Теги

- Machine Learning
- JavaScript
- Data Science
- Artificial Intelligence
- Web Development
- Software Development
- Python
- Coding
- Deep Learning
- React
- AI
- Software Engineering
- Nodejs
- Java
- Front End Development
- Tech
- Computer Science
- Typescript
- Algorithms
- Development
- Angular
- NLP
- Data
- Neural Networks
- Startup
- CSS
- Programming Languages
- Reactjs
- Computer Vision
- Education
- Tutorial
- Productivity
- HTML
- Learning To Code
- Learning
- Javascript Tips
- Code
- Open Source

Вопросы

- Новые
- Популярные

Наивный Байес в деталях Объяснение

Home / Публикации / Наивный Байес в деталях Объяснение

В этой статье мы изучим метод классификации на основе вероятностей, называемый Наивным Байесом.

В этом блоге мы рассмотрим следующие темы:

1. Что такое Наивный Байес?
2. Математика алгоритма наивного Байеса
3. Наивный байесовский пример
4. Наивный байесовский анализ текстовых данных и сглаживание по Лапласу
5. Наивный Байес для данных большой размерности
6. Компромисс между дисперсией Байса, важность признаков и интерпретация наивного байесовского метода
7. Типы наивных байесовских классификаторов
8. Плюсы и минусы наивного Байеса
9. Приложения наивного байесовского алгоритма

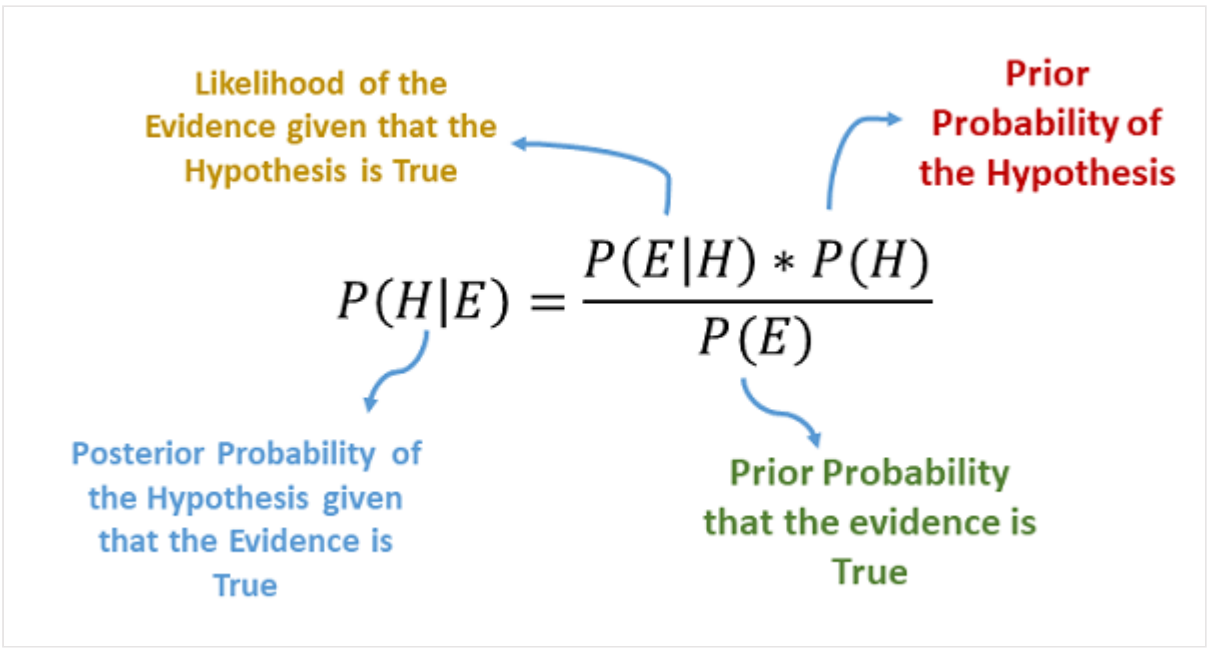
1. Что такое Наивный Байес?

