1. **Постановка задачи**

Необходимо разработать программное обеспечения для классификации классов различных объектов по их признакам, используя метод байесовской классификации

1. **Теоретический материал**

Наивный байесовский классификатор - простой вероятностный классификатор, основанный на применении теоремы Байеса со строгими предположениями о независимости.

Сама теорема Байеса позволяет высчитать вероятность выполнения некоторого события при условии, что произошло другое статистически взаимосвязанное с ним событие.

Математическая интерпретация теоремы выражается формулой:

(1)

где

P(A|B) – вероятность гипотезы A при наступлении события B;

P(B|A) – вероятность события B при истинности гипотезы A;

P(B) – полная вероятность наступления события B;

P(A) – априорная вероятность гипотезы А.

Формулу можно трактовать по-разному. Для нашей задачи классификации будем использовать следующую:

Предположим, что мы имеем некоторый объект X, который обладает свойством B, по формуле (1) мы можем посчитать вероятность того, что X принадлежит классу A

P(A|B) – вероятность принадлежности классу A при наличии свойства B;

P(B|A) – вероятность принадлежности объекта со свойством B классу А;

P(B) – полная вероятность наличия свойства B;

P(A) – априорная вероятность принадлежности классу А.

Если же свойств B не одно, то есть B = (, , …, ), то из предположения условной независимости: предположим, что каждое свойство условно независимо от любого другого , получим

(2)

Таким образом, чтобы выявить принадлежность объекта какому-то из классов на основе имеющихся свойств, нужно вычитать значение вероятность P(A|B) по формуле (1) с учетом формулы (2) для каждого из классов из некоторого априорно известного множества, тот класс, при котором высчитанная вероятность будет наибольшей, логичней всего и положить классом классифицируемого объекта.

Однако так как полная вероятность наличия свойства B никак не зависит от проверяемого класса, её можно убрать из формулы, так как она никак не влияет результат сравнений.

Тогда итоговая формула (принцип), на которой работает классификатор выглядит следующим образом:

(3)

Почему это работает?

Такой вопрос может возникнуть, есть внимательно отнестись к тому, мы используем предположение об условной независимости признаков. В самом деле, когда речь идет о реальных объектах, многие их свойства, скорее будут влиять друг на друга, чем окажутся независимыми (это предположение о независимости признаков весьма наивно, откуда о название классификатора). Таким образом, мы почти всегда будем получать очень грубую оценку вероятности принадлежности объекта с некоторыми свойствами данному классу. Однако, так как такие грубые оценки получаются для каждого из классов, вероятность принадлежности к которому мы высчитываем, на результат сравнения полной вероятности и нашей грубой оценки этой вероятности остается неизменным.

Несколько свойств наивного байесовского классификатора

1. Модель байесовского классификатора требует обучения для задания априорных вероятностей классов и вероятности принадлежности свойств классу (P(B|A) в формуле (1)).
2. Вероятности принадлежности свойства классу вычисляется просто как количество объектов этого классу с данными свойством к общему количеству классов. Все эти вероятности нужно запоминать, делается это составлением таблиц. Таким образом, если у классифицируемых объектов много свойств и/или много возможных вариантов значений каких-то свойств, задача может стать очень трудоемкой как по памяти, так и по времени.
3. Байесовский классификатор лучше работает с категориальными значениями атрибутов (свойств), чем с непрерывными, так как в случае прогноза на наборе, где значение какого-то из атрибутов не встречалось в обучающем множестве, это свойство не внесет никакого вклада в вычисление нужно нам для сравнений вероятности. Такие ситуации чаще возникают при непрерывных значениях атрибутов.
4. Тем не менее, несмотря на п.3, байесовский классификатор может быть достаточно устойчив к потере данных или к ситуациям, когда на множестве для прогноза встречаются значение, не встречающиеся раннее (обе эти ситуации далее будем называть утерей значения). Связано это с тем, что если такое значение признака не встречалось ни у одного из классов, то никаких ошибочных выводов не последует (этот признак не принесёт никакого вклада в вероятность ни для одного из классов). Проблемы могут возникнуть, если утерян значимый признак (например, имеются два различных класса, но в большинстве их объектов различные значения встречаются только у одного из атрибутов. При утере значения именного этого свойства, классы станут неразличимы) или если утерянных значений очень много, что может часто возникать в случае непрерывных значений атрибутов.
5. **Описание алгоритма решения задачи**

Обучение:

1. Проходим по всем элементам обучающего множества и записываем количество различных классов в словарь, в котором ключ – имя класса, значение – количество раз, которое этот класс встретился
2. Снова проходим по элементам обучающего множества, но теперь смотрим на атрибуты:
3. Если значение текущего атрибута для текущего номера свойства уже встречалось, меняем вероятность встречи значения этого атрибута при таком классе, прибавляя в нужную ячейку (вид таблицы см. ниже на рис.1) 1 деленную на количество этих классов (значение берем из уже составленного словаря)
4. Если такое значение свойства ещё не встречалось, добавляем новую строку в таблицу и меняем вероятность под нужным классом
5. Значение по каждому из ключей словаря, делим на общее количество примеров. Так мы получаем словарь из априорных оценок класса

В результате обучения получаем следующие таблицы:





Рис 1. Примеры таблицы свойств

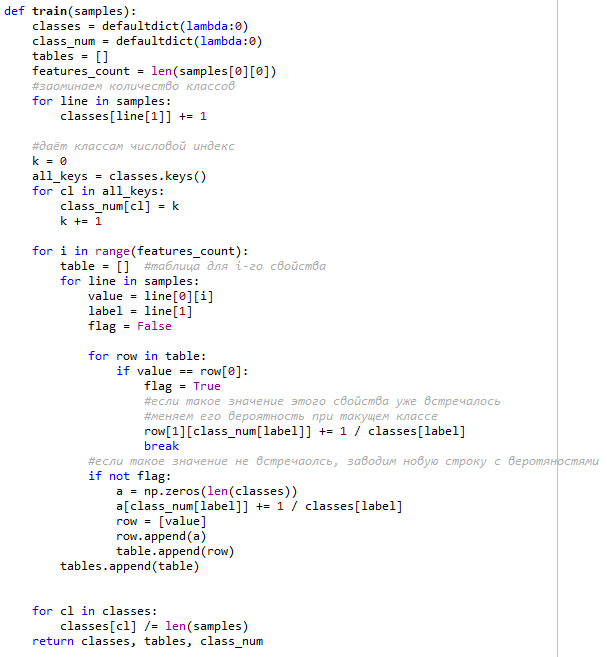
На Рис. 1 мы имеем таблицы для двух свойств. Информация в них распределена так: каждый элемент таблицы состоит из 2 частей – значения свойства (например, ‘yellow’) и массив из вероятностей принадлежности объекта с таким значением атрибута к классу, имена которых как будто написаны над массивом в порядке, определенном в словаре классов. Необходимость создавать для каждого свойства свою таблицу появляется из того, что у разных атрибутов могут быть одинаковые значения, их нужно чётко разделять



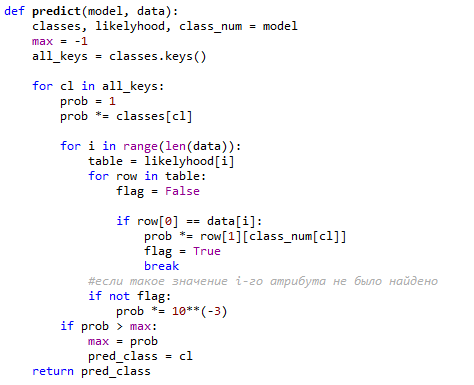
Рис. 2 Пример словаря с априорными вероятностями классов

Прогноз:

1. Запускаем цикл по всем известным классам
2. Задается начальное значение вероятности
3. Умножаем вероятность в априорную вероятность текущего класса
4. В цикле по всем свойствам ищем строку таблицы текущего признака, первый элемент которого равен значению заданного значения этого же свойства
5. Умножаем вероятность на вероятность из массива в той же строке по индексу, равным индексу текущего проверяемого класса
6. Если заданное значение не было найдено в таблице, умножаем вероятность на
7. После выходя из цикла по свойствам, сравниваем полученную вероятность в максимальной. В случае, если получилась больше, меняем значение максимума на полученную вероятность и значение предсказанного класса на текущий проверяемый
8. **Программная реализация алгоритма**



Словарь class\_num содержит только индексы имеющихся классов, в саму модель классификатора он входить не обязан



1. **Результаты тестов**

В качестве обучающей выборки используем следующую базу данных, отражающую некоторую информацию о различных фруктах

