Matematyka stosowana

Systemy decyzyjne

Hung Son Nguyen
son@mimuw.edu.pl
http://www.mimuw.edu.pl/~son

Uniwersytet Warszawski, 2011

Streszczenie. Przegląd metod klasyfikacji i wspomagania podejmowania decyzji na podstawie niepełnych i niepewnych informacji. Przedstawiane metody pochodzą z różnych dziedzin, takich jak uczenie maszynowe (ang. machine learning), statystyka, teoria zbiorów rozmytych (ang. fuzzy sets), teoria zbiorów przybliżonych (ang. rough sets). Przewidziane są zajęcia praktyczne z systemami wspomagania decyzji oraz zadania projektowe polegające na stworzeniu własnych systemów decyzyjnych.

Wersja internetowa wykładu:
http://mst.mimuw.edu.pl/lecture.php?lecture=syd

(może zawierać dodatkowe materiały)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego.

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Skład w systemie LaTeX, z wykorzystaniem m.in. pakietów beamer oraz listings. Szablony podręcznika i prezentacji: Piotr Krzyżanowski; koncept: Robert Dąbrowski.

Spis treści

1. Wprowadzenie do systemów decyzyjnych

1.1. Wprowadzenie do systemów decyzyjnych

1.1.1. Elementy systemów decyzyjnych

Wprowadzenie do teorii uczenia się

Kto się uczy?
Ograniczymy się do programów komputerowych zwanych "*algorytmami uczącymi się*".

Czego się uczy?
— **pojęć**: – np. odróżnienie "krzeseł" od innych mebli.
— **nieznanych urządzeń** – np. używanie VCR
— **nieznanych środowisk** – np. nowe miasto
— **procesów** – np. pieczenie ciasta
— **rodzin podobnych wzorców** – np. rozp. mowy, twarzy lub pisma.
— **funkcji**: (np. funkcje boolowskie)

Wymagania
skuteczność, efektywność, ...

Każdy "uczeń" powinien mieć zdolność uogólnienia, t.j. zdolność rozpoznawania różnych obiektów tego samego pojęcia.

Np. jeśli uczymy się funkcji, to ważne jest aby "algorytm uczenia się" nie ograniczał się do jednej konkretnej funkcji. Żądamy aby "modele uczenia" działały skutecznie na klasach funkcji.

Uczeń może pozyskać informacje o dziedzinie poprzez:

1. **Przykłady:** Uczeń dostaje pozytywne i/lub negatywne przykłady. Przykłady mogą być zdobywane w sposób:
   a) losowy: według pewnego znanego lub nieznanego rozkładu;
   b) arbitralny;
   c) złośliwy: (np. przez kontrolera, który chciałby wykryć sytuację, kiedy algorytm zachowuje się najgorzej);
   d) specjalny przez życzliwego nauczyciela: (np., starającego ułatwiać proces uczenia się)
2. **Zapytania:** uczeń zdobywa informacje o dziedzinie przez zadawanie nauczycielowi zapytań.
3. **Eksperymentowanie:** aktywne uczenie się.

— **Podejście indukcyjne:** wnioskowanie na podstawie skończonego zbioru obserwacji;
— Np. Pokazać, że dla każdego $n \in \mathbb{N}$

$$1^2 + 2^2 + ... + n^2 = \frac{n(n+1)(2n+1)}{6}$$

— Jakie prawa rządzą w podejściu uczenia indukcyjnego?

Szukamy teorii pozwalającej na oszacowanie

- Prawdopodobieństwa wyuczenia się pojęć;
- Liczby niezbędnych przykładów treningowych;
- Złożoności przestrzeni hipotez;
- Skuteczności aproksymacji;
- Jakość reprezentacji danych treningowych;

Skąd wiemy, czy uczeń się nauczył lub jak dobrze się nauczył?

- Miara jakości wsadowa (ang. off-line, batch) i miara interaktywna (ang. on-line, interactive).
- Jakość opisu vs. jakość predykcji
- Skuteczność: obliczona na podstawie błędu klasyfikacji, dokładności opisu ...
- Efektywność uczenia: wymagana jest wielomianowa złożoność obliczeniowa.

- Załóżmy, że chcemy nauczyć się pojęcia "człowieka o średniej budowie ciała". Dane – czyli osoby – są reprezentowane przez punkty $(wzrost(cm), waga(Kg))$ i są etykietowane przez $+$ dla pozytywnych przykładów i $-$ dla negatywnych.
- Dodatkowa wiedza: szukane pojęcie można wyrazić za pomocą PROSTOKĄTA
- Na przykład dany jest etykietowany zbiór:
  $((84, 184), +), ((70, 170), +), ((75, 163), -), ((80, 180), +), ((81, 195), -), ((63, 191), -), ((77, 187), -), ((68, 168), +)$
- Znajdź etykietę $((79, 183, ?)$

Roważany problem możemy zdefiniować problem następująco:

- **Cel:** Znaleźć w $\Re^2$ prostokąt R o bokach równoległych do osi.
- **Wejście:** Zbiór zawierający przykłady w postaci punktów $((x, y), +/-)$. Punkty z tego zbioru zostały wygenerowane losowo.
- **Wyjście:** Hipotetyczny prostokąt R' będący "'dobrą aproksymacją" R.
- **Dodatkowe wymagania:** Algorytm powinien być efektywny (ze wzgledu na złożoność obliczeniową) i powinien używać do uczenia jak najmniejszej liczby przykładów .

Przy ustalonych zbiorach pojęć $\mathbb{C}$ (dotyczących obiektów ze zbioru X - skończonego lub nie) oraz hipotez $\mathbb{H}$ rozważamy nastepujacy problem

- **Dane:**
  - skończona próbka D obiektów $x_1, ..., x_m \in X$ wraz z wartościami pewnej funkcji c ze zbioru $\mathbb{C}$ na tych obiektach;
- **Szukane:**
  - hipoteza $h \in \mathbb{H}$ będąca dobrą aproksymacją pojęcia c.
- **Żądania:**
  - dobra jakość aproksymacji
  - szybki czas działania.

- **Uczenie półosi (lub dyskretyzacji)**:

$$X = \Re; \qquad \mathbb{C} = \mathbb{H} = \{[\lambda, \infty) : \alpha \in \Re\}$$

- **Uczenie hiperpłaszczyzny**:

$$X = \Re^n; \qquad \mathbb{H} = \{f_{w_0,w_1,...,w_n} : \Re^n \to \{0, 1\}|\}$$

gdzie $f_{w_0,...,w_n}(x_1, ..., x_n) = sgn(w_0 + w_1x_1 + ... + w_nx_n)$.

- **Uczenie jednomianów Boolowskich**:

$$X = \{0, 1\}^n; \qquad c : \{0, 1\}^n \to \{0, 1\};$$

$\mathbb{H} = M_n =$ zbiór jednomianów Boolowskich o n zmiennych.

Niech
— X – zbiór wszystkich obiektów.
— $\Omega = (X, \mu)$ – przestrzeń probabilistyczna określona na X.
Błąd hipotezy $h \in \mathbb{H}$ względem pojęcia c (funkcji docelowej):

$$er_\Omega(h, c) = er^c_\Omega(h) = \mu\{x \in X | h(x) \neq c(x)\}$$

Pytanie: Dane jest pojęcie c, hipoteza h i zbiór przykładów D. Jak oszacować rzeczywisty błąd hipotezy h na podstawie jej błędu er^c_D na zbiorze D?
Odp.: Jeśli przykłady z D są wybrane zgodnie z miarą prawdopodobieństwa μ *niezależnie od tej hipotezy i niezależnie od siebie nawzajem* oraz $|D| \geqslant 30$, to
— najbardziej prawdopodobną wartością $er_\Omega(c, h)$ jest er^c_D,
— z prawdopodobieństwem $(1 - \varepsilon)$

$$|er^c_\Omega - er^c_D| \leqslant s_{\frac{\varepsilon}{2}} \sqrt{\frac{er^c_D(1 - er^c_D)}{|D|}}$$

Systemy informacyjne i tablice decyzyjne

— Teoria zbiorów przybliżonych została wprowadzona w latach 80-tych przez prof. Zdzisława Pawlaka.
— Głównym celem jest dostarczanie narzędzi dla problemu aproksymacji pojęć (zbiorów).
— Zastosowania w systemach decyzyjnych:
  — Redukcja danych, selekcja ważnych atrybutów
  — Generowanie reguł decyzyjnych
  — Odkrywanie wzorców z danych: szablony, reguły asocjacyjne
  — Odkrywanie zależności w danych

Przykład

Pacjent	Wiek	Płeć	Chol.	ECG	Rytm serca	Chory?
p_1	53	M	203	hyp	155	Tak
p_2	60	M	185	hyp	155	Tak
p_3	40	M	199	norm	178	Nie
p_4	46	K	243	norm	144	Nie
p_5	62	F	294	norm	162	Nie
p_6	43	M	177	hyp	120	Tak
p_7	76	K	197	abnorm	116	Nie
p_8	62	M	267	norm	99	Tak
p_9	57	M	274	norm	88	Tak
p_{10}	72	M	200	abnorm	100	Nie

Tablica decyzyjna

Jest to struktura $\mathbb{S} = (U, A \cup \{dec\})$, gdzie
— U jest zbiorem obiektów:

$$U = \{u_1, ..., u_n\};$$

— A jest zbiorem atrybutów postaci

$$a_j : U \to V_j;$$

— dec jest specjalnym atrybutem zwanym decyzją

$$dec : U \to \{1, ..., d\}.$$

Tablica decyzyjna powstaje ze zwykłych tablic danych poprzez sprecyzowanie:

— **Atrybutów (nazwanych warunkowymi)**: cechy, których wartości na obiektach są dostępne, np. pomiary, parametry, dane osobowe, ...

— **Decyzji (atrybut decyzyjny):**, t.j. cecha "ukryta" związana z pewną znaną częściowo wiedzą o pewnym pojęciu:
  — Decyzja jest znana tylko dla obiektów z (treningowej) tablicy decyzyjnej;
  — Jest podana przez eksperta (np. lekarza) lub na podstawie późniejszych obserwacji (np. ocena giełdy);
  — Chcemy podać metodę jej wyznaczania dla dowolnych obiektów na podstawie wartości atrybutów warunkowych na tych obiektach.

Przedstawiona tablica decyzyjna zawiera:

— 8 obiektów będących opisami pacjentów
— 3 atrybuty: Headache Muscle pain, Temp.
— Decyzję stwierdzącą czy pacjent jest przeziębiony czy też nie. lub nie

U	Ból głowy	Ból mięśni	Temp.	Grypa
p1	Tak	Tak	N	Nie
p2	Tak	Tak	H	Tak
p3	Tak	Tak	VH	Tak
p4	Nie	Tak	N	Nie
p5	Nie	Nie	H	Nie
p6	Nie	Tak	VH	Tak
p7	Nie	Tak	H	Tak
p8	Nie	Nie	VH	Nie

Dane są obiekty $x, y \in U$ i zbiór atrybutów $B \subset A$, mówimy, że

— x, y są **rozróżnialne przez** B wtw, gdy istnieje $a \in B$ taki, że $a(x) \neq a(y)$;
— x, y są **nierozróżnialne przez** B, jeśli one są identyczne na B, tzn. $a(x) = a(y)$ dla każdego $a \in B$;
— $[x]_B$ = zbiór obiektów nierozróżnialnych z x przez B.

— Dla każdych obiektów x, y:
  — albo $[x]_B = [y]_B$;
  — albo $[x]_B \cap [y]_B = \emptyset$.
— Relacja

$$x \; IND_B \; y := x, y \text{ są nierozróżnialne przez } B$$

jest relacją równoważności.

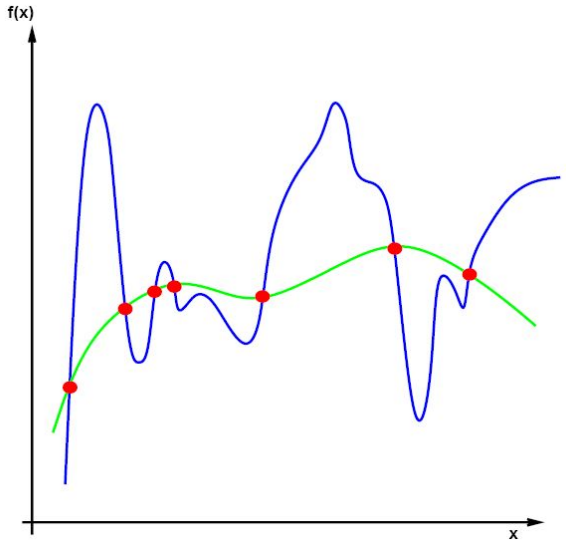
— Każdy zbiór atrybutów $B \subset A$ wyznacza podział zbioru obiektów na klasy nierozróżnialności.

Dla $B = \{Blgowy, Blmini\}$

— obiekty $p1, p2, p3$ są nierozróżnialne;
— są 3 klasy nierozróżnialności relacji IND_B:
  — $[p1]_B = \{p1, p2, p3\}$
  — $[p4]_B = \{p4, p6, p7\}$
  — $[p5]_B = \{p5, p8\}$

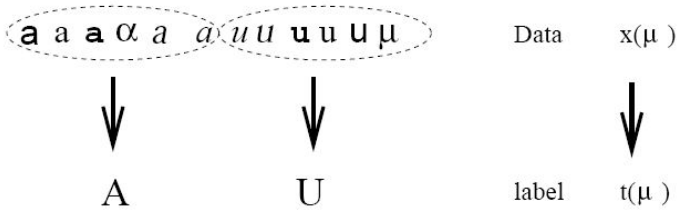
U	Ból głowy	Ból mięśni	Temp.	Grypa
p1	Tak	Tak	N	Nie
p2	Tak	Tak	H	Tak
p3	Tak	Tak	VH	Tak
p4	Nie	Tak	N	Nie
p5	Nie	Nie	H	Nie
p6	Nie	Tak	VH	Tak
p7	Nie	Tak	H	Tak
p8	Nie	Nie	VH	Nie

Aproksymacja pojęć



Aproksymacja funkcji

— Sztuczna sieć neuronowa;
— Twierdzenie Kolmogorowa;
— Modele sieci.



Aproksymacja pojęć

— Uczenie indukcyjne;
— COLT;
— Metody uczenia się.

Wnioskowanie aproksymacyjne

— Wnioskowanie rozmyte;
— Wnioskowanie Boolowskie, teoria zbiorów przybliżonych;
— Inne: wnioskowanie Bayesowskie, sieci przekonań, ...

— Klasyfikatory (algorytmy klasyfikujące) i metody oceny klasyfikatorów
— Metody rozumowania Boolowskiego
— Teoria zbiorów przybliżonych
— Reguły decyzyjne, drzewo decyzyjne i lasy decyzyjne
— Klasyfikatory Bayesowskie
— Sieci neuronowe
— COLT: Obliczeniowa Teoria Uczenia się
— Metody przygotowywania danych
— SVM: Maszyna wektorów podpierających
— Metody wzmacniania klasyfikatorów (ang. Boosting)

***Nie ma nic za darmo* czyli "Non Free Lunch Theorem"**

— Znaleźć optimum nieznanej funkcji $f : S \to W$ ($f \in \mathcal{F}$), gdzie S, W są skończonymi zbiorami.
— Działanie algorytmu przeszukiwania $\mathcal{A}$ dla funkcji f jest identyfikowane z wektorem:

$$V_{\mathcal{A}}(f,t) = \langle (s_1, f(s_1)), (s_2, f(s_2)), ..., (s_t, f(s_t)) \rangle$$

— Ocena algorytmu: $M : \{V_{\mathcal{A}}(f,t) | \mathcal{A}, f, t\} \to \mathbb{R}$. Np.

$$M(V_{\mathcal{A}}(f,t)) = \min_{i \in \{1,..,t\}} \{i | f(s_i) = f_{\max}\}$$

— **Warunek NFL:** Dla dowolnej funkcji M, i dla dowolnych algorytmów $\mathcal{A}, \mathcal{A}'$

$$\sum_{f \in \mathcal{F}} M(V_{\mathcal{A}}(f, |S|)) = \sum_{f \in \mathcal{F}} M(V_{\mathcal{A}'}(f, |S|))$$

— $\mathcal{F}$ **jest zamknięta względem permutacji:** dla dowolnej funkcji $f \in \mathcal{F}$ i dowolnej permutacji $\sigma \in Perm(S)$ mamy $\sigma f \in \mathcal{F}$

Twierdzenie o NFL
— Zachodzi równoważność

$$NFL \Leftrightarrow \mathcal{F} \text{ jest zamknięta względem permutacji.}$$

— Prawdopodobieństwo wylosowania niepustej klasy funkcji zamkniętej wzg. permutacji wynosi:

$$\frac{2^{\binom{|S|+|W|-1}{|S|}} - 1}{2^{|S|^{|W|}} - 1}$$

— Algorytm $\mathfrak{L}$ dobrze się uczy pojęcia c jeśli er^c_Ω jest mały.
— Niech $\mathbb{P}(X) = \{c : X \to \{0,1\}\}$.
Czy można stwierdzić wiedzieć, że algorytm $\mathfrak{L}_1$ wyuczy się wszystkich pojęć z $\mathbb{P}(X)$ lepiej niż algorytm $\mathfrak{L}_2$?
— "No Free Lunch theorem" (Wolpert, Schaffer) w wersji problemów uczenia się głosi, że:
  — Żaden algorytm nie może być najlepszy w wyuczeniu wszystkich pojęć.
  — Każdy algorytm jest najlepszy dla takiej samej liczby pojęć
  — Ale interesuje nas tylko pewna klasa problemów czyli klasa pojęć $\mathbb{C} \subset \mathbb{P}(X)$
  — Wniosek: Należy znaleźć odpowiedni algorytm do każdego problemu.

1.1.2. Sprawy organizacyjne

— Obecność: 20%
— Projekt: 40%
— Egzamin: 40%

2. Problem klasyfikacji i klasyfikatory

2.1. Problem klasyfikacji i klasyfikatory

2.1.1. Wprowadzenie

— **Założenie:** Dany jest skończony zbiór obiektów $\mathbf{T} = \{x_1, ..., x_n\}$. Ten zbiór nazywamy *zbiorem treningowym*
  — Każdy obiekt (rekord) jest opisany wektorem informacyjnym, składającym się z wartości pochodzących z dziedziny pewnych atrybutów warunkowych.
  — Wyróżniony jest jeden atrybut, zwany też atrybutem decyzyjnym

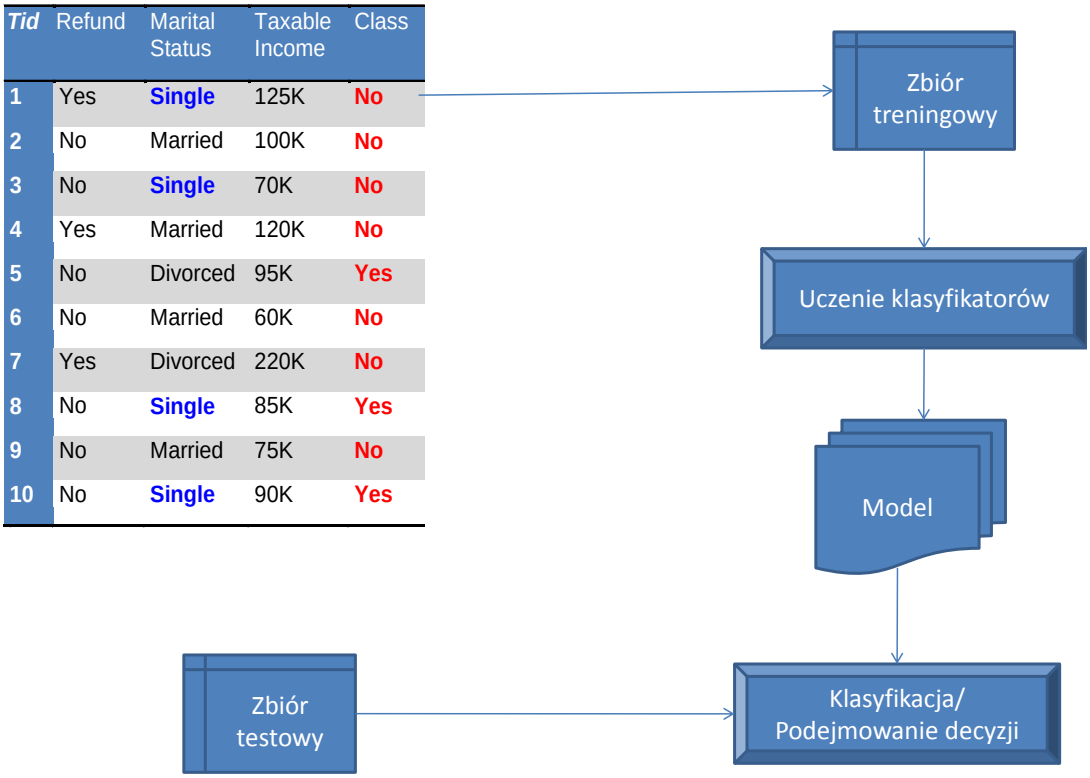
$$x_i \longrightarrow (\langle x_{i1}, \ldots, x_{ik} \rangle, d_i)$$

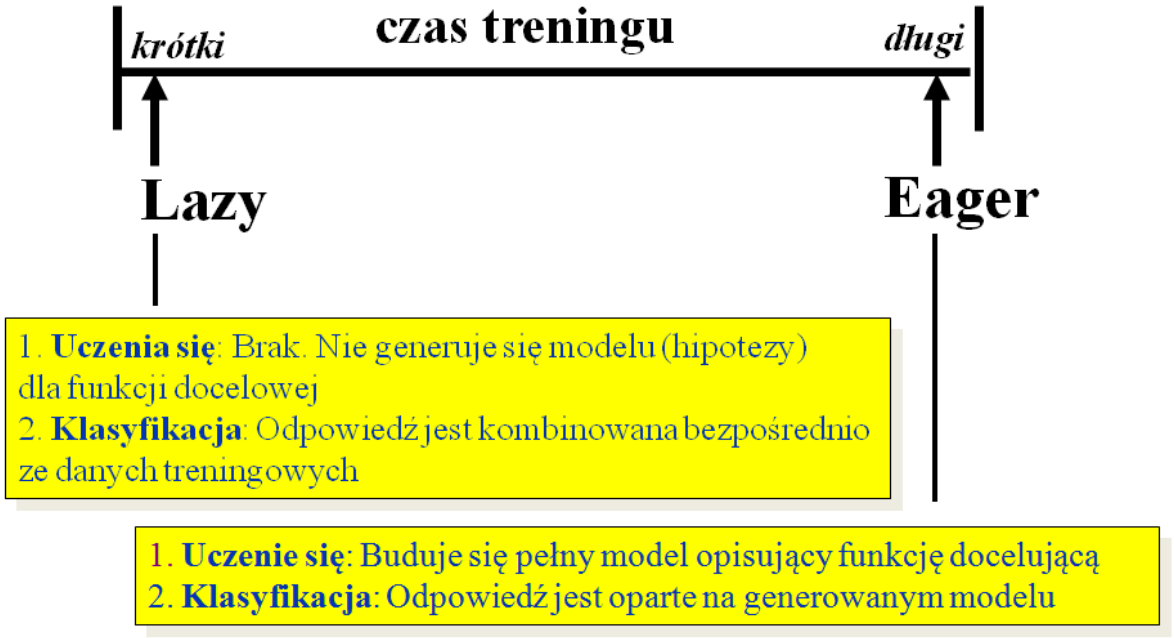
— **Cel:** wyznaczyć klasę decyzyjną, do której należy nowy nieznany dotąd obiekt.
— **Jak?** Znaleźć zależność (najlepiej funkcyjną) między atrybutem decyzyjnym a atrybutami warunkowymi.

1. Tworzenie modelu: opisywanie klas decyzyjnych
   — Klasa decyzyjna = zbiór obiektów mających taką samą wartość na atrybucie decyzyjnym
   — **Klasyfikator:** algorytm określenia klasy decyzyjnej obiektów za pomocą ich wartości na atrybutach warunkowych.
   — Klasyfikatory mogą być opisane za pomocą formuł logicznych, drzew decyzyjnych lub formuł matematycznych.
2. Korzystanie z modelu do przypisanie nowym nieznanym obiektom ich klasy decyzyjne.

Problem: Jak oceniać model?

— Wiele problemów decyzyjnych można opisać jako problem klasyfikacji
— DSS (Decision Support System) jest systemem komputerowym dostarczającym narzędzia do rozwiązywania problemów klasyfikacji.
  Wejście: Zbiór treningowy
  Wyjście: Klasyfikator(y)
— Wymagania:
  — Szybkość podejmowania decyzji, skalowalność
  — Skuteczność
  — Racjonalność
  — Możliwość współpracy z ekspertem: doradztwo, adaptacja, negocjacja, adopcja wiedzy eksperskiej, ...





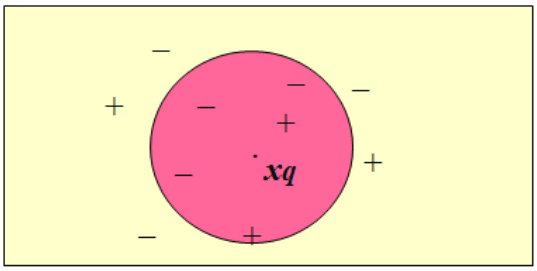
2.1.2. Przegląd metod klasyfikacji

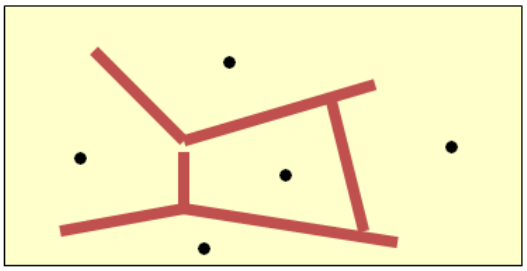
Metody oparte o przykłady

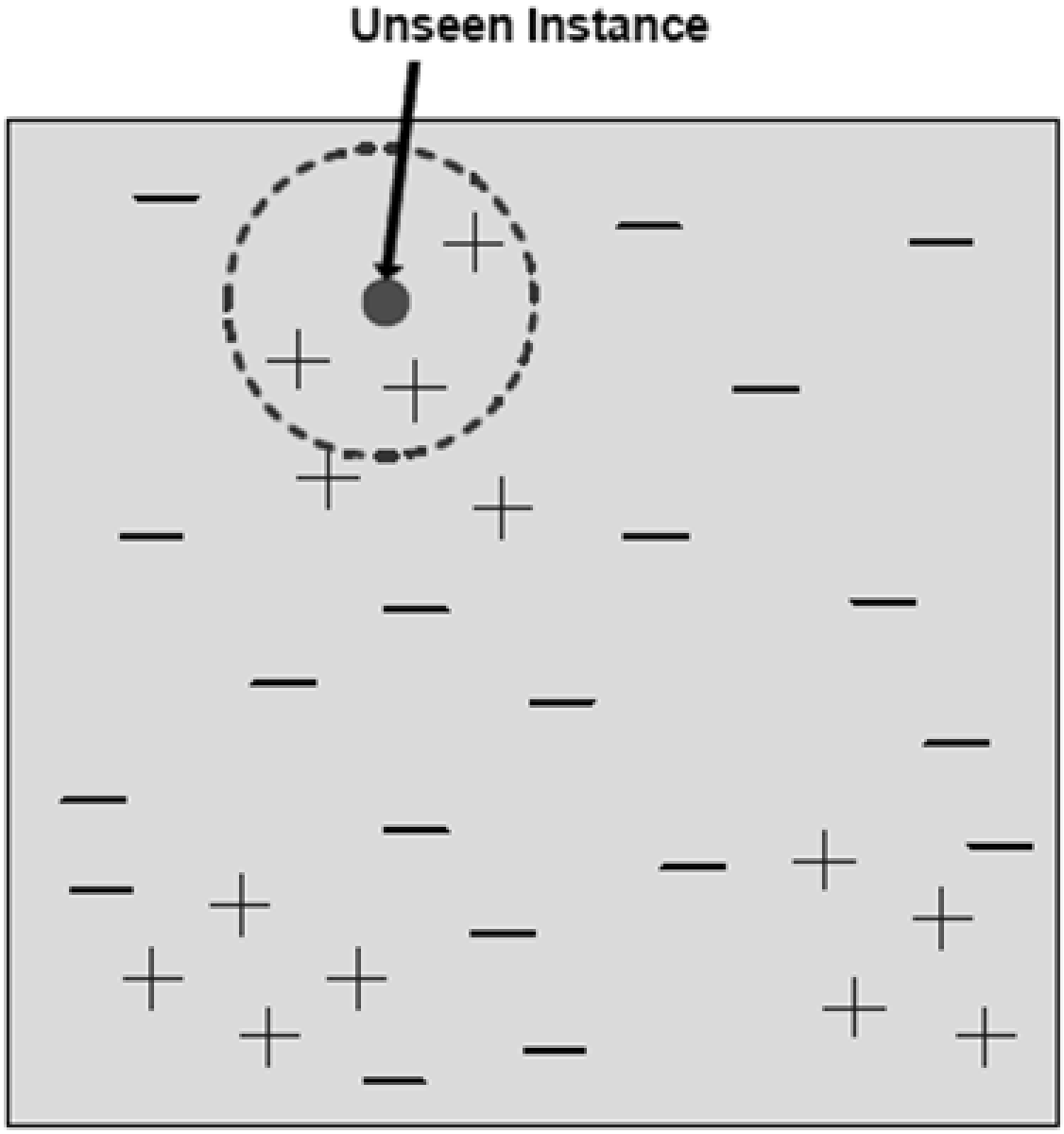
— Są to metody leniwej klasyfikacji, które opóźniają proces przetwarzania danych aż do momentu kiedy pojawi się nowy obiekt do klasyfikowania
— Typowe metody:

- ◇ k-nearest neighbor approach
  - ∗ Przykłady treningowe są przedstawione jako punkty w metrycznej przestrzeni.
- ◇ Locally weighted regression
  - ∗ Konstruuje lokalne aproksymacje pojęć
- ◇ Case-based reasoning
  - ∗ Korzysta z symbolicznych reprezentacji i metod wnioskowania w oparciu o bazę wiedzy

algorytm najbliższych sąsiadów
Input: Tablica decyzyjna D, nowy obiekt x_q
Output: Klasa decyzyjna, do której należy x_q (czyli $Dec(x_q)$)
Krok 1: Szukaj w zbiorze D, k najbliżej położonych obiektów $R(x_q) = \{x_1, x_2, ..., x_k\}$
Krok 2: Wyznacz klasę dla x_q na podstawie $Dec(x_1), Dec(x_2), ..., Dec(x_k)$







— Najczęściej wykorzystany dla danych z atrybutami numerycznymi
— Wymagania:
  — Zbiór treningowy
  — Funkcja odległości między obiektami
  — Wartość parametru k, liczba rozpatrywanych sąsiadów
— Podczas klasyfikacji:
  — Wyznaczanie k najbliższych sąsiadów
  — Wyznaczenie klasy decyzyjnej nowego obiektu na podstawie klas decyzyjnych najbliższych sąsiadów (np. przez głosowanie).
— Jest to przykład metody leniwej, gdyż
  — Nie buduje jawnego modelu wiedzy
  — Proces klasyfikacji może być czasochłonny
— Jeśli k jest za mała, klasyfikator będzie wrażliwa na drobne szumy w danych
— Jeśli k jest zbyt duża:

  - Wysoka złożoność obliczeniowa
  - Otoczenia mogą zawierać obiekty z innych klas
- Algorytm k-NN dla ciągłej decyzji
  - Wystarczy obliczyć średnią z decyzji najbliższych sąsiadów

- Problem skalowania atrybutów
  - Np. opis człowieka:
    - (Wzrost [m], Waga [kg], Klasa)
    - Wzrost odchyla się od 1.5 m do 1.85 m
    - Waga może mieć wartość od 45 kg do 120 kg

    Odległość euklidesowa jest bardziej wrażliwa na różnicę wag niż różnicę wzrostu.
- Przekleństwo wymiarów
- Może produkować wyniki niezgodne z intuicją (np. klasyfikacji dokumentów)
  - Rozwiązanie: Normalizacja
- Sąsiedzi ważeni względem odległośc
  - Wpływ sąsiada x_i na obiekt testowy x_q jest ważony przez $w_i = \frac{1}{d^2(x_q, x_i)}$
  - Bliżsi sąsiedzi mają większy wpływ

Wnioskowanie Bayesowskie

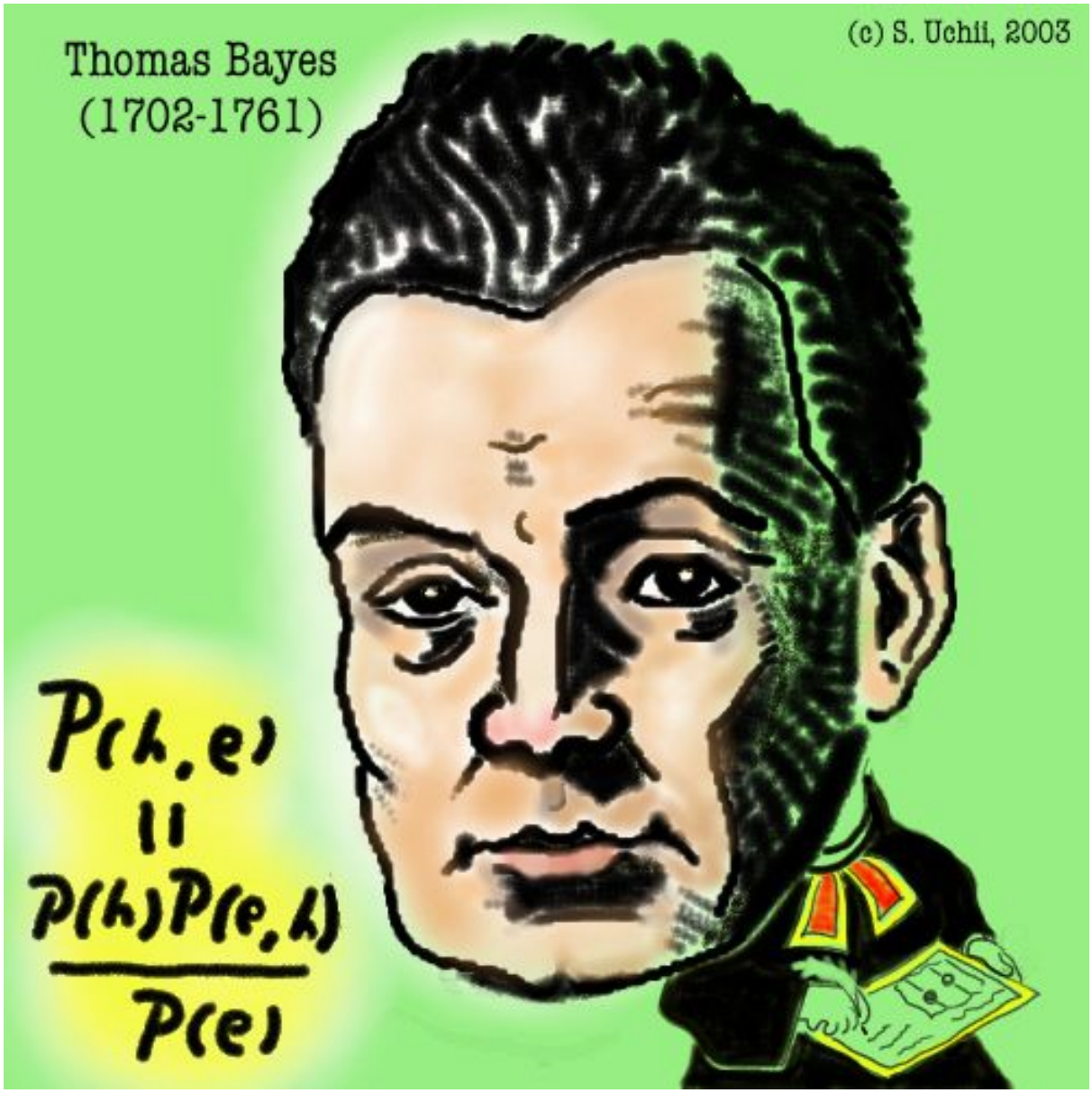
- Dany jest zbiór treningowy D, prawdopodobieństwo posteriori hipotezy h – $P(h|D)$ – można liczyć wzorem Bayesa.

