Кросс-валидация

Кросс-валидация или **скользящий контроль** — процедура эмпирического оценивания обобщающей способности алгоритмов. С помощью кросс-валидации эмулируется наличие тестовой выборки, которая не участвует в обучении, но для которой известны правильные ответы.

Содержание

- 1 Определения и обозначения
- 2 Разновидности кросс-валидации
  - 2.1 Валидация на отложенных данных (Hold-Out Validation)
  - 2.2 Полная кросс-валидация (Complete cross-validation)
  - 2.3 k-fold кросс-валидация
  - 2.4 t×k-fold кросс-валидация
  - 2.5 Кросс-валидация по отдельным объектам (Leave-One-Out)
  - 2.6 Случайные разбиения (Random subsampling)
  - 2.7 Критерий целостности модели (Model consistency criterion)
- 3 См. также
- 4 Примечания
- 5 Источники информации

Определения и обозначения

Пусть X — множество признаков, описывающих объекты, а Y — конечное множество меток.

$T^l = (x_i, y_i)_{i=1}^l, x_i \in X, y_i \in Y$ — обучающая выборка,

Q — мера качества,

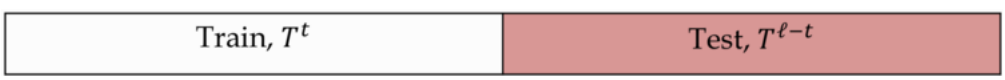
A — модель,

$\mu : (X \times Y)^l \to A,$ — алгоритм обучения.

Разновидности кросс-валидации

Валидация на отложенных данных (Hold-Out Validation)

Обучающая выборка один раз случайным образом разбивается на две части $T^l = T^t \cup T^{l-t}$



После чего решается задача оптимизации:

$HO(\mu, T^t, T^{l-t}) = Q(\mu(T^t), T^{l-t}) \to min,$

Метод Hold-out применяется в случаях больших датасетов, т.к. требует меньше вычислительных мощностей по сравнению с другими методами кросс-валидации. Недостатком метода является то, что оценка существенно зависит от разбиения, тогда как желательно, чтобы она характеризовала только алгоритм обучения.

Полная кросс-валидация (Complete cross-validation)

1. Выбирается значение t;
2. Выборка разбивается всеми возможными способами на две части $T^l = T^t \cup T^{l-t}$.

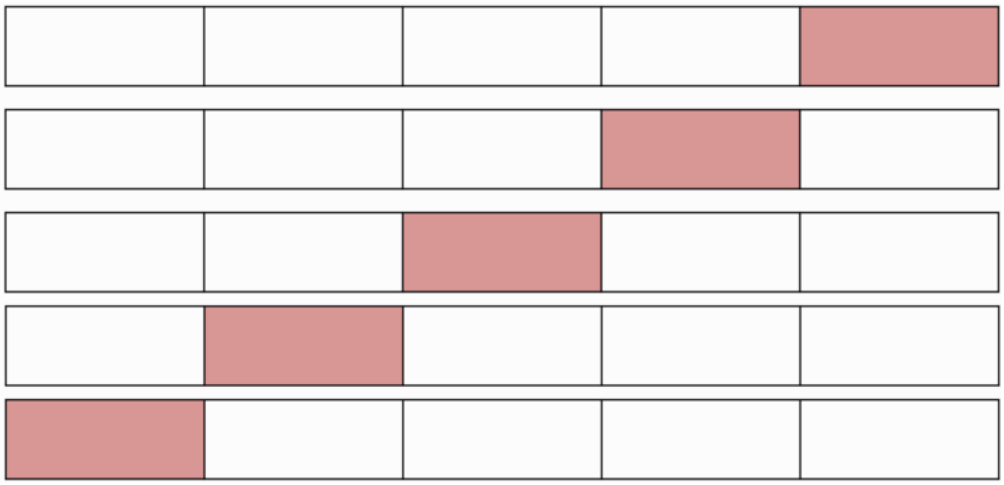
$$CVV_t = \frac{1}{C_l^{l-t}} \sum_{T^l = T^t \cup T^{l-t}} Q(\mu(T^t), T^{l-t}) \to min,$$

Здесь число разбиений C_l^{l-t} становится слишком большим даже при сравнительно малых значениях t, что затрудняет практическое применение данного метода.

k-fold кросс-валидация

1. Обучающая выборка разбивается на k непересекающихся одинаковых по объему частей;
2. Производится k итераций. На каждой итерации происходит следующее:
    1. Модель обучается на $k - 1$ части обучающей выборки;
    2. Модель тестируется на части обучающей выборки, которая не участвовала в обучении.

