Notatki praca inżynierska

INBA

31 stycznia 2019

1 Przekrojowe

1.1 Znalezienie min / max elementu w tablicy

Algorytm złożoności $O(n)$.

1. Przypisz jako najmniejszy / największy element (x) zawartość pierwszej komórki w tablicy
2. Przesuwaj się po tablicy, jeśli napotkany element jest mniejszy / większy to przypisz go do x
3. Po przejściu całej tablicy zmienna

1.2 sortowania + złożoności

1.3 złożonośćć obliczeniowa algorytmu (hierarchia)

Rząd złożoności obliczeniowej:

1. 1 – stała
2. $\log n$ – logarytmiczna
3. n – liniowa
4. $n \log n$ – liniowo–logarytmiczna
5. n^2 – kwadratowa
6. n^c – wielomianowa
7. c^n – wykładnicza

1.4 notacje asymptotyczne

1.5 rekurencja

Sposoby liczenia złożoności obliczeniowej dla algorytmów rekurencyjnych:

- Podstawa - zgadnij i udowodnij indukcyjnie

- Drzewo rekursji - patrząc po wysokości drzewa, co znajduje się w węzłach i sumowanie
- Przez zamianę zmiennych - np $n = 2^m$ i wtedy obliczamy $T(n) = T(2^m) = S(m)$ i proste obliczenia
- Metoda iteracyjna - rozpisujemy (rozwijamy równanie) aż znajdziemy zależność
- Master Theorem - wzór oparty o to, czy złożoność generuje korzeń, liście, czy całe drzewo

1.6 Master Theorem

Master Theorem – narzędzie służące do analizy złożoności obliczeniowej algorytmów rekurencyjnych. Nie można go jednak zastosować do każdego rodzaju rekurencji – nie działa, gdy

- $T(n)$ nie jest jednostajny np. $\sin n$
- $f(n)$ nie jest wielomianem np. $T(n) = 2T(\frac{n}{2}) + 2^n$

$$T(n) = aT\Big(\frac{n}{b}\Big) + f(n)$$
$$a \geqslant 1, b \geqslant 2, c \geqslant 0$$

gdzie:

- n – rozmiar problemu
- a – ilość podproblemów w rekursji
- $\frac{n}{b}$ – rozmiar podproblemu
- $f(n)$ – koszt dzielenia i scalania ($f(n) \in \Theta(n^d)$, gdzie $d \geqslant 0$)

$$T(n) = \begin{cases} \Theta(n^d) & a < b^d \\ \Theta(n^d \log n) & a = b^d \\ \Theta(n^{\log_b a}) & a > b^d \end{cases}$$

1.7 Stos

Stos – liniowa struktura danych, w której dane dokładane są na wierzch stosu i z niego pobierane (LIFO – last in, first out).

Wykorzystywana na przykład do zapamiętywania rejestrów proceosra, zmiennych lokalnych.

Podstawowe operacje na kolejce:

- push ($O(1)$) – wstawienie elementu na stos
- pop ($O(1)$) – ściągnięcie elementu ze stosu
- isEmpty ($O(1)$) – sprawdzenie czy stos jest pusty

1.8 Kolejka

1.9 drzewo BST, najdłuższa ścieżka

maksymalna liczba węzłów drzewa wysokości h

$$2^{h+1} - 1$$

maksymalna liczba węzłów drzewa wysokości h-1

$$2^{h} - 1$$

minimalna liczba węzłw drzewa o wyskośćci h
maksymalna liczba węzłów drzewa wysokości h

$$2^{h} - 1 + 1$$

1.10 Kolejka priorytetowa

Kolejka priorytetowa – abstrakcyjna struktura danych w której każdy z elementów ma przypisany priorytet. Dane w niej są uporządkowane nierosnąco lub niemalejąco. Im wyższy priorytet tym szybciej zostanie dany element usunięty z kolejki. Kolejka priorytetowa może zostać wykorzystana na przykład przy szeregowaniu zadań –np. te ważniejsze znajdują się na jej przedzie. Implementację kolejki priorytetowej można wykonać np. przy użyciu kopca.

Operacje na kolejce priorytetowej (max):

- maximum ($O(1)$) – zwrócenie elementu max (pierwszego w kolejce)
- extract maximum ($O(logn)$) – usunięcie elementu max i przearanżowanie kopca
- increase key ($O(logn)$) – zwiększenie priorytetu elementu i przearanżowanie kopca
- insert value ($O(logn)$) – wstawienie nowego elementu i przearanżowanie kopca

2 Algorytmy i struktury danych

2.1 Notacja asymptotyczna

2.1.1 Notacja „duże O”

f jest **co najwyżej rzędu** g jeśli istnieją takie stałe $n_0 > 0$ oraz $c > 0$, że:

$$\forall n > n_0 : f(n) \leqslant c \cdot g(n)$$

co zapisujemy $f(n) \in O(g(n))$

2.1.2 Notacja „małe O”

$f(n) = o(g(n)) \equiv \exists_{c>0} \exists_{n_0>0} \forall_{n>n_0} |f(n)| < c|g(n)|$

2.1.3 Notacja „Ω”

$f(n) = \Omega(g(n)) \equiv \exists_{c>0} \exists_{n_0>0} \forall_{n>n_0} |f(n)| \geqslant c|g(n)|$

2.1.4 Notacja „ω”

$f(n) = w(g(n)) \equiv \exists_{c>0} \exists_{n_0>0} \forall_{n>n_0} |f(n)| > c|g(n)|$

2.1.5 Notacja „Θ”

$f(n) = \Theta(g(n)) \equiv \exists_{c_1,c_2>0} \exists_{n_0>0} \forall_{n>n_0} c_1|g(n)| < |f(n)| < c_2|g(n)|$

2.2 Dziel i rządź

Jedna z głównych metod projektowania algorytmów w informatyce, prowadząca do efektywnych rozwiązań. Schemat postępowania

- Podziel problem na mniejsze podproblemy
- Rozwiąż rekurencyjnie podproblemy
- Scal rozwiązania podproblemów

2.3 Programowanie dynamiczne

Podobnie jak metoda dziel i rządź, rozwiązuje problem poprzez rozwiązanie podproblemów. W programowaniu dynamicznym w przeciwieństwie do metody dziel i rządź, każdy z podproblemów jest rozwiązywany tylko jeden raz. Schemat:

- ...

