



# Machine learning

## Naive Bayes & Natural Language Processing

---

Wouter Gevaert & Marie Dewitte

# Inhoud

**Discriminative vs generative classification**

**Bayes rule**

**Naive Bayes - tekstclassificatie**

**Tekstclassificatie in de praktijk**

→ Naive Bayes

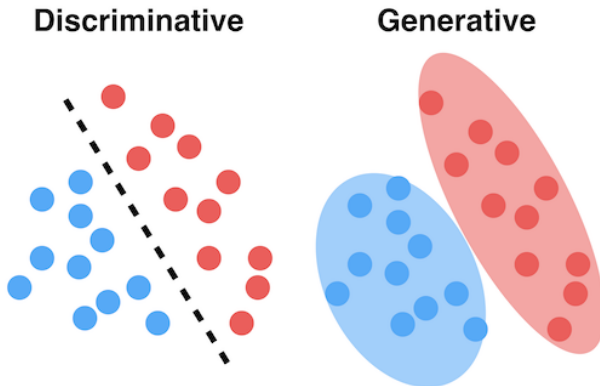
→ Smellheid

→ Pygmo. v.

## Discriminative vs generative classification

---

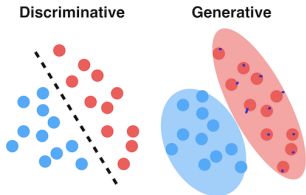
# Discriminative vs generative classification



Discriminative classifiers leren de **grens tussen twee klassen**.

Generative classifiers leren de **distributie van de verschillende klassen**.

# Discriminative vs generative classification



**Discriminative classifier:** leert  $p(y|x)$

*gegeven  
↳ features*

*p = probab. l. f. y*

**Generative classifier:** leert  $p(x|y)$  en ook  $p(y)$

Zoekt naar hoe de features van een bepaalde klasse er uit zien.

$p(y)$  noemt men de prior

## Bayes rule

---

# Bayes rule

## Voorbeeld - kanker

De kans dat iemand kanker heeft bedraagt 1%  $\Rightarrow P(kanker) = 0,01$

Van een test is geweten dat:

- In 90% van de gevallen is de test positief als je effectief kanker hebt (sensitiviteit).  $\rightarrow 10\%$ .
- In 90% van de gevallen is de test negatief als je geen kanker hebt (specificiteit).  $\rightarrow 10\%$ .

VRAAG: Als de test positief blijkt, wat is de kans dat je kanker hebt?

# Bayes rule

## Voorbeeld - kanker

$$P(\text{kanker}|\text{positief}) = \frac{P(\text{positief}|\text{kanker}) \times P(\text{kanker})}{P(\text{positief})}$$

*Sporen (waarschijnlijk)*

$$P(\text{kanker}|\text{positief}) = \frac{0,9 \times 0,01}{0,01 \times 0,90 + 0,1 \times 0,99} = 0,0833 \approx 8\%$$

**Prior:**  $P(\text{kanker})$ : Zonder de testresultaten te kennen, hoe waarschijnlijk is het dat iemand kanker heeft?

**Likelihood:**  $P(\text{positief}|\text{kanker})$  hoe waarschijnlijk is het dat de test positief is wanneer de persoon effectief kanker heeft?

**Marginal:**  $P(\text{positief})$  Hoe waarschijnlijk is het dat de test positief is?

**Posterior:**  $P(\text{kanker}|\text{positief})$ : Wat is de kans dat iemand effectief kanker heeft als de test positief blijkt?



# Bayes rule

## Likelihood

How probable is the evidence  
given that our hypothesis is true?

## Prior

How probable was our hypothesis  
before observing the evidence?