Наивный байесовский алгоритм — это вероятностный алгоритм, используемый в машинном обучении для задач классификации. Он основан на теореме Байеса, которая гласит, что вероятность события при наличии предварительных знаний о связанных событиях может быть рассчитана с использованием условной вероятности.

Наивный Байес «наивен», потому что предполагает, что характеристики точки данных независимы друг от друга. Это часто неверно для реальных данных, но предположение упрощает расчеты и все же может давать хорошие результаты на практике.

Теорема Байеса:

Теорема Байеса описывает вероятность события, основанную на предварительном знании условий, которые могут быть связаны с этим событием.



Что делает наивный байесовский алгоритм «наивным»?

Наивный байесовский классификатор предполагает, что функции, которые мы используем для прогнозирования цели, независимы и не влияют друг на друга. Хотя в реальных данных функции зависят друг от друга при определении цели, но это игнорируется наивным байесовским классификатором.

Хотя предположение о независимости никогда не бывает верным в реальных данных, на практике оно часто работает хорошо. чтобы он назывался **"Наивный".**

2. Математика наивного байесовского алгоритма

Учитывая вектор признаков X=(x1,x2,...,xn) и переменную класса y, теорема Байеса утверждает, что:

Вопросы по теме

Получение количества ошибок в шаблоне с тегами шаблона

Подсчитайте, сколько записей находится в базе данных с интервалом в 5 минут.

Как заставить слайдер двигаться автоматически

Как отправить SMTP-сообщение с помощью Python?

Составление ролей в reStructuredText

RxKotlin - Неправильная подписка, наблюдайте за изменением потока для субъекта вне активности?

Как запускать скрипт Python онлайн каждые N минут?

Фоновые изображения CSS в MVC3 — продолжение

Необходимо заменить символы с диакритическими знаками / диакритическими знаками / не ASCII в файле фиксированной ширины на пробел

SSRS Поворот столбца

Выберите столбцы из кадра данных в другой кадр данных на основе типа данных столбца в Apache Spark Scala

Панель действий Шерлок Темная тема со светлой темой + Темная панель действий

Служба Spring MVC REST - встроенный в maven Jetty дает 404

$$P(y|X) = \frac{P(X|y) * P(y)}{P(X)}$$

Нас интересует вычисление апостериорной вероятности P(y | X) из вероятности P(X | y) и априорных вероятностей P(y), P(X).

Используя цепное правило, вероятность P(X I y) можно разложить следующим образом:

$$P(X|y) = P(x_1, x_2, \ldots, x_n|y)$$

$$= P(x_1|x_2, \ldots, x_n, y) * P(x_2|x_3, \ldots, x_n, y) \ldots P(x_n|y)$$

но из-за допущения Наива об условной независимости условные вероятности не зависят друг от друга.

$$P(X|y) = P(x_1|y) * P(x_2|y) \ldots . P(x_n|y)$$

Таким образом, по условной независимости имеем:

$$P(y|X) = \frac{P(x_1|y) * P(x_2|y) \ldots . P(x_n|y) * P(y)}{P(x_1) * P(x_2) \ldots P(x_n)}$$

А поскольку знаменатель остается постоянным для всех значений, апостериорная вероятность может быть:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

Наивный байесовский классификатор сочетает эту модель с решающим правилом. Одно общее правило — выбирать наиболее вероятную гипотезу; это известно как максимальное апостериорное правило или правило принятия решения MAP.

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

3. Наивный байесовский пример:

Давайте объясним это на примере, чтобы было понятно:

Рассмотрим вымышленный набор данных, описывающий погодные условия для игры в гольф. Учитывая погодные условия, каждый кортеж классифицирует условия как подходящие («Да») или непригодные («Нет») для игры в гольф.

Вот табличное представление нашего набора данных.

Apache Pig 0.12.0 на Hue не выполняет предварительную обработку операторов, как ожидалось

Как добавить скрипт в ‹head› добавить скрипт в конец ‹body›

Как добиться sudo su - root и запустить всю команду в ansible

Имеют ли SIM-карты NFC внутреннюю антенну? Как установить апплеты на SIM-карты?

Изменить заголовок при открытии диалога из нокаутирующей привязки

Как показать общее количество онлайн-друзей в шаблоне (django)?