2.4 Złożoność obliczeniowa O

– tzw. oszacowanie górne

$$\forall n \geqslant n_0 : f(n) \leqslant c \cdot g(n)$$

Wynik funkcji g(n) pomnożony przez jakąś stałą c będzie większy bądź równy wynikowi funkcji f(n). Własność ta jest spełniona dla wszystkich n, które będą większe od n0.

2.5 Złożoność obliczeniowa Ω

– tzw. oszacowanie dolne

$$\forall n \geqslant n_0 : f(n) \geqslant c \cdot g(n)$$

Wynik funkcji g(n) pomnożony przez jakąś stałą c będzie mniejszy bądź równy wynikowi funkcji f(n).

2.6 Złożoność obliczeniowa θ

– odwzorowanie mniejsze niż to, większe niż tamto

$$\forall n \geqslant n_0 : c_1 \cdot g(n) \geqslant f(n) \geqslant c_2 \cdot g(n)$$

Wynik funkcji g(n) pomnożony przez stałą c1 będzie większy bądź równy wartości funkcji f(n). Jednocześnie będzie mniejszy bądź równy wartości funkcji f(n) jeśli pomnożymy go przez stałą c2

2.7 P vs NP

Wiadome jest, że problemy klasy P należą do klasy NP – problem deterministyczny jest możliwy do rozwiązania przy użyciu niedeterministycznej maszyny. Istnieją jednak problemy NP, dla których nie znamy rozwiązania w P – wiemy, że są w NP, ale nie czy w P (komiwojażer). Jeśli ktoś kiedykolwiek znajdzie rozwiązanie wielomianowe do problemu NP–zupełnego to automatycznie znajdzie dla wszystkich $\rightarrow P = NP$.

Problem decyzyjny – problem na który algorytm odpowiada tak lub nie. Problem decyzyjny należy do:

- P jeśli istnieje algorytm o złożoności wielomianowej rozwiązujący to zagadnienie
- NP jeśli istnieje algorytm o złożoności wielomianowej dla niedeterministycznej maszyny

Poli–redukowalny (poly–time reducible) –

Problem NP–zupełny :

- jest NP
- każdy inny problem NP może zostać do nich zredukowany w czasie wielomianowym

Problem NP–trudny – problem, który jest co najmniej tak trudny jak NP–zupełny problem.

2.8 Maszyna Turinga

– abstrakcyjny model maszyny logicznej stworzony przez Alana Turinga w '37.

Maszyna Turinga zbudowana jest z trzech głównych elementów:

- Nieskończonej taśmy zawierającej komórki z przetwarzanymi symbolami – odpowiednik pamięci
- Ruchomej głowicy zapisująco-odczytującej – odpowiednik IO

- Układu sterowania głowicą – odpowiednik procesora

Formalnie opisuje się ją za pomocą krotki:

$$MT =< Q, \Sigma, \delta, \Gamma, q_0, B, F >,$$

gdzie:

- Q – skończony zbiór stanów
- q_0 – stan początkowy
- Γ – skończony zbiór dopuszczalnych symboli
- Σ – zbiór symboli wejściowych
- B – symbol pusty
- F – zbiór stanów końcowych
- δ – funkcja pobierająca aktualny stan maszyny oraz symbol wejściowy a dającą w wyniku symbol, jaki ma się pojawić na taśmie, kolejny stan maszyny oraz przesunięcie głowicy maszyny (lewo, prawo lub bez przesunięcia)

Jeśli problem jest rozwiązalny na komputerze, to można zdefiniować rozwiązującą go maszynę Turinga.

2.9 Algorytmy sortowania

Insertion Sort - bierz element i iteruj tablicę tak długo aż natrafisz na element większy od siebie - np trzymasz karty w ręce i dociągasz kolejną

Złożoność obliczeniowa:

- Best – $O(n)$
- Average – $O(n^2)$
- Worst – $O(n^2)$

Merge Sort Podziel tablicę na pół, posortuj połowy i złącz:

MergeSort(A, n):
B=MergeSort(A[0, $\frac{n}{2}$], $\frac{n}{2}$)
C=MergeSort(A[$\frac{n}{2}$, n], $\frac{n}{2}$)
A = Merge(B, C)
Złożoność obliczeniowa:

- Best – $O(nlog(n))$
- Average – $O(nlog(n))$
- Worst – $O(nlog(n))$

Quick Sort - podziel tablicę A względem elemtnu x (z tej tablicy) i podziel na dwie tablice - z elementami mniejszymi od x i większymi od x. Powtórz algorytm dla każdej z tych dwóch tablic.

Złożoność obliczeniowa:

- Best – $O(nlog(n))$
- Average – $O(nlog(n))$
- Worst – $O(n^2)$

Counting sort Kosztem złożoności pamięciowej, algorytm zlicza ilość wystąpień każdego z elementów (do `hashmapy` z dostępem $O(1)$). Po zliczeniu ilości wystąpień, do zliczonych ilości dodawane jest ilość poprzednich licząc od 2. elementu tak, aby hashmapa trzymała pozycję pierwszego z elementów w posortowanej tablicy. Posortowana tablica jest od generowana z hashmapy.

Złożoność obliczeniowa:

- Best – $O(n + k)$
- Average – $O(n + k)$
- Worst – $O(n + k)$

Radix sort - sortuje zaczynając od sortowania każdej liczby od najmniej znaczącego bitu

Złożoność obliczeniowa:

- Best – $O(nk)$
- Average – $O(nk)$
- Worst – $O(nk)$

2.10 Struktury danych

Drzewo binarne Struktura w której każdy węzeł może mieć maksymalnie dwójkę dzieci - lewe mniejsze od rodzica i prawe większe od niego, węzeł bez żadnego potomka nazywany jest liściem, a jeden wyróżniony węzeł bez rodzica - korzeniem.

Niemal wszystkie operacje uzależniają złożoność od wysokości drzewa - w najlepszym, zbalansowanym przypadku to O(log n), ale dla worst case to O(n).