Каждая из k частей единожды используется для тестирования. Как правило, $k = 10$ (5 в случае малого размера выборки).



$$T^l = F_1 \cup \cdots \cup F_k, |F_i| \approx \frac{l}{k},$$.

$$CV_k = \frac{1}{k} \sum_{i=1}^{k} Q(\mu(T^l \setminus F_i), F_i) \to min$$

```
# Пример кода для k-fold кросс-валидации:
# Пример классификатора, способного проводить различие между всего лишь двумя
# классами, "пятерка" и "не пятерка" из набор данных MNIST
import numpy as np
from sklearn.model_selection import StratifiedKFold
from sklearn.datasets import fetch_openml
from sklearn.base import clone
from sklearn.linear_model import SGDClassifier

mnist = fetch_openml('mnist_784', version=1)
```

```
X, y = mnist["data"], mnist["target"]
y = y.astype(np.uint8)
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
y_train_5 = (y_train == 5)  # True для всех пятерок, False для всех остальных цифр. Задача опознать пятерки
y_test_5 = (y_test == 5)
sgd_clf = SGDClassifier(random_state=42) # классификатор на основе метода стохастического градиентного спуска (Stochastic
# Разбиваем обучающий набора на 3 блока
# выработку прогнозов и их оценку осуществляем на каждом блоке с использованием модели, обученной на остальных блоках</fo
skfolds = StratifiedKFold(n_splits=3, random_state=42)
for train_index, test_index in skfolds.split(X_train, y_train_5):
    clone_clf = clone(sgd_clf)
    X_train_folds = X_train[train_index]
    y_train_folds = y_train_5[train_index]
    X_test_fold = X_train[test_index]
    y_test_fold = y_train_5[test_index]
    clone_clf.fit(X_train_folds, y_train_folds)
    y_pred = clone_clf.predict(X_test_fold)
    n_correct = sum(y_pred == y_test_fold)
    print(n_correct / len(y_pred))
# print 0.95035
#       0.96035
```

t×k-fold кросс-валидация

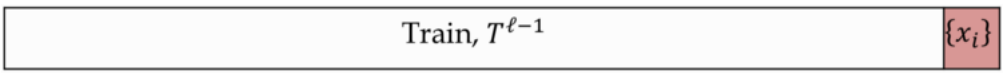
1. Процедура выполняется t раз:
    1. Обучающая выборка случайным образом разбивается на k непересекающихся одинаковых по объему частей;
    2. Производится k итераций. На каждой итерации происходит следующее:
        1. Модель обучается на $k-1$ части обучающей выборки;
        2. Модель тестируется на части обучающей выборки, которая не участвовала в обучении.

$$T^l = F_{(1,1)} \cup \cdots \cup F_{(k,1)} = \cdots = F_{(1,t)} \cup \cdots \cup F_{(k,t)}, |F_{(i,j)}| \approx \frac{l}{k},$$

$$CV_{t\times k} = \frac{1}{tk} \sum_{j=1}^{t} \sum_{i=1}^{k} Q(\mu(T^l \setminus F_{(i,j)}), F_{(i,j)}) \to min.$$

Кросс-валидация по отдельным объектам (Leave-One-Out)

Выборка разбивается на $l-1$ и 1 объект l раз.



$$LOO = \frac{1}{l} \sum_{i=1}^{l} Q(\mu(T^l \setminus p_i), p_i) \to min, \text{ где } p_i = (x_i, y_i).$$

Преимущества LOO в том, что каждый объект ровно один раз участвует в контроле, а длина обучающих подвыборок лишь на единицу меньше длины полной выборки.

Недостатком LOO является большая ресурсоёмкость, так как обучаться приходится L раз. Некоторые методы обучения позволяют достаточно быстро перенастраивать внутренние параметры алгоритма при замене одного обучающего объекта другим. В этих случаях вычисление LOO удаётся заметно ускорить.

Случайные разбиения (Random subsampling)

Выборка разбивается в случайной пропорции. Процедура повторяется несколько раз.

Train, T^t	Test, $T^{\ell-t}$

Критерий целостности модели (Model consistency criterion)

Не переобученый алгоритм должен показывать одинаковую эффективность на каждой части.

Train-1, T^t	Train-2, $T^{\ell-t}$

$$D_1 = (\mu, T^{l-t}) = \frac{1}{l}\sum_{i=1}^{l}(\mu(T^t)(x_i) - \mu(T^{l-t})(x_i)),$$

Метод может быть обобщен как аналог $CV_{t\times 2}$.

См. также

- Общие понятия
- Модель алгоритма и ее выбор
- Мета-обучение

Примечания

1. Кросс-валидация (https://en.wikipedia.org/wiki/Cross-validation_(statistics))

Источники информации

1. Скользящий контроль (http://www.machinelearning.ru/wiki/index.php?title=%D0%A1%D0%BA%D0%BE%D0%BB%D1%8C%D0%B7%D1%8F%D1%89%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C) - статья на MachineLearning.ru
2. Model assessment and selection (https://drive.google.com/open?id=1p9CTAa1_gJpj94RXBEcQ09aVOa-KTlrd)

Источник — «http://neerc.ifmo.ru/wiki/index.php?title=Кросс-валидация&oldid=85440»

- Эта страница последний раз была отредактирована 4 сентября 2022 в 19:32.