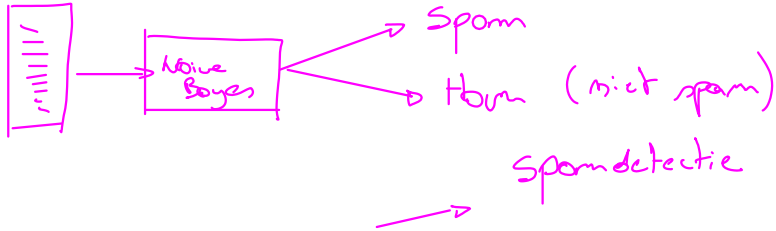
$$P(H | e) = \frac{P(e | H) P(H)}{P(e)}$$

## Posterior

How probable is our hypothesis  
given the observed evidence?  
(Not directly computable)

## Marginal

How probable is the new evidence  
under all possible hypotheses?  
 $P(e) = \sum P(e | H_i) P(H_i)$




## Naive Bayes - tekstclassificatie

---

# Naive Bayes - tekstclassificatie

## Spamdetectie

Kans dat die specifiek tekst (evidence) in een spam bericht voorkomt.


$$p(H|e) = \frac{p(e|H) \cdot p(H)}{p(e)}$$

Kans dat een bericht spam (los van de inhoud)

$H$  = hypothese (het bericht is spam)       $e$  = evidence (de tekst in het bericht)

$p(H|e)$ : de kans dat een bericht spam is gegeven de tekst van het bericht.


$p(e|H)$ : de kans dat we deze tekst vinden in een spam bericht.

$p(H)$ : de kans dat een willekeurig bericht spam is.

$p(e)$ : de kans dat we deze tekst tegenkomen / waarnemen. De tekst moet worden weergegeven als meerdere stukken evidence, namelijk de woorden  $w_1, w_2, \dots, w_n$ .

# Naive Bayes - tekstclassificatie

## Spamdetectie

$$p(\text{Spam} | w_1, \dots, w_n) = \frac{p(w_1, \dots, w_n | \text{Spam}) \cdot p(\text{Spam})}{p(w_1, \dots, w_n)}$$


$$p(\text{Spam} | w_1, \dots, w_n) = \frac{p(w_1 | w_2, \dots, w_n, \text{Spam}) \cdot p(w_2 | w_3, \dots, w_n, \text{Spam}) \cdot \dots \cdot p(\text{Spam})}{p(w_1, \dots, w_n)}$$

$p(w_1 | w_2, \dots, w_n, \text{Spam})$  is de kans om het eerste woord te vinden gegeven alle

andere woorden en gegeven dat het bericht spam is.

Lastig te berekenen  $\Rightarrow$  **Naive** Bayes veronderstelling.

Vereenvoudigde  
veronderstelling; zie.

Naive Bayes: beschouw elk woord onafhankelijk van de andere woorden.

# Naive Bayes - tekstclassificatie

## Spamdetectie

*1. bel. cond.*

*naive*

$$\rightarrow p(\text{Spam} | w_1, \dots, w_n) = \frac{p(w_1 | \text{Spam}) \cdot p(w_2 | \text{Spam}) \cdot p(w_3 | \text{Spam}) \dots p(w_n | \text{Spam}) \cdot p(\text{Spam})}{p(w_1, \dots, w_n)}$$

$$\Rightarrow p(\text{Ham} | w_1, \dots, w_n) = \frac{p(w_1 | \text{Ham}) \cdot p(w_2 | \text{Ham}) \dots p(w_n | \text{Ham}) \cdot p(\text{Ham})}{p(w_1, \dots, w_n)}$$

$p(w_1 | \text{Spam})$  = de kans dat  $w_1$  in een spambericht voorkomt.

~~X~~

~~y~~

$$p(\text{Spam} | w_1, \dots, w_n) = \frac{p(\text{Spam}) \prod_{i=1}^n p(w_i | \text{Spam})}{p(w_1, \dots, w_n)}$$

# Naive Bayes - tekstclassificatie

## Spamdetectie

**Vereenvoudigen:** kies spam of ham op basis van welke van deze klassen de grootste kans heeft.

We hebben geen kansen nodig.  $\Rightarrow$  laat de noemer (marginal) vallen.

$$p(\text{Spam}|w_1, \dots, w_n) \propto p(\text{Spam}) \prod_{i=1}^n p(w_i|\text{Spam})$$

