COMP 3361 Natural Language Processing

Lecture 4: Text Classification

Spring 2025

Many materials from CS224n@Stanford and COS484@Princeton with special thanks!

Announcements

- Assignment 1 was out. Due in 4 weeks.
    - TA will provide in-class coding tutorials to help you with each assignment
- Course enrollment: talk to me after class

Latest AI news

EXCLUSIVE TECHNOLOGY

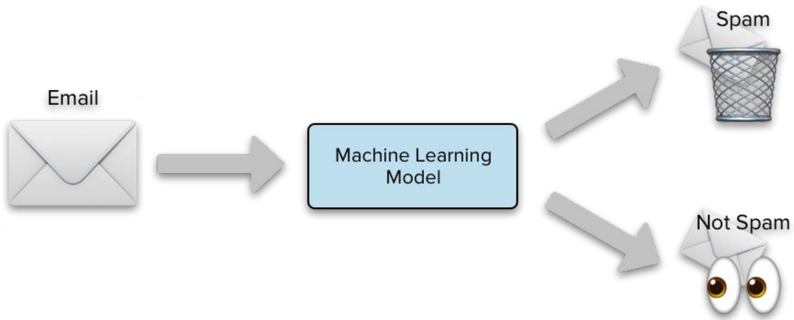
Elon Musk-Led Group Makes $97.4 Billion Bid for Control of OpenAI

Unsolicited offer complicates Sam Altman's plans to convert OpenAI to a for-profit company

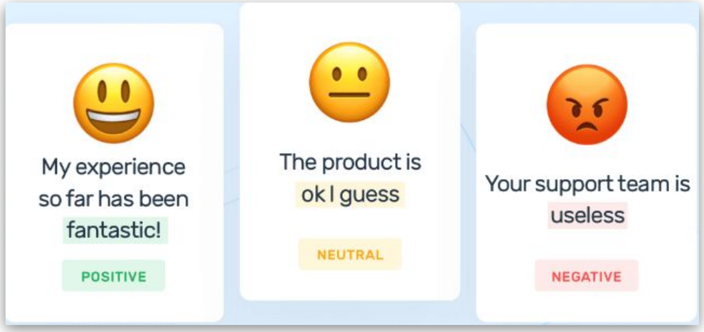
Lecture plan

- Recap of language modeling
- Naive Bayes and sentiment classification
- Logistic Regression for text classification

Why text classification?



Spam email detection



Sentiment analysis

Q: any other examples?

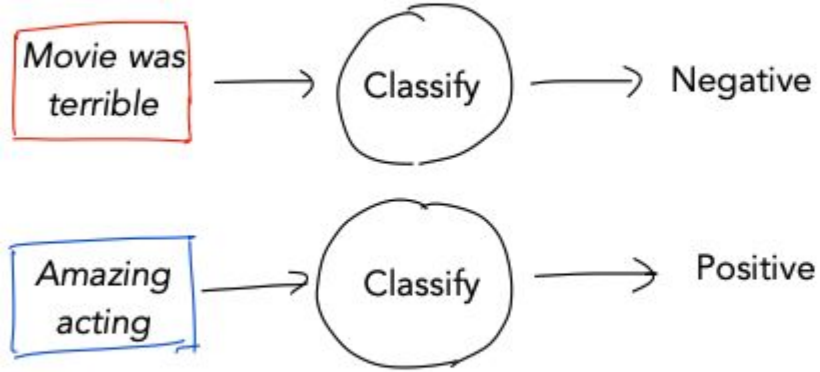
Text classification

Inputs:

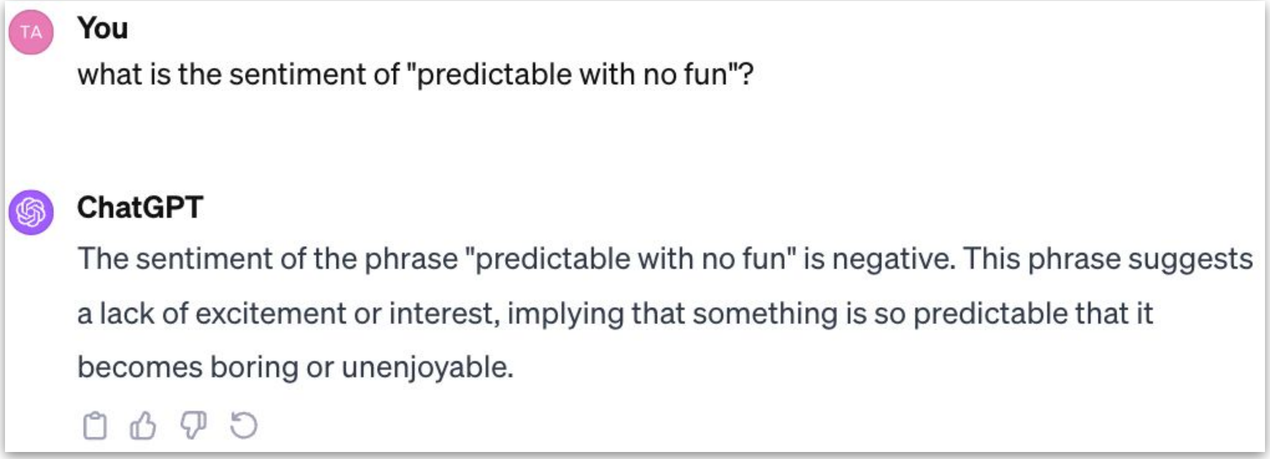
- A document d
- A set of classes C (m classes)

Output:

- Predicted class $c \in C$ for document d



Prompting ChatGPT for text classification



Prompting ChatGPT for text classification

You

what is the sentiment of "predictable with no fun"? just tell me: positive, negative, or neutral.

ChatGPT

Negative.

Rule-based text classification

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 (if rules carefully refined by expert)
- - Rules may be hard to define (and some even unknown to us!)
- - Labor intensive and expensive
- - Hard to generalize and keep up-to-date

Supervised Learning: Let's use statistics!

Let the machine figure out the best patterns using data

Inputs:

- Set of m classes C
- Set of n 'labeled' documents: $\{(d_1, c_1), (d_2, c_2), \ldots, (d_n, c_n)\}$, $d_i \in \mathcal{D}, c_i \in C$

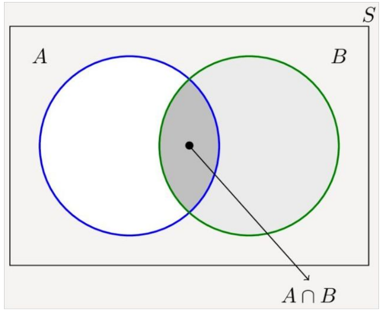
Output:

- Trained classifier, $F : \mathcal{D} \rightarrow C$

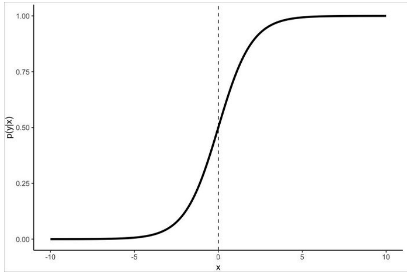
Key questions:

- What is the form of F?
- How do we learn F?

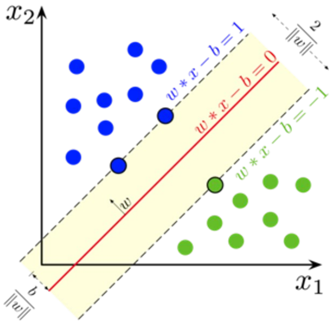
Types of supervised classifiers



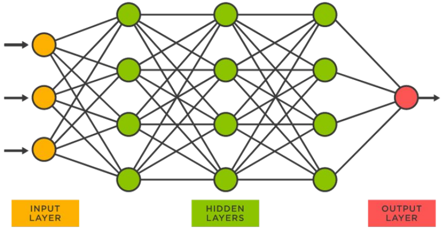
Naive Bayes



Logistic regression

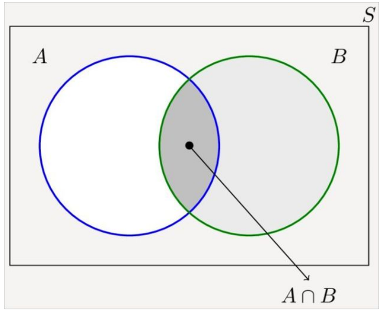


Support vector machines



Neural networks

Naive Bayes



Naive Bayes

Naive Bayes classifier

Simple classification model making use of Bayes rule

- Bayes rule:

d: document, c: class

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

Naive Bayes classifier

d: document, c: class

$$c_{\text{MAP}} = \text{argmax}_{c \in C} P(c \mid d)$$

MAP is "maximum a posteriori" estimate = most likely class

$$= \text{argmax}_{c \in C} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes' rule

