Свертка матриц

С.И.Хашин
http://math.ivanovo.ac.ru/dalgebra/Khashin/index.html

Ивановский государственный университет

.

Иваново-2023

План

Сглаживание вектора

Пусть

$$V = \{\dots, v_{-1}, v_0, v_1, v_2, v_3, v_4, \dots, v_n, \dots\},$$

Усредняем по 3-м соседним:

$$W = \{\dots, \frac{v_{-1} + v_0 + v_1}{3}, \frac{v_0 + v_1 + v_2}{3}, \frac{v_1 + v_2 + v_3}{3}, \frac{v_2 + v_3 + v_4}{3}, \dots\},$$

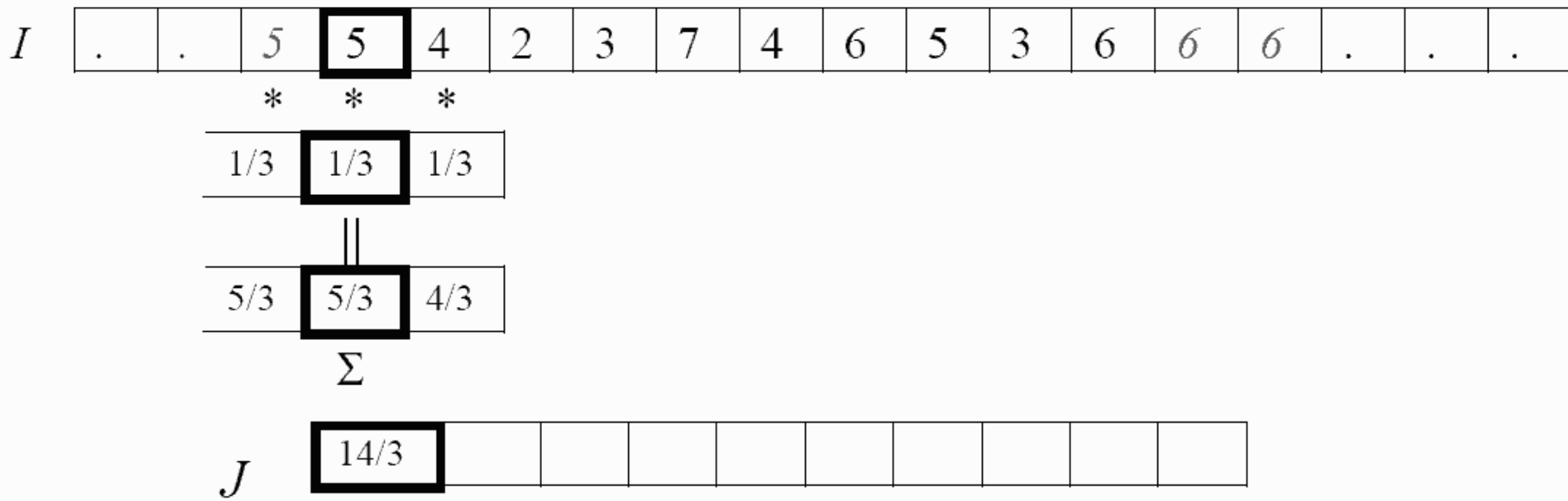
или

$$w_i = \frac{1}{3}v_{i-1} + \frac{1}{3}v_i + \frac{1}{3}v_{i+1}.$$

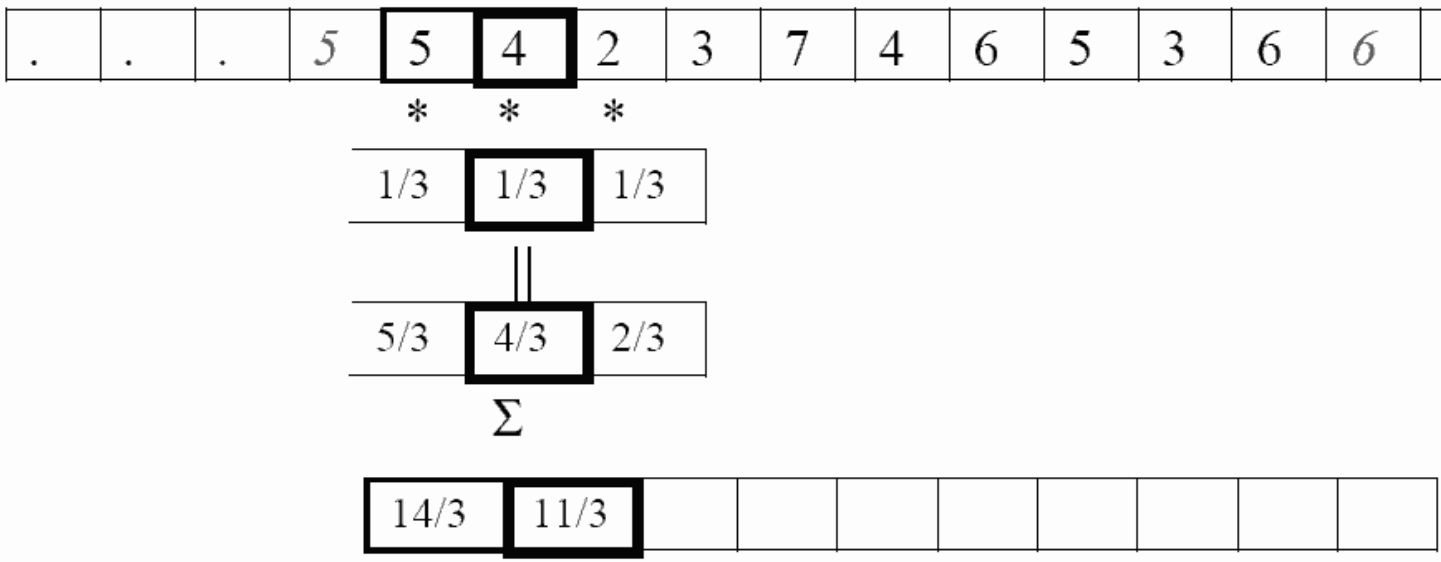
или по 5-м соседним:

$$w_i = \frac{1}{5}v_{i-2} + \frac{1}{5}v_{i-1} + \frac{1}{5}v_i + \frac{1}{5}v_{i+1} + \frac{1}{5}v_{i+2}.$$

Сглаживание вектора



Сглаживание вектора



Сглаживание вектора

Пусть

$$V = \{\dots, v_{-1}, v_0, v_1, v_2, v_3, v_4, \dots, v_n, \dots\},$$

Усредняем по 3-м соседним с весами $(1/4,\ 1/2,\ 1/4)$:

$$W = \{\dots, \frac{1}{4}v_{-1}+\frac{1}{2}v_0+\frac{1}{4}v_1,\ \frac{1}{4}v_0+\frac{1}{2}v_1+\frac{1}{4}v_2,\ \frac{1}{4}v_1+\frac{1}{2}v_2+\frac{1}{4}v_3, \dots\},$$

или

$$w_i = \frac{1}{4}v_{i-1} + \frac{1}{2}v_i + \frac{1}{4}v_{i+1}.$$

Гауссово сглаживание

Коэффициенты гауссового сглаживания радиуса r:

$$u_k = \frac{1}{r\sqrt{2\pi}} e^{-\frac{k^2}{2r}}$$

Формула при $r = 1$:

$$w_i = 0.3989 v_i + 0.2420(v_{i-1} + v_{i+1}) + 0.0540(v_{i-2} + v_{i+2}) +$$

$$+0.0044(v_{i-3} + v_{i+3}) + 0.0001(v_{i-4} + v_{i+4}).$$

Численное дифференцирование

По определению

$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}.$$

Или, при достаточно малом ε:

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x)}{\varepsilon}.$$

Или, более точная формула:

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon}.$$

Численное дифференцирование

Если же функция задана только в целых точках $f(k) = v_k$, то приходится брать $\varepsilon = 1$:

$$w_k = f'(k) \approx \frac{v_{k+1} - v_{k-1}}{2},$$

то есть

$$w_k = -\frac{1}{2}v_{k-1} + \frac{1}{2}v_{k+1}.$$

Есть и более точная формула:

$$w_k = \frac{2}{3}(v_{k+1} - v_{k-1}) - \frac{1}{12}(v_{k+2} - v_{k-2}).$$

Вторая производная

Опять $f(k) = v_k$, численно аппроксимируем 2-ю производную:

$$w_k = f''(k) \approx v_{k+1} - 2v_k + v_{k-1},$$

то есть

$$w_k = v_{k-1} - 2v_k + v_{k+1}.$$

Свёртка в одномерном случае

Пусть есть два вектора бесконечной длины

$$V = \{\dots, v_{-n}, \dots, v_0, v_1, \dots, v_n, \dots\},$$

$$U = \{\dots, u_{-n}, \dots, u_0, u_1, \dots, u_n, \dots\}.$$

Их свёрткой (convolution) будем называть вектор $W = V * U = \{w_i\}$, где

$$w_i = \sum_{j=-\infty}^{+\infty} v_j \cdot u_{i-j}$$

Подробнее при $i = 0$:

$$w_0 = \dots + v_2 u_{-2} + v_1 u_{-1} + v_0 u_0 + v_{-1} u_1 + v_{-2} u_2 + \dots.$$

Свёртка в одномерном случае

$$w_i = \sum v_j \cdot u_{i-j}$$

Подробнее при $i = 1$:

$$w_1 = \cdots + v_2 u_{-1} + v_1 u_0 + v_0 u_1 + v_{-1} u_2 + v_{-2} u_3 + \ldots.$$

При $i = 2$:

$$w_2 = \cdots + v_2 u_0 + v_1 u_1 + v_0 u_2 + v_{-1} u_3 + v_{-2} u_4 + \ldots.$$

и так далее.

Свёртка в одномерном случае

Пусть среди v_i отличны от нуля только (v_{-1}, v_0, v_1). Тогда

$$w_i = v_1 u_{i-1} + v_0 u_i + v_{-1} u_{i+1}.$$

То есть, свёртка с вектором $(1/3,\ 1/3,\ 1/3)$:

$$w_i = \frac{1}{3} v_{i-1} + \frac{1}{3} v_i + \frac{1}{3} v_{i+1}.$$

Свёртка с вектором $(1/2,\ 0,\ -1/2)$:

$$w_i = -\frac{1}{2} v_{i-1} + \frac{1}{2} v_{i+1}.$$

Примеры

Усреднение по 3 точкам является свёрткой с вектором

$$(1/3,\ 1/3,\ 1/3).$$

По 5 точкам: $(1/5,\ 1/5,\ 1/5,\ 1/5,\ 1/5)$.
Сглаживание по 3 точкам: $(1/4,\ 1/2,\ 1/4)$.
Гауссово сглаживание при $r = 1$:

$$(0.0044,\ 0.0540\ 0.2420,\ 0.3989,\ 0.2420,\ 0.0540\ 0.0044).$$

Производная: $(1/2,\ 0,\ -1/2)$ или

$$(-1/12,\ 2/3,\ 0,\ -2/3,\ 1/12).$$

Вторая производная: $(1,\ -2,\ 1)$.

Свойства свёртки

Коммутативность:

$$U * V = V * U.$$

Ассоциативность:

$$(U * V) * W = U * (V * W).$$

Линейность (дистрибутивность):

