Naive Bayes and Text Classification I

Introduction and Theory

Sebastian Raschka
se.raschka@gmail.com

October 4, 2014

Contents

1 Introduction

Starting more than half a century ago, scientists became very serious about addressing the question: "Can we build a model that learns from available data and automatically makes the right decisions and predictions?" Looking back, this sounds almost like a rhetoric question, and the answer can be found in numerous applications that are emerging from the fields of pattern classification, machine learning, and artificial intelligence.

Data from various sensoring devices combined with powerful learning algorithms and domain knowledge led to many great inventions that we now take for granted in our everyday life: Internet queries via search engines like Google, text recognition at the post office, barcode scanners at the supermarket, the diagnosis of diseases, speech recognition by Siri or Google Now on our mobile phone, just to name a few.

One of the sub-fields of *predictive modeling* is *supervised pattern classification*; supervised pattern classification is the task of training a model based on labeled training data which then can be used to assign a pre-defined class label to new objects. One example that we will explore throughout this article is spam filtering via naive Bayes classifiers in order to predict whether a new text message can be categorized as spam or not-spam. Naive Bayes classifiers, a family of classifiers that are based on the popular Bayes' probability theorem, are known for creating simple yet well performing models, especially in the fields of document classification and disease prediction.

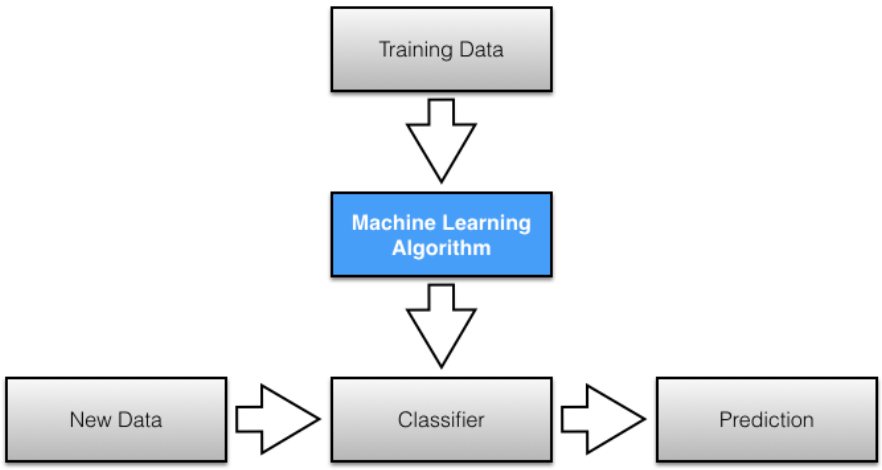

Figure 1: A simplified diagram of the general model building procedure for pattern classification.

A more detailed overview of predictive modeling can be found in my pre-

vious article *Predictive Modeling, Supervised Machine Learning, and Pattern Classification - The Big Picture.*

2 Naive Bayes Classification

2.1 Overview

Naive Bayes classifiers are linear classifiers that are known for being simple yet very efficient. The probabilistic model of naive Bayes classifiers is based on Bayes' theorem, and the adjective *naive* comes from the assumption that the features in a dataset are mutually independent. In practice, the independence assumption is often violated, but naive Bayes classifiers still tend to perform very well under this unrealistic assumption [1]. Especially for small sample sizes, naive Bayes classifiers can outperform the more powerful alternatives [2].

Being relatively robust, easy to implement, fast, and accurate, naive Bayes classifiers are used in many different fields. Some examples include the diagnosis of diseases and making decisions about treatment processes [3], the classification of RNA sequences in taxonomic studies [4], and spam filtering in e-mail clients [5]. However, strong violations of the independence assumptions and non-linear classification problems can lead to very poor performances of naive Bayes classifiers. We have to keep in mind that the type of data and the type problem to be solved dictate which classification model we want to choose. In practice, it is always recommended to compare different classification models on the particular dataset and consider the prediction performances as well as computational efficiency.

In the following sections, we will take a closer look at the probability model of the naive Bayes classifier and apply the concept to a simple toy problem. Later, we will use a publicly available SMS (text message) collection to train a naive Bayes classifier in Python that allows us to classify unseen messages as spam or ham.

2.2 Posterior Probabilities

In order to understand how naive Bayes classifiers work, we have to briefly recapitulate the concept of Bayesian probabilities. The probability model that was formulated by Thomas Bayes (1701-1761) is quite simple yet powerful; it can be written down in simple words as follows:

$$\text{posterior probability} = \frac{\text{conditional probability} \cdot \text{prior probibility}}{\text{evidence}} \tag{1}$$

Bayes' theorem forms the core of the whole concept of naive Bayes classification. The *posterior probability*, in the context of a classification problem, can be interpreted as: "What is the probability that a particular object belongs to class i given its observed feature values?" A more concrete example would be: "What is the probability that a person has diabetes given a certain

