ВМК МГУ

Разбор практического задания №1 по байесовским рассуждениям

Курс: Байесовские методы в машинном обучении, осень 2015

В данном тексте разбираются вопросы эффективной численной реализации подсчёта некоторых распределений из первого задания. Кроме того, рассматривается аналитический вывод формул для статистик распределений $p(c|a)$ и $p(c|b)$.

Содержание

Вычисление распределений для вероятностных моделей из первых двух вариантов

Рассмотрим следующую вероятностную модель:

$$\begin{aligned} &p(a,b,c,d) = p(d|c)p(c|a,b)p(a)p(b),\\ &d|c \sim c + \text{Bin}(c, p_3),\\ &c|a,b \sim \text{Bin}(a, p_1) + \text{Bin}(b, p_2),\\ &a \sim \text{U}[a_{min}, a_{max}],\\ &b \sim \text{U}[b_{min}, b_{max}]. \end{aligned} \tag{1}$$

Задача 1. *Реализовать вычисление априорного распределения $p(c)$ для всех $c = 0, \ldots, a_{max} + b_{max}$.*

Запишем формулу для вычисления $p(c)$, пользуясь стандартными формулами работы с вероятностями:

$$p(c) = \sum_{\substack{a=a_{min}\\b=b_{min}}}^{\substack{a_{max}\\b_{max}}} p(c,a,b) = \sum_{\substack{a=a_{min}\\b=b_{min}}}^{\substack{a_{max}\\b_{max}}} p(c|a,b)p(a)p(b) = \{\text{в модели (1) } p(a), p(b) \text{ не зависят от } a, b\} =$$

$$= p(a)p(b) \sum_{\substack{a=a_{min}\\b=b_{min}}}^{\substack{a_{max}\\b_{max}}} p(c|a,b) = p(a)p(b) \sum_{\substack{a=a_{min}\\b=b_{min}}}^{\substack{a_{max}\\b_{max}}} \sum_{k=0}^{c} \underbrace{C_a^k p_1^k (1-p_1)^{c-k}}_{\text{обозначим } v_1(a,k)} \underbrace{C_b^{c-k} p_2^{c-k} (1-p_2)^{b-(c-k)}}_{\text{обозначим } v_2(b,c-k)}. \tag{2}$$

Предположим, что вероятности $v_1(a,k)$ и $v_2(b,k)$ вычислены для всех возможных значений a, b и k. Оценим сложность прямого вычисления по формуле (2). Пусть свёртка двух векторов $v_1(a,:)$ и $v_2(b,:)$ для конкретных значений a, b вычисляется за $O(a_{max}b_{max})$[1]. Тогда вычисление $p(c)$ по формуле (2) имеет сложность $O((a_{max} - a_{min})(b_{max} - b_{min})a_{max}b_{max})$.

[1] Эта величина является оценкой сверху. Строго говоря, прямое вычисление для свёртки требует $O(ab)$ операций, а при использовании ДПФ эту сложность можно понизить до $O((a+b)\log(a+b))$.

Однако, эту сложность можно значительно снизить, если в формуле (2) поменять порядок суммирования:

$$p(c) = p(a)p(b) \sum_{\substack{a=a_{min} \\ b=b_{min}}}^{\substack{a_{max} \\ b_{max}}} \sum_{k=0}^{c} v_1(a,k)v_2(b,c-k) = p(a)p(b) \sum_{k=0}^{c} \sum_{\substack{a=a_{min} \\ b=b_{min}}}^{\substack{a_{max} \\ b_{max}}} v_1(a,k)v_2(b,c-k) =$$

$$= p(a)p(b) \sum_{c=0}^{k} \underbrace{\left[\sum_{a=a_{min}}^{a_{max}} v_1(a,k) \right]}_{\text{обозначим } w_1(k)} \underbrace{\left[\sum_{b=b_{min}}^{b_{max}} v_2(b,c-k) \right]}_{\text{обозначим } w_2(c-k)}. \quad (3)$$

Таким образом, двойная сумма по a, b распадается на независимое суммирование по a и b. При вычислении по формуле (3) нужно сначала найти векторы $w_1(:)$ и $w_2(:)$ за $O((a_{max} - a_{min})a_{max})$ и $O((b_{max} - b_{min})b_{max})$ соответственно. Затем свёртка этих векторов находится за $O(a_{max}b_{max})$. В итоге при равенстве a_{max} и b_{max}, а также обнулении a_{min}, b_{min} сложность вычислений по формуле (3) по сравнению с (2) снижается на два порядка с $O(a_{max}^4)$ до $O(a_{max}^2)$.

