Домашнее задание №7: «Лос-Анджелес Доджерс и EM-алгоритм»

Дедлайн 1 (20 баллов): 23 апреля, 23:59
Дедлайн 2 (10 баллов): 30 апреля, 23:59

Домашнее задание нужно написать на Python и сдать в виде одного файла. Правило именования файла: `name_surname_7.py`. Например, если вас зовут Иван Петров, то имя файла должно быть: `ivan_petrov_7.py`.

Рис. 1: Логотип бейсбольного клуба Лос-Анджелес Доджерс

Бейсбол для американцев — больше чем бейсбол. На матчи бейсбольных клубов приезжают семьями из соседних городов. В этом задании предлагается научиться предсказывать, идёт ли матч, по количеству машин вблизи стадиона команды Лос-Анджелес Доджерс.

По ссылке[1] находятся данные от индукционной петли, установленной на 101 North freeway, рядом с городом Glendale. Последняя колонка каждой строки — индикатор того, что на стадионе проходит матч, предпоследняя — количество машин, зафиксированных индукционной петлёй за последние 5 минут. Значения остальных колонок указаны в заголовке файла.

1 Мы будем использовать распределение Пуассона для моделирования количества машин вблизи стадиона. Распределение Пуассона — дискретное распределение с одним параметром $\lambda \in \mathbb{R}+$ и функцией вероятности

$$P(x; \lambda) = \frac{\lambda^k}{x!} e^{-\lambda}$$

Выведите EM-алгоритм для скрытого Марковского классификатора для случая двух классов $Y \equiv \{0, 1\}$ и приложите PDF к письму с решением домашнего задания. Альтернативно убедите себя, что EM-алгоритм должен выглядеть так:

[1] `https://gist.github.com/superbobry/eedc693e0793fbc305a5`

- E-шаг:

$$Q_i(y_i) = \frac{\alpha_{y_i}(i)\beta_{y_i}(i)}{\sum\limits_{y \in Y} \alpha_y(i)\beta_y(i)}$$

- M-шаг:

$$\hat{P}_y = \frac{\sum\limits_{i=1}^{N} Q_i(y)}{N} \qquad \hat{P}_{ys} = \frac{\sum\limits_{i=1}^{N} Q_i(y_{i-1}, y_i)}{\sum\limits_{i=1}^{N} Q_i(y)} \qquad \hat{\lambda}_y = \frac{\sum\limits_{i=1}^{N} Q_i(y) x_i}{\sum\limits_{i=1}^{N} Q_i(y)}$$

2 Реализуйте метод `PoissonHMM.sample`, который принимает целое неотрицательное число — размер выборки и возвращает пару, первый элемент которой — случайная выборка указанного размера, а второй — истинные метки классов для сгенерированной выборки. Можно считать, что классы нумеруются целыми числами $\{0, 1, \ldots, |Y| - 1\}$.

Вам могут быть полезны функции `choice` и `poisson` из пакета `numpy.random`.

Реализацию метода можно проверить, построив гистограмму полученной выборки:

```
import numpy as np
from matplotlib import pyplot as plt

phmm = PoissonHMM([0.5, 0.5], [[0.5, 0.5], [0.5, 0.5]], [4, 12])
X, _y = phmm.sample(1024)
plt.hist(X, bins=10)
plt.show()  # гистограмма должна быть бимодальной
```

Общая структура класса `PoissonHMM`:

```
class PoissonHMM:
    def __init__(self, prior_probs, trans_probs, rates, *,
                 threshold=1e-2):
        self.prior_probs = prior_probs  # P_y
        self.trans_probs = trans_probs  # P_ys
        self.rates = rates              # λ_y
        self.n_classes = len(rates)

    def _log_forward(self):
        ...

    def _log_backward(self):
        ...

    def _update_parameters(self):
        ...

    def fit(self, X):
        ...
        return self

    def predict_log_proba(self, X):
        ...

    def score(self, X):
        ...
```

3 Реализуйте методы `PoissonHMM._log_forward` и `PoissonHMM._log_backward`, вычисляющие $\ln \alpha_y(i)$ и $\ln \beta_y(i)$. Вы можете изменить сигнатуру методов удобным для вас способом.

Обратите внимание на функции `scipy.misc.logsumexp` и `numpy.logaddexp`.

Для вычисления логарифма функции вероятности случайной величины, распределённой по Пуассону, можно воспользоваться методом `logpmg` класса `scipy.stats.poisson`.

4 Реализуйте метод `PoissonHMM.predict_log_proba`, принимающий вектор X и возвращающий матрицу, (y, i)-й элемент которой — апостериорная вероятность класса y для примера x_i, то есть $Q_i(y)$.

5 Реализуйте метод `PoissonHMM.score`, вычисляющий натуральный логарифм правдоподобия для вектора X по формуле[2]:

$$\ln \sum_{y \in Y} \alpha_y(N)$$

6 Реализуйте метод `PoissonHMM.fit`, повторяющий E- и M- шаги до тех пор, пока разница в правдоподобии не будет меньше чем `threshold`.

7 Убедитесь в корректности своей реализации, обучившись по случайной выборке с известными метками классов:

```
from scipy.cluster.vq import kmeans

X, y_true = phmm.sample(1024)
# начальное приближение для λ_y находим с помощью k-means
# для P_y и P_ys используем равномерное распределение (все
# классы и переходы равновероятны).
rates, _ = kmeans(X.astype(float), 2)
model = PoissonHMM([0.5, 0.5], [[0.5, 0.5], [0.5, 0.5]], rates)

# обучаем модель и считаем количество ошибок
log_proba = model.fit(X).predict_log_proba(X)
y_pred = log_proba.argmax(axis=1)
correct = (y_true == y_pred).sum()
print("Correct MAP predictions: {} (out of {}).".format(corect, len(X)))
```

8 Примените `PoissonHMM` к данным о количестве машин. Начальные значения λ_y можно оценить с помощью k-means. Логично предположить, что в среднем количество машин до и после матча должно быть больше, чем количество машин в любое другое время, поэтому начальные значения нужно упорядочить так: $\lambda_0 \leq \lambda_1$. В результате работы EM-алгоритма порядок будет сохранён.

Постройте вектор $\vec{y}$ по методу максимума апостериорной вероятности и оцените точность и полноту полученных предсказаний. Можно воспользоваться уже реализованной вами функцией `print_precision_recall` или аналогичной функцией `classification_report` из пакета `sklearn.metrics`.

[2] Чтобы понять, почему это так, вспомните определение $\alpha_y(i)$.