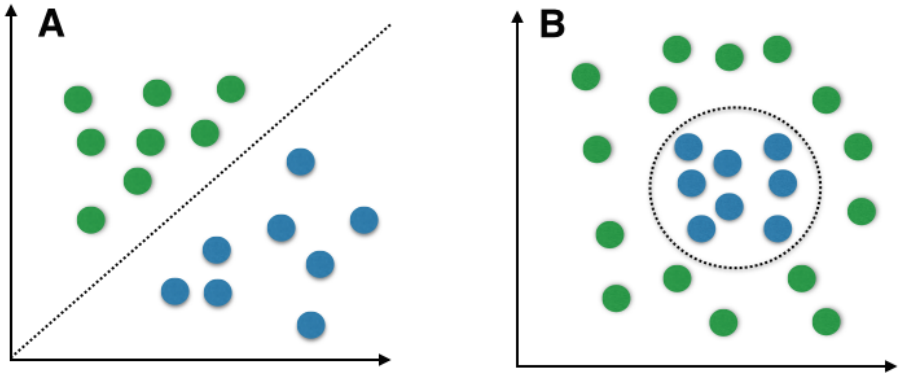

Figure 2: Linear (A) vs. non-linear problems (B). Random samples for two different classes are shown as colored spheres, and the dotted lines indicate the class boundaries that classifiers try to approximate by computing the decision boundaries. A non-linear problem (B) would be a case where linear classifiers, such as naive Bayes, would not be suitable since the classes are not linearly separable. In such a scenario, non-linear classifiers (e.g.,instance-based nearest neighbor classifiers) should be preferred.

value for a pre-breakfast blood glucose measurement and a certain value for a post-breakfast blood glucose measurement?"

$$P(\text{diabetes} \mid \mathbf{x}_i) \, , \quad \mathbf{x}_i = [90\text{mg/dl}, 145\text{mg/dl}] \tag{2}$$

Let

- $\mathbf{x}_i$ be the feature vector of sample i, $i \in \{1, 2, ..., n\}$,
- ω_j be the notation of class j, $j \in \{1, 2, ..., m\}$,
- and $P(\mathbf{x}_i \mid \omega_j)$ be the probability of observing sample $\mathbf{x}_i$ given that is belongs to class ω_j.

The general notation of the posterior probability can be written as

$$P(\omega_j \mid \mathbf{x}_i) = \frac{P(\mathbf{x}_i \mid \omega_j) \cdot P(\omega_j)}{P(\mathbf{x}_i)} \tag{3}$$

In the naive Bayes probability model, objects are classified based on the most probable hypothesis, and the *Maximum A Posteriori (MAP)* estimation can be used to formulate a decision rule given the training data. In other words, the assignment of a class label ω_j to a particular observation $\mathbf{x}_i$ is determined by the maximization of the posterior probability:

$$\text{predicted class label} \leftarrow \underset{j=1...,m}{\arg\max} \, P(\omega_j \mid \mathbf{x}_i) \tag{4}$$

In contrast to a *maximum-likelihood (ML)* approach for parameter estimation, MAP includes *prior knowledge* in order to make a prediction. To continue

with our example above, we can formulate the decision rule based on the posterior probabilities as follows:

$$\begin{aligned} &\text{person has diabetes if} \\ &P(\text{diabetes} \mid \mathbf{x}_i) \geq P(\text{not-diabetes} \mid \mathbf{x}_i), \\ &\text{else classify person as healthy.} \end{aligned} \tag{5}$$

2.3 Class-conditional Probabilities

One assumption that Bayes classifiers make is that the samples are *i.i.d.* The abbreviation *i.i.d.* stands for "independent and identically distributed" and describes random variables that are independent from one another and are drawn from a similar probability distribution. Independence means that the probability of one observation does not affect the probability of another observation (e.g., time series and network graphs are not independent). One popular example of *i.i.d.* variables is the classic coin tossing: The first coin flip does not affect the outcome of a second coin flip and so forth. Given a fair coin, the probability of the coin landing on "heads" is always 0.5 no matter of how often the coin if flipped.

An additional assumption of naive Bayes classifiers is the *conditional independence* of features. Under this *naive* assumption, the *class-conditional probabilities* or (*likelihoods*) of the samples can be directly estimated from the training data instead of evaluating all possibilities of $\mathbf{x}$. Thus, given a d-dimensional feature vector $\mathbf{x}$, the class conditional probability can be calculated as follows:

$$P(\mathbf{x} \mid \omega_j) = P(x_1 \mid \omega_j) \cdot P(x_2 \mid \omega_j) \cdot \ldots \cdot P(x_d \mid \omega_j) = \prod_{k=1}^{d} P(x_k \mid \omega_j) \tag{6}$$

Here, $P(\mathbf{x} \mid \omega_j)$ simply means: "How likely is it to observe this particular pattern $\mathbf{x}$ given that it belongs to class ω_j?" The "individual" likelihoods for every feature in the feature vector can be estimated via the maximum-likelihood estimate, which is simply a frequency in the case of categorical data:

$$\hat{P}(x_i \mid \omega_j) = \frac{N_{i,c}}{N_i} \quad (i = (1, ..., d)) \tag{7}$$

- $N_{i,c}$: Count of feature x_i in class ω_j.
- N_ic: Count of feature x_i in all classes.