$$(U_1 + U_2) * V = U_1 * V + U_2 * V.$$

$$U * (V_1 + V_2) * V = U * V_1 + U * V_2.$$

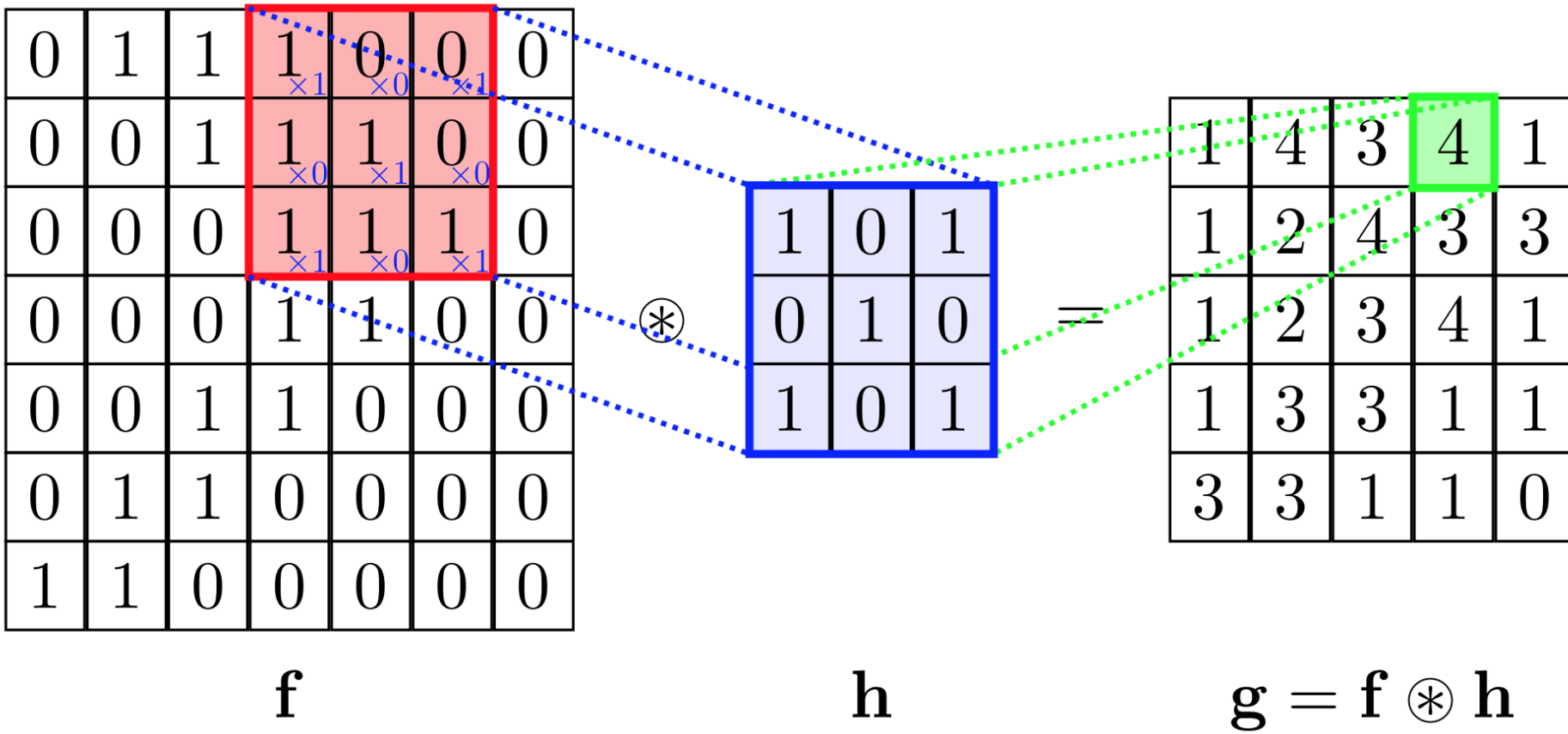
$$(\lambda U) * V = \lambda(U * V) = U * (\lambda V) \quad \forall \lambda \in \mathbb{R}.$$

