

Teoretyczne podstawy informatyki



Wykład 12a:

Prawdopodobieństwo i algorytmy probabilistyczne

<http://hibiscus.if.uj.edu.pl/~erichter/Dydaktyka2010/TPI-2010>

Teoria prawdopodobieństwa

- Teoria prawdopodobieństwa, szeroko stosowana we współczesnej nauce, ma również wiele zastosowań w informatyce, np.:
 - szacowanie czasu działania programów dla przypadków ze średnimi, czyli typowymi danymi wejściowymi,
 - wykorzystanie do projektowania algorytmów „podejmujących decyzje” w niepewnych sytuacjach, np. najlepsza możliwa diagnoza medyczna na podstawie dostępnej informacji,
 - algorytmy typu Monte Carlo,
 - różnego rodzaju symulatory procesów,
 - **prawie zawsze „prawdziwe” rozwiązania.**

Teoria prawdopodobieństwa

□ **Przestrzeń probabilistyczna Ω :**

Skończony zbiór punktów, z których każdy reprezentuje jeden z możliwych wyników doświadczenia. Każdy punkt x jest związany z taką nieujemną liczbą rzeczywistą zwaną prawdopodobieństwem x , że suma prawdopodobieństw wszystkich punktów wynosi **1**.

Istnieje także pojęcie nieskończonych przestrzeni probabilistycznych ale nie mają one większego zastosowania w informatyce.

□ **Zdarzenie E :**

Podzbiór punktów w przestrzeni probabilistycznej.

Prawdopodobieństwo zdarzenia, $P(E)$, jest sumą prawdopodobieństw punktów należących do tego zdarzenia.

□ **Dopełnienie zdarzenia E czyli \bar{E} :**

Zbiór punktów przestrzeni probabilistycznej które nie należą do zdarzenia E .

$$P(E) + P(\bar{E}) = 1.$$

Prawdopodobieństwo warunkowe

- Prawdopodobieństwem warunkowym zajścia zdarzenia **F** pod warunkiem zajścia zdarzenia **E**, gdzie $P(E) > 0$ nazywamy liczbę:
 $P(F | E) = P(E \cap F) / P(E)$.
- Jest to iloraz prawdopodobieństwa części wspólnej zdarzeń **E**, **F** i prawdopodobieństwa zdarzenia **E**.
- Zdarzenia **E**, **F** nazywamy niezależnymi jeśli zachodzi:
 $P(E \cap F) = P(E) \cdot P(F)$
- W przeciwnym wypadku zdarzenia są zależne.
- Dla zdarzeń niezależnych **E**, **F** zachodzi:
 $P(F | E) = P(F)$

Przykłady

□ Zdarzenia niezależne:

Rzucamy dwoma kostkami, wyrzucenie liczby „1” na pierwszej kostce (zdarzenie E) nie wpływa na możliwość pojawienia się liczby „1” na drugiej kostce (zdarzenie F). $P(F | E) = P(F)$

□ Zdarzenia zależne:

Ciągniemy dwa razy kartę z talii kart. Wyciągnięcie jako pierwszej karty asa (zdarzenie E), wpływa na możliwość wyciągnięcia jako drugiej karty asa (zdarzenie F). $P(F | E) \neq P(F)$.

- W niektórych sytuacjach liczenie prawdopodobieństw jest łatwiejsze jeżeli podzielimy przestrzeń probabilistyczna na rozdzielne obszary R_1, R_2, \dots, R_k . Wówczas $P(E) = \sum_{i=1}^k P(E | R_i) \cdot P(R_i)$.