# Naive Bayes - tekstclassificatie

## Spamdetectie - voorbeeld

aantal spam berichten: 280

$$\frac{90}{280} = 0,32$$

aantal ham berichten: 930

WOORD	SPAM		HAM	
	frequentie	kans	frequentie	kans
free	90	0,32	200	0,21
Viagra	60	0,21	55	0,059
buy	120	0,43	290	0,31
advice	50	0,18	380	0,41
get	180	0,64	510	0,55

$$\frac{200}{930} = 0,21$$

$$p(\text{Spam}) = \frac{280}{280 + 930} = \frac{280}{1210} = 0,23 = 23\%$$

Prior

$$p(\text{Ham}) = \frac{930}{280 + 930} = \frac{930}{1210} = 0,77 = 77\%$$

# Naive Bayes - tekstclassificatie

## Spamdetectie - voorbeeld

Classificeer de zin: "Get free advice"

$$p(\text{spam}|\text{get free advice}) \propto p(\text{spam}) \cdot p(\text{get}|\text{spam}) \cdot p(\text{free}|\text{spam}) \cdot p(\text{advice}|\text{spam})$$

$$= 0,23 \cdot 0,64 \cdot 0,32 \cdot 0,18 = 0,0085$$

prior      get      free      advice

$$p(\text{ham}|\text{get free advice}) \propto p(\text{ham}) \cdot p(\text{get}|\text{ham}) \cdot p(\text{free}|\text{ham}) \cdot p(\text{advice}|\text{ham})$$

$$= 0,77 \cdot 0,55 \cdot 0,21 \cdot 0,41 = 0,0365$$

get      free      advice

$0,0365 > 0,0085 \Rightarrow$  bericht is ham



# Naive Bayes - tekstclassificatie

## Laplacian smoothing

Classificeer de zin: "Get free Valium"

$$p(\text{Valium} | \text{spam}) = 0$$
$$p(\text{Valium} | \text{ham}) = 0$$

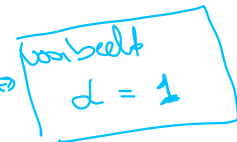
Valium behoort niet tot de trainingset  $\Rightarrow p(\text{Valium} | \text{spam}) = 0$  en  $p(\text{Valium} | \text{ham}) = 0$

$p(\text{spam} | \text{get free Valium}) = 0$  en  $p(\text{ham} | \text{get free Valium}) = 0$

- **Oplossing: Laplacian smoothing (= add-one smoothing):** kent niet geziene woorden toch een zekere kans van voorkomen toe.

# Naive Bayes - tekstclassificatie

## Laplacian smoothing

$$P(w_s) = \frac{C(w_s) + \boxed{\alpha}}{N + \boxed{\alpha}V}$$


A handwritten note in a blue box, tilted to the right, containing the text "voorbeeld" and "alpha = 1". An arrow points from the boxed alpha in the denominator of the equation to this note.

$P(w)$  is de kans op het woord  $w$

$C(w)$  het aantal keer dat het woord  $w$  voorkomt

$N$  is het totaal aantal woorden

$V$  is het aantal verschillende woorden (vocabulair grootte)

# Naive Bayes - tekstclassificatie

Laplacian smoothing - voorbeeld met  $\alpha = 1$

WOORD	SPAM		HAM	
	freq.	Laplacian smoothing	freq.	Laplacian smoothing
free	90	$\frac{90 + 1}{280 + 1 \times 6} = \frac{91}{286} = 0.318$	200	$\frac{200 + 1}{930 + 1 \times 6} = \frac{201}{936} = 0.215$
Viagra	60	$\frac{60 + 1}{280 + 1 \times 6} = \frac{61}{286} = 0.213$	55	$\frac{55 + 1}{930 + 1 \times 6} = \frac{56}{936} = 0.060$
buy	120	$\frac{120 + 1}{280 + 1 \times 6} = \frac{121}{286} = 0.423$	290	$\frac{290 + 1}{930 + 1 \times 6} = \frac{291}{936} = 0.311$
advice	50	$\frac{50 + 1}{280 + 1 \times 6} = \frac{51}{286} = 0.178$	380	$\frac{380 + 1}{930 + 1 \times 6} = \frac{381}{936} = 0.407$
get	180	$\frac{180 + 1}{280 + 1 \times 6} = \frac{181}{286} = 0.633$	510	$\frac{510 + 1}{930 + 1 \times 6} = \frac{511}{936} = 0.546$
Valium	0	$\frac{0 + 1}{280 + 1 \times 6} = \frac{1}{286} = 0.003$	0	$\frac{0 + 1}{930 + 1 \times 6} = \frac{1}{936} = 0.001$