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (maximum posteriori) hypothesis

$$h_{MAP} = \arg\max_{h \in H} P(h|D) = \arg\max_{h \in H} P(D|h)P(h)$$

- Trudności: ta metoda wymaga znajomości wielu rozkładów prawdopodobieństw $\implies$ wysoki koszt obliczeniowy



— Klasyfikacja obiektu opisanego przez $\mathbf{x}$

$$P(dec = c|\mathbf{x}) = \frac{P(\mathbf{x}|dec = c) \cdot P(dec = c)}{P(\mathbf{x})}$$

— $P(\mathbf{x})$ jest wspólna dla wszystkich hipotez;
— $P(dec = c)$ – częstość występowania klasy c;
— Znaleźć c tak, że $P(dec = c|\mathbf{x})$ było maksymalne, czyli, aby $P(\mathbf{x}|dec = c) \cdot P(dec = c)$ było maksymalne;
— Problem: obliczenie $P(\mathbf{x}|dec = c)$ jest czasochłonne!

— Naiwne założenie: atrybuty są warunkowo niezależne!
Wówczas

$$P(x_1, \ldots, x_k|C) = P(x_1|C) \cdot \ldots \cdot P(x_k|C)$$

— Czyli

$$P(dec = c|\mathbf{x}) \approx P(dec = c) \prod_{i=1}^{k} P(x_i|dec = c) \cdot$$

To założenie znacznie obniża złożoność obliczeniowy.

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	**P(sunny|n) = 3/5**
P(overcast|p) = 4/9	**P(overcast|n) = 0**
P(rain|p) = 3/9	**P(rain|n) = 2/5**
temperature	
P(hot|p) = 2/9	**P(hot|n) = 2/5**
P(mild|p) = 4/9	**P(mild|n) = 2/5**
P(cool|p) = 3/9	**P(cool|n) = 1/5**
humidity	
P(high|p) = 3/9	**P(high|n) = 4/5**
P(normal|p) = 6/9	**P(normal|n) = 2/5**
windy	
P(true|p) = 3/9	**P(true|n) = 3/5**
P(false|p) = 6/9	**P(false|n) = 2/5**

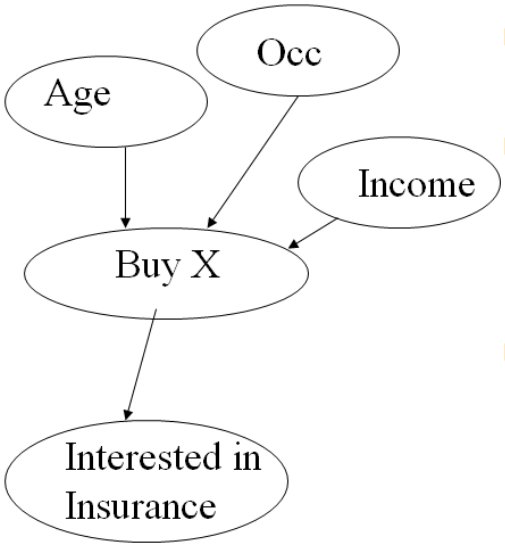
— Nowy obiekt $\mathbf{x} = \langle rain, hot, high, false \rangle$

$$P(X|p)\Delta P(p) = P(rain|p)\Delta P(hot|p)\Delta P(high|p)\Delta P(false|p)\Delta P(p) = 0.010582$$
$$P(X|n)\Delta P(n) = P(rain|n)\Delta P(hot|n)\Delta P(high|n)\Delta P(false|n)\Delta P(n) = 0.018286$$

— X jest klasyfikowany do klasy n (don't play)

— Założenie o niezależności:
  — ... powoduje, że obliczenia staje się możliwe
  — ... optymalny klasyfikator o ile ono jest prawdziwe
  — ... ale warunek bardzo rzadko spełniony w praktyce (atrybuty są często korelowane).
— Próby pokonania te ograniczenia:
  — **Sieci Bayesowskie**, które łączą metody wnioskowania Bayesowskiego z zależnościami między atrybutami
  — **Drzewa decyzyjne**, które przeprowadzają dedukcyjne kroki na pojedyńczych atrybutach, począwszy od najważniejszego

— Sieci Bayesowskie dopuszczają "warunkową niezależność" podzbiorów zmiennych.
— Jest to graficzny model zależności przyczynowo-skutkowych
— Naive Bayes jest szczególnym przypadkiem sieci Bayesowskiej (jakim?)
— Problemy związane uczeniem sieci Bayesowskich:
  — Prypadek najłatwiejszy: zarówno struktura sieci, jak i wszystkie zmienne są znane.
  — Znana jest struktura, ale brakuje rozkładów zmiennych.
  — Nawet struktura sieci nie jest z góry zadana.



- Zmienne Age, Occupation i Income mają wpływ na to czy klient kupuje produkt
- Jeśli klient kupuje produkt to jego zainteresowanie kupnem ubezpieczenia jest niezależne od Age, Occupation, Income.
- Zatem

  P(Age, Occ, Inc, Buy, Ins) =
  = P(Age)P(Occ)P(Inc)
  P(Buy|Age,Occ,Inc)P(Int|Buy)

Ogólna formuła

$$P(x_1, \ldots, x_k|C) = \prod_{i=1}^{k} P(x_i|parent(x_i), C)$$

3. Metody oceny klasyfikatorów

3.1. Metody oceny klasyfikatorów

3.1.1. Skuteczność predykcji

Błąd klasyfikacji

$$Błąd\ klasyfikacji = \frac{liczba\ błędów}{liczba\ obiektów\ testowych}$$

gdzie:
— *Sukces:* gdy obiekt jest prawidłowo klasyfikowany
— *Błąd:* gdy obiekt jest źle klasyfikowany
— *Błąd klasyfikacji* (lub odsetka błędów podczas klasyfikacji) powinien być wyznaczony na losowych i nieznanych danych.

— Są dwa rodzaje błędu:

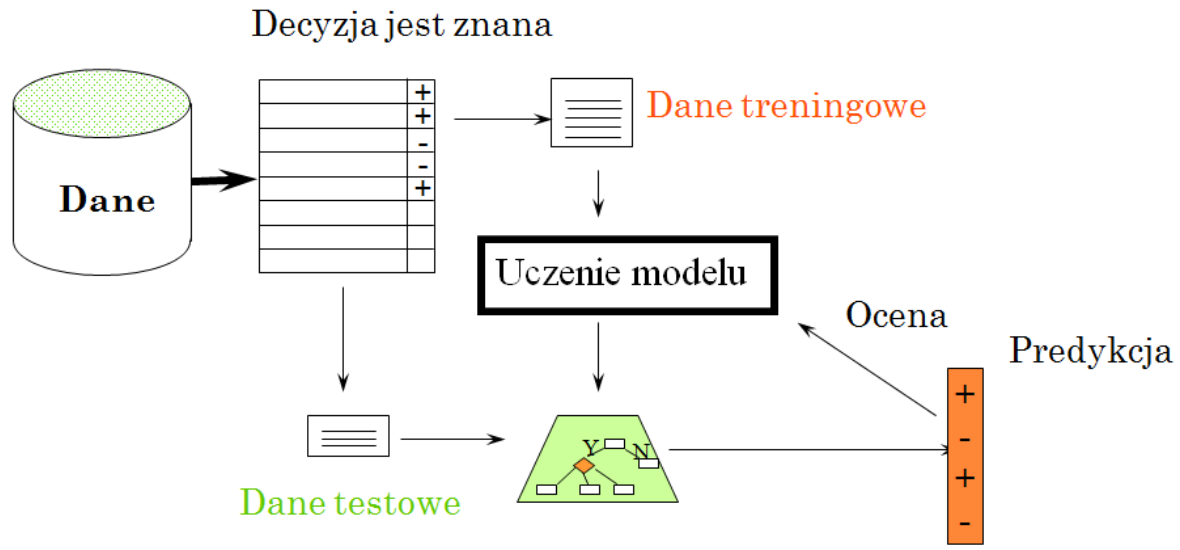
Predicted Value (predicted by the test)		Actual Value (as confirmed by experiment): positives	Actual Value (as confirmed by experiment): negatives
	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

— W "systemach uczących się": minimalizujemy FP+FN lub miarę skuteczności (ang. Accuracy) (ACC):

$$ACC = (TP + TN)/(TP + FP + TN + FN)$$

— W marketingu: maksymalizujemy TP.

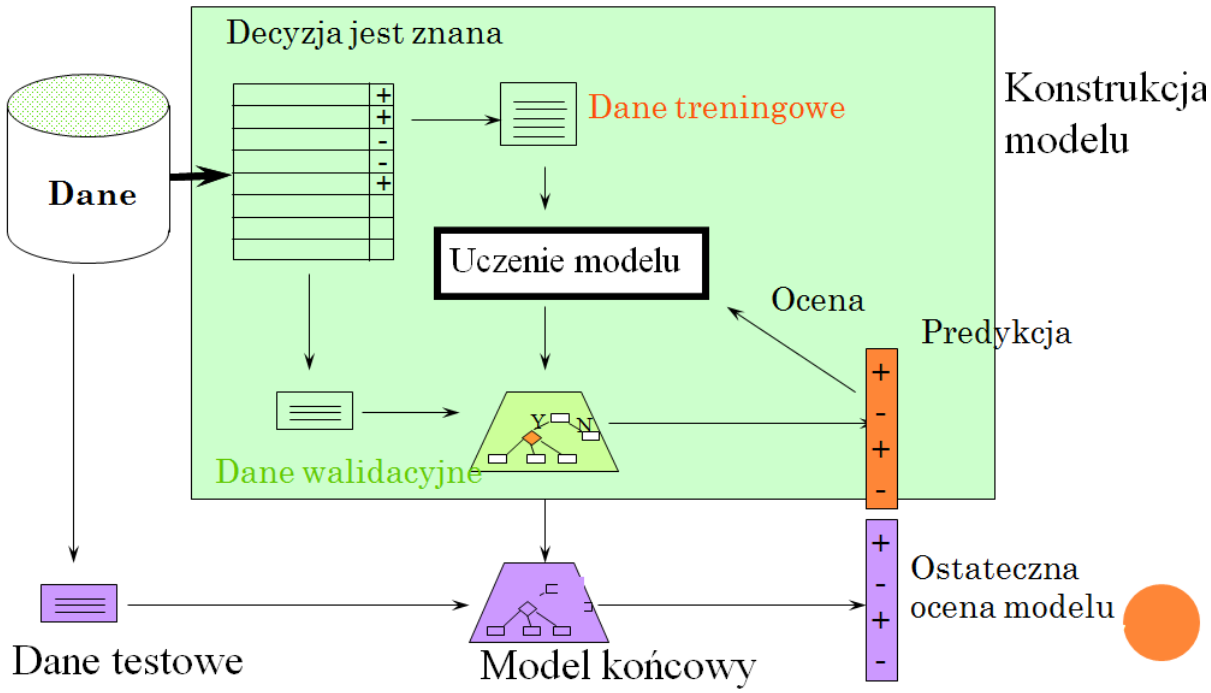
— Podział zbioru danych na część treningową i testową;
— Uczenie lub poszukiwanie modelu
— Ocena klasyfikatora



— Niektóre metody uczenia działają w dwóch etapach:
— Etap 1: Buduje strukturę
— Etap 2: Optymalizuje parametry

Uwaga:
Nie używaj danych testowych do budowy klasyfikatorów!

— Właściwa procedura powinna zawierać 3 zbiory: treningowe, walidacyjne i testowe
— Dane walidacyjne używane są do optymalizacji parametrów



3.1.2. Przedział ufności miar ocen

— Przykład: $S = 750$ sukcesów w $N = 1000$ próbach

  - Estymowana skuteczność: 75%
  - Jak bliska jest ta estymacja do prawdziwej skuteczności p?
  - Odp: z 80% pewności możemy twierdzić, że $p \in [73.2, 76.7]$
- Inny przykład: S=75 i N=100
  - Estymowana skuteczność: 75%;
  - $p \in [69.1, 80.1]$ z 80% pewności.
- Rozpatrujemy rozkład Bernoulliego: $p, p(1-p)$
- Oczekiwany odsetek sukcesu w N próbach: $f = S/N$
- Wartość oczekiwana i wariancja dla f:

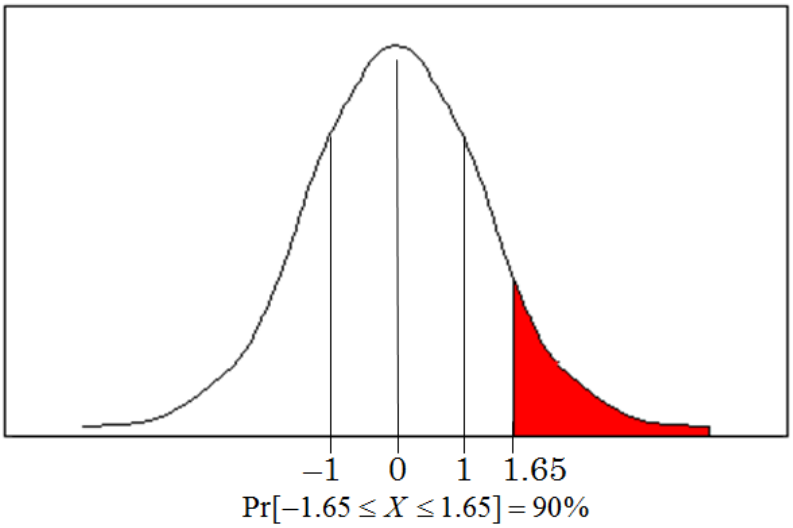
$$p, p(1-p)/N$$

- Dla dużych N, zm.l. f ma rozkład zbliżony do rozkładu normalnego;
- $[-z \leqslant X \leqslant z]$ nazywamy *przedziałem ufności na poziomie c% dla zm.l. X o zerowej wartości oczekiwanej* wtw:

$$P[-z \leqslant X \leqslant z] = c$$

- Dla rozkładu symetrycznego mamy:

$$P[-z \leqslant X \leqslant z] = 1 - 2P[X \geqslant z]$$



Pr[X ≥ z]	z
0.1%	3.09
0.5%	2.58
1%	2.33
5%	1.65
10%	1.28
20%	0.84
40%	0.25

Dla zmiennej X o rozkładzie N(0,1)

- Wartość oczekiwana i wariancję dla f: $p, p(1-p)/N$
- Normalizacja zm. f:

$$\frac{f-p}{\sqrt{p(1-p)/N}}$$

- Mamy równanie na p:

$$Pr\left(-z \leqslant \frac{f-p}{\sqrt{p(1-p)/N}} \leqslant z\right) = c$$

- Rozwiązanie dla p: $p \in [p_1, p_2]$, gdzie

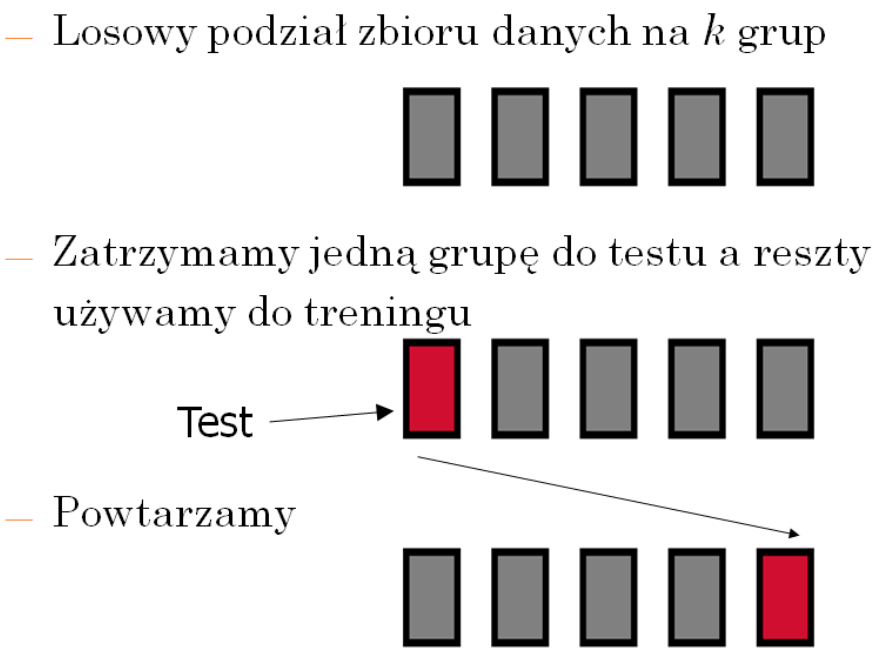
$$p_{1,2} = \frac{f + \frac{z^2}{2N} \pm z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

3.1.3. Metody walidacji danych

Ogólna zasada:
— Im większy zbiór treningowy, tym lepszy jest klasyfikator
— Im większy jest zbiór testowy, tym lepiej można aproksymować błąd klasyfikacji.

Praktyczna rada:
Kiedy proces oceniania się zakończy, wszystkie dane mogą być wykorzystywane do skonstruowania ostatecznego klasyfikatora

— Walidacja krzyżowa nie pozwala na wielokrotne testowanie tego samego obiektu
  — Krok 1: Podział zbioru danych na k równych podzbiorów
  — Krok 2: Testowanie każdego podzbioru używając pozostałych jako zbiór treningowy
— To się nazywa k-CV = k-fold cross-validation
— Zwykle obiekty są przetasowane przed dokonaniem podziału.
— Błędy wszystkich iteracji są uśrednione, aby otrzymać błąd globalny.



— Standardowa metoda ocena klasyfikatorów: 10-krotna walidacja krzyżowa
— Liczba 10 została wyznaczona w wyniku wielu doświadczeń.
— Walidacja pozwala na zmniejszenie długości przedziału ufności
— Jeszcze lepsza metoda oszacowania parametrów:
  Walidacja z powtórzeniami!

— **Leave-one-out:** przypadek szczególny walidacji krzyżowej
  Liczba grup = liczba przykładów
  — Dla n obiektów budujemy klasyfikator n razy
  — Najlepiej ocenia klasyfikatora
  — Obliczeniowo kosztowna metoda (wyjątek: kNN)
— **Bootstraping:** próbkuje ze zwracaniem, żeby stworzyć różne zbiory treningowe i testowe
  — Próbkuje ze zwracaniem n razy
  — Wybrane obiekty tworzą zbiór treningowy
  — Reszta – zbiór testowy.

3.1.4. Krzywy Lift i ROC

— Miara "sensitivity" lub "true positive rate" (TPR)

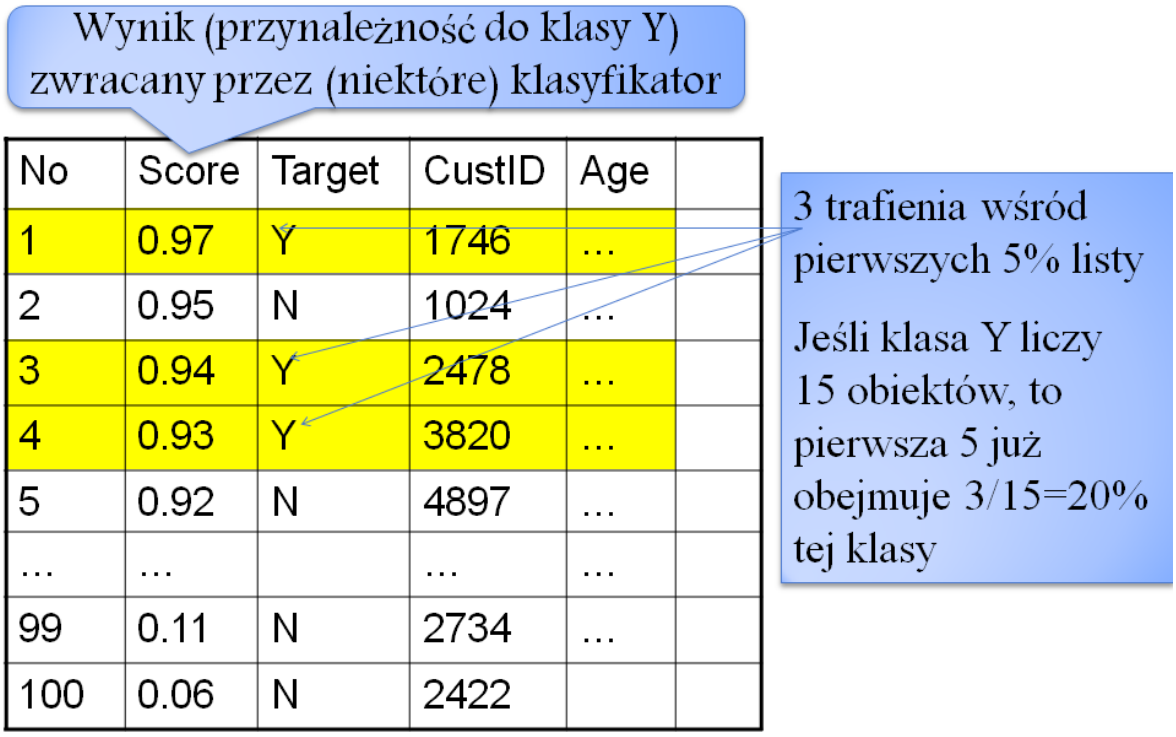
$$TPR = TP/(TP + FN)$$

czasem nazywa się też "recall" lub "hit rate".
— Specificity (SPC) lub True Negative Rate $SPC = TN/(FP + TN)$
— false positive rate (FPR): $FPR = FP/(FP + TN) = 1 - FPC$
— positive predictive value (PPV) lub precision: $PPV = TP/(TP + FP)$
— negative predictive value (NPV): $NPV = TN/(TN + FN)$
— false discovery rate (FDR): $FDR = FP/(FP + TP)$
— Matthew's correlation coefficient (MCC)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN)(TP + FP)(FN + TN)(FP + TN)}}$$

— F1 score: $F1 = 2TP/[(TP + FN) + (TP + FP)]$ lub

$$\frac{1}{F1} = \frac{\frac{1}{recall} + \frac{1}{precision}}{2}$$



No	Score	Target	CustID	Age	
1	0.97	Y	1746	…	
2	0.95	N	1024	…	
3	0.94	Y	2478	…	
4	0.93	Y	3820	…	
5	0.92	N	4897	…	
…	…		…	…	
99	0.11	N	2734	…	
100	0.06	N	2422		

Funkcje
— p - parametr określający początek listy rankingowej
— CPH - (ang. Cumulative Percentage Hit)
$CPH(p)$ = część klasy docelowej znajdująca się wsród p% pierwszych obiektów z listy rankingowej.
— zysk (ang. lift): $Lift(p) = CPH(p)/p$
— Trafienie lub true positive rate: $TPR(p) = TP/(TP + FN)$
— Odsetek fałszywych alarmów $FPR(p) = FP/(FP + TN)$

Wyróżnione krzywy

— Gain chart:
  $Ox : p$
  $Oy : CPH(p)$
— Lift chart:
  $Ox : p$
  $Oy : Lift(p)$
— ROC (receiver operating characteristic):
  $Ox : FPR(p)$
  $Oy : TPR(p)$

4. Zbiory przybliżone

4.1. Zbiory przybliżone

4.1.1. Wprowadzenie do teorii zbiorów przybliżonych

Systemy informacyjne i tablice decyzyjne

— Teoria zbiorów przybliżonych została wprowadzona w latach 80-tych przez prof. Zdzisława Pawlaka.
— Głównym celem jest dostarczanie narzędzi dla problemu aproksymacji pojęć (zbiorów).
— Zastosowania w systemach decyzyjnych:
  — Redukcja danych, selekcja ważnych atrybutów;
  — Generowanie reguł decyzyjnych;
  — Odkrywanie wzorców z danych: szablony, reguły asocjacyjne;
  — Odkrywanie zależności w danych.

Definicja
Jest to para $\mathbb{S} = (U, A)$, gdzie
— U – skończony niepusty zbiór obiektów (ang. cases, states, patients, observations ...);
— A – skończony, niepusty zbiór atrybutów. Każdy $a \in A$ odpowiada pewnej funkcji $a : U \to V_a$ zwanej *wartościowaniem*, gdzie V_a jest nazwana dziedziną atrybutu a.

Dla $B \subseteq A$, definiujemy
— B-*sygnatura* obiektu $x \in U$ (ang. B-*information vector*) jako

$$inf_B(x) = \{(a, a(x)) : a \in B\}$$

— Zbiór sygnatur względem B o obiektach z U (ang. B-*information set*):

$$INF(\mathbb{S}) = \{inf_B(x) : x \in U\}$$

Tablica decyzyjna powstaje ze zwykłych tablic danych poprzez sprecyzowanie:
— **Atrybutów (nazwanych warunkowymi)**: cechy, których wartości na obiektach są dostępne, np. pomiary, parametry, dane osobowe, ...
— **Decyzji (atrybut decyzyjny):**, t.j. cecha "ukryta" związana z pewną znaną częściowo wiedzą o pewnym pojęciu:
  — Decyzja jest znana tylko dla obiektów z (treningowej) tablicy decyzyjnej;
  — Jest podana przez eksperta (np. lekarza) lub na podstawie późniejszych obserwacji (np. ocena giełdy);
  — Chcemy podać metodę jej wyznaczania dla dowolnych obiektów na podstawie wartości atrybutów warunkowych na tych obiektach.

Przedstawiona tablica decyzyjna zawiera:
— 8 obiektów będących opisami pacjentów
— 3 atrybuty: Headache Muscle pain, Temp.
— Decyzję stwierdzącą czy pacjent jest przeziębiony czy też nie. lub nie

U	Ból głowy	Ból mięśni	Temp.	Grypa
p1	Tak	Tak	N	Nie
p2	Tak	Tak	H	Tak
p3	Tak	Tak	VH	Tak
p4	Nie	Tak	N	Nie
p5	Nie	Nie	H	Nie
p6	Nie	Tak	VH	Tak
p7	Nie	Tak	H	Tak
p8	Nie	Nie	VH	Nie

Dane są obiekty $x, y \in U$ i zbiór atrybutów $B \subset A$, mówimy, że

— x, y są **rozróżnialne przez** B wtw, gdy istnieje $a \in B$ taki, że $a(x) \neq a(y)$;
— x, y są **nierozróżnialne przez** B, jeśli one są identyczne na B, tzn. $a(x) = a(y)$ dla każdego $a \in B$;
— $[x]_B$ = zbiór obiektów nierozróżnialnych z x przez B.

— Dla każdych obiektów x, y:
  — albo $[x]_B = [y]_B$;
  — albo $[x]_B \cap [y]_B = \emptyset$.
— Relacja

$$x \; IND_B \; y := x, y \text{ są nierozróżnialne przez } B$$

jest relacją równoważności.
— Każdy zbiór atrybutów $B \subset A$ wyznacza podział zbioru obiektów na klasy nierozróżnialności.

Dla $B = \{Blgowy, Blmini\}$

— obiekty $p1, p2, p3$ są nierozróżnialne;
— są 3 klasy nierozróżnialności relacji IND_B:
  — $[p1]_B = \{p1, p2, p3\}$
  — $[p4]_B = \{p4, p6, p7\}$
  — $[p5]_B = \{p5, p8\}$

U	Ból głowy	Ból mięśni	Temp.	Grypa
p1	Tak	Tak	N	Nie
p2	Tak	Tak	H	Tak
p3	Tak	Tak	VH	Tak
p4	Nie	Tak	N	Nie
p5	Nie	Nie	H	Nie
p6	Nie	Tak	VH	Tak
p7	Nie	Tak	H	Tak
p8	Nie	Nie	VH	Nie

— Każdy zbiór obiektów X (np. klasa decyzyjna, pojęcie) może być opisany za pomocą atrybutów ze zbioru B *dokładnie* lub *w przybliżeniu*
  — *dokładny opis:* jeśli X jest sumą pewnych klas nierozróznialności definiowanych przez B (ZBIORY DOKŁADNE)
  — *przybliżony opis:* w przeciwnym przypadku (ZBIORY PRZYBLIŻONE)
— W obu przypadkach X może być opisany przez 2 "dokładne zbiory" zwane dolną i górną aproksymacją zbioru X

$$\underline{B}(X) = \{x : [x]_B \subset X\} \quad \overline{B}(X) = \{x : [x]_B \cap X \neq \emptyset\}$$

— Obszar brzegowy (ang. B-boundary region) pojęcia X zawiera obiekty, dla których nie możemy jednoznacznie zdecydować czy należą one do X czy nie na podstawie atrybutów z B

— Obszar wewnętrzny (ang. B-inside region of X) zawiera obiekty, które możemy pewnie klasyfikować jako elementy pojęcia X mając do dyspozycji atrybuty z B.
— Zbiór jest przybliżony (ang. rough set) jeśli obszar brzegowy jest niepusty, w przeciwnym przypadku zbiór jest nazwany dokładny (ang. crisp set).

Niech $B = \{a_1, a_2\}$

$$
\begin{aligned}
IND(B) = \{&\{1,2\}, && (sunny, hot)\\
&\{3,13\}, && (overcast, hot)\\
&\{4,10,14\}, && (rainy, mild)\\
&\{5,6\}, && (rainy, cool)\\
&\{8,11\}, && (sunny, mild)\\
&\{7\},\{9\},\{12\}\}
\end{aligned}
$$

Chcemy aproksymować pojęcie definiowane przez klasę decyzyjną (play=no)

$$X = CLASS_{no} = \{1,2,6,8,14\}$$

Aproksymacje pojęcia X:

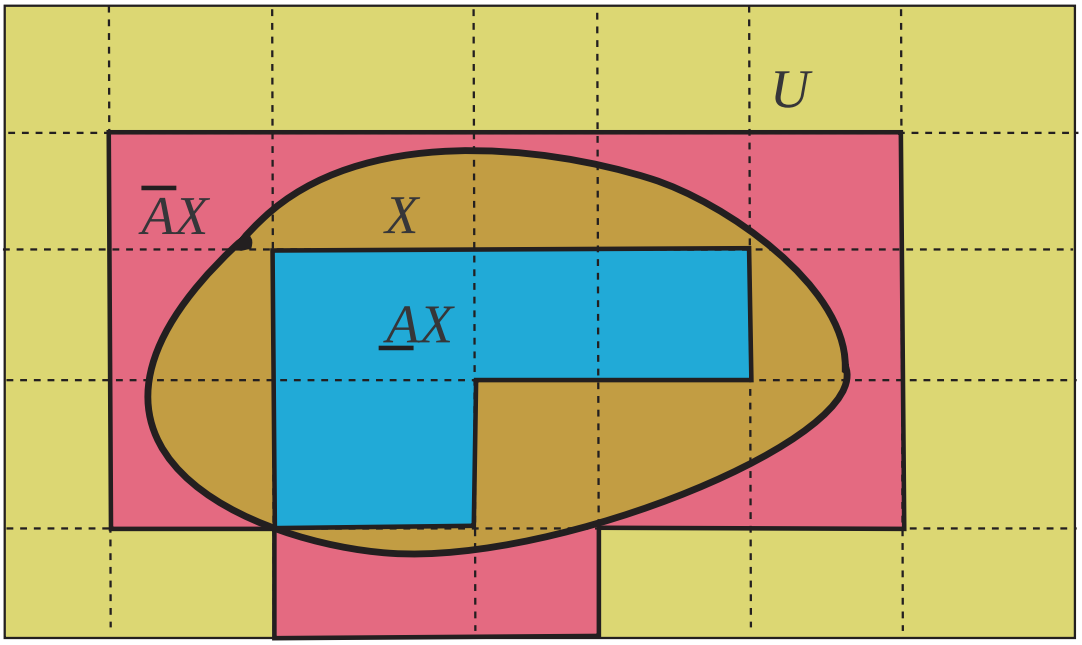
$$
\begin{aligned}
\mathbf{L}_B(X) &= \{1,2\}\\
\mathbf{U}_B(X) &= \{1,2,5,6,8,11,4,10,14\}
\end{aligned}
$$

Reguła pewna:
If $B(x) = (sunny, hot)$ **then** $d(x) = no$

$\mathbb{A}$	a_1	a_2	a_3	a_4	d
ID	outlook	temp.	hum.	windy	play
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Reguły przybliżone:

If $B(x) = (rainy, cool)$ **then** $d(x) = no$
If $B(x) = (rainy, mild)$ **then** $d(x) = no$
If $B(x) = (sunny, mild)$ **then** $d(x) = no$



Jakość aproksymacji (ang. accuracy of approximation)

$$\alpha_B(X) = \frac{|\underline{B}(X)|}{|\overline{B}(X)|}$$

— $0 \leqslant \alpha_B(X) \leqslant 1$
— Jeśli $\alpha_B(X) = 1$, to zbiór X jest dokładnie definiowany przez B;
— Jeśli $\alpha_B(X) < 1$, to zbiór X jest aproksymacyjnie definiowany przez B;

Redukty

— W systemach decyzyjnych, nie wszystkie atrybuty są potrzebne do procesie podejmowania decyzji;
— Chcemy wybrać pewne podzbiory atrybutów niezbędnych do tego celu;
— Redukty to minimalne podzbiory atrybutów zachowujących charakterystykę całego zbioru atrybutów.
— W teorii zbiorów przybliżonych, istnieją co najmniej 2 pojęcia reduktów: informacyjne i decyzyjne.

Definicja reduktu informacyjnego

Zbiór atrybutów $B \subset A$ nazywamy reduktem tablicy decyzyjnej A wtw, gdy
— B zachowuje rozróżnialność zbioru A:
t.j. dla dowolnych obiektów $x, y \in U$,
jeśli x, y są rozróżnialne przez A, to są również rozróżnialne przez B
— B jest niezredukowalny:
tzn. żaden właściwy podzbiór B nie zachowuje rozróżnialności zbioru A (t.j., B jest minimalny pod względem rozróżnialności)

Definicja reduktu decyzyjnego

Zbiór atrybutów $B \subset A$ nazywamy reduktem tablicy A wtw, gdy
— B zachowuje rozróżnialność zbioru A względem decyzji dec:
t.j. dla dowolnych obiektów $x, y \in U$,
jeśli $dec(x) \neq dec(y)$ i x, y są rozróżnialne przez A, to są również rozróżnialne przez B

— B jest niezredukowalny:
tzn. żaden właściwy podzbiór B nie zachowuje rozróżnialności zbioru A (B jest minimalny pod względem rozróżnialności)

— $RED(\mathbb{S})$ = zbiór wszystkich reduktów tablicy decyzyjnej $\mathbb{S}$;
— Jeśli $\mathbb{S} = (U, A \cup \{dec\})$ i $|A| = n$ to

$$|RED(\mathbb{S})| \leqslant \binom{n}{n/2}$$

— **rdzeń:** zbiór atrybutów będących we wszystkich reduktach

$$K = \bigcap_{B \in RED(\mathbb{S})} B.$$

— Znaleźć rdzeń danej tablicy decyzyjnej;
— Znaleźć jakiś redukt;
— Znaleźć krótkie redukty;
— Znaleźć długie redukty.

