Politechnika Warszawska

MSI

9. Uczenie przez indukcję - tworzenie pojęć, uczenie się klasyfikacji

Włodzimierz Kasprzak

Treść

1. Uczenie przez indukcję
2. Tryb podawania próbek
3. Tworzenie pojęć (grupowanie)
4. Kwantyzacja wektorowa
5. Uczenie się dyskretnej hipotezy
6. Drzewo decyzyjne w zadaniu klasyfikacji
7. Klasyfikator Bayesa
8. Klasyfikator kNN. K-d drzewa
9. Klasyfikator SVM

1. Uczenie przez indukcję

- **Uczenie przez indukcję** polega na nabywaniu wiedzy na **podstawie obserwacji** (zakłada się w nim dostępność próbek uczących (inna nazwa: przykłady trenujące):

$$T \Rightarrow h \wedge W ,$$

gdzie h – **hipoteza** wygenerowana w wyniku wnioskowania indukcyjnego, W - wiedza wrodzona, T – przykłady trenujące (próbki uczące), $\Rightarrow$ - logiczna implikacja (h, W, T są prawdziwe).

- Indukcyjna hipoteza wraz z wiedzą wrodzoną (być może pustą) wyjaśnia **możliwie dokładnie** otrzymaną informację uczącą.
- Celem indukcyjnego wnioskowania jest nie tylko odkrycie pewnych zależności w danych i wyjaśnieniu przykładów trenujących, ale przede wszystkim **przewidywanie nowych** faktów i obserwacji.
- **Założenie o indukcji**: hipoteza wybrana na podstawie zbioru uczącego sprawdza się także dla przykładów spoza tego zbioru.

Pojęcie (*concept*)

- **Pojęcia** są formą reprezentacji wiedzy. Pojęcie przypisuje obiektom etykiety ich kategorii.

  Przez „pojęcie” rozumiemy podzbiór obiektów lub zdarzeń definiowanych w ramach pewnego większego zbioru.

Np. pojęcie „ptak” jest podzbiorem obiektów w zbiorze wszystkich rzeczy takich, które zaliczamy do kategorii „ptaków”.

- Alternatywnie: „pojęcie” to funkcja charakterystyczna zbioru – przyjmuje wartość „True” dla przykładu należącego do kategorii i false w przeciwnym przypadku:

$$c(x) = \begin{cases} 1 & gdy\ x\ należy\ do\ tej\ kategorii\ (przykład\ pozytywny) \\ 0 & w\ przeciwnym\ przypadku\ (przykład\ negatywny) \end{cases}$$

- Gdy kategorii jest więcej niż dwa mówimy wtedy o pojęciach wielokrotnych : $c(x) = c_i$, gdzie $c_i \in C$

Próbki (przykłady trenujące)

- Oznaczmy: T - zbiór przykładów trenujących pochodzących z pewnej dziedziny X.
- Wiedza zdobyta na ograniczonym zbiorze przykładów T ma być użyteczna do przewidywania właściwości przykładów x z całej dziedziny X.
- Założymy, że dziedzina jest opisana za pomocą ustalonego zbioru atrybutów: $\mathrm{A} = \{a_1, a_2, \ldots, a_\mathrm{n}\}$, gdzie

$$a_1 : X \to A_1, \quad a_2 : X \to A_2, \quad \ldots, \quad a_\mathrm{n} : X \to A_\mathrm{n}.$$

- Przykład (próbkę) utożsamimy z wektorem wartości atrybutów:

$$\{x \in X : x = [a_1(x) = v_1, a_2(x) = v_2, \ldots, a_\mathrm{n}(x) = v_\mathrm{n},]\}$$

Sposoby uczenia przez indukcję

Wyróżnimy trzy podstawowe sposoby indukcyjnego uczenia:

1. **tworzenie pojęć** - uczenie się tworzenia klas (np. typów obiektów, wzorców) - proces nienadzorowanego uczenia;
2. **uczenie się pojęć** - uczenie się sposobu przydzielania obiektów (wzorców) do klas - uczenie się **klasyfikacji** - proces nadzorowanego uczenia;
3. **uczenie się aproksymacji funkcji** - uczenie się odwzorowywania obiektów na liczby rzeczywiste (proces nadzorowanego uczenia).

A) Tworzenie pojęć

Niech próbki (przykłady) uczące pozbawione są etykiety klasy, do której należą. W procesie tworzenia pojęć należy znaleźć zależności (podobieństwa cech) pomiędzy przykładami i pogrupować przykłady w klasy.

Grupowanie (klasteryzacja, samoorganizacja) odbywa się na zasadzie minimalizowania różnic przykładów z jednej grupy i maksymalizowania różnić pomiędzy grupami. Hipoteza określa zarówno tworzone klasy jak i przynależność przykładów do nich.

B) Uczenie się pojęć

- **Uczenie się pojęcia** polega na budowie pewnej funkcji h, która na podstawie wartości atrybutów przykładu x potrafi przydzielić związane z nim pojęcie (klasę) c , $c = h(x)$, $h: X \rightarrow C$
- Najczęściej funkcja jedynie przybliża rzeczywiste klasy w lepszym lub gorszym stopniu: $h(x) \approx c(x)$
- Uczenie polega na znalezieniu odwzorowania, które możliwie najlepiej przydziela pojęcia docelowe dla zbioru etykietowanych przykładów $(x, c(x))$. Wybór funkcji ma minimalizować błąd rzeczywisty na całej dziedzinie, który estymuje się za pomocą błędu na zbiorze trenującym.
- Uczenie się pojęcia polega na wyznaczeniu funkcji (**klasyfikatora**) zdefiniowanej na przestrzeni atrybutów (cech) obiektu, na podstawie zadanych próbek uczących podających wejście (atrybuty, cechy) i oczekiwane wyjście (etykietę klasy) tej funkcji.

C) Uczenie się aproksymacji funkcji

- Uczenie się aproksymacji funkcji jest podobne do uczenia się pojęć z tą różnicą, że liczba klas jest zbiorem liczb rzeczywistych, a nie określonym z góry skończonym zbiorem klas jak w przypadku uczenia się pojęć.
- Próbka ucząca ma postać wartości argumentów funkcji oraz wartości funkcji dla tych argumentów, zamiast etykiety klasy, jak było w przypadku klasyfikacji.
- Celem uczenia jest aproksymacja postaci (nieznanej) funkcji na podstawie przykładów reprezentacyjnych dla całej dziedziny funkcji.

2. Tryb podawania próbek

Algorytmy indukcyjnego uczenia się można implementować jako strategie przeszukiwania przestrzeni hipotez:

1. poszukujemy dobrej hipotezy w pewnej przestrzeni możliwych pojęć czy funkcji, zdefiniowanej w języku wybranym dla tego zadania;
2. przestrzeń hipotez powinna zawierać wszystkie hipotezy jakie może skonstruować uczeń;
3. przynajmniej jedna z tych hipotez jest poprawna.

Definicja. Hipoteza jest **nadmiernie dopasowana** do zbioru uczącego, jeśli inna hipoteza o większym błędzie próbki na tym zbiorze ma mimo to mniejszy błąd rzeczywisty.

Nadmierne dopasowanie

Aby ograniczyć ryzyko nadmiernego dopasowania należy preferować hipotezy proste o małym błędzie próbki na zbiorze uczącym.

Można ograniczyć liczbę pomyłek stosując odpowiedni *tryb podawania* próbek uczących.

Tryby podawania próbek uczących

1. wsadowy
2. inkrementacyjny
3. epokowy

Tryby podawania próbek

1. Tryb wsadowy

W trybie tym cały zbiór trenujący jest od razu dostępny, a po jego przetworzeniu podawana jest hipoteza. Powiększenie zbioru trenującego (pojawienie się nowych przykładów) powoduje konieczność rozpoczęcia uczenia się od nowa.

2. Tryb inkrementacyjny

Uczeń otrzymuje na raz tylko jeden przykład i zgodnie z nim zmienia swoją hipotezę. W każdej chwili uczenia dostępna jest hipoteza uznawana za ostateczną, gdy wyczerpią się próbki.

3. Tryb epokowy

Proces uczenia zorganizowany jest w cykle zwane epokami, w każdej z których jest przetwarzany pewien zbiór przykładów trenujących. Gdy epoka składa się jedynie z jednej próbki to otrzymujemy tryb inkrementacyjny.

3. Tworzenie pojęć (grupowanie)

Problem **grupowania (klasteryzacji)** próbek uczących będących wektorami liczb, a pozbawionych etykiet klas, jest przykładem **uczenia bez nadzoru** w celu **tworzenia pojęć**.

1) Klasteryzacja "*k*-średnich"

Jest to prosty deterministyczny algorytm klasteryzacji i organizacji danych.

2) Klasteryzacja EM

EM to ogólna technika statystyczna wyznaczania rozkładów poprzez iteracyjne poprawianie hipotez.

Klasteryzacja k-średnich

Klasteryzacja „k-średnich”

DANE: pewna liczba obserwacji (wektorów cech) c_j (j=1,2,..., n), zadana liczba klas K.

1. Inicjalizacja centrów k klastrów: $\mu^{(i)}$.
2. FOR każda obserwacja – wektor cech: c_j DO
- Przypisz obserwację do jej najbliższego klastra , tj. $\xi(c_j) \leftarrow i$, gdzie $\mathrm{d}(\mu^{(i)}, c_j) = \min_k \mathrm{d}(\mu^{(k)}, c_j)$, a d jest przyjętą miarą odległości (np. odległość Euklidesa).
3. FOR każdy klaster i (o $n^{(i)}$ obserwacjach) DO
- Wyznacz centrum klastra jako średnią z obserwacji tworzących klaster:

$$\mu^{(i)} = \frac{1}{n^{(i)}} \sum_{j,\xi(c_j)=i} c_j$$

4. Powtarzaj kroki 2 i 3 zadaną liczbę razy.

Tworzenie GMM - klasteryzacja EM

Algorytm EM (ang. „expectation-maximization) jest to ogólny sposób nienadzorowanego uczenia się (klasteryzacji) stosowany do estymacji parametrów modelu, między innymi do tworzenia modelu o postaci mieszaniny rozkładów Gaussa:

$$p(\boldsymbol{c} \mid \boldsymbol{\theta}) = \sum_{i=1}^{K} \alpha^i N(\boldsymbol{c} \mid \mu^i, \Lambda^i)$$

gdzie $N(.)$ oznacza rozkład normalny, a θ to zbiór parametrów, które należy określić: $\theta = \{ \alpha^i, \mu^i, \Lambda^i \mid i=1,2,\ldots,K\}$.

