**6 простых шагов для освоения наивного байесовского алгоритма (с примером кода на Python)**

23.09.2015

Теги: [байесовский алгоритм Python](http://datareview.info/tag/bayesovskiy-algoritm-python/)[задача классификации](http://datareview.info/tag/zadacha-klassifikatsii/)[ИТ-room](http://datareview.info/tag/it-room/)[наивный байесовский алгоритм](http://datareview.info/tag/naivnyiy-bayesovskiy-algoritm/)[переводы](http://datareview.info/tag/%D0%BF%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%B4%D1%8B/)

Представьте себе следующую ситуацию: вы работаете над задачей классификации, уже создали набор гипотез и сформировали признаки. Через час заказчики хотят увидеть первый вариант модели.

Перед вами обучающий набор данных, содержащий несколько сотен тысяч элементов и большое количество признаков. Что вы будете делать? На вашем месте я бы воспользовался **наивным байесовским алгоритмом** (naive Bayes algorithm, НБА), который превосходит по скорости многие другие алгоритмы классификации. В его основе лежит теорема Байеса.

В этой статье мы рассмотрим основы этого алгоритма, чтобы вы получили в свое распоряжение эффективный инструмент для работы с большими наборами данных. Кроме того, начинающим изучать Python будет полезен пример кода, представленный в конце статье.

Содержание

1. Что такое наивный байесовский алгоритм?
2. Как он работает?
3. Положительные и отрицательные.
4. 4 приложения наивного байесовского алгоритма.
5. Как создать базовую модель на его основе с помощью Python?
6. Советы по оптимизации модели.

Что такое наивный байесовский алгоритм?

Наивный байесовский алгоритм – это алгоритм классификации, основанный на [теореме Байеса](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D0%91%D0%B0%D0%B9%D0%B5%D1%81%D0%B0) с допущением о независимости признаков. Другими словами, НБА предполагает, что наличие какого-либо признака в классе не связано с наличием какого-либо другого признака. Например, фрукт может считаться яблоком, если он красный, круглый и его диаметр составляет порядка 8 сантиметров. Даже если эти признаки зависят друг от друга или от других признаков, в любом случае они вносят независимый вклад в вероятность того, что этот фрукт является яблоком. В связи с таким допущением алгоритм называется «наивным».

Модели на основе НБА достаточно просты и крайне полезны при работе с очень большими наборами данных. При своей простоте НБА способен превзойти даже некоторые сложные алгоритмы классификации.

Теорема Байеса позволяет рассчитать апостериорную вероятность *P(c|x)* на основе *P(c)*, *P(x)* и *P(x|c)*.



На рисунке выше:

* *P(c|x)* – апостериорная вероятность данного класса *c* (т.е. данного значения целевой переменной) при данном значении признака *x*.
* *P(c)* – априорная вероятность данного класса.
* *P(x|c)* – правдоподобие, т.е. вероятность данного значения признака при данном классе.
* *P(x)* – априорная вероятность данного значения признака.

Как работает наивный байесовский алгоритм?

Давайте рассмотрим пример. Ниже представлен обучающий набор данных, содержащий один признак «Погодные условия» (weather) и целевую переменную «Игра» (play), которая обозначает возможность проведения матча. На основе погодных условий мы должны определить, состоится ли матч. Чтобы сделать это, необходимо выполнить следующие шаги.

Шаг 1. Преобразуем набор данных в частотную таблицу (frequency table).

Шаг 2. Создадим таблицу правдоподобия (likelihood table), рассчитав соответствующие вероятности. Например, вероятность облачной погоды (overcast) составляет 0,29, а вероятность того, что матч состоится (yes) – 0,64.



*Sunny – Солнечная погода
Rainy – Дождливая погода
Overcast – Облачная погода*

Шаг 3. С помощью теоремы Байеса рассчитаем апостериорную вероятность для каждого класса при данных погодных условиях. Класс с наибольшей апостериорной вероятностью будет результатом прогноза.