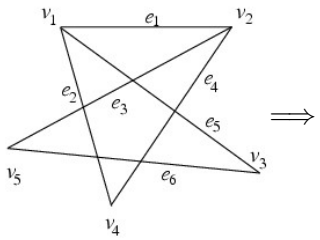
4.1.2. Złożoność problemu szukania reduktów

Problem najkrótszego reduktu
Dane: Tablica decyzyjna $\mathbb{S} = (U, A \cup \{dec\})$;
Szukane: "najkrótszy redukt tablicy decyzyjnej $\mathbb{S}$", tzn. taki redukt decyzyjny $B \in \mathbf{RED}(\mathbb{S}, dec)$, że

$$\forall_{X \in \mathbf{RED}(\mathbb{S}, dec)} |B| \leqslant |X|$$

Twierdzenie
Problem szukania najkrótszego reduktu jest NP-zupełny.

— **Ogólnie musimy pokazać, że** jakiś znany NP-zupełny problem jest "łatwiejszy" od problemu najkrótszego reduktu;
— Wybierzmy problem minimalnego pokrycia wierzchołkami:
"Dany jest graf $G = (V, E)$. Znaleźć minimalny zbiór wierzchołków $X \subset V$ o takiej własności, że każda krawędź z E posiada co najmniej jeden z końców w X."
— Wielomianowa transformacja:



$\Longrightarrow$

$\mathbb{S}(\mathbf{G})$	a_{v_1}	a_{v_2}	a_{v_3}	a_{v_4}	a_{v_5}	a^*
x^*	0	0	0	0	0	0
u_{e_1}	1	1	0	0	0	1
u_{e_2}	1	0	0	1	0	1
u_{e_3}	0	1	0	0	1	1
u_{e_4}	0	1	0	1	0	1
u_{e_5}	1	0	1	0	0	1
u_{e_6}	0	0	1	0	1	1

— Można też pokazać, że problem najkrótszego reduktu jest "łatwiejszy" niż znany problem "Minimal Hitting Set" (minimalny zbiór przecinający?)
  — **Dane:** Zbiór S i rodzina jego podzbiorów $\{S_1, S_2, ..., S_n\}$. Zbiór $X \subset S$ nazywamy zbiorem przecinającym, jeśli X ma niepuste przecięcie z każdym ze zbiorów $S_1, S_2, ..., S_n$.
  — **Szukane:** minimalny zbiór przecinający.
— Te problemy są obliczeniowo równoważne!
Czyli każdy algorytm dla jednego problemu można przekształcić w czasie wielomianowym na odpowiedni algorytmu dla drugiego problemu, który posiada te same cechy złożonościowe.

5. Wnioskowanie Boolowskie w obliczaniu reduktów i reguł decyzyjnych

5.1. Wnioskowanie Boolowskie w obliczaniu reduktów i reguł decyzyjnych

5.1.1. Metody wnioskowań Boolowskich w szukaniu reduktów

Algebry Boola

Algebra Boola Jest to struktura algebraiczna

$$\mathcal{B} = (B, +, \cdot, 0, 1)$$

spełniająca następujące aksjomaty

- **Przemienność:**
  $(a + b) = (b + a)$ oraz $(a \cdot b) = (b \cdot a)$
- **Rozdzielność:**
  $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$, oraz $a + (b \cdot c) = (a + b) \cdot (a + c)$
- **Elementy neutralne:**
  $a + 0 = a$ oraz $a \cdot 1 = a$
- **Istnienie elementu komplementarnego:**
  $a + \overline{a} = 1$ and $a \cdot \overline{a} = 0$

*) Czasem negacja jest dodana do sygnatury algebry Boola.

1. Algebra zbiorów: $\mathbb{P}(X) = (2^X, \cup, \cap, -, \emptyset, X)$;
2. Rachunek zdań
$$(\{zdania\ logiczne\}, \vee, \wedge, \neg, \bot, \top)$$
3. Binarna algebra Boola $\mathcal{B}_2 = (\{0, 1\}, +, \cdot, 0, 1)$ to jest najmniejsza, ale najważniejsza algebra Boola w zastosowaniu.

x	y	$x + y$	$x \cdot y$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

x	$\neg x$
0	1
1	0

Przykłady zastosowań:
— projektowanie układów scalonych;
— rachunek zdań.

Łączność (ang. Associative law): $(x + y) + z = x + (y + z)$ oraz $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
Idempotence: $x + x = x \quad oraz \quad x \cdot x = x (dual)$
Działania z 0 i 1: $x + 1 = 1 \quad oraz \quad x \cdot 0 = 0 (dual)$
Pochłanienie (ang. Absorption laws): $(y \cdot x) + x = x \quad oraz \quad (y + x) \cdot x = x (dual)$
Inwolocja (ang. Involution laws): $\overline{(\overline{x})} = x$
Prawa DeMorgana (ang. DeMorgan's laws):

$$\neg(x + y) = \neg x \cdot \neg y \quad oraz \quad \neg(x \cdot y) = \neg x + \neg y (dual)$$

Prawa konsensusu (ang. Consensus laws):

$$(x+y)\cdot(\overline{x}+z)\cdot(y+z) = (x+y)\cdot(\overline{x}+z) \text{ oraz}$$
$$(x\cdot y)+(\overline{x}\cdot z)+(y\cdot z) = (x\cdot b)+(\overline{x}\cdot z)$$

Zasada dualności: Każda algebraiczna równość pozostaje prawdziwa jeśli zamieniamy operatory + na $\cdot$, $\cdot$ na +, 0 na 1 oraz 1 na 0.

Funkcje Boolowskie i wnioskowanie Boolowskie

— Każde odwzorowanie $f : \{0,1\}^n \to \{0,1\}$ nazywamy **(zupełną) funkcją Boolowską**.
— Funkcje Boolowskie ≡ **formuły Boolowskie**:
  — Stałe, zmienne, operatory -, + oraz .
  — Literały, mintermy (jednomiany), maxtermy, ...
  — Kanoniczna postać dyzjunkcyjna (DNF), np. $f = xy\overline{z} + x\overline{y}z + \overline{x}yz + xyz$;
  — Kanoniczna postać koniunkcyjna (CNF), np. $f = (x+y+z)(\overline{x}+y)$
— Term $t = x_{i_1}...x_{i_m}\overline{x_{j_1}}...\overline{x_{j_k}}$ nazywamy **implikantem** funkcji f jeśli
$$\boxed{\forall_{\mathbf{a}\in\{0,1\}^n} \quad t(\mathbf{a}) = 1 \Rightarrow f(\mathbf{a}) = 1}$$
— **Implikant pierwszy:** jest to implikant, który przestaje nim być po usunięciu dowolnego literału.
— **Kanoniczna postać Blake'a**: każdą funkcję Boolowską można przedstawić jako sumę wszystkich jej imlikantów pierwszych:
$$f = t_1 + t_2 + ... + t_k$$

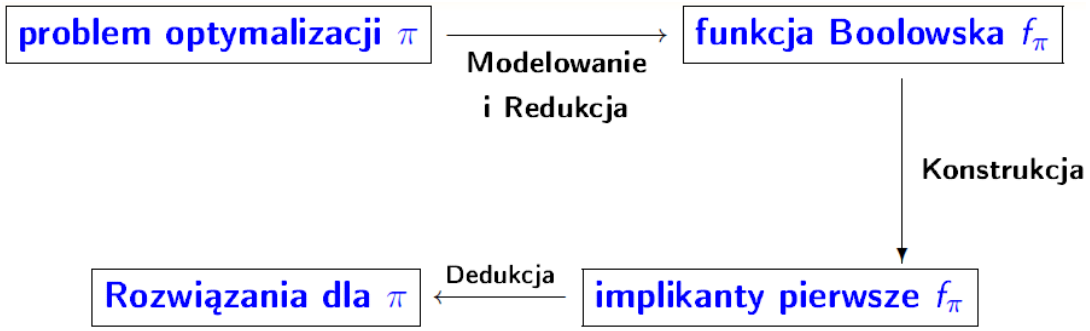
Wiele formuł reprezentuje tę samą funkcję;
$\phi_1 = xy\overline{z} + x\overline{y}z + \overline{x}yz + xyz$
$\phi_2 = (x+y+z)(\overline{x}+y+z)(x+\overline{y}+z)(x+y+\overline{z})$
$\phi_3 = xy + xz + yz$
  $xy\overline{z}$ jest implikantem
  xy jest implikantem pierwszym

x	y	z	f
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	1
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	1

— Niech $\prec$ oznacza relację częściowego porządku w $\{0,1\}^n$
— Funkcja f jest monotoniczna (niemalejąca) wtw, gdy dla każdych $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ jeśli $\mathbf{x} \prec \mathbf{y}$ to $f(\mathbf{x}) \leqslant f(\mathbf{y})$
— monotoniczne funkcje Boolowskie można zapisać bez użycia negacji.
— term $\boxed{f' = x_{i_1} \cdot x_{i_2} ... \cdot x_{i_k}}$ nazywamy implikantem pierwszym funkcji monotonicznej f jeśli
  — $f'(\mathbf{x}) \leqslant f(\mathbf{x})$ dla każdego wektora $\mathbf{x}$ (jest implikantem)
  — każda funkcja większa od f' nie jest implikantem
— Np. funkcja

$$f(x_1, x_2, x_3) = (x_1 + x_2)(x_2 + x3)$$

posiada 2 implikanty pierwsze: $f_1 = x_2$ i $f_2 = x_1 \wedge x_3$



1. **Modelowanie:** Kodowanie problemu za pomocą układu równań Boolowskich;
2. **Redukcja:** Sprowadzenie układu równań do pojedynczego równania postaci $f = 0$ lub $f = 1$
3. **Konstrukcja:** Znalezienie wszystkich implikantów pierwszych funkcji f (konstrukcja kanonicznej postaci Blake'a);
4. **Dedukcja:** Zastosowanie pewnej sekwencji wnioskowań transformujących implikanty pierwsze do rozwiązań problemu.

Problem:
A, B, C, D are considering going to a party. Social constrains:
— If A goes than B won't go and C will;
— If B and D go, then either A or C (but not both) will go
— If C goes and B does not, then D will go but A will not.

Problem modeling:

$$\begin{aligned} A \to \overline{B} \wedge C &\leftrightsquigarrow A(B + \overline{C}) &&= 0 \\ \ldots &\leftrightsquigarrow BD(AC + \overline{AC}) &&= 0 \\ \ldots &\leftrightsquigarrow \overline{B}C(A + \overline{D}) &&= 0 \end{aligned}$$

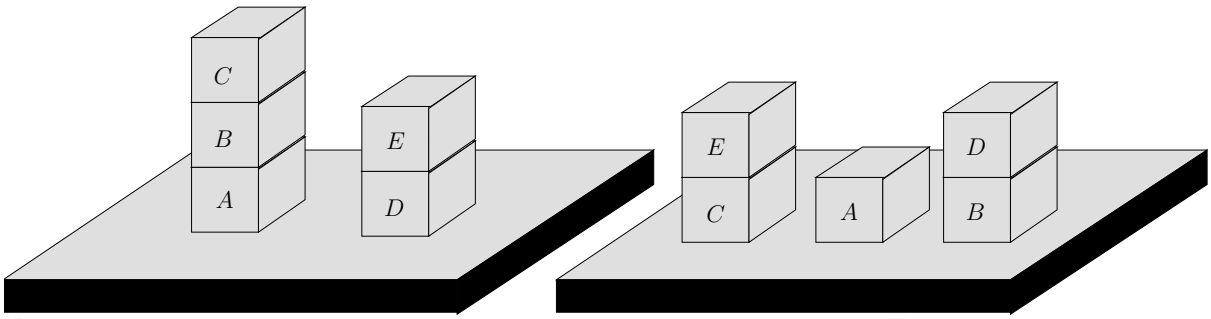
— **After reduction:**
$f = A(B + \overline{C}) + BD(AC + \overline{AC}) + \overline{B}C(A + \overline{D}) = 0$
— **Blake Canonical form:** $f = B\overline{C}D + \overline{B}C\overline{D} + A = 0$
— **Facts:**

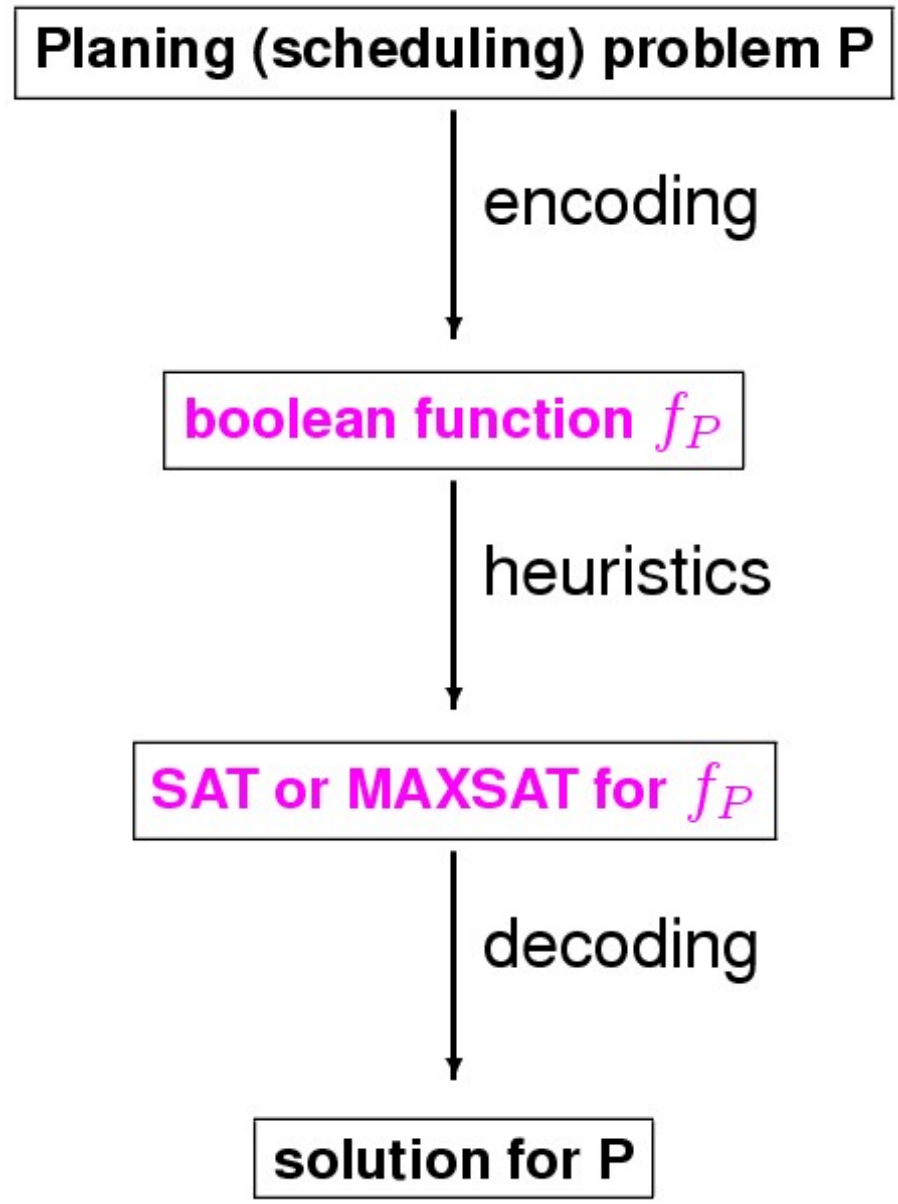
$$\begin{aligned} BD &\longrightarrow C \\ C &\longrightarrow B \vee D \\ A &\longrightarrow 0 \end{aligned}$$

— **Reasoning:** (theorem proving)
e.g., show that
"nobody can go alone."

— **Problem SAT**: czy równanie
$f(x_1, \ldots, x_n) = 1$
posiada rozwiązanie?
— SAT odgrywa ważną rolę w teorii złożoności (twierdzenie Cooka, dowodzenie NP-zupełności ...)
— Każdy SAT-solver może być używany do rozwiązywania problemu planowania.
— *Blocks world problem*: Po redukcji formuła boolowska nadal zawiera $O(tk^2)$ zmiennych i $O(tk^3)$ klauzuli (k, t - liczby bloków i kroków).

— Np. Dla $k = 15$, $t = 14$, mamy 3016 zmiennych i 50457 klausuli





1. Konstrukcja funkcji odróżnialności:
   — Zmienne boolowskie: atrybuty $a_1, ..., a_k$;
   — Klauzula:

$$\phi_{u,v} = \bigvee \{a \in A : a(u) \neq a(v)\}$$

— Funkcja rozróżnialności:

$$\mathcal{F}_{\mathbb{S}}(a_1, ..., a_k) = \bigwedge_{x,y \in U : dec(x) \neq dec(y)} \phi_{x,y}(a_1, ..., a_k)$$

2. przekształcenie funkcji rozróżnialności do postaci DNF
3. każdy implikant pierwszy odpowiada jednemu reduktowi;

Hurt.	Jakość obsługi	Jakość towaru	Obok autostrady?	Centrum?	decyzja
ID	a_1	a_2	a_3	a_4	dec
1	dobra	dobra	nie	nie	strata
2	dobra	dobra	nie	tak	strata
3	bdb	dobra	nie	nie	zysk
4	slaba	super	nie	nie	zysk
5	slaba	niska	tak	nie	zysk
6	slaba	niska	tak	tak	strata
7	bdb	niska	tak	tak	zysk
8	dobra	super	nie	nie	strata
9	dobra	niska	tak	nie	zysk
10	slaba	super	tak	nie	zysk
11	dobra	super	tak	tak	zysk
12	bdb	super	nie	tak	zysk
13	bdb	dobra	tak	nie	?
14	slaba	super	nie	tak	?

$\mathbb{M}$	1	2	6	8
3	a_1	a_1, a_4	a_1, a_2, a_3, a_4	a_1, a_2
4	a_1, a_2	a_1, a_2, a_4	a_2, a_3, a_4	a_1
5	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_4	a_1, a_2, a_3
7	a_1, a_2, a_3, a_4	a_1, a_2, a_3	a_1	a_1, a_2, a_3, a_4
9	a_2, a_3	a_2, a_3, a_4	a_1, a_4	a_2, a_3
10	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_2, a_4	a_1, a_3
11	a_2, a_3, a_4	a_2, a_3	a_1, a_2	a_3, a_4
12	a_1, a_2, a_4	a_1, a_2	a_1, a_2, a_3	a_1, a_4

$$\begin{aligned} f \quad = \quad & (\alpha_1)(\alpha_1 \vee \alpha_4)(\alpha_1 \vee \alpha_2)(\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_4) \\ & (\alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_3)(\alpha_4)(\alpha_2 \vee \alpha_3)(\alpha_2 \vee \alpha_4) \\ & (\alpha_1 \vee \alpha_3)(\alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_4) \end{aligned}$$

$$\begin{aligned} f = \quad & (\alpha_1)(\alpha_1 \vee \alpha_4)(\alpha_1 \vee \alpha_2)(\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_4) \\ & (\alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_3)(\alpha_4)(\alpha_2 \vee \alpha_3)(\alpha_2 \vee \alpha_4) \\ & (\alpha_1 \vee \alpha_3)(\alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_4) \end{aligned}$$

— usuwanie alternatyw regułą *pochłaniania* (t.j. $p \wedge (p \vee q) \equiv p$):

$$f = (\alpha_1)(\alpha_4)(\alpha_2 \vee \alpha_3)$$

— sprowadzanie funkcji f z postaci CNF (koniunkcja alternatyw) do postaci DNF (alternatywa koniunkcji)

$$f \quad = \quad \alpha_1\alpha_4\alpha_2 \vee \alpha_1\alpha_4\alpha_3$$

— Każdy składnik jest reduktem! Zatem mamy 2 redukty: $R_1 = \{A_1, A_2, A_4\}$ i $R_2 = \{A_1, A_3, A_4\}$

Heurystyka Johnsona

— Atrybut jest ważniejszy jeśli częściej występuje w klauzulach;
— **Selekcja:** W każdym kroku wybierzmy atrybut, który najczęściej występuje w funkcji rozróżnialności;
— **Usuwanie:** Usuwamy z tej funkcji te klauzule, które zawierają wybrany atrybut;
— Powtarzamy **Selekcja** i **Usuwanie** dopóty, póki funkcja rozróżnialności zawiera jeszcze jakąś klazulę.

Heurystyka Johnsona

1. Znaleźć atrybut, który występuje najczęściej w macierzy rozróżnialności;
2. Usuwnąć wszystkie pola zawierające wybrany atrybut;
3. Powtarzamy kroki 1 i 2 dopóki wszystkie pola macierzy są puste.

$\mathbb{M}$	1	2	6	8
3	a_1	a_1, a_4	a_1, a_2, a_3, a_4	a_1, a_2
4	a_1, a_2	a_1, a_2, a_4	a_2, a_3, a_4	a_1
5	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_4	a_1, a_2, a_3
7	a_1, a_2, a_3, a_4	a_1, a_2, a_3	a_1	a_1, a_2, a_3, a_4
9	a_2, a_3	a_2, a_3, a_4	a_1, a_4	a_2, a_3
10	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_2, a_4	a_1, a_3
11	a_2, a_3, a_4	a_2, a_3	a_1, a_2	a_3, a_4
12	a_1, a_2, a_4	a_1, a_2	a_1, a_2, a_3	a_1, a_4

— *W macierzy nierozróznialności z poprzedniego przykładu*
a_1 *- występuje 23 razy* a_2 *- występuje 23 razy*
a_3 *- występuje 18 razy* a_4 *- występuje 16 razy*
— *Jeśli wybieramy* a_1*, to po usunięciu pól zawierających* a_1 *zostało 9 niepustych pól macierzy nierozróżnialności. Wśród nich:*
a_2 *- występuje 7 razy* a_3 *- występuje 7 razy* a_4 *- występuje 6 razy*
— *Jeśli wybieramy tym razem* a_2 *to zostały 2 niepuste pola i wśród nich* a_4 *jest zdecydowanym faworytem.*
— *Możemy dostać w ten sposób redukt:* $R_1 = \{a_1, a_2, a_4\}$.

— Niech $X = \{(u, v) \in U^2 : dec(u) \neq dec(v)\}$;
— Niech $S_{a_i} = \{(u, v) \in X : a_i(u) \neq a_i(v)\}$;
— Niech $C^* \subset A$ będzie minimalnym reduktem, lecz $C = \{a_1, ..., a_k\}$ będzie reduktem znalezionym przez heurystykę Johnsona;
— Załóżmy, że HEURYSTYKA JOHNSONA musi płacić 1zł za każdym razem, gdy dodała a_i do C.
— Rozłóżmy ten koszt tym elementom, które zostały po raz pierwszy pokryte przez zbiór S_{a_i}
— Niech c_x = koszt ponoszony przez x. Jeśli x jest pokryty po raz pierwszy przez a_i, to

$$c_x = \frac{1}{|S_{a_i} - (S_{a_1} \cup ... \cup S_{a_{i-1}})|}$$

— HEURYSTYKA JOHNSONA ponosi łączny koszt $= |C|$. Mamy

$$|C| = \sum_{x \in X} c_x \leqslant \sum_{a \in C^*} \sum_{x \in S_a} c_x$$

— Gdybyśmy pokazali, że dla dowolnego altrybutu $a \in A$

$$\sum_{x \in S_a} c_x \leqslant H(|S_a|)$$

gdzie $H(n) = \sum_{i=1}^{n} \frac{1}{i}$, wówczas możemy oszacować dalej:

$$\begin{aligned} |C| &\leqslant \sum_{a \in C*} \sum_{x \in S_a} c_x \leqslant |C^*| \cdot H(\max\{|S_a : a \in A|\}) \\ &\leqslant |C^*| \cdot H(|X|) \leqslant |C^*| \cdot \ln(|X| + 1) \end{aligned}$$

Lemat:
Dla dowolnego altrybutu $a \in A$

$$\sum_{x \in S_a} c_x \leqslant H(|S_a|)$$

Dowód:
— Niech $u_i = |S_a - (S_{a_1} \cup ... \cup S_{a_{i-1}})|$, mamy:

$$u_0 = |S_a| \geqslant ... \geqslant u_{i-1} \geqslant u_i \geqslant ... \geqslant u_k = 0;$$

gdzie k jest najmniejszym indeksem t., że $S_a = S_{a_1} \cup ... \cup S_{a_k}$
— Zatem

$$\begin{aligned}\sum_{x \in S_a} c_x &= \sum_{i=0}^{k} (u_{i-1} - u_i) \cdot \frac{1}{|S_{a_i} - (S_{a_1} \cup ... \cup S_{a_{i-1}})|} \\ &\leqslant \sum_{i=0}^{k} (u_{i-1} - u_i) \cdot \frac{1}{u_{i-1}} \leqslant \sum_{i=0}^{k} (H(u_{i-1}) - H(u_i)) = H(|S_a|)\end{aligned}$$

Inne heurystyki

— ILP (Integer Linear Program)
— Symulowane wyżarzanie (simulated annealing)
— Algorytmy genetyczne
— SAT solver
— ???

Dana jest funkcja $f : \{0,1\}^n \to \{0,1\}$ zapisana w postaci CNF za pomocą zmiennych $x_1, ..., x_n$

1. Utwórz nowe zmienne $\{y_1, ..., y_{2n}\}$ odpowiadające literałom $x_1, \overline{x_1}, ..., x_n, \overline{x_n}$;
2. Dla każdej zmiennej x_i, utwórzmy nierówność $y_{2i-1} + y_{2i} \leqslant 1$
3. Zamień każdą klausulę $\omega_i = (l_{i_1} \vee ... \vee l_{i_{n_i}})$ na nierówność $y_{i_1} + ... + y_{i_{n_i}} \geqslant 1$;
4. Utwórzmy układ nierówności z poprzednich punktów:

$$\mathbf{Ay} \geqslant \mathbf{b}$$

5. Zagadnienie programowania liniowego liczb całkowitych (ILP) jest definiowane jako problem szukania

$$\min \sum_{i=1}^{n} y_i \quad \text{przy ograniczeniu: } \mathbf{Ay} \geqslant \mathbf{b}.$$

5.1.2. Systemy decyzyjne oparte o zbiory przybliżone

Reguły decyzyjne

Dla danego zbioru atrybutów A definiujemy język deskryptorów jako trójkę

$$\mathcal{L}(A) = (\mathbf{D}, \{\vee, \wedge, \neg\}, \mathbf{F})$$

gdzie
— $\mathbf{D}$ jest zbiorem *deskryptorów*

$$\mathbf{D} = \{(a = v) : a \in A \text{ and } v \in Val_a\};$$

— $\{\vee, \wedge, \neg\}$ jest zbiorem standardowych operatorów logicznych;
— $\mathbf{F}$ jest zbiorem formuł logicznych zbudowanych na deskryptorach z $\mathbf{D}$.
— Dla każdego zbioru atrybutów $B \subseteq A$ oznaczamy:
$\mathbf{D}|_B = \{(a = v) : a \in B \text{ and } v \in Val_a\}$, czyli zbiór deskryptorów obciętych do B.
$\mathbf{F}|_B$ zbiór formuł zbudowanych na $\mathbf{D}|_B$.

Semantyka:

Niech $\mathbb{S} = (U, A)$ będzie tablicą informacyjną. Każda formuła $\phi \in \mathbf{F}$, jest (semantycznie) skojarzona ze zbiorem $[[\phi]]_{\mathbb{S}}$ zawierającym obiekty spełniające ϕ.

Formalnie możemy indukcyjnie definiować semantykę jak następująco:

$$[[(a = v)]]_{\mathbb{S}} = \{x \in U : a(x) = v\} \tag{5.1}$$
$$[[\phi_1 \vee \phi_2]]_{\mathbb{S}} = [[\phi_1]]_{\mathbb{S}} \cup [[\phi_2]]_{\mathbb{S}} \tag{5.2}$$
$$[[\phi_1 \wedge \phi_2]]_{\mathbb{S}} = [[\phi_1]]_{\mathbb{S}} \cap [[\phi_2]]_{\mathbb{S}} \tag{5.3}$$
$$[[\neg\phi]]_{\mathbb{S}} = U \setminus [[\phi]]_{\mathbb{S}} \tag{5.4}$$

Każda formuła ϕ może być charakteryzowana przez:
— $length(\phi)$ = liczba deskryptorów w ϕ;
— $support(\phi) = |[[\phi]]_{\mathbb{S}}|$ = liczba obiektów spełniających formułę

Definicja reguł decyzyjnych Niech $\mathbb{S} = \{U, A \cup \{dec\}\}$ będzie tablicą decyzyjną. Regułą decyzyjną dla $\mathbb{S}$ nazywamy formuły postaci:

$$\phi \Rightarrow \delta$$

gdzie $\phi \in \mathbf{F}_A$ i $\delta \in \mathbf{F}_{dec}$.

Formułę ϕ nazywamy *poprzednikiem* (lub założeniem) reguły $\mathbf{r}$, a δ nazywamy *następnikiem* (lub tezą) reguły $\mathbf{r}$.

Oznaczamy poprzednik i następnik reguły $\mathbf{r}$ przez $prev(\mathbf{r})$ oraz $cons(\mathbf{r})$.

Reguły atomowa: Są to reguły postaci:

$$\mathbf{r} \equiv (a_{i_1} = v_1) \wedge ... \wedge (a_{i_m} = v_m) \Rightarrow (dec = k) \tag{5.5}$$

Każda reguła decyzyjna $\mathbf{r}$ postaci (5.5) może być charakteryzowana następującymi cechami:

$length(\mathbf{r})$ = liczba deskryptorów występujących w założeniu reguły $\mathbf{r}$
$[\mathbf{r}]$ = nośnik reguły $\mathbf{r}$, czyli zbiór obiektów z U spełniających założenie reguły $\mathbf{r}$
$support(\mathbf{r})$ = liczba obiektów z U spełniających założenie reguły$\mathbf{r}$: $support(\mathbf{r}) = card([\mathbf{r}])$
$confidence(\mathbf{r})$ = wiarygodność reguły $\mathbf{r}$: $confidence(\mathbf{r}) = \frac{|[\mathbf{r}] \cap DEC_k|}{|[\mathbf{r}]|}$

Mówimy, że reguła $\mathbf{r}$ jest *niesprzeczna* z $\mathbb{A}$ jeśli

$$confidence(\mathbf{r}) = 1$$

Minimalne niesprzeczne reguły: Niech $\mathbb{S} = (U, A \cup \{dec\})$ będzie daną tablicą decyzyjną. Niesprzeczną regułę

$$(a_{i_1} = v_1) \wedge ... \wedge (a_{i_m} = v_m) \Rightarrow (dec = k)$$

nazywamy *minimalną niesprzeczną regułą decyzyjną* jeśli usunięcie któregokolwiek z deskryptorów spowoduje, że reguła przestaje być niesprzezcna z $\mathbb{S}$.

Regułowe systemy decyzyjne

Klasyfikatory regułowe działają w trzech fazach:

1. Faza treningu: Generuje pewien zbiór reguł $RULES(\mathbb{A})$ z danej tablicy decyzyjnej $\mathbb{A}$.
2. Faza selekcji reguł: Szuka w $RULES(\mathbb{A})$ tych reguł, które są wspierane przez obiekt x. Oznaczamy zbiór tych reguł przez $MatchRules(\mathbb{A}, x)$.
3. Faza klasyfikacji: wyznacza klasędecyzyjną dla x za pomocą reguł z $MatchRules(\mathbb{A}, x)$ według następującego schematu:
   a) Jeśli $MatchRules(\mathbb{A}, x)$ jest pusty: odpowiedź dla x jest "$NIEWIEM$", tzn. nie mamy podstaw, aby klasyfikować obiekt x do którejkolwiek z klas;
   b) Jeśli $MatchRules(\mathbb{A}, x)$ zawiera tylko obiekty z k-tej klasy: wówczas $dec(x) = k$;
   c) Jeśli $MatchRules(\mathbb{A}, x)$ zawiera reguły dla różnych klas decyzyjnych: wówczas decyzja dla x określimy za pomocą pewnego, ustalonego schematu głosowania między regułami z $MatchRules(\mathbb{A}, x)$.

Szukanie minimalnych reguł decyzyjnych

— Każda reguła powstaje poprzez skracanie opisu jakiegoś obiektu.
— Redukty lokalne
— Te same heurystyki dla reduktów decyzyjnych.

Hurt.	Jakość obsługi	Jakość towaru	Obok autostrady?	Centrum?	decyzja
ID	a_1	a_2	a_3	a_4	dec
1	dobra	dobra	nie	nie	strata
2	dobra	dobra	nie	tak	strata
3	bdb	dobra	nie	nie	zysk
4	slaba	super	nie	nie	zysk
5	slaba	niska	tak	nie	zysk
6	slaba	niska	tak	tak	strata
7	bdb	niska	tak	tak	zysk
8	dobra	super	nie	nie	strata
9	dobra	niska	tak	nie	zysk
10	slaba	super	tak	nie	zysk
11	dobra	super	tak	tak	zysk
12	bdb	super	nie	tak	zysk
13	bdb	dobra	tak	nie	?
14	slaba	super	nie	tak	?

