Байесовская классификация

Содержание [убрать]

1 Вероятностная постановка задачи классификации
2 Оптимальный байесовский классификатор
3 Наивный байесовский классификатор
4 Применение
5 Примеры кода
5.1 Пример кода scikit-learn
5.2 Пример на языке Java
5.3 Пример на языке R
6 См. также
7 Источники информации

Вероятностная постановка задачи классификации

Пусть X множество объектов, Y конечное множество имён классов, множество $X \times Y$ является вероятностным пространством с плотностью распределения $p(x,y) = P(y)p(x|y)$. Вероятности появления объектов каждого из классов $P_y = P(y)$ называются *априорными вероятностями классов*. Плотности распределения $p_y(x) = p(x|y)$ называются *функциями правдоподобия классов*.

Вероятностная постановка задачи классификации разделяется на две независимые подзадачи:

- Имеется простая выборка $X^l = (x_i, y_i)_{i=1}^l$ из неизвестного распределения $p(x,y) = P_y p_y(x)$. Требуется построить *эмпирические оценки* априорных вероятностей P'_y и функций правдоподобия $p'_y(x)$ для каждого из классов $y \in Y$.
- По известным плотностям распределения $p_y(x)$ и априорным вероятностям P_y всех классов $y \in Y$ построить алгоритм $a(x)$, минимизирующий вероятность ошибочной классификации.

Априорные вероятности классов P_y можно оценить согласно закону больших чисел, тогда частота появления объектов каждого из классов равна $P'_y = \frac{l_y}{l}$ где $l_y = |X_y^l|, y \in Y$ сходится по вероятности к P_y при $l_y \to \infty$. Чем больше длина выборки, тем точнее выборочная оценка P'_y.

Оптимальный байесовский классификатор

Рассмотрим произвольный алгоритм $a : X \to Y$. Он разбивает множество X на не пересекающиеся области $A_y = \{x \in X | a(x) = y\}, y \in Y$. Вероятность того,что появится объект класса y и алгоритм a отнесёт его к классу s, равна $P_y P(A_s|y)$. Каждой паре $(y,s) \in Y \times Y$ поставим в соответствие величину потери λ_{ys} при отнесении объекта класса y к классу s .

Определение:

Функционал среднего риска — ожидаемая величина потери при классификации объектов алгоритмом a:

$$R(a) = \sum_{y \in Y} \sum_{s \in Y} \lambda_{ys} P_y P(A_s|y)$$

Теорема (об оптимальности байесовского классификатора):

Если известны априорные вероятности P_y и функции правдоподобия $p_y(x)$,

то минимум среднего риска $R(a)$ достигается алгоритмом

$$a(x) = \arg\min_{s \in Y} \sum_{y \in Y} \lambda_{ys} P_y p_y(x)$$

Доказательство:

▷

Для произвольного $t \in Y$ запишем функционал среднего риска:

$$R(a) = \sum_{y \in Y} \sum_{s \in Y} \lambda_{ys} P_y P(A_s|y) = \sum_{y \in Y} \lambda_{yt} P_y P(A_t|y) + \sum_{s \in Y \setminus \{t\}} \sum_{y \in Y} \lambda_{ys} P_y P(A_s|y).$$

Применив формулу полной вероятности, $P(A_t \mid y) = 1 - \sum_{s \in Y \setminus \{t\}} P(A_s \mid y)$, получим:

$$R(a) = \sum_{y \in Y} \lambda_{yt} P_y + \sum_{s \in Y \setminus \{t\}} \sum_{y \in Y} (\lambda_{ys} - \lambda_{yt}) P_y P(A_s|y) =$$

$$= const(a) + \sum_{s \in Y \setminus \{t\}} \int_{A_s} \sum_{y \in Y} (\lambda_{ys} - \lambda_{yt}) P_y p_y(x) dx.$$

Введём для сокращения записи обозначение $g_s(x) = \sum_{y \in Y} \lambda_{ys} P_y p_y(x)$, тогда $R(a) = const(a) + \sum_{s \in Y \setminus \{t\}} \int_{A_s} (g_s(x) - g_t(x)) dx$.

Минимум интеграла достигается, когда A_s совпадает с областью неположительности подынтегрального выражения.

$A_s = \{x \in X \mid g_s(x) \leq g_t(x), \forall t \in Y, t \leq s\}$.

С другой стороны, $A_s = \{x \in X \mid a(x) = s\}$. Значит, $a(x) = s$ тогда и только тогда, когда

$s = \arg\min_{s \in Y} g_s(x)$.

◁

Наивный байесовский классификатор

Допустим, что объекты $x \in X$ описываются n числовыми признаками $f_j : X \to R, j = 1, \ldots, n$. Обозначим через $x = (\xi_1, \ldots, \xi_n)$ произвольный элемент пространства объектов $X = R^n$, где $\xi_j = f_j(x)$.

Предположим, что признаки $f_1(x), \ldots, f_n(x)$ являются независимыми случайными величинами. Следовательно, функции правдоподобия классов представимы в виде:

$$p_y(x) = \prod_{i=1}^{n} p_{yj}(\xi_i)$$

где $p_{yj}(\xi_j)$ плотность распределения значений j-го признака для класса y. Алгоритмы классификации исходящие из этого предположения, называются *наивными байесовскими*.