To illustrate this concept with an example, let's assume that we have a collection of 500 documents where 100 documents are *spam* messages. Now, we want to calculate the class-conditional probability for a new message "Hello World" given that it is spam. Here, the pattern consists of two features: "hello" and

"world," and the the class-conditional probability is the product of the "probability of encountering 'hello' given the message is spam" — the probability of encountering "world" given the message is spam."

$$P(\mathbf{x} = [\text{hello, world}] \mid \omega = \text{spam}) = P(\text{hello} \mid \text{spam}) \cdot P(\text{world} \mid \text{spam}) \quad (8)$$

Using the training dataset of 500 documents, we can use the maximum-likelihood estimate to estimate those probabilities: We'd simply calculate how often the words occur in the corpus of all spam messages. E.g.,

$$\hat{P}(\mathbf{x} = [\text{hello, world}] \mid \omega = \text{spam}) = \frac{20}{100} \cdot \frac{2}{100} = 0.004 \quad (9)$$

However, with respect to the *naive* assumption of conditional independence, we notice a problem here: The *naive* assumption is that a particular word does not influence the chance of encountering other words in the same document. For example, given the two words "peanut" and "butter" in a text document, intuition tells us that this assumption is obviously violated: If a document contains the word "peanut" it will be more likely that it also contains the word "butter" (or "allergy"). In practice, the conditional independence assumption is indeed often violated, but naive Bayes classifiers are known to perform still well in those cases [6].

2.4 Prior Probabilities

In contrast to a frequentist's approach, the concept of *Bayesian inference* introduces an additional *prior probability* (or just *prior*) that can be interpreted as the *prior belief* or *a priori* knowledge.

$$\text{posterior probability} = \frac{\text{conditional probability} \cdot \text{prior probibility}}{\text{evidence}} \quad (10)$$

In the context of pattern classification, the prior probabilities are also called *class priors*, which describe "the general probability of encountering a particular class." In the case of spam classification, the priors could be formulated as

$$P(\text{spam}) = \text{"the probability that any new message is a spam message"} \quad (11)$$

and

$$P(\text{ham}) = 1 - P(\text{spam}). \quad (12)$$

If the priors are following a uniform distribution, the posterior probabilities will be entirely determined by the class-conditional probabilities and the evidence term. And since the evidence term is a constant, the decision rule will entirely depend on the class-conditional probabilities (similar to a frequentist's approach and maximum-likelihood estimate).

Eventually, the *a priori* knowledge can be obtained, e.g., by consulting a domain expert or by estimation from the training data (assuming that the training data is *i.i.d.* and a representative sample of the entire population. The maximum-likelihood estimate approach can be formulated as

$$\hat{P}(\omega_j) = \frac{N_{\omega_j}}{N_c} \tag{13}$$

- N_{ω_j}: Count of samples from class ω_j.
- N_c: Count of all samples.

And in context of *spam classification*:

$$\hat{P}(\text{spam}) = \frac{\text{\# of spam messages in training data}}{\text{\# of all mesages in training data}} \tag{14}$$

Figure figure3 illustrates the effect of the prior probabilities on the decision rule. Given an 1-dimensional pattern $\mathbf{x}$ (continuous attribute, plotted as "x" symbols) that follows a normal distribution and belongs to one out of two classes (*blue* and *green*). The patterns from the first class (ω_1 = blue) are drawn from a normal distribution with mean $x = 4$ and a standard deviation $\sigma = 1$. The probability distribution of the second class (ω_2 = green) is centered at x=10 with a similar standard deviation of $\sigma = 1$. The bell-curves denote the probability densities of the samples that were drawn from the two different normal distributions. Considering only the class conditional probabilities, the maximum-likelihood estimate in this case would be

$$\begin{aligned} &P(x = 4 \mid \omega_1) \approx 0.4 \text{ and } P(x = 10 \mid \omega_1) < 0.001 \\ &P(x = 4 \mid \omega_2) < 0.001 \text{ and } P(x = 10 \mid \omega_2) \approx 0.4. \end{aligned} \tag{15}$$

Now, given uniform priors, that is $P(\omega_1) = P(\omega_2) = 0.5$, the decision rule would be entirely dependent on those class-conditional probabilities, so that the decision rule would fall directly between the two distributions

$$P(x \mid \omega_1) = P(x \mid \omega_2). \tag{16}$$

However, if the prior probability was $P(\omega_1) > 0.5$, the decision region of class ω_1 would expand as shown in Figure figure3. In the context of spam classification, this could be interpreted as encountering a new message that only contains words which are equally likely to appear in *spam* or *ham* messages. In this case, the decision would be entirely dependent on *prior knowledge*, e.g., we could assume that a random message is in 9 out of 10 cases not *spam* and therefore classify the new message as *ham*.