$$= \text{argmax}_{c \in C} P(d \mid c)P(c)$$

Dropping the denominator

Naive Bayes classifier

d: document, c: class

$$c_{\mathrm{MAP}} = \operatorname{argmax}_{c \in C} P(c \mid d)$$

MAP is "maximum a posteriori" estimate = most likely class

$$= \operatorname{argmax}_{c \in C} \frac{P(d \mid c) P(c)}{P(d)}$$

Bayes' rule

$$= \operatorname{argmax}_{c \in C} P(d \mid c) P(c)$$

Dropping the denominator

$P(d \mid c)$: conditional probability of generating document d from class c

$P(c)$: prior probability of class c

How to represent $P(d \mid c)$?

- Option 1: represent the entire sequence of words
  - $P(w_1, w_2, \ldots, w_K | c)$ Too many sequences!

How to represent $P(d \mid c)$?

- Option 1: represent the entire sequence of words
  - $P(w_1, w_2, \ldots, w_K \mid c)$ Too many sequences!
- Option 2: Bag of words

$$P(w_1, w_2, \ldots, w_K \mid c) = P(w_1 \mid c) P(w_2 \mid c) \ldots P(w_K \mid c)$$

  - Assume position of each word doesn't matter
  - Probability of each word is *conditionally independent* of the other words given class c

Bag of words (BoW)

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:

$$
\begin{aligned}
c_{\mathrm{MAP}} &= \operatorname{argmax}_{c \in C} P(d \mid c) P(c) \\
&= \operatorname{argmax}_{c \in C} P(w_1, w_2, \ldots, w_K \mid c) P(c) \\
&= \operatorname{argmax}_{c \in C} P(c) \prod_{i=1}^{K} P(w_i \mid c)
\end{aligned}
$$

Equivalent as $c_{\mathrm{MAP}} = \operatorname{argmax}_{c \in C} \left(\log P(c) + \sum_{i=1}^{K} \log P(w_i \mid c) \right)$

How to estimate probabilities?

Given a set of n 'labeled' documents:

$\{(d_1, c_1), (d_2, c_2), \ldots, (d_n, c_n)\}$

$$\text{argmax}_{c \in C} P(c) \prod_{i=1}^{K} P(w_i \mid c)$$

Maximum likelihood estimates:

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

→ How many documents are class c_j in the training set

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

→ Fraction of times word w_i appears among all words in documents of class c_j

Data sparsity problem

What if count('fantastic', *positive*) = *0?*

➡ Implies P('fantastic' | *positive*) = 0

This term becomes 0 for c = *positive*

$$\text{argmax}_{c \in C} P(c) \prod_{i=1}^{K} P(w_i \mid c)$$

This sounds familiar…

Solution: Smoothing!

Laplace smoothing:

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

- Simple, easy to use
- Effective in practice

Overall process

Input: a set of labeled documents $\{(d_i, c_i)\}_{i=1}^{n}$

A. Compute vocabulary V of all words

B. Calculate $\hat{P}(c_j) = \dfrac{\text{Count}(c_j)}{\text{n}}$

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

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

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

prior - important!

Overall process

Input: a set of labeled documents $\{(d_i, c_i)\}_{i=1}^n$

A. Compute vocabulary V of all words

Q. What about words that appear in the testing set but not in V?

B. Calculate $\hat{P}(c_j) = \dfrac{\text{Count}(c_j)}{\text{n}}$

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

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

$$c_{MAP} = \arg\max_c \boxed{\hat{P}(c)} \prod_{i=1}^{K} \hat{P}(w_i \mid c)$$

prior - important!

Overall process

Input: a set of labeled documents $\{(d_i, c_i)\}_{i=1}^{n}$

A. Compute vocabulary V of all words

B. Calculate $\hat{P}(c_j) = \dfrac{\text{Count}(c_j)}{\text{n}}$

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

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

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

prior - important!