Przykład z kartami

- Ciągniemy dwie karty z talii **52** kart. Liczba możliwych wyników tego doświadczenia (czyli wariacji bez powtórzeń) wynosi $|\Omega| = 52 * 51 = 2652$.
- Oznaczmy poprzez **E** zdarzenie polegające na wyciągnięciu jako pierwszej karty As'a.
 $|E| = 4 * 51 = 204$.
 $P(E) = |E| / |\Omega| = 204/2652 = 1/13$.
- Prawdopodobieństwo wyciągnięcia As'a jako drugiej karty, (**zdarzenie F**), jeżeli pierwsza wyciągnięta karta to był As jest $P(F|E) = P(E \cap F) / P(E) = 4 \cdot 3 / 204 = 1/17$.
 $P(E) \cdot P(F|E) = 1/13 \cdot 1/17 = 1/221$.
- Podzielmy przestrzeń na dwa obszary:
 - R_1 - pierwszą kartą jest As, $|R_1| = 4 \cdot 51 = 204$.
 - R_2 - pierwszą kartą nie jest As, $|R_2| = 2652 - 204 = 2448$.
 - $P(E \cap F) = P((E \cap F) | R_1) \cdot P(R_1) + P((E \cap F) | R_2) \cdot P(R_2)$
 - $P(E \cap F | R_2) = 0$
 - $P(E \cap F | R_1) = 4 \cdot 3 / 4 \cdot 51 = 1/17$
 - $P(R_1) = 204 / 2652 = 1/13$
 - $P(E \cap F) = P((E \cap F) | R_1) \cdot P(R_1) + P((E \cap F) | R_2) \cdot P(R_2) = 1/17 \cdot 1/13 + 0 \cdot P(R_2) = 1/221$.
- Gdybyśmy po wyciągnięciu pierwszej karty zwracali ją z powrotem do talii, to mielibyśmy $P(F|E) = P(F) = 1/13$ (zdarzenia niezależne).
- Wówczas $P(E) \cdot P(F|E) = 1/13 \cdot 1/13 = 1/169$.

Reguły związane z wieloma zdarzeniami

Oznaczmy:

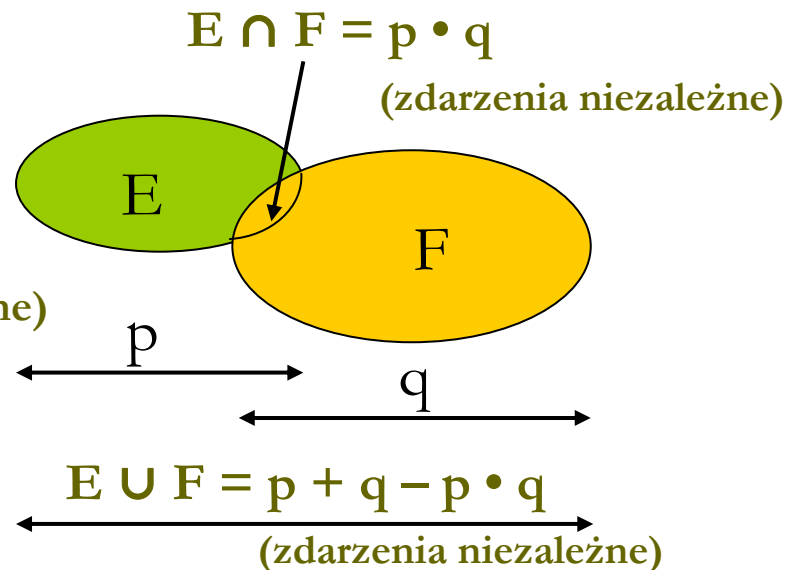
$$P(E) = p, P(F) = q.$$

Wówczas:

$$\max(0, (p+q-1)) \leq P(E \cap F) \leq \min(p, q)$$

$$P(E \cup F) = p + q - P(E \cap F)$$

$$P(E \cup F) = p + q - p \cdot q \text{ (zdarzenia niezależne)}$$



- W zastosowaniach czasem akceptujemy że nie możemy wyznaczyć dokładnie prawdopodobieństw oraz zależności między zdarzeniami. Potrafimy tylko wskazać sytuacje najmniej lub najbardziej prawdopodobne.
- **Zastosowanie: różnego typu diagnostyka**

