Metody rozpoznawania obrazów

Rozpoznawanie Obrazów
Krzysztof Krawiec
Instytut Informatyki, Politechnika Poznańska

Metody rozpoznawania obrazów

(przypomnienie z wykładu 2)
Podziały:

- globalne i lokalne,
- wektorowe i strukturalne,
- bezpośrednie i pośrednie,
- oparte na cechach i na modelu,
- wykorzystujące uczenie lub nie,
- z rozpoznawaniem sterowanym obrazem lub modelem,
- z rozpoznawaniem skupionym na obiekcie lub widoku,
- *constrained* i *unconstrained*,

2

Metody wektorowe

(przypomnienie i uzupełnienia)

Klasyfikatory

Omawiane w ramach przedmiotu "Uczenie Maszynowe", w szczególności:

- Metody minimalnoodległościowe
- Metody probabilistyczne
- Sieci neuronowe
- Podejścia symboliczne: reguły i drzewa decyzyjne.

Tradycyjnie w RO przeważają podejścia oparte o funkcje podobieństwa/odległości oraz metody statystyczne/probabilistyczne.

4

Metody wektorowe statystyczne

Dominujące
- klasyfikatory Bayesowskie,
- Support Vector Machines,

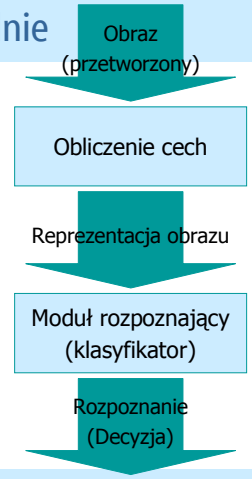
Zagadnienia znane z uczenia maszynowego:
- klątwa wymiarowości (*curse of dimensionality*),
- selekcja cech (*feature selection*),
- konstrukcja cech (*feature construction*)
  - np. PCA

5

Podejścia niebezpośrednie

Problem: zamieszczony schemat jest niepełny, brakuje w nim dodatkowych elementów, np.
- jeżeli cecha obiektu nie spełnia ograniczeń to...

Problem w RO:
- brak uniwersalnych modeli dla kodowania/reprezentacji takiej wiedzy



6

Metody strukturalne

Rozpoznawanie Obrazów

Podejście strukturalne - etapy

1. Dekompozycja obrazu na składowe pierwotne (składowe strukturalne),
2. Analiza poszczególnych składowych pierwotnych i relacji zachodzących pomiędzy nimi.

8

Podejście strukturalno - wektorowe

Idea: „Ręczna" agregacja cech składowych - wykorzystanie miar statystyki opisowej (tendencji centralnej i rozproszenia), np.:

- odchylenie standardowe cechy (np. pola powierzchni)
- momenty,
- wielowymiarowe (warunkowe) rozkłady prawdopodobieństwa, np. *liczba obiektów, których pole powierzchni* = ...

Zaleta: prostota koncepcyjna.

Wady: doraźny charakter, częściowe lub całkowite zaniedbanie aspektu strukturalnego.

9

Podejścia syntaktyczne

Idea

- Obraz jest słowem; aby go zaklasyfikować, trzeba sprawdzić, czy należy do pewnego języka.
- Założenie: Obraz składa się ze (wcześniej zlokalizowanych) *składowych pierwotnych* (*primitives*, prymitywy), które pozostają ze sobą w pewnych *relacjach*.
- Definiuje się produkcje, gramatyki, i rozpoznające słowa automaty skończone.

