

Faculty for Informatics

Technical
University
of Munich



Natural Language Processing IN2361

PD Dr. Georg Groh

Social Computing
Research Group

Chapter 4:

Naïve Bayes and Sentiment Classification

- content is based on [1] and [2]
- certain elements (e.g. equations or tables) were taken over or taken over in a modified form from [1] or [2]
- citations of [1] and [2] or from [1] or [2] are omitted for legibility
- errors are fully in the responsibility of Georg Groh
- BIG thanks to Dan and James for a great book!

Repetition of Naïve Bayes from ML1-Class

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), (x^{(1)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$$

$$\begin{aligned} p(\mathcal{D}|\Theta) &= \prod_{n=1}^N p(x^{(n)}, y^{(n)}|\theta, \pi) \\ &= \prod_{n=1}^N p(x^{(n)}|y^{(n)}, \theta) p(y^{(n)}|\pi) \\ &= \prod_{n=1}^N \prod_{v=1}^V p(x_v^{(n)}|y^{(n)}, \theta) p(y^{(n)}|\pi) \\ &= \prod_{n=1}^N \prod_{c=1}^C \prod_{v=1}^V p(x_v^{(n)}|\theta_{vc}) \mathbb{1}(y^{(n)}=c) \prod_{c'=1}^C \pi_{c'} \mathbb{1}(y^{(n)}=c') \end{aligned}$$

Repetition of Naïve Bayes from ML1-Class

$$\bar{\mathcal{D}} = ((x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)}))$$

likelihood for model parameters

$$p(\mathcal{D}|\Theta) = \prod_{n=1}^N p(x^{(n)}, y^{(n)}|\theta, \pi)$$

generative classifier

$$= \prod_{n=1}^N \underbrace{p(x^{(n)}|y^{(n)}, \theta)}_{\text{class conditional density} = \text{class-likelihood}} \underbrace{p(y^{(n)}|\pi)}_{\text{class prior}}$$

"naïve" assumption

$$= \prod_{n=1}^N \prod_{v=1}^V p(x_v^{(n)}|y^{(n)}, \theta) p(y^{(n)}|\pi)$$

categorical class priors

$$= \prod_{n=1}^N \prod_{c=1}^C \prod_{v=1}^V p(x_v^{(n)}|\theta_{vc}) \mathbb{1}(y^{(n)}=c) \prod_{c'=1}^C \pi_{c'} \mathbb{1}(y^{(n)}=c')$$

- **bag of words model:** position in document / word sequence does not matter in determining feature(s) for a word
- **naïve assumption:** conditional independence of features

feature vector $x^{(n)}$: e.g. V-dim. vector of term-frequencies
(notation in Jurafksy: $x^{(n)} := d$ (a representation of a document))

Repetition of Naïve Bayes from ML1-Class

$$\log p(\mathcal{D}|\Theta) = \sum_{c=1}^C \sum_{v=1}^V \sum_{\{n|y^{(n)}=c\}} \log p(x_v^{(n)}|\theta_{vc}) + \sum_{c=1}^C N_c \log \pi_c$$

- Simple MLE solution for π_c :

$$0 = \partial/\partial\pi_c \log p(\mathcal{D}|\Theta) + \text{Lagrange multiplier} \implies$$

$$\pi_c^{MLE} = \frac{N_c}{N} \text{ as usual for a categorical class prior}$$

Jurafsky notation:

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

Repetition of Naïve Bayes from ML1-Class

$$\log p(\mathcal{D}|\Theta) = \sum_{c=1}^C \sum_{v=1}^V \sum_{\{n|y^{(n)}=c\}} \log p(x_v^{(n)}|\theta_{vc}) + \sum_{c=1}^C N_c \log \pi_c$$

- Simple MLE solution for π_c :

$$0 = \partial/\partial\pi_c \log p(\mathcal{D}|\Theta) + \text{Lagrange multiplier} \implies$$

$$\pi_c^{MLE} = \frac{N_c}{N} \text{ as usual for a categorical class prior}$$

Jurafsky notation:

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

- Simple MLE solution for θ_{vc} :

Multinomial distribution: (Rolling a class-specific V -sided dice $M^{(n)}$ times to create a document $x^{(n)}$ of length $M^{(n)}$)

$$p(x^{(n)}|y^{(n)} = c, \theta) = Mu(x_1^{(n)}, \dots, x_V^{(n)}|\theta_{1c}, \dots, \theta_{Vc}) = \frac{M^{(n)}!}{\prod_{v=1}^V x_v^{(n)}!} \prod_{v=1}^V \overbrace{\theta_{vc}^{x_v^{(n)}}}^{p(x_v^{(n)}|\theta_{vc})}$$

Repetition of Naïve Bayes from ML1-Class

- Simple MLE solution for θ_{vc} (contd.):

(for better understanding (no change in model / results): conceptually concatenate all documents in class c into one big document \tilde{d} of length \tilde{N}_c
→ words in \tilde{d} have multinomial distribution)

$0 = \partial / \partial \theta_{vc} \log p(\mathcal{D} | \Theta)$ + Lagrange multiplier \Rightarrow

$$\theta_{vc}^{MLE} = \frac{N_{vc}}{\tilde{N}_c} = \frac{\sum_{\{n | x^{(n)} \in c\}} x_v^{(n)}}{\sum_{\{n | x^{(n)} \in c\}} M^{(n)}} = \frac{\sum_{\{n | x^{(n)} \in c\}} x_v^{(n)}}{\sum_{\{n | x^{(n)} \in c\}} \sum_{v=1}^V x_v^{(n)}} = \frac{N_{vc}}{\sum_{v=1}^V N_{vc}}$$

(as usual for a
multinomial
class conditional
density)

Jurafsky notation ($v = „i“$):

$$\hat{P}(w_i | c) = \frac{\text{count}(w_i, c)}{\sum_{w \in V} \text{count}(w, c)}$$

Repetition of Naïve Bayes from ML1-Class

- Simple **MAP** solution for θ_{vc} using a **Dirichlet prior**:

$$\begin{aligned} p(\theta_c|D) &\propto p(D|\theta_c)p(\theta_c|\alpha) \\ &\propto \prod_{v=1}^V \theta_{vc}^{N_{vc}} \theta_{vc}^{\alpha_v-1} \\ &\propto \prod_{v=1}^V \theta_{vc}^{N_{vc}+\alpha_v-1} \\ &\propto \text{Dir}(\theta|(\alpha_1 + N_{1c}, \alpha_2 + N_{2c}, \dots, \alpha_V + N_{Vc})) \end{aligned}$$

$$0 = \partial/\partial\theta_{vc} \log p(\theta_c|\mathcal{D}) + \text{Lagrange multiplier} \quad \Rightarrow$$

$$\theta_{vc}^{MAP} = \frac{N_{vc} + \alpha_v - 1}{\widetilde{N}_c + (\sum_{v=1}^V \alpha_v) - V} \quad \text{as usual for a Dirichlet posterior}$$

Special MAP = Add One Smoothing

- Using weak prior $\alpha = (2, 2, 2, 2, \dots)$ we get **Add-One-Smoothing** (Laplace smoothing)

in Jurafsky notation:

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

- **Using the trained classifier** (trained == model parameters have been determined) on a new unseen document x is simple:

$$\begin{aligned} \operatorname{argmax}_c p(y = c|x, \Theta_{MAP/MLE}) \\ = \operatorname{argmax}_c p(x|y = c, \Theta_{MAP/MLE}) * p(y|\Theta_{MAP/MLE}) \end{aligned}$$