Klasteryzacja EM dla modelu GMM

DANE: obserwacje (wektory cech) c_j (j=1,2,..., n), liczba klas K.

1. **Inicjalizuj** parametry: $\theta=\{\ \alpha^i, \mu^i\ , \Lambda^i \mid i=1,...,K\ \}$.

REPEAT kroki E i M:

2. Krok E: FOR każda próbka c_j DO

FOR każda klasa i DO

$$P_j^i = \frac{p(c_j \mid \xi(c_j)=i,\theta)\alpha^i}{\sum_i p(c_j \mid \xi(c_j)=i,\theta)\alpha^i}$$

3. Krok M: FOR każda klasa i DO

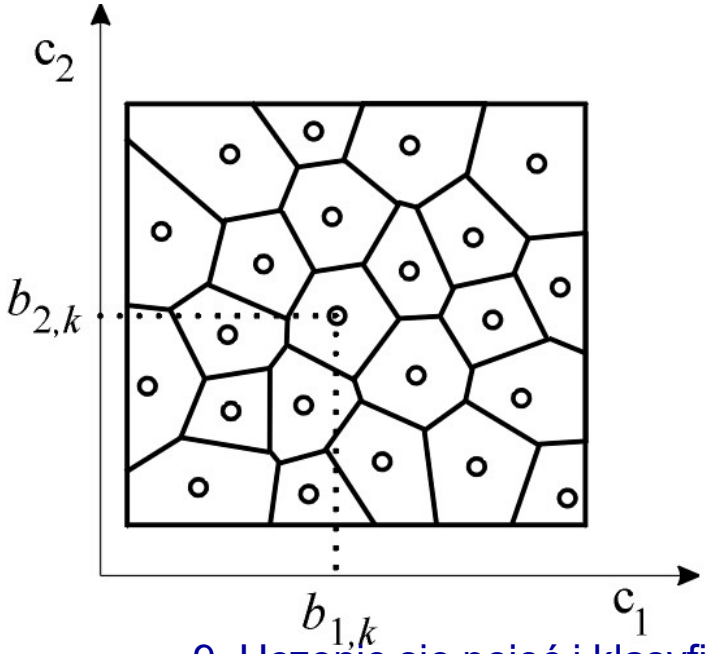
$$\mu^i = \frac{\sum_j c_j P_j^i}{\sum_j P_j^i} \qquad \Lambda^i = \frac{\sum_j (c_j-\mu^i)(c_j-\mu^i)^T P_j^i}{\sum_j P_j^i} \qquad \alpha^i = \frac{\sum_j P_j^i}{\sum_i \sum_j P_j^i}$$

UNTIL nie wykonano zadanej liczby iteracji

4. Kwantyzacja wektorowa

Kwantyzacja wektorowa stosuje metodę „*k*-średnich” dla wyznaczenia podziału przestrzeni reprezentacji na podobszary odpowiadające klastrom a następnie przydziela reprezentantom klastrów odpowiednie kody:

– W kroku uczenia wyznaczany jest słownik kodowy złożony z reprezentantów *K* klastrów.
– W kroku kodowania *kwantyzator* zastępuje wektor cech przez indeks reprezentanta w słowniku kodowym.



ν	**b**
1	$(b_{1,1}, b_{2,1})$
⋮	
k	$(b_{1,k}, b_{2,k})$
⋮	
L	$(b_{1,L}, b_{2,L})$

Sieć neuronowa Kohonena

Sieć neuronowa Kohonena realizuje kwantyzację wektorową.

Sieć Kohonena jest siecią 2-warstwową. W pierwszej warstwie każde z wyjść i jest połączone z każdym wejściem j, zaś funkcja i-tego wyjścia wynosi: $y_i = \mathbf{w}_i^T \mathbf{x}$. Wektor $\mathbf{w}_i$ stanowi reprezentanta i-tego klastra. Tym samym wyjście y_i jest iloczynem skalarnym wektora wejściowego i reprezentanta klastra. Często przyjmuje się, że zarówno wektory wag jak i dane wejściowe są znormalizowane – mają jednostkowy moduł. Jeśli tak nie jest, to aby uzyskać znormalizowany współczynnik korelacji stosuje się normalizację wyjścia:

$$z_i = f(y_i) = \frac{y_i}{|\mathbf{x}||\mathbf{w}_i|}$$

W drugiej warstwie następuje wybór jednego neuronu wyjściowego k o najwyższej wartości funkcji aktywacji, na drodze iteracyjnej modyfikacji wartości aktywacji neuronów, aż do chwili, gdy $z_k \approx 1$ i $z_i \approx 0$, dla $i \neq k$.

W tej warstwie każdy neuron i jest połączony z innymi m łukami pobudzającymi, zaś ze sobą samym łukiem osłabiającym pobudzenie:

$$v_{im} = \begin{cases} -\varepsilon & dla\ i \neq m \\ 1 & dla\ i = m \end{cases}$$

Uczenie w warunkach konkurencji

Wagi pierwszej warstwy sieci Kohonena uczone są metodą nienadzorowanego uczenia „w warunkach konkurencji” (ang. *unsupervised competitive learning*). Jest to realizacja grupowania (klasteryzacji) danych.

Idea algorytmu uczenia w warunkach konkurencji: wybierany jest neuron wyjściowy o najwyższej aktywacji i jego wagi $\mathbf{w}_l$ są korygowane „w kierunku” aktualnego wektora wejściowego. Jednocześnie od wektora wejściowego „odsuwana” jest najbliższa z pozostałych waga $\mathbf{w}_k$.

Niech N – zadana liczba neuronów wyjściowych $\mathbf{w}_i$, $i=1,...,N$

Z założenia wektory wag i próbki uczące $\{ \mathbf{x} \}$ są jednostkowe.

FOR (dla próbki $\mathbf{x}(t)$) DO

(1) Po określeniu wartości pobudzenia wyjść znajdujemy dwóch zwycięzców - k i l - stosując miarę odległości wejścia $\mathbf{x}$ od wag:

$$|\mathbf{x}(t) - \mathbf{w}_k| < |\mathbf{x}(t) - \mathbf{w}_l| < |\mathbf{x}(t) - \mathbf{w}_i| \text{ , .}$$

(2) Przesuwamy wektor wag najbliższy wejściu w kierunku wejścia:

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \alpha\,[\mathbf{x}(t) - \mathbf{w}_k(t)]$$

(3) Drugi najbliższy wektor wag ewentualnie „odsuwamy“ od wektora $\mathbf{x}(t)$:

$$\mathbf{w}_l(t+1) = \mathbf{w}_l(t) - \alpha\,[\mathbf{x}(t) - \mathbf{w}_l(t)]$$

5. Uczenie się dyskretnej hipotezy

Uczenie się dyskretnego reprezentanta (hipotezy) pojęcia: wnioskowanie definicji *pojęcia* na podstawie zbioru próbek o zadanych **atrybutach**, (o wartościach ze skończonej dziedziny) etykietowanych jako „przynależnych” lub „nie” do tego pojęcia.

W tym celu wyznaczona będzie dyskretna dziedzina funkcji binarnej klasyfikacji w przestrzeni wyznaczonej przez dyskretne dziedziny atrybutów obiektu.

Funkcja klasyfikacji binarnej w przestrzeni dyskretnej X:

$$c: X \rightarrow \{0, 1\}$$

Np. $c: \text{Niebo} \times \text{Temperatura} \times \text{Wilgotność} \times \text{Wiatr} \times \text{Woda} \times \text{Prognoza} \rightarrow \{\text{Tak}, \text{Nie}\}$

Hipoteza pojęcia - charakterystyka obiektów przynależnych do pojęcia w postaci ograniczeń nakładanych na atrybuty obiektu:

$$h: X \rightarrow P, \text{ gdzie } P \subseteq X, c(p) = 1$$

Przykład: sporty wodne

- **Pojęcie:** właściwe dni dla uprawiania sportów wodnych (wartości: Tak, Nie)
- **Atrybuty / cechy:**
  - Niebo (wartości: jasne, zachmurzone, deszczowe)
  - Temperatura powietrza (wartości: ciepło, chłodno)
  - Wilgotność (wartości: normalna, wysoka)
  - Wiatr (wartości: silny, słaby)
  - Woda (wartości: ciepła, zimna)
  - Prognoza (wartości: stała, zmiana)
- **Przykład próbki:**

  <słoneczne, ciepło, wysoka, silny, ciepła, stała, Tak>

Przykład, c.d.

Zbiór próbek:

Dzień	Niebo	Tempe-ratura	Wilgot-ność	Wiatr	Woda	Prog-noza	Sporty wodne
1	Jasne	Ciepło	Normalna	Silny	Ciepła	Stała	Tak
2	Jasne	Ciepło	Wysoka	Silny	Ciepła	Stała	Tak
3	Deszcz	Chłodno	Wysoka	Silny	Ciepła	Zmiana	Nie
4	Jasne	Ciepło	Wysoka	Silny	zimna	Zmiana	Tak

Reprezentacja hipotezy:

- **koniunkcja ograniczeń dla każdego atrybutu**:
  - "?" : "możliwa jest dowolna wartość"
  - "Ø" : "żadna wartość nie jest możliwa"
  - konkretna wartość.

Przykład hipotezy: h = <?, zimno, wysoka, ?, ?, ?>

- chłodny dzień i wysoka wilgotność → sporty wodne.

Przestrzeń wersji dla hipotez

Algorytm uczenia może mieć postać **przeszukiwania *przestrzeni wersji***.