Подставим эмпирические оценки одномерных плотностей в байесовский классификатор. Получим алгоритм:

$$a(x) = \arg\max_{y \in Y} (\ln \lambda_y P'_y + \sum_{j=1}^{n} \ln p'_{yj}(\xi_j)).$$

Основные его преимущества — простота реализации и низкие вычислительные затраты при обучении и классификации. В тех редких случаях, когда признаки почти независимы, наивный байесовский классификатор близок к оптимальному. Достаточно малое количество данных необходимо для обучения, оценки параметров и классификации.

Основной его недостаток — низкое качество классификации в общем случае.

Применение

Из-за своего низкого качества классификации наивный байесовскими классификатор в основном он используется либо как эталон при экспериментальном сравнении алгоритмов, либо как элементарный строительный блок в алгоритмических композициях.

Рассмотрим частое применение байесовского классификатора к задаче классификации документов по их содержимому, а именно к классификации электронных писем на два класса — спам (S) и не-спам ($\neg S$), предполагая что вероятность слов в тексте не зависит друг от друга:

Программные спам-фильтры, построенные на принципах наивного байесовского классификатора, делают «наивное» предположение о том, что события, соответствующие наличию того или иного слова в электронном письме или сообщении, являются независимыми по отношению друг к другу. Это упрощение в общем случае является неверным для естественных языков:

$$P(a\ very\ close\ game) = P(a) \times P(very) \times P(close) \times P(game)$$

Исходя из такого предположения, для решения задачи классификации сообщений лишь на 2 класса: S (спам) и $H = \neg S$ («хэм», то есть не спам) из теоремы Байеса можно вывести следующую формулу оценки вероятности «спамовости» всего сообщения D, содержащего слова $W_1, W_2, \ldots W_N$:

$$p(S \mid D) = p(S \mid W_1, W_2, \ldots W_N) = \frac{p(W_1, W_2, \ldots W_N \mid S) \cdot p(S)}{p(W_1, W_2, \ldots W_N)} = [\text{так как } W_i \text{ предполагаются независимыми}] =$$

$$= \frac{\prod_i p(W_i \mid S) \cdot p(S)}{p(W_1, W_2, \ldots W_N)} = \frac{\prod_i p(S \mid W_i)}{\prod_i (p(S \mid W_i)) + \left(\frac{p(\neg S)}{p(S)}\right)^{1-N} \cdot \prod_i p(\neg S \mid W_i)}$$

Результат p обычно сравнивают с некоторым порогом (например, 0.5), чтобы решить, является ли сообщение спамом или нет. Если p ниже, чем порог, сообщение рассматривают как вероятный «ham», иначе его рассматривают как вероятный спам.

$$\ln \frac{p(S \mid D)}{p(\neg S \mid D)} > h$$

Примеры кода

Пример кода scikit-learn

Классификатор `GaussianNB` реализует наивный байесовский классификатор в предположении что изначальное распределение было гауссовым:

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

```
from sklearn import datasets
from sklearn.metrics import f1_score, accuracy_score
from sklearn.naive_bayes import GaussianNB
iris = datasets.load_iris()
gnb = GaussianNB()
pred = gnb.fit(iris.data, iris.target).predict(iris.data)
accuracy = accuracy_score(iris.target, pred)
f1 = f1_score(iris.target, pred, average="micro")
print("accuracy:", accuracy, "f1:", f1)
```

Вывод:

```
accuracy: 0.96 f1: 0.96
```

Пример на языке Java

Пример классификации с применением `weka.classifiers.bayes.NaiveBayes` [1]

`Maven` зависимость:

```
<dependency>
  <groupId>nz.ac.waikato.cms.weka</groupId>
  <artifactId>weka-stable</artifactId>
  <version>3.8.0</version>
</dependency>
```

```
import weka.classifiers.bayes.NaiveBayes;
import weka.classifiers.evaluation.Evaluation;
import weka.core.converters.ConverterUtils;
import java.util.Random;
```

```
// load dataset
var source = new DataSource("/iris.arff");
var dataset = source.getDataSet();
// set class index to the last attribute
dataset.setClassIndex(dataset.numAttributes() - 1);
// create and build the classifier
var nb = new NaiveBayes();
nb.buildClassifier(dataset);
// cross validate model
var eval = new Evaluation(dataset);
eval.crossValidateModel(nb, dataset, 10, new Random(41));
System.out.println("Estimated Accuracy: "+ Double.toString(eval.pctCorrect()));
```

Пример на языке R

*Основная статья: **Примеры кода на R***

```
# importing package and it's dependencies
library(e1071)

# reading data
data <- read.csv("input.csv", sep = ',', header = FALSE)

# splitting data into training and test data sets
index <- createDataPartition(y = data$target, p = 0.8, list = FALSE)
training <- data[index,]
testing <- data[-index,]

# create objects x and y for predictor and response variables
x <- training[, -9]
y <- training$target

# training model
model <- train(x, y, 'nb', trControl = trainControl(method = 'cv', number = 10))

# predicting results
predictions <- predict(model, newdata = testing)
```

См. также

- Байесовские сети
- Независимые события
- Формула Байеса

Источники информации

- Википедия — Наивный байесовский классификатор
- К.В.Воронцов Математические методы обучения по прецедентам
- scikit-learn 1.9. Supervised learning - Naive Bayes
- ↑ Weka. Naive Bayes

Категории: Машинное обучение | Классификация и регрессия