text{argmax}_y \left[\log P(y) + \sum_{i=1}^{n} \log P(x_i|y) \right]$$

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (*j* indexes over examples)
- Find values of $P(y),\ P(x_i|y)$ that maximize data likelihood (generative):

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji}|y_j) \right]$$

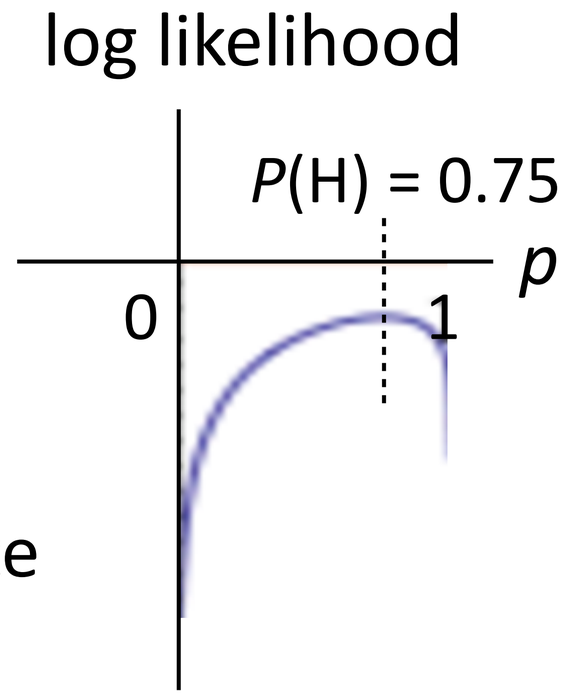
data points (*j*) features (*i*) *i*th feature of *j*th example

Maximum Likelihood Estimation

again...

- Imagine a coin flip which is heads with probability *p*
- Observe (H, H, H, T) and maximize likelihood: $\prod_{j=1}^{m} P(y_j) = p^3(1-p)$
- Easier: maximize *log* likelihood

$$\sum_{j=1}^{m} \log P(y_j) = 3 \log p + \log(1-p)$$



- Maximum likelihood parameters for binomial/ multinomial = read counts off of the data + normalize

Maximum Likelihood Estimation

- Data points (x_j, y_j) provided (*j* indexes over examples)
- Find values of $P(y),\ P(x_i|y)$ that maximize data likelihood (generative):

$$\prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji}|y_j) \right]$$

data points (*j*) features (*i*) *i*th feature of *j*th example

- Equivalent to maximizing logarithm of data likelihood:

$$\sum_{j=1}^{m} \log P(y_j, x_j) = \sum_{j=1}^{m} \left[\log P(y_j) + \sum_{i=1}^{n} \log P(x_{ji}|y_j) \right]$$

- Can do this by counting and normalizing distributions!

Problem with MLE

- what if we have never seen the word „fantastic“ in the positive class?

$$P(y|x) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

- we will be *certain* that test case is not positive, no matter what else is in document!
- same problem as with n-gram models: can’t trust information from not having seen something
- same type of solution: smoothing (e.g., add 1 smoothing)

Problems with Naive Bayes

- In reality, features are correlated

$$P(x_{\text{funds}} = 1|\text{spam}) = 0.1$$
$$P(x_{\text{funds}} = 1|\text{ham}) = 0.01$$
$$P(x_{\text{transfer}} = 1|\text{spam}) = 0.1$$
$$P(x_{\text{transfer}} = 1|\text{ham}) = 0.01$$

- This one sentence will make the probability of spam very high!

Hi, in order to close on the house we need you to transfer the requested funds to the escrow account.

Ham

Spam

???