Przestrzeń wersji jest podzbiorem hipotez zgodnych z przykładami uczącymi (jednym lub więcej), uporządkowanym relacją generalizacji hipotezy.

Strategia przeszukiwania może korzystać z uporządkowania hipotez w przestrzeni wersji (za pomocą **relacji generalizacji**) i realizować ukierunkowane przeszukiwanie przestrzeni.

Generalizacja następuje dzięki operacjom:

1. Zastąpienie stałych zmiennymi,
2. Wyeliminowanie warunku z koniunkcji,
3. Dodanie warunku do wyrażenia,
4. Zastąpienie cechy cechą nadrzędną w hierarchii klas.

Przeszukiwanie przestrzeni wersji

- **Cel przeszukiwania:**

  znaleźć „najlepszy” opis pojęcia ze zbioru wszystkich możliwych wersji (hipotez); “najlepszy” oznacza taki, który najlepiej uogólnia wszystkie (znane lub nieznane) elementy przestrzeni atrybutów obiektów.
- **Najbardziej ogólna hipoteza –** wszystkie wartości są dozwolone. Np. <?, ?, ?, ?, ?, ?>
- **Najbardziej specyficzna hipoteza –** żadne wartości nie są dozwolone. Np. < Ø, Ø, Ø, Ø, Ø, Ø >

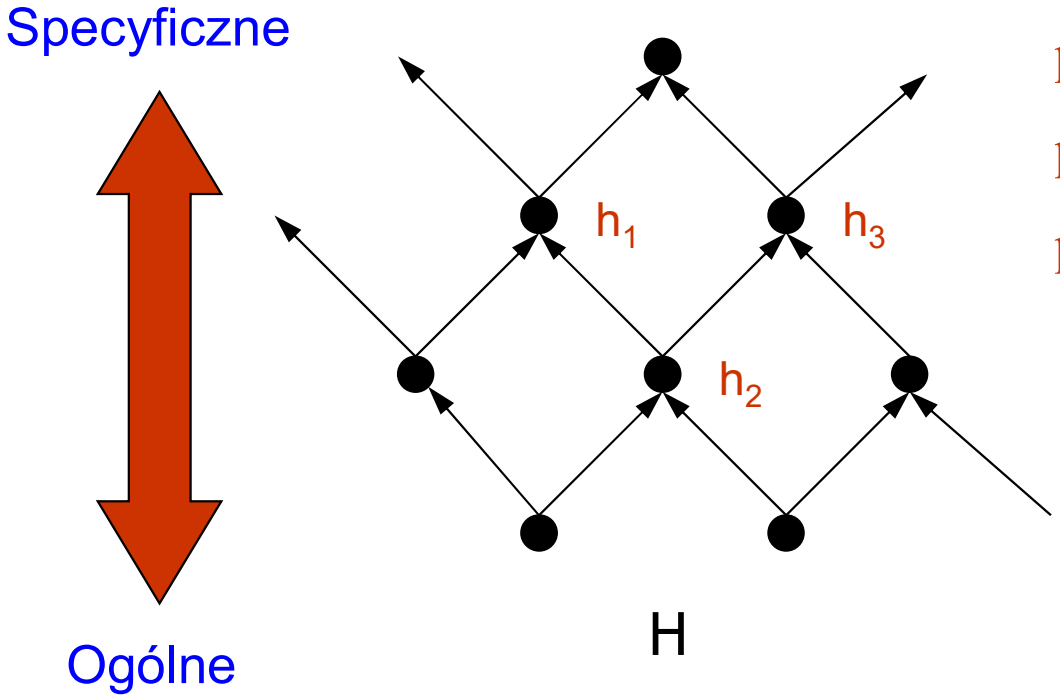
Uporządkowanie hipotez

Relacja generalizacji hipotez

Niech h_i, h_k są funkcjami boolowskimi (hipotezami). h_i jest bardziej lub równie ogólne jak h_k (ozn. $h_j \geq_g h_k$) wtw.

$$\forall x \in X: \; [h_k(x) = 1] \Rightarrow \; [\, h_j(x) = 1]$$

Przykład



h_1 = <jasny, ?, ?, silny , ? , ? >

h_2 = <jasny, ?, ?, ? , ? , ? >

h_3 = <jasny, ?, ?, ? , chłodny, ? >

$h_2 \geq_g h_1$,

$h_2 \geq_g h_3$,

Algorytm eliminacji kandydatów

Algorytm Eliminacji Kandydatów (CEA: *Candidate Elimination Algorithm*) realizuje przeszukiwanie *przestrzeni wersji* jednocześnie w dwóch kierunkach wyznaczonych przez relację generalizacji hipotez:

- przechodząc od szczegółu do ogółu i na odwrót.

Algorytm **CEA** redukuje aktualny rozmiar przestrzeni wersji wraz z dodaniem kolejnych przykładów.

Niech G będzie zbiorem hipotez maksymalnie ogólnych, zaś S zbiorem hipotez maksymalnie szczegółowych. Algorytm CEA na przemian uszczegółowia G i uogólnia S, aż zbiegną się one do jednego pojęcia docelowego.

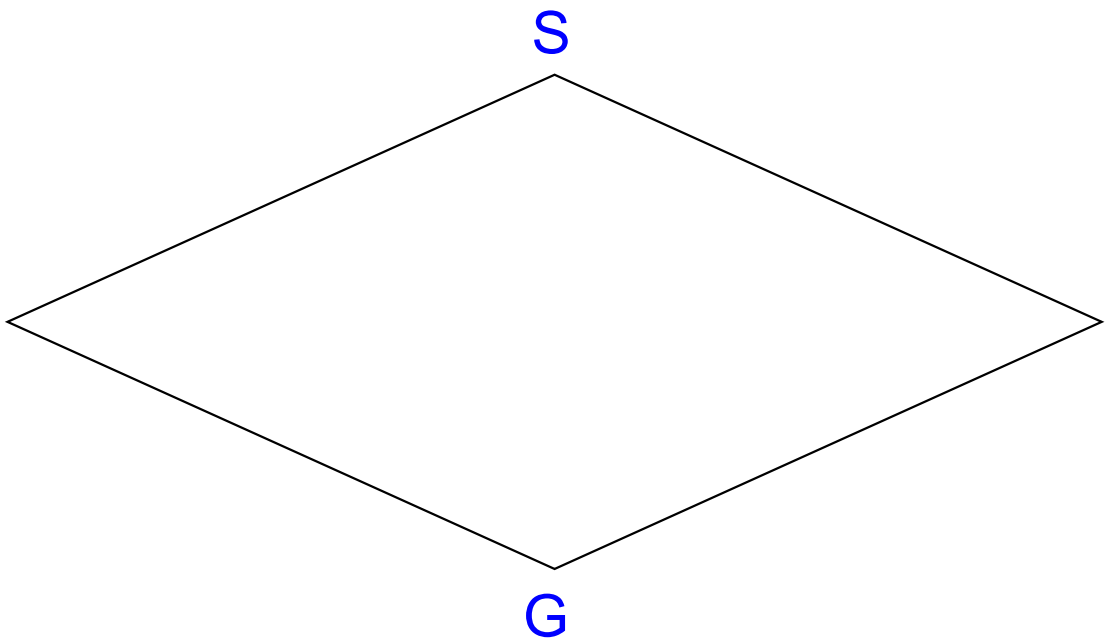
Zwarta reprezentacja przestrzeni wersji

- Algorytm CEA stosuje zwartą reprezentację przestrzeni wersji: zamiast przeliczać wszystkie hipotezy zgodne z przykładami trenującymi, można reprezentować ich **najbardziej specyficzne i najbardziej ogólne ograniczenie** (kontur), a hipotezy zawarte pomiędzy tymi ograniczeniami generować tylko w miarę potrzeby.
- **Ogólnym ograniczeniem** *G*, ze względu na przestrzeń hipotez *H* i dane trenujące *D*, jest zbiór maksymalnie ogólnych hipotez w *H* zgodnych z *D*.
- **Specyficznym ograniczeniem** *S*, ze względu na przestrzeń hipotez *H* i dane trenujące *D*, jest zbiór minimalnie ogólnych (tzn. maksymalnie specyficznych) hipotez w *H* zgodnych z *D*.

Ograniczenia w przestrzeni wersji

- Zwarta reprezentacja przestrzeni wersji:

$$\langle G, S \rangle = \{ h \in H \mid \exists g \in G \; \exists s \in S : \; g \geq_g h \geq_g s \}$$



Algorytm CEA (1)

funkcja CEA(*przykłady*) **zwraca** pojedyncze *pojęcie*
{
  Przypisz G najogólniejsze *pojęcie* (*hipotezę*) z przestrzeni;
  Przypisz S pierwszy pozytywny *przykład uczący*;
  for każdy nowy pozytywny *przykład* p {
    Usuń z G *hipotezy* niezgodne z p ;
    for każde $s \in S$
      if (s zgodne z p) **then** pozostaw;
      else zastąp s jej wszystkimi najbliższymi uogólnieniami
        h zgodnymi z p i takimi, że dla każdego nowego h
        istnieje hipoteza bardziej ogólna w G ;
    Usuń z S takie *hipotezy*, które są ogólniejsze od innej
      hipotezy w S ;
}

Algorytm CEA (2)

for każdy nowy negatywny *przykład* n {
  Usuń z S *pojęcia* niezgodne z n ;
  for (dla każdego $g \in G$)
    if g zgodne z n **then** zostaw;
    else zastąp g najbliższymi bardziej specyficznymi hipotezami h zgodnymi z n i dla których istnieją w S hipotezy bardziej specyficzne;
  Usuń z G takie *hipotezy*, które są bardziej specyficzne niż inna *hipoteza* w G;
}
// Jeśli $((G == S) \wedge (G$ i S zawierają jedno *pojęcie*$))$
// to $G = S$ jest unikalnie zadanym znalezionym *pojęciem*.
return G;
}

Przykład - Algorytm eliminacji kandydatów

Zastosujemy algorytm eliminacji kandydatów do **nauczenia się pojęć** umożliwiających binarną klasyfikację pogody nad morzem na sprzyjającą wodnym sportom lub nie. Próbka ucząca jest agregacją następujących obserwacji:

<niebo, temperatura, wilgotność, wiar, woda, prognoza>.

