Лаб.4. Заглянем глубже: наивная байесовская классификация

Наивные байесовские модели — группа исключительно быстрых и простых алгоритмов классификации, зачастую подходящих для наборов данных очень высоких размерностей. В силу их быстроты и столь небольшого количества настраиваемых параметров они оказываются очень удобны в качестве грубого эталона для задач классификации. В данной лабораторной работе мы наглядно объясним функционирование наивных байесовских классификаторов вместе с двумя примерами их работы на некоторых наборах данных.

Байесовская классификация

Наивные байесовские классификаторы основаны на байесовских методах классификации, в основе которых лежит теорема Байеса — уравнение, описывающее связь условных вероятностей статистических величин. В байесовской классификации нас интересует поиск вероятности метки (категории) при определенных заданных признаках, являющихся результатами наблюдений/экспериментов, обозначенной $P(\text{L} \mid \text{признаков})$. Теорема Байеса позволяет выразить это в терминах величин, которые мы можем вычислить напрямую:

Один из способов выбора между двумя метками (L_1 и L_2) — вычислить отношение апостериорных вероятностей для каждой из них:

Все, что нам теперь нужно, — модель, с помощью которой можно было бы вычислить $P(\text{признаков} \mid L_i)$ для каждой из меток. Подобная модель называется *порождающей моделью* (generative model), поскольку определяет гипотетический случайный процесс генерации данных. Задание порождающей модели для каждой из меток/категорий — основа обучения подобного байесовского классификатора. Обобщенная версия подобного шага обучения — непростая задача, но мы упростим ее, приняв некоторые упрощающие допущения о виде модели.

Именно на этом этапе возникает слово «наивный» в названии «наивный байесовский классификатор»: сделав очень «наивное» допущение относительно порождающей модели для каждой из меток/категорий, можно будет отыскать грубое приближение порождающей модели для каждого класса, после чего перейти к байесовской классификации. Различные виды наивных байесовских классификаторов основываются на различных «наивных» допущениях относительно данных, мы рассмотрим несколько из них в следующих разделах. Начнем с обычных импортов:

```
In[1]: %matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()
```

Гауссов наивный байесовский классификатор

Вероятно, самый простой для понимания наивный байесовский классификатор — Гауссов. В этом классификаторе допущение состоит в том, что данные всех категорий взяты из простого нормального распределения. Пускай у нас имеются следующие данные (рис. 4.1):

```
In[2]: from sklearn.datasets import make_blobs
```

```
X, y = make_blobs(100, 2, centers=2, random_state=2, cluster_std=1.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu');
```