$\mathbb{M}$	1	2	6	8
3	a_1	a_1, a_4	a_1, a_2, a_3, a_4	a_1, a_2
4	a_1, a_2	a_1, a_2, a_4	a_2, a_3, a_4	a_1
5	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_4	a_1, a_2, a_3
7	a_1, a_2, a_3, a_4	a_1, a_2, a_3	a_1	a_1, a_2, a_3, a_4
9	a_2, a_3	a_2, a_3, a_4	a_1, a_4	a_2, a_3
10	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_2, a_4	a_1, a_3
11	a_2, a_3, a_4	a_2, a_3	a_1, a_2	a_3, a_4
12	a_1, a_2, a_4	a_1, a_2	a_1, a_2, a_3	a_1, a_4

$$f_{u_3} = (\alpha_1)(\alpha_1 \vee \alpha_4)(\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_2) = \alpha_1$$

Reguły:

$$(a_1 = \text{bdb}) \implies dec = \text{zysk}$$

$\mathbb{M}$	1	2	6	8
3	a_1	a_1, a_4	a_1, a_2, a_3, a_4	a_1, a_2
4	a_1, a_2	a_1, a_2, a_4	a_2, a_3, a_4	a_1
5	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_4	a_1, a_2, a_3
7	a_1, a_2, a_3, a_4	a_1, a_2, a_3	a_1	a_1, a_2, a_3, a_4
9	a_2, a_3	a_2, a_3, a_4	a_1, a_4	a_2, a_3
10	a_1, a_2, a_3	a_1, a_2, a_3, a_4	a_2, a_4	a_1, a_3
11	a_2, a_3, a_4	a_2, a_3	a_1, a_2	a_3, a_4
12	a_1, a_2, a_4	a_1, a_2	a_1, a_2, a_3	a_1, a_4

$$\begin{aligned} f_{u_8} &= (\alpha_1 \vee \alpha_2)(\alpha_1)(\alpha_1 \vee \alpha_2 \vee \alpha_3)(\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4)(\alpha_2 \vee \alpha_3) \\ &\quad (\alpha_1 \vee \alpha_3)(\alpha_3 \vee \alpha_4)(\alpha_1 \vee \alpha_4) \\ &= \alpha_1(\alpha_2 \vee \alpha_3)(\alpha_3 \vee \alpha_4) = \alpha_1\alpha_3 \vee \alpha_1\alpha_2\alpha_4 \end{aligned}$$

Reguły:

— $(a_1 = \text{dobra}) \wedge (a_3 = \text{nie}) \implies dec = \text{strata}$

— $(a_1 = \text{dobra}) \wedge (a_2 = \text{super}) \wedge (a_4 = nie) \implies dec = \text{strata}$

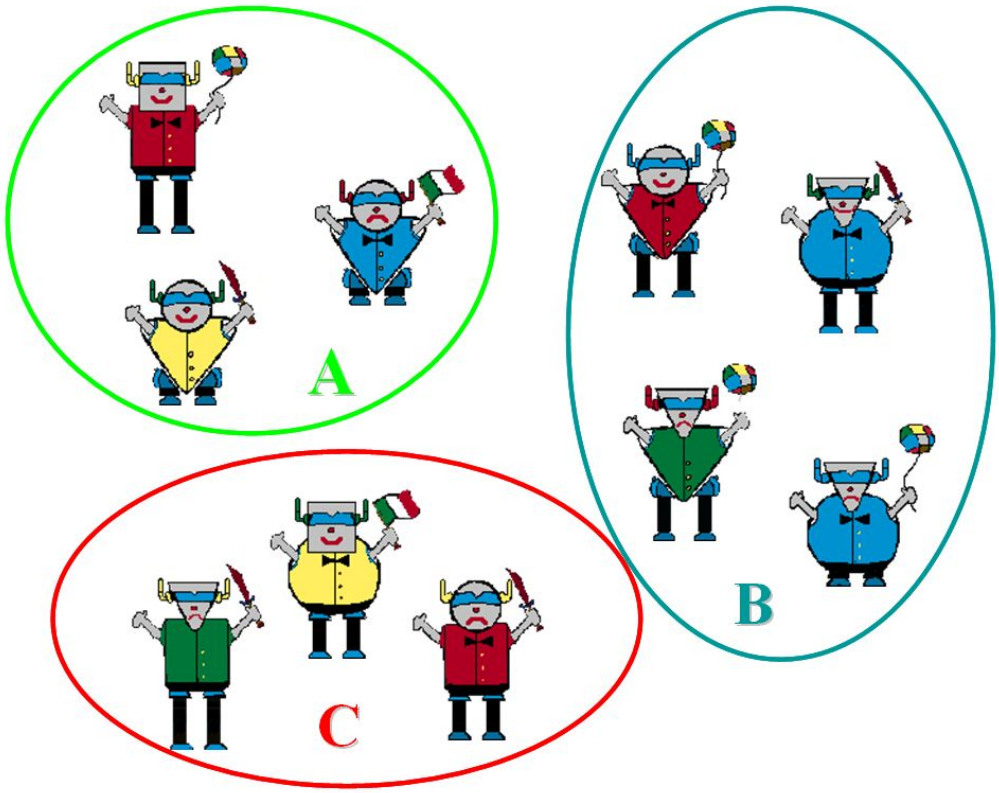
6. Metoda drzew decyzyjnych

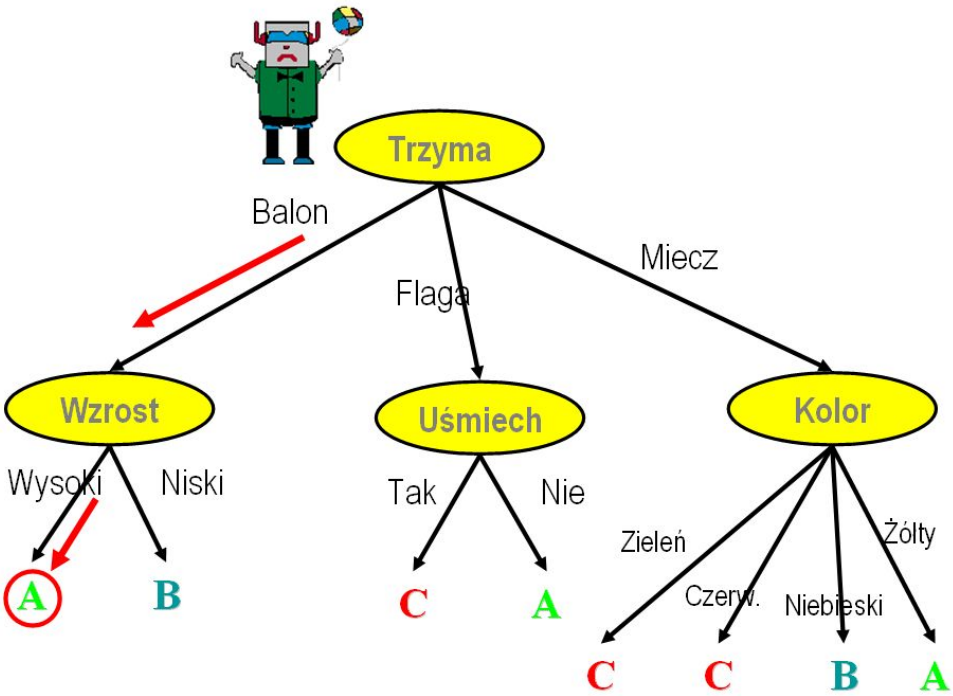
6.1. Metoda drzew decyzyjnych

6.1.1. Wprowadzenie

Definicje

— **Jest to struktura drzewiasta, w której**
  — **węzły wewnętrzne** zawierają testy na wartościach atrybutów
  — z każdego węzła wewnętrznego wychodzi tyle **gałęzi**, ile jest możliwych wyników testu w tym węzle;
  — **liście** zawierają decyzje o klasyfikacji obiektów
— **Drzewo decyzyjne koduje program zawierający same instrukcje warunkowe**


A
B
C


Trzyma
Balon
Flaga
Miecz
Wzrost
Wysoki
Niski
Uśmiech
Tak
Nie
Kolor
Zieleń
Czerw.
Niebieski
Żółty
A
B
C
A
C
C
B
A

Funkcje testu

x	outlook	Temperature	humidity	wind	play(x)
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cold	normal	weak	yes
6	rain	cold	normal	strong	no
7	overcast	cold	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cold	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Wyróżniamy 2 klasy funkcji testów

— Testy operują na wartościach pojedyńczego atrybutu (ang. univariate tree):

$$t : V_a \rightarrow R_t;$$

— Testy będące kombinacją wartości kilku atrybutów (ang. multivariate tree):

$$t : V_{a_1} \times V_{a_2} \times ... \times V_{a_k} \rightarrow R_t;$$

gdzie
— Va : dziedzina atrybutu a;
— Rt : zbiór możliwych wyników testu;

— Dla atrybutów nominalnych a_i oraz obiektu x:
  — test tożsamościowy: $t(x) \rightarrow a_i(x)$
  — test równościowy: $t(x) = \begin{cases} 1 & \text{if } (a_i(x) = v) \\ 0 & \text{otherwise} \end{cases}$
  — test przynależnościowy: $t(x) = \begin{cases} 1 & \text{if } (a_i(x) \in V) \\ 0 & \text{otherwise} \end{cases}$
— Dla atrybutów o wartościach ciągłych:
  — test nierównościowy: $t(x) = \begin{cases} 1 & \text{if } (a_i(x) > c) \\ 0 & \text{otherwise, i.e., } (a_i(x) \leqslant c) \end{cases}$ gdzie c jest wartością progową lub cięciem

Optymalne drzewo

— Jakość drzewa ocenia się
  — za pomocą rozmiaru: im drzewo jest mniejsze, tym lepsze
    — mała liczba węzłów,
    — mała wysokość, lub
    — mała liczba liści;
  — za pomocą dokładności klasyfikacji na zbiorze treningowym
  — za pomocą dokładności klasyfikacji na zbiorze testowym

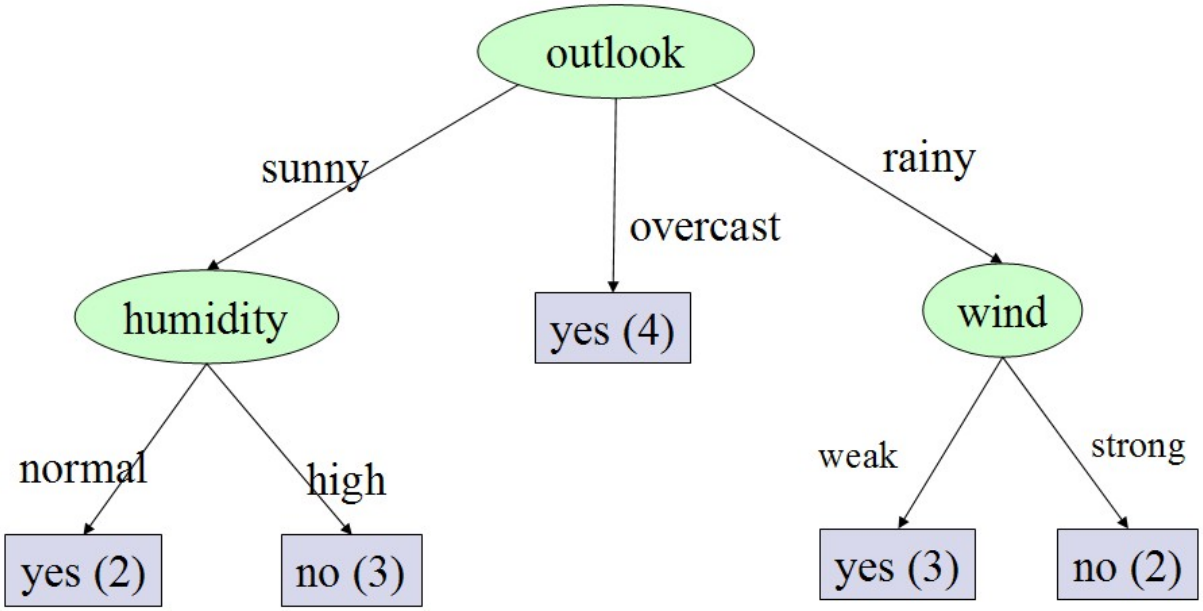
— Na przykład:

$$Q(T) = \alpha \cdot size(T) + \beta \cdot accuracy(T, P)$$

gdzie α, β są liczbami rzeczywistymi
size(.) jest rozmiarem drzewa
accuracy(.,.) jest jakością klasyfikacji

Problem konstrukcji drzew optymalnych: **Dane są:**
— tablica decyzyjna $\mathbb{S}$
— zbiór funkcji testów **TEST**,
— kryterium jakości Q
Szukane: drzewo decyzyjne **T** o najwyższej jakości $Q(\mathbf{T})$.

— Dla większości parametrów, problem szukania optymalnego drzewa jest NP-trudny !
— **Wnioski:**
Trudno znaleźć optymalne drzewo w czasie wielomianowym;
Konieczność projektowania heurystyk.
— **Quiz:** Czy drzewo z przykładu jest optymalne?



6.1.2. Konstrukcja drzew decyzyjnych

Funkcja rekurencyjna *buduj_drzewo*$(U, dec, \mathbf{T})$**:**

```
1: if (kryterium_stopu(U, dec) = true) then
2:     T.etykieta = kategoria(U, dec);
3:     return;
4: end if
5: t := wybierz_test(U, TEST);
6: T.test := t;
7: for v ∈ R_t do
8:     U_v := {x ∈ U : t(x) = v};
9:     utwórz nowe poddrzewo T';
10:    T.gałąź(v) = T';
11:    buduj_drzewo(U_v, dec, T')
12: end for
```

— Kryterium stopu: Zatrzymamy konstrukcji drzewa, gdy aktualny zbiór obiektów:
  — jest pusty lub
  — zawiera obiekty wyłącznie jednej klasy decyzyjnej lub
  — nie ulega podziale przez żaden test
— Wyznaczenie etykiety zasadą większościową:

$$kategoria(P, dec) = \arg \max_{c \in V_{dec}} |P_{[dec=c]}|$$

tzn., etykietą dla danego zbioru obiektów jest klasa decyzyjna najliczniej reprezentowana w tym zbiorze.
— Kryterium wyboru testu: heurytyczna funkcja oceniająca testy.

Kryterium wyboru testu

Każdy zbiór obiektów X ulega podziałowi na klasy decyzyjne:

$$X = C_1 \cup C_2 \cup ... \cup C_d$$

gdzie $C_i = \{u \in X : dec(u) = i\}$.

Wektor $(p_1, ..., p_r)$, gdzie $p_i = \frac{|C_i|}{|X|}$, nazywamy rozkładem klas decyzyjnych w X.

$$Conflict(X) = \sum_{i<j} |C_i| \times |C_j| = \frac{1}{2}\left(|X|^2 - \sum |C_i|^2\right)$$

$$\begin{aligned} Entropy(X) &= -\sum \frac{|C_i|}{|X|} \cdot \log \frac{|C_i|}{|X|} \\ &= -\sum p_i \log p_i \end{aligned}$$

Funkcja $conflict(X)$ oraz $Ent(X)$ przyjmują

— największą wartość, gdy rozkład klas decyzyjnych w zbiorze X jest równomierny.
— najmniejszą wartość, gdy wszystkie obiekty w X są jednej kategorii (X jest **jednorodny**)
W przypadku 2 klas decyzyjnych:

$$Conflict(p, 1-p) = |X|^2 \cdot p(1-p)$$
$$Entropy(p, 1-p) = -p \log p - (1-p) \log(1-p)$$

Niech t definiuje podział X na podzbiory: $X_1 \cup ... \cup X_r$. Możemy stosować następujące miary do oceniania testów:

— liczba par obiektów rozróżnionych przez test t.

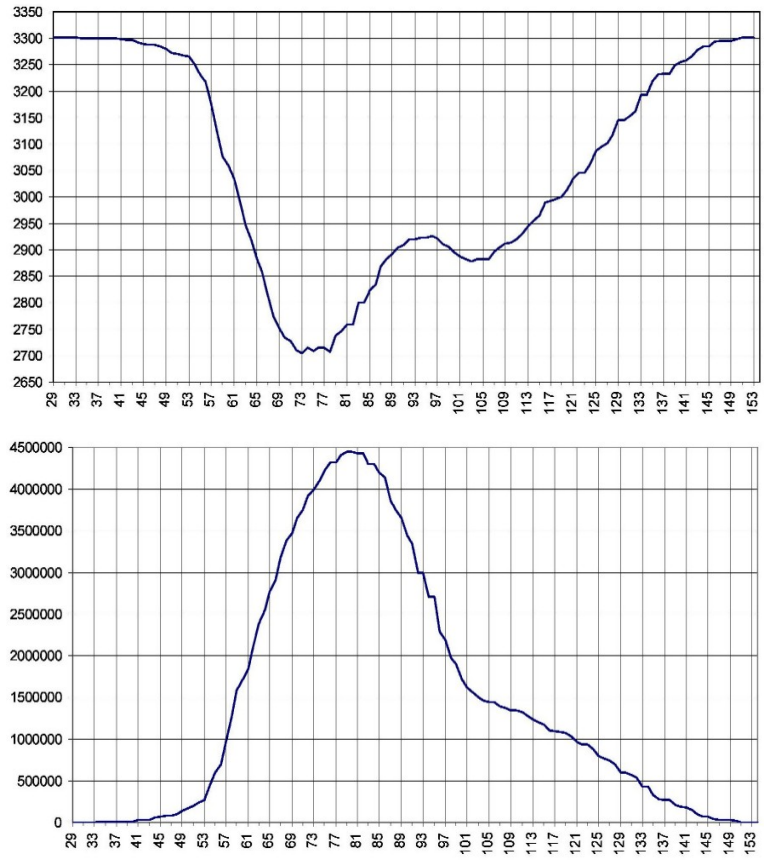
$$disc(t, X) = conflict(X) - \sum conflict(X_i)$$

— kryterium przyrostu informacji (ang. Inf. gain).

$$Gain(t, X) = Entropy(X) - \sum_i p_i \cdot Entropy(X_i)$$

Im większe są wartości tych ocen, tym lepszy jest test.

$$N \times \sum_i p_i \cdot Entropy(X_i)$$



— Monotoniczność: Jeśli t' definiuje drobniejszy podział niż t to

$$Gain(t', X) \geqslant Gain(t, X)$$

(analogiczną sytuację mamy dla miary $conflict()$.

— Funkcje ocen testu t przyjmują małe wartości jeśli rozkłady decyzyjne w podzbiorach wyznaczanych przez t są zbliżone.

Zamiast bezwzględnego przyrostu informacji, stosujemy współczynnik przyrostu informacji

$$Gain\_ratio = \frac{Gain(t, X)}{iv(t, X)}$$

gdzie $iv(t, X)$, zwana wartością informacyjną testu t (information value), jest definiowana jak nast.:

$$iv(t, X) = -\sum_{i=1}^{r} \frac{|X_i|}{|X|} \cdot \log \frac{|X_i|}{|X|}$$

Ocena funkcji testu

— Rozróżnialność:

$$disc(t, X) = conflict(X) - \sum conflict(X_i)$$

— Przyrostu informacji (Information gain).

$$Gain(t, X) = Entropy(X) - \sum_i p_i \cdot Entropy(X_i)$$

— Współczynnik przyrostu informacji (gain ratio)

$$Gain\_ratio = \frac{Gain(t, X)}{-\sum_{i=1}^{r} \frac{|X_i|}{|X|} \cdot \log \frac{|X_i|}{|X|}}$$

— Inne (np. Gini's index, test χ^2, ...)

Przycinanie drzew

— Problem nadmiernego dopasowania do danych trenujących (prob. przeuczenia się).
— Rozwiązanie:
  — zasada najkrótszego opisu: skracamy opis kosztem dokładności klasyfikacji w zbiorze treningowym
  — zastąpienie podrzewa nowym liściem (przycinanie) lub mniejszym podrzewem.
— Podstawowe pytania:
  Q: Kiedy poddrzewo może być zastąpione liściem?
  A: Jeśli nowy liść jest niegorszy niż istniejące poddrzewo dla nowych obiektów (nienależących do zbioru treningowego).
  Q: Jak to sprawdzić?
  A: Testujemy na próbce zwanej "zbiorem przycinania"!

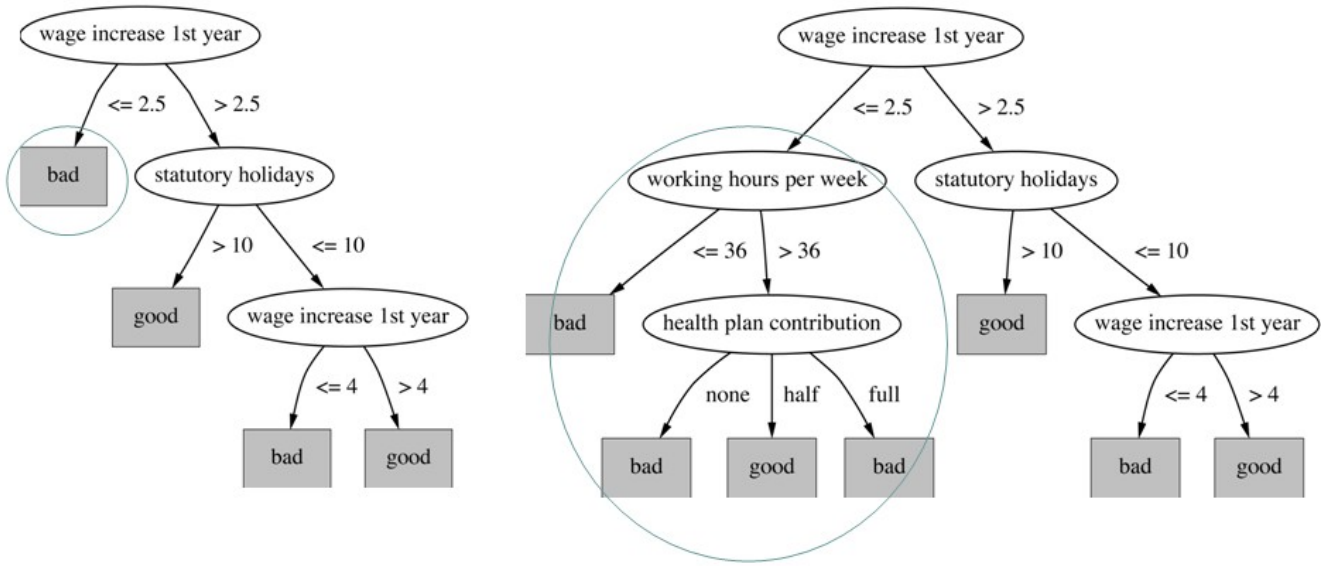
Funkcja *przytnij*$(\mathbf{T}, P)$

1: **for all** $n \in \mathbf{T}$ **do**
2: utwórz nowy liść l etykietowany kategorią dominującą w zbiorze P_n
3: **if** (liść l jest niegorszy od poddrzewa o korzeniu w n pod względem zbioru P) **then**
4: zastąp poddrzewo o korzeniu w n liściem l;
5: **end if**
6: **end for**
7: return $\mathbf{T}$

— Niech
$e_T(l)$ - błąd klasyfikacji kandydującego liścia l,
$e_T(n)$ - błąd klasyfikacji poddrzewa o korzeniu w n.
— Przycinanie ma miejsce, gdy

$$e_T(l) \leqslant e_T(n) + \mu\sqrt{\frac{e_T(n)(1-e_T(n))}{|P_{T,n}|}}$$

na ogół przyjmujemy $\mu = 1$.



Problem brakujących wartości (ang. null-values)

Możliwe są następujące rozwiązania:
— Zredukowanie wartości kryterium wyboru testu (np. przyrostu informacji) dla danego testu o współczynnik równy:

$$\frac{\text{liczba obiektów z nieznanymi wartościami}}{\text{liczba wszystkich obiektów}}$$

— Wypełnienie nieznanych wartości atrybutu najczęściej występującą wartością w zbiorze obiektów związanych z aktualnym węzłem
— Wypełnienie nieznanych wartości atrybutu średnią ważoną wyznaczoną na jego zbiorze wartości.

Możliwe rozwiązania:
— Zatrzymanie procesu klasyfikacji w aktualnym węźle i zwrócenie większościowej etykiety dla tego węzła (etykiety, jaką ma największą liczbę obiektów trenujących w tym węźle)
— Wypełnienie nieznanej wartości według jednej z heurystyk podanych wyżej dla przypadku konstruowania drzewa
— Uwzględnienie wszystkich gałęzi (wszystkich możliwych wyników testu) i połączenie odpowiednio zważonych probabilistycznie rezultatatów w rozkład prawdopodobieństwa na zbiorze możliwych klas decyzyjnych dla obiektu testowego.

7. Problem dyskretyzacji

7.1. Problem dyskretyzacji

7.1.1. Przypomnienia podstawowych pojęć

Tablicą decyzyjną nazywamy strukturę $\mathbb{S} = (U, A \cup \{dec\})$ gdzie
U nazywa się **zbiorem obiektów**

$$\boxed{U = \{u_1, ..., u_n\}}$$

A jest zbiorem **atrybutów** postaci

$$\boxed{a_j : U \longrightarrow V_j}$$

dec jest specjalnym atrybutem zwanym **decyzją**

$$\boxed{dec : U \longrightarrow \{1, ..., d\}}$$

A

$\mathbb{S}$	a_1	a_2	...	dec
u_1	100	27	...	1
u_2	120	86	...	1
u_3	70	52	...	1
u_4	95	18	...	1
...	...	...	...	...
u_{1200}	71	82	...	2
...	...	...	...	...

— **Klasy decyzyjne:**
dec definiuje podział $U = DEC_1 \cup ... \cup DEC_d$ gdzie

$$\boxed{DEC_k = \{x \in U : dec(x) = k\}}$$

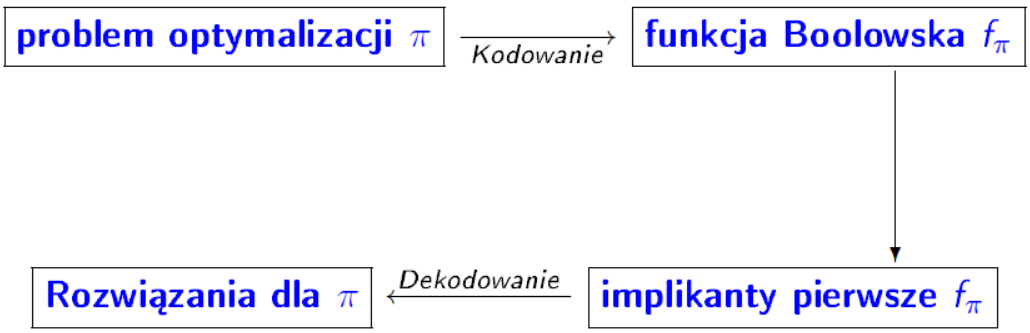
— **Rozróżnialność:** Dane są obiekty $x, y \in U$ zbiór atrybutów $B \subset A$, mówimy, że x, y są **rozróżnialne przez** B wtw, gdy istnieje $a \in B$ taki, że $\boxed{a(x) \neq a(y)}$

Zbiór atrybutów $B \subset A$ nazywamy **reduktem** tablicy $\mathbb{S}$ wtw, gdy
— dla dowolnych obiektów $x, y \in U$
jeśli $dec(x) \neq dec(y)$ i x, y są rozróżnialne przez A,
to są również rozróżnialne przez B
(B *zachowuje rozróżnialność zbioru* A)
— B jest niezredukowalny (tzn. żaden właściwy podzbiór B nie zachowuje rozróżnialności zbioru A)
Problemy:
— Czy istnieje redukt zawierający k atrybutów?
— Znaleźć redukt o najmniejszej liczbie atrybutów.

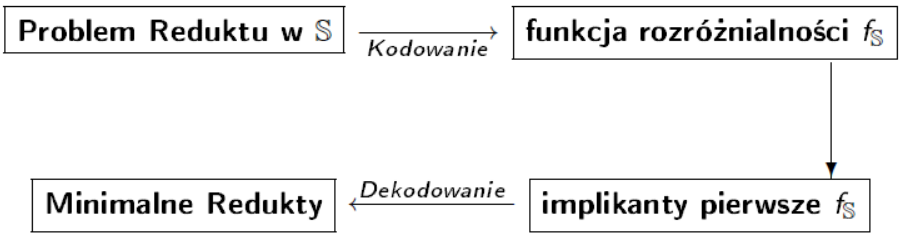
— funkcje $f : \{0,1\}^n \to \{0,1\}$ nazywamy Boolowskimi.
— monotoniczne funkcje Boolowskie można zapisać bez użycia negacji.
— jednomian $\boxed{f' = x_{i_1} x_{i_2} ... x_{i_k}}$ nazywamy implikantem pierwszym funkcji monotonicznej f jeśli
  — $f'(\mathbf{x}) \leqslant f(\mathbf{x})$ dla każdego wektora $\mathbf{x}$ (jest implikantem)
  — każda funkcja większa od f' nie jest implikantem
— Np. funkcja

$$f(x_1, x_2, x_3) = (x_1 + x_2)(x_2 + x3)$$

posiada 2 implikanty pierwsze: $f_1 = x_2$ i $f_2 = x_1 x_3$



Przykład problemu reduktu



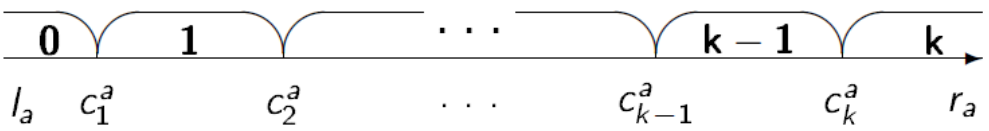
7.1.2. Problem dyskretyzacji

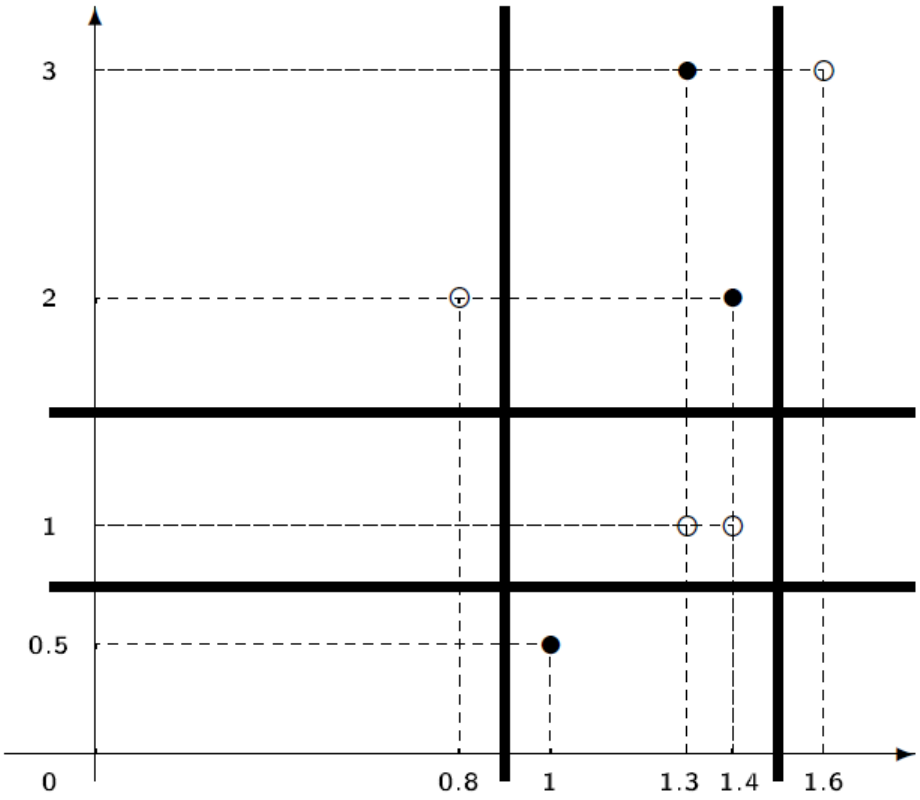
$\mathbb{S}$	a	b	d
u_1	0.8	2	1
u_2	1	0.5	0
u_3	1.3	3	0
u_4	1.4	1	1
u_5	1.4	2	0
u_6	1.6	3	1
u_7	1.3	1	1

$$\Longrightarrow$$

$$\mathbf{P} = \{(a, 0.9), (a, 1.5), (b, 0.75), (b, 1.5)\}$$

$\mathbb{S}^P$	a^P	b^P	d
u_1	0	2	1
u_2	1	0	0
u_3	1	2	0
u_4	1	1	1
u_5	1	2	0
u_6	2	2	1
u_7	1	1	1



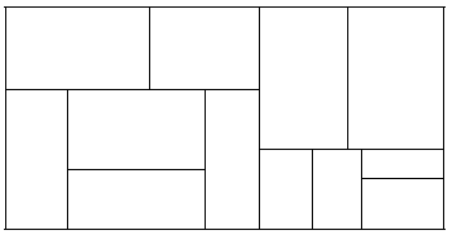


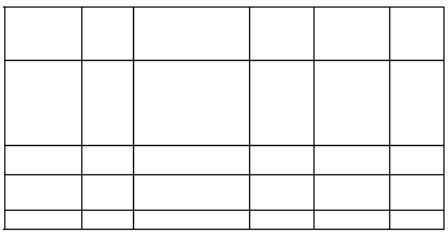
Dana jest niesprzeczna tablica decyzyjna $\mathbb{S} = (U, A \cup \{dec\})$

— Mówimy, że cięcie (a, c) rozróżnia obiekty x, y jeśli albo $a(x) < c < a(y)$ lub $a(y) < c < a(x)$.
— Zbiór cięć $\mathbf{P}$ nazywamy niesprzecznym z $\mathbb{S}$ jeśli dla każdej pary obiektów $x, y \in U$ takich, że $d(x) \neq d(y)$ istnieje cięcie $(a, c) \in \mathbf{P}$ rozróżniające x i y.
— Zbiór cięć $\mathbf{P}_{opt}$ nazywamy optymalnym dla $\mathbb{S}$ jeśli $\mathbf{P}_{opt}$ posiada najmniejszą liczbę cięć wśród niesprzecznych zbiorów cięć.