Drzewo czerwono czarne

- Obowiązują zasady BST
- Każdy węzeł jest czerwony lub czarny
- Korzeń i liście są czarne
- Jeżeli węzeł jest czerwony to ma czarne dzieci
- Każda ścieżka od korzenia do liścia ma taką samą liczbę czarnych węzłów

Dzięki tym zasadom drzewa czerwono-czarne mają zbalansowaną wysokość i operacje są logarytmiczne. Udowodnienie wysokości logarytmicznej można sprowadzić do scalenia czerwonych węzłów do czarnych rodziców - czarna wysokość jest taka sama, czyli drzewo jest zbalansowane, a zasada, że czerwony węzeł musi mieć czarne potomstwo gwarantuje ograniczoną liczbę czerwonych węzłów w ścieżce.

Stos Liniowa struktura danych. Dane umieszczane są na samym szczycie stosu, dostęp jest tylko do tego elementu. Wszelkie operacje na nim przeprowadzane (push, pop, isEmpty) mają złożoność $O(1)$.

Sterta – struktura będąca zbalansowanym drzewem np. binarnym. Reprezentowana za pomocą tablicy, gdzie odpowiednie indeksy oznaczają odpowiednie dzieci i ojców danych węzłów.

Istnieją dwie wersje sterty: max–heap i min–heap. Polegają na tym, że dzieci każdego węzła są odpowiednio mniejsze (max) bądź większe (min) od rodzica.

Indeks lewego dziecka – 2n + 1
Indeks prawego dziecka – 2n + 2
Insert:

1. Add the element to the bottom level of the heap.
2. Compare the added element with its parent; if they are in the correct order, stop.
3. If not, swap the element with its parent and return to the previous step.

Extract:

1. Replace the root of the heap with the last element on the last level.
2. Compare the new root with its children; if they are in the correct order, stop.
3. If not, swap the element with one of its children and return to the previous step. (Swap with its smaller child in a min-heap and its larger child in a max-heap.)

Złożoność obliczeniowa BCS / ACS:

- find-min: $O(1)$ / $O(1)$
- delete-min: $O(\log n)$ /$O(\log n)$
- insert: $O(1)$ / $O(\log n)$

Kolejka priorytetowa

...

2.11 Pozostałe

Znalezienie i–tego najmniejszego elementu tablicy – algorytm podobny do QuickSort, da się jednak lepiej – Median Of Medians.

Szukamy i–tego elementu.
1) Wykonaj partition na tablicy
2) Jeśli pozycja pivotu jest równa i → mamy i–ty element
3) Jeśli jest większa → wykonaj rekurencyjnie dla lewej podtablicy
4) Jeśli jest mniejsza → wykonaj rekurencyjnie dla prawej podtablicy

Złożoność obliczeniowa:

- $ACS : O(n)$
- $WCS : O(n^2)$

Partition Złożoność obliczeniowa:

$$O(n)$$

...

2.12 Problem znalezienia mediany

Naiwne rozwiązanie zakładałoby posortowanie tablicy i wybranie mediany, ale sortowania zazwyczaj nie zadziałają lepiej niż $O(nlogn)$. Rozwiązaniem jest algorytm mediany median.

Mediana median – algorytm pozwalający uzyskać medianę z tablicy w czasie $O(n)$.

1. Jeśli partycjonowana wielkość jest $n \leqslant 5 \rightarrow$ posortuj i zwróć medianę
2. W p.p. podziel tablicę na 5 elementowe podtablice, rekursywnie znajduj mediany i umieszczaj na początku tablicy

3. Na początku tablicy mamy $\frac{n}{5}$ median – odpal rekurencyjnie szukanie dla $\frac{n}{10}$ elementu
4. Zrób partition wokół mediany
5. Jeśli pivot == szukany slot ($\frac{n}{2}$) to mamy medianę median, jeśli $\geqslant$ jeśli $\leqslant$
6.

Złożoność obliczeniowa:

- WCS: $O(n)$
- ACS: $O(n)$

3 Obliczenia naukowe

3.1 Reprezentacja liczb

Każdą liczbę rzeczywistą $x \neq 0$ można zapisać w postaci

$$x = \pm m \cdot \beta^C,$$

gdzie:
m – mantysa $m \in [\frac{1}{\beta}, 1)$
β – baza rozwinięcia (określająca system liczbowy) np. 2, 8, 10, 16
c – cecha, czyli liczba całkowita, $a_i \in 0, 1, \ldots, \beta - 1, a_1 \neq 0$
Na przykład dla $\beta = 10$ liczba $732.5051 = 0.7325951 \cdot 10^3$.

Arytmetyka zmiennopozycyjna – jedna z numerycznych reprezentacji liczb. Liczbę w arytmetyce zmiennopozycyjnej oznaczamy przez $rd(x)$.

$$rd(x) = \pm m_t \cdot \beta^C,$$

gdzie:
t – stała liczba cyfr mantysy
c – ustalony zakres cechy $c \in [c_{min}, c_{max}]$

Konwersja liczby rzeczywistej do zmiennopozycyjnej – następuje poprzez zaokrąglenie mantysy do najbliższej liczby t–cyfrowej (domyślnie), bądź obcięcie jej po t cyfrach.

Precyzja arytmetyki

IEEE 754 – jeden ze standardów reprezentacji liczb.
Niezerową liczbę rzeczywistą zgodnie z IEEE 754 zapisujemy w postaci

$$x = \pm m \cdot 2^c = \pm(1.f)_2 \cdot 2^c = \pm(1.a_1, a_2, \ldots)_2 \cdot 2^c,$$

gdzie:
$m \in [1, 2)$,
f – część ułamkową mantysy (t-1 bitów),
c – cecha liczbą całkowitą (d bitów) $c_{min} = -2^{d-1} + 1 \leqslant c \leqslant 2^{d-1} = c_{max}$

$a_i \in 0, 1$ (czyli działamy w systemie dwójkowym).

Liczba w IEEE754 składa się z 1 bita znaku, d bitów cechy i t-1 bitów na część ułamkową mantysy.