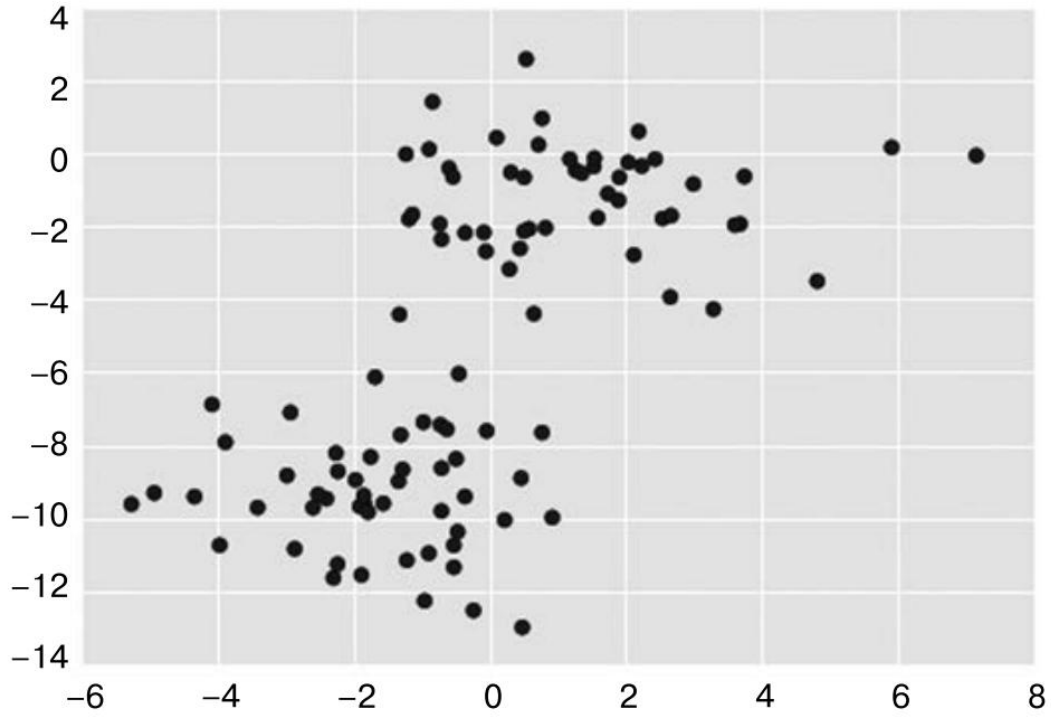

Рис. 4.1. Данные для наивной байесовской классификации

Один из самых быстрых способов создания простой модели — допущение о том, что данные подчиняются нормальному распределению без ковариации между измерениями. Для обучения этой модели достаточно найти среднее значение и стандартное отклонение точек внутри каждой из категорий — это все, что требуется для описания подобного распределения. Результат этого наивного Гауссова допущения показан на рис. 4.2.

Эллипсы на этом рисунке представляют Гауссову порождающую модель для каждой из меток с ростом вероятности по мере приближении к центру эллипса. С помощью этой порождающей модели для каждого класса мы можем легко вычислить вероятность P(признаков | L_i) для каждой точки данных, а следовательно, быстро рассчитать соотношение для апостериорной вероятности и определить, какая из меток с большей вероятностью соответствует конкретной точке.

Эта процедура реализована в оценивателе sklearn.naive_bayes.GaussianNB :

```
In[3]: from sklearn.naive_bayes import GaussianNB
       model = GaussianNB()
       model.fit(X, y);
```

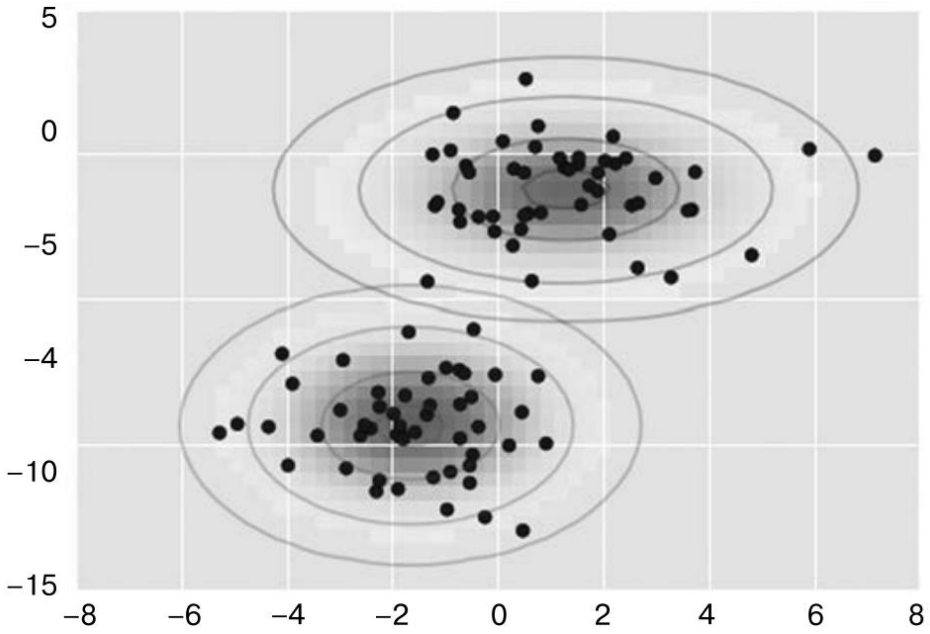

Рис. 4.2. Визуализация Гауссовой наивной байесовской модели

Сгенерируем какие-нибудь новые данные и выполним предсказание метки:

```
In[4]: rng = np.random.RandomState(0)
       Xnew = [-6, -14] + [14, 18] * rng.rand(2000, 2)
       ynew = model.predict(Xnew)
```