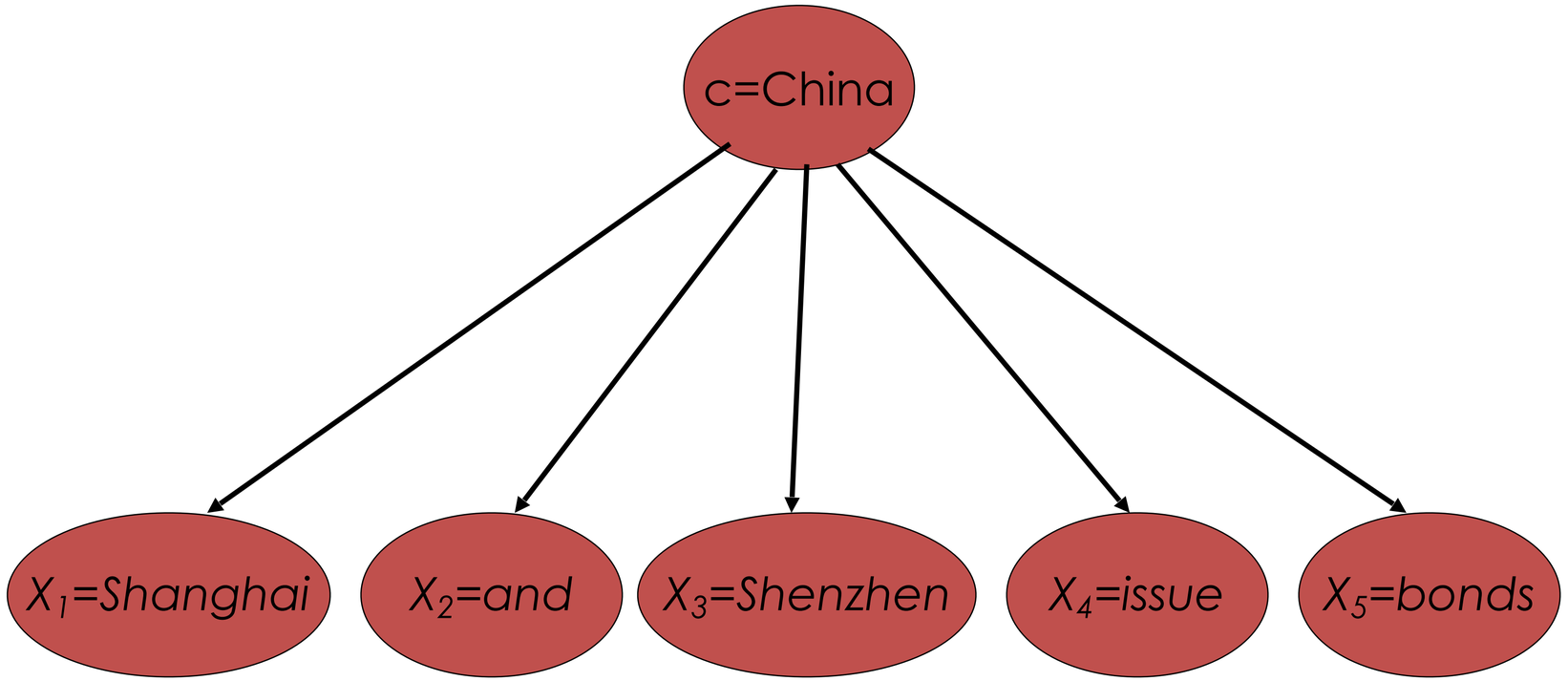
Ham

- Bad independence assumption in NB: these words are not independent!

Problems with Naive Bayes

- In reality, features are correlated
- Bad independence assumption in NB: these words are not independent!
- We may even want to create features that definitely aren't independent (words + bi-grams; word prefixes; etc.)
- Solution: better model, algorithms that explicitly minimize loss rather than maximizing data likelihood

Generative Model for Multinomial Naïve Bayes



Naïve Bayes and Language Modeling

- Naïve Bayes classifiers can use any sort of feature
  - URL, email address, dictionaries, network features
- But if, as in the previous slides
  - We use **only** word features
  - we use **all** of the words in the text (not a subset)
- Then
  - Naïve bayes has an important similarity to language modeling.

Each class = a unigram language model

- Assigning each word: P(word | c)
- Assigning each sentence: P(s | c)=Π P(word | 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

Naïve Bayes as a Language Model

- Which class assigns the higher probability to s?

Model pos		Model neg	
0.1	I	0.2	I
0.1	love	0.001	love
0.01	this	0.01	this
0.05	fun	0.005	fun
0.1	film	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

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

Generative vs. Discriminative Models

- Generative models: $P(x, y)$
  - Bayes nets / graphical models / Naive Bayes
  - Some of the model capacity goes to explaining the distribution of *x*; prediction uses Bayes rule post-hoc
- Discriminative models: $P(y|x)$
  - SVMs, logistic regression, CRFs, most neural networks
  - Model is trained to be good at prediction, but doesn't model *x*
- Up next: discriminative classifiers
- We'll come back to this distinction throughout this class

Classification Evaluation

Precision, Recall, F-Measure

Accuracy

- simplest evaluation: percent correctly assigned
- But what if there is only 1 spam message in 100 messages?