Very Simple Sentiment Classifiers

- For whole documents (e.g. a Facebook comment) or (better) individual sentences: classify into two (**positive, negative**) or three (**positive, neutral, negative**) **classes of sentiment**
- **unknown words** (words in test set but not in training set): ignore 😊
- **stop words** (I, at, in, can (?!),....): *maybe* remove them

	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

Very Simple Sentiment Classifiers

- Instead of term-frequencies for words also possible: **binary features** (word present (=1) or not (=0))

→ generative process: instead of assuming rolling a V-sided dice M_n times to “create” a document, for each word toss a θ_{vc} coin to determine if present in the document.

principal form of MLE solution for θ_{vc} **does not change** (it's a linear model)

Binary Naïve Bayes: often **works better** for sentiment analysis

Four original documents:

- it was pathetic the worst part was the boxing scenes
- no plot twists or great scenes
- + and satire and great plot twists
- + great scenes great film

After per-document binarization:

- it was pathetic the worst part boxing scenes
- no plot twists or great scenes
- + and satire great plot twists
- + great scenes film

	NB Counts		Binary Counts	
	+	–	+	–
and	2	0	1	0
boxing	0	1	0	1
film	1	0	1	0
great	3	1	2	1
it	0	1	0	1
no	0	1	0	1
or	0	1	0	1
part	0	1	0	1
pathetic	0	1	0	1
plot	1	1	1	1
satire	1	0	1	0
scenes	1	2	1	2
the	0	2	0	1
twists	1	1	1	1
was	0	2	0	1
worst	0	1	0	1

Very Simple Sentiment Classifiers

- **Dealing with Negations:** prepend the prefix NOT to every word after a token of logical negation (n't, not, no, never) until the next punctuation mark → new words that indicate opposite sentiment

...didn't like this movie, but I... 

...didn't NOT_like NOT_this NOT_movie , but I...

- **Further possibility:** work with only **two features** per document:
number of words with (a priori) **positive** sentiment,
number of words with (a priori) **negative** sentiment

- (a priori) word sentiment: from **sentiment lexica**
(e.g. LWIC (2007), MPQA (2005))

+ : *admirable, beautiful, confident, dazzling, ecstatic, favor, glee, great*

– : *awful, bad, bias, catastrophe, cheat, deny, envious, foul, harsh, hate*

Naïve Bayes as Set of Class-Specific Unigram Language Models

- having learned the $\theta_{vc} = P(w_v|c)$, computing the probability of a sentence S in our model is easy:

$$P(S = (w_{v_1}, w_{v_2}, \dots, w_{v_{|S|}}) | \theta, c) = \prod_{i=1}^{|S|} P(w_{v_i} | \theta, c) = \prod_{i=1}^{|S|} \theta_{v_i c} = \prod_{v=1}^V \theta_{vc}^{N_{vc}}$$

- **example:** for a two class sentiment classifier (using term-frequencies or binary word features) the class likelihoods for a sentence are

$$P(\text{"I love this fun film"} | +) = 0.1 \times 0.1 \times 0.01 \times 0.05 \times 0.1 = 0.0000005$$

$$P(\text{"I love this fun film"} | -) = 0.2 \times 0.001 \times 0.01 \times 0.005 \times 0.1 = .0000000010$$

assuming

w	P(w +)	P(w -)
I	0.1	0.2
love	0.1	0.001
this	0.01	0.01
fun	0.05	0.005
film	0.1	0.1
...