Теперь у нас есть возможность построить график этих новых данных и понять, где пролегает граница принятия решений (decision boundary) (рис. 4.3):

```
In[5]: plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='RdBu')
       lim = plt.axis()
       plt.scatter(Xnew[:, 0], Xnew[:, 1], c=ynew, s=20, cmap='RdBu', alpha=0.1)
       plt.axis(lim);
```

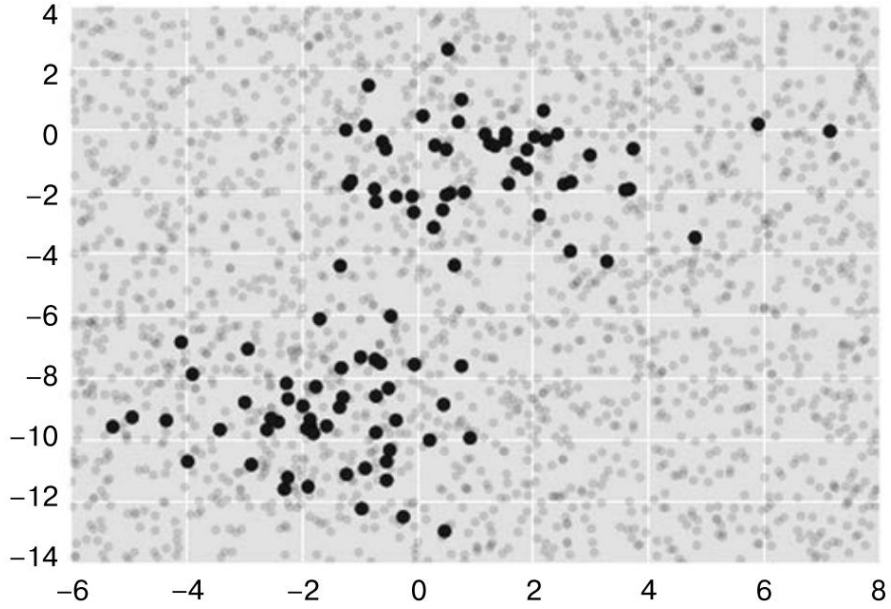

Рис. 4.3. Визуализация Гауссовой наивной байесовской классификации

Мы видим, что граница слегка изогнута, в целом граница при Гауссовом наивном байесовском классификаторе соответствует кривой второго порядка.

Положительная сторона этого байесовского формального представления заключается в возможности естественной вероятностной классификации, рассчитать которую можно с помощью метода predict_proba:

```
In[6]: yprob = model.predict_proba(Xnew)
       yprob[-8:].round(2)

Out[6]: array([[ 0.89 ,   0.11 ]
               [ 1.    ,0.    ]
               [ 1.    ,0.    ]
               [ 1.    ,0.    ]
               [ 1.    ,0.    ]
               [ 1.    ,0.    ]
               [ 0.    ,1.    ]
               [ 0.15 ,  0.85 ]])
```

Столбцы отражают апостериорные вероятности первой и второй меток соответственно. Подобные байесовские методы могут оказаться весьма удобным подходом при необходимости получения оценок погрешностей в классификации.

Качество получаемой в итоге классификации не может превышать качества исходных допущений модели, поэтому Гауссов наивный байесовский классификатор зачастую не демонстрирует слишком хороших результатов. Тем не менее вомногих случаях — особенно при значительном количестве признаков — исходные допущения не настолько плохи, чтобы нивелировать удобство Гауссова наивного байесовского классификатора.

Полиномиальный наивный байесовский классификатор

Гауссово допущение — далеко не единственное простое допущение, которое можно использовать для описания порождающего распределения для всех меток. Еще один полезный пример — полиномиальный наивный байесовский классификатор с допущением о том, что признаки сгенерированы на основе простого полиномиального распределения. Полиномиальное распределение описывает вероятность наблюдения количеств вхождений в несколько категорий, таким образом, полиномиальный наивный байесовский классификатор лучше всего подходит для признаков, отражающих количество или частоту вхождения.

Основная идея остается точно такой же, но вместо моделирования распределения данных с оптимальной Гауссовой функцией мы моделируем распределение данных с оптимальным полиномиальным распределением.