11

Rodzaje podejść syntaktycznych

Typy:

- ciągowe
  - kody łańcuchowe Freemana
  - język opisu obrazu (PDL, Shaw); bardziej złożone, bo definiuje operatory łączące prymitywy
- drzewowe
  - T – drzewa proste (bez etykiet i skierowania)
  - EDT – *edge-labelled directed tree*
- grafowe

  wykorzystywane raczej do opisu obrazów, mniej do rozpoznawania

12

Syntaktyczne rozpoznawanie ciągów

Gramatyka generująca ciągi (*string grammar*):

$$G = (N, \Sigma, P, S)$$

gdzie:

- N - skończony zbiór symboli nieterminalnych,
- Σ - skończony zbiór symboli terminalnych,
- P - zbiór produkcji,
- $S \in N$ - symbol początkowy

Notacja: V* - zbiór wszystkich ciągów (zdań) złożonych z symboli z V

13

Gramatyki

Gramatyki wykorzystywane szczególnie często w rozpoznawaniu obrazów:

- regularne: zawierają jedynie produkcje typu:
  - $A \rightarrow aB$
  - $A \rightarrow a$
    (A,B - nieterminale, b - terminal)
- bezkontekstowe:
  - $A \rightarrow \alpha$ $\quad \alpha \in (N \cup \Sigma)^* \setminus \lambda$
    (A - nieterminal, λ - pusty ciąg)

14

Przykład

a

b

c

$$N = \{A, B, S\}$$
$$\Sigma = \{a, b, c\}$$
$$P = \{S \rightarrow aA, A \rightarrow bA, A \rightarrow bB, B \rightarrow c\}$$

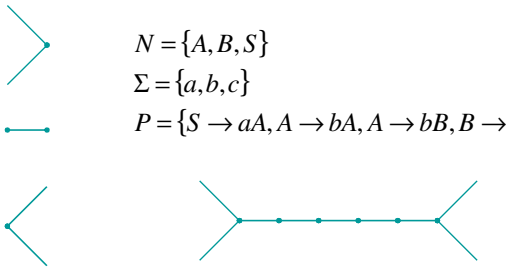
15

Przykład

a

b

c



$$N = \{A, B, S\}$$
$$\Sigma = \{a, b, c\}$$
$$P = \{S \rightarrow aA, A \rightarrow bA, A \rightarrow bB, B \rightarrow c\}$$

Wywód: S⇒aA⇒abA⇒abbA⇒...⇒abbbbbA ⇒abbbbbc

⇒ - krok wywodu

$$L(G) = \{ab^n c \mid n \geq 1\}$$

16

Semantyka

Czasami dogodnie jest stworzyć bazę danych zawierającą *reguły semantyczne*, formułujące pewne warunki narzucane na składowe pierwotne (SP), w tym:

- reguły dotyczące dopuszczalnych sposobów łączenia SP (np. łączenie tylko na końcach),
- wielkość SP,
- orientacja SP,
- ograniczenia na liczbę użyć danej produkcji w wywodzie (<, >).

17

Semantyka - przykład

1) $S \rightarrow aA$
2) $A \rightarrow bA$
3) $A \rightarrow bB$
4) $B \rightarrow c$

Reguły semantyczne użyte niejawnie w wywodzie w poprzednim przykładzie:

1) połączenia tylko w punktach; kierunek a = dwusieczna kąta wyznaczonego "ramionami"; długość 3cm
2) połączenia tylko w punktach; wielokrotne połączenia są niedozwolone; kierunek b musi być taki sam jak kierunek a; długość b 0.25cm; wolno stosować co najwyżej 10 razy
3) kierunki a i b muszą być zgodne; połączenia tylko w punktach;
4) kierunki b i c muszą być zgodne; połączenia tylko w punktach;

18

Gramatyki a automaty

- Zachodzi wzajemnie jednoznaczna odpowiedniość pomiędzy gramatykami regularnymi a automatami skończonymi.
- Dla każdej gramatyki regularnej G da się zbudować automat skończony który akceptuje jedynie słowa z L(G).