# Naive Bayes - tekstclassificatie

Laplacian smoothing - voorbeeld met  $\alpha = 1$

Classificeer de zin: "Get free Valium"

$$p(\text{spam}|\text{get free Valium}) \propto p(\text{spam}) \cdot p(\text{get}|\text{spam}) \cdot p(\text{free}|\text{spam}) \cdot p(\text{Valium}|\text{spam})$$

$$= 0,23 \cdot 0,633 \cdot 0,318 \cdot \underbrace{0,003}_{\text{Valium}} = \underbrace{0,0001389}$$

$$p(\text{ham}|\text{get free advice}) \propto p(\text{ham}) \cdot p(\text{get}|\text{ham}) \cdot p(\text{free}|\text{ham}) \cdot p(\text{Valium}|\text{ham})$$

$$= 0,77 \cdot 0,546 \cdot 0,215 \cdot \underbrace{0,001}_{\text{Valium}} = \underbrace{0,00009}$$

$$0,0001389 > 0,00009 \Rightarrow \text{bericht is } \underline{\text{spam}}$$

$\alpha = 5$   
underfitting      overfitting

# Naive Bayes - tekstclassificatie

## Laplacian smoothing

Invloed van  $\alpha$

$$P(w_s) = \frac{C(w_s) + \alpha}{N + \alpha V}$$

- Kleine  $\alpha$ : neiging tot overfitting
- Grote  $\alpha$ : neiging underfitting

C { SUM  
Log Reg

$\alpha$  { Naive Bayes

# Naive Bayes - tekstclassificatie

## Log likelihood

Vermenigvuldigen van veel kansen kan resulteren in een floating-point underflow.

Oplossing: gebruiken van log likelihood.

$$\log(0,1 \times 0,3 \times 0,05) \\ \rightarrow \log(0,1) + \log(0,3) + \log(0,05)$$

$$\log(xy) = \log(x) + \log(y)$$

$$\log(p(\textit{Spam}|w_1, \dots, w_n)) \propto \log(p(\textit{Spam})) + \sum_{i=1}^n \log(p(w_i|\textit{Spam}))$$

# Naive Bayes - tekstclassificatie

## log likelihood - Classificeer de zin: "Get free Valium"

$$p(\text{spam}|\text{get free Valium}) \propto p(\text{spam}) \cdot p(\text{get}|\text{spam}) \cdot p(\text{free}|\text{spam}) \cdot p(\text{Valium}|\text{spam})$$

$$= 0,23 \cdot 0,633 \cdot 0,318 \cdot 0,003 = 0,0001389$$

$$\log(p(\text{spam}|\text{get free Valium})) \propto \log(p(\text{spam})) + \log(p(\text{get}|\text{spam})) + \log(p(\text{free}|\text{spam})) + \log(p(\text{Valium}|\text{spam}))$$

$$= \log(0,23) + \log(0,633) + \log(0,318) + \log(0,003) = \underline{-3,857}$$

$$p(\text{ham}|\text{get free advice}) \propto p(\text{ham}) \cdot p(\text{get}|\text{ham}) \cdot p(\text{free}|\text{ham}) \cdot p(\text{Valium}|\text{ham})$$

$$= 0,77 \cdot 0,546 \cdot 0,215 \cdot 0,001 = 0,00009$$

$$\log(p(\text{ham}|\text{get free Valium})) \propto \log(p(\text{ham})) + \log(p(\text{get}|\text{ham})) + \log(p(\text{free}|\text{ham})) + \log(p(\text{Valium}|\text{ham}))$$

$$= \log(0,77) + \log(0,546) + \log(0,215) + \log(0,001) = \underline{-4,044}$$

$-3,857 > -4,044 \Rightarrow$  bericht is spam

## Tekstclassificatie in de praktijk

---



# Tekstclassificatie in de praktijk

## Preprocessing - opkuisen van de tekst

### → Verwijder html

```
from bs4 import BeautifulSoup
text_no_html = BeautifulSoup(str(text), "html.parser").get_text()
```

