Naive Bayes Classifiers

Michela Papandrea

University of Applied Sciences and Arts of Southern Switzerland

michela.papandrea@supsi.ch

2.12.2021

Naive Bayes classifiers

Family of classifiers based on Bayes' Theorem

(+) they tend to be faster in training wrt linear models
- they learn parameters by looking at each feature individually
- they collect simple per-class statistics from each feature

(+) easy to build and particularly useful for very large data sets

(+) might outperform even highly sophisticated classification methods

(–) often provide generalization performance slightly worse than linear classifiers

- all classifiers of the family assume *independence among predictors*
  - the presence of a particular feature in a class is unrelated to the presence of any other feature
  - even if the features depend on each other, all of these properties independently contribute to the probability that a sample belongs to a specific class c (that is why it is known as '*Naive*').

Bayes Theorem

Bayes Theorem

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{1}$$

- $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

Naive Bayes algorithm

Example: Naive Bayes algorithm steps

DATASET: training data set of weather and corresponding target variable `Play` (suggesting possibilities of playing)

CLASSIFICATION TASK:we have to classify whether players will play or not based on weather condition.

(1) Convert the data set into a frequency table

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Naive Bayes algorithm

Example: Naive Bayes algorithm steps

(2) Create Likelihood table by finding the probabilities of the predictors given class, and the probability of the classes

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

(3) Use Naive Bayes equation to calculate the posterior probability for each class.
$\rightarrow$ The class with the highest posterior probability is the outcome of prediction.

Naive Bayes algorithm

Practice 1

Players are going to play if weather is sunny. Is this statement is correct?

Naive Bayes algorithm

Solution - Practice 1

We can solve it using above discussed method of posterior probability.

$$P(Yes|Sunny) = \frac{P(Sunny|Yes) * P(Yes)}{P(Sunny)}$$

Where:

- $P(Sunny|Yes) = \frac{3}{9} = 0.33$
- $P(Sunny) = \frac{5}{14} = 0.36$
- $P(Yes) = \frac{9}{14} = 0.64$

Hence:

$$P(Yes|Sunny) = \frac{0.33 * 0.64}{0.36} = 0.60$$

$\Rightarrow$ *Yes* is the class with higher probability.

Naive Bayes algorithm

If the feature vector is represented as

$$X = (x_1, x_2, x_3, ..., x_n)$$

then

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

For each possible entry in the dataset, the denominator does not change $\Rightarrow$ the denominator can be removed and a proportionality can be introduced

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

In case of a multiclass classification problem, we need to find the class y with maximum probability

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Naive Bayes algorithm

Practice 2

Players are going to play if weather is 'Overcast' and temperature is 'Mild'. What is the probability that they are going to play?

	weather	temp	play
0	Sunny	Hot	No
1	Sunny	Hot	No
2	Overcast	Hot	Yes
3	Rainy	Mild	Yes
4	Rainy	Cool	Yes
5	Rainy	Cool	No
6	Overcast	Cool	Yes
7	Sunny	Mild	No
8	Sunny	Cool	Yes
9	Rainy	Mild	Yes
10	Sunny	Mild	Yes
11	Overcast	Hot	Yes
12	Rainy	Mild	No

Naive Bayes algorithm

Practice 2

Features need to be encoded first ('Label Encoder').

```
1 from sklearn.naive_bayes import GaussianNB
2
3 #Create a Gaussian Classifier
4 model = GaussianNB()
5
6 # Train the model using the training sets
7 model.fit(features.values,label)
8
9 #Predict Output
10 predicted= model.predict([[0,2]]) # 0:Overcast, 2:Mild
11 print("Predicted Value:{}".format(predicted))
```

Naive Bayes Algorithms

There exist different types of Naive Bayes classifiers:

- **Bernoulli**
  - it assumes **binary data**
  - counts how often every feature is not zero for each class
  - **Example** in a document data (text classification problem): '*bag of words*' model where the 1s and 0s are "*word occurs in the document*" and "*word does not occur in the document*" respectively.
- **Multinominal**
  - it assumes **count data** (that is, each feature represents an integer count of something)
  - **Example** in a document data: "*count how often each word occurs in the document*"
  - takes into account the average value of each feature for each class
- **Gaussian**
  - can be applied to any **continuous data**
  - it assumes that features follow a normal distribution
  - stores the average value as well as the standard deviation of each feature for each class
  - the formula for conditional probability changes in this case

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Naive Bayes Algorithms

Prediction

- A data point is compared to the statistics for each of the classes
- the best matching class is predicted

- **Gaussian** is mostly used on *very high-dimensional data*
- **Multinomial** and **Bernoulli** are widely used for *sparse count data*

  example: text.
- **Multinomial** usually performs better than **Bernoulli** on datasets with a relatively large number of nonzero features

  example: large documents

Naive Bayes Algorithms

NB parameters

- **Multinomial** and **Bernoulli** have a single parameter α which controls model complexity.
- The way α works is that the algorithm adds to the data α many virtual data points that have positive values for all the features.
  $\rightarrow$ this results in a *smoothing* of the statistics.
  - large $\alpha \Rightarrow$ more smoothing, resulting in less complex models.
- The algorithm's performance is relatively robust to the setting of alpha, meaning that setting alpha is not critical for good performance
- tuning α usually improves accuracy somewhat.

Naive Bayes Algorithms

Strengths

- Naive Bayes models are very fast to train and to predict
- the training procedure is easy to understand.
- The models work very well with high-dimensional sparse data
- The models are relatively robust to the parameters.
- Naive Bayes models are great baseline models and are often used on very large datasets, where training even a linear model might take too long.
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It perform well in case of *categorical* input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed (strong assumption).

Naive Bayes Algorithms

Weaknesses

- If categorical variable has a category which was not observed in the data set, then model will assign a 0 (zero) probability and will be unable to make a prediction.
  $\rightarrow$ this is often known as "*Zero Frequency*". To solve this, we can use the smoothing technique.
- Naive Bayes is also known as a *bad estimator* so the probability outputs are not to be taken too seriously
- Another limitation of NB is the assumption of *independent predictors*: in real life, it is almost impossible that we get a set of predictors which are completely independent

References

Andreas C. Müller & Sarah Guido (2017)

Introduction to Machine Learning with Python

Chapter 2: Supervised Learning (pp. 70 – 72)

Published by O'Reilly Media, Inc..