19

Automaty do rozpoznawania ciągów

Automat skończony:

$$A_f = (Q, \Sigma, \delta, q_0, F)$$

gdzie:

- Q - skończony niepusty zbiór stanów,
- Σ - skończony alfabet wejściowy,
- δ - mapowanie: $Q \times \Sigma \rightarrow 2^Q$ (Uwaga! zbiór)
- q_0 - stan początkowy,
- $F \subset Q$ - zbiór stanów końcowych (akceptujących)

20

Konstrukcja automatu dla gramatyki

Oznaczmy nieterminale gramatyki przez X_i

$$G = (N, \Sigma, P, X_0),\ N = \{X_0, X_1, \ldots, X_n\}$$

Zbiór stanów Q konstruowanego automatu:

$$Q = \{q_0, q_1, \ldots, q_n, q_{n+1}\}$$

gdzie:

- q_i odpowiada X_i dla i=0..n
- q_{n+1} jest stanem końcowym

Alfabet wejściowy Σ - tożsamy ze zbiorem terminali gramatyki.

Konstrukcja automatu dla gramatyki

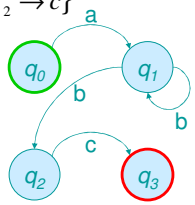
Mapowanie δ: dla każdego i,j=0..n

- jeżeli P zawiera produkcję $X_i \rightarrow aX_j$, to $\delta(q_i,a)$ zawiera q_j,
- jeżeli P zawiera produkcję $X_i \rightarrow a$, to $\delta(q_i,a)$ zawiera q_{n+1} (stan końcowy)

Konstrukcja automatu - przykład

$$N = \{A, B, S\}$$
$$\Sigma = \{a, b, c\}$$
$$P = \{S \rightarrow aA, A \rightarrow bA, A \rightarrow bB, B \rightarrow c\}$$
$$P = \{X_0 \rightarrow aX_1, X_1 \rightarrow bX_1, X_1 \rightarrow bX_2, X_2 \rightarrow c\}$$

$$Q = \{q_0, q_1, q_2, q_3\}, \Sigma = \{a, b, c\}, F = \{q_3\}$$
$$\delta(q_0, a) = \{q_1\}$$
$$\delta(q_1, b) = \{q_1, q_2\}$$
$$\delta(q_2, c) = \{q_3\}$$

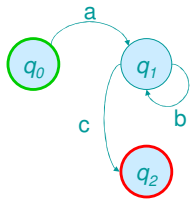


Dla wszystkich pozostałych przejsc δ(_,_)=∅

Konstrukcja automatu – przykład – wer. II

$$N = \{A, B, S\}$$
$$\Sigma = \{a, b, c\}$$
$$P = \{S \rightarrow aA, A \rightarrow bA, A \rightarrow bB, B \rightarrow c\}$$
$$P = \{X_0 \rightarrow aX_1, X_1 \rightarrow bX_1, X_1 \rightarrow c\}$$

$$Q = \{q_0, q_1, q_2\}, \Sigma = \{a, b, c\}, F = \{q_2\}$$
$$\delta(q_0, a) = \{q_1\}$$
$$\delta(q_1, b) = \{q_1\}$$
$$\delta(q_1, c) = \{q_2\}$$



Dla wszystkich pozostałych przejsc δ(_,_)=∅

Gramatyki drzewowe

Syntaktyczne rozpoznawanie drzew

Gramatyka generująca drzewa (*tree grammar*):

$$G = (N, \Sigma, P, r, S)$$

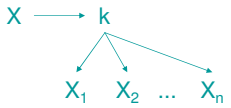
gdzie:

- N - skończony zbiór symboli nieterminalnych,
- Σ - skończony zbiór symboli terminalnych,
- P - zbiór produkcji; produkcje mają postać:
  - $T_i \rightarrow T_j$, gdzie T_i i T_j są drzewami,
- r - funkcja rangująca (*ranking function*), podająca liczbę bezpośrednich potomków terminala
- $S \in N$ - symbol początkowy (może być drzewem)

26

Syntaktyczne rozpoznawanie drzew

Szczególny przypadek, najczęściej wykorzystywany w rozoznawaniu obrazów: drzewowe gramatyki ekspansywne (*expansive tree grammars*), w których wszystkie produkcje są postaci

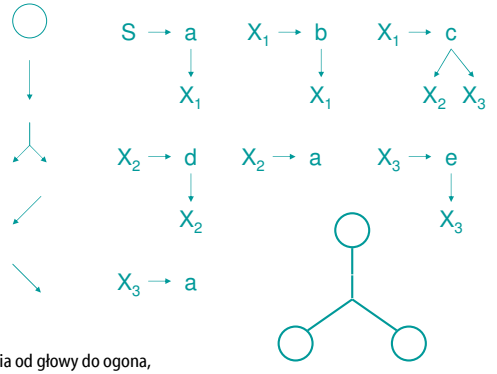


gdzie X_i to nieterminale, a *k* to terminal.