Oczekiwane wartości obliczeń i analiza probabilistyczna

- Przypuśćmy, że mamy pewną funkcję określoną na przestrzeni probabilistycznej $f(\mathbf{x})$. Wartość oczekiwana tej funkcji po wszystkich punktach przestrzeni $E(f) = \sum f(\mathbf{x}) P(\mathbf{x})$.
- Mamy tablicę n liczb całkowitych, sprawdzamy czy jakaś liczba całkowita „ \mathbf{x} ” jest elementem tej tablicy. Algorytm przegląda całą tablicę, po napotkaniu $A[i] = \mathbf{x}$ kończy działanie.
 - Jeżeli $A[0] = \mathbf{x}$ to algorytm $O(1)$
 - Jeżeli $A[n-1] = \mathbf{x}$ to algorytm $O(n)$
- $E(f) = \sum_{i=0}^{n-1} (c i + d) \cdot (1/n) = c \cdot (n-1) / 2 + d$
- $E(f) \sim c \cdot n/2$ dla dużego n

1	$A[0]$
8	
7	
5	
3	
4	$A[i]$
8	
9	
7	$A[n-1]$

Algorytmy wykorzystujące prawdopodobieństwo

- Jest bardzo wiele różnych typów algorytmów wykorzystujących prawdopodobieństwo.
- Jeden z nich to tzw. **algorytmy Monte-Carlo** które wykorzystują liczby losowe do zwracania albo wyniku pożądanego („prawda”), albo żadnego („nie wiem”).
Wykonując algorytm **stałą** liczbę razy, możemy rozwiązać problem, dochodząc do wniosku, że jeśli żadne z tych powtórzeń nie doprowadziło nas do odpowiedzi „prawda”, to odpowiedzią jest „fałsz”.
Odpowiednio dobierając liczbę powtórzeń, możemy dostosować prawdopodobieństwo niepoprawnego wniosku „fałsz” do tak niskiego poziomu, jak w danym przypadku uznamy za konieczne.
- **Nigdy** jednak nie osiągniemy prawdopodobieństwa popełnienia błędu na poziomie **zero**.

Co to są liczby losowe?

- ❑ Mówimy, że wyniki pewnych doświadczeń są **losowe**, co oznacza że wszystkie możliwe wyniki są równie prawdopodobne.
- ❑ Przykładowo, jeżeli rzucamy normalną (prawkidłową) kostką do gry to zakładamy że nie ma możliwości fizycznego kontrolowania wyniku tego rzutu w taki sposób aby jeden wynik był bardziej prawdopodobny od drugiego.
- ❑ Podobnie zakładamy że mając uczciwie potasowana talie kart, nie możemy wpłynąć na wynik - prawdopodobieństwo otrzymania w rozdaniu każdej karty jest identyczne.

Co to są liczby losowe?

- **Wszystkie** generowane przez komputer **losowe** sekwencje są wynikiem działania specjalnego rodzaju algorytmu zwanego **generatorem liczb losowych** (ang. random number generator). Zaprojektowanie takiego algorytmu wymaga specjalistycznej wiedzy matematycznej. Przykład prostego generatora który całkiem dobrze sprawdza się w praktyce to tzw. “**liniowy generator kongurencyjny**”. Wyznaczamy **stałe** $a \geq 2$, $b \geq 1$, $x_0 \geq 0$ oraz **współczynnik** $m > \max(a, b, x_0)$. Możemy teraz wygenerować sekwencje liczb x_1, x_2, \dots za pomocą wzoru:

$$x_{n+1} = (a x_n + b) \bmod(m)$$

- Dla właściwych wartości stałych a, b, m oraz x_0 , sekwencja wynikowa będzie wyglądała na losową, mimo że została ona wygenerowana przy użyciu konkretnego algorytmu i na podstawie “jądra” x_0 .
- **Dla szeregu zastosowań istotna jest odtwarzalność sekwencji liczb losowych.**