Рассмотрим также приближённую вероятностную модель для модели (1), в которой $p(c|a,b) = \text{Poiss}(ap_1 + bp_2)$. Прямое вычисление в приближённой модели распределения $p(c)$ предполагает вычисление вероятности распределения Пуассона для всех возможных значений a, b и c с дальнейшим суммированием по всем a и b. Сложность такого вычисления составляет $O((a_{max} + b_{max})(a_{max} - a_{min})(b_{max} - b_{min}))$. Таким образом, при равенстве a_{max}, b_{max} и обнулении a_{min}, b_{min} сложность прямого вычисления в приближённой модели составляет $O(a_{max}^3)$, что на порядок выше вычисления по формуле (3).

Теперь реализуем вычисление априорного распределения $p(c)$ как для модели (1), так и для её приближённой версии:

```
import numpy as np
import scipy.stats as stats

def pc(params, model):

    c = np.arange(params['amax'] + params['bmax'] + 1) # носитель распределения p(c)

    a = np.arange(params['amin'], params['amax'] + 1) # все возможные значения a
    b = np.arange(params['bmin'], params['bmax'] + 1) # все возможные значения b

    if model == 1 or model == 3:
        v1 = stats.binom.pmf(np.arange(params['amax'] + 1), a[:,np.newaxis], params['p1'])
        v2 = stats.binom.pmf(np.arange(params['bmax'] + 1), b[:,np.newaxis], params['p2'])
    else:
        v1 = stats.poisson.pmf(np.arange(params['amax'] + 1), a[:,np.newaxis]*params['p1'])
        v2 = stats.poisson.pmf(np.arange(params['bmax'] + 1), b[:,np.newaxis]*params['p2'])

    w1 = np.sum(v1, axis = 0)
    w2 = np.sum(v2, axis = 0)
    p = np.convolve(w1, w2) # свёртка векторов w1 и w2
    p /= np.sum(p) # нормировка

    return p, c
```

Запустим данный код для параметров из задания:

```
params = {'amin': 75, 'amax': 90,
          'bmin': 500, 'bmax': 600,
          'p1': 0.1, 'p2': 0.01, 'p3':  0.3}

%timeit pc(params, model=1)
%timeit pc(params, model=2)

100 loops, best of 3: 13.8 ms per loop
100 loops, best of 3: 7.84 ms per loop
```

Задача 2. *Вычислить* $p(b|a,d)$ *для всех* $b = b_{min}, \ldots, b_{max}$ *и фиксированных* a, d.

Запишем формулу для вычисления $p(b|a,d)$:

$$p(b|a,d) = \frac{p(a,b,d)}{p(d)} = \frac{\sum_{c=0}^{c_{max}} p(a,b,c,d)}{p(d)} = \frac{\sum_{c=0}^{c_{max}} p(d|c)p(c|a,b)p(a)p(b)}{p(d)} =$$
$$= \left\{ \frac{p(a)p(b)}{p(d)} \text{ является нормировочной константой} \right\} = \frac{1}{Z} \sum_{c=0}^{c_{max}} p(d|c) \sum_{k=0}^{c} \underbrace{C_a^k p_1^k (1-p_1)^{c-k}}_{v_1(a,k)} \underbrace{C_b^{c-k} p_2^{c-k} (1-p_2)^{b-(c-k)}}_{v_2(b,c-k)}. \tag{4}$$

Таким образом, здесь достаточно вычислить числитель формулы (4) для всех значений $b = b_{min}, \ldots, b_{max}$, а затем отнормировать результат. При вычислении числителя необходимо вычислить свёртку вектора $v_1(a,:)$ с набором векторов $v_2(b,:)$ для всех $b = b_{min}, \ldots, b_{max}$. Рассмотрим этот этап подробнее. Можно показать, что операцию свёртки для двух произвольных векторов $\boldsymbol{x} \circledast \boldsymbol{y}$ длины N и M соответственно можно представить как операцию произведения матрицы X на вектор $\boldsymbol{y}$:

$$\boldsymbol{x} \circledast \boldsymbol{y} = X\boldsymbol{y}. \tag{5}$$

Здесь матрица X имеет размер $(N+M-1)\times M$ и строится по вектору $\boldsymbol{x} = [x_1, \ldots, x_N]^T$ и длине M вектора $\boldsymbol{y}$ следующим образом:

$$X = \begin{bmatrix} x_1 & 0 & 0 & \ldots & 0 \\ x_2 & x_1 & 0 & \ldots & 0 \\ x_3 & x_2 & x_1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_N & x_{N-1} & x_{N-2} & \ldots & 0 \\ 0 & x_N & x_{N-1} & \ldots & 0 \\ 0 & 0 & x_N & \ldots & x_1 \\ 0 & 0 & 0 & \ldots & x_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & x_N \end{bmatrix} \tag{6}$$