The 2-by-2 contingency table

	correct	not correct
selected	tp	fp
not selected	fn	tn

Precision and recall

- **Precision**: % of selected items that are correct
  Recall: % of correct items that are selected

	correct	not correct
selected	tp	fp
not selected	fn	tn

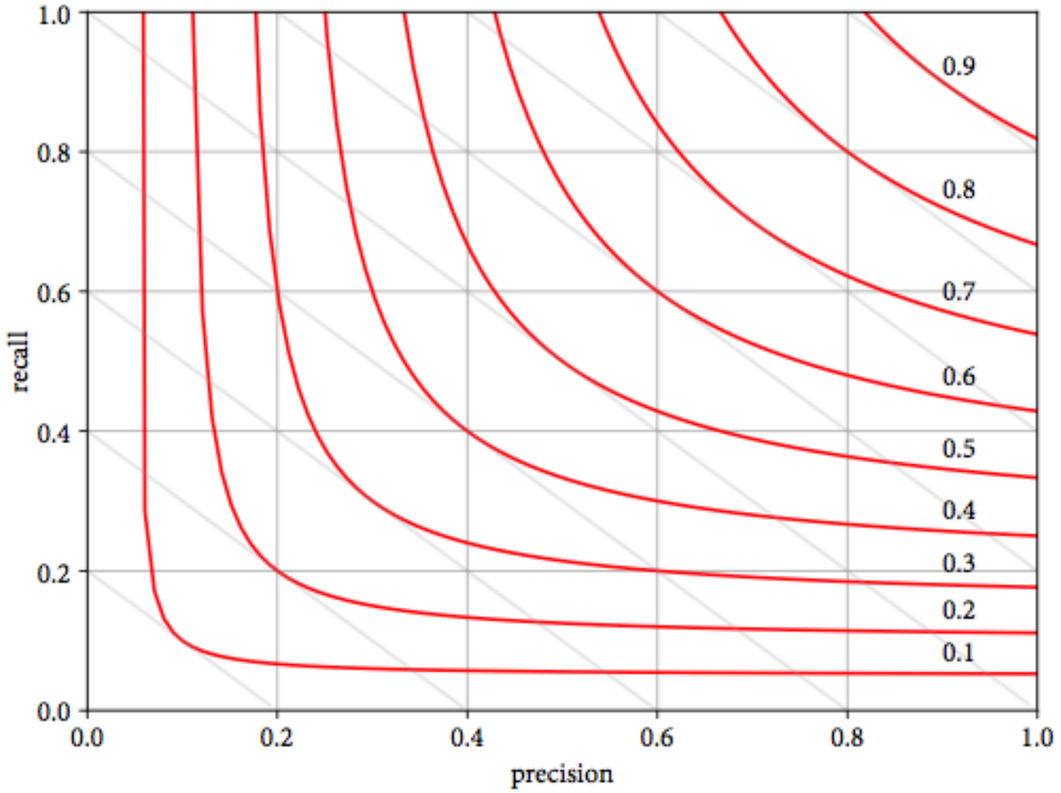
A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

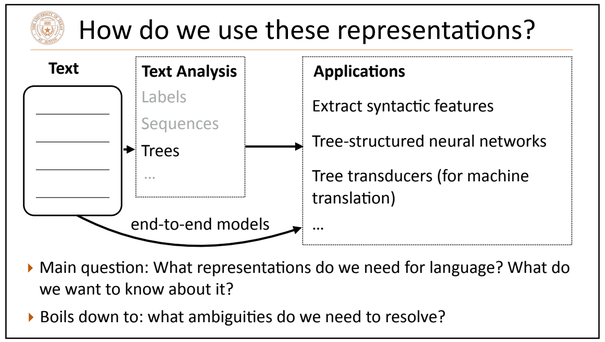
- The harmonic mean is a very conservative average
- People usually use balanced F1 measure
  - i.e., with $\beta = 1$ (that is, $\alpha = ½$): $F = 2PR/(P+R)$

F_1-measure


1.0
0.8
0.6
0.4
0.2
0.0
recall
0.0
0.2
0.4
0.6
0.8
1.0
precision
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1

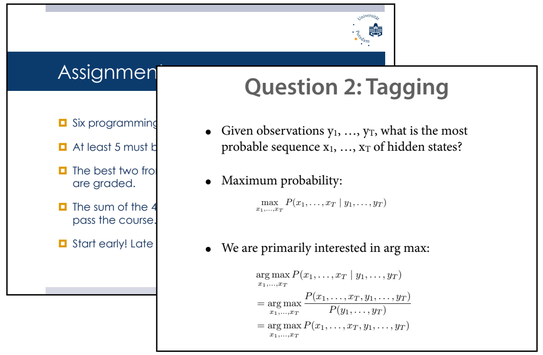
slide credits

slides that look like this



come from

CS388 given by Greg Durrett at U Texas, Austin



earlier editions of this class (ANLP), given by Tatjana Scheffler and Alexander Koller

and their use is gratefully acknowledged. I try to make any modifications obvious, but if there are errors on a slide, assume that I added them.