Включите очень большое число в C++

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Набор данных разделен на две части: **матрица признаков** и **вектор отклика**.

- Матрица признаков содержит все векторы (строки) набора данных, в которых каждый вектор состоит из значений **зависимых признаков**. В приведенном выше наборе данных функциями являются «Перспективы», «Температура», «Влажность» и «Ветер».
- Вектор ответа содержит значение **переменной класса** (прогноз или вывод) для каждой строки матрицы признаков. В приведенном выше наборе данных имя переменной класса — «Играть в гольф».

Этап обучения:

На этапе обучения нам нужно вычислить таблицу вероятностей из обучающих данных,

нам нужно найти,

(1) P(outlook = O / Play Golf=b);

где O ∈ **{Солнечно, Пасмурно, Дождь}, b ∈ {да, нет}**

(2) ***P(температура = t / Play Golf=b);***

где t ∈ **{Горячий, Мягкий, Прохладный**

(3) ***P(Влажность = h / Играть в гольф=b);***

где h ∈ **{высокий, нормальный**

(4) ***P(ветер = w / Играть в гольф=b);***

где w ∈ **{True, False}**

Этап классификации:

Если мы получим новый экземпляр,

x’ = (Обзор = солнечно, температура = прохладно, влажность = высокая, ветер = правда)

что такое класс x’?

Таким образом, **вероятность игры в гольф определяется**:

P(Играть в гольф = да / x’)

= (P(Обзор = солнечно/Играть в гольф = да) * P(Температура = Прохладно/Играть в гольф = да) * P(Влажность = Высокая/Играть в гольф = да) * P(Ветер = Правда/Играть в гольф = да) * P(Играть в гольф = да)) / P(x')

Вероятность не играть в гольф определяется:

P(Играть в гольф = Нет/x’)

= (P(Обзор = солнечно/Играть в гольф = Нет) * P(Температура = Прохладно/Играть в гольф = Нет) * P(Влажность = Высокая/Играть в гольф = Нет) * P(Ветер = Правда/Играть в гольф = Нет) * P(Играть в гольф = Нет)) / P(x')

Поскольку знаменатель P(x’) является общим для обеих вероятностей, мы можем игнорировать P(x’) и найти пропорциональные вероятности как:

Итак, **вероятность игры в гольф**:

P(Играть в гольф = да / x’)

= (P(Обзор = солнечно/Играть в гольф = да) * P(Температура = Прохладно/Играть в гольф = да) * P(Влажность = Высокая/Играть в гольф = да) * P(Ветер = Правда/Играть в гольф = да) * P(Играть в гольф = да))

= (2/9)*(3/9)*(3/9)*(3/9)*(9/14)

=0.0053

вероятность не играть в гольф:

P(Играть в гольф = Нет/x’)

= (P(Обзор = солнечно/Играть в гольф = Нет) * P(Температура = Прохладно/Играть в гольф = Нет) * P(Влажность = Высокая/Играть в гольф = Нет) * P(Ветер = Правда/Играть в гольф = Нет) * P(играть в гольф = нет))

= (3/5)*(1/5)*(4/5)*(3/5)*(5/14)

=0.0205

здесь P(Играть в гольф = Нет/x’) › P(Играть в гольф = да/x’)

Таким образом, прогноз о том, что в гольф будут играть, — «Нет».

мы видели, как Наивный Байес хорошо работает с категориальными данными,

Теперь мы увидим Naive Bayes на текстовых данных.

4. Наивный байесовский анализ текстовых данных и сглаживание по Лапласу:

4.1 Наивный байесовский анализ текстовых данных:

Наивный байес хорошо работает с текстовыми данными,

например, спам-фильтр (электронная почта: спам/не спам), обзор (+ve/-ve) являются приложениями наивного Байеса.

например, у нас есть такие обзорные данные,

наша задача — предсказать, будет ли отзыв +ve/-ve.

Задание: сравнить P(y=1/text(i)) и P(y=0/text(i)) для заданного text(i). В зависимости от того, что выше, мы выбираем этот класс как класс заданного текста (i).