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,long,banana

yellow,sweet,short,banana

yellow,sweet,short,banana

yellow,sweet,short,banana

orange,sweet,round,orange

orange,sweet,round,orange

orange,sweet,round,orange

orange,bitter,round,orange

orange,bitter,round,orange

orange,bitter,round,orange

yellow,sweet,round,orange

yellow,sweet,round,orange

yellow,sweet,round,orange

yellow,bitter,round,orange

yellow,bitter,round,orange

green,acidic,round,apple

green,acidic,round,apple

green,acidic,round,apple

green,acidic,round,apple

green,acidic,round,apple

green,sweet,round,apple

green,sweet,round,apple

green,sweet,round,apple

red,acidic,round,apple

red,acidic,round,apple

red,acidic,round,apple

red,sweet,round,apple

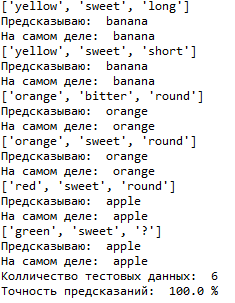
red,sweet,round,apple

red,sweet,round,apple

red,sweet,round,apple

Имеем 3 класса: banana, orange, apple, каждый из которых имеет 3 свойства отражающий их цвет, вкус и форму. Обратим внимание, что у объектов одинаковых классов могут быть различные значения свойств, а том числе совпадающие со значениями этих же свойств объектов других классов

Протестировав классификатор на этом наборе получили:



Сначала подается набор признаков, затем предсказанный класс, затем истинный. Видим, что классификатор правильно определяет класс как для часто встречающихся записей (длинный банан), так и для более редких (короткий банан). 4-й и 5-й наборы отличаются только одним значением, но его достаточно, чтобы определить верный класс. В последнем наборе вовсе утеряно одно значение, но класс определен правильно

Для проверки способности обучаться на различных наборах воспользуемся методом кросс-валидации:

Принцип метода заключается в разделении исходного множества на некоторое количество подмножеств, выделение одной части как тестовый набор, а всё оставшиеся - как обучающий

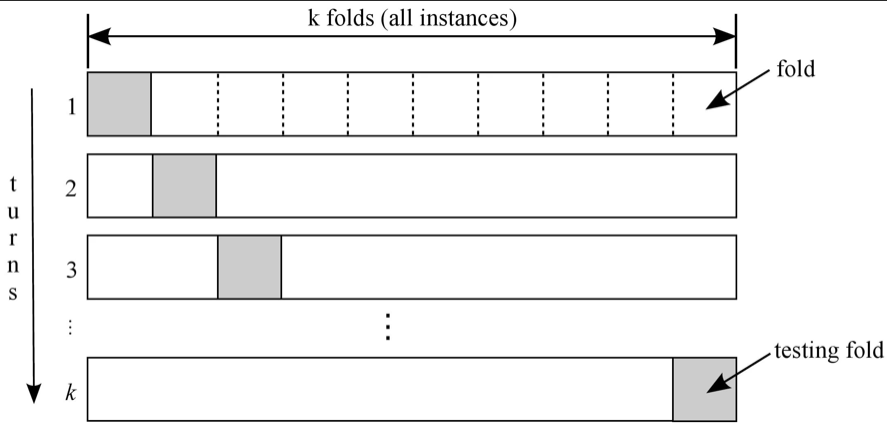


Рис. 3 Схема кросс-валидации

На Рис.3 изображено на разделение на k подмножеств. Серый квадрат отражает тестовый сет, остальные белые – тренировочный. Так мы получаем k тестов на различных тренирующих наборах

Протестируем этот метод тестирования на стандартном наборе данных ирисов Фишера

При k = 6 получили:

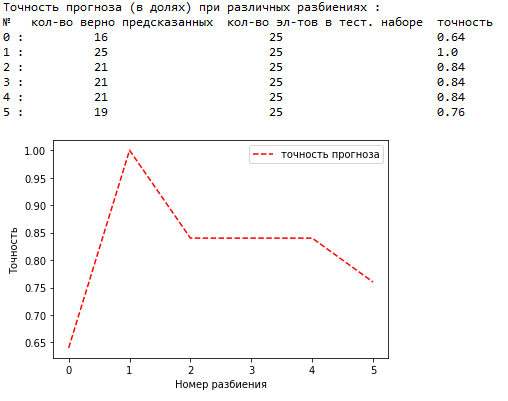


Рис. 4 Результаты кросс-валидации на ирисах Фишера

Как видно из Рис. 4, выбор обучающего набора имеет большое значение при построении модели классификатора. Связано это с тем, что, если в обучающее множество попадет недостаточно записей об объектах какого-то из классов, у модели могут неправильно выставиться априорные оценки классов, а также в таблицу признаков не запишутся значения свойств, свойственных объектам этого класса.

1. **Вывод**

Метод байесовской классификации – очень простой для понимания алгоритм, основанный лишь на одной формуле теории вероятности. Классификатор даёт достаточно хорошие результаты на любых наборах данных, вне зависимости от количества свойств или множества принимаемых значений этих свойств. Однако, наиболее хорошие прогнозы будут при использовании категориальных значений атрибутов. Так же стоит очень аккуратно выбирать тренирующую выборку, стоит следить за тем, чтобы в ней было представлено наиболее возможное количество объектов различных классов и с разнообразными наборами свойств.

