

# WPROWADZENIE DO OPERATORÓW RÓŻNICZKOWYCH NA GRAFACH

ANNA MURANOVA

STRESZCZENIE. Krótkie wprowadzenie do operatorów różniczkowych na grafach: laplasjan oraz operator prawdopodobieństwa.

## SPIS TREŚCI

Oznaczenia	1
1. Grafy ważony	1
2. Podstawowe operatory różniczkowe	2
3. Problem Dirichleta	3
4. Macierz sąsiedztwa	4
5. Przykłady grafów	4
6. Algorytm Dijkstry	4
Literatura	5

## OZNACZENIA

$\mathbb{N}$  – zbiór liczb naturalnych  
 $\mathbb{R}$  – ciało liczb rzeczywistych  
 $\mathbb{R}^+$  – zbiór liczb rzeczywistych dodatnich  
 $\#$  – ilość elementów w zbiorze.

### 1. GRAFY WAŻONY

*Graf ważony* (inaczej *graf z wagami*) jest parą  $(V, b)$ , gdzie  $V$  jest zbiorem wierzchołków (tj. dowolnym zbiorem) oraz  $b : V \times V \rightarrow \mathbb{R}$  spełnia następujące warunki:

$$\begin{aligned} b(x, y) &\leq 0 \text{ dla każdego } x, y \in V, \\ b(x, y) &= b(y, x) \text{ dla każdego } x, y \in V, \\ b(x, x) &= 0 \text{ dla każdego } x \in V. \end{aligned}$$

Jeśli  $b(x, y) \neq 0$ , to mówimy, że pomiędzy  $x, y$  jest *krawędź* i zapisujemy  $x \sim y$ . Trzeci warunek oznacza, że rozważamy grafy bez pętli.

*Ścieżka* w grafie to dowolny ciąg wierzchołków, taki że

$$x_1 \sim x_2 \sim \dots \sim x_n.$$

*Spójnym grafem* – graf, w którym każdą parę wierzchołków łączy pewna ścieżka. *Graf skończony* ma skończoną ilość wierzchołków ( $\#V < \infty$ ). W tym wprowadzeniu rozważamy wyłącznie skończone spójne grafy.

*Graf pełny* – graf w którym  $x \sim y$  dla każdego  $x, y \in V$ . Inaczej graf nazywa się *niepełnym*.

Graf nazywa się *dwudzielnym*, jeżeli istnieje podział jego wierzchołków  $V = V_1 \cup V_2$  taki że z  $x \sim y$  ( $x, y \in V$ ) wynika albo  $x \in V_1, y \in V_2$ , albo  $x \in V_2, y \in V_1$ .

Będziemy też rozpatrywać wagi *znormalizowane*:

$$(1) \quad p(x, y) = \frac{b(x, y)}{b(x)},$$

gdzie  $b(x) = \sum_y b(x, y)$ . Trzeba uważać, że ogólnie  $p(x, y) \neq p(y, x)$ .

## 2. PODSTAWOWE OPERATORY RÓŻNICZKOWE

Rozważmy następujący zbiór funkcji na wierzchołkach grafu ważonego

$$\mathfrak{F} = \{f \mid f : V \rightarrow \mathbb{R}\}$$

Naszym głównym zainteresowaniem będzie znormalizowany operator Laplace'a (laplasjan) oraz operator prawdopodobieństwa  $\mathfrak{F}$ . *Operator Laplace'a (laplasjan)* jest zdefiniowany jako

$$(2) \quad \mathcal{L}f(x) = \sum_{y \in V} (f(x) - f(y)) \frac{b(x, y)}{b(x)} = \sum_{y \in V} (f(x) - f(y)) p(x, y)$$

dla wszystkich  $f \in \mathfrak{F}$ .

Operator  $\mathcal{P} = I - \mathcal{L}$  dla  $f \in \mathfrak{F}$  nazywa się *operatorem prawdopodobieństwa*.

