

第四讲 - 文本分类

张建章

阿里巴巴商学院
杭州师范大学

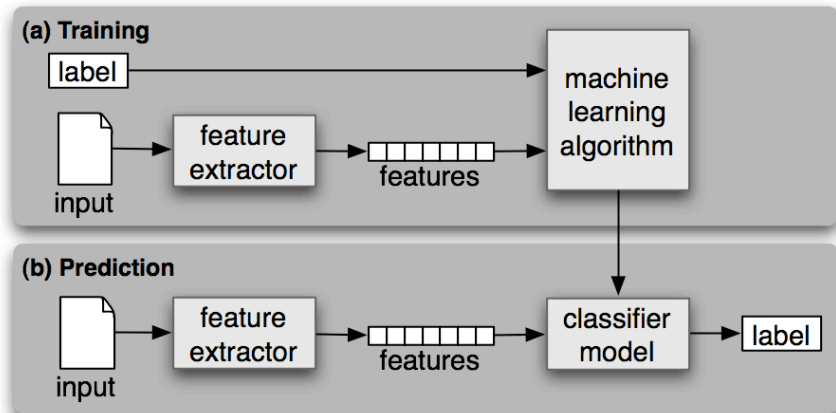
2023-02-22



- 1 文本分类任务定义
- 2 常用文本分类算法
- 3 伯努利朴素贝叶斯文本分类实例
- 4 常用的文本二分类评价指标
- 5 文本多分类评价指标
- 6 计算文本分类器性能代码示例
- 7 课后实践

监督文本分类流程

文本分类：将一段给定的文本分配到一个或多个预定义的类别中，商业中广泛用于客户反馈情感分析、文档资料聚合等业务活动。文本分类模型通常采用有监督机器学习算法构建。



任务形式化定义

形式化定义： 给定一个文本集合 $D = \{d_1, d_2, \dots, d_n\}$ ，其中每个文本 d_i 包含一个或多个单词，即 $d_i = \{w_1, w_2, \dots, w_m\}$ 。假设预定义类别集合为 $C = \{c_1, c_2, \dots, c_k\}$ 。文本分类模型的目标是找到一个函数 f ，将每个文本 d_i 分类到对应的类别 c_j 中，即 $f(d_i) = c_j$ ，函数 f 通常通过计算文本 d_i 属于每个类别 c_j 的概率来实现，即

$$P(c_j|d_i)$$

根据最大概率原则，将文本 d_i 分类到具有最高概率的类别中，即

$$f(d_i) = \arg \max_{c_j} P(c_j|d_i).$$



常用模型： 朴素贝叶斯，逻辑回归，支持向量机，深度神经网络。




文本分类任务的不同类型

单标签分类：每个样本只属于类别集合中的一个类别。如，某一封邮件是垃圾邮件或正常邮件。

多标签分类：每个样本可以同时属于类别集合中的多个类别。如，某条新闻可同时涉及教育和文化话题 (国学课外辅导机构)。

多标签问题转换：解决多标签文本分类问题常用的一种简单有效方法是二元关联法。基于问题转换策略，将多标签问题拆分为多个独立的二元分类问题，每个类别有一个二元分类器，负责判断样本是否属于该类别。最后，将所有分类器的预测结果组合成一个多标签预测结果。

| Instance | Target |
|----------|---|
| X1 |  |
| X2 |  |
| X3 |  |

| Instance | Target |
|----------|---|
| X1 |  |
| X2 |  |
| X3 |  |






| Instance | Target |
|----------|---|
| X1 |   |
| X2 |    |
| X3 |  |

图 2: 不同类型的文本分类

朴素贝叶斯 I

朴素贝叶斯 (Naive Bayes) 是一种基于贝叶斯定理的简单概率分类算法, 它假设特征之间是条件独立的。

形式化表示: 给定一个数据集 $D = (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$, 其中 $x_i = (x_{i1}, x_{i2}, \dots, x_{im})$ 是特征向量, y_i 是对应的类别标签。目标是对一个新的输入样本 $x = (x_1, x_2, \dots, x_m)$ 进行分类。根据贝叶斯定理, x 属于类别 c_j 的概率可以表示为:

$$P(c_j|x) = \frac{P(x|c_j)P(c_j)}{P(x)} \quad (1)$$

由条件独立假设, 可将联合概率 $P(x|c_j)$ 分解为各特征的条件概率乘积:

$$P(x|c_j) = \prod_{i=1}^m P(x_i|c_j) \quad (2)$$

朴素贝叶斯 II

将 (2) 式带入 (1) 式可得：

$$P(c_j|x) = \frac{\prod_{i=1}^m P(x_i|c_j)P(c_j)}{P(x)} \quad (3)$$

由全概率公式可知：

$$P(x) = \sum_{j=1}^n P(x|c_j)P(c_j)$$

对所有类别来说， $P(x)$ 是一个常数，在进行分类时可忽略。因此，找到具有最大后验概率的类别的分类目标可表示为：

$$\hat{c} = \arg \max_{c_j} P(c_j|x) = \arg \max_{c_j} \left(\prod_{i=1}^m P(x_i|c_j)P(c_j) \right)$$

朴素贝叶斯 III

算法的关键在于计算条件概率 $P(x_i|c_j)$ 和先验概率 $P(c_j)$ ，通常在在训练数据集上进行最大似然估计来获得。根据特征的分布类型采用不同的方法来计算条件概率。

多项式朴素贝叶斯：在文本分类 (尤其是多分类) 中，特征表示某个词语在文档中的词频或 **TF-IDF** 值。对每个类别，计算每个词语在该类别下的相对频率，并用多项式分布来估计条件概率。例如，新闻多分类。

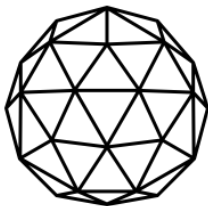


图 3: 多项式分布 (多面体骰子)

朴素贝叶斯 IV

伯努利朴素贝叶斯：在文本分类 (尤其是二分类) 中，将每个词语的存在或不存在作为一个二值特征，计算每个词语在该类别下的存在概率，并用伯努利分布来估计条件概率。例如，评论情感极性二分类。



图 4: 伯努利分布 (硬币的两面)

多项式分布中每次试验有多种可能结果，且 $\sum_{i=1}^k p_i = 1$ 。伯努利分布中，每次试验只有发生和不发生两种结果 (正面朝上)，且成功 (发生) 的概率为 p ，失败 (反面朝上) 为 $1 - p$ 。

对数字商品的评论进行情感二分类，正负评论训练样本如下：

正面 (预处理结果)

I love this movie (love movie)

The film is fantastic (film fantastic)

Great movie, I enjoyed it (great movie enjoyed)

负面 (预处理结果)

What a terrible film (terrible film)

I don't like this movie (don't like movie)

Boring and uninteresting (boring uninteresting)

① 计算先验概率: $P(positive) = \frac{3}{6} = 0.5$, $P(negative) = \frac{3}{6} = 0.5$ 。

② 计算词语的条件概率: 为处理训练集中未出现的词，避免出现 0 概率值，使用参数为 1 的拉普拉斯平滑，计算公式如下：

$$P(w_i|c) = \frac{\text{count}(w_i, c) + \alpha}{|c| + \alpha * M}$$

其中, w_i 是一个特定的单词, c 是一个类别, $\text{count}(w_i, c)$ 是训练集中类别为 c 且包含单词 w_i 的文档数, $|c|$ 为训练集类别为 c 的文档数, α 是平滑参数 (通常取值为 1), M 是类别数。

使用伯努利贝叶斯算法对下面的新评论进行情感二分类:

I love this film (预处理后为: *love film*)