Q. What about words that appear in the testing set but not in V?

A. We can simply ignore them

A worked example for sentiment analysis

	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

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5

P(+) = 2/5

A worked example for sentiment analysis

	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

1. Prior from training:

$$\hat{P}(c_j) = \frac{N_{c_j}}{N_{total}}$$

P(-) = 3/5
P(+) = 2/5

2. Drop "with"

A worked example for sentiment analysis

	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

3. Estimating the conditional probs

$$p(w_i|c) = \frac{count(w_i, c) + 1}{(\sum_{w \in V} count(w, c)) + |V|}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \quad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \quad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

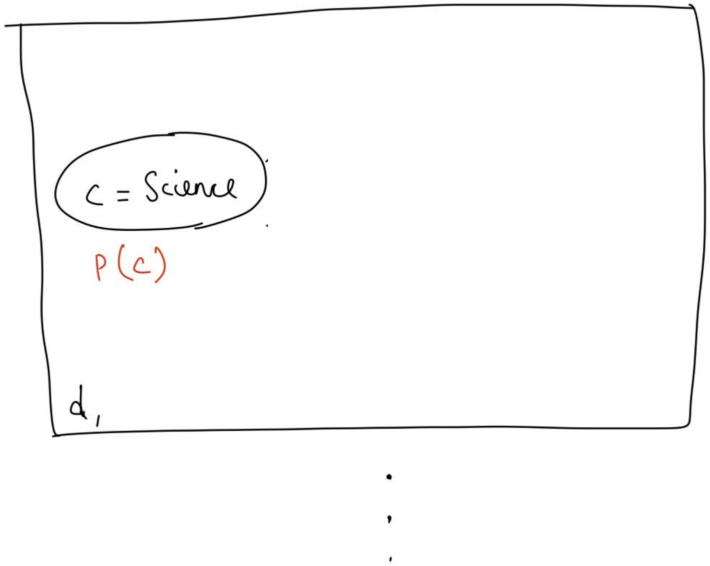
$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \quad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

4. Scoring the test example

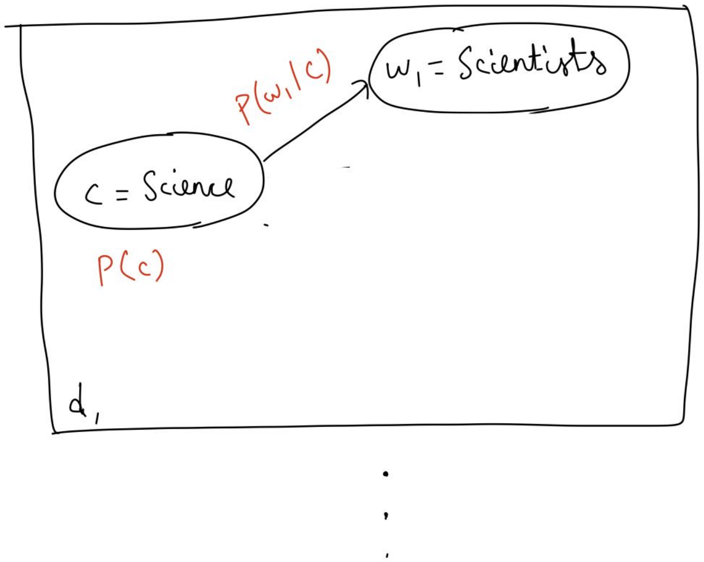
$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