### → Verwijder niet-letters

```
import re #regular expressions
text_alpha_chars = re.sub("[^a-zA-Z ']", "_", str(text_no_html))
```

### → Converteer naar lowercase

```
text_lower = text_alpha_chars.lower()
```

# Tekstclassificatie in de praktijk

## Preprocessing - opkuisen van de tekst (vervolg)

### Verwijder stopwoorden

```
from nltk.corpus import stopwords
stops = set(stopwords.words('english'))
text_no_stop_words = ''
for w in text_lower.split():
    if w not in stops:
        text_no_stop_words = text_no_stop_words + w + ' '
```

The  
a  
thin

# Tekstclassificatie in de praktijk

## Preprocessing - Herleiden van woorden tot de stam

### → Stemming

```
from nltk.stem.snowball import SnowballStemmer

text_stemmer = '_'
stemmer = SnowballStemmer(language)
for w in text_no_stop_words.split():
    text_stemmer = text_stemmer + stemmer.stem(w) + '_'
```



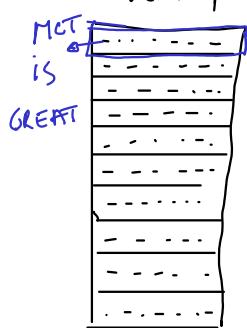
# Tekstclassificatie in de praktijk → BAG OF WORDS

Preprocessing - Verwijder te korte woorden

Verwijder korte woorden na stemming

```
text_no_short_words = ''  
for w in text_stemmer.split():  
    if len(w) >= minWordSize:  
        text_no_short_words = text_no_short_words + w + ' '
```

X-vec prepoc.



208 353 503  
 (501 300 8) 208  
 353  
 503

Sparse  
 Vocabulary



10000

one-hot vector  
 MCT is GREAT  
 0 0 0  
 1 0 0  
 0 1 0  
 0 0 1  
 0 0 0  
 0 0 0  
 0 0 0  
 0 0 0  
 0 0 0



# Tekstclassificatie in de praktijk

## Opbouwen van feature vectors - bag of words

**Bag of words** = collectie van unieke woorden die voorkomen in de volledige trainingset. De uiteindelijke feature vector heeft dezelfde dimensie als de bag of words.

**Multi-variate Bernoulli Naive Bayes:** er wordt enkel bijgehouden of een woord in de bag of words al dan niet voorkomt in een document (0 of 1) en dus niet de frequentie van voorkomen.

**Multinomial Naive Bayes :** de frequentie (aantal keer) waarmee een woord uit de bag of words voorkomt in het document wordt bijgehouden.

In Scikit-learn: CountVectorizer

# Tekstclassificatie in de praktijk

## Opbouwen van feature vectors - bag of words

**tf-idf**: term frequency-inverse document frequency.

Sommige woorden die veel voorkomen in documenten zijn minder belangrijk.

$$tfidf_{i,j} = tf_{i,j} \times \log \frac{N}{df_i} \quad (1)$$

$tf_{i,j}$  = het aantal keer dat woord  $i$  voorkomt in document  $j$


$df_i$  = het aantal documenten waar woord  $i$  in voorkomt

$N$  = totaal aantal documenten

In Scikit-learn: TfidfTransformer

# Tekstclassificatie in de praktijk

## Opbouwen van feature vectors - bag of words



```
count_vect = CountVectorizer()  
X_train_bag_of_words = count_vect.fit(X_train)  
X_train_bag_of_words = count_vect.transform(X_train)
```

```
tfidf_transformer = TfidfTransformer()  
tf_transformer = TfidfTransformer().fit(X_train)  
X_train_tf = tf_transformer.transform(X_train_bag_of_words)
```