Algorytmy wykorzystujące prawdopodobieństwo

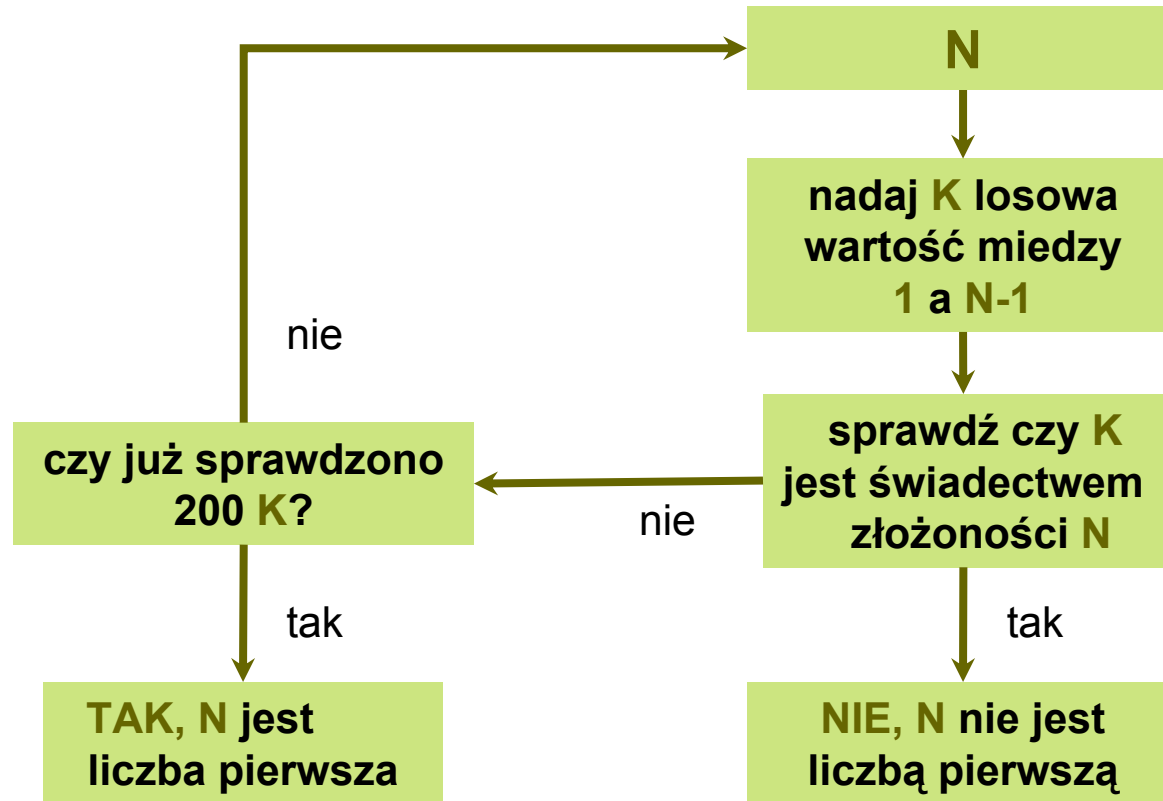
- Mamy pudełko w którym jest n -procesorów, nie mamy pewności czy zostały przetestowane przez producenta. Zakładamy że prawdopodobieństwo że procesor jest wadliwy (w nieprzetestowanym pudełku) jest 0.10.
- **Co możemy zrobić aby potwierdzić czy pudełko dobre?**
 - przejrzeć wszystkie procesory \rightarrow algorytm $O(n)$
 - losowo wybrać k procesorów do sprawdzenia \rightarrow algorytm $O(1)$
 - błąd polegałby na uznaniu że pudełko dobre (przetestowane) jeżeli nie było takie.
- Losujemy **$k=131$** procesorów.
Jeżeli procesor jest dobry odpowiadamy „nie wiem”. Prawdopodobieństwo że „nie wiem” dla każdego z k -procesorów $(0.9)^k = (0.9)^{131} = 10^{-6}$.
 10^{-6} to jest prawdopodobieństwo że pudełko uznamy za dobre choć nie było testowane przez producenta.
Za cenę błędu $= 10^{-6}$, zamieniliśmy algorytm z $O(n)$ na $O(1)$.
Możemy regulować wielkość błędu/czas działania algorytmu zmieniając k .

Probabilistyczne algorytmy sprawdzania

„ Czy liczba N jest liczbą pierwszą ?”