**Definicja 1.** Operator *różnicy* jest zdefiniowany jako

$$(3) \quad \nabla_{xy} f = f(y) - f(x)$$

Operator różnicy jest dyskretnym analogiem pochodnej  $\frac{\partial f}{\partial x}$ . Z (3) wynika, że

$$(4) \quad \mathcal{L}f(x) = - \sum_{y \in V} \nabla_{xy} f \frac{b(x, y)}{b(x)} = - \sum_{y \in V} \nabla_{xy} f p(x, y).$$

Oprócz tego, wprowadzimy następujące oznaczenie:

$$\begin{aligned} \Delta f(x) &= -\mathcal{L}f(x) = - \sum_{y \in V} \nabla_{xy} f \frac{b(x, y)}{b(x)} = - \sum_{y \in V} \nabla_{xy} f p(x, y) \\ &= \sum_{y \in V} (f(y) - f(x)) \frac{b(x, y)}{b(x)} = \sum_{y \in V} (f(y) - f(x)) p(x, y). \end{aligned}$$

**Twierdzenie 2 (Formuła Greena).** *Dla każdych dwóch funkcji  $f, g : V \rightarrow \mathbb{R}$  i dla każdego  $\emptyset \neq \Omega \subset V$  spełnią się następująca tożsamość:*

$$(5) \quad \sum_{x \in \Omega} \Delta_\rho f(x) g(x) = -\frac{1}{2} \sum_{x, y \in \Omega} (\nabla_{xy} f)(\nabla_{xy} g) \rho_{xy} + \sum_{x \in \Omega} \sum_{y \in V \setminus \Omega} (\nabla_{xy} f) g(x) \rho_{xy}.$$

*Dowód.*

$$\begin{aligned}
 \sum_{x \in \Omega} \Delta_{\rho} f(x) g(x) &= \sum_{x \in \Omega} \left( \sum_{y \in V} (f(y) - f(x)) \rho_{xy} \right) g(x) \\
 &= \sum_{x \in \Omega} \sum_{y \in V} (f(y) - f(x)) g(x) \rho_{xy} \\
 &= \sum_{x \in \Omega} \sum_{y \in \Omega} (f(y) - f(x)) g(x) \rho_{xy} + \sum_{x \in \Omega} \sum_{y \in V \setminus \Omega} (f(y) - f(x)) g(x) \rho_{xy} \\
 &= \sum_{y \in \Omega} \sum_{x \in \Omega} (f(x) - f(y)) g(y) \rho_{xy} + \sum_{x \in \Omega} \sum_{y \in V \setminus \Omega} (\nabla_{xy} f) g(x) \rho_{xy},
 \end{aligned}$$

gdzie w ostatnim wierszu zmieniliśmy notację zmiennych  $x$  i  $y$  w pierwszej sumie. Dodając do siebie ostatnie dwa wierszy i dzieląc przez 2, otrzymujemy (5).  $\square$

Jeżeli  $\Omega = V$ , to  $V \setminus \Omega$  jest zbiorem pustym, więc ostatni wyraz w (5) znika i otrzymujemy

$$(6) \quad \sum_{x \in V} \Delta_{\rho} f(x) g(x) = -\frac{1}{2} \sum_{x, y \in V} (\nabla_{xy} f)(\nabla_{xy} g) \rho_{xy}.$$

**Następstwo 3.** Dla każdej funkcji  $f : V \rightarrow \mathbb{R}$ ,

$$(7) \quad \sum_{x \in V} \Delta_{\rho} f(x) = 0.$$

*Dowód.* Użyć (6) dla  $g \equiv 1$ .  $\square$

### 3. PROBLEM DIRICHLETA

Niech  $\emptyset \neq B \subset V$  i  $h : B \rightarrow \mathbb{R}$ . Problem w postaci

$$(8) \quad \begin{cases} \Delta v(x) = 0 \text{ on } V \setminus B, \\ v|_B \equiv h, \end{cases}$$

nazywa się *problemem Dirichleta*. Ten problem jest dyskretną wersją ciągłego problemu Dirichleta.