Пример: классификация текста. Полиномиальный наивный байесовский классификатор нередко используется при классификации текста, где признаки соответствуют количеству слов или частотам их употребления в классифицируемых документах. Здесь же, чтобы продемонстрировать классификацию коротких документов по категориям, мы воспользуемся

разреженными признаками количеств слов из корпуса текста 20 Newsgroups («20 дискуссионных групп»).

Скачаем данные и изучим целевые названия:

```
In[7]: from sklearn.datasets import fetch_20newsgroups
       data = fetch_20newsgroups()
       data.target_names
Out[7]: ['alt.atheism',
          'comp.graphics',
          'comp.os.ms-windows.misc',
          'comp.sys.ibm.pc.hardware',
          'comp.sys.mac.hardware',
          'comp.windows.x',
          'misc.forsale',
          'rec.autos',
          'rec.motorcycles',
          'rec.sport.baseball',
          'rec.sport.hockey',
          'sci.crypt',
          'sci.electronics',
          'sci.med',
          'sci.space',
          'soc.religion.christian',
          'talk.politics.guns',
          'talk.politics.mideast',
          'talk.politics.misc',
          'talk.religion.misc']
```

Для простоты выберем лишь несколько из этих категорий, после чего скачаем обучающую и контрольную последовательности:

```
In[8]:
categories = ['talk.religion.misc', 'soc.religion.christian', 'sci.space',
                'comp.graphics']
train = fetch_20newsgroups(subset='train', categories=categories)
test = fetch_20newsgroups(subset='test', categories=categories)
```

Вот типичный образец записи из этого набора данных:

```
In[9]: print(train.data[5])
```

From: dmcgee@uluhe.soest.hawaii.edu (Don McGee)

Subject: Federal Hearing

Originator: dmcgee@uluhe

Organization: School of Ocean and Earth Science and Technology

Distribution: usa

Lines: 10

Fact or rumor....? Madalyn Murray O'Hare an atheist who eliminated the use of the bible reading and prayer in public schools 15 years ago is now going to appear before the FCC with a petition to stop the reading of the Gospel on the airways of America. And she is also campaigning to remove Christmas programs, songs, etc from the public schools. If it is true then mail to Federal Communications Commission 1919 H Street Washington DC 20054 expressing your opposition to her request. Reference Petition number 2493.

Чтобы использовать эти данные для машинного обучения, необходимо преобразовать содержимое каждой строки в числовой вектор. Для этого воспользуемся векторизатором TF–IDF и создадим конвейер, присоединяющий его последовательно к полиномиальному наивному байесовскому классификатору:

```
In[10]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.pipeline import make_pipeline
        model = make_pipeline(TfidfVectorizer(), MultinomialNB())
```

С помощью этого конвейера мы можем применить модель к обучающей последовательности и предсказать метки для контрольных данных:

```
In[11]: model.fit(train.data, train.target)
        labels = model.predict(test.data)
```

Теперь, предсказав метки для контрольных данных, мы изучим их, чтобы выяснить эффективность работы оценивателя. Например, вот матрица различий между настоящими и предсказанными метками для контрольных данных (рис. 4.4):

```
In[12]:
from sklearn.metrics import confusion_matrix
mat = confusion_matrix(test.target, labels)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=train.target_names, yticklabels=train.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label');
```