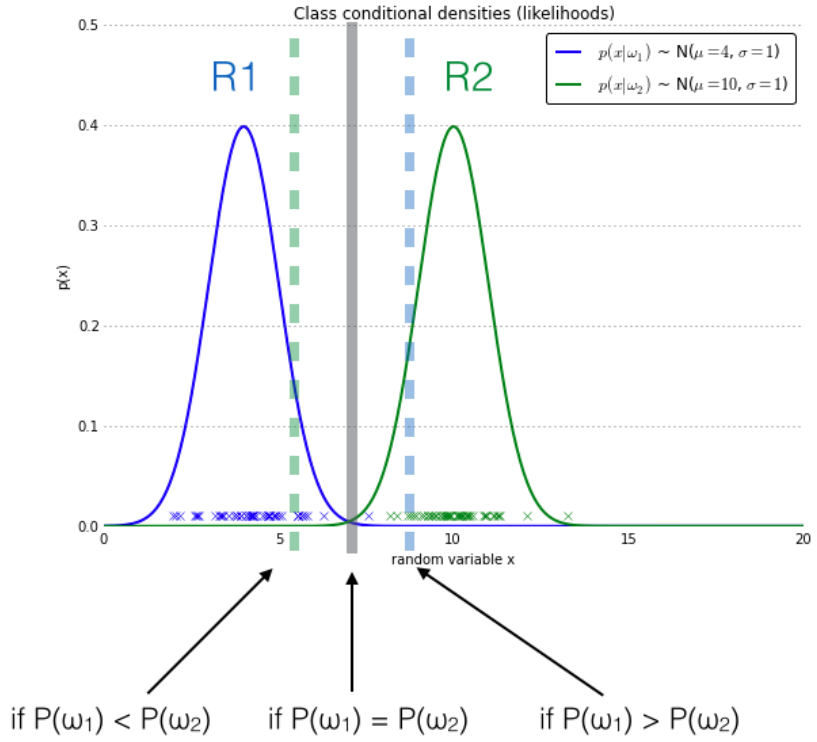

Figure 3: The effect of prior probabilities on the decision regions. The figure shows a 1-dimensional random sample from two different classes (blue and green crosses). The data points for both classes are normally distributed with standard deviation 1, and the bell curves denote the class conditional probabilities. If the class priors are equal, the decision boundary of a naive Bayes classifier is placed at the center between both distributions (gray bar). An increase of the class-conditional probability of the blue class (ω_2) leads to an extension of the decision region R1 by moving the decision boundary (blue-dotted bar) towards the other class and vice versa.

2.5 Evidence

After defining the *class-conditional probability* and *prior probability*, there is only one term missing in order to compute *posterior probability*, that is the *evidence*.

$$\text{posterior probability} = \frac{\text{conditional probability} \cdot \text{prior probibility}}{\text{evidence}} \tag{17}$$

The evidence $P(\mathbf{x})$ can be understood as the probability of encountering a particular pattern $\mathbf{x}$ independent from the class label. Given the more formal

definition of posterior probability

$$P(\omega_j \mid \mathbf{x}_i) = \frac{P(\mathbf{x}_i \mid \omega_j) \cdot P(\omega_j)}{P(\mathbf{x}_i)}, \tag{18}$$

the evidence can be calculated as follows (ω_j^C stands for "complement" and basically translate to "**not** class ω_j."):

$$P(\mathbf{x}_i) = P(\mathbf{x}_i \mid \omega_j) \cdot P(\omega_j) \cdot P(\mathbf{x}_i \mid \omega_j^C) \cdot P(\omega_j^C) \tag{19}$$

Although the evidence term is required to accurately calculate the posterior probabilities, it can be removed from the *decision rule* "Classify sample $\mathbf{x}_i$ as ω_1 if $P(\omega_1 \mid \mathbf{x}_i) > P(\omega_2 \mid \mathbf{x}_i)$ else classify the sample as ω_2," since it is merely a scaling factor:

$$\frac{P(\mathbf{x}_i \mid \omega_1) \cdot P(\omega_1)}{P(\mathbf{x}_i)} > \frac{P(\mathbf{x}_i \mid \omega_2) \cdot P(\omega_2)}{P(\mathbf{x}_i)} \tag{20}$$

$$\propto P(\mathbf{x}_i \mid \omega_1) \cdot P(\omega_1) > P(\mathbf{x}_i \mid \omega_2) \cdot P(\omega_2) \tag{21}$$

2.6 Multinomial Naive Bayes - A Toy Example

After covering the basics concepts of a naive Bayes classifier, the *posterior probabilities* and *decision rules*, let us walk through a simple toy example based on the training set shown in Figure figure4.

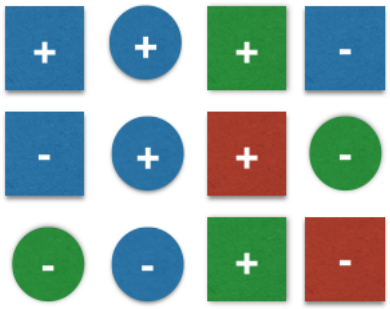

Figure 4: A simple toy dataset of 12 samples 2 different classes $+, -$. Each sample consists of 2 features: color and geometrical shape.

Let

- ω_j be the class labels: $\omega_j \in \{+, -\}$
- and $\mathbf{x}_i$ be the 2-dimensional feature vectors: $\mathbf{x}_i = [x_{i1} \;\; x_{i2}]$, $x_{i1} \in \{\text{blue}, \text{green}, \text{red}, \text{yellow}\}$, $x_{i2} \in \{\text{circle}, \text{square}\}$.