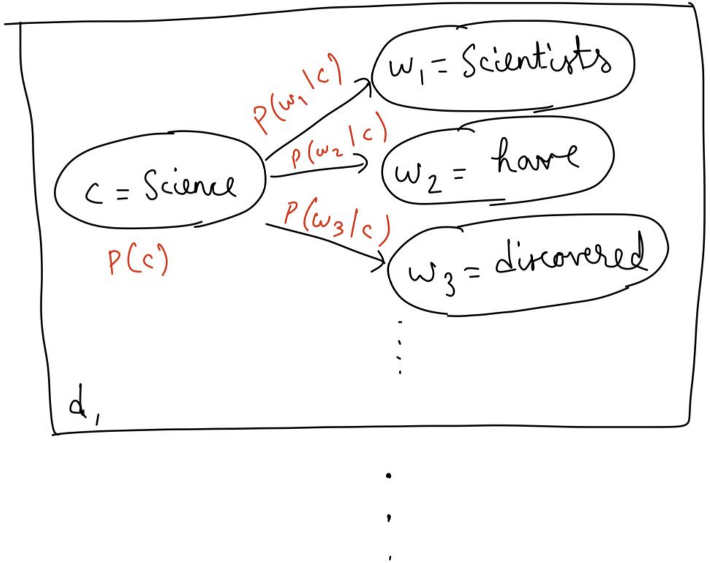
Naive Bayes vs. language models



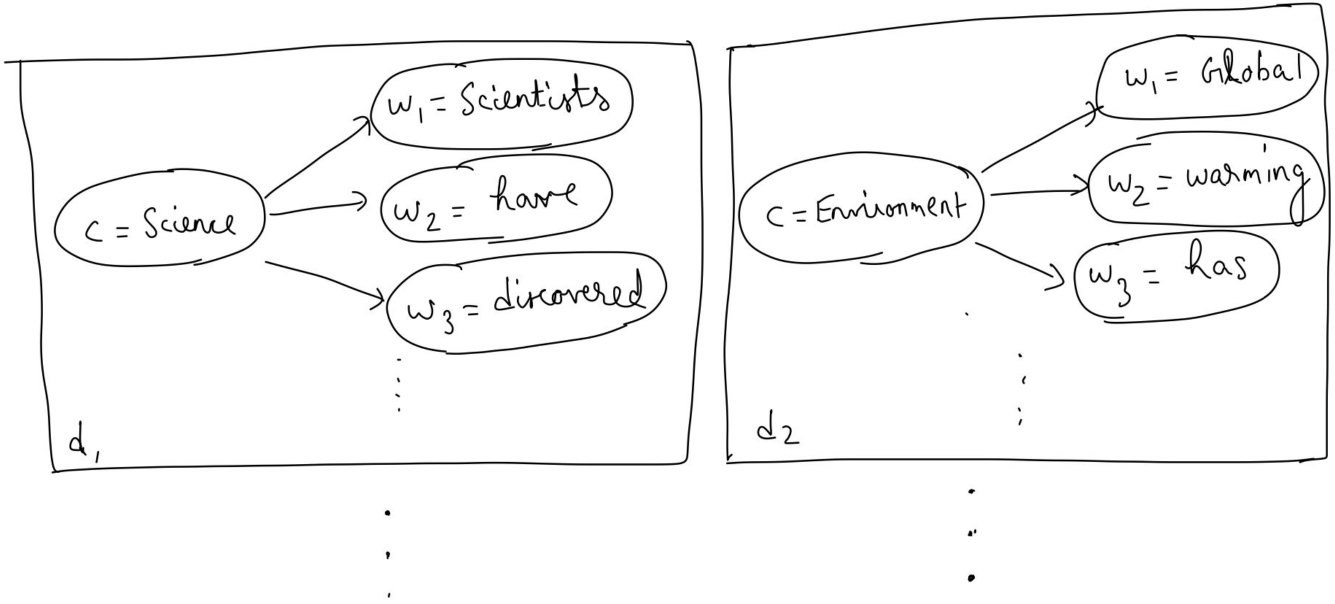
Naive Bayes vs. language models



Naive Bayes vs. language models



Naive Bayes vs. language models



Since $P(w_1, w_2, \ldots, w_K \mid c) = P(w_1 \mid c)P(w_2 \mid c) \ldots P(w_K \mid c)$

Each class = a unigram language model!

Naive Bayes vs. language models

- 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

Sentence s

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

A) pos B) neg C) both equal

Naive Bayes vs. language models

- 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

Sentence s

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)

Naive Bayes: pros and cons

- (+) Very fast, low storage requirements
- (+) Work well with very small amounts of training data
- (+) 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
- (-) The independence assumption is too strong
- (-) Doesn’t work well when the classes are highly imbalanced
  - Potential solutions: **complement Naive Bayes** (Rennie et al., 2003)

Naive Bayes can use any features!

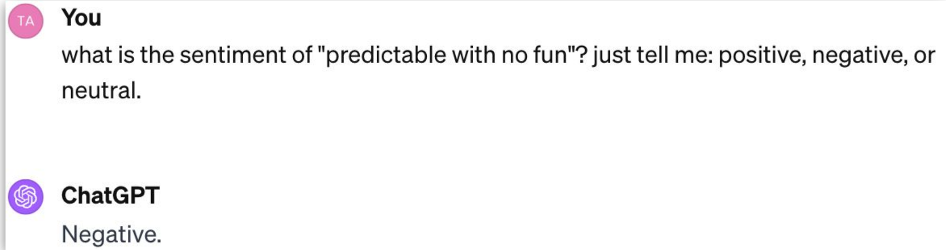
- In general, Naive Bayes can use any set of features, not just words:
  - URLs, email addresses, Capitalization, …
  - Domain knowledge crucial to performance

$$P(d|c) = P(f_1|c)P(f_2|c)\ldots P(f_{K'}|c)$$

Spam Attachments Features					
Habul Dataset			Botnet Dataset		
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
		(c)			(d)

Top features for spam detection

Wait, we already have ChatGPT, why still NB?