Classifiers: Error / Performance Measures: Confusion Matrix

		Actual class	
		Cat	Non-cat
Predicted class	Cat	5 True Positives	2 False Positives
	Non-cat	3 False Negatives	17 True Negatives

		Actual class		
		Cat	Dog	Rabbit
Predicted class	Cat	5	2	0
	Dog	3	3	2
	Rabbit	0	1	11

Classifiers: Error / Performance Measures: Two Classes

Accuracy:

$$ACC = \frac{TP + TN}{TP + FP + FN + TN}$$

		Actual	
		y=1	y=0
Predicted	y=1	TP	FP <small>type I error</small>
	y=0	FN <small>type II error</small>	TN

Precision (positive predictive value):

$$PREC = \frac{TP}{TP + FP}$$

Recall (sensitivity, true positive rate):

$$REC = \frac{TP}{TP + FN}$$

Specificity (true negative rate):

$$TNR = \frac{TN}{FP + TN}$$

False Negative Rate (miss rate):

$$FNR = \frac{FN}{TP + FN}$$

False Positive Rate (fall out):

$$FPR = \frac{FP}{FP + TN}$$

F1 Score (harmonic mean of Recall and Precision):

$$F1 = \frac{2 * PREC * REC}{PREC + REC}$$

Classifiers: Error / Performance Measures: >2 Classes

- usually: use **average of one vs rest** (macro averaging): example:

Accuracy:

$$ACC = \frac{1}{C} \sum_{c=1}^C \frac{TP_c + TN_c}{TP_c + FP_c + FN_c + TN_c} = \frac{1}{C} \sum_{c=1}^C ACC_c$$

- possible: **weighted approach** (e.g. using class-priors / inverse class priors to emphasize importance of frequent / infrequent classes): example:

Accuracy:

$$ACC = \sum_{c=1}^C \pi_c ACC_c$$

- micro-averaging μ vs macro-averaging M** example:

Precision $_{\mu}$

$$PREC = \frac{\sum_{c=1}^C TP_c}{\sum_{c=1}^C TP_c + FP_c}$$

Precision $_M$:

$$PREC = \frac{1}{C} \sum_{c=1}^C \frac{TP_c}{TP_c + FP_c} = \frac{1}{C} \sum_{c=1}^C PREC_c$$

Classifiers: Error / Performance Measures: >2 Classes

example

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

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

$\text{microaverage precision} = \frac{268}{268+99} = .73$

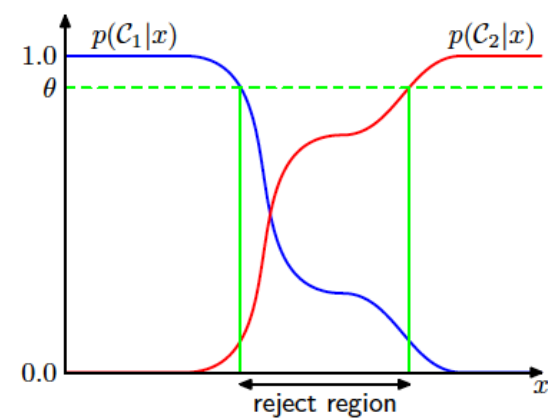
$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = .60$