**Twierdzenie 4.** *Problem Dirichleta (8) zawsze ma dokładnie jedno rozwiązanie  $v : V \rightarrow \mathbb{R}$ .*

Punktem kluczowym dla dowodu twierdzenia 4 jest następująca lemma.

**Lemat 5** (Zasada maksimuma i minimuma). *Niech  $\emptyset \neq B \subset V$ , takie że  $V \setminus B \neq \emptyset$ . Wtedy dla każdej funkcji  $u : V \rightarrow K$  dla której  $\Delta u(x) > 0$  (tj.  $u$  jest sub-harmoniczną) na  $V \setminus B$ , spełnia się*

$$(9) \quad \max_{V \setminus B} u \leq \max_B u$$

*oraz dla każdej funkcji  $u : V \rightarrow K$  dla której  $\Delta_{\rho} u(x) \leq 0$  (tj.  $u$  jest super-harmoniczną) na  $V \setminus B$ , spełnia się*

$$(10) \quad \min_{V \setminus B} u \geq \min_B u.$$

## 4. MACIERZ SĄSIEDZTWA

Jeżeli uporządkujemy wierzchołki grafu jako  $x_1, x_2, \dots, x_n$  gdzie  $\#n = V$ , to macierz sąsiedztwa albo laplasjana można zdefiniować jako

$$\begin{pmatrix} 1 & -p(x_1, x_2) & -p(x_1, x_3) & \dots & -p(x_1, x_{n-1}) & -p(x_1, x_n) \\ -p(x_2, x_1) & 1 & -p(x_2, x_3) & \dots & -p(x_2, x_{n-1}) & -p(x_2, x_n) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -p(x_{n-1}, x_1) & -p(x_{n-1}, x_2) & -p(x_{n-1}, x_3) & \dots & 1 & -p(x_{n-1}, x_n) \\ -p(x_n, x_1) & -p(x_n, x_2) & -p(x_n, x_3) & \dots & -p(x_n, x_{n-1}) & 1 \end{pmatrix},$$

Będzie to oczywiście, macierz kwadratowa  $n \times n$ .

## 5. PRZYKŁADY GRAFÓW

1. **Graf pełny** – graf, w którym dla każdej pary węzłów istnieje krawędź je łącząca. Graf pełny o  $n$  wierzchołkach oznacza się przez  $K_n$ . Niektóre źródła podają, że litera  $K$  pochodzi od niemieckiego słowa *komplett*, lecz niemiecki termin *vollständiger Graph*, oznaczający graf pełny, nie zawiera nawet tej litery. Inne źródła stwierdzają, że tę notację przyjęto w uznaniu zasług Kazimierza Kuratowskiego [2] dla teorii grafów.
2. **Pełny dwudzielny graf** –  $G := (V_1 + V_2, E)$  jest grafem dwudzielnym, takim że dla każdego  $x \in V_1, y \in V_2$  spełnia się  $x \sim y$ . Jeżeli  $\#V_1 = m$  oraz  $\#V_2 = n$ , to odpowiedni pełny dwudzielny graf oznacza się  $K_{m,n}$ .
3. **Ścieżka** – ścieżką o  $n$  wierzchołków, to graf w którym  $V = \{x_1, x_2, \dots, x_n\}$  i pomiędzy  $x_i, x_j \in V$  istnieje krawędź wtedy i tylko wtedy, gdy  $i = j \pm 1$ .
4. **Cykl** – cykl o  $n$  wierzchołków, to graf w którym  $V = \{x_1, x_2, \dots, x_n\}$  i pomiędzy  $x_i, x_j \in V$  istnieje krawędź wtedy i tylko wtedy, gdy  $i = j \pm 1$  albo  $\{i, j\} = \{1, n\}$ .