Istniejące metody dyskretyzacji

1. *Lokalne a globalne metody*:





2. *Statyczne a dynamiczne metody*: Metody statyczne poszukują zbioru cięć dla każdego atrybutu w sposób niezależny od innych atrybutów. Metody dynamiczne szukają cięć na wszystkich atrybutach jednocześnie
3. *Z nadzorem lub bez*:

— Podział na przedziały o równych długościach lub równych częstotliwościach;
— Metoda OneR

— Testy statystyczne

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{r} \frac{(n_{ij} - E_{ij})^2}{E_{ij}}$$

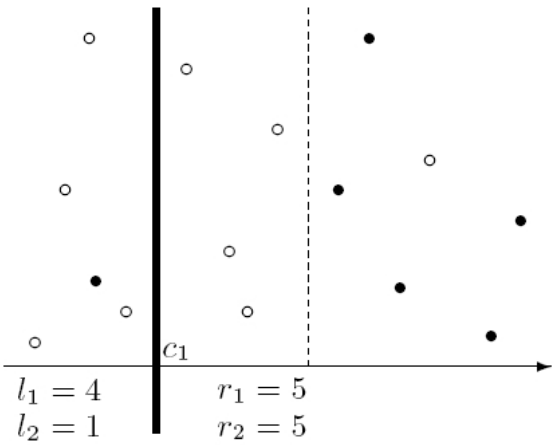
— Z użyciem funkcji entropii;

$$Gain\,(a; c; U) = Ent\,(U) - E\,(a; c; U)$$

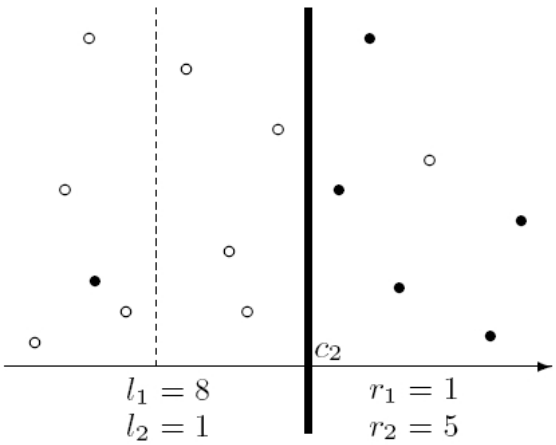
— Gini's index

$$G(a; c; U) = Gini(U) - \frac{|U_L|}{|U|} \cdot Gini(U_L) - \frac{|U_R|}{|U|} \cdot Gini(U_R)$$

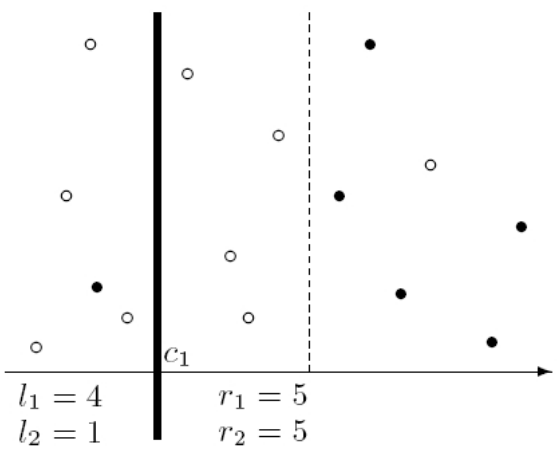
Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	*0.300*	0.343	0.375	0.400	0.420



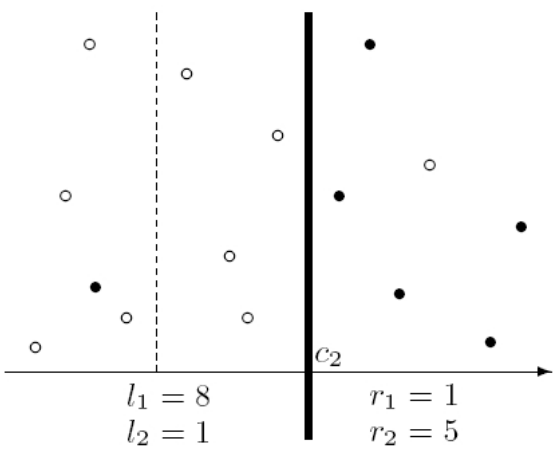
$\chi^2 = 1.25$



$\chi^2 = 7.82$

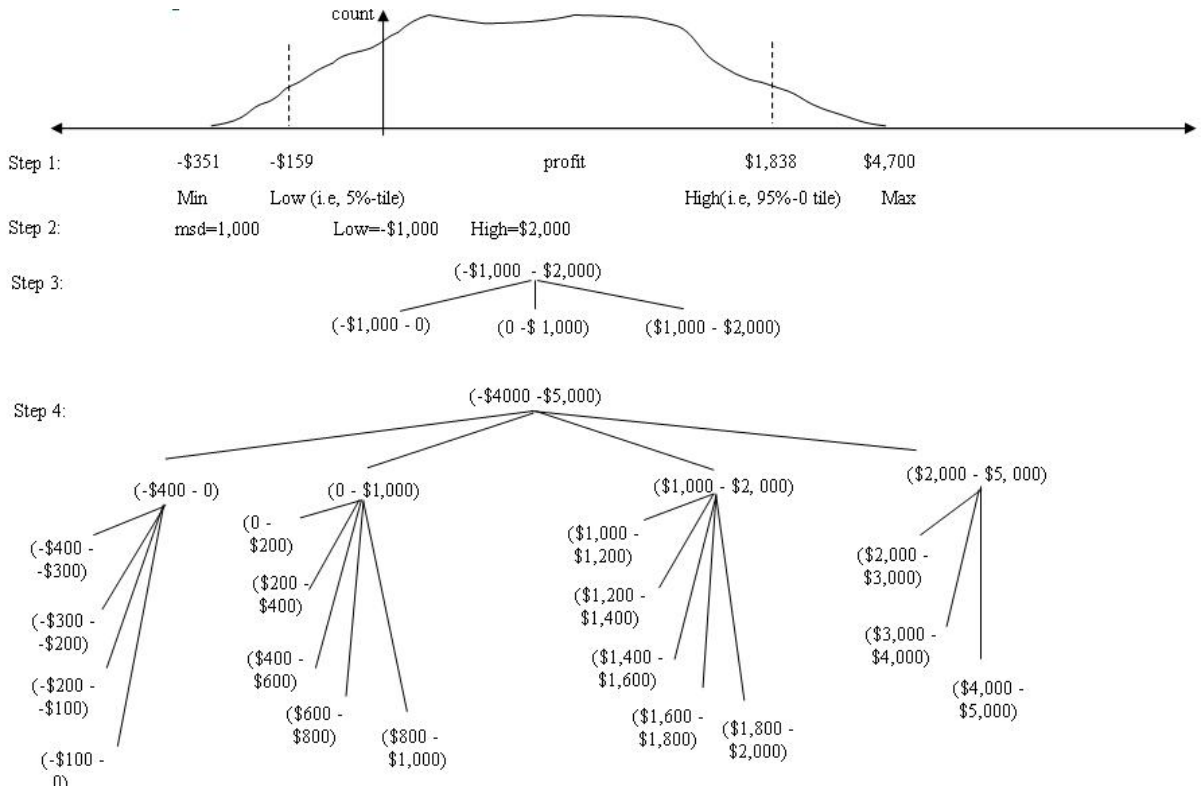


$E(c_1, X) = 0.907$



$E(c_2, X) = 0.562$

— Metoda statyczna bez nadzoru: podział danych numerycznych na równomierne przedziały;
— Rozpatrujemy liczbę różnych najbardziej znaczących cyfr w danym przedziale:
  — jeśli ta liczba wynosi 3,6,7 lub 9 to podziel dany przedział na 3 równe przedziały.
  — jeśli ta liczba wynosi 2,4 lub 8 to podziel dany przedział na 4 równe przedziały.
  — jeśli ta liczba wynosi 1,5 lub 10 to podziel dany przedział na 5 równych przedziałów.



7.1.3. Dyskretyzacja metodą wnioskowania Boolowskiego

Dana jest niesprzeczna tablica decyzyjna $\mathbb{S} = (U, A \cup \{dec\})$
— Niech **C** będzie zbiorem kandydujących cięć dla tablicy $\mathbb{S}$;
— Każde cięcie (a, c) jest skojarzone ze zmienną Boolowską $p_{(a,c)}$;
— Niech $\psi_{x,y}$ będzie funkcją rozróżnialności dla x, y:

$$\psi_{x,y} = \bigvee \{p_{(a,c)} : (a, c) \text{ rozróżnia } x, y\}.$$

— Funkcja boolowska

$$\Psi_{\mathbb{S}} = \prod \{\psi_{x,y} : dec(x) \neq dec(y)\}$$

koduje problem dyskretyzacji.
— Minimalny implikant pierwszy $\Psi_{\mathbb{S}} \Rightarrow$ optymalny zbiór cięć.
Ciecia kandydujące

$$(a, 0.9); (a, 1.15); (a, 1.35); (a, 1.5); \quad (b, 0.75); (b, 1.5); (b, 2.5).$$

Oznaczmy przez $p_1^a, p_2^a, p_3^a, p_4^a, \quad p_1^b, p_2^b, p_3^b$ zmienne Boolowskie odpowiadające cięciom. Wówczas

$$\begin{array}{ll}
\psi(2,1) = p_1^a + p_1^b + p_2^b; & \psi(2,4) = p_2^a + p_3^a + p_1^b; \\
\psi(2,6) = p_2^a + p_3^a + p_4^a + p_1^b + p_2^b + p_3^b; & \psi(2,7) = p_2^a + p_1^b; \\
\psi(3,1) = p_1^a + p_2^a + p_3^b; & \psi(3,4) = p_2^a + p_2^b + p_3^b; \\
\psi(3,6) = p_3^a + p_4^a; & \psi(3,7) = p_2^b + p_3^b; \\
\psi(5,1) = p_1^a + p_2^a + p_3^a; & \psi(5,4) = p_2^b; \\
\psi(5,6) = p_4^a + p_3^b; & \psi(5,7) = p_3^a + p_2^b.
\end{array}$$

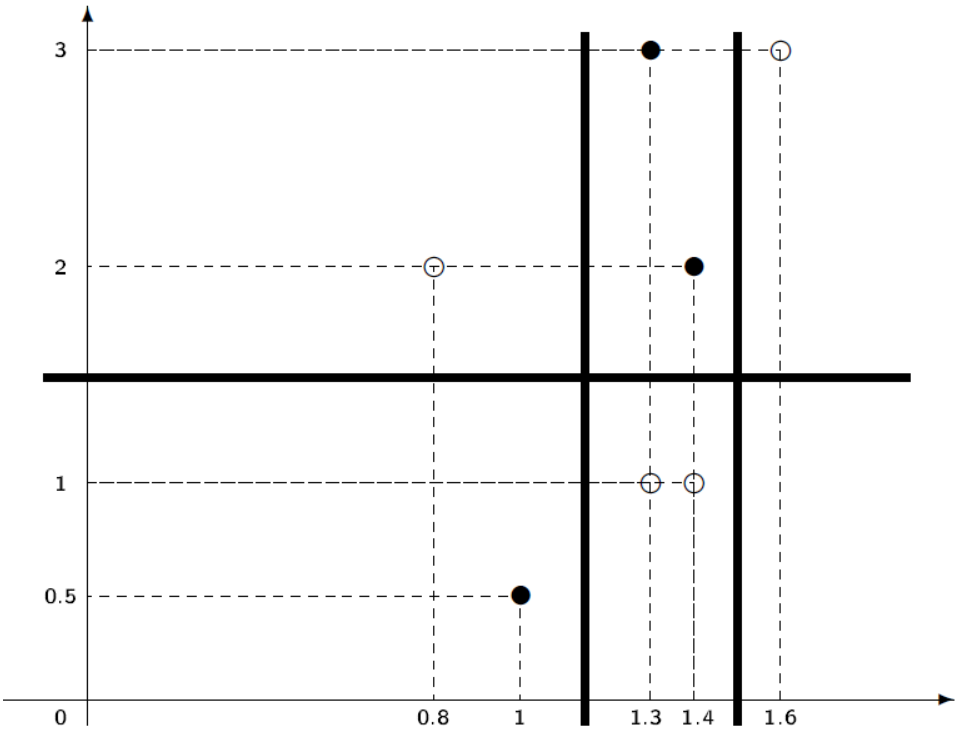
Funkcja kodująca problem dyskretyzacji

$$\begin{aligned}\Phi_{\mathbb{S}} = \quad & \left(p_1^a + p_1^b + p_2^b\right)\left(p_1^a + p_2^a + p_3^b\right)\left(p_1^a + p_2^a + p_3^a\right)\left(p_2^a + p_3^a + p_1^b\right) \\ & p_2^b\left(p_2^a + p_2^b + p_3^b\right)\left(p_2^a + p_3^a + p_4^a + p_1^b + p_2^b + p_3^b\right)\left(p_3^a + p_4^a\right) \\ & \left(p_4^a + p_3^b\right)\left(p_2^a + p_1^b\right)\left(p_2^b + p_3^b\right)\left(p_3^a + p_2^b\right).\end{aligned}$$

Po sprowadzeniu do postaci DNF mamy:

$$\Phi_{\mathbb{S}} = p_2^a p_4^a p_2^b + p_2^a p_3^a p_2^b p_3^b + p_3^a p_1^b p_2^b p_3^b + p_1^a p_4^a p_1^b p_2^b.$$

Czyli optymalnym zbiorem cięć jest $\{(a, 1.15), (a, 1.5), (b, 1.5)\}$



— W algorytmie zachłannym, preferujemy cięcia rozróżniające największą liczbę par obiektów.
— Miara rozróżnialności dla danego cięcia względem zbioru obiektów X:

$$disc(c, X) = conflict(X) - conflict(X_L) - conflict(X_R)$$

gdzie $conflict(X)$ = liczba par obiektów różnych decyzji w zbiorze X.
— Można realizować zachłanną heurystykę w czasie $O(nk \log n|\mathbf{P}|)$, gdzie n jest liczbą obiektów, k jest liczbą atrybutów, $\mathbf{P}$ jest zbiorem cięć znalezionych przez algorytm

$\mathbb{S}^*$	p_1^a	p_2^a	p_3^a	p_4^a	p_1^b	p_2^b	p_3^b	d^*
(u_1, u_2)	1	0	0	0	1	1	0	1
(u_1, u_3)	1	1	0	0	0	0	1	1
(u_1, u_5)	1	1	1	0	0	0	0	1
(u_4, u_2)	0	1	1	0	1	0	0	1
(u_4, u_3)	0	0	1	0	0	1	1	1
(u_4, u_5)	0	0	0	0	0	1	0	1
(u_6, u_2)	0	1	1	1	1	1	1	1
(u_6, u_3)	0	0	1	1	0	0	0	1
(u_6, u_5)	0	0	0	1	0	0	1	1
(u_7, u_2)	0	1	0	0	1	0	0	1
(u_7, u_3)	0	0	0	0	0	1	1	1
(u_7, u_5)	0	0	1	0	0	1	0	1
new	0	0	0	0	0	0	0	0

Uogólnienia problemu dyskretyzacji

— Hiperpłaszczyzny jako cięcia;
— Krzywe wyższego rzędu;
— Grupowanie wartości nominalnych;
— Inne kryteria optymalizacji.

— Dyskretyzacja jako proces wstępnego przetwarzania
— Miarę rozróżnialności można stosować do konstrukcji drzew decyzyjnych.
— Drzewa generowane tą miarą mają dużo ciekawych własności i dużą skuteczność w procesie klasyfikacji.

— Rozumowanie Boolowskie jest prostym, ale mocnym narzędziem w dziedzinie rozpoznawania wzorców, eksploracji danych (ang. Data Mining), sztucznej inteligencji ...
— Złożoność funkcji Boolowskiej kodującej dany problem może być miarą trudności tego problemu.

8. Sieci neuronowe

8.1. Sieci neuronowe

8.1.1. Sztuczne sieci neuronowe

Inspiracje biologiczna/neurologiczna

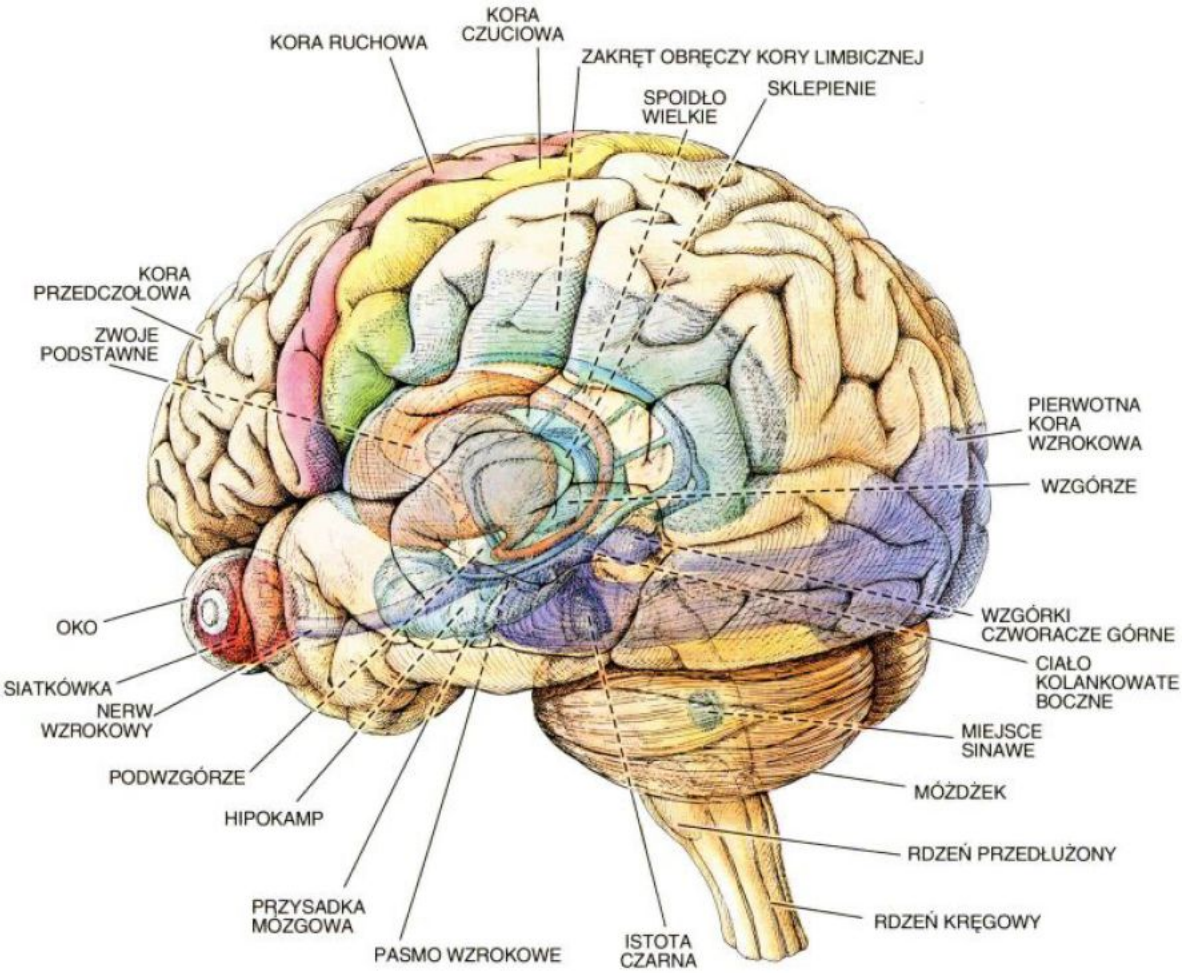
— Podobnie jak symulowane wyżarzanie (SA) i algorytmy genetyczne (GA) sztuczne sieci neuronowe czerpią inspiracje ze zjawisk i procesów zachodzących w otaczającej nas przyrodzie.
— Techniki wykorzystywane przy tworzeniu sztucznych sieci neuronowych sa inspirowane wiedzą o mechanizmach myślenia zaczerpniętymi z fizjologii, neurologii i badań nad procesem poznawania (cognitive science).
— **W układach nerwowych żywych organizmów stosunkowo proste elementy – neurony – współdziałając są w stanie realizować wysokopoziomowe, złożone funkcje.**
— **U podstaw sztucznych sieci neuronowych (ANN) leży idea wykorzystania prostych elementów obliczeniowych – sztucznych neuronów – do tworzenia układów zdolnych rozwiązywać skomplikowane zadania. Siła rozwiązania leży nie w samych elementach obliczeniowych, a w sposobie ich łączenia.**
— **Celem jest otrzymanie systemu, który przejawia cechy podobne do układu nerwowego.**

— Dawno, dawno temu – powstały neurony i układ nerwowy.
— 1868 – J.C. Maxwell opisał mechanizm sprzężenia zwrotnego.
— 1942 – N. Wiener formułuje podstawy współczesnej cybernetyki.
— 1943 – W.S. McCulloch i W.H. Pitts przedstawiają ideę działania sztucznego neuronu.
— 1949 – D.O. Hebb w dziele "The Organization of Behavior" opisuje zasadę (regułę Hebba), w oparciu o którą następuje adaptacja w sieciach neuronowych.
— 1957-1962 – Badania F. Rosenblatta nad pierwszymi fizycznymi modelami sieci neuronowych - perceptronami.
— 1960 – G. Widrow i M. Hoff przedstawiają pierwsze modele sieci jednowarstwowych – ADALINE/MADALINE.
— 1969 – M. Minsky i S. Pappert w książce "Perceptrons" poddają miażdżącej krytyce dotychczasowe osiagnięcia perceptroniki. Wykazują słabości i ograniczoność modelu jednowarstwowego. Rezultatem jest zawieszenie na niemal dekadę ok. 70% badań nad ANN.
— Od 1972 – S. Amari, D. Grossberg, J. Anderson i inni – pierwsze badania nad samoorganizacją w sieciach neuronowych. Wykorzystanie sprzężenia zwrotnego do tworzenia układów naśladujących pamięć asocjacyjną u człowieka.
— 1974 – P. Werbos w swojej pracy doktorskiej podał ideę propagacji wstecznej (backpropagation), dzięki której możliwe stało sie opracowanie efektywnego mechanizmu adaptacji (uczenia się) w sieciach wielowarstwowych.
— 1982 – J. Hopfield przedstawia model sieci rekurencyjnej realizujacej zadania rozpoznawania wzorców i optymalizacji.
— 1986 – D.E. Rumelhart i J.L. McClelland w książce "Parallel Distributed Processing" opisali propagację wsteczną i sformułowali algorytm zmiany wag (reguły delty). Ich praca spo-

wodowała rozpowszechnienie tych idei i w rezultacie lawinowy rozwój badań nad teorią i zastosowaniami ANN.

— 1990 – C. Mead przedstawia pierwsze analogowo-cyfrowe układy elektroniczne wykorzystujące architekture ANN. Pojawiają się poważne zastosowania w przemyśle i obronności.
— 1992 – T. Kohonen przedstawia ideę sieci z samoorganizacją (Self Organising Map – SOM) nazwanej potem jego imieniem. Opisuje także mechanizm uczenia sie kwantyzacji wektorów (Learning Vector Quantisation).
— Od 1988 prowadzone są intensywne badania podstawowe i stosowane mające na celu podanie lepszych metod konstruowania, uczenia i oceny działania sieci neuronowych. Prowadzone są także badania nad podstawami teoretycznymi działania modelu ANN.
— Dzień dzisiejszy – sztuczne sieci neuronowe sa uznanym i powszechnie stosowanym narzędziem w wielu działach nauki i techniki.

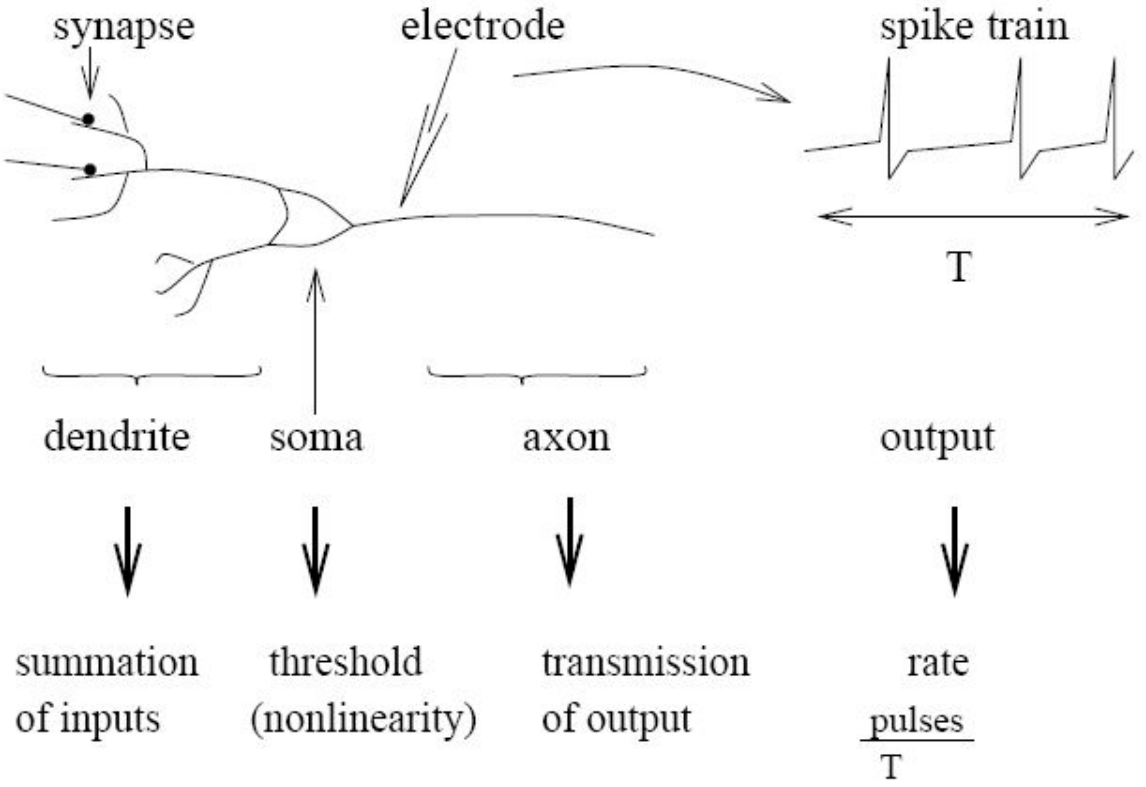
— Masa ok. 1,5 kg - spora wariancja.
— Objetość ok. 1600 cm3, powierzchnia ok. 2000 cm2 czyli niemal trzykrotnie więcej niż kula o tej samej objętości.
— Kora - grubość 2-4 mm:
ok. 10^{10} komórek nerwowych;
ok. 10^{12} komórek glejowych;
ok. 10^{15} połączeń (średnio 7000 na neuron).

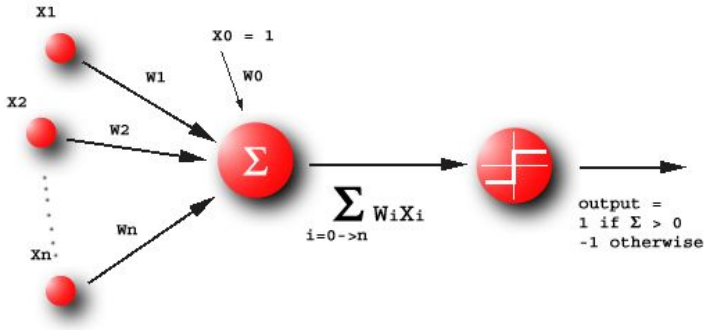


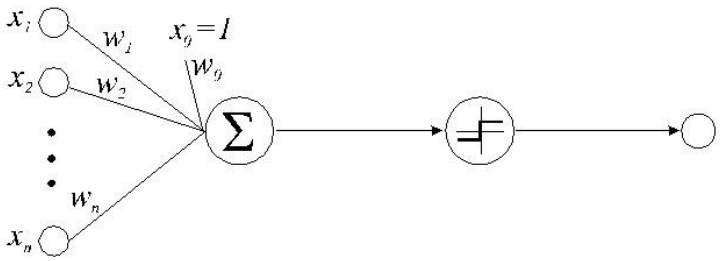
Nie znaleziono korelacji między rozmiarem mózgu a sprawnoscią intelektualną. Jest natomiast związek między rozmiarem kory a zdolnościami mózgu.

Perceptron

1. Potencjały odebrane z innych komórek za pomocą dendrytów są zbierane na błonie ciała komórki.
2. Gdy zebrane potencjały przekroczą wartość progową neuron staje sie aktywny i wysyła sygnały elektryczne (elektrochemiczne) przez akson.
3. Inne neurony odbierają sygnał zależnie od przepustowości synaps.







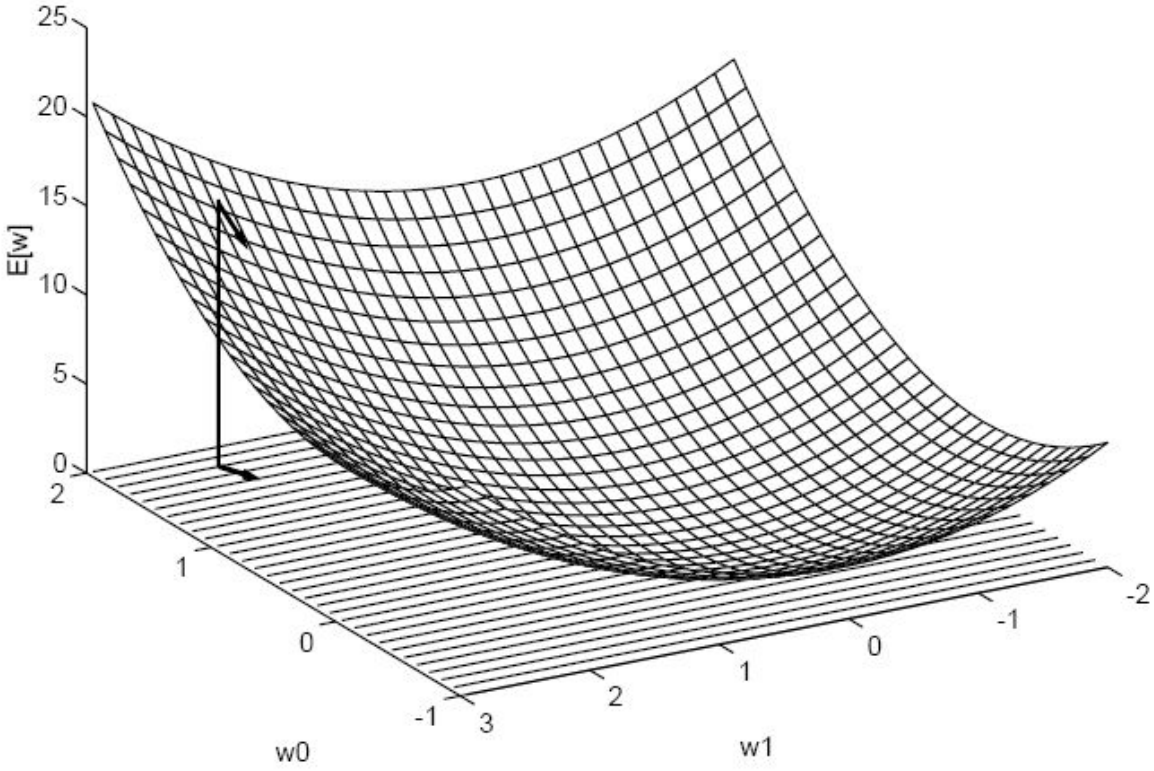
Rozważmy najprotszy przypadek

$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Szukajmy wektora $\mathbf{w} = (w_1, ..., w_n)$ minimalizujące funkcję błędu:

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

gdzie D jest zbiorem przykładów treningowych.



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

Alg. gradientu

1. Inicjalizujemy wagi neuronu (sieci) np. losowo.
2. Powtarzamy
   - Inicjalizujemy $\Delta\vec{w} = \vec{0}$;
   - Podajemy kolejny przykład (x_i, d_i) do sieci i liczymy wartość $o(x_i)$;
   - Obliczamy błąd i poprawiamy wagi według reguły uczenia:

$$\vec{w}^{nowy} \leftarrow \vec{w}^{stary} + \Delta w_i$$

gdzie

$$\Delta\vec{w} = -\eta \nabla E[\vec{w}]$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}$$

Pierwszym krokiem mającym naprawić niedoskonałości perceptronu, jest wprowadzenie nieliniowych, ciagłych i różniczkowalnych funkcji aktywacji dla neuronów. Standardowo, stosuje się:

— Logistyczna funkcje sigmoidalna (unipolarna):

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

o ważnej własności: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$.

— Tangens hiperboliczny (bipolarny):

$$\phi(x) = \tanh(\beta x)$$

o ważnej własności: $\frac{d\phi(x)}{dx} = \beta(1 - \phi^2(x))$.

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 = \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\ \frac{\partial}{\partial w_i}(t_d - o_d) = \sum_d (t_d - o_d)\left(-\frac{\partial o_d}{\partial w_i}\right) \\
&= -\sum_d (t_d - o_d)\ \frac{\partial o_d}{\partial net_d}\ \frac{\partial net_d}{\partial w_i}
\end{aligned}$$

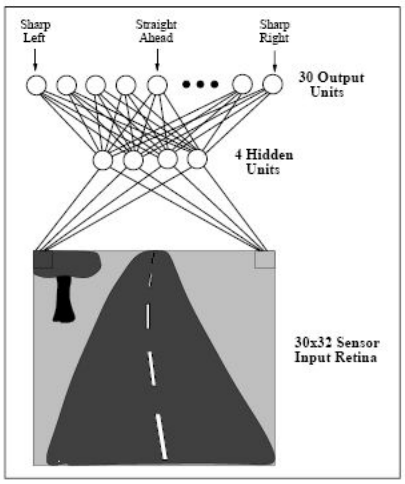
Wiemy, że

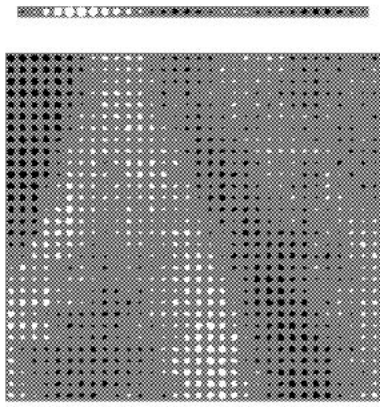
$$\begin{aligned}
\frac{\partial o_d}{\partial net_d} &= \frac{\partial \sigma(net_d)}{\partial net_d} = o_d(1 - o_d) \\
\frac{\partial net_d}{\partial w_i} &= \frac{\partial(\vec{w}\cdot\vec{x}_d)}{\partial w_i} = x_{i,d}
\end{aligned}$$

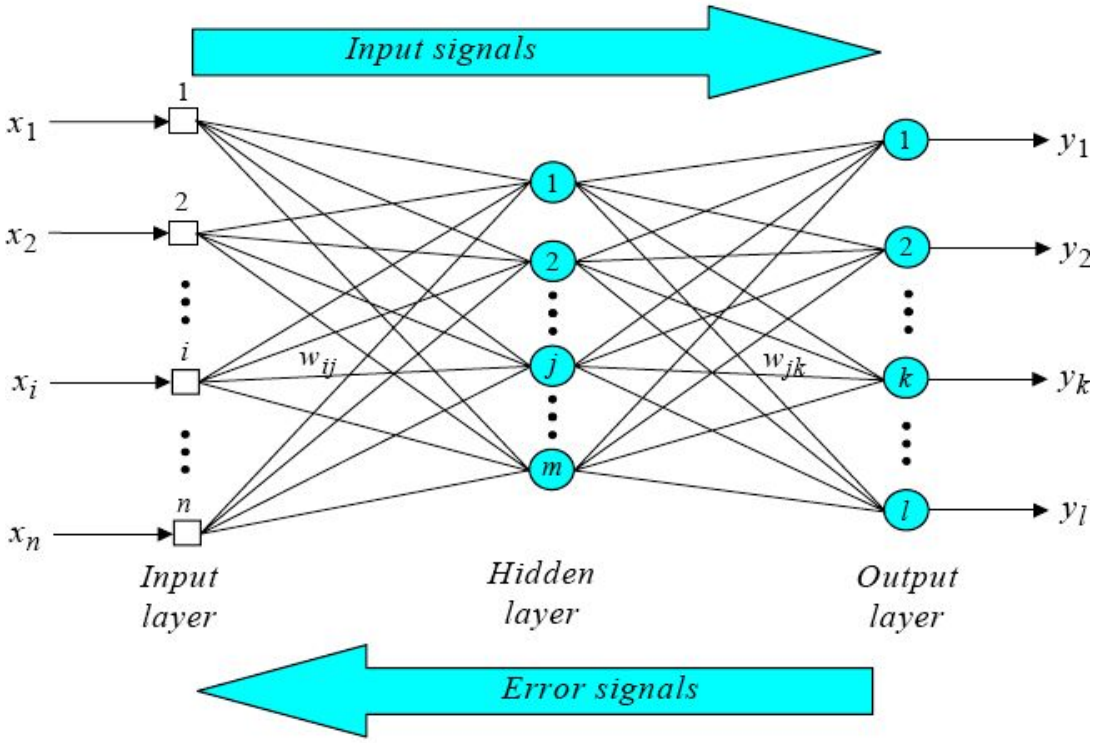
Więc

$$\frac{\partial E}{\partial w_i} = -\sum_{d\in D}(t_d - o_d)o_d(1 - o_d)x_{i,d}$$

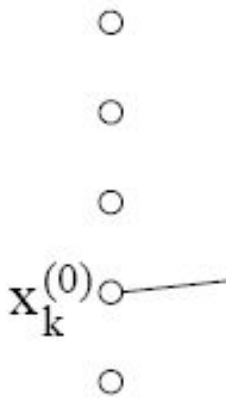
Warstwowe siec







Until satisfied, Do
— **For each training example, Do**



1. **Input the training example to the network and compute the network outputs**

2. **For each output unit** k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. **For each hidden unit** h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. **Update each network weight** $w_{i,j}$

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j}$$

where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}$$

Inne modele i zastosowania

— **Predykacja:** Sieci neuronowe są często wykorzystywane, aby na podstawie pewnych danych wejściowych przewidywała dane wyjściowe. Wazną zaletą jest to, ze sieć może nauczyć się przewidywania sygnałów wyjściowych bez jawnego zdefiniowania związku między danymi wejściowymi a wyjściowymi. Tego typu układy są też często przydatne w zadaniach związanych ze sterowaniem.

— **Klasyfikacja i rozpoznawanie:** Zadanie polega na przewidywaniu identyfikatora klasy, do której dany obiekt należy na podstawie wcześniej zaobserwowanych (nauczonych) przykładów..

— **Kojarzenie danych:** Sieci neuronowe, dzięki zdolności uczenia się i uogólniania doświadczeń, pozwalają zautomatyzować procesy wnioskowania i pomagają wykrywać istotne powiązania pomiędzy danymi. Sieci neuronowe, dzięki zdolności uczenia się i uogólniania doświadczeń, pozwalają zautomatyzować procesy wnioskowania i pomagają wykrywać istotne powiązania pomiędzy danymi.