- ❑ W połowie lat 70-tych odkryto **dwa bardzo eleganckie probabilistyczne algorytmy** sprawdzające, czy liczba jest pierwsza. Były one jednymi z pierwszych rozwiązań probabilistycznych dla trudnych problemów algorytmicznych. Wywołały fale badań które doprowadziły do probabilistycznych rozwiązań wielu innych problemów.
- ❑ Oba algorytmy wykonują się w czasie wielomianowym (niskiego stopnia), zależnym od liczby cyfr w danej liczbie N (czyli $O(\log N)$).
- ❑ Oba algorytmy są oparte na losowym szukaniu pewnych rodzajów potwierzeń lub **świadcstw złożoności** liczby N .
- ❑ Po znalezieniu takiego świadectwa algorytm może się bezpiecznie zatrzymać z odpowiedzią „**nie, N nie jest liczbą pierwszą**”, ponieważ istnieje bezdyskusyjny dowód że N jest liczbą złożoną.
- ❑ Poszukiwanie musi być przeprowadzone w taki sposób aby w pewnym rozsądnym czasie algorytm mógł przerwać szukanie odpowiadając, że N jest liczbą pierwszą z bardzo małą szansą omyłki.
- ❑ Trzeba zatem znaleźć dająca się **szybko sprawdzać** definicje świadectwa złożoności.

„Czy liczba N jest liczbą pierwszą ?



jeśli to jest odpowiedź to jest to prawda z błędem $1/2^{200}$

jeśli to jest odpowiedź to jest to prawda

Świadectwa złożoności (zarys)

- Każda liczba parzysta poza 2 to jest złożona
- Jeżeli suma cyfr liczby jest podzielna przez 3 to liczba jest złożona (iteracyjny prosty algorytm liniowo zależny od liczby cyfr)
- **Test pierwszości Fermata:**
 - jeśli n jest liczbą pierwszą oraz k jest dowolna liczba całkowita $(1, n-1)$, to $k^{n-1} \equiv 1 \pmod{n}$.
 - natomiast jeśli n jest liczbą złożoną (z wyjątkiem kilku złych liczb złożonych – liczb Carmichael’a) oraz jeśli k wybierzemy losowo z przedziału $(1, n-1)$ to prawdopodobieństwo tego że $k^{n-1} \not\equiv 1 \pmod{n}$ jest mniejsze niż $1/2$.
 - Zatem liczby złożone (poza liczbami Carmichael’a) spełniają warunek testu dla danego k z prawdopodobieństwem nie mniejszym niż $1/2$.
- **Test pierwszości Solovay-Strassena:**
 - jeśli k i n nie mają wspólnych dzielników (co by było świadectwem złożoności) policz:
 $X = k^{(n-1)/2} \pmod{n}$, $Y = Js(n,k)$ (symbol Jacobiego),
jeśli $X \neq Y$ to k jest świadectwem złożoności liczby n .
 - dla tego testu nie ma ‘złych’ liczb złożonych.

Podsumowanie

- ❑ Przestrzeń probabilistyczna składa się z punktów z których każdy reprezentuje wynik jakiegoś doświadczenia. Każdy punkt x związany jest z nieujemną liczbą zwaną prawdopodobieństwem punktu x . Suma prawdopodobieństw wszystkich punktów składających się na przestrzeń probabilistyczna wynosi **1**.
- ❑ Zdarzenie jest podzbiorem punktów z przestrzeni probabilistycznej. Prawdopodobieństwo zdarzenia jest sumą prawdopodobieństw należących do niego punktów. Prawdopodobieństwo każdego zdarzenia mieści się w przedziale od **0** do **1**.

Podsumowanie

- Reguła sum określa, że prawdopodobieństwo tego, że zajdzie jedno z dwóch zdarzeń **E** lub **F** jest większe lub równe większemu z prawdopodobieństw obu zdarzeń, ale nie większa niż suma tych prawdopodobieństw.
- Reguła iloczynów określa, że prawdopodobieństwo tego, że wynikiem pewnego doświadczenia będą dwa zdarzenia **E** i **F**, jest nie większe niż mniejsze z prawdopodobieństw obu zdarzeń.
- Wykonując algorytm **Monte Carlo** stałą liczbę razy, możemy rozwiązać problem, dochodząc do wniosku, że jeśli żadne z tych powtórzeń nie doprowadziło nas do odpowiedzi „prawda”, to odpowiedzią jest „fałsz”.