**Задача**. Состоится ли матч при солнечной погоде (sunny)?

Мы можем решить эту задачу с помощью описанного выше подхода.

*P(Yes | Sunny) = P(Sunny | Yes) \* P(Yes) / P(Sunny)*

Здесь мы имеем следующие значения:

*P(Sunny | Yes) = 3 / 9 = 0,33*

*P(Sunny) = 5 / 14 = 0,36*

*P(Yes) = 9 / 14 = 0,64*

Теперь рассчитаем *P(Yes | Sunny)*:

*P(Yes | Sunny) = 0,33 \* 0,64 / 0,36 = 0,60*

Значит, при солнечной погоде более вероятно, что матч состоится.

Аналогичным образом с помощью НБА можно прогнозировать несколько различных классов на основе множества признаков. Этот алгоритм в основном используется в области классификации текстов и при решении задач многоклассовой классификации.

Положительные и отрицательные стороны наивного байесовского алгоритма

**Положительные стороны:**

* Классификация, в том числе многоклассовая, выполняется легко и быстро.
* Когда допущение о независимости выполняется, НБА превосходит другие алгоритмы, такие как логистическая регрессия (logistic regression), и при этом требует меньший объем обучающих данных.
* НБА лучше работает с категорийными признаками, чем с непрерывными. Для непрерывных признаков предполагается нормальное распределение, что является достаточно сильным допущением.

**Отрицательные стороны:**

* Если в тестовом наборе данных присутствует некоторое значение категорийного признака, которое не встречалось в обучающем наборе данных, тогда модель присвоит нулевую вероятность этому значению и не сможет сделать прогноз. Это явление известно под названием «нулевая частота» (zero frequency). Данную проблему можно решить с помощью сглаживания. Одним из самых простых методов является сглаживание по Лапласу (Laplace smoothing).
* Хотя НБА является хорошим классификатором, значения спрогнозированных вероятностей не всегда являются достаточно точными. Поэтому не следует слишком полагаться на результаты, возвращенные методом *predict\_proba*.
* Еще одним ограничением НБА является допущение о независимости признаков. В реальности наборы полностью независимых признаков встречаются крайне редко.

4 приложения наивного байесовского алгоритма

* **Классификация в режиме реального времени.** НБА очень быстро обучается, поэтому его можно использовать для обработки данных в режиме реального времени.
* **Многоклассовая классификация.** НБА обеспечивает возможность многоклассовой классификации. Это позволяет прогнозировать вероятности для множества значений целевой переменной.
* **Классификация текстов, фильтрация спама, анализ тональности текста.** При решении задач, связанных с классификацией текстов, НБА превосходит многие другие алгоритмы. Благодаря этому, данный алгоритм находит широкое применение в области фильтрации спама (идентификация спама в электронных письмах) и анализа тональности текста (анализ социальных медиа, идентификация позитивных и негативных мнений клиентов).
* **Рекомендательные системы.** Наивный байесовский классификатор в сочетании с[коллаборативной фильтрацией](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%84%D0%B8%D0%BB%D1%8C%D1%82%D1%80%D0%B0%D1%86%D0%B8%D1%8F) (collaborative filtering) позволяет реализовать рекомендательную систему. В рамках такой системы с помощью методов машинного обучения и интеллектуального анализа данных новая для пользователя информация отфильтровывается на основании спрогнозированного мнения этого пользователя о ней.

Как создать базовую модель на основе наивного байесовского алгоритма с помощью Python?

В этом нам поможет библиотека scikit-learn. Данная библиотека содержит три типа моделей на основе наивного байесовского алгоритма:

* **Gaussian**(нормальное распределение). Модель данного типа используется в случае непрерывных признаков и предполагает, что значения признаков имеют нормальное распределение.
* **Multinomial** (мультиномиальное распределение). Используется в случае дискретных признаков. Например, в задаче классификации текстов признаки могут показывать, сколько раз каждое слово встречается в данном тексте.
* **Bernoulli** (распределение Бернулли). Используется в случае двоичных дискретных признаков (могут принимать только два значения: 0 и 1). Например, в задаче классификации текстов с применением подхода «мешок слов» (bag of words) бинарный признак определяет присутствие (1) или отсутствие (0) данного слова в тексте.