— **Analiza danych:** Zadanie polega na znalezieniu związków pomiędzy danymi. Realizacja tego zadania przez sieci neuronowe daje nowe możliwości w zakresie prowadzenia analiz ekonomicznych.

— **Filtracja sygnałów** Dane gospodarcze pochodzące z różnych źródeł są zakłócone. Klasyczne metody eliminacji "szumów" pozwalają usunąć zakłócenia o charakterze losowym, lecz nie dają podstaw do eliminacji przekłamań systematycznych.

— **Optymalizacja** Sieci neuronowe - zwłaszcza sieci Hopfielda - dobrze nadają się do optymalizacji decyzji gospodarczych. Doświadczalnie potwierdzono możliwości sieci do rozwiązywania zadań optymalizacji statycznej i dynamicznej. Szczególnie ciekawe jest zastosowanie sieci do optymalizacji kombinatorycznej i zagadnień (NP)trudnych obliczeniowo, np. TSP.

Model sieci	Zastosowanie
Wielowarstwowa z propagacją wsteczną	Klasyfikacja Predykcja
Sieć rekurencyjna (Hopfielda)	Pamięć skojarzeniowa
Sieć konkurencyjna samo-organizująca sieć (Kohonen)	PCA Analiza skupień

9. Teoria systemów uczących się

9.1. Teoria systemów uczących się

9.1.1. Wstęp do komputerowego uczenia się pojęć

— Np. Pokazać, że dla każdego $n \in \mathbb{N}$ zachodzi

$$\Psi(n): \boxed{1^2 + 2^2 + ... + n^2 = \frac{n(n+1)(2n+1)}{6}}$$

— Indukcja pełna:

$$\Psi(1) \quad \text{oraz} \quad \forall_{n \geqslant 1}[\Psi(n) \implies \Psi(n+1)]$$

— Indukcja niepełna: czy wystarczy sprawdzić, np.

$$\Psi(1), \Psi(2), \Psi(3), \Psi(4)?$$

Podejście indukcyjne:
Wnioskowanie na podstawie skończonego zbioru obserwacji

Jakie prawa rządzą procesem indukcyjnego uczenia się pojęć?

Szukamy teorii obejmującej zagadnienia:

— **Szansy na skuteczne wyuczanie się pojęć;**
— **Niezbędnej liczby przykładów treningowych;**
— **Złożoności przestrzeni hipotez;**
— **Jakości aproksymacji;**
— **Metod reprezentacji danych treningowych;**

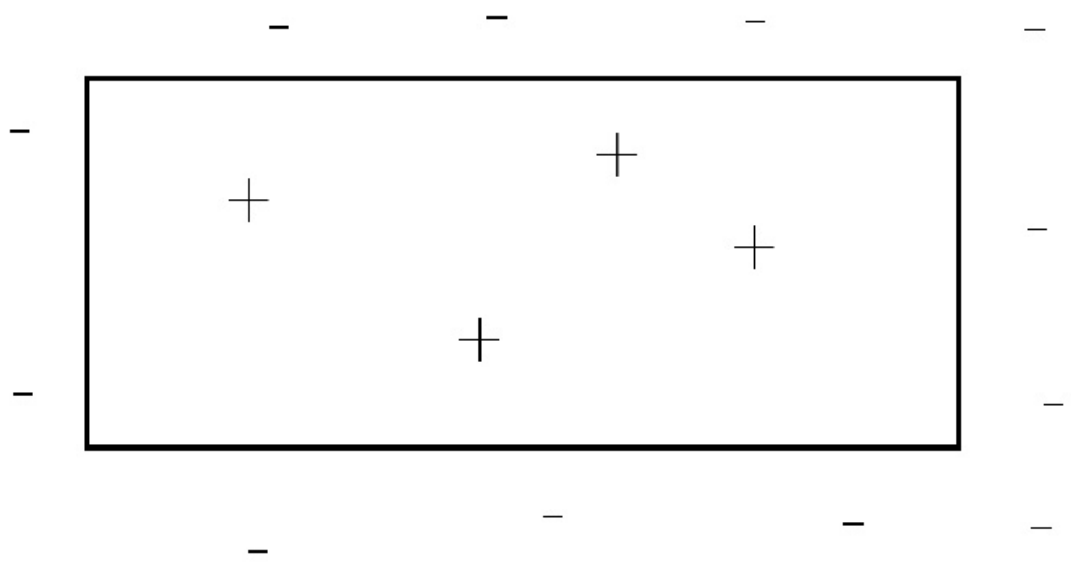
— Niech
  — $\mathcal{X}$ – (skończony lub nieskończony) zbiór obiektów;
  — $\mathbb{C}$ – klasa pojęć w $\mathcal{X}$, tj. $\mathbb{C} = \{f : \mathcal{X} \to \{0, 1\}\}$
  — $c \in \mathbb{C}$ – pojęcie docelowe lub funkcja celu;
— **Dane są**
  — skończona próbka etykietowanych obiektów:

$$D = \{\langle x_1, c(x_1)\rangle, ..., \langle x_m, c(x_m)\rangle\} \in \mathcal{S}(m, c)$$

  gdzie $x_1, ..., x_m \in \mathcal{X}$.
  — przestrzeń hipotez $\mathbb{H} = \{h : \mathcal{X} \to \{0, 1\}\}$;
— **Szukana**
  — hipoteza $h \in \mathbb{H}$ będąca dobrą aproksymacją pojęcia c.
— **Wymagane**
  — dobra jakość aproksymacji

— szybki czas wyuczania.

— Pojęcie: "człowieka o średniej budowie ciała".
— Dane – czyli osoby – są reprezentowane przez ich *wagę*(kg) i *wzrost*(cm) i są etykietowane przez $+$ i $-$.
— Dodatkowa wiedza: szukane pojęcie można wyrazić za pomocą PROSTOKĄTA



Uczenie prostokąta
— $\mathcal{X} = \Re^2$;
— $\mathbb{C} = \mathbb{H} =$ zbiór prostokątów;
— Przykład zbioru treningowego
$((84, 184), +), ((70, 170), +), ((75, 163), -), ((80, 180), +), ((81, 195), -), ((63, 191), -), ((77, 187), -), ((68, 168), +)$
— $((79, 183, ?)$

— **Uczenie półosi (lub dyskretyzacji)**:

$$\mathcal{X} = \Re; \qquad \mathbb{C} = \mathbb{H} = \{[\lambda, \infty) : \alpha \in \Re\}$$

— **Uczenie hiperpłaszczyzny**:

$$\mathcal{X} = \Re^n; \qquad \mathbb{H} = \{f_{w_0,w_1,...,w_n} : \Re^n \to \{0, 1\}|\}$$

gdzie $f_{w_0,...,w_n}(x_1, ..., x_n) = sgn(w_0 + w_1x_1 + ... + w_nx_n)$.
— **Uczenie jednomianów Boolowskich**:

$$\mathcal{X} = \{0, 1\}^n; \qquad c : \{0, 1\}^n \to \{0, 1\};$$

$\mathbb{H} = M_n =$ zbiór jednomianów Boolowskich o n zmiennych.

Błąd rzeczywisty
— $\Omega = (\mathcal{X}, \mu)$ – przestrzeń probabilistyczna na $\mathcal{X}$;
— Błąd hipotezy $h \in \mathbb{H}$ względem funkcji celu c:

$$\boxed{er_\Omega(h, c) = er^c_\Omega(h) = \mu(\mathcal{X}_{h\neq c})}$$

gdzie $\mathcal{X}_{h\neq c} = \{x \in \mathcal{X} : h(x) \neq c(x)\}$.

Statystyka: Jeśli przykłady z D są wybrane zgodnie z miarą prawdopodobieństwa μ w sposób niezależny oraz $|D| \geqslant 30$, to
— $er^c_\Omega(h) \approx er^c_D(h) = \frac{|D \cap \mathcal{X}_{h \neq c}|}{|D|}$,
— z prawdopodobieństwem $(1 - \varepsilon)$

$$|er^c_\Omega - er^c_D| \leqslant s_{\frac{\varepsilon}{2}} \cdot \sqrt{\frac{er^c_D(1 - er^c_D)}{|D|}}$$

9.1.2. Model PAC (probably approximately correct)

Idea modelu PAC (Probably Approximately Correct):
Określenie warunków, przy których uczeń (algorytm uczenia się) z „dużym prawdopodobieństwem" znajdzie „dobrą hipotezę" na podstawie danych D.

PAC-owy uczeń
Niech $\mathfrak{L}$ będzie algorytmem uczenia się, jeśli

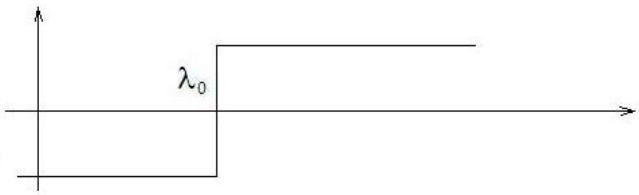
dla każdych $0 < \varepsilon, \delta < 1$, istnieje liczba $m_0 = m_0(\varepsilon, \delta)$ taka, że dla dowolnego pojęcia $c \in \mathbb{C}$, dla dowolnego rozkładu Ω na X i dla $m > m_0$ mamy

$$\mu^m\{D \in \mathcal{S}(m, c) : er_\Omega(\mathfrak{L}(D)) < \varepsilon\} > 1 - \delta$$

Wówczas mówimy w skrócie, że $\mathfrak{L}$ *jest* **PAC** *dla klasy* $\mathbb{C}$ (*"prawdopodobnie aproksymacyjnie poprawny"*).

ε = dopuszczalny poziom błędu; $(1 - \delta)$ = poziom zaufania.

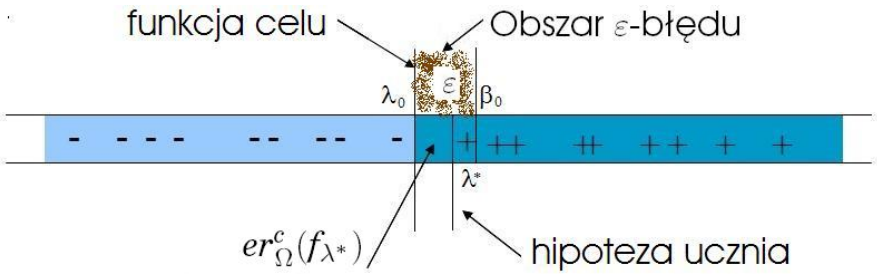
— $\mathbb{H} = \mathbb{C} = \{f_\lambda : \Re \to \{0, 1\} : f_\lambda(x) = 1 \Leftrightarrow x \geqslant \lambda\}$
— $c = f_{\lambda_0}$



— znaleźć λ_0 na podstawie losowo wygenerowanych przykładów $D = \{\langle x_1, f_{\lambda_0}(x_1)\rangle, ..., \langle x_m, f_{\lambda_0}(x_m)\rangle\}$

Algorytm:
1. Set $\lambda^* := \min_{i \in \{1,...,m\}} \{x_i : f_{\lambda_0}(x_i) = 1\}$;
2. $\mathcal{L}(D) := f_{\lambda^*}$;

Twierdzenie:
Powyższy algorytm jest PAC



— $er^c_\Omega(f_{\lambda^*}) = \mu([\lambda_0, \lambda^*))$.
— Niech $\beta_0 = \sup\{\beta : \mu([\lambda_0, \beta)) < \varepsilon\}$.
— Wówczas $er^c_\Omega(f_{\lambda^*}) \geqslant \varepsilon \Leftrightarrow \forall_{x_i \in D} : x_i \notin [\lambda_0, \beta_0]$;
— Stąd

$$\mu^m\{(x_1, ..., x_m) : \forall_{x_i \in D} : x_i \notin [\lambda_0, \beta_0]\} \leqslant (1-\varepsilon)^m$$
$$\mu^m\{D \in \mathcal{S}(m, f_{\lambda_0}) : er_\Omega(f_{\lambda^*}) \leqslant \varepsilon\} \geqslant 1 - (1-\varepsilon)^m$$

— Aby to prawdopodobieństwo było $> 1-\delta$, wystarczy przyjąć $m \geqslant m_0 = \left\lceil \frac{1}{\varepsilon} \ln \frac{1}{\delta} \right\rceil$
— Niech Ω będzie rozkładem dyskretnym zdefiniowanym przez $\mu_1 = \mu(x_1)$, ..., $\mu_n = \mu(x_n)$ – dla pewnych $x_1, ..., x_n \in X$ – takich, że $\mu_1 + ... + \mu_n = 1$. Niech $\varepsilon_{\min} = \min_i \mu_i$.
— Jeśli $\mathfrak{L}$ jest PAC, i jeśli $\varepsilon \leqslant \varepsilon_{\min}$ to warunek $er^c_\Omega(L(D)) < \varepsilon$ jest równoważny z $er^c_\Omega(L(D)) = 0$. Stąd dla każdego δ, istnieje $m_0 = m_0(\varepsilon_{\min}, \delta)$ taka, że dla dowolnego $c \in \mathbb{C}$ i Ω

$$m > m_0 \Rightarrow \mu^m\{D \in \mathcal{S}(m, t) | er_\Omega(L(D)) = 0\} > 1 - \delta$$

— Wówczas mówimy, że prawdopodobnie $\mathfrak{L}$ jest dokładnym algorytmem (*jest PEC – probably exactly correct*)

9.1.3. Wyuczalność klasy pojęć

— Niech $D = \{\langle x_1, c(x_1)\rangle, ..., \langle x_m, c(x_m)\rangle\}$ i niech

$$\boxed{\mathbb{H}^c(D) = \{h \in \mathbb{H} : h|D = c|D\}}$$

zbiór hipotez zgodnych z c na próbce D.
— $\boxed{\mathbb{B}^c_\varepsilon = \{h \in \mathbb{H} : er_\Omega(h) \geqslant \varepsilon\}}$ – zbiór ε-złych hipotez

Definicja: Potencjalna wyuczalność
Mówimy, że $\mathbb{C}$ jest potencjalnie wyuczalna za pomocą $\mathbb{H}$, jeśli dla każdego rozkładu Ω na $\mathcal{X}$ i dowolnego pojęcia $c \in \mathbb{C}$ oraz dla dowolnych $0 < \varepsilon, \delta < 1$ istnieje $m_0 = m_0(\varepsilon, \delta)$ takie, że

$$m \geqslant m_0 \Rightarrow \mu^m\{D \in \mathcal{S}(m, c) : \mathbb{H}^c(D) \cap \mathbb{B}^c_\varepsilon = \emptyset\} > 1 - \delta$$

Algorytm $\mathfrak{L}$ nazywamy niesprzecznym jeśli $\mathfrak{L}(D) \in \mathbb{H}^c(D)$ dla każdego zbioru D.

Twierdzenie
W przestrzeni potencjalnie wyuczalnej, każdy wzorowy uczeń (niesprzeczny algorytm) jest PAC-owy.

Twierdzenie (Haussler, 1988)
Jeśli $\mathbb{C} = \mathbb{H}$ i $|\mathbb{C}| < \infty$, to $\mathbb{C}$ jest potencjalnie wyuczalna.

Dowód: Niech $h \in \mathbb{B}_\varepsilon$ (tzn. $er_\Omega(h) \geqslant \varepsilon$). Wówczas

$$\mu^m\{D \in \mathcal{S}(m, c) : er_D(h) = 0\} \leqslant (1-\varepsilon)^m$$

$$\Rightarrow \mu^m\{D : \mathbb{H}^c(D) \cap \mathbb{B}_\varepsilon \neq \emptyset\} \leqslant |\mathbb{B}_\varepsilon|(1-\varepsilon)^m \leqslant |\mathbb{H}|(1-\varepsilon)^m$$

Aby $|\mathbb{H}|(1-\varepsilon)^m < \delta$ wystarczy wybrać $m \geqslant m_0 = \left\lceil \frac{1}{\varepsilon} \ln \frac{|\mathbb{H}|}{\delta} \right\rceil$

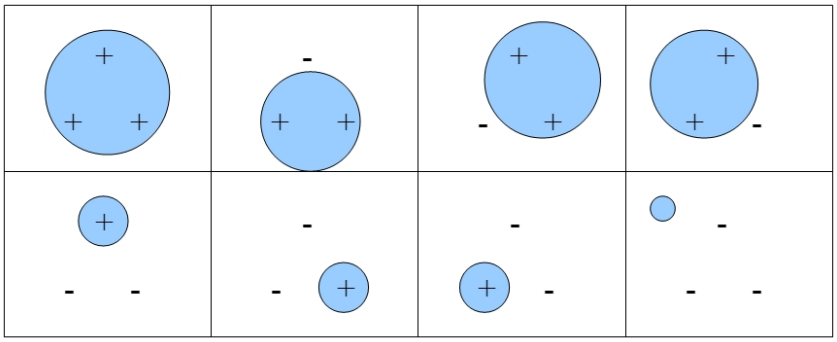
9.2. Wymiar Vapnika-Chervonenkisa

9.2.1. Wymiar Vapnika Chervonenkisa (ang. VC dimension)

— Niech $\overrightarrow{\mathbf{x}} = \langle x_1, ..., x_m \rangle \in \mathcal{X}^m$. Niech

$$\Pi_{\mathbb{H}}(\overrightarrow{\mathbf{x}}) = |\{\langle h(x_1), ..., h(x_m)\rangle \in \{0,1\}^m : h \in \mathbb{H}\}|$$

— $\Pi_{\mathbb{H}}(\overrightarrow{\mathbf{x}})$ jest liczbą podziałów zbioru elementów $\overrightarrow{\mathbf{x}}$ wyznaczonych przez $\mathbb{H}$. Mamy $\Pi_{\mathbb{H}}(\overrightarrow{\mathbf{x}}) \leqslant 2^m$.
— Gdy $\Pi_{\mathbb{H}}(\overrightarrow{\mathbf{x}}) = 2^m$, mówimy, że $\mathbb{H}$ rozbija $\mathbf{x}$.



— Niech $\Pi_{\mathbb{H}}(m) = \max_{\overrightarrow{\mathbf{x}} \in \mathcal{X}^m} \Pi_{\mathbb{H}}(\overrightarrow{\mathbf{x}})$
— **Na przykład:** W przypadku klasy pojęć "półosi" postaci $[\alpha, \infty)$ mamy $\Pi_{\mathbb{H}}(m) = m + 1$.

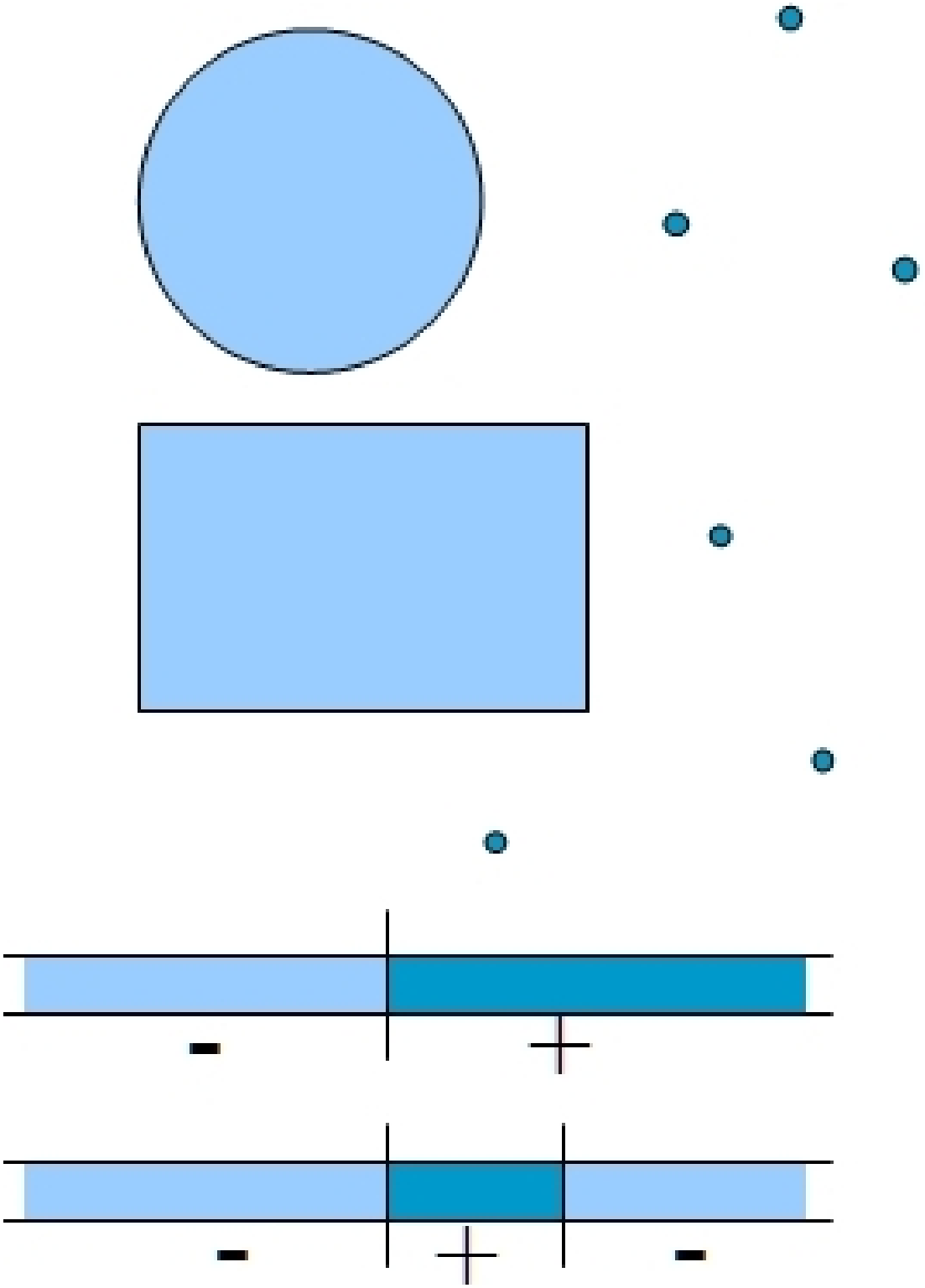
Uwagi:
— Jeśli $\Pi_{\mathbb{H}}(m) = 2^m$, to istnieje pewien zbiór o mocy m taki, że $\mathbb{H}$ może definiować każdy jego podzbiór ($\mathbb{H}$ rozbija ten zbiór).
— Maksymalna wartość m, dla której $\Pi_{\mathbb{H}}(m) = 2^m$ można uważać za siłę wyrażalności przestrzeni $\mathbb{H}$

Definicja: wymiar $VCdim$
Wymiarem Vapnika-Chervonenkisa przestrzeni hipotez $\mathbb{H}$ nazywamy liczbę

$$VCdim(\mathbb{H}) = \max\{m : \Pi_{\mathbb{H}}(m) = 2^m\}$$

gdzie maksimum wynosi ∞ jeśli ten zbiór jest nieograniczony.

— $H = \{$okręgi ... $\} \implies VC(H) = 3$
— $H = \{$prostokąty ... $\} \implies VC(H) = 4$
— $H = \{$funkcje progowe ... $\} \implies$
$VC(H) = 1$ jeśli "+" są zawsze po prawej stronie;
$VC(H) = 2$ jeśli "+" mogą być po obu stronach
— $H = \{$przedziały ... $\} \implies$
VC(H) = 2 jeśli "+" są zawsze w środku
VC(H) = 3 jeśli w środku mogą być zarówno "+" i "-"
— $H = \{$ półpłaszczyzny w $\Re^2$ ... $\} \implies VC(H) = 3$
— czy istnieje H dla której $VC(H) = \infty$?



Dla każdej liczby naturalnej n, niech P_n będzie perceptronem o n wejściach rzeczywistych. Wówczas

$$VCdim(P_n) = n + 1$$

Dowód:

— $\boxed{VCdim(P_n) \leqslant n+1:}$ Wynika z Twierdzenia Radona: Dla dowolnego zbioru E zawierającego $n+2$ punktów w przestrzeni $\mathbb{R}^n$ istnieje niepusty podzbiór $S \subset E$ taki, że

$$conv(S) \cap conv(E \setminus S) \neq \emptyset$$

— $\boxed{VCdim(P_n) \geqslant n+1:}$ Wystarczy wybrać $\mathbf{x} = \{0, e_1, ..., e_n\}$ i pokazać, że każdy jego podzbiór jest definiowany przez jakiś perceptron.

Twierdzenie

1. Jeśli $|\mathbb{H}| < \infty$ to $\boxed{VCdim(\mathbb{H}) \leqslant \log|\mathbb{H}|}$.
2. (**Lemat Sauer'a**) Jeśli $VCdim(\mathbb{H}) = d \geqslant 0$ i $m \geqslant 1$, to

$$\Pi_{\mathbb{H}}(m) \leqslant 1 + \binom{m}{1} + ... + \binom{m}{d} = \Phi(d, m)$$

3. Wniosek: $\Phi(d, m) \leqslant \left(\frac{em}{d}\right)^d \Rightarrow \Pi_{\mathbb{H}}(m) \leqslant \left(\frac{em}{d}\right)^d$
4. Jeśli $|\mathcal{X}| < \infty$, $\mathbb{H} \subset 2^{\mathcal{X}}$ oraz $\mathbb{H} > 1$

$$|\mathcal{X}| < \infty \implies \boxed{VCdim(\mathbb{H}) > \frac{ln|\mathbb{H}|}{1 + ln|\mathcal{X}|}}$$

9.2.2. Podstawowe twierdzenia teorii uczenia się

Twierdzenie: (Warunek konieczny)
Jeśli przestrzeń hipotez ma *nieskończony* wymiar $VCdim$ to *nie jest potencjalnie wyuczalna.*

Twierdzenie: (fundamentalne)
Jeśli przestrzeń hipotez ma skończony wymiar VC, to jest ona potencjalnie wyuczalna.

1. Definiujemy
$$Q_m^{\varepsilon} = \{D \in \mathcal{S}(m, c) : H^c[D] \cap B_{\varepsilon} \neq \emptyset\}$$
2. Szukamy górnego ograniczenia $f(m, \varepsilon)$ dla $\mu^m(Q_m^{\varepsilon})$, które powinno
   - być niezależne od $c \in \mathbb{C}$ i μ (rozkład).
   - dążyć do 0 przy $m \to \infty$
3. **Twierdzenie** Niech $\mathbb{H}$ będzie przestrzenią hipotez określonych na X. Dla dowolnych c, μ, ε (ale ustalonych) mamy
$$\mu^m(Q_m^{\varepsilon}) < 2\Pi_{\mathbb{H}}(2m)2^{-\varepsilon m/2}$$
o ile $m \geqslant 8/\varepsilon$.
4. Korzystamy z lematu Sauer'a, aby pokazać, że $\mu^m(Q_m^{\varepsilon}) < \delta$ dla dostatecznie dużych m.

— Dla skończonych przestrzeni hipotez $\mathbb{H}$ mamy

$$m_L(\mathbb{H}, \delta, \varepsilon) \leqslant \left\lceil \frac{1}{\varepsilon} \ln \frac{|\mathbb{H}|}{\delta} \right\rceil = \left\lceil \frac{1}{\varepsilon}(\ln|\mathbb{H}| + \ln(1/\delta)) \right\rceil$$

— **Twierdzenie** Niech $VCdim(\mathbb{H}) = d \geqslant 1$. Wówczas każdy algorytm niesprzeczny $\mathfrak{L}$ jest PAC oraz wymagana liczba przykładów dla $\mathfrak{L}$ wynosi

$$m_L(\mathbb{H}, \delta, \varepsilon) \leqslant \left\lceil \frac{4}{\varepsilon}\left(d \log \frac{12}{\varepsilon} + \log \frac{2}{\delta}\right)\right\rceil$$

— Dolne ograniczenia:
  — $m_L(\mathbb{H}, \delta, \varepsilon) \geqslant d(1-\varepsilon)$
  — Jeśli $\delta \leqslant 1/100$ i $\varepsilon \leqslant 1/8$, to $m_L(\mathbb{H}, \delta, \varepsilon) > \frac{d-1}{32\varepsilon}$
  — $m_L(\mathbb{H}, \delta, \varepsilon) > \frac{1-\varepsilon}{\varepsilon} \ln \frac{1}{\delta}$

1. Wyuczalność
Kiedy każdy „wzorowy uczeń" będzie PAC-owy?

2. Liczba przykładów
Ile przykładów musi mieć uczeń, by się nauczyć?

Skończoność wymiaru $VCdim()$
1. $VCdim(\mathbb{C}) = d < \infty \Leftrightarrow \mathbb{C}$ jest wyuczalna;
2. Wówczas $L(\frac{1}{\varepsilon}, \frac{1}{\delta}, d) < m(\varepsilon, \delta) < U(\frac{1}{\varepsilon}, \frac{1}{\delta}, d)$

3. Ocena ucznia

$$R(\alpha) = \min_{\alpha \in A} \int Q^c_\Omega(h_\alpha) d\mu$$

na podstawie N losowych przykładów

$$R(\alpha_N) = \min_{\alpha_i \in D} \frac{1}{N} \sum_{i=1}^{N} Q^c(h_{\alpha_i})$$

Kiedy i jak szybko $R(\alpha_N) \to R(\alpha)$?

Skończoność wymiaru $VCdim()$
3 Dla algorytmów typu ERM, $R(\alpha_N) \to R(\alpha)$ szybko.

$$Prob\{R(\alpha) - R(\alpha_N) > \varepsilon\} < e^{-2c\varepsilon^2 N}$$

9.2.3. Appendix: „Nie ma nic za darmo" czyli "Non Free Lunch Theorem"

— Znaleźć optimum nieznanej funkcji $f : S \to W$ ($f \in \mathcal{F}$), gdzie S, W są skończonymi zbiorami.
— Działanie algorytmu przeszukiwania $\mathcal{A}$ dla funkcji f jest identyfikowany z wektorem:

$$V_{\mathcal{A}}(f, t) = \langle (s_1, f(s_1)), (s_2, f(s_2)), ..., (s_t, f(s_t)) \rangle$$

— Ocena algorytmu: $M : \{V_{\mathcal{A}}(f,t) | \mathcal{A}, f, t\} \to \mathbb{R}$;
Np. $M(V_{\mathcal{A}}(f,t)) = \min\{i | f(s_i) = f_{\max}\}$

— **Warunek NFL:** Dla dowolnej funkcji M, i dla dowolnych algorytmów $\mathcal{A}, \mathcal{A}'$

$$\sum_{f \in \mathcal{F}} M(V_{\mathcal{A}}(f, |S|)) = \sum_{f \in \mathcal{F}} M(V_{\mathcal{A}'}(f, |S|))$$

— $\mathcal{F}$ **jest zamknięta wzg. permutacji:** dla dowolnej funkcji $f \in \mathcal{F}$ i dowolnej permutacji $\sigma \in Perm(S)$ mamy $\sigma f \in \mathcal{F}$

Twierdzenie o NFL
— zachodzi równoważność

$$NFL \Leftrightarrow \mathcal{F} \text{ jest zamknięta wzg. permutacji}$$

— Prawdopodobieństwo wylosowania niepustej klasy funkcji zamkniętej wzg. permutacji wynosi:

$$\frac{2^{\binom{|S|+|W|-1}{|S|}} - 1}{2^{|S|^{|W|}} - 1}$$

— Algorytm $\mathfrak{L}$ dobrze się uczy pojęcia c jeśli er^c_Ω jest mały.
— Niech $\mathbb{P}(X) = \{c : X \to \{0, 1\}\}$.
Czy można stwierdzić wiedzieć, że $\mathfrak{L}_1$ uczy się wszystkich pojęć z $\mathbb{P}(X)$ lepiej od $\mathfrak{L}_2$?
— "No Free Lunch theorem" (Wolpert, Schaffer) w wersji problemów uczenia się głosi, że:
  — Żaden algorytm nie może być najlepszy w uczeniu wszystkich pojęć.
  — Każdy algorytm jest najlepszy dla takiej samej liczby pojęć
  — Ale interesuje nas tylko pewna klasa problemów czyli klasa pojęć $\mathbb{C} \subset \mathbb{P}(X)$
  — Wniosek: Należy znaleźć odp. algorytm do każdego problemu.

