Chapter 02

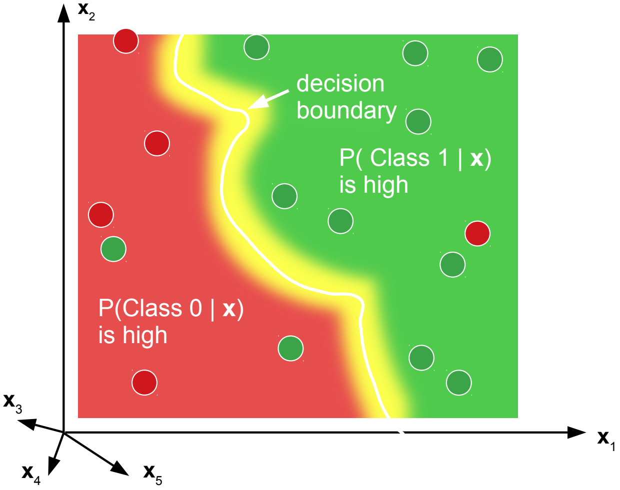
Naive Bayes

Prof. Dr. Adrian Ulges

RheinMain University of Applied Sciences

Recap: Classification Problems

- Feature vectors $\mathbf{x}$ are points in feature space.
- The ML model estimates decision boundaries between classes.



Outline

1. Decision Theory

2. Naive Bayes

Risk Minimization

		true class c'			
		1	**2**	**...**	**C'**
chosen class c	**1**	0	*L(1,2)*	...	*L(1,C')*
	2	*L(2,1)*	0	...	*L(2,C')*
	...	...	...	0	...
	C'	*L(C',1)*	*L(C',2)*	...	0

In classification, an object $\mathbf{x}$ is given, and our model picks a class $c \in \{1, ..., C'\}$.

Approach: Random Process

- $\mathbf{x} \in \mathbb{R}^d$ is sampled from a random variable X.
- C (the class) is also a random variable.
- Question: What is the optimal class for our model to pick?

Answer: Minimizing Cost / Risk

- Our model picks class c, while the true class is c'.
- We define a cost function

$$L : \{1, ..., C'\} \times \{1, ..., C'\} \to \mathbb{R}$$

  where $L(c, c')$ is the cost resulting from misclassifying c' as c.
- Note that some misclassifications come with higher cost than others. Example: spam classification
  - misclassifying spam as ham: not too bad.
  - misclassifying ham as spam: bad.

Risk Minimization (cont'd)

When deciding for class c, we define this decision's risk $R(c|\mathbf{x})$ as the expected cost:

		true class c'			
		1	**2**	**...**	**C'**
chosen class c	**1**	0	*L(1,2)*	...	*L(1,C')*
	2	*L(2,1)*	0	...	*L(2,C')*
	...	...	...	0	...
	C'	*L(C',1)*	*L(C',2)*	...	0

Optimal Strategy: Choose the class c^* that minimizes risk:

Zero-One-Loss

A common choice is **zero-one loss**:
Correct decisions cost 0,
misclassifications cost 1.

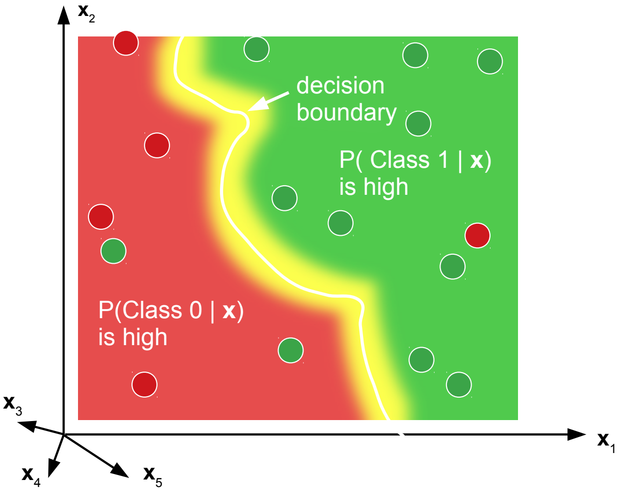
What is the best decision in this case?

	true class c'				
		1	**2**	**...**	**C'**
chosen class c	**1**	0	*1*	...	*1*
	2	*1*	0	...	*1*
	...	...	...	0	...
	C'	*1*	*1*	...	0

$\Rightarrow$ To make optimal decisions, we need to estimate $P(c|\mathbf{x})$.