1. **Список используемой литераторы**
2. <https://ru.wikipedia.org/wiki/Теорема_Байеса>
3. <https://ru.wikipedia.org/wiki/Наивный_байесовский_классификатор>
4. <http://datascientist.one/naive-bayes/>
5. <https://archive.ics.uci.edu/ml/index.php>
6. **Листинг программного кода**

from collections import defaultdict

from random import randrange

import numpy as np

import matplotlib.pyplot as plt

def flat\_list(list\_of\_list):

res = []

for fold in list\_of\_list:

for sample in fold:

res.append(sample)

return res

def train(samples):

classes = defaultdict(lambda:0)

class\_num = defaultdict(lambda:0)

tables = []

features\_count = len(samples[0][0])

#заоминаем количество классов

for line in samples:

classes[line[1]] += 1

#даёт классам числовой индекс

k = 0

all\_keys = classes.keys()

for cl in all\_keys:

class\_num[cl] = k

k += 1

for i in range(features\_count):

table = [] #таблица для i-го свойства

for line in samples:

value = line[0][i]

label = line[1]

flag = False

for row in table:

if value == row[0]:

flag = True

#если такое значение этого свойства уже встречалось

#меняем его вероятность при такущем классе

row[1][class\_num[label]] += 1 / classes[label]

break

#если такое значение не встречаолсь, заводим новую строку с веротяностями

if not flag:

a = np.zeros(len(classes))

a[class\_num[label]] += 1 / classes[label]

row = [value]

row.append(a)

table.append(row)

tables.append(table)

for cl in classes:

classes[cl] /= len(samples)

return classes, tables, class\_num

def predict(model, data):

classes, likelyhood, class\_num = model

max = -1

all\_keys = classes.keys()

for cl in all\_keys:

prob = 1

prob \*= classes[cl]

for i in range(len(data)):

table = likelyhood[i]

for row in table:

flag = False

if row[0] == data[i]:

prob \*= row[1][class\_num[cl]]

flag = True

break

#если такое значение i-го атрибута не было найдено

if not flag:

prob \*= 10\*\*(-3)

if prob > max:

max = prob

pred\_class = cl

return pred\_class

def get\_features(feats):

features = []

for feat in feats[:-1]:

features.append(feat)

return features

def get\_label(feats):

return feats[-1]

path\_to\_data = 'C:\\Users\\Admin\\Kurs \\'

def do\_samples(path):

file = open(path)

samples = []

for line in file:

if line[-1] == '\n':

line = line[:-1]

feats = line.split(',')

samples.append([get\_features(feats),get\_label(feats)])

return samples

def split\_samples(samples, k):

copy = samples

list\_of\_folds = []

fold\_len = int(len(samples)/k)

for i in range(k):

fold = []

while len(fold) < fold\_len:

random\_index = randrange(len(copy))

fold.append(copy.pop(random\_index))

list\_of\_folds.append(fold)

return list\_of\_folds

def run\_cross\_validation(samples,k):

list\_of\_folds = split\_samples(samples,k)

acc = []

for fold in list\_of\_folds:

train\_set = []

for x in list\_of\_folds:

train\_set.append(x)

test\_set = []

for x in fold:

test\_set.append(x)

train\_set.remove(test\_set) #всё ещё список спиков свойств

train\_set = flat\_list(train\_set)

#train\_set обучающее множество fold - остается тестовым

model = train(train\_set)

accuracy = 0

for line in test\_set:

prediction = predict(model, line[0])

if prediction == line[1]:

accuracy += 1

count = len(test\_set)

acc.append(accuracy/count)

return acc

train\_file = path\_to\_data + 'iris.data'

samples = do\_samples(train\_file)

cross\_validate = False

if cross\_validate:

k = 6

acc = run\_cross\_validation(samples, k)

print('Точность разбиения в долях при различных рахбиениях :',acc)

plt.plot(acc, color = 'r', label = 'точность прогноза', linestyle = '--')

plt.ylabel('Точность')

plt.xlabel('Номер разделения')

plt.legend()

else:

model = train(samples)

test\_file = path\_to\_data + 'test.data'

op\_file = open(test\_file)

count = 0

acc = 0

for line in op\_file:

if line[-1] == '\n':

line = line[:-1]

line = line.split(',')

print(get\_features(line))

prediction = predict(model, get\_features(line))

count += 1

print('Предсказываю: ' , prediction)

print('На самом деле: ', get\_label(line))

if prediction == get\_label(line):

acc += 1

print('Колличество тестовых данных: ', count)

acc = acc / count \* 100

print('Точность предсказаний: ', acc, '%')