The 2 class labels are $\omega_j \in \{+, -\}$ and the feature vector for sample i can be written as

$$\begin{aligned} &\mathbf{x}_i = [x_{i1}\ x_{i2}] \\ &\text{for } i \in \{1, 2, ..., n\}, \quad \text{with } n = 12 \\ &\text{and } x_{i1} \in \{\text{blue}, \text{green}, \text{red}, \text{yellow}\}, \quad x_{i2} \in \{\text{circle}, \text{square}\} \end{aligned} \tag{22}$$

The task now is to classify a new sample — pretending hat we don't know that its true class label is "+":

Figure 5: A new sample from class + and the features $\mathbf{x}$ = [blue, square] that is to be classified using the training data in Figure figure4.

2.6.1 Maximum-Likelihood Estimates

The *decision rule* can be defined as

$$\begin{aligned} &\text{Classify sample as } + \text{ if} \\ &P(\omega = + \mid \mathbf{x} = [\text{blue, square}]) \geq P(\omega = \text{-} \mid \mathbf{x} = [\text{blue, square}]) \\ &\text{else classify sample as } - . \end{aligned} \tag{23}$$

Under the assumption that the samples are *i.i.d*, the *prior probabilities* can be obtained via the maximum-likelihood estimate (i.e., the frequencies of how often each class label is represented in the training dataset):

$$\begin{aligned} P(+) &= \frac{7}{12} = 0.58 \\ P(\text{-}) &= \frac{5}{12} = 0.42 \end{aligned} \tag{24}$$

Under the *naive* assumption that the features "color" and "shape" are mutually independent, the *class-conditional probabilities* can be calculated as a simple product of the individual conditional probabilities.

Via maximum-likelihood estimate, e.g., $P(\text{blue} \mid -)$ is simply the frequency of observing a "blue" sample among all samples in the training dataset that belong to class "+."

$$\begin{aligned} P(\mathbf{x} \mid +) &= P(\text{blue} \mid +) \cdot P(\text{square} \mid +) = \frac{3}{7} \cdot \frac{5}{7} = 0.31 \\ P(\mathbf{x} \mid -) &= P(\text{blue} \mid -) \cdot P(\text{square} \mid -) = \frac{3}{5} \cdot \frac{3}{5} = 0.36 \end{aligned} \tag{25}$$

Now, the *posterior probabilities* can be simply calculated as the product of the class-conditional and prior probabilities:

$$\begin{aligned} P(+ \mid \mathbf{x}) &= P(\mathbf{x} \mid +) \cdot P(+) = 0.31 \cdot 0.58 = 0.18 \\ P(- \mid \mathbf{x}) &= P(\mathbf{x} \mid -) \cdot P(-) = 0.36 \cdot 0.42 = 0.15 \end{aligned} \tag{26}$$

2.6.2 Classification

Putting it all together, the new sample can be classified by plugging in the posterior probabilities into the decision rule:

$$\begin{aligned} &\text{If } P(+ \mid \mathbf{x}) \geq P(\text{-} \mid \mathbf{x}) \\ &\text{classify as } +, \\ &\text{else classify as } - \end{aligned} \tag{27}$$

Since $0.18 > 0.15$ the sample can be classified as $+$. Taking a closer look at the calculation of the posterior probabilities, this simple example demonstrates the effect of the prior probabilities affected on the decision rule. If the prior probabilities were equal for both classes, the new pattern would be classified as $-$ instead of $+$. This observation also underlines the importance of *representative* training datasets; in practice, it is usually recommended to additionally consult a domain expert in order to define the prior probabilities.

2.6.3 Additive Smoothening

The classification was straight-forward given the sample in Figure figure5. A trickier case is a sample that has a "new" value for the color attribute that is not present in the training dataset, e.g., *yellow*, as shown in Figure figure5.

Figure 6: A new sample from class + and the features $\mathbf{x}$ = [yellow, square] that is to be classified using the training data in Figure figure4.

If the color *yellow* does not appear in our training dataset, the class-conditional probability will be 0, and as a consequence, the posterior probability will also be 0 since the posterior probability is the product of the prior and class-conditional probabilities.

$$\begin{aligned} P(\omega_1 \mid \mathbf{x}) &= 0 \cdot 0.42 = 0 \\ P(\omega_2 \mid \mathbf{x}) &= 0 \cdot 0.58 = 0 \end{aligned} \tag{28}$$

In order to avoid the problem of *zero* probabilities, an additional smoothening term can be added to the *multinomial Bayes* model. The most common variants of additive smoothening are the so-called *Lidstone smoothening* ($\alpha < 1$) and *Laplace smoothening* ($\alpha = 1$).

$$\hat{P}(x_i \mid \omega_j) = \frac{N_{i,c} + \alpha}{N_i + \alpha\, d} \quad (i = (1, ..., d)) \tag{29}$$

where

- $N_{i,c}$: Count of observing feature x_i in class ω_j.
- N_ic: Count of observing feature x_i in all classes.
- α: Parameter for additive smoothening.
- d: Dimensionality of the feature vector $\mathbf{x} = [x_1, ..., x_d]$.