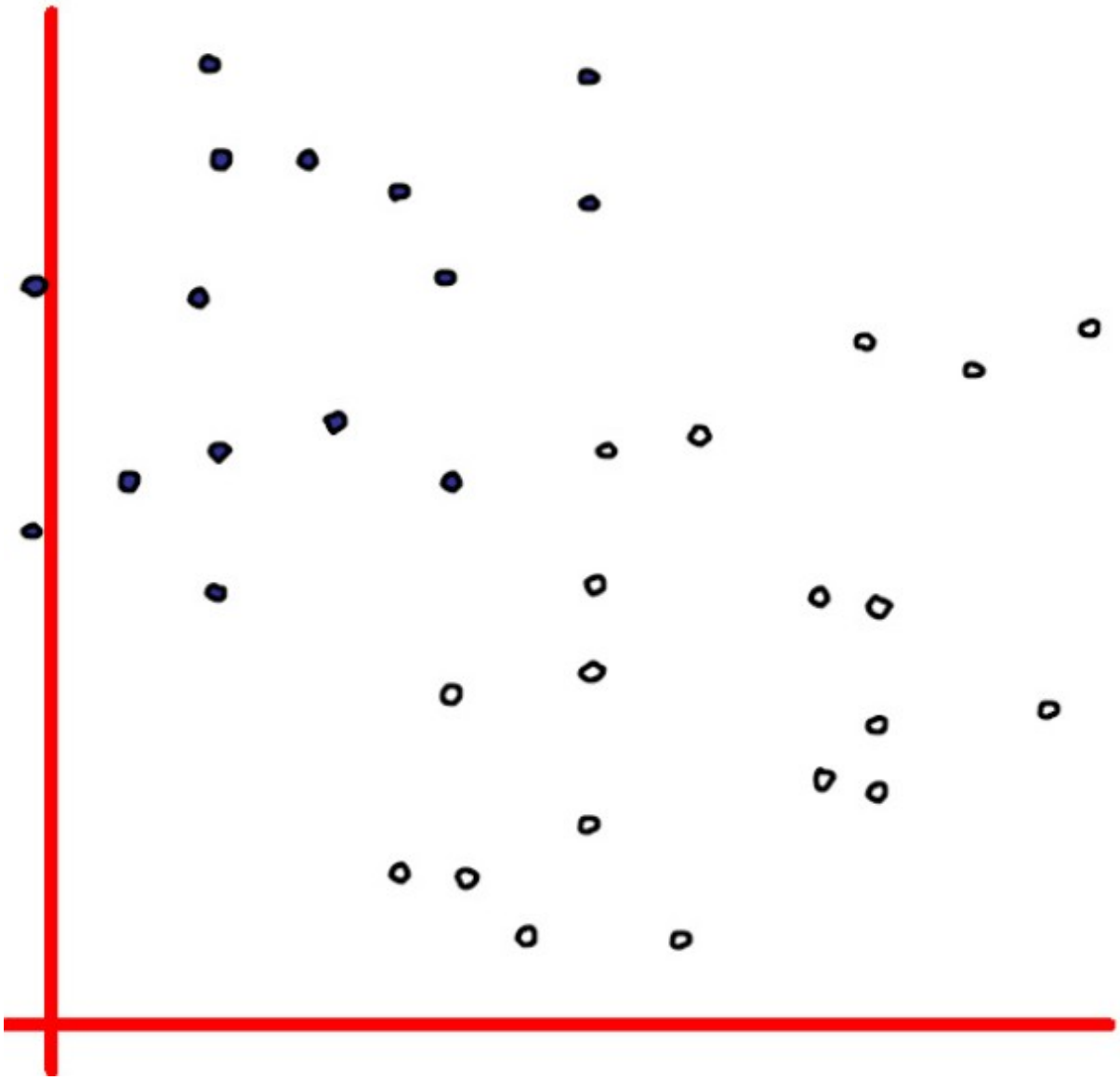
10. SVM: Maszyny wektorów podpierających

10.1. SVM: Maszyny wektorów podpierających

10.1.1. Wprowadzenie

Dana jest próbka treningowa

$$D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\} \text{ gdzie } y_i \in \{-1, 1\},\ \mathbf{x}_i \in \Re^p$$

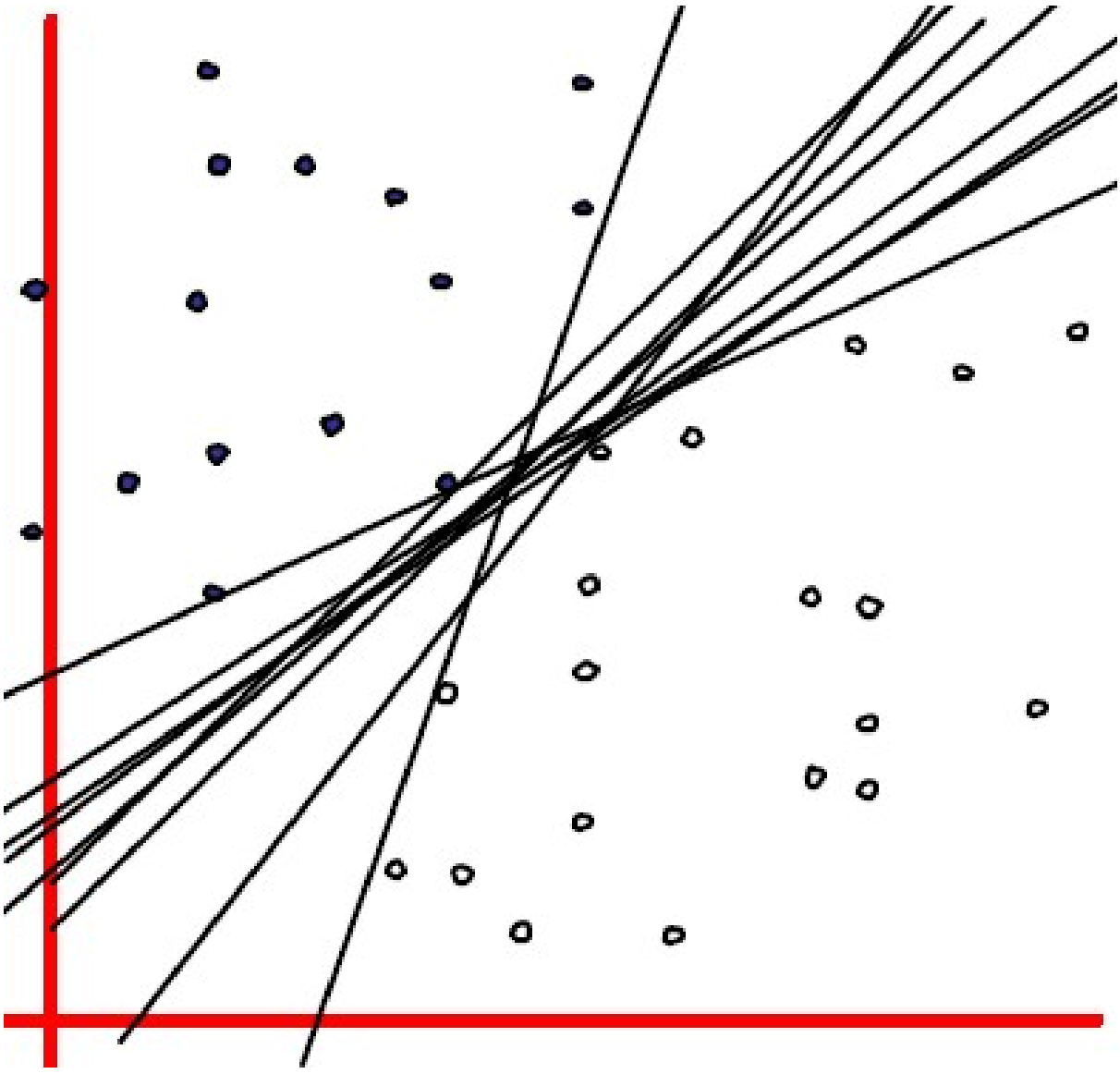


— Klasyfikatorem liniowym nazywamy funkcję

$$f_{\mathbf{w},b}(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

— Próbka D jest liniowo seperowana jeśli istnieje klasyfikator liniowy $f_{\mathbf{w},b}(\ )$ taki, że

$$\mathbf{w} \cdot \mathbf{x}_i + b \geqslant +1, \text{ gdy } y_i = +1$$
$$\mathbf{w} \cdot \mathbf{x}_i + b < -1, \text{ gdy } y_i = -1$$

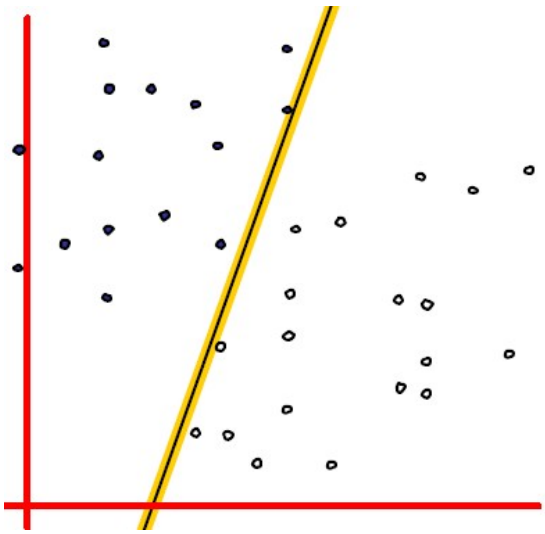


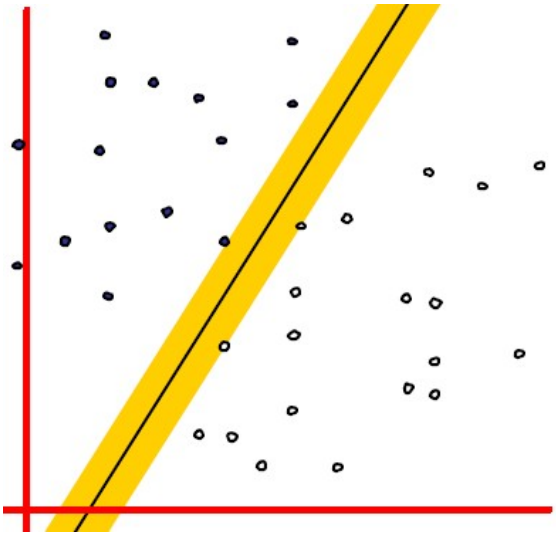
— Uproszczony warunek:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant 1,$$

dla każdego i.
— Każda z tych linii dobrze seperuje punkty;
— Która z nich jest najlepsza?

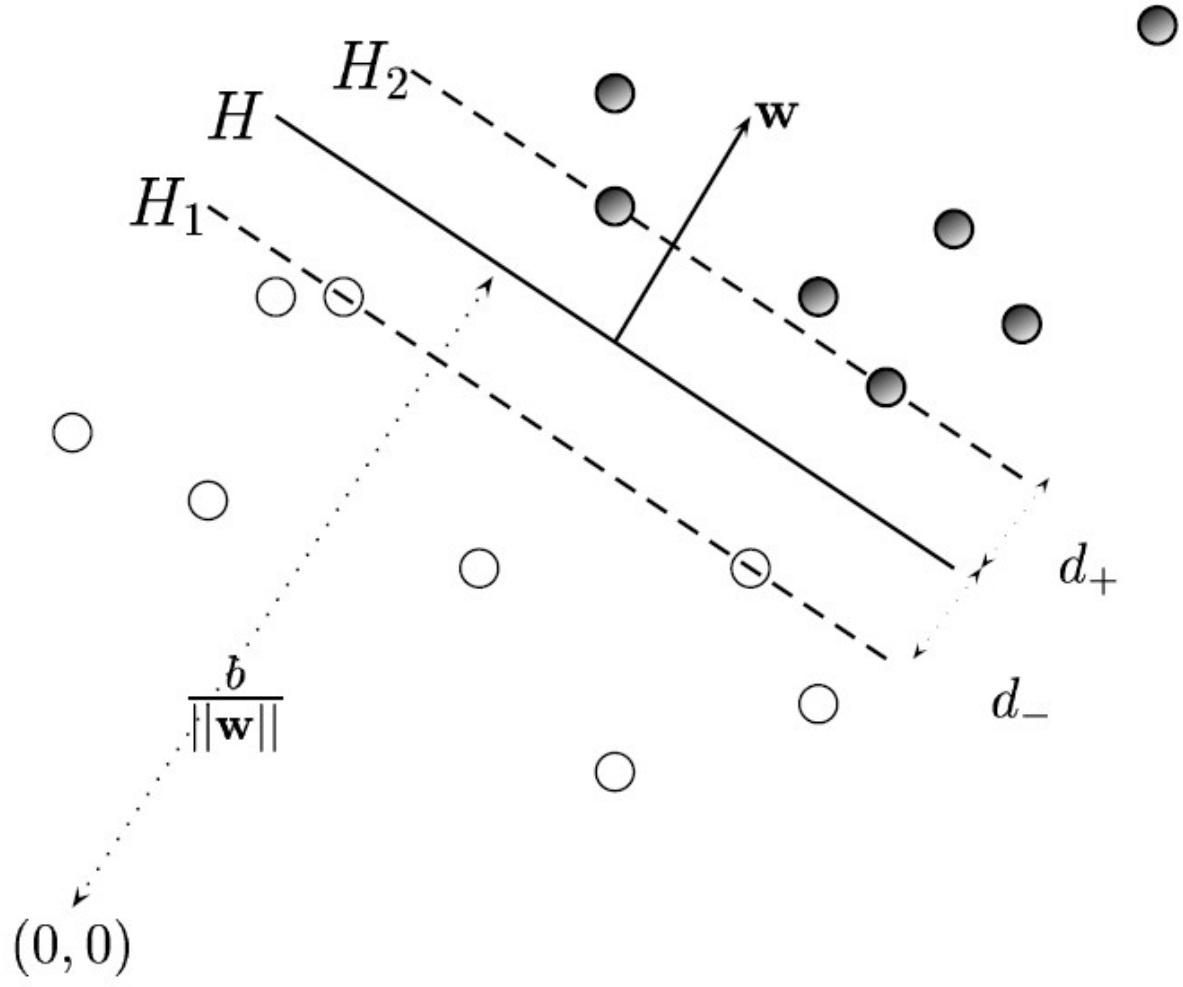
— Margines = szerokość bezpiecznego obszaru po obu stronach hiperpłaszczyzny;
— Margines jest wyznaczony przez 2 hiperpłaszczyzny.
— Chcemy znaleźć klasyfikator z maksymalnym marginesem





— Niech H_1, H_2 będą brzegami marginesu;
— Powiniśmy oprzeć brzegi o punkty z próbki treningowej
— Te punkty nazywamy *wektorami podpierającymi*
— Równanie H_1 i H_2:

$$H_1 : \mathbf{w} \cdot \mathbf{x} + b = 1$$
$$H_2 : \mathbf{w} \cdot \mathbf{x} + b = -1$$



Odległość między H_1 a H_2:

$$d(H_1, H_2) = \frac{2}{||\mathbf{w}||}$$

Maksymalizujemy $d(H_1, H_2)$ lub minimalizujemy $\frac{||\mathbf{w}||}{2}$.

Zadanie LSVM
Znaleźć $\mathbf{w}$ i b które minimalizują

$$\frac{||\mathbf{w}||^2}{2}$$

przy ograniczeniach

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geqslant 0; i = 1, ..., n$$

Q: Jak rozwiązać tego typu zagadnienia?
A: Metodą gradientu?, symulowanego wyżrzania?, odwracanie macierzy? EM? Newton? PROGRAMOWANIE KWADRATOWE?

— Znaleźć

$$\arg\max_{\mathbf{u}} c + \mathbf{d}^T\mathbf{u} + \frac{\mathbf{u}^T R\mathbf{u}}{2}$$

— przy założeniach:

$$A_1\mathbf{u} \leqslant \mathbf{b}_1$$

— oraz

$$A_2\mathbf{u} = \mathbf{b}_2$$

To zagadnienie jest dobrze zbadane i istnieją bardzo efektywne algorytmy rozwiązujące ten problem

— Rozwiązywanie zagadnienia LSVM $\Leftrightarrow$ znalezienie *punktu siodłowego* wielomianu Lagrange'a:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) - \sum_{i=1}^{n} \alpha_i \{[(\mathbf{x}_i \cdot \mathbf{w}) - b]y_i - 1\}$$

gdzie $\alpha = (\alpha_1, ..., \alpha_n)$ jest wektorem nieujemnych współczynników Lagrange'a
— Punkt siodłowy = maksimum funkcji względem $\alpha_i \geqslant 0$ i minimum funkcji względem $\mathbf{w}$ i b.

Twierdzenie Karusha-Kuhna-Tuckera: warunkiem koniecznym istnienia punktu siodlowego jest
— zerowanie się gradientu wzg. $\mathbf{w}$, czyli

$$\mathbf{w} = \sum_{i=1}^{n} y_i\alpha_i\mathbf{x}_i \tag{10.1}$$

— zerowanie się pochodnej wzg. b, czyli

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{10.2}$$

— spełnienie warunku:

$$\alpha_i \{[(\mathbf{x}_i \cdot \mathbf{w}_0) - b_0]y_i - 1\} = 0, \text{ for } i = 1, ..., n \tag{10.3}$$

Po uwzględnieniu tych warunków (na istnienie punktu siodłowego) mamy nowy problem optymalizacyjny:
— Znaleźć wektor α będący maksimum funkcji

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{ij=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \tag{10.4}$$

— przy ograniczeniach

$$\alpha_i \geqslant 0, i = 1, .., n \quad oraz \quad \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{10.5}$$

Niech wektor $(\alpha_1^0, ..., \alpha_n^0)$ będzie rozwiązaniem maksymalizującym (4) przy ograniczeniach (5).
— Z (3) wynika, że jeśli $\mathbf{x}_i$ nie jest wektorem podpierającym to $\alpha_i^0 = 0$;
— Równania (4) i (5) odbywają się faktycznie tylko po wektorach podpierających.
— Hyperpłaszczyzna rozdzielająca ma postać $\mathbf{w}_0 \mathbf{x} - b_0 = 0$ gdzie

$$\mathbf{w}_0 = \sum_{\text{wektory podp.}} y_i \alpha_i^0 \mathbf{x}_i \quad b_0 = \frac{1}{2}[(\mathbf{w}_0 \cdot \mathbf{x}^{(1)}) + (\mathbf{w}_0 \cdot \mathbf{x}^{(-1)})]$$

tu $\mathbf{x}^{(1)}$ i $\mathbf{x}^{(-1)}$ są dowolnymi wektorami podpierającymi z każdej z klas.

Ostateczna funkcja decyzyjna:

$$f(\mathbf{x}) = \text{sgn}(\sum_{\text{wektory podp.}} y_i \alpha_i^0 (\mathbf{x}_i \cdot \mathbf{x}) + b_0) \tag{10.6}$$

10.1.2. Brak liniowej separowalności danych

Nieznaczna nieseparowalność

— Założenie o separowalności klas jest nienaturalna;
— Modyfikacja:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geqslant 1 - \xi_i, \text{ gdy } y_i = +1$$
$$\mathbf{w} \cdot \mathbf{x}_i + b \leqslant -1 + \xi_i, \text{ gdy } y_i = -1$$

gdzie stałe ξ_i spełniają warunki:

$$\xi_i \geqslant 0, i = 1, ..., n.$$

— Zmodyfikowana funkcja do minimalizacji

$$\frac{||\mathbf{w}||^2}{2} + C \sum_{i=1}^{n} \xi_i$$

C jest stałą "karą" za niespełnienie idealnych warunków.

Ogólne zadanie SVM
Znaleźć $\mathbf{w}$ i b które minimalizują

$$\frac{||\mathbf{w}||^2}{2} + C \sum_{i=1}^{n} \xi_i$$

przy ograniczeniach

$$\begin{aligned}&\mathbf{w}\cdot\mathbf{x}_i+b\geqslant 1-\xi_i,\ \text{gdy } y_i=+1\\&\mathbf{w}\cdot\mathbf{x}_i+b\leqslant -1+\xi_i,\ \text{gdy } y_i=-1\\&\xi_i\geqslant 0, i=1,...,n.\end{aligned}$$

Można pokazać, że zarówno i w tym przypadku, możemy sprowadzić do problemu

— Znaleźć wektor $(\alpha_1^0,...,\alpha_n^0)$ będący maksimum funkcji

$$W(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{ij=1}^{n}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j) \tag{10.7}$$

— przy ograniczeniach

$$C\geqslant\alpha_i\geqslant 0, i=1,..,n\quad oraz\quad \sum_{i=1}^{n}\alpha_i y_i=0 \tag{10.8}$$

A następnie stosować funkcję decyzyjną:

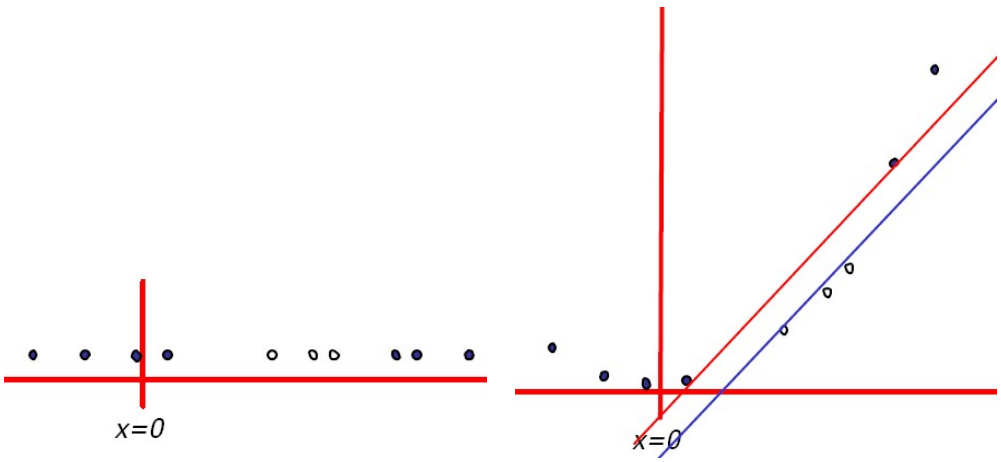
$$f(\mathbf{x})=\text{sgn}\left(\sum_{\text{wektory podp.}} y_i\alpha_i^0(\mathbf{x}_i\cdot\mathbf{x})+b_0\right) \tag{10.9}$$

Zmiana przetrzeni atrybutów

— Przekształcimy dane do bogatszej przetrzeni cech:

$$\phi:\Re^p\to\mathcal{F}\subseteq\Re^N,\quad N\gg p$$

— Wystarczy zamienić iloczyny skalarne $(\mathbf{x_i}\cdot\mathbf{x_j})$ we wszystkich wzorach na $(\phi(\mathbf{x_i})\cdot\phi(\mathbf{x_j}))$



— PROBLEM OBLICZENIOWY: czas wykonania iloczynu skalarnego $(\phi(\mathbf{x_i})\cdot\phi(\mathbf{x_j}))$ wynosi $O(N^2)$

— TRIK: Kernel functions

$$K((\mathbf{x_i}\cdot\mathbf{x_j}))=(\phi(\mathbf{x_i})\cdot\phi(\mathbf{x_j}))$$

— Zmieniona funkcja:

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{ij=1}^{n} \alpha_i \alpha_j y_i y_j (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j))$$
$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{ij=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix} \begin{matrix} 1 \\ + \\ \sum_{i=1}^{m} 2a_ib_i \\ + \\ \sum_{i=1}^{m} a_i^2 b_i^2 \\ + \\ \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2a_ia_jb_ib_j \end{matrix}$$

$$\begin{aligned} (\mathbf{a}.\mathbf{b}+1)^2 &= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1 \\ &= \left(\sum_{i=1}^{m} a_ib_i \right)^2 + 2\sum_{i=1}^{m} a_ib_i + 1 \\ &= \sum_{i=1}^{m} \sum_{j=1}^{m} a_ib_ia_jb_j + 2\sum_{i=1}^{m} a_ib_i + 1 \\ &= \sum_{i=1}^{m} (a_ib_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_ib_ia_jb_j + 2\sum_{i=1}^{m} a_ib_i + 1 \end{aligned}$$

10.1.3. Implementacja

Problem

Wejście: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$; K: funkcja jądrowa.
Wyjście: wektor $(\alpha_1^0, ..., \alpha_n^0)$ będący maksimum funkcji

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{ij=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i \cdot \mathbf{x}_j)$$

przy ograniczeniach

$$C \geqslant \alpha_i \geqslant 0, i = 1, .., n \quad oraz \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial W(\alpha)}{\partial \alpha_i} = 1 - y_i \sum_{j=1}^{l} \alpha_j y_j K(x_i, x_j)$$

$$\alpha_i \leftarrow \alpha_i + \eta \frac{\partial W(\alpha)}{\partial \alpha_i} \qquad \eta \text{ = learning rate}$$

$$parameter\ \omega \in (0,2)$$

$$\alpha_i \leftarrow \alpha_i + \frac{\omega}{K(x_i, x_i)} \left(1 - y_i \sum_{j=1}^{l} \alpha_j y_j K(x_i, x_j)\right)$$

Given Training set S and learning rate $\eta \in R^+$

$\alpha \leftarrow 0$

Repeat

for all train set $i = 1 \ to \ l$

update α_i

if

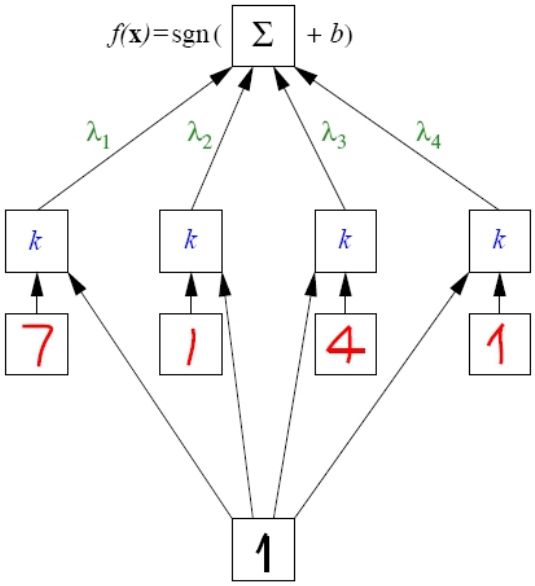
$\alpha < 0$, *then* $\alpha \leftarrow 0$

$\alpha > C$, *then* $\alpha \leftarrow C$

End for

Until stop criterion satisfied

return α



classification

$f(\mathbf{x}) = \mathrm{sgn}\left(\sum \lambda_i k(\mathbf{x}, \mathbf{x}_i) + b\right)$

weights

comparison: $k(\mathbf{x}, \mathbf{x}_i)$, e.g. $k(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^d$

$k(\mathbf{x}, \mathbf{x}_i) = \exp(-\|\mathbf{x} - \mathbf{x}_i\|^2 / c)$

support vectors $\mathbf{x}_1 \ldots \mathbf{x}_4$

$k(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{x}_i) + \theta)$

input vector $\mathbf{x}$

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum_{\text{wektory podp.}} y_i \alpha_i^0 K(\mathbf{x}_i \cdot \mathbf{x}) + b_0\right)$$

11. Metoda wzmacnienia klasyfikatorów (ang. Boosting)

11.1. Metoda wzmacnienia klasyfikatorów (ang. Boosting)

11.1.1. Wstęp

Boosting jest ogólną metodą służącą zwiększeniu skuteczności dowolnego algorytmu uczenia.

Idea
Budowanie "mocnego i złożonego klasyfikatora" ze "słabych i prostych klasyfikatorów".

— Leslie Valiant i Michael Kearns byli pierwszymi, którzy pokazali, że "słabe" algorytmy uczące, których skuteczność jest nawet niewiele lepsza niż losowe zgadywanie, mogą być wykorzystane do stworzenia dowolnie skutecznego "silnego" klasyfikatora.
— Robert Schapire jako pierwszy przedstawił algorytm wzmacniania działający w czasie wielomianowym.

— Yoav Freund zaproponował znacznie bardziej efektywny algorytm wzmacniania, który mimo tego, że był w pewnym sensie optymalny, miał pewne istotne w praktyce wady.
— Pierwsze eksperymenty z tymi wczesnymi algorytmami wzmacniania zostały przeprowadzone przez zespół Drucker, Schapire, Simard i dotyczyły zadania OCR (ang. optical character recognition), w którym sieci neuronowe były użyte jako "proste" klasyfikatory.
— Freund i Schapire przedstawili algorytm **AdaBoost**, który rozwiązał wiele praktycznych trudności wcześniejszych algorytmów wzmacniania.

11.1.2. AdaBoost - opis algorytmu

— Algorytm na wejściu otrzymuje zbiór treningowy $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$, gdzie każdy x_i należy do pewnej dziedziny problemu X, natomiast każda etykieta (decyzja) y_i należy do pewnego zbioru Y. Dla ułatwienia będziemy na razie zakładać, że $Y = \{-1, +1\}$.
— **AdaBoost** (Adaptive Boosting) wywołuje wybrany "słaby" algorytm uczący w serii T iteracji. Zakładamy, że błąd uzyskiwanych klasyfikatorów na zbiorze treningowym jest mniejszy niż $\frac{1}{2}$.
— Jedną z głównych idei algorytmu jest strojenie rozkładu (lub wag elementów) dla zbioru treningowego. Wagę i-tego elementu ze zbioru treningowego w iteracji t będziemy oznaczali przez $D_t(i)$.

— Początkowo wszystkie wagi są ustawione na równe wartości;
— Po każdej iteracji, wagi elementów źle klasyfikowanych są zwiększane. Dzięki temu mamy możliwość skierowania uwagi "słabego" klasyfikatora na pewne elementy (trudne do wyuczenia) ze zbioru treningowego.
— Zadaniem "słabego" algorytmu uczącego jest zbudowanie klasyfikatora (ang. hypothesis) $h_t : X \to Y$ odpowiedniego dla aktualnego rozkładu D_t.

— Skuteczność takiego klasyfikatora jest mierzona przez jego błąd (z uwzględnieniem rozkładu D_t):

$$\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i)\neq y_i} D_t(i)$$

— W praktyce "słaby" algorytm uczący może być algorytmem, który uwzględnia rozkład D_t. Nie jest to jednak konieczne. Kiedy algorytm nie pozwala na bezpośrednie uwzględnienie rozkładu D_t, losuje się (względem D_t) podzbiór zbioru treningowego, na którym następnie wywołuje się algorytm uczący.
— Kiedy AdaBoost dostaje klasyfikator h_t dobierany jest parametr α_t. Intuicyjnie α_t odpowiada za wagę jaką przykładamy do klasyfikatora h_t. Zauważmy, że $\alpha_t \geqslant 0$ gdy $\varepsilon_t \leqslant \frac{1}{2}$. Ponadto α_t rośnie kiedy ε_t maleje.
— Rozkład D_t jest następnie zmieniany tak, aby zwiększyć (zmniejszyć) wagi elementów zbioru treningowego, które są źle (dobrze) klasyfikowane przez h_t. Stąd wagi mają tendencję do skupiania się na "trudnych" przykładach.

Dane: $(x_1, y_1), \ldots, (x_m, y_m)$, gdzie $x_i \in X$, $y_i \in Y = \{-1, +1\}$ **Inicjalizacja:** $D_1(i) = \frac{1}{m}$ dla $i = 1, \ldots, m$ **for** $t = 1, \ldots, T$ **do**:
— Wykorzystując "słaby" algorytm uczący zbuduj klasyfikator $h_t : X \to \{-1, +1\}$ (uwzględniając rozkład D_t).
— $\varepsilon_t = Pr_{D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(i)\neq y_i} D_t(i)$
— $\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
— Uaktualnij wagi elementów zbioru treningowego:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{jeśli } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{jeśli } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{1}{Z_t} \times D_t(i) \times e^{-\alpha_t y_i h_t(x_i)} \end{aligned}$$

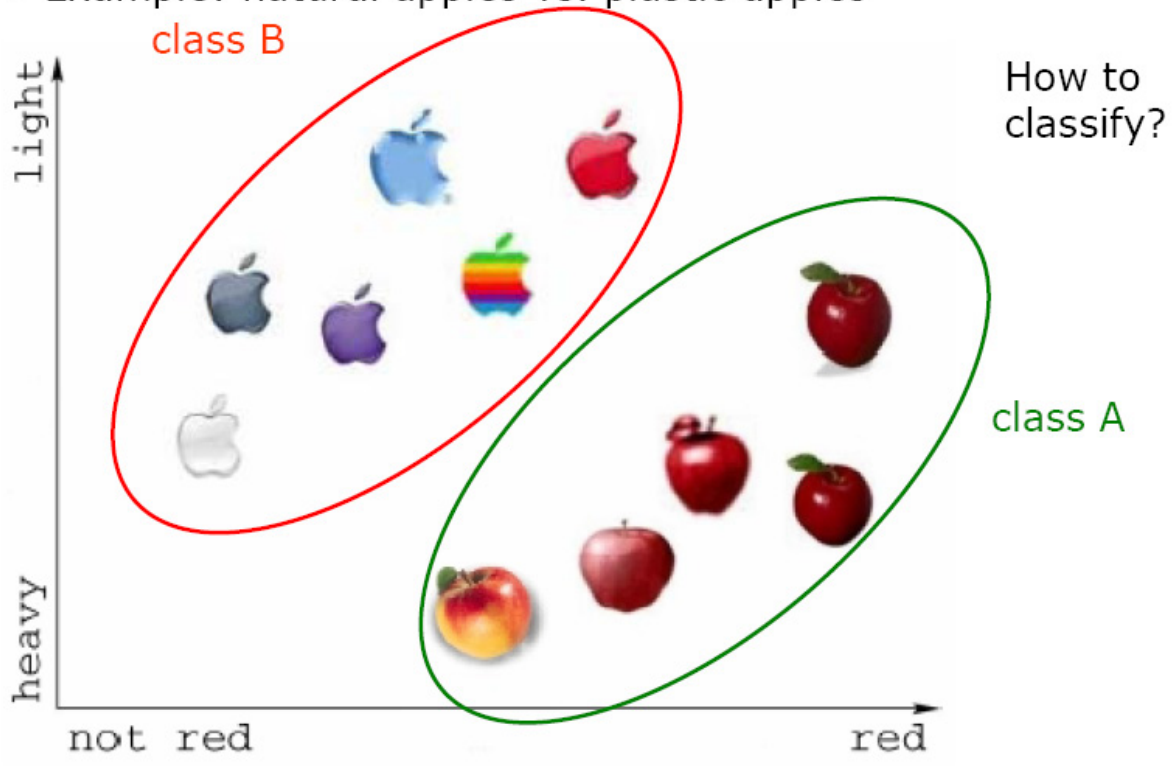
Z_t jest czynnikiem normalizującym (wybranym tak, aby D_{t+1} było rozkładem).

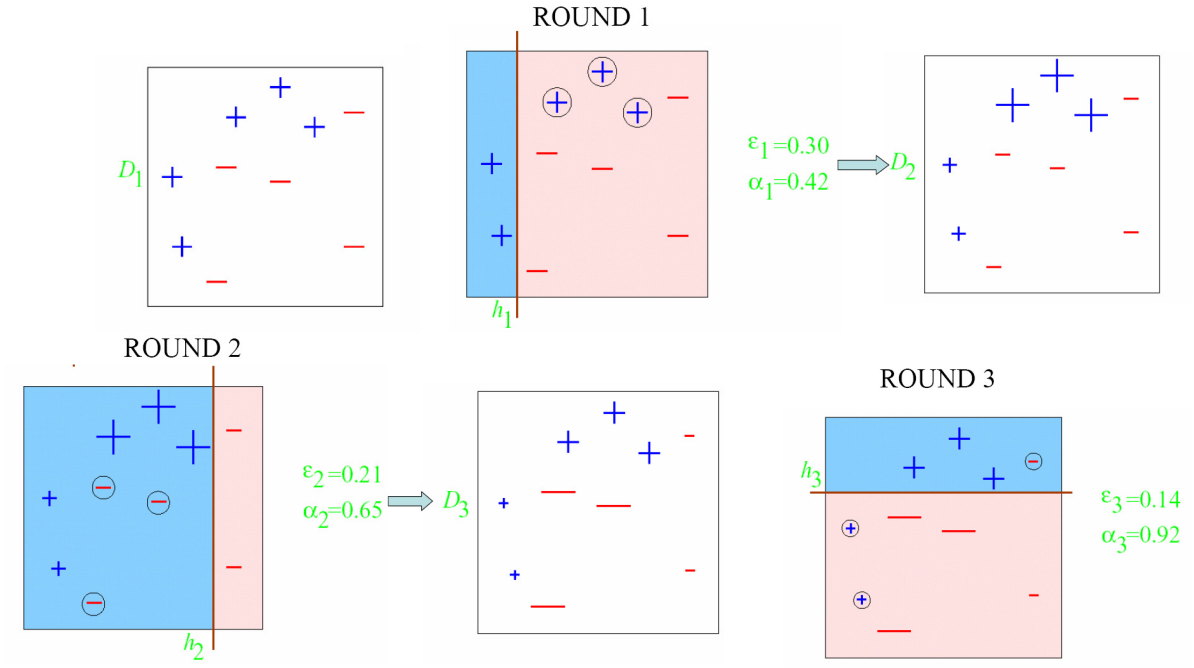
Wynikowy klasyfikator powstaje za pomocą ważonego głosowania klasyfikatorów h_t, gdzie α_t jest wagą przypisaną klasyfikatorowi h_t.