TA **You**

what is the sentiment of "predictable with no fun"? just tell me: positive, negative, or neutral.

ChatGPT

Negative.

Naive Bayes

Transformers, neural networks and many others
e.g., ChatGPT

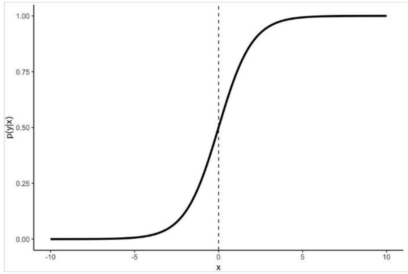
Wait, we already have ChatGPT, why still NB?

- Computational efficiency, cost
- Simplicity and interpretability
- Small data performance
- Out of domain
  - Requires domain experts to design features
- …

Naive Bayes

Transformers, neural networks and many others
e.g., ChatGPT

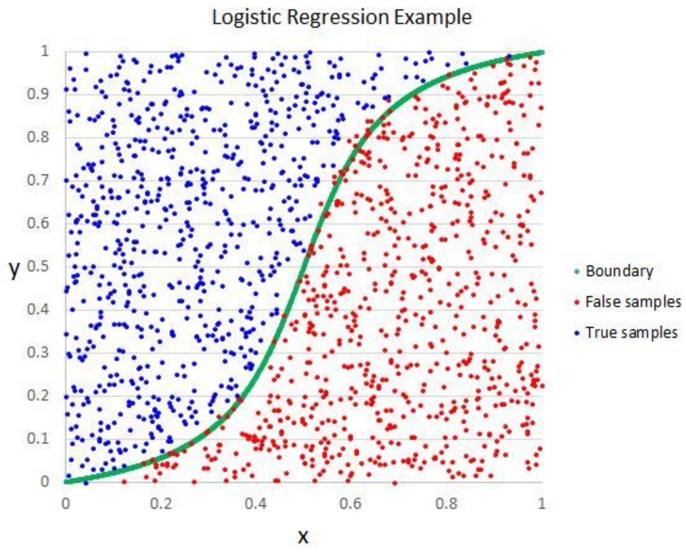
Logistic regression



Logistic regression

Study yourself!

Logistic regression



- Powerful supervised model
- Baseline approach for many NLP tasks
- Foundation of neural networks
- Binary (two classes) or multinomial (>2 classes)

https://machine-learning.paperspace.com/wiki/logistic-regression

Generative vs. discriminative models

- Naive Bayes is a ***generative*** model $\text{argmax}_{c \in C} P(d \mid c) P(c)$
- Logistic regression is a ***discriminative*** model $\text{argmax}_{c \in C} P(c \mid d)$

Suppose we're distinguishing cat from dog images

imagenet

imagenet

Generative classifiers

- Build a model of what is in a cat image
  - Knows about whiskers, ears, eyes
  - Assigns a probability to any image - how cat-y is this image?

- Also build a model for dog images

- Now given a new image:
  - **Run both models and see which one fits better?**

Discriminative classifiers

Just try to distinguish dogs from cats

Oh look, dogs have collars!
Let's ignore everything else

Overall process: Discriminative classifiers

Input: a set of labeled documents $\{(d_i, y_i)\}_{i=1}^{n}$

$y_i = 0$ or 1 (binary)

$y_i = 1, \ldots, m$ (multinomial)

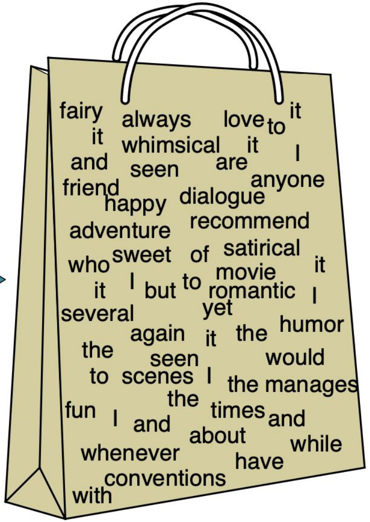
- Components:

  1. Convert d_i into a **feature representation** x_i
  2. **Classification function** to compute $\hat{y}$ using $P(\hat{y} \mid x)$ → Using either sigmoid or softmax!
  3. **Loss function** for learning
  4. Optimization **algorithm**

- Train phase: Learn the **parameters** of the model to minimize **loss function** on the training set
- Test phase: Apply **parameters** to predict class given a new input x (feature representation of testing document d)

1. Feature representation

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!