3 Naive Bayes and Text Classification

This section will introduce some of the main concepts and procedures that are needed to apply the naive Bayes model to text classification tasks. Although the examples are mainly concerning a two 2-class problem — classifying text messages as *spam* or *ham* — the same approaches are applicable to multi-class problems such as classification of documents into different topic areas (e.g., "Computer Science", "Biology", "Statistics", "Economics", "Politics", etc.).

3.1 The Bag of Words Model

One of the most important sub-tasks in pattern classification are *feature extraction* and *selection*; the three main criteria of good features are listed below:

- *Salient.* The features are important and meaningful with respect to the problem domain.
- *Invariant.* Invariance is often described in context of image classification: The features are insusceptible to distortion, scaling, orientation, etc. A nice example is given by C. Yao *et al.* in *Rotation-Invariant Features for Multi-Oriented Text Detection in Natural Images* [7].
- *Discriminatory.* The selected features bear enough information to distinguish well between patterns when used to train the classifier.

Prior to fitting the model and using machine learning algorithms for training, we need to think about how to best represent a text document as a feature vector. A commonly used model in *Natural Language Processing* is the so-called *bag of words* model. The idea behind this model really is as simple as it sounds. First comes the creation of the *vocabulary* — the collection of all different words that occur in the training set and each word is associated with a count of how

it occurs. This vocabulary can be understood as a set of non-redundant items where the order doesn't matter. Let D_1 and D_2 be two documents in a training dataset:

- D_1: "Each state has its own laws."
- D_2: "Every country has its own culture."

Based on these two documents, the vocabulary could be written as

$$V = \{each : 1, state : 1, has : 2, its : 2, own : 2, \\ laws : 1, every : 1, country : 1, culture : 1\} \tag{30}$$

The vocabulary can then be used to construct the d-dimensional feature vectors for the individual documents where the dimensionality is equal to the number of different words in the vocabulary ($d = |V|$). This process is called *vectorization*.

Table 1: Bag of words representation of two sample documents D_1 and D_2.

	each	state	has	its	own	laws	every	country	culture
$\mathbf{x}_{D1}$	1	1	1	1	1	1	0	0	0
$\mathbf{x}_{D2}$	0	0	1	1	1	0	1	1	1
$\sum$	1	1	2	2	2	1	1	1	1

Given the example in Table table1 one question is whether the 1s and 0s of the feature vectors are binary counts (1 if the word occurs in a particular document, 0 otherwise) or absolute counts (how often the word occurs in each document). The answer depends on which probabilistic model is used for the naive Bayes classifier: The *Multinomial* or *Bernoulli* model — more on the probabilistic models in Section subsection3.3 and Section subsection3.4.

3.1.1 Tokenization

Tokenization describes the general process of breaking down a text corpus into individual elements that serve as input for various natural language processing algorithms. Usually, tokenization is accompanied by other optional processing steps, such as the removal of stop words and punctuation characters, stemming or lemmatizing, and the construction of *n-grams*. Below is an example of a simple but typical tokenization step that splits a sentence into individual words, removes punctuation, and converts all letters to lowercase.

Table 2: Example of tokenization.

A swimmer likes swimming, thus he swims.						
↓						
a	swimmer	likes	swimming	thus	he	swims

3.1.2 Stop Words

Stop words are words that are particularly common in a text corpus and thus considered as rather un-informative (e.g., words such as *so*, *and*, *or*, *the*, ..."). One approach to stop word removal is to search against a language-specific stop word dictionary. An alternative approach is to create a *stop list* by sorting all words in the entire text corpus by frequency. The stop list — after conversion into a *set* of non-redundant words — is then used to remove all those words from the input documents that are ranked among the top n words in this stop list.

Table 3: Example of stop word removal.

A swimmer likes swimming, thus he swims.					
↓					
swimmer	likes	swimming	,	swims	.

3.1.3 Stemming and Lemmatization

Stemming describes the process of transforming a word into its root form. The original stemming algorithm was developed my Martin F. Porter in 1979 and is hence known as *Porter stemmer* [8].

Table 4: Example of Porter stemming.

A swimmer likes swimming, thus he swims.							
↓							
a	swimmer	like	swim	,	thu	he	swim	.

Stemming can create non-real words, such as "thu" in the example above. In contrast to stemming, *lemmatization* aims to obtain the canonical (grammatically correct) forms of the words, the so-called *lemmas*. Lemmatization is computationally more difficult and expensive than stemming, and in practice, both stemming and lemmatization have little impact on the performance of text classification [9].

Table 5: Example of lemmatization.

A swimmer likes swimming, thus he swims.								
↓								
A	swimmer	like	swimming	,	thus	he	swim	.

The stemming and lemmatization examples were created by using the Python NLTK library (http://www.nltk.org).

3.1.4 *N*-grams

In the *n-gram* model, a token can be defined as a sequence of n items. The simplest case is the so-called *unigram* (1-gram) where each word consists of exactly one word, letter, or symbol. All previous examples were unigrams so far. Choosing the optimal number n depends on the language as well as the particular application. For example, Andelka Zecevic found in his study that n-grams with $3 \leq n \leq 7$ were the best choice to determine authorship of Serbian text documents [10]. In a different study, the n-grams of size $4 \leq n \leq 8$ yielded the highest accuracy in authorship determination of English text books [11] and Kanaris *e. al.* report that n-grams of size 3 and 4 yield good performances in anti-spam filtering of e-mail messages [12].