Recap: Classification Problems

- Feature vectors $\mathbf{x}$ are points in feature space.
- The ML model estimates decision boundaries between classes.



Outline

Machine Learning and Bayes' Rule

Our **goal** is to compute $P(c|\mathbf{x})$ (commonly called the **posterior**).
We could do this using Bayes' rule:

Two Types of Classifiers

In general, there are two general **kinds of classifiers**:

1. **Generative** methods: compute $P(c)$ and $P(\mathbf{x}|c)$ and plug them into Bayes' rule.
2. **Discriminative** methods: use a direct model for $P(c|\mathbf{x})$ *(or, alternatively, the decision boundary)*.

Generative Models

Let's look at generative models, i.e. compute $P(c)$ and $P(\mathbf{x}|c)$.

The Prior $P(c)$

- ... is simple: We estimate each **class's frequency**.
  - 'two of three e-mails are spam'
    $\rightarrow P(\text{spam}) = 0.67$
  - 'women and men are equally likely'
    $\rightarrow P(woman) = 0.5$

The class-conditional Density $P(\mathbf{x}|c)$

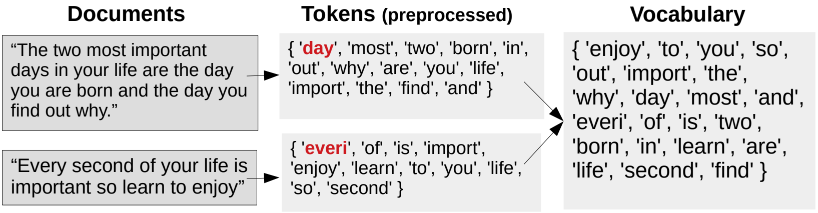
- ... is a bit more **tricky** to compute.
- For discrete features (e.g., text), $P(\mathbf{x}|c)$ is a probability table.
- For continuous features, $P(\mathbf{x}|c)$ is a probability density *(e.g., a normal distribution)*.

Application: Document Classification

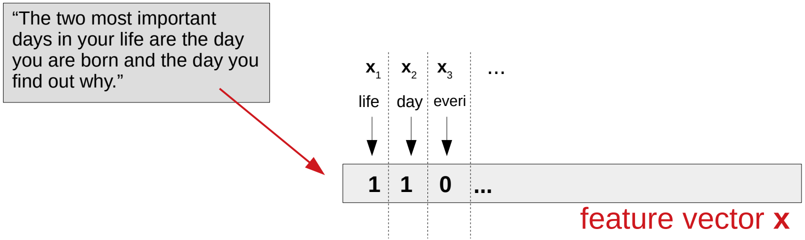
In this lecture, we'll apply a generative model *(more precisely, Naive Bayes)* to classify documents. There are many applications: *spam filtering, news classification, software issue routing, sentiment analysis, ...*

Feature Extraction (here, 'Bag-of-Words' features)

- ... transforms text documents into feature vectors $\mathbf{x}$.
- **Step 1**: Preprocessing *(lowercasing, stemming)*.
- **Step 2**: Collect all tokens in a vocabulary $\{t_1, ..., t_m\}$.



Application: Document Classification



Step 3: Compute the feature vector $\mathbf{x}$

- Every document is transformed to a boolean vector $\mathbf{x} = (x_1, \ldots, x_m) \in \{0, 1\}^m$.
- Each entry x_i is 1 if term t_i appears in the document *(and $x_i = 0$ otherwise)*.
- Note that the *order of terms* is neglected!

Generative Text Classification

Example Model

- $\mathbf{x}$ contains $m = 2$ boolean features
  - x_1 (is 'mom' in the e-mail?)
  - x_2 (is 'viagra' in the e-mail?)
- 2 classes, spam and ham.
- We assume $P(spam) = 10\%$.
- $P(\mathbf{x}|c)$ is a probability table *(right)*.

Class 1 (SPAM)

x_1(„mom“)	x_2(„viagra“)	$P(x_1,x_2\|spam)$
0	0	0.77
0	1	0.20
1	0	0.01
1	1	0.02

Class 2 (HAM)

x_1(“mom“)	x_2(„viagra“)	$P(x_1,x_2\|ham)$
0	0	0.18
0	1	0.02
1	0	0.78
1	1	0.02

„20% of all spam mails do **not** contain the term „mom“, but **do** contain the term „viagra“.