Przykład – reprezentacja $x = \frac{2}{3}$ w formacie single ($t - 1 = 23, d = 8$).

$$\frac{2}{3} = (0.10101010\ldots)_2, \texttt{więc} \approx (1.0101010\ldots11)_2 \cdot 2^{-1}$$

Reasumując:

$$x \approx \pm 1.f \cdot 2^c$$

3.2 Uwarunkowanie zadania

Jeśli niewielkie względne zmiany danych powodują duże względne odkształcenia wyników, to zadanie nazywamy źle uwarunkowanym.

4 Analiza Matematyczna

Funkcja ciągła – intuicyjnie:

- funkcja, w której mała zmiana argumentu powoduje małą zmianę wartości funkcji; inaczej mówiąc, dla argumentów leżących blisko siebie wartości funkcji też leżą blisko,
- funkcja rzeczywista, której wykresem jest ciągła linia tj. linia narysowana bez odrywania ołówka od papieru.

Formalnie:

- (Cauchy) funkcja jest ciągła w sensie Cauchy'ego w punkcie $x_0 \in M$, jeśli

$$(\forall \varepsilon > 0)(\exists \delta > 0)(\forall x \in M)(|x_0 - x| < \delta) \rightarrow (|f(x_0) - f(x)| < \varepsilon)$$

4.1 Ciągi

Ciągiem liczb rzeczywistych nazywamy dowolną funkcję $f : \mathbb{N} \rightarrow \mathbb{R}$ Wartości f (n) to wyrazy ciągu.

Granica właściwa ciągu

$$\lim_{n \to \infty} a_n = a,$$

gdy:

$$(\forall \varepsilon \geqslant 0)(\exists n_0 \in \mathbb{N})(\forall n \geqslant n_0 \in \mathbb{N})(|a_n - a| \leqslant \varepsilon)$$

Od pewnego miejsca wartości ciągu są bardzo bliskie prostej $y = a$, odległość od niej jest mniejsza od ε. Dla każdego epsilonu znajdziemy taki indeks ciągu, że każdy kolejny wyraz jest bliżej prostej $y = 0$ niż epsilon.

Ciąg zbieżny – ciąg który posiada granicę (dokładnie jedną). Każdy ciąg monotoniczny i ograniczony jest zbieżny. **Jak sprawdzić czy ciąg jest zbieżny?** – sprawdzić czy **jest monotoniczny i ograniczony**.

Ciąg monotoniczny – ciąg, który dla każdego $n \in \mathbb{N}$ spełnia pewien warunek:

- $a_{n+1} > a_n$ – rosnący
- $a_{n+1} \geqslant a_n$ – niemalejący
- $a_{n+1} < a_n$ – malejący
- $a_{n+1} \leqslant a_n$ – nierosnący

Monotoniczność badamy poprzez sprawdzenie $\frac{a_{n+1}}{a_n}$, w zależności od wyniku wskazuje to ciąg rosnący, stały, bądź malejący.

Ciąg ograniczony – ciąg, którego wszystkie wyrazy należą do pewnego przedziału skończonego, jest on jednocześnie ograniczony z góry i z dołu.

- ograniczony z góry jeżeli wszystkie wyrazy są mniejsze od pewnej ustalonej liczby.
- ograniczony z dołu jeżeli wszystkie wyrazy są większe od pewnej ustalonej liczby

4.2 Pochodna

Pochodna jest miarą szybkości zmian wartości funkcji względem zmiany argumentu – narzędzie dzięki któremu jesteśmy w stanie określić jak zmienia się wartość funkcji.

4.3 Całka

$TODO:$

4.4 Gradient

Gradient jest generalizacją pochodnej – pochodna określa to jak zmienia się funkcja jednej zmiennej, dla wielu zmiennych korzysta się z gradientu. Pochodna jest wartością skalarną, to gradient jest wektorową.

Funkcja, która każdemu punktowi przestrzeni przyporządkowuje pewną wielkość wektorową. Wskazuje ona kierunki najszybszych wzrostów wartości

danego obszar (pola skalarnego) w poszczególnych punktach.

W przypadku widzenia komputerowego (a dokładniej histogramu zorientowanych gradientów) gradient jest miarą zmiany piksela wzdłuż osi x i y dla każdego piksela (bądź obszaru).

Pole skalarne – przypisanie każdemu punktowi pewnej wielkości skalarnej (czyli jakiejś liczby R).

4.5 Macierz Jacobiego

Macierz Jacobiego jest macierzą stworzoną z pochodnych cząstkowych funkcji, której składowymi są funkcje rzeczywiste:

$$F(x, y, z, f(x, y, z), g(x, y, z))$$

Jakobian – wyznacznik macierzy Jacobiego. Wykorzystywana do badania lokalnego zachowania funkcji $\mathbb{R}^n \to \mathbb{R}^n$

Macierz Jacobiego vs. Gradient Gradient – wektor stworzony z pochodnych cząstkowych funkcji skalarnej. Macierz Jacobiego – macierz stworzona z pochodnych cząstkowych funkcji wektorowej.

$$\nabla f(x, y) = \begin{bmatrix} f'_x \\ f'_y \end{bmatrix}$$

$$J(f(x, y), g(x, y)) = \begin{bmatrix} f'_x & g'_x \\ f'_y & g'_y \end{bmatrix} = [\nabla f; \nabla g]$$

5 Algebra

Baza – zbiór liniowo niezależnych wektorów pokrywających całą przestrzeń

5.1 Macierz

5.1.1 Rząd macierzy

Maksymalna liczba liniowo niezależnych wektorów będących wierszami (lub kolumnami) tej macierzy

5.1.2 Wyznacznik macierzy

Wielkość mówiąca nam jak zmienia się wyjściowa powierzchnia obszaru zakreślonego przez wektory bazowe po transformacji. Innymi słowy – jeśli po transformacji pole rośnie 6 razy, to wyznacznik równy jest 6. Minusowe wartości wyznacznika oznaczają, że orientacja została odwrócona.

Jeśli wyznacznik jest równy 0, to wektory są liniowo zależne.