Niech będą dane następujące próbki uczące (3 pozytywne i 1 negatywna):

Próbka	Niebo	Temp.	Wilgotność	Wiatr	Woda	Prognoza	Sprzyja?
1	Słońce	Ciepło	Normalna	Silny	Ciepła	Stała	Tak
2	Słońce	Ciepło	Wysoka	Silny	Ciepła	Stała	Tak
3	Deszcz	Zimno	Wysoka	Silny	Ciepła	Zmiana	Nie
4	Słońce	Ciepło	Wysoka	Silny	Chłodna	Zmiana	Tak

Przykład (1)

Inicjalizujemy dwa zbiory:

- G_0 - zbiór maksymalnie ogólnych hipotez w przestrzeni hipotez tego problemu H,
- S_0 – zbiór najbardziej specyficznych hipotez w przestrzeni H (efektywniejsze rozwiązanie – pierwsza pozytywna próbka).

W tym przypadku:

- $S_0 = \{<\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>\}$,

lub $S_0 = \{<$słońce, ciepło, normalna, silny, ciepła, stała$>\}$

- $G_0 = \{<?, ?, ?, ?, ?, ?>\}$

Przykład (2)

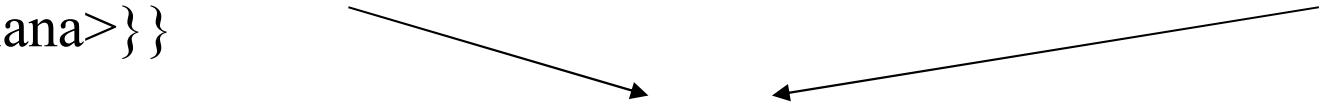
Kolejne iteracje:

- Uwzględniamy próbkę 1:
  - S1 = {<słońce, ciepło, normalna, silny, ciepła, stała>}
  - G1 = {<?, ?, ?, ?, ?, ?>}
- Uwzględniamy próbkę 2:
  - S2 = {<słońce, ciepło, ?, silny, ciepła, stała>}
  - G2 = {<?, ?, ?, ?, ?, ?>}
- Uwzględniamy próbkę 3:
  - S3 = {<słońce, ciepło, ?, silny, ciepła, stała>}
  - G3 = {<słońce, ?, ?, ?, ?, ?>,
    <?, ciepło, ?, ?, ?, ?>,
    <?, ?, ?, ?, ?, stała>, … }

Przykład (3)

- Uwzględniamy próbkę 4:

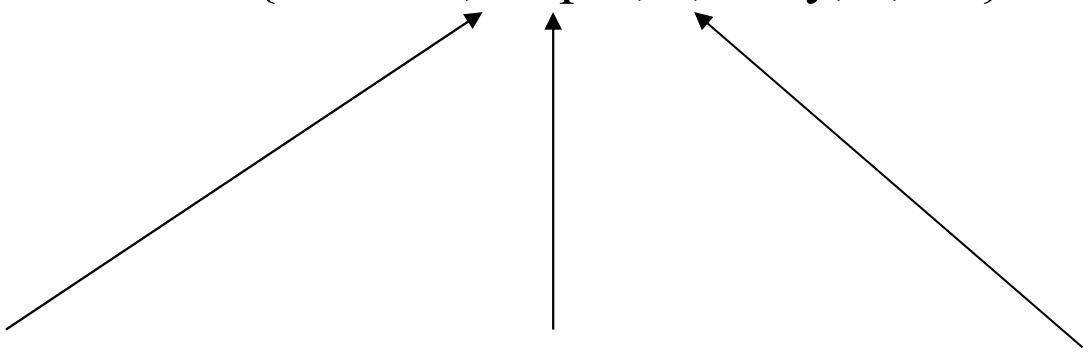
S4 = {{<słońce, ciepło, ?, silny, ciepła, stała>}, {<słońce, ciepło, wysoka, silny, chłodna, zmiana>}}



S4 = {<słońce, ciepło, ?, silny, ?, ?>}

G4 zdąża do S4 i proces kończy się w sytuacji, gdy (G = S = S4)

G4 = {<słońce, ciepło, ?, silny, ?, ?>}



<słońce, ?, ?, silny, ?, ?> <słońce, ciepło, ?, ?, ?, ?> <?, ciepło, ?, silny,?,?>

Zbieżność wyniku CEA

- Przestrzeń wersji w algorytmie CEA zbiega do poprawnego pojęcia jeśli:
  - Przestrzeń hipotez H zawiera szukane pojęcie;
  - Próbki uczące nie zawierają błędów.
- Efektywna strategia pobierania próbek: kolejna wymagana próbka ucząca powinna rozdzielać alternatywne hipotezy zawarte w aktualnej przestrzeni wersji.
- Jeśli pojęcie jest nauczone jedynie częściowo (istnieją alternatywne hipotezy) to można zastosować wielokrotną klasyfikację w oparciu o każda hipotezę a następnie ustalić klasę poprzez **głosowanie większościowe**.

6. Drzewo decyzyjne w klasyfikacji

W drzewie decyzyjnym występują:

- **węzły** niekońcowe – odpowiadają one testom przeprowadzanym na atrybutach przykładów,
- **gałęzie** (krawędzie) – odpowiadają one możliwym wynikom tych testów,
- **liście** (węzły końcowe) – odpowiadają one etykietom kategorii (klas).

Klasyfikacja przykładu za pomocą drzewa decyzyjnego polega na przejściu ścieżki od korzenia do liścia drzewa wzdłuż gałęzi wyznaczanych przez wyniki testów związanych z odwiedzanymi kolejno węzłami.

Osiągnięcie liścia wyznacza kategorię (klasę) dla tego przykładu – drzewo decyzyjne reprezentuje **hipotezę**.

Przykład problemu decyzyjnego

Przykład 1. Problem decyzyjny: czy czekać na wolny stolik w restauracji? Załóżmy, że dla podjęcia decyzji korzystamy z następujących **atrybutów**:

- **Alternatywa**: czy jest inna restauracja w okolicy?
- **Bar**: czy można oczekiwać w barze?
- **Piątek**: czy dziś jest piątek czy już sobota?
- **Głód**: czy jesteśmy głodni?
- **Osoby**: liczba osób w restauracji (nikogo, trochę, pełno)
- **Cena**: zakres cenowy (+, ++, +++)
- **Deszcz**: czy pada deszcz?
- **Rezerwacja**: czy mamy rezerwację?
- **Typ**: rodzaj restauracji (francuska, włoska, tajlandzka, *burger*)
- **Czas**: szacowany czas czekania (0-10 min.,10-30, 30-60, >60).

Próbki uczące

Próbki uczące (przykłady trenujące) są opisane poprzez wartości atrybutów, które w naszym przykładzie 1 mogą być typu logicznego, dyskretne (liczby całkowite) lub ciągłe. Na potrzeby drzewa decyzyjnego ciągła dziedzina zostaje podzielona na przedziały.

Próbki uczące w przykładzie 1 opisują sytuacje, w których będziemy czekali lub nie czekali na stolik. Klasyfikujemy każdy przykład (decyzja) na jedną z dwóch klas:

- pozytywną – decyzja „*czekać*” (T) lub
- negatywną – decyzja „*nie czekać*” (F).

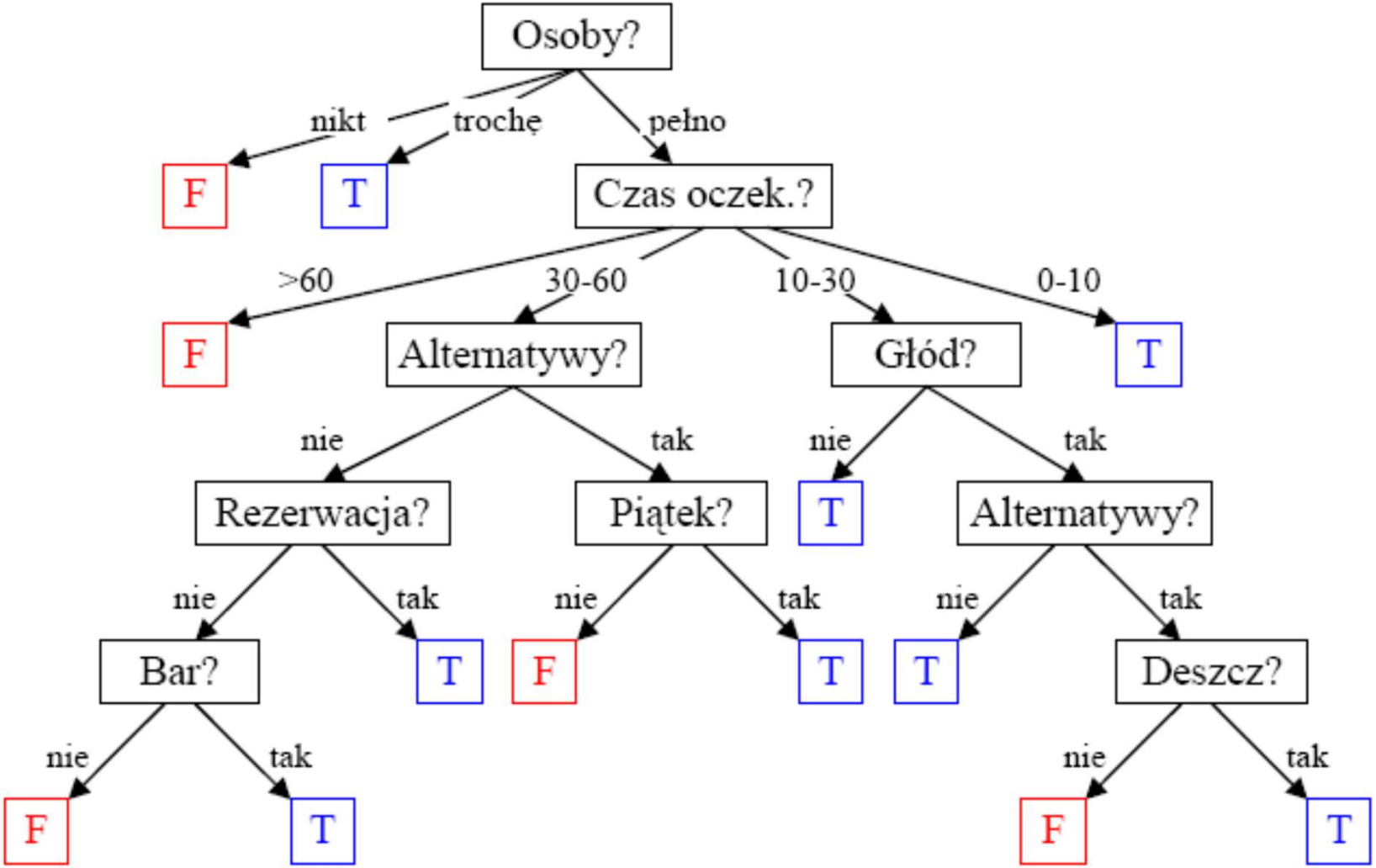
Próbki uczące – c.d.

