Naive Bayes

Name: Nour El-Din Salah
Code: 4624

Introduction

Naive Bayes is a family of algorithms based on applying Bayes theorem with a strong (naive) assumption, that every feature is independent of the others, in order to predict the category of a given sample. They are probabilistic classifiers, therefore will calculate the probability of each category using Bayes theorem, and the category with the highest probability will be output. Naive Bayes classifiers have been successfully applied to many domains, particularly Natural Language Processing (NLP).

Theory Behind Bayes' Theorem

```
P(H|E) = (P(E|H) * P(H)) / P(E)
```

Where:

- **P(H|E)** is the probability of hypothesis **H** given the event **E**, a posterior probability.
- **P(E|H)** is the probability of event **E** given that the hypothesis **H** is true.
- **P(H)** is the probability of hypothesis **H** being true (regardless of any related event), or prior probability of **H**.
- **P(E)** is the probability of the event occurring (regardless of the hypothesis).

Advantages

- Naive Bayes is a simple and easy to implement algorithm. Because it assumes that all features are independent
- Naive Bayes works well with numerical and categorical data.

Limitations

- Given the construction of the theorem, it does not work well when you are missing certain combination of values in your training data. In other words, if you have no occurrences of a class label and a certain attribute value together (e.g. class="spam", contains="$$$") then the frequency-based probability estimate will be zero. Given Naive-Bayes' conditional independence assumption, when all the probabilities are multiplied you will get zero.

- Naive Bayes works well as long as the categories are kept simple. For instance, it works well for problems involving keywords as features **(e.g. spam detection)**, but it does not work when the relationship between words is important **(e.g. sentiment analysis)**.

Examples

We are given a sentence" **A very close game"**, a training set of five sentences, and their corresponding category **(Sports or Not Sports)**.

We need to calculate the probability of both "A very close game is **Sports**", as well as it's **Not Sports**. The one with the higher probability will be the result.

Feature Engineering

In the first step, feature engineering, we focus on extracting features of text. We need numerical features as input for our classifier. So, an intuitive choice would be **word frequencies**, i.e., counting the occurrence of every word in the document.

$$P(sports|a\,very\,close\,game) = \frac{P(a\,very\,close\,game|sports) \times P(sports)}{P(a\,very\,close\,game)}$$

We can rewrite the probability we wish to calculate accordingly:

$$P(a\,very\,close\,game) = P(a) \times P(very) \times P(close) \times P(game)$$

$$P(a\,very\,close\,game|Sports) = P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports)$$

Laplace smoothing

A technique for smoothing categorical data. A small-sample correction, or pseudo-count, will be incorporated in every probability estimate. Consequently, no probability will be zero. this is a way of regularizing Naive Bayes.

Given an observation x = (x1, ..., xd) from a multinomial distribution with N trials and parameter vector θ = (θ1, ..., θd), a "smoothed" version of the data gives the estimator:

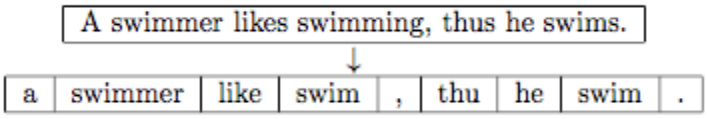
$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d),$$

where the pseudocount α > 0 is the **smoothing parameter** (α = 0 corresponds to no smoothing).

techniques for improving the basic Naive Bayes classifier

- **Removing stop words**. These are common words that don't really add anything to the classification, such as a, able, either, else, ever and so on. So, for our purposes, The election was over would be election over and a very close game would be very close game.

- **Stemming.** 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**.**



- **Using 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. Choosing the optimal number n depends on the language as well as the particular application.

- 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	...

Results

	precision	recall	f1-score	support
0	0.82	0.79	0.80	579
1	0.60	0.65	0.63	286
accuracy			0.74	865
macro avg	0.71	0.72	0.72	865
weighted avg	0.75	0.74	0.75	865