Applying the Model

- A new e-mail contains 'viagra' but not 'mom' ($\mathbf{x} = (0, 1)$):

$$P(spam, \mathbf{x}) = P(spam) \times P(\mathbf{x}|spam) = 0.1 \times 0.2 = 2\%.$$
$$P(ham, \mathbf{x}) = P(ham) \times P(\mathbf{x}|ham) = 0.9 \times 0.02 = 1.8\%.$$
$$P(spam|\mathbf{x}) = P(spam, \mathbf{x}) / [P(spam, \mathbf{x}) + P(ham, \mathbf{x})] = 52.6\%$$

- We decide the e-mail to be spam (but are not very confident).

Generative Methods: Naive Bayes

Problem with this approach?

- When m becomes large, the probability table becomes huge (2^m entries)!
- In practice, $\mathbf{x}$ is a vector with $> 10,000$ entries *(which terms do appear in the e-mail, which do not?)*.
- The probability tables would have $2^{10,000}$ entries!

m=1

„Viagra“	P(x|spam)
0	0.78
1	0.22

m=2

„Viagra“	„nigeria“	P(x|spam)
0	0	0.58
0	1	0.22
1	0	0.16
1	1	0.04

m=3

„Viagra“	„nigeria“	„mom“	P(x|spam)
0	0	0	0.35
0	0	1	0.13
0	1	0	0.17
0	1	1	0.03
1	0	0	0.19
1	0	1	0.03
1	1	0	0.09
1	1	1	0.01

… **m=10,000** ?

We need to learn ***each*** of these entries from a (limited) training set!

Naive Bayes

- Our goal is to simplify $P(\mathbf{x}|c)$!
- Approach : We assume the single terms' entries in $\mathbf{x}$ to be *independent* (hence *Naive* Bayes).

$$P(\mathbf{x}|c) = P(x_1|c) \cdot P(x_2|c) \cdot \ldots \cdot P(x_m|c).$$

- The decision rule becomes:

$$\begin{aligned} c^* &= \arg\max_c \; P(c|\mathbf{x}) \\ &= \arg\max_c \; \frac{P(c) \cdot P(\mathbf{x}|c)}{P(\mathbf{x})} && \text{// Bayes' rule} \\ &= \arg\max_c \; P(c) \cdot P(\mathbf{x}|c) && \text{// } P(\mathbf{x}) \text{ does not influence } c^* \\ &= \arg\max_c \; P(c) \cdot \prod_i P(x_i|c). && \text{// independence of features} \end{aligned}$$

Text Classifier (with Naive Bayes)

- ... same setting as above, but now Naive Bayes.
- We (still) assume $P(spam) = 10\%$.
- How there is one probability table per feature: $P(x_1|c), P(x_2|c)$.

Class 1 (SPAM)

x_1(„mom“)	$P(x_1\|spam)$
0	0.95
1	0.05

x_2(„viagra“)	$P(x_2\|spam)$
0	0.66
1	0.34

Class 2 (HAM)

x_1(„mom“)	$P(x_1\|ham)$
0	0.70
1	0.30

x_2(„viagra“)	$P(x_2\|ham)$
0	0.99
1	0.01

„34% of all spam mails do contain the term „viagra“ (independent of the term „mom“).

Applying the Model

- new e-mail: x_1=1 (viagra), x_2=0 (mom)

$$
\begin{aligned}
P(spam, \mathbf{x}) &= P(spam) \times P(x_1{=}1|spam) \times P(x_2{=}0|spam) \\
&= 0.1 \times 0.34 \times 0.95 = 3.23\%. \\
P(ham, \mathbf{x}) &= P(ham) \times P(x_1{=}1|ham) \times P(x_2{=}0|ham) \\
&= 0.9 \times 0.01 \times 0.7 = 0.63\%. \\
P(spam|\mathbf{x}) &= P(spam, \mathbf{x}) / [P(spam, \mathbf{x}) + P(ham, \mathbf{x})] = 83.7\%
\end{aligned}
$$

Naive Bayes

Remarks

- The entries to be learned decrease from 2^n to ... $2n$.
- We can estimate these $2n$ entries, even from limited-size training sets.
- Note that the independence assumption is usually heavily violated in text, as terms influence each other (e.g., $P(superbowl = 1) << P(superbowl = 1 \,|\, patriots = 1)$).

References I