Tabela 1

Próbka	Atrybuty										CEL
	Alt	Bar	Piąt	Głód	Osoby	Cena	Deszcz	Rez	Typ	Czas	Czekaj
X1	Tak	Nie	Nie	Tak	Trochę	+++	Nie	Tak	franc	0-10	True
X2	Tak	Nie	Nie	Tak	Pełno	+	Nie	Nie	tajski	30-60	False
X3	Nie	Tak	Nie	Nie	Trochę	+	Nie	Nie	burger	0-10	True
X4	Tak	Nie	Tak	Tak	Pełno	+++	Nie	Nie	tajski	10-30	True
X5	Tak	Nie	Tak	Nie	Pełno	+++	Nie	Tak	franc	>60	False
X6	Nie	Tak	Nie	Tak	Trochę	++	Tak	Tak	włoski	0-10	True
X7	Nie	Tak	Nie	Nie	Nikt	+	Tak	Nie	burger	0-10	False
X8	Nie	Nie	Nie	Tak	Trochę	++	Tak	Tak	tajski	0-10	True
X9	Nie	Tak	Tak	Nie	Pełno	+	Tak	Nie	burger	>60	False
X10	Tak	Tak	Tak	Tak	Pełno	+++	Nie	Tak	włoski	10-30	False
X11	Nie	Nie	Nie	Nie	Nikt	+	Nie	Nie	tajski	0-10	False
X12	Tak	Tak	Tak	Tak	Pełno	+	Nie	Nie	burger	30-60	True

Drzewo decyzyjne

Przykładowe drzewo decyzyjne dla „czekania w restauracji”.



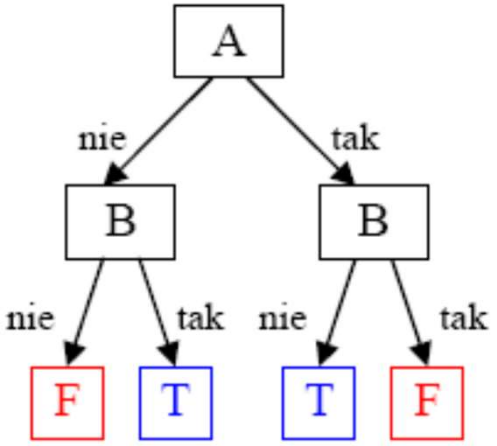
Siła wyrazu drzew decyzyjnych

Drzewa decyzyjne mogą wyrazić każdą **funkcję logiczną lub dyskretną** zależną od zbioru atrybutów wejściowych.

Np. dla funkcji logicznych (o wartościach typu Boolean) pojedynczy wiersz w tabeli prawdy odpowiada pojedynczej ścieżce w drzewie od korzenia do liścia.

A	B	A ⊗ B
F	F	F
F	T	T
T	F	T
T	T	F

⊗ oznacza funkcję „XOR”



Funkcja logiczna (lewa strona) i jej reprezentacja w postaci drzewa decyzyjnego (prawa strona).

Złożoność problemu uczenia drzewa decyzyjnego

Jaka jest liczba możliwych drzew decyzyjnych dla *n* atrybutów typu logicznego (Boolean)?

Każda funkcja logiczna o *n* binarnych atrybutach odpowiada tablicy prawdy o 2^n wierszach. Liczba różnych tablic prawdy o takim rozmiarze wynosi 2^{2^n}.

Np. dla n=6 istnieje 18,446,744,073,709,551,616 drzew.

Tak duża przestrzeń problemu uczenia wymaga stosowania nietrywialnych algorytmów uczenia drzew decyzyjnych.

Uczenie się drzewa decyzyjnego

Problem uczenia drzewa decyzyjnego

1. Dla deterministycznej funkcji zmiennych *X* zawsze można utworzyć drzewo decyzyjne w pełni zgodne z próbkami uczącymi takie, że każdej próbce odpowiadać będzie pojedyncza ścieżka od korzenia do liścia.
2. Jednak drzewo z pkt. 1 zwykle nie będzie dobrze uogólniać (przewidywać) nieznanych próbek. Dlatego należy znaleźć bardziej zwarte drzewo decyzyjne.

Idea zstępującej metody uczenia drzewa decyzyjnego

1. Cel – szukamy zwartego (niedużego) drzewa zgodnego z próbkami uczącymi.
2. Postępowanie – rekursywnie wybieramy "najważniejszy" atrybut do roli korzenia pod-drzewa.

Uczenie się drzewa decyzyjnego (Algorytm ID3 - Ross Quinlan)

```
function DTL(próbki, atrybuty, domyślna) returns drzewo decyzyjne
{  if (próbki == ∅)  then return domyślna;
     else if (wszystkie próbki należą do jednej klasy)
          then return KLASA(próbki);
          else if (atrybuty == ∅)
               then return KLASAWIĘKSZOŚCIOWA(próbki);
               else
               {    best ← WYBIERZATRYBUT(atrybuty, próbki);
                    drzewo ← nowe drzewo o korzeniu testującym atrybut best ;
                    m ← KLASAWIĘKSZOŚCIOWA(próbki);
                    for each (wartość v_i ∈ best) do
                    {    próbki_i ← {elementy w próbki o best = v_i};
                         poddrzewo ←  DTL(próbki_i, atrybuty - best, m);
                         dodaj gałąź do drzewa o etykiecie v_i i poddrzewie poddrzewo;
                    }
               }
               return drzewo;
}
```

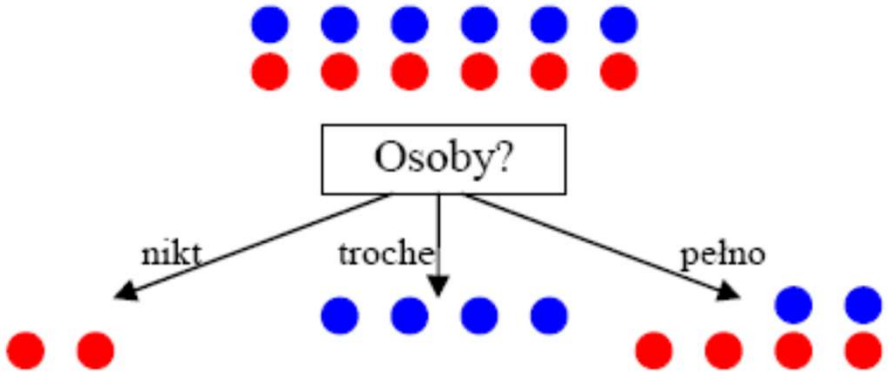
Kryterium „zysku informacji” dla wyboru testów

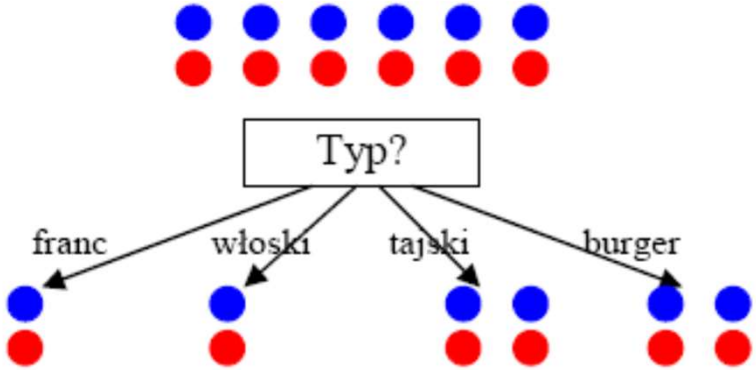
Realizacja podfunkcji WybierzAtrybut w algorytmie DTL.

Idea wyboru atrybutu

Wybór następuje według zasady: “dobry” atrybut to taki, który dzieli próbki na podzbiory, które są (w idealnym przypadku) “wszystkie pozytywne” lub “wszystkie negatywne”.

Przykład 2. Dla próbek z przykładu 1 testując atrybut *Osoby* (lewa strona) zbliżamy się do rozdzielenia przypadków pozytywnych od negatywnych. Testowanie atrybutu *Typ* (prawa strona) zasadniczo niczego w tym zakresie nie wnosi.





Entropia

Formalnie biorąc implementacja podfunkcji **WybierzAtrybut** oparta może być na teorii informacji.

Oczekiwana wartość informacji (**entropia**) dla zmiennej losowej X o n wartościach wynosi:

$$H(P_X(v_1), \ldots, P_X(v_n)) = - \Sigma_{i=1}^{n} [P_X(v_i) \log_2 P_X(v_i)].$$

Im bardziej prawdopodobna realizacja zmiennej losowej tym mniej informacji ona zawiera. Z kolei im większa niepewność zdarzenia tym więcej zawiera ono informacji.

Dla zmiennej losowej o dwóch równie prawdopodobnych możliwych wartościach $([0.5, 0.5])$ entropia wynosi 1 [bit].