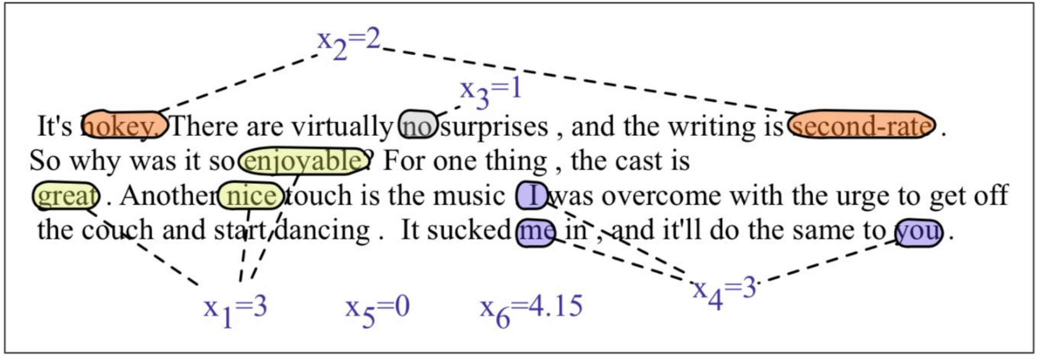
it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

Bag of words

$$\mathbf{x} = [x_1, x_2, \ldots, x_k]$$

In BoW representations, $k = |V|$ and the vector could be very sparse

Example: Sentiment classification



Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

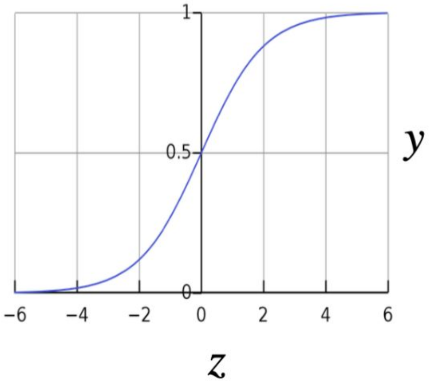
Remember that the values make up the feature vector!

2. Classification function

- *Given*: Input feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_k]$
- *Output*: $P(y = 1 \mid \mathbf{x})$ and $P(y = 0 \mid \mathbf{x})$ *(binary classification)*

Weight vector $\mathbf{w} = [w_1, w_2, \ldots, w_k]$ bias

- Given input features $\mathbf{x}$: $z = \mathbf{w} \cdot \mathbf{x} + b$
- Therefore, $\hat{y} = P(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = \dfrac{1}{1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)}}$
- Decision boundary: $= \begin{cases} 1 & \text{if } \hat{y} > 0.5 \\ 0 & \text{otherwise} \end{cases}$



Example: Sentiment classification

Var	Definition	Value
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

- Assume weights $\mathbf{w} = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$

$$
\begin{aligned}
p(+|x) = P(Y=1|x) &= \sigma(w \cdot x + b) \\
&= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) \\
&= \sigma(.805) \\
&= 0.69 \\
p(-|x) = P(Y=0|x) &= 1 - \sigma(w \cdot x + b) \\
&= 0.31
\end{aligned}
$$

3. Loss function

- For n data points (x_i, y_i), $\hat{y}_i = P(y_i = 1 \mid x_i)$
- Classifier probability: $\Pi_{i=1}^{n} P(y_i \mid x_i) = \Pi_{i=1}^{n} \hat{y}_i^{y_i} (1 - \hat{y}_i)^{1-y_i}$
- Loss: $-\log \prod_{i=1}^{n} P(y_i | x_i) = -\sum_{i=1}^{n} \log P(y_i | x_i)$

$$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

Example: Computing CE loss

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon) $\in$ doc)	3
x_2	count(negative lexicon) $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	log(word count of doc)	$\ln(64) = 4.15$

$$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

- Assume weights $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ and bias $b = 0.1$
- If y = 1 (positive sentiment), $L_{CE} = -\log(0.69) = 0.37$
- If y = 0 (negative sentiment), $L_{CE} = -\log(0.31) = 1.17$

$P(y = 1 \mid x) = 0.69$

$P(y = 0 \mid x) = 0.31$

Properties of CE loss

- $L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$
- What values can this loss take?