5.1.3 Dodawanie i odejmowanie macierzy

Macierze muszą być tych samych wymiarów, dodajemy / odejmujemy elementy na tych samych indeksach

5.1.4 Mnożenie / dzielenie przez skalar

Mnożymy / dzielimy każdy element macierzy

5.1.5 Mnożenie macierzy

Liczba kolumn pierwszej macierzy musi być równa liczbie wierszy drugiej

Właśności

- nie jest symetryczne $A \cdot B \neq B \cdot A$
- jest przechodnie $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

5.1.6 Macierz osobliwa

Wyznacznik równy zero.

5.1.7 Macierz jednostkowa

Wymnożenie przez nią nic nie zmienia.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.1.8 Odwrotność

Macierz odwrotna do A to A^{-1}. Wynikiem $A \cdot A^{-1}$ jest macierz jednostkowa.

5.1.9 Transpozycja

Obrócenie macierzy o 90 stopni zgodnie ze wskazówkami zegara.

$$A = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}, A^T = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

5.2 Wektory

5.3 Wektory bazowe

$\hat{i}$ i $\hat{j}$

Każdy wektor może zostać opisany poprzez liniową transformację tych wektorów. W przekształceniach liniowych macierz przekształceń zawiera informację gdzie $\hat{i}$ i $\hat{j}$ się znajdują po przejściu.

$$\begin{bmatrix} i_x & j_x \\ i_y & j_y \end{bmatrix}$$

5.3.1 Wektor liniowo zależny

Wektory x i y są liniowo zależne jeśli

$$(\exists k)(\overrightarrow{x} = k \cdot \overrightarrow{y})$$

5.4 Transformacja liniowa

Sposób przemieszenia przestrzeni spełniające dwa warunki:

- wszystkie linie pozostają liniami
- środek układu współrzędnych nie może się przesunąć

(linie siatki pozostają równoległe i równo odległe)

Kompozycja – możemy dwa działania zawrzeć w jednej macierzy (zamiast robić macierz shear, potem rotation, użyjmy jednej macierzy będącej iloczynem tych dwóch).

5.5 Transformacja afiniczna

Przekształcenie wykorzystywane np. w grafice komputerowej. Przekształcenie jest afiniczne, jeśli jest w postaci

$$x \to Ax + b,$$

gdzie:

A – macierz przekształcenia liniowego, b – wektor przesunięcia.

Standardowo używa się macierzy do reprezentacji przekształceń liniowych, a dodawania wektorów do reprezentacji przesunięć. Da się jednak przedstawić to w formie jednej macierzy – dodając po prawej stronie kolumnę – wektor przesunięcia, a na dole wiersz samych 0 (oprócz kolumny gdzie jest wektor – tam 1).

Zasady:

- linie pozostają liniami
- współliniowość zachowana
- punkt niekoniecznie ten sam

Rodzaje przekształceń:

Przesunięcie

$$\begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix}$$

Obrót

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Skala

$$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.6 Metoda Gaussa–Jordana

Służy do:

- obliczania rzędu macierzy
    1. Sprowadź do macierzy schodkowej
    2. Liczba schodków niezerowych to rząd macierzy
- obliczania macierzy odwrotnej
- obliczenia wartości wyznacznika
- wyznaczenia rozkładu LU
- rozwiązywania układów równań liniowych

6 Grafika

6.1 Transformata

Wynik działania transformacji.

6.2 Dyskretna transformacja kosinusowa

Transformacja popularna w dziedzinie stratnej kompresji danych. Najczęściej używana w plikach typu jpeg i mpeg. Wykorzystuje dzielenie na bloki o stałych rozmiarach. Ze względu na skoki jakości między blokami nie jest wykorzystywana do przetwarzania sygnałów dźwiękowych (słychać by było trzaski).

6.3 Transformata Fouriera

Transformacja Fouriera rozkłada funkcję okresową na szereg funkcji okresowych tak, że uzyskana transformata podaje w jaki sposób poszczególne częstotliwości składają się na pierwotną funkcję.

Przechodzimy z dziedziny czasu na dziedzinę częstotliwości.

Wykorzystywane w przetwarzaniu sygnałów – przykładowo w edycji dźwięku. Jeśli chcemy usunąć pewne piski, bądź szumy z nagrania to najpierw używamy transformacji Fouriera, by znaleźć odpowiednie częstotliwości składowe, a następnie wyciszamy niepożądaną. Powrót do wyjściowej formy zapewnia odwrotna transformacja Fouriera.

Inny przykład – algorytm Retinex.

6.4 FFT

Algorytm wyznaczenia dyskretnej transformaty Fouriera. Złożoność wynosi $O(n \log n)$, gdzie standardowo jest to $O(n^2)$.

6.5 Operacje morfologiczne

6.5.1 Splot

Splotem nazywamy operację, która wykonywana jest na dwóch funkcjach – np. $f(t)$ oraz $g(t)$. Jej wynikiem jest trzecia funkcja, która jest postrzegana

jako zmodyfikowana wersja oryginalnych funkcji. Konwolucja jest operacją symetryczną (f * g == g * f).

Jej wynikiem jest nowa funkcja – $h(t)$.

Całka iloczynu dwóch funkcji, z czego jedna jest odwrócona i przesunięta.

$$h(t) = (f \bullet g)(t) = \int_0^t f(x)g(t-x)dx$$

Możliwe jest również wykonanie splotu na wartościach dyskretnych. Wykorzystywane jest to np. w cyfrowym przetwarzaniu obrazu.

$$h[m,n] = (f \bullet g)[m,n] = \sum_i^m \sum_j^n f[i,j]g[m-i,n-j]$$

Jest to wymnożenie wartości z obu sygnałów wejściowych, a następnie ich zsumowanie.

Jądra W zależności od zadania stosuje się różne jądra – macierze służące do odpowiedniego przekształcenia macierzy wejściowej (czyli obrazu).

Przykładowo:

- zwiększająca ostrość macierz górnoprzepustowa (same -1, w środku 9)

Problem brzegu – następuje w przypadku gdy jądro wyjdzie poza zakres. Przykładowe rozwiązania:

- poza sygnałem wartości to 0
- powtórzenie obrazu
- modyfikacja maski filtru na brzegach, by nie wychodziła poza zakres

7 Grafy

Graf – zbiór wierzchołków, które mogą być połączone krawędziami w taki sposób, że każda krawędź kończy się i zaczyna w którymś z wierzchołków.