“Jakość” atrybutu jako „zysk informacji”

Możemy oceniać “jakość” danego atrybutu na podstawie ilości informacji, jaka pozostanie, tzn. będzie jeszcze zawarta w zbiorze próbek po jego testowaniu zadanym atrybutem.

Niech zbiór próbek S **dwóch klas** zawiera p próbek dla klasy „pozytywnej” i n dla „negatywnej”. Wtedy zawartość informacyjna tego zbioru próbek („przed testowaniem”) wynosi:

$$\mathrm{H(S)} = \mathrm{H}\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Wybrany atrybut A dzieli zbiór próbek E na podzbiory, $E_1, \ldots, E_\mathrm{v}$, odpowiednio do przyjmowanych przez nie wartości dla A, gdy A ma v różnych wartości. Mamy nadzieję, że każdy z podzbiorów będzie zawierał już mniej informacji po testowaniu ze względu na atrybut A niż przedtem.

Zysk informacji

Niech w każdym podzbiorze będzie p_i próbek o klasie pozytywnej i n_i próbek o klasie negatywnej. Oczekiwana wartość pozostałej entropii to ważona suma entropii dla każdego podzbioru:

$$\mathrm{H}_{zostało}(S|A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} \cdot \mathrm{H}\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

Tym samym zysk informacji (ZI) odpowiadający redukcji entropii po wykonaniu testu dla atrybutu wynosi:

$$\mathrm{ZI}(S|A) = \mathrm{H}(S) - \mathrm{H}_{zostało}(S|A) = \mathrm{H}\left(\frac{p}{p + n}, \frac{n}{p + n}\right) - \mathrm{H}_{zostało}(S|A)$$

W podfunkcji **WybierzAtrybut** stosujemy więc zasadę wyboru atrybutu o **największym zysku informacji**:

$$\arg\max_{A} \mathrm{ZI}(A)$$

Przykład wyboru atrybutu

Przykład 2 (c.d.)

Dla zbioru próbek z tabeli 1 zachodzi:

$$p = n = 6;\ \mathrm{H(S)} = \mathrm{H}([6/12, 6/12]) = 1 \text{ bit/znak}$$

Określimy zysk informacji dla atrybutów *Osoby* i *Typ* testowanych na podanym zbiorze:

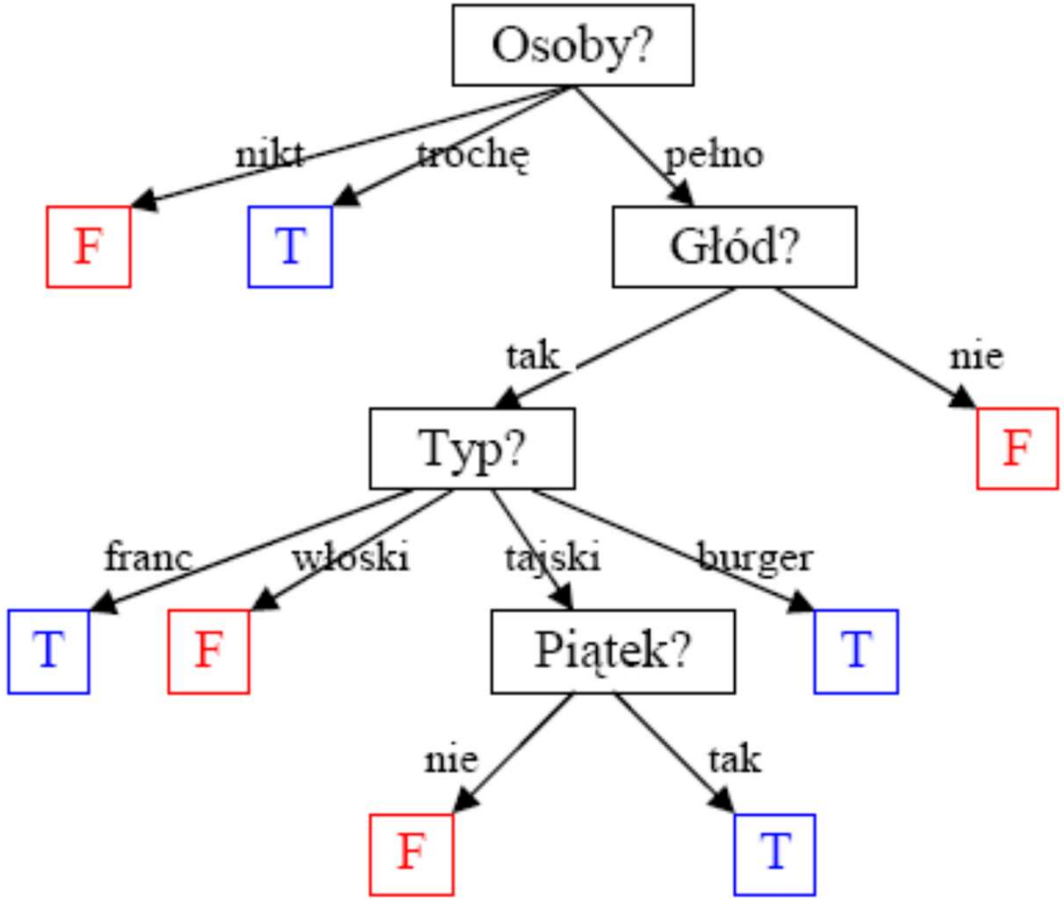
$$\mathrm{ZI}(S|\,Osoby) = 1 - [\frac{2}{12}\mathrm{H}([0,1]) + \frac{4}{12}\mathrm{H}([1,0]) + \frac{6}{12}\mathrm{H}([\frac{2}{6},\frac{4}{6}])] \cong 0.541 \text{ [bit/znak]}$$

$$\mathrm{ZI}(S|\,Typ) = 1 - [\frac{2}{12}\mathrm{H}([\frac{1}{2},\frac{1}{2}]) + \frac{2}{12}\mathrm{H}([\frac{1}{2},\frac{1}{2}]) + \frac{4}{12}\mathrm{H}([\frac{2}{4},\frac{2}{4}]) + \frac{4}{12}\mathrm{H}([\frac{2}{4},\frac{2}{4}])] = 0$$

Można sprawdzić, że atrybut *Osoby* zapewnia najwyższy zysk informacji ZI spośród atrybutów i zostanie on wybrany przez algorytm DTL do testowania próbek w korzeniu drzewa.

Przykład drzewa decyzyjnego

Przykład 3 Drzewo decyzyjne dla problemu z tabeli 1 powstałe w wyniku uczenia algorytmem DTL z kryterium zysku informacji dla zbioru 12 podanych próbek.



Iloraz zysku informacji

Innym popularnym kryterium dla wyboru atrybutów w algorytmie uczenia się drzewa decyzyjnego jest iloraz zysku (przyrostu) informacji.

Podział informacji (*Split*) – to entropia obliczana dla atrybutu ze względu na jego zbiór wartości w zbiorze uczącym (poprzednio liczyliśmy entropię na podstawie podzbiorów dla klas decyzyjnych).

$$Split(S|A) = -\sum_{a \in Wartości(A)} p(a|A) \cdot \log[p(a|A)]$$

Iloraz zysku (przyrostu) informacji (ang. *Gain ratio*):

$$IZI(S|A) = \frac{ZI(S|A)}{Split(S|A)}$$

Iloraz zysku informacji (2)

Podobnie jak zysk informacji ZI(S|A) chcemy maksymalizować „Iloraz zysku informacji” – wybieramy atrybut A w węźle S dający największy IZI(S|A).

Normalizacja zysku informacji przez zastosowanie *splitu dla* tego atrybutu daje sprawiedliwsze porównywanie ze sobą atrybutów o różnej liczbie możliwych wartości (w kryterium zysku informacji atrybuty o większej liczbie wartości są niejawnie preferowane).

Uwaga: warto zapoznać się też z bardziej zaawansowanymi algorytmami uczenia drzew decyzyjnych: metodami

- „Classification and regression tree” (CART)
- Algorytm C4.5 Rossa Quinlana.

7. Klasyfikator Bayesa

Model klas obejmuje:

- rozkład a priori prawdopodobieństwa klas: $p(\Omega_k)$, $k = 1, \ldots, K$.
- rozkład prawdopodobieństwa wektora cech pod warunkiem klasy $p(\boldsymbol{c}|\Omega_k)$

Reguła decyzyjna :

$$\varsigma(c) = \arg\max_{\lambda} p(\Omega_\lambda \mid c) = \arg\max_{\lambda} p(\Omega_\lambda)\, p(c \mid \Omega_\lambda).$$

Uczenie się (estymacja) elementów modelu:

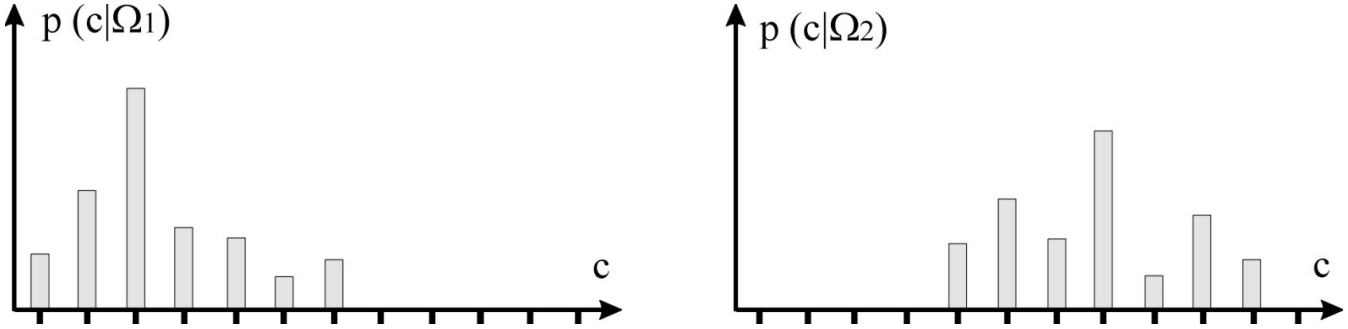
1. Rozkład $p(\Omega_k)$ odpowiada względnej częstości występowania próbek danej klasy w zbiorze uczącym;
2. Rozkład $p(\boldsymbol{c} \mid \Omega_k)$ może być postaci **nie-parametrycznej** (np. znormalizowany histogram) lub **parametrycznej** – przyjmujemy wspólny typ funkcji rozkładu prawdopodobieństwa dla klas i estymujemy parametery tych funkcji na podstawie zbioru próbek.