A) 0 to ∞ B) $-\infty$ to ∞ C) $-\infty$ to 0 D) 1 to ∞

Properties of CE loss

- $$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

- What values can this loss take?

A) 0 to ∞ B) $-\infty$ to ∞ C) $-\infty$ to 0 D) 1 to ∞

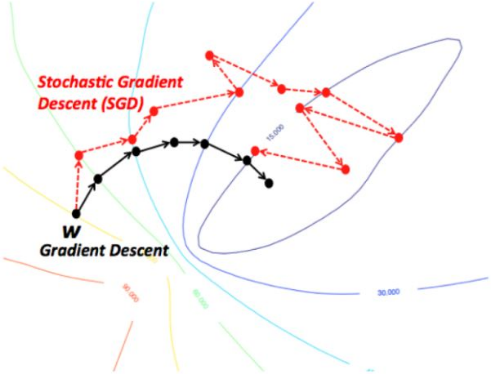
  - The answer is A) - Ranges from 0 (perfect predictions) to ∞
  - Lower the value, better the classifier

4. Optimization

- We have our **classification function** and **loss function** - how do we find the best w and b?

$$\theta = [w; b]$$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L_{CE}(y_i, x_i; \theta)$$



- **Optimization algorithm**: gradient descent!
- Cross entropy loss for logistic regression is **convex** (i.e. has only one global minimum) so gradient descent is guaranteed to find the minimum.

You should know what is learning rate, and what is stochastic gradient descent..

Gradient for logistic regression

$$\hat{y}_i = \sigma(\mathbf{w} \cdot \mathbf{x}_i + b)$$

$$L_{CE} = -\sum_{i=1}^{n} [y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)]$$

- Gradient, $\frac{dL_{CE}(\mathbf{w}, b)}{dw_j} = \sum_{i=1}^{n} [\hat{y}_i - y_i]x_{i,j}$

  The j-th value of the feature vector $\mathbf{x}_i$

- $\frac{dL_{CE}(\mathbf{w}, b)}{db} = \sum_{i=1}^{n} [\hat{y}_i - y_i]$

Regularization

- Training objective: $\hat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log P(y_i | x_i)$
- This might fit the training set too well! (including noisy features), and lead to poor generalization to the unseen test set — **Overfitting**
- **Regularization** helps prevent overfitting

$$\hat{\theta} = \arg\max_{\theta} \left[\sum_{i=1}^{n} \log P(y_i | x_i) - \alpha R(\theta)\right]$$

- L2 regularization:

$$\hat{\theta} = \arg\max_{\theta} \left[\sum_{i=1}^{n} \log P(y_i | x_i) - \alpha \sum_{j=1}^{d} \theta_j^2\right]$$

Multinomial logistic regression

- What if we have more than 2 classes?
- Need to model $P(y = c \mid x) \quad \forall c \in \{1, \ldots, m\}$
- Generalize **sigmoid** function to **softmax**

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}} \quad 1 \leq i \leq m$$

- The classifier probability is defined as:

$$P(y = c \mid x) = \frac{e^{\mathbf{w}_c \cdot \mathbf{x} + b_c}}{\sum_{j=1}^{m} e^{\mathbf{w}_j \cdot \mathbf{x} + b_j}}$$

Features in multinomial LR

- Features need to include both input (x) and class (c)

$$P(y = c \mid x) = \frac{e^{\mathbf{w}_c \cdot \mathbf{x} + b_c}}{\sum_{j=1}^{m} e^{\mathbf{w}_j \cdot \mathbf{x} + b_j}}$$

Var	Definition	Wt
$f_1(0,x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	-4.5
$f_1(+,x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	2.6
$f_1(-,x)$	$\begin{cases} 1 & \text{if “!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1.3

Learning

- Generalize binary loss to multinomial CE loss:

$$L_{CE}(\hat{y}, y) = -\sum_{c=1}^{m} 1\{y = c\} \log P(y = c \mid x)$$

$$= -\sum_{c=1}^{m} 1\{y = c\} \log \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{m} e^{w_j \cdot x + b_j}}$$

- Gradient:

$$\frac{dL_{CE}}{dw_c} = -(1\{y = c\} - P(y = c \mid x))x$$

$$= -\left(1\{y = c\} - \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{m} e^{w_j \cdot x + b_j}}\right)x$$

Next lecture: word embeddings