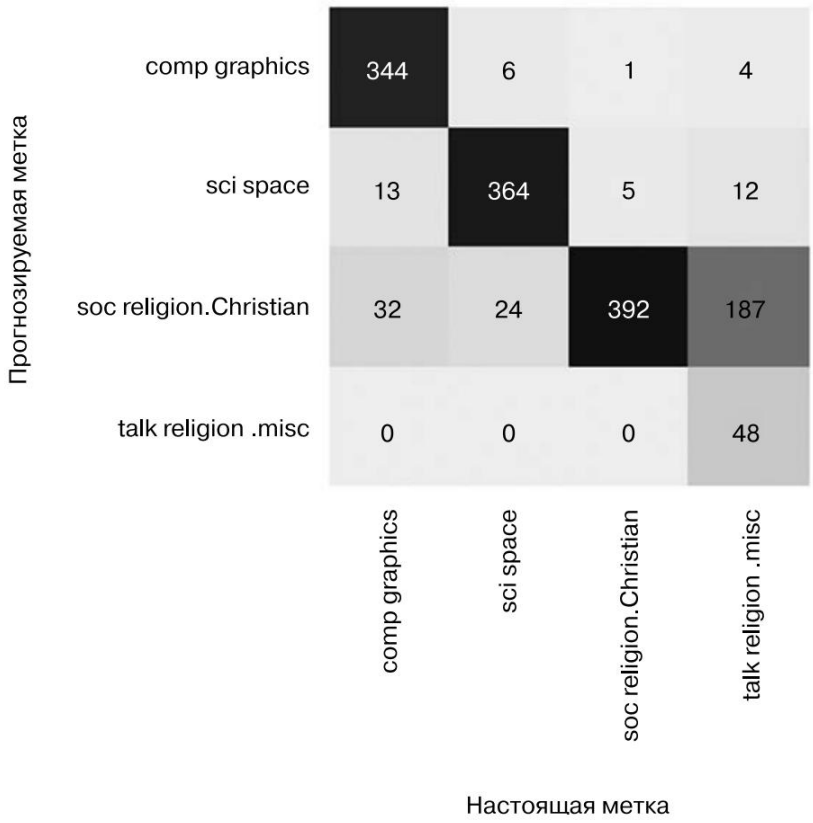

Рис. 4.4. Матрица различий для полиномиального наивного байесовского классификатора текста

Даже этот очень простой классификатор может легко отделять обсуждения космоса от дискуссий о компьютерах, но он путает обсуждения религии вообще и обсуждения христианства. Вероятно, этого следовало ожидать!

Хорошая новость состоит в том, что у нас теперь есть инструмент определения категории для любой строки с помощью метода predict() нашего конвейера. Следующий фрагмент кода описывает простую вспомогательную функцию, возвращающую предсказание для отдельной строки:

```
In[13]: def predict_category(s, train=train, model=model):
            pred = model.predict([s])
            return train.target_names[pred[0]]
```

Попробуем ее на деле:

```
In[14]: predict_category('sending a payload to the ISS')
Out[14]: 'sci.space'
In[15]: predict_category('discussing islam vs atheism')
Out[15]: 'soc.religion.christian'
In[16]: predict_category('determining the screen resolution')
Out[16]: 'comp.graphics'
```

Это лишь простая вероятностная модель (взвешенной) частоты каждого из слов в строке, тем не менее результат поразителен. Даже очень наивный алгоритм

может оказаться удивительно эффективным при разумном использовании и обучении на большом наборе многомерных данных.

Когда имеет смысл использовать наивный байесовский классификатор

В силу столь строгих допущений относительно данных наивные байесовские классификаторы обычно работают хуже, чем более сложные модели. Тем не менее у них есть несколько достоинств:

- они выполняют как обучение, так и предсказание исключительно быстро;
- обеспечивают простое вероятностное предсказание;
- их результаты часто очень легки для интерпретации;
- у них очень мало (если вообще есть) настраиваемых параметров.

Эти достоинства означают, что наивный байесовский классификатор зачастую оказывается удачным кандидатом на роль первоначальной эталонной классификации. Если оказывается, что он демонстрирует удовлетворительные результаты, то поздравляем: вы нашли для своей задачи очень быстрый классификатор, возвращающий очень удобные для интерпретации результаты. Если же нет, то вы всегда можете начать пробовать более сложные модели, уже имея представление о том, насколько хорошо они должны работать.

Наивные байесовские классификаторы склонны демонстрировать особенно хорошие результаты в следующих случаях:

- когда данные действительно соответствуют наивным допущениям (на практике бывает очень редко);
- для очень хорошо разделяемых категорий, когда сложность модели не столь важна;
- для данных с очень большим числом измерений, когда сложность модели не столь важна.

Два последних случая кажутся отдельными, но на самом деле они взаимосвязаны: по мере роста количества измерений у набора данных вероятность близости любых двух точек падает (в конце концов, чтобы находиться рядом, они должны находиться рядом по каждому из измерений). Это значит, что кластеры в многомерных случаях имеют склонность к более выраженной изоляции, чем кластеры в случаях с меньшим количеством измерений (конечно, если новые измерения действительно вносят дополнительную информацию). Поэтому упрощенные классификаторы, такие как наивный байесовский классификатор, при росте количества измерений начинают работать не хуже, а то и лучше более сложных: когда данных достаточно много, даже простая модель может оказаться весьма эффективной.