Klasyfikator ML („maksymalnej wiarygodności”): klasyfikator Bayesa dla **jednorodnego** rozkładu prawdopodobieństwa klas. Reguła decyzyjna upraszcza się do postaci:

$$\varsigma(c) = \arg\max_{\lambda} p(\Omega_\lambda \mid c) = \arg\max_{\lambda} p(c \mid \Omega_\lambda).$$

Uczenie klasyfikatora Bayesa

Nieparametryczny rozkład



Np. warunkowy rozkład prawdopodobieństwa cech odpowiada znormalizowanym histogramom tworzonym osobno dla każdej klasy

Parametryczny rozkład

Stosując estymator ML („maksymalnej wiarygodności”) dla klasy Ω_k znajdujemy wartości zbioru parametrów $\boldsymbol{\theta}^{(k)}$ które maksymalizują prawdopodobieństwo próbek zbioru uczącego (obserwacji):

$$\max_{\theta^{(k)} \in \Theta} \sum_{c_j \in C^{(k)}} \log P\left(c_j, \boldsymbol{\theta}^{(k)}\right)$$

W tym celu należy rozwiązać:

$$\frac{\partial}{\partial \theta_i} \sum_{c_j \in C^{(k)}} \log P(c_j, \boldsymbol{\theta}^{(k)}) = 0, \qquad \theta_i \in \boldsymbol{\theta}^{(k)}$$

8. Klasyfikator minimalnej odległości

Jest to specyficzny przypadek znacznie uproszczonego klasyfikatora Bayesa, w którym rozkład klas jest jednorodny (klasyfikator ML) rozkład warunkowy cech jest rozkładem normalnym Gaussa, zaś zamiast miary probabilistycznej stosowana jest miara odległości Euklidesa,

$$d(\Omega_k , \boldsymbol{c}) = \ln p(\Omega_k | \boldsymbol{c}).$$

Reguła decyzyjna: wybierz klasę najbliższą klasyfikowanej próbce.

Załóżmy, że dla problemu 2 klas i próbki $\mathbf{c}$ zachodzi:

$$p(\Omega_1)\, p(\mathbf{c} \mid \Omega_1) > p(\Omega_2)\, p(\mathbf{c} \mid \Omega_2)$$

Przy założeniu rozkładu Gaussa dla $p(\mathbf{c} \mid \Omega)$ otrzymamy:

$$\ln p(\Omega_1) - \frac{1}{2}(c-\mu_1)^T \Sigma_1^{-1}(c-\mu_1) > \ln p(\Omega_2) - \frac{1}{2}(c-\mu_2)^T \Sigma_2^{-1}(c-\mu_2)$$

Przy założeniu jednorodnego rozkładu klas, zachodzi $p(\Omega_1) = p(\Omega_2)$.

W szczególnym przypadku $(\Sigma_1 = \Sigma_2 = \mathrm{I})$ równanie upraszcza się do:

$$|c-\mu_1|^2 < |c-\mu_2|^2$$

czyli do reguły decyzyjnej stosującej miarę Euklidesowa.

9. Klasyfikator k-NN

Uczenie

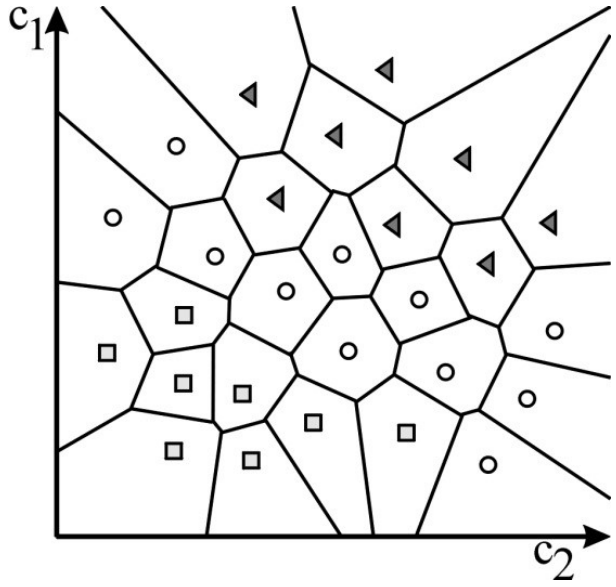
Zbiór próbek uczących, $c^i \in \mathbf{C}=\{c^1, c^2, \ldots, c^n\}$, etykietowanych przynależnością do klasy $\zeta(c^i)$, zostaje **zapamiętany** jako zbiór reprezentantów klas. Niejawnie wyznaczony zostaje **podział Voronoia** przestrzeni cech: obszar przestrzeni cech, dla punktów którego najbliższym reprezentantem jest próbka c^i tworzy komórkę Voronoia o reprezentancie c^i.

Reguła decyzyjna *k*-NN $\zeta(c)$:

wybierz klasę najczęściej występującą

Wśród etykiet klas dla *k* najbliższych

sąsiadów klasyfikowanej próbki c.



10. Klasyfikator SVM - maszyna wektorów wspierających

SVM („*Support Vector Machine*”) jest klasyfikatorem binarnym. Problem klasyfikacji wielu klas sprowadza się do wielokrotnej decyzji pomiędzy dwiema klasami (lub grupami klas) (oznaczanymi zwykle jako „+1”, „-1”).

Podczas procesu uczenia maszyny SVM poszukuje się optymalnej hiperpłaszczyzny (dla przypadku liniowego) lub hiperpowierzchni (dla nieliniowej maszyny).

Rozmiar Vapnika-Czervonenkisa (rozmiar VC) h jest miarą zbioru funkcji rozdzielających. Dla problemu dwóch klas h wyznacza maksymalną liczbę punktów, które mogą być rozdzielone do 2 klas na wszystkie możliwe sposoby – liczba takich podziałów wynosi 2^h.

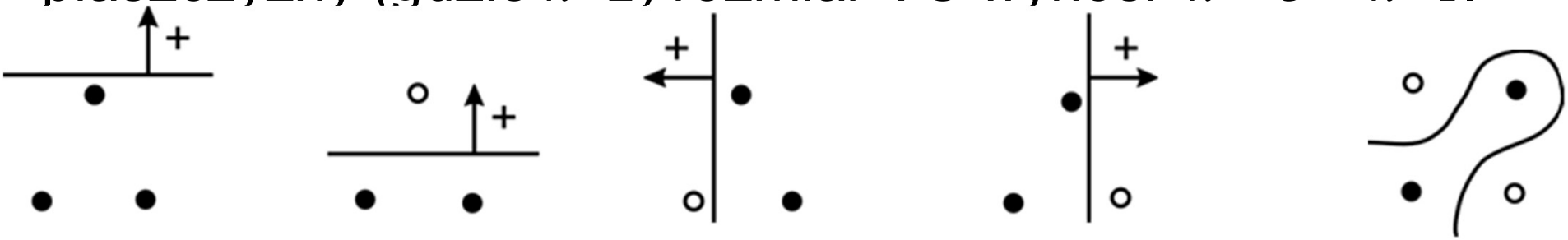
Specyficznym zbiorem funkcji rozdzielających jest zbiór zorientowanych **hiperpłaszczyzn**: $d_{\tilde{\mathbf{a}}}(\mathbf{c}) = \mathbf{c}^T\mathbf{a} + a_0$, $\tilde{a} = \begin{pmatrix} a_0 \\ \mathbf{a} \end{pmatrix}$

Rozmiar VC

Punkt $\mathbf{c}$ może leżeć po „dodatniej” stronie lub „ujemnej” stronie płaszczyzny lub na płaszczyźnie rozdzielającej. **Liczbę punktów** potrzebnych dla wyznaczenia hiperpłaszczyzny podaje rozmiar VC danej przestrzeni cech.

Twierdzenie. Rozmiar VC zbioru zorientowanych **hiperpłaszczyzn** w przestrzeni $\Re^n$ wynosi $h = n + 1$.

Przykład. Na płaszczyźnie zbiór trzech punktów da się rozdzielić za pomocą jednej zorientowanej prostej na możliwe $2^3 = 8$ sposobów. Ale 4 punkty już nie. Czyli dla płaszczyzny (gdzie n=2) rozmiar VC wynosi $h = 3 = n+1$.



Liniowy klasyfikator SVM

- Zakładamy istnienie zbioru próbek dwóch klas **liniowo separowalnych**. Niech istnieje N próbek uczących zaklasyfikowanych jako „dodatnie próbki” $(y_i=1)$ lub „negatywne próbki” $(y_i = -1), j=1,2,...,N.$
- Jeśli istnieje hiperpłaszczyzna o parametrach $\mathbf{a} = [a_0, a_1, ..., a_n]^T$, która rozdziela próbki uczące to zachodzi:

$$\begin{aligned} {}^j\mathbf{c}^T\mathbf{a} + a_0 &\geq +1, \quad if \quad y_j = +1 \\ {}^j\mathbf{c}^T\mathbf{a} + a_0 &\leq -1, \quad if \quad y_j = -1 \\ y_j({}^j\mathbf{c}^T\mathbf{a} + a_0) &\geq +1, \quad \forall\, {}^j\mathbf{c} \in \omega \end{aligned}$$

- Hiperpłaszczyzny $\mathbf{c}^T\mathbf{a} + a_o = 1$ i $\mathbf{c}^T\mathbf{a} + a_o = -1$

posiadają ten sam wektor normalny $\mathbf{a}$, a ich wzajemny odstęp wynosi $2 / |\mathbf{a}|$.