- unigram (1-gram):

a	swimmer	likes	swimming	thus	he	swims

- bigram (2-gram):

a swimmer	swimmer likes	likes swimming	swimming thus	...

- trigram (3-gram):

a swimmer likes	swimmer likes swimming	likes swimming thus	...

3.2 The Decision Rule for Spam Classification

In context of spam classification the decision rule of a naive Bayes classifier based on the posterior probabilities can be expressed as

$$\begin{aligned} &\text{if } P(\mathbf{x} \mid \omega = \text{spam}) \geq P(\mathbf{x} \mid \omega = \text{ham}) \text{ classify as spam,} \\ &\text{else classify as ham.} \end{aligned} \tag{31}$$

As described in Section subsection2.2 the posterior probability is the product of the class-conditional probability and the prior probability; the evidence term in the denominator can be dropped since it is constant for both classes.

$$\begin{aligned} P(\mathbf{x} \mid \omega = \text{spam}) &= P(\omega = \text{spam} \mid \mathbf{x}) \cdot P(\text{spam}) \\ P(\mathbf{x} \mid \omega = \text{ham}) &= P(\omega = \text{ham} \mid \mathbf{x}) \cdot P(\text{ham}) \end{aligned} \tag{32}$$

The prior probabilities can be obtained via the maximum-likelihood estimate based on the frequencies of spam and ham messages in the training dataset:

$$\hat{P}(\omega = \text{spam}) = \frac{\text{\# of spam msg.}}{\text{\# of all msg.}}$$
$$\hat{P}(\omega = \text{ham}) = \frac{\text{\# of ham msg.}}{\text{\# of all msg.}} \tag{33}$$

Assuming that the words in every document are conditionally independent (according to the *naive* assumption), two different models can be used to compute the class-conditional probabilities: The *Multi-variate Bernoulli* model (Section subsection3.3) and the *Multinomial* model (Section subsection3.4).

3.3 Mutli-variate Bernoulli Naive Bayes

The *Multi-variate Bernoulli* model is based on binarydata: Every token in the feature vector of a document is associated with the value 1 or 0. The feature vector has d dimensions where d is the number of words in the whole vocabulary (in Section subsection3.1); the value 1 means that the word occurs in the particular document, and 0 means that the word does not occur in this document. Empirical studies showed that the mutli-variate Bernoulli model is inferior to the multinomial Bayes model, and the latter has been shown to reduce the error by 27% on average [13]. The Bernoulli trials can be written as

$$P(\mathbf{x} \mid \omega_j) = \prod_{i=1}^{m} P(x_i \mid \omega_j)^b \cdot (1 - P(x_i \mid \omega_j))^{(1-b)} \quad (b \in 0, 1). \tag{34}$$

Let $\hat{P}(x_i \mid \omega_j)$ be the maximum-likelihood estimate that a particular word (or token) x_i occurs in class ω_j.

$$\hat{P}(x_i \mid \omega_j) = \frac{df_{xi,y} + 1}{df_y + 2} \tag{35}$$

where

- $df_{xi,y}$ is the number of documents in the training dataset that contain the feature x_i and belong to class ω_j.
- $df_{xi,y}$ is the number of documents in the training dataset that contain x_i and belong to class ω_j.
- df_y is the number of documents in the training dataset that belong to class ω_j.
- $+1$ and $+2$ are the parameters of *Laplace smoothening* (Section subsubsection2.6.3).

3.4 Multinomial Naive Bayes

3.4.1 Term Frequency

A alternative approach to characterize text documents — rather than binary values — is the *term frequency (tf(t, d))*. The term frequency is typically defined as the number of times a given term t (i.e., word or token) appears in a document d (this approach is sometimes also called *raw frequency*). In practice, the term frequency is often normalized by dividing the raw term frequency by the document length.

$$\text{normalized term frequency} = \frac{tf(t, d)}{n_d} \tag{36}$$

where

- $tf(t, d)$: Raw term frequency (the count of term t in document d).
- n_d: The total number of terms in document d.

The term frequencies can then be used to compute the maximum-likelihood estimate based on the training data to estimate the class-conditional probabilities in the multinomial model:

$$\hat{P}(x_i \mid \omega_j) = \frac{\sum tf(x_i, d \in \omega_j) + \alpha}{\sum N_{d \in \omega j} + \alpha \cdot V} \tag{37}$$

where

- x_i: A word from the feature vector $\mathbf{x}$ of a particular sample.
- $\sum tf(x_i, d \in \omega_j)$: The sum of raw term frequencies of word x_i from all documents in the training sample that belong to class ω_j.
- $\sum N_{d \in \omega j}$: The sum of all term frequencies in the training dataset for class ω_j.
- α: An addititive smoothening parameter ($\alpha = 1$ for Laplace smoothening).
- V: The size of the vocabulary (number of different words in the training set).