③ 根据式 (3) 计算新评论属于正面和负面的概率:

$$\begin{aligned} P(\text{positive}|\text{love film}) &\propto P(\text{positive}) \times P(\text{love}|\text{positive}) \times P(\text{film}|\text{positive}) \\ &= \frac{1}{2} * \frac{1+1}{3+2} * \frac{1+1}{3+2} = \frac{2}{25} \end{aligned}$$

$$P(\text{negative}|\text{love film}) \propto \frac{1}{2} * \frac{0+1}{3+2} * \frac{1+1}{3+2} = \frac{1}{25}$$

3. 伯努利朴素贝叶斯文本分类实例

划分训练-测试数据集

```
X = sport_docs + other_docs
```

```
y = ['S'] * 1200 + ['O'] * 1200
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
↪ test_size=0.2, random_state=100)
```

二进制词频矩阵向量化文本

```
count_vectorizer = CountVectorizer(tokenizer=lambda text:text,
```

```
↪ lowercase=False, min_df = 5)
```

```
X_train_count_matrix = count_vectorizer.fit_transform(X_train)
```

```
X_train_binary_matrix = np.where(X_train_count_matrix.toarray()
```

```
↪ > 0, 1, 0)
```

```
X_test_count_matrix = count_vectorizer.transform(X_test)
```

```
X_test_binary_matrix = np.where(X_test_count_matrix.toarray() >
```

```
↪ 0, 1, 0)
```

训练伯努利贝叶斯分类器并测试

```
clf = BernoulliNB()
```

```
clf.fit(X_train_binary_matrix, y_train)
```

```
y_pred = clf.predict(X_test_binary_matrix)
```

4. 常用的文本二分类评价指标

准确率 (accuracy): 全部分类结果中，正确结果的比例。

查准率 (precision): 某一类别的分类结果中正确结果的比例。

查全率 (recall): 某一类别的样本被正确分类的比例。

F_1 值 (F_1 score): 查准率和查全率的调和平均值，综合平等地考虑了查准率和查全率。

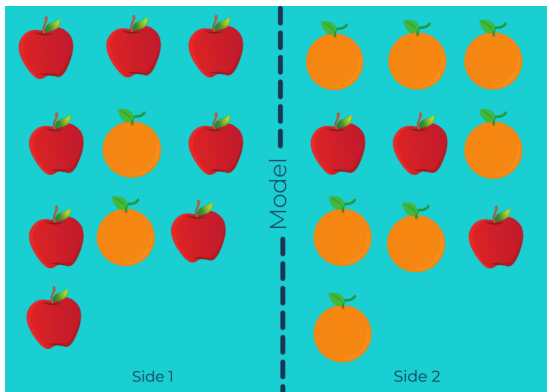


图 5: 分类结果示意图

计算实例

假定上图是某一文本分类模型进行体育新闻二分类的结果示意图，颜色表示样本的真实类别标签，其中红色表示体育新闻，橙色表示非体育新闻，虚线两侧表示模型的预测结果，左侧表示体育新闻，右侧表示非体育新闻。该模型的分类性能指标计算如下：

$$Accuracy = \frac{8 + 7}{20} = 0.75$$

$$Precision = \frac{8}{10} = 0.8$$

$$Recall = \frac{8}{11} \approx 0.73$$

$$F_1 = \frac{2 * precision * recall}{precision + recall} \approx 0.76$$

混淆矩阵

将分类结果中的正确和错误结果的数量整理为混淆矩阵，既便于计算前述指标，也便于对分类结果进行整体错误分析。

| | | PREDICTIVE VALUES | |
|---------------|--------------|-------------------|--------------|
| | | POSITIVE (1) | NEGATIVE (0) |
| ACTUAL VALUES | POSITIVE (1) | TP | FN |
| | NEGATIVE (0) | FP | TN |