Rząd grafu – liczba wierzchołków w grafie

Stopień wierzchołka – nazywamy liczbę krawędzi, które łączą się z danym wierzchołkiem. Jeśli graf posiada pętle, to liczymy je za 2.

Droga – uporządkowany ciąg kolejnych krawędzi, po których należy przejść, aby dotrzeć z wierzchołka startowego do wierzchołka końcowego.

Cykl – droga zamknięta, czyli taka, której koniec (ostatni wierzchołek) jest identyczny z początkiem (pierwszym wierzchołkiem).

7.0.1 Rodzaje grafów:

Graf planarny – da się go narysować na płaszczyźnie tak, aby żadne jego krawędzie się nie przecinały.

Graf pełny – każda para wierzchołków łączy się krawędzią.

Graf cykliczny – graf w którym jesteśmy w stanie znaleźć cykl.

Graf acykliczny – graf niezawierający cykli.

Graf skierowany – graf w którym ruch możliwy jest tylko w kierunkach wskazywanych przez krawędzie (sieć ulic, z których każda jest jednokierunkowa)

Cykl Hamiltona – cykl przechodzący przez wszystkie wierzchołki grafu dokładnie raz.

Graf hamiltonowski – graf posiadający cykl Hamiltona.

Cykl Eulera – taki cykl w grafie, który przechodzi przez każdą jego krawędź dokładnie raz.

Graf eulerowski – graf zawierający graf Eulera

Graf półeulerowski –

Graf Petersena –

Las – acykliczny graf nieskierowany

7.0.2 Algorytmy na grafie

BFS – „Breadth First Search", Zaczynając od korzenia, polega na dodawaniu sąsiadujących wierzchołków na kolejkę, następnie braniu pierwszego z kolejki i powtarzaniu całości (jakby był rootem). Gdy nie ma gdzie iść, bierzemy kolejne z kolejki i powtarzamy. W przypadku wyczerpania kolejki i braku ruchów - odwiedziliśmy wszystkie.

DFS – „Depth First Search", Zaczynając od korzenia, polega na wrzucaniu sąsiadujących wierzchołków na stos, następnie ściąganiu pierwszego elementu ze stosu i powtarzaniu algorytmu, traktując ściągnięty element jako korzeń. W przypadku wyczerpania stosu i braku ruchów - odwiedziliśmy wszystkie.

Dijkstra – Algorytm znajdowania najkrótszej ścieżki w grafie ważonym między wierzchołkiem, a wszystkimi pozostałymi. Do optymalnego działania używa **kopca**.

Prim – Algorytm znajdywania minimalnego drzewa rozpinającego grafu, tj.

Kruskal – algorytm wyznaczający MST, jeśli graf jest spójny. Algorytm zachłanny.

Złożoność czasowa – $O(E \cdot \log V)$

8 Probabilistyka

Zmienna losowa – funkcja przypisująca zdarzeniom elementarnym liczby

Zbiór – jedno z podstawowych pojęć matematycznych, nie ma jasno określonego opisu, jednakże rozumiemy go jako kolekcję niepowtarzających się obiektów bez wyznaczonej kolejności.

Rodzina zbiorów – zbiór zbiorów

8.1 Prawdopodobieństwo

Nieformalnie – szansa na wystąpienie jakiegoś zdarzenia.

Formalnie (Laplace) – niech Ω to skończony zbiór wszystkich możliwych zdarzeń elementarnych, a podzbiór A zbioru Ω to zdarzenie. **Prawdopodobieństwem** nazywamy stosunek liczby zdarzeń elementarnych do liczby zdarzeń zbioru Ω – $\mathbb{P}(A) = \frac{|A|}{|\Omega|}$

8.2 Przestrzeń probabilistyczna

8.3 Zdarzenia

8.3.1 Elementarne

Pojęcie pierwotne (niedefiniowane), niepodzielny wynik pewnego doświadczenia. Zbiór wszystkich zdarzeń elementarnych jest przestrzenią zdarzeń elementarnych – Ω

Przestrzeń zdarzeń elementarnych może być zbiorem:

- skończonym
- przeliczalnym
- nieprzeliczalnym

8.3.2 pewne

cały zbiór Ω (zdarzenie które musi zajść)

8.4 Przeliczalnie addytywne ciało zdarzeń

Niech $\Omega \subseteq \mathbb{R}$, czyli jest n–wymiarową przestrzenią kartezjańską lub jej podzbiorem.

Niech S^* oznacza dowolną rodzinę podzbiorów Ω spełniającą warunki:

- $\Omega \in S^*$ –
- $(\forall A \in S^*)(\bar{A} = \Omega - A \in S^*)$ – zarówno zdarzenie, jak i jego dopełnienie należy do S
-

Zbiór borelowski – najmniejsza z rodzin przeliczalnie addytywnych ciał zdarzeń

8.5 Rozkład prawdopodobieństwa

Definiujemy funkcję P następująco:

- $P : S \to \mathbb{R}$ – funkcja mapująca element zbioru borelowskiego na liczbę rzeczywistą (np. prawdopodobieństwo 50%)
- $(\forall A \in S)(P(A) \geqslant 0)$ – prawdopodobieństwo każdego z elementów zbioru jest większe, równe 0
- $P(\Omega) = 1$ – prawdopodobieństwo całego zbioru wynosi 1
- jeżeli $A_1, \ldots$ jest dowolnym ciągiem elementów rodziny S parami rozłącznych $(A_i, \cap A_j = \emptyset$ dla każdych $i \neq j)$, to prawdopodobieństwo sumy zbiorów jest równe sumie prawdopodobieństw każdego z elementów np. prawdopodobieństwo Ω jest równe sumie prawdopodobieństw każdego z jej elementów

$$(\forall A_1, \ldots \in S)(\forall A_i, A_j)(A_i \cap A_j = \emptyset) \to P(A_1, \ldots \in S) = P\Big(\bigcup_{i=1}^{\infty} A_i\Big) = \sum_{i=1}^{\infty} P(A_i)$$