Теперь для вычисления свёртки фиксированного вектора $\boldsymbol{x}$ с набором векторов $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K$ достаточно одного матричного произведения:

$$\begin{bmatrix} \boldsymbol{x} \circledast \boldsymbol{y}_1 \\ \boldsymbol{x} \circledast \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{x} \circledast \boldsymbol{y}_K \end{bmatrix} = XY, \quad Y = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \ldots & \boldsymbol{y}_K \end{bmatrix}.$$

Заметим, что сложность прямого вычисления одной свёртки $\boldsymbol{x} \circledast \boldsymbol{y}$ и произведения матрицы на вектор по формуле (5) имеет одинаковую сложность. Однако, последняя операция имеет эффективную реализацию в `NumPy` и становится более выгодной при вычислении большого числа свёрток с фиксированным вектором $\boldsymbol{x}$.

Вычисление распределений для вероятностных моделей из варианта 3

Рассмотрим следующую вероятностную модель:

$$\begin{aligned} & p(a,b,c_1,\ldots,c_N,d_1,\ldots,d_N) = p(a)p(b)\prod_{n=1}^{N} p(d_n|c_n)p(c_n|a,b), \\ & d_n|c_n \sim c_n + \mathrm{Bin}(c_n, p_3), \\ & c_n|a,b \sim \mathrm{Bin}(a,p_1) + \mathrm{Bin}(b,p_2), \\ & a \sim \mathrm{U}[a_{min}, a_{max}], \\ & b \sim \mathrm{U}[b_{min}, b_{max}]. \end{aligned} \tag{7}$$

Задача 3. *Реализовать вычисление* $p(b|a, d_1, \ldots, d_N)$ *для всех* $b = b_{min}, \ldots, b_{max}$ *и фиксированных* $a, d_1, \ldots, d_N$.

Запишем формулу для вычисления $p(b|a, d_1, \ldots, d_N)$:

$$p(b|a, d_1, \ldots, d_N) \propto p(a, b, d_1, \ldots, d_N) \propto \sum_{c_1,\ldots,c_N} p(a, b, c_1, \ldots, c_N, d_1, \ldots, d_N) \propto \sum_{c_1,\ldots,c_N} p(a)p(b)\prod_{n=1}^{N} p(d_n|c_n)p(c_n|a, b) \propto$$

$$\propto \{p(a), p(b) \text{ являются константами}\} \propto \prod_{n=1}^{N}\sum_{c_n} p(d_n|c_n)p(c_n|a, b) \propto \prod_{n=1}^{N} p(d_n|a, b). \quad (8)$$

Таким образом, для вычисления требуемого распределения для всех $b = b_{min}, \ldots, b_{max}$ достаточно вычислить две матрицы:

1. Матрицу значений $p(d|c)$ для всех $d = d_1, \ldots, d_N$ и всех $c = 0, \ldots, a_{max} + b_{max}$;
2. Матрицу значений $p(c|a, b)$ для всех $c = 0, \ldots, a_{max} + b_{max}$ и всех $b = b_{min}, \ldots, b_{max}$.

Произведение этих матриц даст матрицу со значениями $p(d|a, b)$ для всех $d = d_1, \ldots, d_N$ и всех $b = b_{min}, \ldots, b_{max}$. Теперь для получения искомого распределения достаточно провести поколоночное произведение элементов матрицы $p(d|a, b)$ с дальнейшей нормировкой полученного вектора.

Теперь реализуем вычисление распределения $p(b|a, d_1, \ldots, d_N)$:

```
from scipy.misc import logsumexp

def pb_ad(a, d, params, model):

    b = np.arange(params['bmin'], params['bmax'] + 1) # носитель распределения
    c = np.arange(params['amax'] + params['bmax'] + 1) # все возможные значения c

    # Вычисляем log p(d|c) для всех d и c
    log_p_d_c = stats.binom.logpmf(d[:,np.newaxis] - c, c, params['p3'])

    # Вычисляем log p(c|a,b) для всех c и b
    if model == 1 or model == 3:
        log_w1 = stats.binom.logpmf(np.arange(params['amax'] + 1), a, params['p1'])
        log_W1 = convmatrix(log_w1, params['bmax']+1, -np.inf) # формируем матрицу свёртки (6)
        log_v2 = stats.binom.logpmf(np.arange(params['bmax'] + 1)[:,np.newaxis], b, params['p2'])
    else:
        log_w1 = stats.poisson.logpmf(np.arange(params['amax'] + 1), a*params['p1'])
        log_W1 = convmatrix(log_w1, params['bmax']+1, -np.inf) # формируем матрицу свёртки (6)
        log_v2 = stats.poisson.logpmf(np.arange(params['bmax'] + 1)[:,np.newaxis], b*params['p2'])

    log_p_c_ab = logsumexp(log_W1[:,:,np.newaxis] + log_v2[np.newaxis,:,:], axis=1)

    # Вычисляем log p(d|a,b) для всех d и b
    log_p_d_ab = logsumexp(log_p_d_c[:,:,np.newaxis] + log_p_c_ab[np.newaxis,:,:], axis=1)

    # Поколоночное произведение и нормировка
    logp_d_ab = np.sum(log_p_d_ab, axis=0)
    logp_d_ab_max = np.max(logp_d_ab)
    p = np.exp(logp_d_ab - logp_d_ab_max)
    p /= np.sum(p)

    return p, b
```