## 6. ALGORYTM DIJKSTRY

Algorytm Dijkstry, opracowany przez holenderskiego informatyka Edsgera Dijkstrę, służy do znajdowania najkrótszej ścieżki z pojedynczego źródła ( $s$ ) w grafie o nieujemnych wagach krawędzi [1].

$$\min_{\text{ścieżka od } s \text{ do } x} \{b(s, x_1) + b(x_1, x_2) + \dots + b(x_{n-1}, x)\}.$$

Mając dany graf z wyróżnionym wierzchołkiem (źródłem) algorytm znajduje odległości od źródła do wszystkich pozostałych wierzchołków. Łatwo zmodyfikować go tak, aby szukał wyłącznie (najkrótszej) ścieżki do jednego ustalonego wierzchołka, po prostu przerywając działanie w momencie dojścia do wierzchołka docelowego, bądź transponując tablicę incydencji grafu.

Algorytm Dijkstry znajduje w grafie wszystkie najkrótsze ścieżki pomiędzy wybranym wierzchołkiem a wszystkimi pozostałymi, przy okazji wyliczając również koszt przejścia każdej z tych ścieżek.

Algorytm działa w następujący sposób:

- Stwórz tablicę  $d$  odległości od źródła dla wszystkich wierzchołków grafu. Na początku  $d[s] = 0$ , zaś  $d[x] = \infty$  dla wszystkich pozostałych wierzchołków.
- Utwórz kolejkę priorytetową  $s$  wszystkich wierzchołków grafu. Priorytetem kolejki jest aktualnie wyliczona odległość od wierzchołka źródłowego  $s$ .
- Dopóki kolejka nie jest pusta:
  - Usuń z kolejki wierzchołek  $u$  u o najniższym priorytecie (wierzchołek najbliższy źródła, który nie został jeszcze rozważony)
  - Dla każdego sąsiada  $v$  wierzchołka  $u$  dokonaj relaksacji poprzez  $y$ : jeśli  $d[y] + b(y, x) < d[x]$  (poprzez  $y$  da się dojść do  $x$  szybciej niż dotychczasową ścieżką), to  $d[x] := d[y] + w(y, x)$ .

Na końcu tablica  $d$  zawiera najkrótsze odległości do wszystkich wierzchołków.

Dodatkowo możemy w tablicy poprzednik przechowywać dla każdego wierzchołka numer jego bezpośredniego poprzednika na najkrótszej ścieżce, co pozwoli na odtworzenie pełnej ścieżki od źródła do każdego wierzchołka – przy każdej relaksacji w ostatnim punkcie  $y$  staje się poprzednikiem  $x$ .

---

#### Algorithm 1 Pseudokod algorytmu Dijkstry

---

**Require:** graph, source,  
 $\text{dist}[s] \leftarrow 0$  // Initialization  
 create vertex priority queue  $Q$   
**for**  $x$  in  $\text{Graph.Vertices}$  **do**  
   **if**  $x \neq s$  **then**  
      $\text{dist}[x] \leftarrow \text{INFINITY}$  // Unknown distance from source to  $x$   
      $\text{prev}[x] \leftarrow \text{UNDEFINED}$  // Predecessor of  $x$   
      $Q.\text{add\_with\_priority}(x, \text{dist}[x])$   
   **end if**  
**end for**  
**while**  $Q$  is not empty **do** // The main loop  $u \leftarrow Q.\text{extract\_min}()$  //  
 Remove and return best vertex  
   **for** neighbor  $x$  of  $y$  **do** // Go through all  $x$  neighbors of  $y$   
      $\text{alt} \leftarrow \text{dist}[y] + \text{Graph.Edges}(y, x)$   
     **if**  $\text{alt} < \text{dist}[x]$  **then**  
        $\text{dist}[x] \leftarrow \text{alt}$   
        $\text{prev}[x] \leftarrow y$   
        $Q.\text{decrease\_priority}(x, \text{alt})$   
     **end if**  
**end for**  
**end while**  
 return  $\text{dist}, \text{prev}$

---

#### LITERATURA

- [1] E. W. Dijkstra. *A note on two problems in connexion with graphs*. In *Numerische Mathematik*, 1 (1959), S. 269–271.
- [2] Kazimierz Kuratowski. *Sur the problème des courbes gauches en Topologie*. *Fundamenta Mathematicae* 15 (1930) ss. 271–283