Przykład:
Rzut monetą z jednakowym prawdopodobieństwem,

$$X(O) = -1, X(R) = 1$$

$$P(X \in A) = \begin{cases} 0 & \text{dla} \quad A = \mathbb{R} \setminus \{-1, 1\} \\ \frac{1}{2} & \text{dla} \quad A = \{-1\} \quad \text{lub} \quad A = \{1\} \\ 1 & \text{dla} \quad A = \{-1, 1\} \end{cases}$$

Prawdopodobieństwo zdarzeń losowych – wartość zwracana przez funkcję rozkładu prawdopodobieństwa

Gęstość rozkładu prawdopodobieństwa (gęstość zmiennej losowej) – funkcja rzeczywista, określona dla rozkładu prawdopodobieństwa, taka że całka z tej funkcji, obliczona w odpowiednich granicach, jest równa prawdopodobieństwu wystąpienia danego zdarzenia losowego.

$$P(B) = \int_B f(x)dx$$

Możemy z niej wyznaczyć dystrybuantę i wartość oczekiwaną.

Dystrybuanta – dystrybuanta rozkładu X w punkcie t to prawdopodobieństwo, że zajdzie zdarzenie mniejsze, bądź równe t:

$$F_x(t) = P(X \leqslant t) = \sum P(x = k) \quad \text{dla} \quad k \leqslant t$$

Przykładowo mamy rozkład $P(X = 1) = \frac{1}{3}$, $P(X = 2) = \frac{1}{3}$, $P(X = 3) = \frac{1}{3}$. Dystrybuanta dla 2 jest równa:

$$F_X(2) = P(X \leqslant 2) = P(X = 1) + P(X = 2) = \frac{2}{3}$$

Prawdopodobieństwo warunkowe Zajście zdarzenia A pod warunkiem zajścia zdarzenia B.

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Zmienna losowa – funkcja przypisująca zdarzeniom elementarnym wartość liczbową. Na przykład – 1, gdy wyrzucimy orła, 0 gdy reszkę. Zwykle zapisywana wielką literą.

Wartość oczekiwana – spodziewany wynik doświadczenia losowego. Dla zmiennej losowej X o **rozkładzie dyskretnym** i przyjmującej skończoną liczbę wartości $x_1, x_2, \ldots, x_n$ z prawdopodobieństwem odpowiednio $p_1, p_2, \ldots, p_n$ wyraża się wzorem:

$$EX = \sum_{i=1}^{n} x_i p_i$$

Wariancja – miara zmienności. Średnia arytmetyczna z kwadratu różnic między: poszczególnymi wartościami, a wartością oczekiwaną.

Odchylenie standardowe – miara zmienności. Intuicyjnie mówi o tym, jak szeroko wartości jakiejś wielkości są rozrzucone wokół jej średniej. Dla zmiennej losowej X o **rozkładzie dyskretnym** i przyjmującej skończoną liczbę wartości $x_1, x_2, \ldots, x_n$ z prawdopodobieństwem odpowiednio $p_1, p_2, \ldots, p_n$ wyraża się wzorem:

$$\sigma = \sqrt{\sum_{i=1}^{n} (x_i - \theta)^2 p_i}$$

gdzie θ to wartość oczekiwana.

9 Machine Learning

9.1 Normalizacja

Procedura wstępnej obróbki danych w celu umożliwienia ich wzajemnego porównywania i dalszej analizy.

Standaryzacja Normalizacja zmiennej losowej w wyniku czego średnia wartość oczekiwana jest równa 0, a odchylenie standardowe 1.

$$z = \frac{x - \Delta}{\sigma}$$

- x – zmienna do ustandaryzowania
- Δ – średnia z populacji
- σ – odchylenie standardowe

Normalizacja wektora Normalizacja wektora polega na przeskalowaniu go przez odwrotność jego długości. W wyniku otrzymujemy wektor jednostkowy – wektor o długości jeden.

9.2 Drzewo decyzyjne

Drzewo decyzyjne to drzewo (acykliczny graf w którym każda para wierzchołków połączona jest dokładnie jedną ścieżką). Używany jest do klasyfikacji na podstawie przykładów uczących. Tworzenie drzewa decyzyjnego polega na cyklicznym podziale wektorów uczących wg. wybranych kryteriów. Działają dobrze przy danych testowych, w przypadku danych prawdziwych nie są jednak tak dobre.

Drzewo regresji (ang. regression tree) – typ drzewa decyzyjnego, którego wynikiem jest liczba rzeczywista

Drzewo klasyfikacji (ang. classification tree) – typ drzewa decyzyjnego, którego wynikiem jest klasa (wartość dyskretna)

9.3 Las losowy

Las losowy – struktura zbudowana z wielu drzew decyzyjnych. Wykorzystywane do tego są losowo wybrane podzbiory danych uczących (boostrapped dataset). Każde z drzew budowane przy wykorzystaniu pewnej ilości cech (losowej) – kolumn. Klasyfikacja wykonywana jest za pomocą głosowania. Sprawdzenie precyzji struktury odbywa się poprzez wykorzystanie danych, które nie zostały użyte do bootstrapowania.

9.4 Sieć neuronowa

Sieć neuronowa – struktura inspirowana sieciami neuronowymi żywego mózgu. **Wynik** przez nią zwracany **jest otrzymywany przez liczne przekształcenia**.

Sieć neuronowa **składa się z warstw – wejściowej, pośrednich i wyjściowej**. Jej budowę można przedstawić za pomocą **grafu**. Krawędzie – **synapsy, mają pewne wagi** determinujące jak ważny jest wynik przez nie przechodzący. Wierzchołki – **neurony, modyfikują wartości**, które się do nich dostaje poprzez synapsy.

Sieć nie działa od razu poprawnie – **wagi są początkowo losowe, bądź jednolite**. Poprzez proces **propagacji wstecznej** dostosowują się one. Backpropagation polega na użyciu par wejście–wyjście. Jeśli wynik zwracany dla danej wejściowej jest niepoprawny to następuje **korekcja wag**, by był on bliższy temu oczekiwanemu.