Optymalna hiper-płaszczyzna

Optymalna płaszczyzna rozdzielająca to H*. Istnieją inne płaszczyzny rozdzielające (np. H') ale nie są one optymalne.

$$H^*: d_{\tilde{\mathbf{a}}}(\mathbf{c}) = \mathbf{c}^T\mathbf{a} + a_0 = 0$$

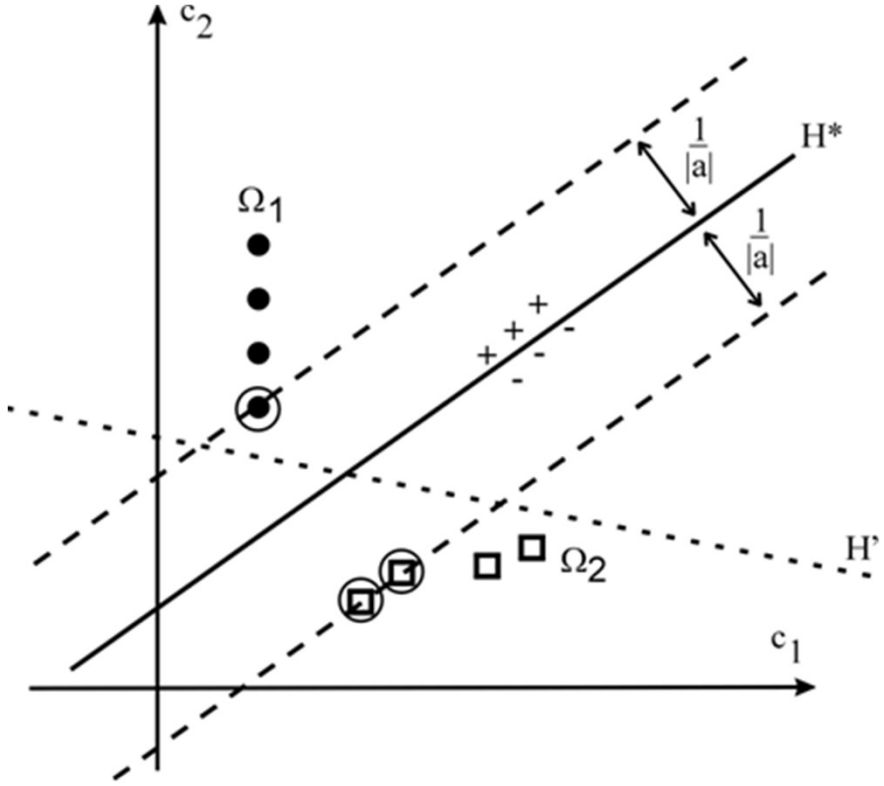
Dla liniowo separowalnego zbioru próbek należących do 2 klas H* wynika z minimalizacji kryterium

$$\min_{\tilde{\mathbf{a}}\in R^{n+1}} f(\tilde{\mathbf{a}}) = \min_{\mathbf{a}\in R^n}(\frac{1}{2}|\mathbf{a}|^2)$$

przy *N* warunkach dodatkowych:

$$C_j(\mathbf{a}) \geq 0\ , \qquad j = 1, \ldots, N;$$

tzn. $y_j({}^j\mathbf{c}^T\mathbf{a} + a_0) - 1 \geq 0, \quad \forall\ {}^j\mathbf{c} \in \Omega$



Reguła decyzyjna jest postaci:

$$\zeta(c) = \begin{cases} \Omega_1, & gdy\ d_{\tilde{\mathbf{a}}}(c) \geq 0 \\ \Omega_2, & gdy\ d_{\tilde{\mathbf{a}}}(c) < 0 \end{cases}$$

Programowanie kwadratowe

Uczenie klasyfikatora SVM wymaga rozwiązania problemu **wypukłego programowania kwadratowego** z nierównościowymi ograniczeniami liniowymi.

Ograniczenia zostaną uwzględnione w funkcji celu po wprowadzeniu mnożników Lagrange'a: $\vartheta = (\vartheta_1, ..., \vartheta_N)^T$.

Zmodyfikowana funkcja celu:

$$L(\tilde{\mathbf{a}}, \vartheta) = \frac{1}{2} |\mathbf{a}|^2 - \sum_{j=1}^{N} \vartheta_j (y_j ({}^j\mathbf{c}^T \mathbf{a} + a_0) - 1)$$

Warunki konieczne (dla liniowych ograniczeń także wystarczające) **Karusha-Kuhna-Tuckera** dla optymalizacji funkcjonału L wynoszą:

$$\frac{\partial L}{\partial \mathbf{a}} = \mathbf{a} - \sum_{j=1}^{N} \vartheta_j y_j {}^j\mathbf{c} = \mathbf{0}$$

$$\frac{\partial L}{\partial a_0} = -\sum_{j=1}^{N} \vartheta_j y_j = \mathbf{0}$$

$$0 \le \vartheta_j, \quad j = 1, \cdots, N$$

$$0 \le y_j ({}^j\mathbf{c}^T \mathbf{a} + a_0) - 1, \quad j = 1, \cdots, N$$

$$0 = \vartheta_j (y_j ({}^j\mathbf{c}^T \mathbf{a} + a_0) - 1), \quad j = 1, \cdots, N$$

Uczenie SVM

Dualne zadanie Lagrange'a - poszukiwanie maksimum dualnej funkcji, dla której zanikają pochodne względem $\mathbf{a}$ i a_0 :

$$L_D(\vartheta) = \inf_{\tilde{\mathbf{a}}} L(\tilde{\mathbf{a}}, \vartheta)$$

Uzyskujemy dualną postać **Wolfe'a**:

$$L_D(\vartheta) = \sum_{j=1}^{N} \vartheta_j - \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{N} \vartheta_j \vartheta_k y_j y_k ({}^{j}\mathbf{c}^{T} \; {}^{k}\mathbf{c})$$

$$0 \le \vartheta_j \; ; \; 0 = \sum_{j=1}^{N} \vartheta_j y_j$$

1. Należy rozwiązać postać Wolfe'a znajdując taki h-elementowy podzbiór próbek uczących $(h = n+1)$, który daje maksymalną wartość marginesu $1/|\mathbf{a}|$.
2. Mając wektor mnożników ϑ odpowiadający h próbkom, wartości parametrów $\mathbf{a}$ uzyskujemy z równań Karush-Kuhn-Tucker, a a_0 z warunków dodatkowych.

Wektory wspierające

Wstawiając wektor parametrów $\mathbf{a}$ wyliczonych z równań Karush-Kuhn-Tucker do równania optymalnej hiper-płaszczyzny H* uzyskamy jej postać zależną jedynie od produktu skalarnego znalezionych h próbek (tzw. **wektorów wspierających**):

$$H*: d_{\tilde{\mathbf{a}}}(\mathbf{c}) = \sum_{j=1}^{N} \vartheta_j y_j (\mathbf{c}^T \, {}^j\mathbf{c}) + a_0 = 0$$

Dla próbek (wektorów cech) leżących dokładnie na hiper-płaszczyźnie H* mnożniki wynoszą: $\vartheta_j=0$ lub $\vartheta_j>0$, a dla próbek nie leżących na hiperpłaszczyźnie, mnożniki są wyzerowane: $\vartheta_j=0$). Do obliczenia optymalnej hiper-płaszczyzny wchodzi tylko h wektorów cech o aktywnych warunkach dodatkowych ($\vartheta_j>0$) **– są to wektory wspierające** - wszystkie pozostałe wektory są zbędne.

Liniowa SVM z zakłóceniami

Jeśli zbiór próbek **nie jest** liniowo separowalny, ale przekłamania są stosunkowo nieliczne, nadal warto stosować liniową SVM. Należy jednak rozszerzyć funkcję celu o dodatkową składową kary za próbki „położone” po złej stronie hiperpłaszczyzny rozdzielającej:

$$\min_{\tilde{a}\in R^n} f(\tilde{\mathbf{a}}) = \min_{\tilde{a}\in R^n}(\frac{1}{2}|\tilde{\mathbf{a}}|^2 + \mathrm{C}\sum_{j=1}^{N}\varepsilon_j)$$

gdzie C jest ustalonym parametrem, ważącym wpływ szumu na łączną funkcję celu, a ε_i to błąd i-tej próbki (jeśli jest po złej stronie to odległość od hiperpłaszczyzny dla danej klasy).

Zamiast $n+1$ zmiennych, mamy teraz $n+1+N$ zmiennych w problemie pierwotnym.

Problem dualny ma identyczną postać co poprzednio, za wyjątkiem nowego ograniczenia mnożników Lagrange’a: $0 \le \vartheta_j \le \mathrm{C}$.

Nieliniowa SVM

Jeśli zbiór próbek zdecydowanie nie spełnia warunków liniowej SVM, to możemy zastosować przekształcenie w wyżej wymiarową przestrzeń $F(\mathbf{c})$, w której próbki mogą być liniowo separowalne.

Uzyskujemy równanie hiperpłaszczyzny w tej wyżej wymiarowej przestrzeni po zastąpieniu wektorów $\mathbf{c}$ i ${}^{j}\mathbf{c}$ przez $F(\mathbf{c})$ i $F({}^{j}\mathbf{c})$:

$$H^{*}: d_{\tilde{\mathbf{a}}}(\mathbf{c}) = \sum_{j=1}^{N} \vartheta_j y_j \left(F(\mathbf{c})^T \, F({}^{j}\mathbf{c})\right) + a_0 = 0$$

Pytania (1)

1. Wymienić i omówić istotę trzech podstawowych sposobów indukcyjnego uczenia.
2. Omówić tryby podawania próbek uczących.
3. Omówić podstawowe algorytmy grupowania.
4. Przedstawić zasadę uczenia sieci Kohonena.
5. Na czym polega przestrzeń wersji?
6. Omówić algorytm uczenia się pojęcia (CEA).
7. Przedstawić klasyfikator stosujący drzewo decyzyjne.
8. Omówić algorytm uczenia drzewa decyzyjnego DTL.

Pytania (2)

9. Przedstawić klasyfikator Bayesa i zasadę jego uczenia.
10. Omówić klasyfikator minimalno-odległościowy.
11. Przedstawić klasyfikator kNN.
12. Omówić zastosowanie programowania kwadratowego do uczenia klasyfikatora SVM.