прежде всего, мы должны выполнить все этапы предварительной обработки текстовых данных, такие как удаление стоп-слов, стемминга, n-грамм. после применения всех этих шагов мы получаем набор слов, затем вычисляем двоичный набор слов.
теперь у нас есть текст → {w1,w2,w3....wd}

Итак, P(y=1/текст(i))
= P(y=1/w1,w2,w3...wd)
∝ P(y=1)*P(w1/y= 1)*P(w2/y=1).....P(wd/y=1)

то же самое для P(y=0/текст(i))
=P(y=0/w1,w2,w3...wd)
∝ P(y=0)*P(w1/y= 0)*P(w2/y=0).....P(wd/y=0)

мы можем вычислить,
P(wi/y=0)
= (количество точек данных, содержащих wi и y=0)/(количество точек данных с y=0)

P(wi/y=1)
= (количество точек данных, содержащих wi и y=1)/(количество точек данных с y=1)

Таким образом, мы можем применить Наивный Байес к текстовым данным.

Примечание.В задачах классификации текста **Наивный байесовский анализ** является очень хорошей **основой**. поэтому для задачи классификации текста **Наивный байесовский алгоритм** находится в **эталоне** по сравнению с другими алгоритмами.

проблема :

Давайте возьмем пример классификации текста, где задача состоит в том, чтобы классифицировать, является ли отзыв положительным или отрицательным. Мы строим таблицу правдоподобия на основе данных обучения. При запросе обзора мы используем значения таблицы правдоподобия, но **что, если слово в обзоре отсутствует в наборе обучающих данных?**

Проверка запроса = w1 w2 w3 w'

У нас есть четыре слова в нашем обзоре запроса, и давайте предположим, что в обучающих данных присутствуют только w1, w2 и w3. Таким образом, у нас будет вероятность для этих слов. Чтобы рассчитать, является отзыв положительным или отрицательным, мы сравниваем P(положительный|отзыв) и P(отрицательный|отзыв).

В таблице правдоподобия у нас есть P(w1|положительный результат), P(w2|положительный результат), P(w3|положительный результат) и P(положительный результат). **Ой, подождите, а где P(w'|положительное)?**

Если слово отсутствует в обучающем наборе данных, то у нас нет его вероятности. **Что нам делать?**

Подход 1 – игнорируйте термин P(w'|положительный)

Игнорирование означает, что мы присваиваем ему значение 1, что означает, что вероятность появления w' в положительном P(w'|положительном) и отрицательном отзыве P(w'|отрицательном) равна 1. Такой подход кажется логически неверным.

Подход 2. В модели «мешок слов» мы подсчитываем количество слов. Вхождения слова w' в обучении равны 0. Согласно этому

P(w'|положительный)=0 и P(w'|отрицательный)=0, но это сделает как P(положительный|отзыв), так и P(отрицательный|отзыв) равными 0, поскольку мы умножаем все вероятности. **Это проблема нулевой вероятности**. **Итак, как решить эту проблему?**

4.2 Лапласово сглаживание:

Сглаживание по Лапласу — это метод сглаживания, который решает проблему нулевой вероятности в наивном байесовском методе. Используя сглаживание Лапласа, мы можем представить P(w'|positive) как

Здесь
альфапредставляет параметр сглаживания,
K представляет количество измерений (признаков) в данных, а
>Nобозначает количество отзывов с y=положительным.

Если мы выберем значение альфа! = 0 (не равное 0), вероятность больше не будет равна нулю, даже если слово отсутствует в наборе обучающих данных.

Интерпретация изменения альфы

Допустим, слово w встречается 3 с y = положительным в обучающих данных. Предположим, что в нашем наборе данных есть 2 функции, то есть K = 2 и N = 100 (общее количество положительных отзывов).

Случай 1 –когда альфа=1

P(w'|положительный) = 3/102

Случай 2 –когда альфа = 100

P(w'|положительный) = 103/300

Случай 3 –когда альфа=1000

P(w'|положительный) = 1003/2100

По мере увеличения альфа вероятность правдоподобия приближается к равномерному распределению (0,5). В большинстве случаев альфа = 1 используется для устранения проблемы нулевой вероятности.