Classifiers: Error / Performance Measures: ROC / AUC

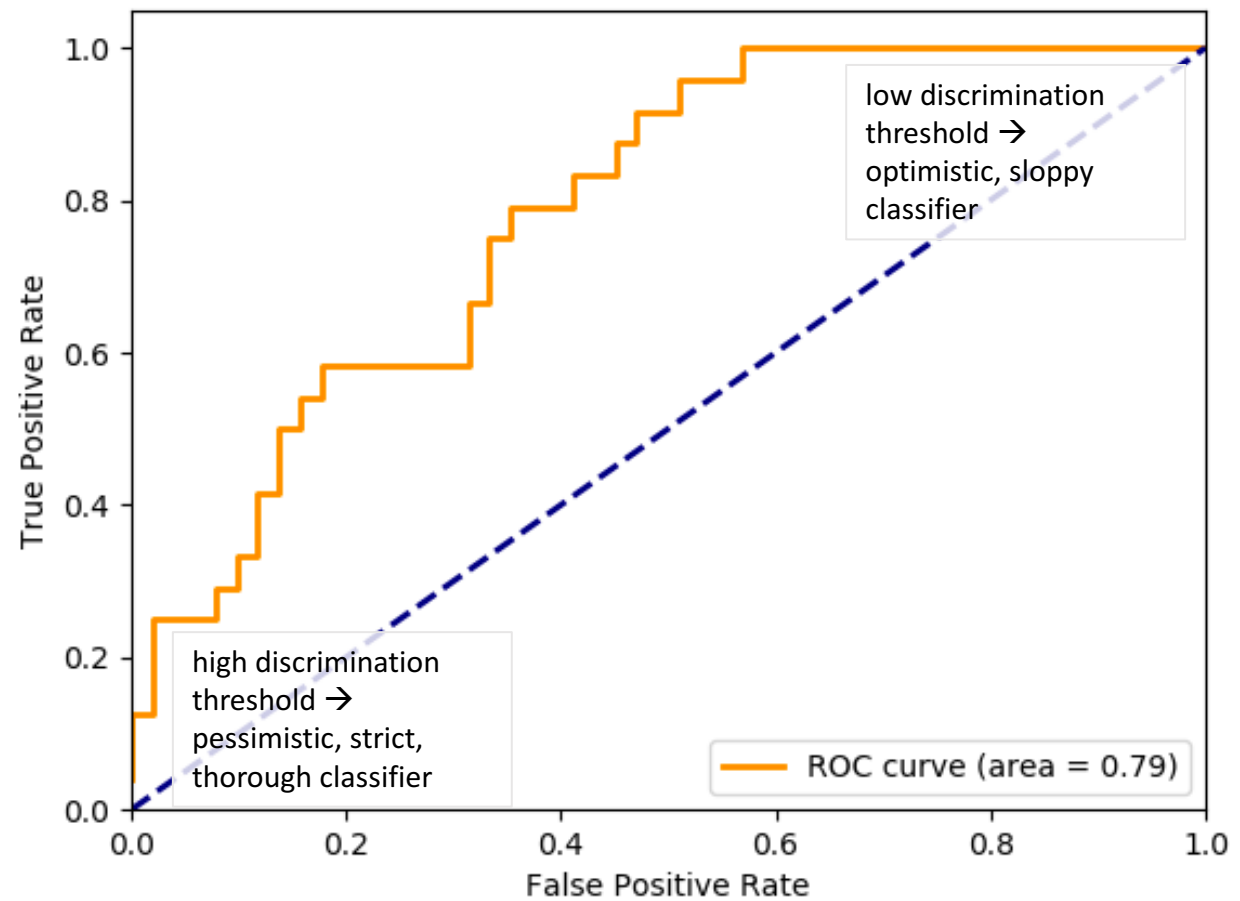
		Actual	
		y=1	y=0
Predicted	y=1	TP	FP
	y=0	FN	TN

Recall (sensitivity, true positive rate):

$$REC = \frac{TP}{TP + FN}$$



Receiver operating characteristic example



False Positive Rate (fall out):

$$FPR = \frac{FP}{FP + TN}$$

		Actual	
		y=1	y=0
Predicted	y=1	TP	FP
	y=0	FN	TN

Figure 1.26, Bishop

Statistical Significance Testing

- **Compare two classifiers** A and B: use F1 score on test set x , yielding e.g. the result $\delta(x) = F1(A) - F1(B) > 0$. (“A better than B”)
But: is result **statistically significant**?
- **Null Hypothesis H_0** : “result ‘A better than B’ was achieved by chance”
→ if X is random variable over all conceivable test-sets, H_0 expects $P(\delta(X) > \delta(x) | H_0)$ to be high.
Reject $H_0 \leftrightarrow$ p-Value $P(\delta(X) > \delta(x) | H_0) < 0.05$.
- Performance measures are rarely normally distributed → paired T-test not applicable; furthermore: test data is scarce → use **bootstrap testing** (can be applied to any performance measure)