$t =$	1	2	...	T
słabe klasyfikatory	h_1	h_2	...	h_T
wagi	α_1	α_2	...	α_T

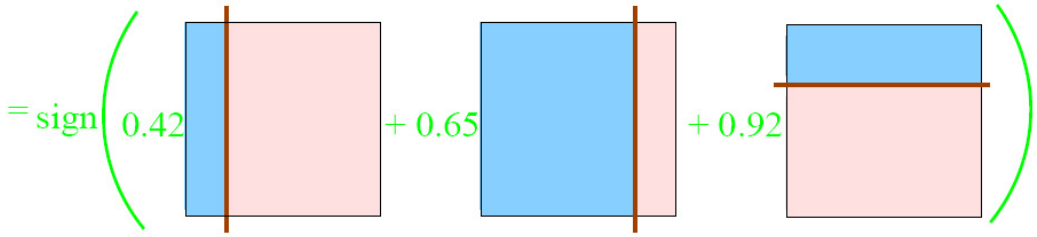
$$H(x) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) = \operatorname{argmax}_{i\in\{-1,+1\}} \left(\sum_{\substack{t \in \{1,\ldots,T\} \\ h(x)=i}} \alpha_t \right).$$

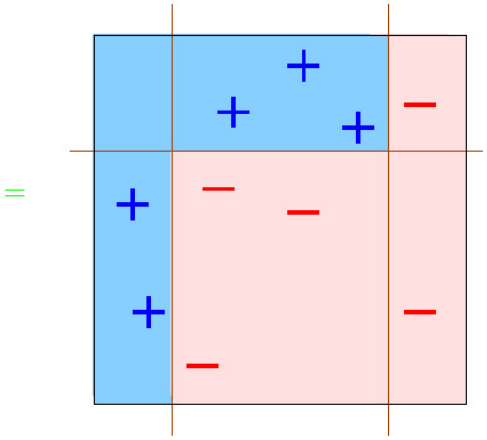
- Example: natural apples vs. plastic apples

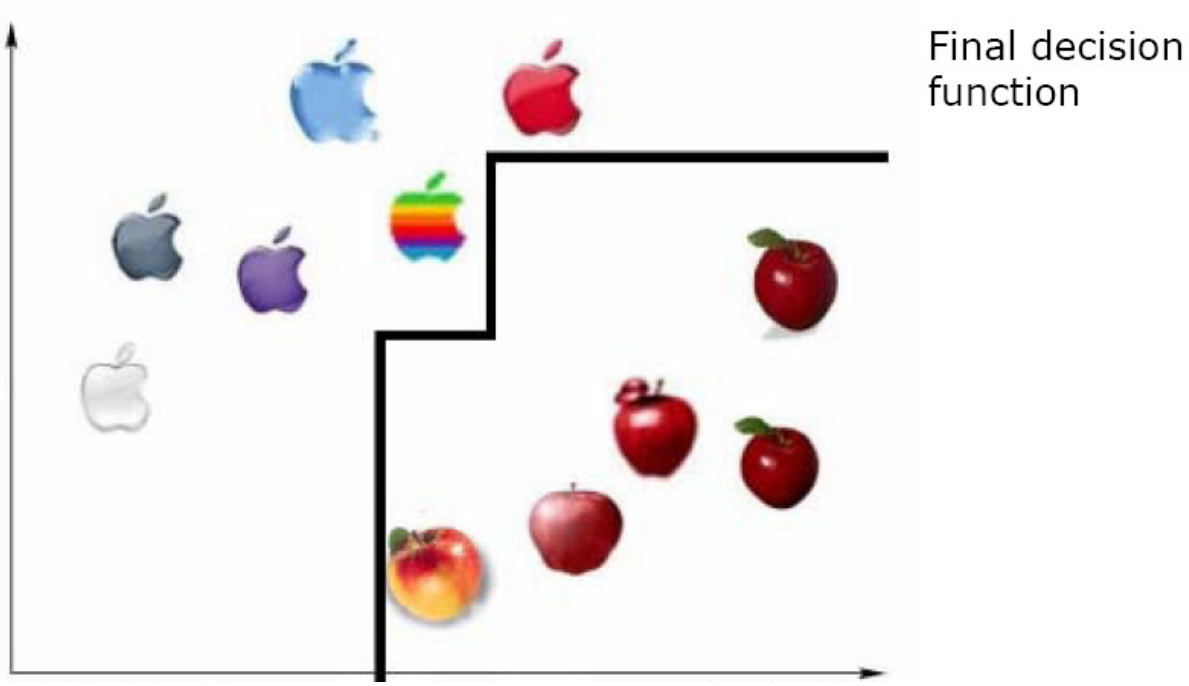



ROUND 1
D_1
h_1
ε_1=0.30
α_1=0.42
D_2
ROUND 2
h_2
ε_2=0.21
α_2=0.65
D_3
ROUND 3
h_3
ε_3=0.14
α_3=0.92

$H_{final} =$







11.1.3. AdaBoost - błąd na zbiorze treningowym

Najbardziej podstawowa własność algorytmu AdaBoost to jego zdolność do zredukowania błędu na zbiorze treningowym. Schapire i Singer pokazali, że błąd wynikowego klasyfikatora na zbiorze treningowym spełnia nierówność:

$$\frac{1}{m}|i : H(x_i) \neq y_i| \leqslant \frac{1}{m}\sum_i e^{-y_i f(x_i)} = \prod_t Z_t, \tag{11.1}$$

gdzie $f(x) = \sum_t \alpha_t h_t(x)$, czyli $H(x) = sign(f(x))$. Nierówność bierze się stąd, że

$$\text{jeżeli } (H(x_i) \neq y_i) \text{ to } (e^{-y_i f(x_i)} \geqslant 1).$$

Udowodnimy teraz równość:

$$\frac{1}{m}\sum_i e^{-y_i f(x_i)} = \prod_t Z_t.$$

Pamiętamy definicję:

$$D_{t+1} = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}.$$

Stąd

$$\frac{Z_t D_{t+1}(i)}{D_t(i)} = e^{-\alpha_t y_i h_t(x_i)}.$$

$$\begin{aligned}
\frac{1}{m}\sum_i e^{-y_i f(x_i)} &= \frac{1}{m}\sum_i e^{-y_i \sum_t \alpha_t h_t(x_i)} \\
&= \frac{1}{m}\sum_i \prod_t \exp(-y_i \alpha_t h_t(x_i)) \\
&= \frac{1}{m}\sum_i \prod_t \frac{Z_t D_{t+1}(i)}{D_t(i)} \\
&= \frac{1}{m}\sum_i \frac{Z_1 D_2(i)}{D_1(i)} \frac{Z_2 D_3(i)}{D_2(i)} \cdots \frac{Z_T D_{T+1}(i)}{D_T(i)} \\
&= \frac{1}{m}\prod_t Z_t \sum_i \frac{D_{T+1}(i)}{D_1(i)} = \prod_t Z_t
\end{aligned}$$

Oszacowanie (11.1) sugeruje (strategię zachłanną), że błąd na zbiorze treningowym może być redukowany poprzez wybieranie takich α_t i h_t, aby w każdej iteracji minimalizować:

$$Z_t = \sum_i D_t(i)e^{-\alpha_t y_i h_t(x_i)}.$$

Wybierając

$$\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t}),$$

dostajemy:

$$\begin{aligned}
Z_t &= \sum_i D_t(i)e^{-\alpha_t y_i h_t(x_i)} \\
&= \sum_{i:y_i = h_t(x_i)} D_t(i)e^{-\alpha_t} + \sum_{i:y_i \neq h_t(x_i)} D_t(i)e^{\alpha_t} \\
&= (1-\varepsilon_t)e^{-\alpha_t} + \varepsilon_t e^{\alpha_t} \\
&= (1-\varepsilon_t)\sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \varepsilon_t\sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \\
&= 2\sqrt{\varepsilon_t(1-\varepsilon_t)}
\end{aligned}$$

Mamy zatem:

$$\begin{aligned}\prod_t Z_t &= \prod_t [2\sqrt{\varepsilon_t(1-\varepsilon_t)}] \\ &= \prod_t [1-4\gamma_t^2] \\ &\leqslant e^{-2\sum_t \gamma_t^2},\end{aligned}$$

gdzie $\varepsilon_t = \frac{1}{2} - \gamma_t$. Ograniczenie to jako pierwsi pokazali Freund i Schapire. Stąd, jeśli "słaby" algorytm produkuje klasyfikatory, dla których γ_t jest większe od zera, błąd na zbiorze treningowym spada wykładniczo szybko do zera.

Jakkolwiek jest to miła własność nie zawsze będziemy nią zainteresowani. Zbieżność błędu na zbiorze treningowym do zera może powodować zbytnie dopasowanie wynikowego klasyfikatora do danych treningowych (ang. overfitting).

11.1.4. AdaBoost - błąd uogólnienia

Freund i Schapire pokazali jak można oszacować ogólny błąd wynikowego klasyfikatora w terminach błędu na zbiorze treningowym, mocy zbioru treningowego m, wymiaru Vapnika-Chervonenkisa d dla klasy "słabych" klasyfikatorów oraz ilości iteracji algorytmu wzmacniania T. Pokazali oni, że z dużym prawdopodobieństwem ogólny błąd wynikowego klasyfikatora jest nie równy co najwyżej:

$$\hat{Pr}[H(x) \neq y] + O(\sqrt{\frac{Td}{m}}),$$

gdzie $\hat{Pr}[\cdot]$ oznacza empiryczne prawdopodobieństwo na zbiorze treningowym.

Wymiar Vapnika-Chervonenkisa mierzy "umiejętności" algorytmów klasyfikacji. Jest on zdefiniowany jako największa liczba punktów jakie algorytm może "poszatkować", "rozbić" w dowolny sposób. Przykłady...

Ograniczenie to sugeruje, że wynikowy klasyfikator (przy zbyt dużej liczbie iteracji) może być zbytnio dopasowany do danych treningowych. W praktyce czasami się to zdarza. Jednakże przeprowadzone przez kilku autorów eksperymenty pokazały empirycznie, że zbytnie dopasowanie nie występuje zbyt często, nawet przy dużej liczbach iteracji ustawianych na duże wartości. Co więcej, obserwowano przypadki, w których AdaBoost zmniejszał ogólny błąd długo po tym jak błąd na zbiorze treningowym osiągnął zero.

11.1.5. Podsumowanie

W praktyce algorytm AdaBoost ma sporo zalet:

— Jest szybki, prosty i łatwy w implementacji.

— Poza ilością iteracji T, nie potrzebuje ustalania żadnych parametrów.

— Nie potrzebuje wcześniejszej wiedzy na temat "słabego" algorytmu uczenia, który wykorzystuje. Co za tym idzie, może być elastycznie połączony z dowolną metodą szukającą "słabych" klasyfikatorów.

— Miłą własnością algorytmu AdaBoost jest jego zdolność do wykrywania przypadków nietypowych. Ponieważ AdaBoost zwiększa wagi elementów źle klasyfikowanych. Elementy, na których skupiony został rozkład, te z największymi wagami, możemy uważać za przypadki nietypowe.

Z drugiej strony można jednak dostrzec kilka wad:

— Faktyczne osiągi wzmacniania dla konkretnego problemu są wyraźnie zależne zarówno od danych jak i zastosowanego "słabego" algorytmu.
— Wzmacnianie może nie przynieść oczekiwanego skutku jeśli algorytm nie dostanie dostatecznej ilości danych, zastosujemy zbyt skomplikowany lub zbyt prosty "słaby" algorytm uczenia.
— Wydaje się być szczególnie wrażliwy na szumy w danych.

12. Ukryty model Markowa

13. Sterowanie rozmyte

13.1. Sterowanie rozmyte

13.1.1. Wprowadzenie

W odróżnieniu od ograniczonego i uporządkowanego języka, który wykorzystujemy do opisywania pojęć, obiektów i relacji podczas tworzenia modeli komputerowych, pojęcia i obiekty w świecie rzeczywistym są opisywane w znacznie mniej restrykcyjny sposób.

Weźmy na przykład takie zdanie z codziennego języka:

Jaś jest wysoki

które może mieć wiele interpretacji (znaczeń), zależnie od perspektywy, miejsca zamieszkania (znaczenie słowa **wysoki** jest nieco inne np. w Japonii) itp.

Lecz jeśli chcemy wprowadzić dane Jasia do komputera, musimy ustalić, że ma on dokładnie 187 cm wzrostu.

A co jeśli nie znamy dokładnego wzrostu Jasia?

W realnym świecie radzimy sobie doskonale ze stwierdzeniami w rodzaju:

*Za **około** 40 minut powinniśmy dojechać na lotnisko o ile ruch uliczny nie będzie **zbyt duży**.*

Co jednak zrobić, aby takie stwierdzenie zrozumiał komputer? Jak przedstawić **około** i **zbyt duży** w systemie komputerowym?

W 1965 roku Lotfi A. Zadeh zaproponował inny sposób opisywania pojęć takich jak: zbiór, podzbiór, zawieranie. Jego celem było umożliwienie wykorzystania w systemach komputerowych obiektów (zbiorów), które ze swojej natury są niedoprecyzowane lub niejednoznaczne - czyli **rozmyte** (ang. fuzzy).

Ponownie rozważmy przykład zdania z języka naturalnego:

Jaś jest wysoki

Jeśli skądinąd wiemy, że Jaś ma 175 cm wzrostu to możemy zacząć zastanawiać czy powyższe stwierdzenie jest prawdziwe. W klasycznym (dokładnym) modelu musimy zerojedynkowo rozstrzygnąć czy 175 cm kwalifikuje Jasia do zbioru osób wysokich czy nie. W modelu rozmytym możemy być bardziej subtelni i próbować wyrazić w jakim stopniu 175 cm czyni z Jasia osobę wysoką.

Zbiory rozmyte

W klasycznej teorii mnogości każdy podzbiór A w pewnej przestrzeni X można utożsamić z jego funkcją charakterystyczną określoną jako:

$$\chi_A(x) = \begin{cases} 1 & \text{gdy} \quad x \in A \\ 0 & \text{gdy} \quad x \notin A \end{cases}$$

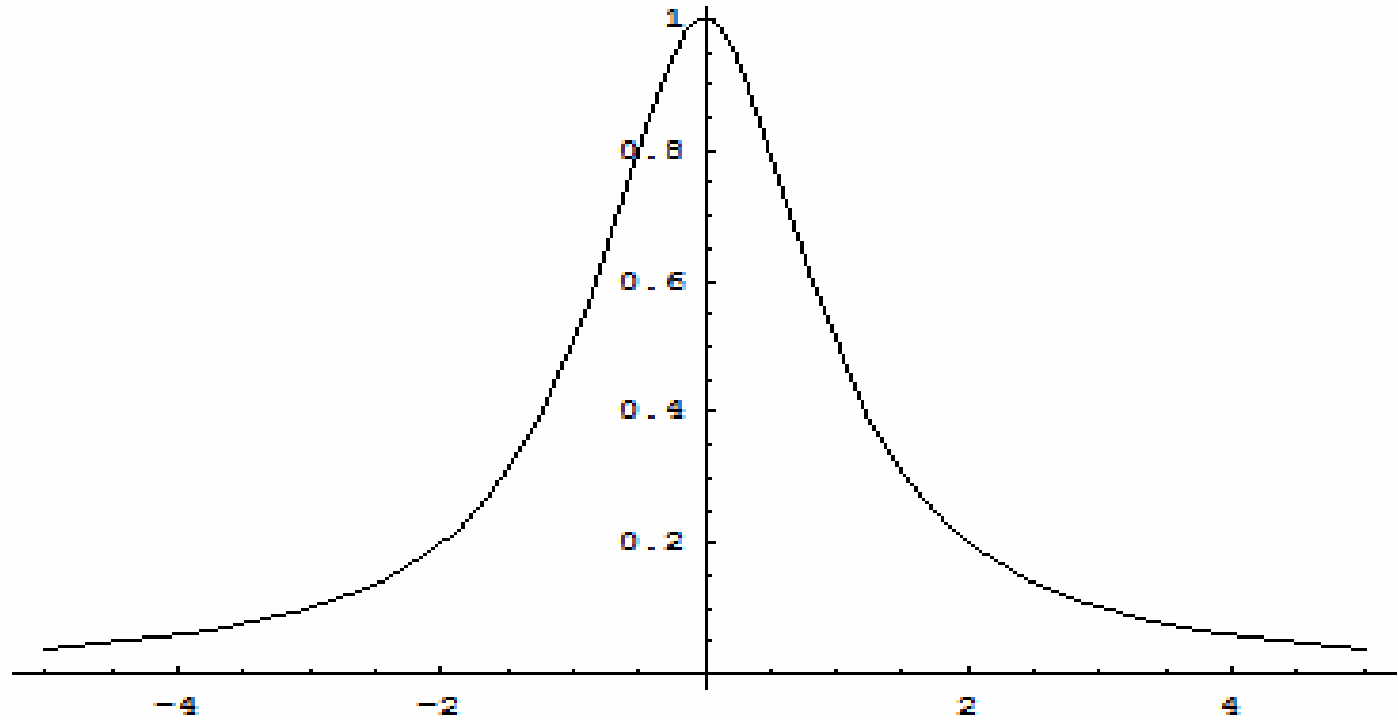
W przypadku teorii zbiorów rozmytych zastępujemy binarną funkcję charakterystyczną χ_A przez funkcję przynależności $\mu_A : X \to [0,1]$. Funkcję μ_A nazywamy *funkcją przynależności* lub

funkcją należenia. Jeżeli $\forall_{x \in X} \mu_A(x) \in \{0, 1\}$ to zbiór A jest zbiorem w zwykłym sensie i jest nazywany zbiorem ostrym (lub dokładnym, ang. crisp).

Klasyczny przykład zbioru rozmytego (fuzzy set) *near − zero* został przedstawiony przez Zadeha dla opisania pojęcia *liczby rzeczywistej bliskiej* 0. Taki zbiór może być zdefiniowany np. przez następującą funkcję należenia:

$$\mu_{near-zero} = \frac{1}{1 + x^2}$$

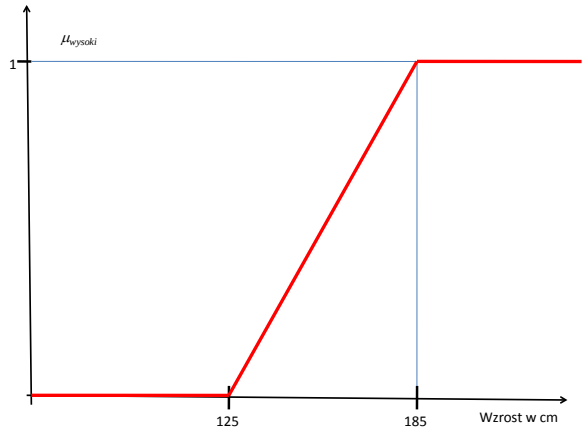
Wyglądającą tak:



Poprzednio rozważane pojęcie *wysokiej* osoby może być opisane przez – dla wzrostu x w centymetrach – przez funkcję:

$$\mu_{wysoki} = \begin{cases} 0 & \text{if} \quad x \leqslant 125 \\ 1 & \text{if} \quad x \geqslant 185 \\ \frac{x-185}{2} + 1 & \text{if} \quad 125 < x < 185 \end{cases}$$

która wygląda tak:



Powiemy, że zbiór rozmyty A zadany przez funkcję przynależności $\mu_A : X \to [0,1]$ jest *normalny* jeśli $\exists_{x \in X}\ \mu_A(x) = 1$.

Dla dobrego określenia własności zbiorów rozmytych musimy wprowadzić podstawowe pojęcia takie jak zawieranie czy zbiór pusty.

Rozmyte zawieranie
Niech A, B - zbiory rozmyte w pewnej przestrzeni X. Powiemy że zbiór rozmyty A jest zawarty w zbiorze rozmytym B (ozn. $A \subseteq B$) wtedy i tylko wtedy gdy $\forall_{x \in X}\ \mu_A(x) \leqslant \mu_B(x)$.

Rozmyty zbiór pusty
Powiemy że zbiór rozmyty $\emptyset$ jest pusty wtedy i tylko wtedy gdy $\forall_{x \in X}\ \mu_\emptyset(x) = 0$

W przypadku zwykłych zbiorów moc zbioru mierzymy liczbą jego elementów. W przypadku zbiorów rozmytych posługujemy się funkcją przynależności.

Moc zbioru rozmytego
Dla danego zbioru rozmytego A określamy jego moc

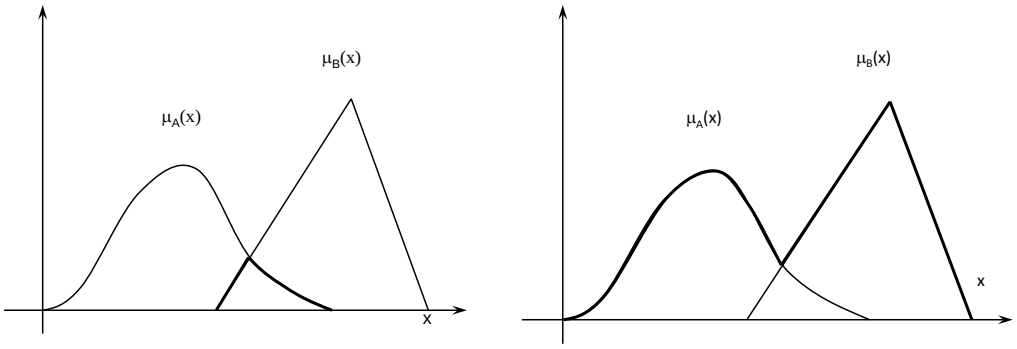
$$Power(A) = \overline{\overline{A}} = \begin{cases} \sum_{i=1}^{n} \mu_A(x) & \text{gdy} \quad X = \{x_1, \ldots, x_n\} \\ \int_X \mu_A(x) dx & \text{w p.p.} \end{cases}$$

Do pełni szczęścia brakuje nam tylko możliwości tworzenia zbiorów rozmytych poprzez sumowanie, przecinanie i uzupełnianie istniejących zbiorów (rozmytych).

Operacje na zbiorach rozmytych

W przypadku zbiorów rozmytych możemy definiować operacje na nich na wiele sposobów. Jednakże w ponad 90% zastosowań wykorzystuje się operacje podane w oryginalnej pracy Zadeha.

Operatory rozmyte
Dla danych zbiorów rozmytych A i B o funkcjach przynależności (odpowiednio) μ_A i μ_B definiujemy:
— Sumę zbiorów rozmytych $A \cup B$ będącą zbiorem rozmytym zadanym przez funkcję przynależności: $\mu_{A \cup B} = \max(\mu_A, \mu_B)$.
— Sumę zbiorów rozmytych $A \cap B$ będące zbiorem rozmytym zadanym przez funkcję przynależności: $\mu_{A \cap B} = \min(\mu_A, \mu_B)$.
— Dopełnienie zbioru rozmytego $\setminus A$ będące zbiorem rozmytym zadanym przez funkcję przynależności: $\mu_{\setminus A} = 1 - \mu_A$.



Inne przykłady operacji na zbiorach rozmytych:
— $\mu_{A\cup B} = \max(0, \mu_A + \mu_B - 1)$, $\mu_{A\cap B} = \min(1, \mu_A + \mu_B)$ - tzw. Łukasiewiczowskie
— $\mu_{A\cup B} = \max(0, \mu_A + \mu_B - \mu_A\mu_B)$, $\mu_{A\cap B} = \mu_A\mu_B$ - tzw. produktowe

Przy okazji uużywania zbiorów rozmytych warto zdawać sobie sprawę z dwóch istotnych zależności:

1. Teoria zbiorów rozmytych **NIE** jest alternatywą dla klasycznej teorii mnogości. Jest to **rozszerzenie** klasycznej teorii, które nie może funkcjonować niezależnie od niej, gdyż potrzebujemy aparatu teorii mnogości do zdefiniowania zbiorów rozmytych.
2. Teoria zbiorów rozmytych, pomimo natychmiastowego skojarzenia związanego z określeniem funkcji przynależności na przedziale $[0, 1]$, nie jest ściśle związana z wnioskowaniami probabilistycznymi. Wynika to z faktu, że układ zbiorów rozmytych w jakiejś przestrzeni może nie odpowiadać żadnemu sensownemu rozkładowi prawdopodobieństwa.

Reguły lingwistyczne

Reguły lingwistyczne (ang. linguistic rules) to wyrażenia postaci:

IF A_1 **AND** A_2 **AND ... AND** A_k **THEN** D

Gdzie warunki $A_1, \ldots, A_k$ i decyzja D odpowiadają pewnym zbiorom rozmytym. Na przykład:

IF pogoda jest **dobra AND** ruch jest **niewielki AND** mamy **dosyć** benzyny **THEN** będziemy na lotnisku za **około 30 minut**

Takie reguły możemy pozyskać od eksperta lub wydobyć z dostępnych źródeł danych. Aby je wykorzystywać w kontekście zbiorów rozmytych będziemy wykorzystywać operatory rozmyte.

13.1.2. Zagadnienia sterowania

Klasyczna teoria sterowania

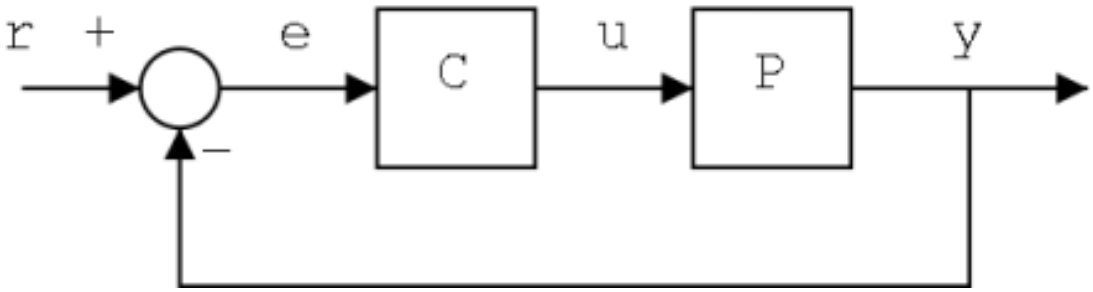
W mnóstwie sytuacji znanych z życia codziennego musimy wykorzystywać algorytmy (metody) pozwalające nam sterować otaczającymi nas urządzeniami. Niemal wszystkie urządzenia, od prostego grzejnika, przez automat do kawy, aż do promu kosmicznego, wymagają jakiejś techniki sterowania.

Podstawowe idee dotyczące sterowania urządzeniami sięgają starożytności, zaś początku współczesnej teorii sterowania upatruje się w pracach Jamesa Clerka Maxwella z 1868 roku, gdzie opisał on m.in. zagadnienie sprzężenia zwrotnego.

Zadaniem systemu sterowania (sterownika) jest mierzenie aktualnego stanu sytemu (układu), porównanie z wartościami spodziewanymi (referencyjnymi) i wyznaczenie ewentualnych poprawek (wartości sterowania), które należy wykonać.

Systemy sterowania dzielą się na dwa ogólne typy:

— Otwarte (open loop). System działa bez "oglądania się wstecz".
— Ze sprzężeniem zwrotnym (feedback loop). System odczytuje aktualny stan, ustala sterowanie, a następnie w pętli mierzy rezultat poprzedniej zmiany i koryguje sterowanie.



r - wartość odniesienia (referencyjna); e - obecny błąd (error); C - sterownik; P - sterowany proces (urządzenie); u - korekta (update); y - zmierzony stan P.

Aby (z sukcesem) zaprojektować system sterowania trzeba:

— Wyznaczyć przyzwoity model matematyczny sterowanego układu (urządzenia).
— Zidentyfikować zmienne sensoryczne i sterowane.
— Zdefiniować oczekiwane zachowanie - wartości referencyjne.
— Rozwiązać równanie (zwykle różniczkowe) sterowania.
— Zmodyfikować rozwiązanie w celu poprawy stabilności, odporności na zakłócenia i optymalności.

Każdy z powyższych kroków może być bardzo trudny (o ile nie niemożliwy) do wykonania w praktyce. Szczególnie modelowanie układu metodami matematycznymi często okazuje się niezwykle trudne. Także spełnienie kryteriów optymalności i stabilności może być istotnym wyzwaniem.

Istotnymi czynnikami oceny przy tworzeniu rozwiązań jest możliwość ich późniejszej implementacji w hardwarze i koszt uzyskania rozwiązania.

Alternatywa - sterowanie rozmyte

W wielu rzeczywistych sytuacjach nie jesteśmy w stanie podać zadowalającego, stabilnego modelu matematycznego dla urządzenia którym chcemy sterować. Co ciekawe, wieloma takimi urządzeniami operator (człowiek) jest w stanie sterować całkiem nieźle. Na przykład człowiek może sterować śmigłowcem, podczas gdy stworzenie autopilota dla helikopterów jest niezwykle trudne.

Możemy wszakże próbować wydobyć reguły sterowania od ekspertów. Takie reguły są jednak zwykle wyrażone w sposób nieprecyzyjny, np.:

— **If** x jest małe **and** y jest średnie **then** uruchom alarm.
— **If** x jest małe **and** y jest małe **then** zbiór z staje się duży.
— **If** x jest duże **then** ustal z jako małe.

Dzięki zbiorom rozmytym możemy przetłumaczyć takie sentencje na konkretne wartości numeryczne.

13.1.3. Sterowanie rozmyte

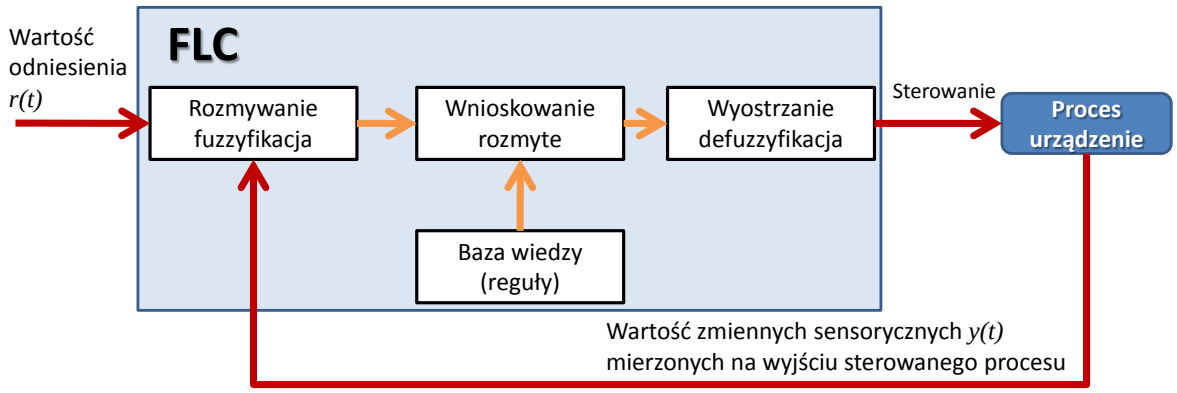
Elementy sterowania rozmytego

Tworzenie sterownika rozmytego (FLC od ang. Fuzzy Logic Control) składa się z kilku etapów:

1. Identyfikacja zmiennych sensorycznych (mierzonych na wejściu) i sterowanych (podawanych na wyjście).
2. Zdefiniowanie (skonstruowanie) zbiorów rozmytych dla zmiennych wejściowych i wyjściowych.
3. Skonstruowanie zestawu reguł sterowania - bazy wiedzy.
4. Rozmycie (fuzyfikacja), czyli zamiana pomierzonych (numerycznych) zmiennych wejściowych na odpowiadające im stopnie przynależności.
5. Zastosowanie reguł z bazy wiedzy. Wyznaczenie rozmytej wartości wyjścia.
6. Wyostrzenie (defuzyfikacja) - zamiana wyznaczonej z reguł rozmytej wartości wyjścia na konkretne wartości numeryczne które następnie są podawane do sterowanego układu.

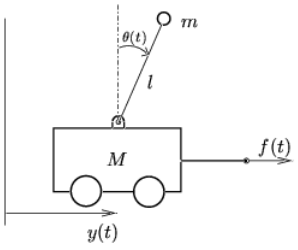
Na każdym z powyższych etapów musimy dokonywać wyborów mogących zadecydować o stosowalności i jakości FLC.

Typowy FLC wykorzystujący architekturę **Mamdaniego** został przedstawiony poniżej.



Przykład tworzenia sterownika rozmytego

Zagadnienie odwróconego wahadła przypomina zabawę, w której staramy się utrzymać kij od szczotki na czubku palca dzięki poruszaniu dłonią. Poruszmy podstawą (dłonią) w celu utrzymania masy (kijka) w pozycji jak najbliższej pionu. To zadanie jest często przedstawione w uproszczonej, jednowymiarowej formie jako zadanie utrzymania w pionie masy na ramieniu przymocowanym na zawiasie do wózka poruszającego się po szynach (patrz rysunek).



Choć pozornie trywialny, problem odwróconego wahadła okazuje się nieprzyjemny dla metod klasycznej teorii sterowania.

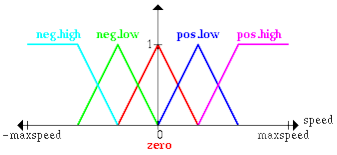
Aby uprościć rozważania i uczynić je bardziej przejrzystymi będziemy używać tylko dwóch zmiennych sensorycznych (pomiarów wejściowych):

1. $\theta(t)$ odpowiada kątowi (in w stopniach) w chwili t. Kąt jest mierzony względem pionu tzn. jest równy 0 gdy ramię jest pionowe. $\theta(t)$ leży w zakresie od -90 do 90 stopni.
2. $\dot{\theta}(t)$ - prędkość kątowa w chwili t.

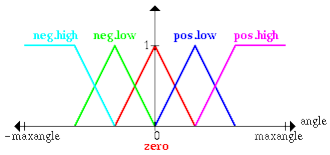
Jako zmienną wyjściową (sterowanie) przyjmiemy prędkość wózka. Będzie ona dodatnia bądź ujemna zależnie od kierunku jego ruchu. Jest to kolejne znaczące uproszczenie, gdyż w rzeczywistości ta prędkość jest funkcją masy wózka M, masy ramienia m i siły $f(t)$ wywieranej w chwili t.

Demonstrator w sieci: http://www.iit.nrc.ca/IR_public/fuzzy/FuzzyPendulum.html

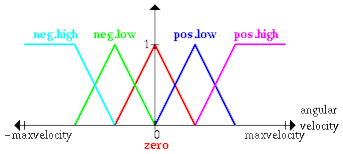
Definiujemy zbiory rozmyte dla zmiennych następująco:



Prędkość (wyjście)



Kąt



Prędkość kątowa

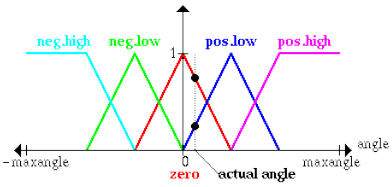
Tworzymy bazę wiedzy złożoną z reguł postaci:

— **If** kąt jest zero **and** prędkość kątowa jest zero **then** prędkość powinna być zero.

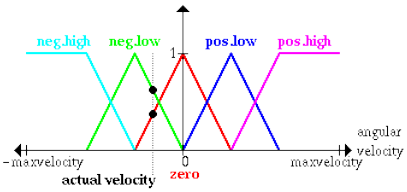
— **If** kąt jest zero **and** prędkość kątowa jest małą dodatnia (PL - positive low) **then** prędkość powinna być positive low.

Prędkość ↘	Kąt NH	NL	Z	PL	PH
Pr. NH			NH		
NL			NL	Z	
k Z	NH	NL	Z	PL	PH
ą PL		Z	PL		
t. PH			PH		

Jeśli weźmiemy konkretne numeryczne wartości dla kąta i prędkości kątowej, otrzymamy:



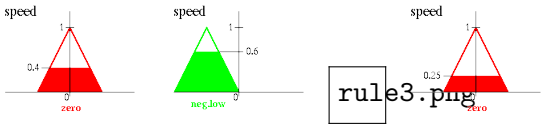
Kąt



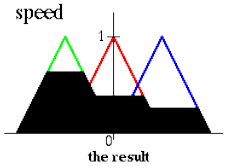
Prędkość kątowa

W naszej bazie są 4 reguły które pasują do tej sytuacji (odpalają):

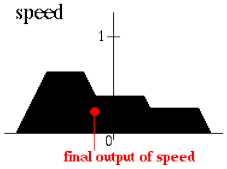
— If kąt zero and prędkość kątowa zero then prędkość zero.
— If kąt zero and prędkość kątowa negative low (NL) then prędkość negative low (NL).
— If kąt PL and prędkość kątowa zero then prędkość PL.
— If kąt PL and prędkość kątowa NL then prędkość zero.



Przez zsumowanie (wzięcie alternatywy) czterech reguł otrzymujemy zbiór rozmyty dla prędkości (wyjścia).



Póki co rezultat działania sterownika jest zbiorem rozmytym (dl prędkości wózka). Musimy teraz wybrać jedną wartość numeryczną reprezentującą ten zbiór. Istnieje wiele heurystyk pozwalających wyznaczyć tą wartość (wyostrzyć). Jedną z nich jest wzięcie środka ciężkości zbioru.



Literatura