Короче говоря, сглаживание Лапласа — это метод сглаживания, который помогает решить проблему нулевой вероятности в наивном байесовском алгоритме машинного обучения. Использование более высоких значений альфа подтолкнет вероятность к значению 0,5, т. Е. Вероятность слова, равная 0,5, как для положительных, так и для отрицательных отзывов. Поскольку мы не получаем от этого много информации, это нежелательно. Поэтому предпочтительно использовать альфа=1.

5. Наивный Байес для данных большой размерности:

Наивный байесовский подход хорошо работает с текстовыми данными большой размерности,
но для числовой стабильности приходится использовать логарифмическую вероятность.

Узнайте больше о **логарифмической вероятности здесь.**

6. Компромисс между дисперсией Байса, важность характеристик и интерпретация наивного байесовского метода:

6.1 Компромисс смещения и дисперсии:

В наивном байесовском методе **α (гиперпараметр)** сглаживания Лапласа определяет недообучение и переоснащение.

маленький α → высокая дисперсия → переоснащение

большое значение α → высокое смещение → недостаточное соответствие

поэтому выберите правильный **α**, используя простую **перекрестную проверку/k-fold CV.**

6.2 Важность функции:

Во многих алгоритмах, таких как KNN, мы должны вычислять важность функции, используя прямой выбор функций или другие методы,
Но в наивном байесовском методе важность функции определяется/получается непосредственно из модели.

для класса +ve: найти слова (wi) с наибольшим значением P(wi/y=1)
для класса -ve: найти слова (wi) с наибольшим значением P(wi/y=0)

оба получены из модели

1. отсортировать все wi на основе P(wi/y=1) в порядке убывания,
   wi с высоким значением P(wi/y=1) → Важные слова/функции при определении того, что точка данных принадлежит +ve классу.
2. отсортировать все wi на основе P(wi/y=0) в порядке убывания,
   wi с высоким значением P(wi/y=0) → Важные слова/функции в определении того, что точка данных принадлежит -ve классу.

6.3 Интерпретация:

В Наивном Байесе мы можем легко интерпретировать нашу модель, используя вероятность/вероятность.

7. Типы наивных байесовских классификаторов:

- **Полиномиальное:** векторы признаков представляют частоты, с которыми определенные события генерируются полиномиальным распределением. Например, подсчитайте, как часто каждое слово встречается в документе. Это модель событий, обычно используемая для классификации документов.
- **Бернулли**. Как и полиномиальная модель, эта модель популярна для задач классификации документов, где используются характеристики бинарного термина (то есть слово встречается в документе или нет), а не частота термина (то есть частота появления слова). слово в документе).
- **Гауссово:** используется в классификации и предполагает, что признаки подчиняются нормальному распределению.

8. Плюсы и минусы наивного Байеса:

Плюсы:

1. В текстовой классификации Наивный байес является базовым/эталонным.
2. Наивный байесовский метод интерпретируется и придает значение функциям.
3. Сложность во время выполнения и во время обучения невелика, пространство во время выполнения также невелико.
4. Когда наивное байесовское предположение об условной независимости верно, оно будет сходиться быстрее, чем дискриминационные модели, такие как логистическая регрессия.

Минусы:

1. Предположение о независимых предикторах/признаках. Наивный Байес неявно предполагает, что все атрибуты взаимно независимы, что почти невозможно найти в реальных данных.
2. В текстовых данных наивный байесовский подход может легко переобучиться, если мы не будем выполнять сглаживание по Лапласу, выбрав правильный **α** с помощью перекрестной проверки.
3. Наивный байесовский подход хорошо работает для категориальных функций, но для функций с реальным значением наивный байесовский метод мало что может использовать.

9. Применение алгоритма наивного Байеса:

- Прогноз в реальном времени.
- Классификация текста/ Фильтрация спама/ Анализ тональности.
- Обнаружение языка и т. д.

Спасибо за прочтение!

пожалуйста! Не забывайте хлопать, если вы ясно поняли тему

Пожалуйста, пишите комментарии, если вы обнаружите что-то неправильное, или вы хотите поделиться дополнительной информацией по теме, обсуждаемой выше.

Naive Bayes | Machine Learning | Classification | Laplace Smoothing | Data Science

04.07.2023