图 6: 二分类混淆矩阵示意图

上图中，纵轴表示真实标签，横轴表示分类器预测标签。TP (True Positive) 表示真阳性的数量，FN (False Negative) 表示假阴性数量，FP (False Positive) 表示假阳性数量，TN (True Negative) 表示真阴性数量。在上例中， $TP = 8$, $FN = 3$, $FP = 2$, $TN = 7$ 。

因此，对于文本二分类问题，依据混淆矩阵，前述 4 个性能指标的计算公式如下：

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

5. 文本多分类评价指标

将文本二分类的混淆矩阵扩展到文本多分类形式，可得形如下图所示多分类混淆矩阵：

| | | PREDICTED | | | |
|--------|--------|-----------|--------|--------|--------|
| | | APPLE | GRAPES | BANANA | ORANGE |
| ACTUAL | APPLE | 10 | 2 | 1 | 2 |
| | GRAPES | 5 | 12 | 1 | 2 |
| | BANANA | 5 | 2 | 18 | 10 |
| | ORANGE | 11 | 3 | 1 | 15 |

图 7: 多分类混淆矩阵示例

对每一个类别，可根据前述公式分别计算其 $Precision(P)$ 、 $Recall(R)$ 、 F_1 值。对各类别的指标值进行平均，可衡量分类器总体性能。

对各类别的性能指标可进行宏平均 (macro averaging) 和微平均 (micro averaging), 前者为每个类别赋予相同权重, 后者为每个样本赋予相同权重。假定测试集中有 N 个类别, 计算公式如下:

$$Macro-P = \frac{P_1 + P_2 + \dots + P_N}{N}$$

$$Macro-R = \frac{R_1 + R_2 + \dots + R_N}{N}$$

$$Micro-P = \frac{TP_1 + TP_2 + \dots + TP_N}{(TP_1 + TP_2 + \dots + TP_N) + (FP_1 + FP_2 + \dots + FP_N)}$$

$$Micro-R = \frac{TP_1 + TP_2 + \dots + TP_N}{(TP_1 + TP_2 + \dots + TP_N) + (FN_1 + FN_2 + \dots + FN_N)}$$

计算相应的查准率和查全率的调和平均值可得宏 (微) F_1 值。

文本分类常用指标的其他变体

加权平均 (Weighted Average) 是宏平均的一种特例，其使用类别样本数对各类别的 P 和 R 进行加权，计算公式如下：

$$\text{Weighted-}P = w_1 P_1 + w_1 P_2 + \dots + w_1 P_N$$

$$\text{Weighted-}R = w_1 R_1 + w_1 R_2 + \dots + w_1 R_N$$

其中， $w_i, i = 1, 2, \dots, N$ 为第 i 个类别样本在测试集中的频率。

F_β Score 是 F_1 的更一般形式：

$$F_\beta = (1 + \beta^2) \frac{P \times R}{(\beta^2 \times P) + R}$$

β 权衡对 P 和 R 的重视程度， $\beta > 1$ 更重视 R ， $\beta < 1$ 更重视 P ，常见的 β 取值有 1, 2, 0.5。

6. 计算文本分类器性能代码示例

```
from sklearn.metrics import classification_report

# 在测试集上进行预测
y_pred = clf.predict(X_test_binary_matrix)
# 打印分类器性能指标
print(classification_report(y_test, y_pred))
```

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.96 | 0.98 | 224 |
| S | 0.97 | 1.00 | 0.98 | 256 |
| accuracy | | | 0.98 | 480 |
| macro avg | 0.98 | 0.98 | 0.98 | 480 |
| weighted avg | 0.98 | 0.98 | 0.98 | 480 |

图 8: 分类器性能报告

1. 根据图7中的混淆矩阵，计算宏 (微、加权) 查准率、查全率和 F_1 值。
2. 使用本课程提供的中文新闻语料库，使用 `scikit-learn` 训练新闻多分类模型，使用包含朴素贝叶斯在内的三种分类算法，计算对比不同分类模型的性能。

THE END