Запустим данный код с параметрами из задания:

```
params = {'amin': 75, 'amax': 90,
          'bmin': 500, 'bmax': 600,
```

```
          'p1': 0.1, 'p2': 0.01, 'p3':  0.3}

a = 75
N = 50
d = 30*np.ones(N)

%timeit pb_ad(a, d, params, model=3)
%timeit pb_ad(a, d, params, model=4)

1 loops, best of 3: 875 ms per loop
1 loops, best of 3: 870 ms per loop
```

Рассмотрим несколько моментов из представленной реализации. Так как нам необходимо вычислить набор вероятностей $p(c|a,b)$ для набора значений c и набора значений b, то здесь удобно воспользоваться матрицей свёртки (6) по аналогии с решением задачи 2. К сожалению, в `SciPy` нет стандартной функции для построения матрицы свёртки (6), поэтому в коде в строках 14, 18 используется самописная функция.

Согласно формуле (8) искомое распределение пропорционально произведению величин $p(d_n|a,b)$. Так как большая часть значений $p(d_n|a,b)$ близка к нулю, то их произведение с ростом N быстро перестаёт укладываться в машинную точность. Поэтому для численной устойчивости здесь необходимо переходить к логарифмам $p(d_n|a,b)$:

$$p(b|a,d_1,\ldots,d_N) = \frac{\prod_{n=1}^N p(d_n|a,b)}{\sum_b \prod_n p(d_n|a,b)} = \frac{\exp(\sum_n \log p(d_n|a,b))}{\sum_b \exp(\sum_n \log p(d_n|a,b))} = \frac{\exp(\sum_n \log p(d_n|a,b) - \text{maxlog})}{\sum_b \exp(\sum_n \log p(d_n|a,b) - \text{maxlog})},$$

где через maxlog обозначена величина $\max\limits_{b=b_{min},\ldots,b_{max}} \sum_{n=1}^N \log p(d_n|a,b)$.

Рассмотрим произвольную матрицу $A \in \mathbb{R}^{N\times M}$, произвольный вектор $\boldsymbol{x} \in \mathbb{R}^M$ и $\boldsymbol{y} = A\boldsymbol{x}$. Тогда

$$\log y_i = \log \sum_{j=1}^M A_{ij}x_j = \log \sum_{j=1}^M \exp(\log A_{ij} + \log x_j) = \text{maxlog}_i + \log \sum_{j=1}^M \exp(\log A_{ij} + \log x_j - \text{maxlog}_i). \qquad (9)$$

Здесь через maxlog_i обозначена величина $\max\limits_{j=1,\ldots,M}(\log A_{ij} + \log x_j)$. Таким образом, зная величины $\log A_{ij}$ и $\log x_j$ мы можем численно устойчиво вычислить $\log y_i$. В `SciPy` для вычисления функций вида «логарифм суммы экспонент» есть стандартная функция `scipy.misc.logsumexp`. В представленном выше коде приём (9) используется два раза: один раз для вычисления произведения матрицы свёртки W_1 на набор векторов v_2 (строка 21), и второй раз для вычисления произведения матрицы вероятностей $p(d|c)$ на матрицу вероятностей $p(c|a,b)$ (строка 24).