Свёртка матриц

Пусть есть две матрицы $V = (v_{ij})$ и $U = (u_{ij})$. Их свёрткой (convolution) будем называть вектор $W = V * U = (w_{ij})$, где

$$w_{ij} = \sum_{k,m} v_{ij} \cdot u_{i-k,j-m}$$

Свёртка матриц



png -> npz

Из png-файла размером (mx,my) массив формы (mx, my, 3):

```
def png_npz(png_name, npz_name):
  root = tk.Tk()
  img = tk.PhotoImage(file=png_name)
  my, mx = img.height(), img.width()
  a = np.zeros((my, mx, 3))
  for y in range(my):
    for x in range(mx):
      a[y,x] = img.get(x,y)
  np.savez_compressed(npz_name, data=a)
...
png_npz('isaak_200.png', 'isaak_200')
a = np.load('isaak_200.npz')['data']
print(a.shape)
```

Вывод RGB-массива на экран

```
def plot_rgb(c_tk, rgb):    # Вывод массива a на экран
  rgb = rgb.astype(np.uint8)
  my, mx, i_colors = rgb.shape
  if i_colors != 3:
    raise Exception(
      f'plot_rgb, rgb.shape={my},{mx},{i_colors}')
  for y in range(my):
    for x in range(mx):
      r,g,b = rgb[y,x]
      c_tk.create_rectangle(x,y, x, y + 1,
        fill=f"#{r:02x}{g:02x}{b:02x}", width = 0)
  c_tk.pack()
  c_tk.update()
```

Проверка plot_rgb

```
a = np.load('isaak_200.npz')['data']
print(a.shape)
my, mx, nothing = a.shape
root = tk.Tk()
c_tk = tk.Canvas(root, height=my, width=mx, bg='white')
plot_rgb(c_tk, a)
input('Press CR')
```

Маски

```
np.set_printoptions(precision=4, linewidth=140,
                    suppress=True)
...
mask1 = np.array((0.25, 0.5, 0.25))
mask1 = np.array((1,2,3,4,3,2,1))/16
mask = np.outer(mask1, mask1)
print(mask, '=mask\n')
```

Свёртка

```
def convol2d_rgb(img, mask):
  '''
  Свёртка матрицы img[:,:,i] с маской mask для всех i
  :param img: массив размера (my, mx, mz)
  :param mask: матрица
  :return: convolve(img, mask)
  '''
  my, mx, mz = img.shape
  sy, sx = mask.shape
  res = np.zeros((my-sy+1, mx-sx+1, mz))
  for z in range(mz):
    for y in range(my-sy+1):
      for x in range(mx-sx+1):
        res[y,x, z] = np.sum(img[y:y+sy, x:x+sx, z]*mask)
  return res
```

Проверка сглаживания

```
a = np.load('isaak_200.npz')['data']
print(a.shape)
my, mx, nothing = a.shape
root = tk.Tk()
c_tk = tk.Canvas(root,height=my,width=mx,bg='white')
plot_rgb(c_tk, a)
input('Press CR')
mask1 = np.array((0.25,0.5,0.25)) или
mask1 = np.array((1,2,3,4,3,2,1))/16
mask = np.outer(mask1, mask1)
print(mask, '=mask\n')

a2 = convol2d_rgb(a, mask)
plot_rgb(c_tk, a2)
input('a2, Press CR')
```

Задание

Проверить маску

```
mask = np.array([[0, -1, 0], # d/dy
                 [0,  0, 0],
                 [0,  1, 0]]
```

Внимание!. После свёртки `a2 = convol2d_rgb(a, mask)` каждую цветовую компоненту массива a2 надо нормализовать, то есть

```
a2[:,:,i_color] = K1*a2[:,:,i_color] + K2
```

для некоторых K1, K2. Их надо подобрать так, чтобы минимум был 0, а максимум — 255.

Задание

Проверить маски

```
mask2 = np.array([[ 0,  0, 0], # d/dx
                  [-1,  0, 1],
                  [ 0,  0, 0]]

mask3 = np.array([[0, 1, 0], # d^2/dy^2
                  [0, -2, 0],
                  [0,  1, 0]]

mask4 = np.array([[ 0,  0, 0], # d^2/dx^2
                  [ 1, -2, 1],
                  [ 0,  0, 0]]

mask5 = np.array([[-1,  0, 1], # d^2/dx dy
                  [ 0,  0, 0],
                  [ 1,  0,-1]]
```