В зависимости от набора данных вы можете выбрать подходящую модель из описанных выше. Далее представлен пример кода для модели Gaussian.

Пример кода на Python

#Import Library of Gaussian Naive Bayes model

from sklearn.naive\_bayes import GaussianNB

import numpy as np

#assigning predictor and target variables

x= np.array([[-3,7],[1,5], [1,2], [-2,0], [2,3], [-4,0], [-1,1], [1,1], [-2,2], [2,7], [-4,1], [-2,7]])

Y = np.array([3, 3, 3, 3, 4, 3, 3, 4, 3, 4, 4, 4])

#Create a Gaussian Classifier

model = GaussianNB()

# Train the model using the training sets

model.fit(x, Y)

#Predict Output

predicted= model.predict([[1,2],[3,4]])

print predicted

Output: ([3,4])

Мы рассмотрели базовую модель на основе НБА. Эффективность базовой модели можно повысить посредством настройки параметров, а также с помощью других методов, которые мы обсудим ниже. Чтобы получить более подробную информацию о классификации текстов с помощью НБА, я рекомендую вам обратиться к [следующему документу](http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note07-2up.pdf).

Советы по оптимизации модели

* Если значения непрерывных признаков не обладают нормальным распределением, тогда с помощью соответствующего преобразования необходимо привести их к такому распределению.
* Если тестовый набор данных имеет проблему «нулевой частоты»()евой сатотый набор данных имеет проблему «лениюных ии текста с помощью данного алгоритма, я рекомендую вам обратиться к , необходимо применить сглаживание по Лапласу.
* Если два признака имеют высокую корреляцию, один из них следует удалить, иначе они будут вносить удвоенный вклад, завышая тем самым свою значимость.
* Наивный байесовский классификатор имеет набор параметров, доступных для настройки. Например, *alpha=1* – активирует сглаживание по Лапласу, *fit\_prior=[True | False]*– определяет, обучать ли модель априорным вероятностям классов. Полный список параметров можно найти [здесь](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html#sklearn.naive_bayes.MultinomialNB). Я рекомендую сосредоточиться на предобработке данных и отборе признаков.
* Следует отметить, что в случае НБА использование ансамблевых методов, таких как бэггинг (bagging) и бустинг (boosting), не дает результатов. Данные подходы направлены на уменьшение дисперсии, что неприменимо по отношению к НБА.

Заключение

В данной статье мы рассмотрели наивный байесовский алгоритм. Основной областью применения этого алгоритма являются задачи классификации. Если вы хорошо разобрались в этом материале, значит, вы уже сделали первый шаг к освоению алгоритма. Теперь все, что вам нужно, – это практика.

Перед применением наивного байесовского алгоритма я рекомендую вам уделять особое внимание предобработке данных и отбору признаков. В следующей статье мы более подробно рассмотрим классификацию текстов с помощью данного алгоритма.

*По материалам:*[*analyticsvidhya.com*](http://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/?utm_source=FBPage&utm_medium=Social&utm_campaign=150914)

**КОММЕНТАРИЕВ 5**

* [**Комментарии2**](http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/#commentlist-container)
* [**Пингбэки3**](http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/#pinglist-container)
1. **Артём**

[14.10.2015 в 22:09](http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/#comment-48128)

В коде есть опечатка, нужно заменить:
model.fit(x, y)
на:
model.fit(x, Y)

[Ответить](http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/?replytocom=48128#respond)

* + **Лариса Шурига**

[15.10.2015 в 19:14](http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/#comment-48183)

Спасибо, поправили

[Ответить](http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/?replytocom=48183#respond)