The class-conditional probability of encountering the text $\mathbf{x}$ can be calculated as the product from the likelihoods of the individual words (under the *naive* assumption of conditional independence).

$$P(\mathbf{x} \mid \omega_j) = P(x_1 \mid \omega_j) \cdot P(x_2 \mid \omega_j) \cdot \ldots \cdot P(x_n \mid \omega_j) = \prod_{i=1}^{m} P(x_i \mid \omega_j) \tag{38}$$

3.4.2 Term Frequency - Inverse Document Frequency (Tf-idf)

The *term frequency - inverse document frequency (Tf-idf)* is another alternative for characterizing text documents. It can be understood as a weighted *term frequency*, which is especially useful if stop words have not been removed from the text corpus. The Tf-idf approach assumes that the importance of a word is inversely proportional to how often it occurs across all documents. Although Tf-idf is most commonly used to rank documents by relevance in different text mining tasks, such as page ranking by search engines, it can also be applied to text classification via naive Bayes.

$$\text{Tf-idf} = tf_n(t, d) \cdot idf(t) \tag{39}$$

Let $tf_n(d, f)$ be the normalized term frequency, and idf, the inverse document frequency, which can be calculated as follows

$$idf(t) = \log\left(\frac{n_d}{n_d(t)}\right), \tag{40}$$

where

- n_d: The total number of documents.
- $n_d(t)$: The number of documents that contain the term t.

4 Variants of the Naive Bayes Model

So far, we have seen two different models for categorical data, namely, the multi-variate Bernoulli (Section subsection3.3) and multinomial (Section subsection3.4) models — and two different approaches for the estimation of class-conditional probabilities. In Section subsection4.1, we will take a brief look at a third model: *Gaussian naive Bayes*.

4.1 Continuous Variables

Text classification is a typical case of categorical data, however, naive Bayes can also be used on continuous data. The *Iris* flower data set would be a simple example for a supervised classification task with continuous features: The Iris dataset contains widths and lengths of petals and sepals measured in centimeters. One strategy for dealing with continuous data in naive Bayes classification would be to discretize the features and form distinct categories or to use a Gaussian kernel to calculate the class-conditional probabilities. Under the assumption that the probability distributions of the features follow a normal (Gaussian) distribution, the Gaussian naive Bayes model can be written as follows

$$P(x_{ik} \mid \omega) = \frac{1}{\sqrt{2\pi\sigma_\omega^2}} \exp\left(-\frac{(x_{ik} - \mu_\omega)^2}{2\sigma_\omega^2}\right), \tag{41}$$

where μ (the sample mean) and σ (the standard deviation) are the parameters that are to be estimated from the training data. Under the naive Bayes assumption of conditional independence, the class-conditional probability can than be computed as the product of the individual probabilities:

$$P(\mathbf{x}_i \mid \omega) = \prod_{k=1}^{d} P(\mathbf{x}_{ik} \mid \omega) \tag{42}$$

4.2 Eager and Lazy Learning Algorithms

Being an *eager learner*, naive Bayes classifiers are known to be relatively fast in classifying new instances. Eeager learners are learning algorithms that learn a model from a training dataset as soon as the data becomes available. Once the model is learned, the training data does not have to be re-evaluated in order to make a new prediction. In case of eager learners, the computationally most expensive step is the model building step whereas the classification of new instances is relatively fast.

Lazy learners, however, memorize and re-evaluate the training dataset for predicting the class label of new instances. The advantage of *lazy learning* is that the model building (training) phase is relatively fast. On the other hand, the actual prediction is typically slower compared to eager learners due to the re-evaluation of the training data. Another disadvantage of lazy learners is that the training data has to be retained, which can also be expensive in terms of storage space. A typical example of a lazy learner would be a *k-nearest neighbor* algorithm: Every time a new instance is encountered, the algorithm would evaluate the k-nearest neighbors in order to decide upon a class label for the new instance, e.g., via the *majority rule* (i.e., the assignment of the class label that occurs most frequently amongst the k-nearest neighbors).

References

[1] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, pages 41–46, 2001.

[2] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[3] Joanna Kazmierska and Julian Malicki. Application of the naïve bayesian classifier to optimize treatment decisions. *Radiotherapy and Oncology*, 86(2):211–216, 2008.

[4] Qiong Wang, George M Garrity, James M Tiedje, and James R Cole. Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Applied and environmental microbiology*, 73(16):5261–5267, 2007.

[5] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105, 1998.

[6] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

[7] Cong Yao, Xin Zhang, Xiang Bai, Wenyu Liu, Yi Ma, and Zhuowen Tu. Rotation-invariant features for multi-oriented text detection in natural images. *PloS one*, 8(8):e70173, 2013.

[8] Martin F Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[9] Michal Toman, Roman Tesar, and Karel Jezek. Influence of word normalization on text classification. *Proceedings of InSciT*, pages 354–358, 2006.

[10] Andelka Zecevic. N-gram based text classification according to authorship. In *Student Research Workshop*, pages 145–149, 2011.

[11] Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, volume 3, pages 255–264, 2003.

[12] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools*, 16(06):1047–1067, 2007.

[13] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.