Bootstrap Testing

- two classes: for each element of x , we can have **four cases**:
 A and B are right (AB), A is right and B wrong ($A\cancel{B}$),
 A is wrong and B right ($\cancel{A}B$), A and B are wrong ($\cancel{A}\cancel{B}$)

	1	2	3	4	5	6	7	8	9	10	A%	B%	$\delta()$
x	AB	$A\cancel{B}$	AB	$\cancel{A}B$	$A\cancel{B}$	$\cancel{A}\cancel{B}$	$A\cancel{B}$	AB	$\cancel{A}\cancel{B}$	$A\cancel{B}$.70	.50	.20
$x^{*(1)}$	$A\cancel{B}$	AB	$A\cancel{B}$	$\cancel{A}B$	$\cancel{A}\cancel{B}$	$A\cancel{B}$	$\cancel{A}\cancel{B}$	AB	$\cancel{A}\cancel{B}$	AB	.60	.60	.00
$x^{*(2)}$	$A\cancel{B}$	AB	$\cancel{A}\cancel{B}$	$\cancel{A}B$	$\cancel{A}\cancel{B}$	AB	$\cancel{A}\cancel{B}$	$A\cancel{B}$	AB	AB	.60	.70	-.10
...													
$x^{*(b)}$													

assuming accuracy instead of F1 for the sake of simplicity

Figure 6.8 The bootstrap: Examples of b pseudo test sets being created from an initial true test set x . Each pseudo test set is created by sampling $n = 10$ times with replacement; thus an individual sample is a single cell, a document with its gold label and the correct or incorrect performance of classifiers A and B.

then apply
 (1.96 σ =) 2 σ -rule
 <-> t-Test:

```
function BOOTSTRAP( $x, b$ ) returns  $p\text{-value}(x)$ 
  Calculate  $\delta(x)$ 
  for  $i = 1$  to  $b$  do
    for  $j = 1$  to  $n$  do    # Draw a bootstrap sample  $x^{*(i)}$  of size  $n$ 
      Select a member of  $x$  at random and add it to  $x^{*(i)}$ 
    Calculate  $\delta(x^{*(i)})$ 
  for each  $x^{*(i)}$ 
     $s \leftarrow s + 1$  if  $\delta(x^{*(i)}) > 2\delta(x)$ 
   $p\text{-value}(x) \approx \frac{s}{b}$ 
  return  $p\text{-value}(x)$ 
```

Feature Selection

- “per feature” feature selection (unlike PCA etc.): compare decision trees: rank features according to their discriminative power:

- Entropy-based (Information Gain)

$$\begin{aligned} G(w) = & -\sum_{i=1}^C P(c_i) \log P(c_i) \\ & + P(w) \sum_{i=1}^C P(c_i|w) \log P(c_i|w) \\ & + P(\bar{w}) \sum_{i=1}^C P(c_i|\bar{w}) \log P(c_i|\bar{w}) \end{aligned}$$

- GINI Index

$$G(w) = 1 - \sum_{i=1}^C P(c_i|w)^2$$

- or select skew directions in feature space: e.g. via PCA, pPCA, Factor Analysis, SVD etc. (compare Murphy chapter 12)



- (1) Dan Jurafsky and James Martin: Speech and Language Processing (3rd ed. draft, version Oct. 2019); Online: <https://web.stanford.edu/~jurafsky/slp3/> (URL Oct 2019) (this slideset is especially based on chapter 4)
- (2) Powerpoint slides from Dan Jurafsky and James Martin: Speech and Language Processing (3rd ed. draft); Online: <https://web.stanford.edu/~jurafsky/slp3/> (URL, Oct 2018)
- (3) K. Murphy: Machine Learning – a Probabilistic Perspective, MIT Press 2012 (especially section 3.5)

Recommendations for Studying

- minimal approach:

work with the slides and understand their contents! Think beyond instead of merely memorizing the contents

- standard approach:

minimal approach + read the corresponding pages in Jurafsky [1]

- interested students

== standard approach