Konwolucyjna sieć neuronowa – sieć będąca grafem skierowanym. Stosowana do np. rozpoznawania obiektów na obrazie. Myślą stojącą za tym rodzajem sieci jest, **wraz z kolejną warstwą, detekcja coraz bardziej szczegółowych cech**.

Warstwy:

- konwolucji – uwydatnienie i ekstrakcja szukanych cech (np. wyostrzenie obrazu, zmiana kontrastu)
- ReLU – nadanie wynikom konwolucji nieliniowości (w p.p. sieć byłaby w stanie obliczać tylko liniowe funkcje)
- pooling – zmniejszenie rozmiaru reprezentacji (przejście jądrem np. 2×2 i zastąpienie jedną wartością – średnią, bądź maksymalną)

- pełne połączenie – zwraca ona wektor o długości równej liczbie klas, zawiera prawdopodobieństwo

9.5 Funkcja straty

Funkcja straty (kosztu) – narzędzie służące do miary dokładności naszej hipotezy.

Na przykładzie liniowej regresji – jest to suma różnica pomiędzy wartością hipotezy w punkcie x, a prawdziwą wartością. Algorytm Gradient Descent jest zagadnieniem optymalizacyjnym mającym na celu minimalizację błędu – dzięki krokowi (learning rate) i pochodnej funkcji kosztu schodzimy po funkcji kosztu, aż dojdziemy do minimum.

Funkcja straty:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Hipoteza (funkcja liniowa):

$$h_\theta(x) = \theta_0 + \theta_1 x$$

9.6 Funkcja Aktywacji

Funkcja aktywacji odpowiedzialna jest za **podjęcie decyzji czy neuron powinien zostać aktywowany** – tj. czy informacja jest potrzebna w dalszej części sieci czy też powinna zostać zignorowana. Jeśli wartość zwracana przez neuron jest większa niż pewien zakres – aktywujemy neuron, w p.p. ignorujemy. Ujemne wartości w funkcjach aktywacji mają pokazać, że rozwiązanie nie polepsza wyniku, a szkodzi mu. Funkcja $\phi : \mathbb{R} \to \mathbb{R}$

Przykłady funkcji aktywacji:

- funkcja progowa unipolarna –

$$A(x) = \begin{cases} 0 & \text{dla} \quad x < a \\ 1 & \text{dla} \quad x \geqslant a \end{cases}$$

- funkcja progowa bipolarna –

$$A(x) = \begin{cases} -1 & \text{dla} \quad x < a \\ 1 & \text{dla} \quad x \geqslant a \end{cases}$$

- sigmoida – funkcja aktywacji, dla której wartości od -2 do 2 zmieniają się bardzo mocno, w dalszych etapach, gdzie funkcja nie rośnie / maleje już tak mocno mamy do czynienia z problemem zanikającego gradientu

$$y(x) = \frac{1}{1 + e^{-x}}$$

- ReLU – jedna z najpopularniejszych funkcji aktywacji, znajduje się w CNN, jest mniej kosztowna obliczeniowo od sigmoidów itd.

$$A(x) = \max(0, x)$$

  Ma wariant leaky, gdzie poniżej zera jest wartość niezerowa (np. $y = 0.01x$)

- softmax – znajduje się na przykład w warstwie pełnego połączenia w CNN. Polega na ściśnięciu prawdopodobieństw wszystkich klas, tak by ich suma była równa 1 – tj. bierze wektor wartości rzeczywistych i normalizuje je tak, że ich suma = 1.

$$f_j(z) = \frac{e^z j}{\sum_k e^z k}$$

9.7 Propagacja wsteczna

Propagacja wsteczna (ang. backpropagation) jest algorytm służący do nauki sieci neuronowej. Porównuje on daną wyjściową z sieci do wartości oczekiwanej i na jej podstawie determinuje jak zmienione powinny zostać wagi, preferencja (ang. bias) i stopień aktywacji w celu obniżenia funkcji kosztu.

$$TODO:$$

Przykładowo w sieci rozpoznającej cyfry który na podstawie różnicy danych wyjściowych z sieci, a wartości oczekiwanej

9.8 K–NN (K-Nearest Neighbors)

K-NN jest algorytmem zarówno wykorzystywanym przy **klasyfikacji**, jak i **regresji**. Nazywa się go nieparametrycznym (non–parametric) – nie ma żadnej fazy uczenia, nie ma żadnych założeń (generalizacji) względem rozkładu

danych – model jest tworzony wprost z danych. Model trzymany w pamięci urządzenia.

Przypisanie działa na zasadzie podobieństwa – wysokopoziomowo chodzi o postawienie nowego przypadku w przestrzeni, obliczenia odległości od innych punktów, wybrania pewnej ilości k najbliższych punktów, a następnie na zasadzie większości przypisanie analizowanego do odpowiedniej klasy.

Używany jest wtedy gdy jest mała wiedza na temat rozkładu danych.

9.9 K-means

Algorytm klasyfikacji nauczania bez nadzoru, gdzie k – oznacza liczbę klastrów. Składa się z następujących etapów:

1. losowo wybierz k centroidów klastrów
2. przypisz punkty do najbliższego klastra
3. przenieś centroid klastra do średniej punktów należących do niego
4. dopóki jest jakaś zmiana powtórz od punktu 2.

9.10 Problem optymalizacyjny

Problem obliczeniowy polegający na znalezieniu najmniejszej (problem minimalizacji), bądź największej (problem maksymalizacji) wartości pewnego parametru problemu – nazywamy go funkcją kosztu (celu).

Przykład:

- regresja liniowa – problem minimalizacji odległości funkcji od danych punktów

9.11 Perceptron

Model składający się

- określonej liczby n wejść
- wagi dla każdego wejścia ($w_i \in \mathbb{R}$)
- funkcji aktywującej

Wynik perceptronu to:

$$\text{out} = \phi(\sum_{i=1}^{n} x_i w_i),$$

gdzie:

- x_i – wejście
- w_i – waga wejścia

9.12 Histogram zorientowanych gradientów

Rozkład gradientów traktowany jest jako features – wynika to z tego powodu, że gradienty są spore na krawędziach i rogach, a to one dają nam dużo informacji o kształcie obiektu.