Задача 4. *Вычислить $p(b|d_1,\ldots,d_N)$ для всех $b = b_{min},\ldots,b_{max}$ и фиксированных $d_1,\ldots,d_N$.*

Запишем формулу для $p(b|d_1,\ldots,d_N)$:

$$p(b|d_1,\ldots,d_N) \propto p(b,d_1,\ldots,d_N) \propto \sum_a p(a,b,d_1,\ldots,d_N) \propto p(a)p(b)\sum_a p(d_1,\ldots,d_N|a,b) \propto \sum_a \prod_{n=1}^N p(d_n|a,b).$$

Заметим, что в отличие от формулы (8) здесь стоит внешняя сумма по a. Это усложняет реализацию вычисления искомого распределения, т.к. требует, в частности, вычисления величин $p(d|a,b)$ для всех $d = d_1,\ldots,d_N$, $a = a_{min},\ldots,a_{max}$ и $b = b_{min},\ldots,b_{max}$.

Аналитические формулы для дисперсий распределений $p(c|a)$ и $p(c|b)$

Введём следующие обозначения:

- C — случайная величина, имеющая распределение $p(c|a,b)$,
- A — случайная величина, имеющая распределение Пуассона в параметром ap_1 для модели 2 или биномиальное распределение с параметрами a, p_1 для модели 1,
- B — случайная величина, имеющая распределение Пуассона в параметром bp_2 для модели 2 или биномиальное распределение с параметрами b, p_2 для модели 1.

Случайная величина C является суммой независимых случайных величин A и B. Поэтому $\mathbb{E}C = \mathbb{E}A + \mathbb{E}B$, $\mathbb{D}C = \mathbb{D}A + \mathbb{D}B$. Из свойств распределения Пуассона и биномиального распределения $\mathbb{E}A = ap_1, \mathbb{D}A = ap_1$ для модели 2 и $\mathbb{E}A = ap_1, \mathbb{D}A = ap_1(1-p_1)$ для модели 1.

Найдем мат.ожидание и дисперсию для распределения $p(c|a)$ для модели 2:

$$\mathbb{E}_{c|a}c = \sum_c cp(c|a) = \sum_c c\sum_b p(b,c|a) = \sum_c c\sum_b p(c|a,b)p(b) = \{\text{Изменение порядка суммирования}\} =$$
$$\sum_b p(b)\sum_c cp(c|a,b) = \sum_b p(b)\mathbb{E}C = \sum_b p(b)(\mathbb{E}A + \mathbb{E}B) = \sum_b p(b)(ap_1 + bp_2) = ap_1 + \mathbb{E}_b bp_2.$$

$$\mathbb{E}_{c|a}c^2 = \sum_b p(b)\sum_c c^2 p(c|a,b) = \sum_b p(b)\mathbb{E}C^2 = \sum_b p(b)(\mathbb{D}C + (\mathbb{E}C)^2) = \sum_b p(b)(ap_1 + bp_2 + (ap_1 + bp_2)^2) =$$
$$= ap_1 + \mathbb{E}_b bp_2 + a^2p_1^2 + 2a\mathbb{E}_b bp_1p_2 + \mathbb{E}_b b^2p_2^2.$$

$$\mathbb{D}_{c|a}c = \mathbb{E}_{c|a}c^2 - (\mathbb{E}_{c|a}c)^2 = \mathbb{D}_b bp_2^2 + \mathbb{E}_b bp_2 + ap_1.$$

Действуя аналогично, можно показать, что

$$\mathbb{D}_{c|b}c = \mathbb{D}_a ap_1^2 + \mathbb{E}_a ap_1 + bp_2 \text{ — для модели 2,}$$
$$\mathbb{D}_{c|a}c = \mathbb{D}_b bp_2^2 + \mathbb{E}_b bp_2(1-p_2) + ap_1(1-p_1) \text{ — для модели 1,}$$
$$\mathbb{D}_{c|b}c = \mathbb{D}_a ap_1^2 + \mathbb{E}_a ap_1(1-p_1) + bp_2(1-p_2) \text{ — для модели 1.}$$

Отсюда очевидно, что множество $\mathbb{D}_{c|b}c = \mathbb{D}_{c|a}c$ для всех a, b представляет собой прямую линию для обеих моделей:

$$ap_1 - bp_2 = \mathbb{D}_a ap_1^2 + \mathbb{E}_a ap_1 - \mathbb{D}_b bp_2^2 - \mathbb{E}_b bp_2 \text{ — для модели 2,}$$
$$ap_1(1-p_1) - bp_2(1-p_2) = \mathbb{D}_a ap_1^2 + \mathbb{E}_a ap_1(1-p_1) - \mathbb{D}_b bp_2^2 - \mathbb{E}_b bp_2(1-p_2) \text{ — для модели 1.}$$

Однако, множество $\mathbb{D}_{c|b}c = \mathbb{D}_{c|a}c$ для всех p_1, p_2 не является прямой линией, если $\mathbb{E}_a a$ или $\mathbb{E}_b b$ не является целым числом.