27

Przykład

- a
- b
- c
- d
- e



Założenia:

- połączenia od głowy do ogona,
- dla *a*: dowolnie (na obwodzie)

28

Przykład

Funkcja rangująca:

$$r(a) = r(e) = \{0,1\}$$
$$r(b) = r(d) = \{1\}$$
$$r(c) = \{2\}$$

Przydatna reguła semantyczna:

- wymaganie, aby produkcje 2, 4 i 6 były wykorzystywane taką samą liczbę razy - da w wyniku obiekt z trzema "nogami" o równej długości.

29

Automat drzewowy (*tree automata*)

Zasadnicza różnica: o ile automat analizujący ciągi znaków przetwarza je od lewej do prawej, automat drzewowy musi równolegle analizować drzewo od liści (skrajnych składowych pierwotnych figury) do korzenia.

30

Automat drzewowy (*tree automata*)

Automat drzewowy (*frontier-to-root*):

$$A_t = (Q, F, \{f_k \mid k \in \Sigma\})$$

gdzie:

- Q - skończony niepusty zbiór stanów,
- Σ - skończony alfabet wejściowy,
- F⊂Q - zbiór stanów końcowych (akceptujących)
- f_k jest relacją w $Q^m \times Q$ taką, że m jest *rangą* terminala k.

31

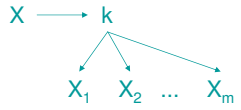
Konstrukcja automatu drzewowego

Dla ekspansywnej gramatyki drzewowej

$$G = (N, \Sigma, P, r, S)$$

konstruujemy automat przyjmując:

$$Q = N$$
$$F = \{S\}$$
$$\forall k \in \Sigma \; f_k : (X_1, X_2, \ldots, X_m, X) \in f_k \Leftrightarrow G \text{ zawiera produkcję}$$



32

Przykład

$G = (N, \Sigma, P, r, S)$

$N = \{S, X\}$

$\Sigma = \{a, b, c, d\}$

$r(a) = \{0\},\ r(b) = \{0\},\ r(c) = \{1\},\ r(d) = \{2\}$

$Q = \{S, X\}$

$F = \{S\}$

$\{f_k \mid k \in \Sigma\} = \{f_a, f_b, f_c, f_d\}$

$f_a = f_b = \{(\varnothing, X)\}$

$f_c = \{(X, X)\}$

$f_d = \{(X, X, S)\}$

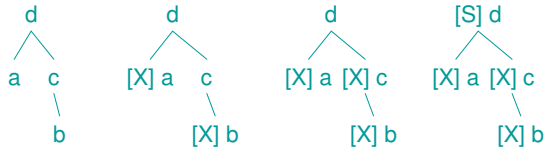
X → a X → b X → c S → d

X X X

Interpretacja:

- fa - węzłowi o etykiecie a bez potomków przypisywany jest stan X
- fd - węzłowi o etykiecie d z dwoma potomkami, każdym mającym stan X, przypisany jest stan S

Przykład



Automat akceptuje (rozpoznaje) to drzewo, ponieważ ostatni analizowany węzeł jest redukowany do stanu S, który jest w zbiorze F stanów końcowych.

Uczenie w podejściach syntaktycznych

- Jawne wyspecyfikowanie gramatyki przez projektanta systemu jest zazwyczaj nierealne.
- Pożądane: możliwość uczenia się (konstruowania) automatów na podstawie przykładowych wzorców (ciągów, drzew, etc.).
- Ponieważ zachodzi odpowiedniość automatów i gramatyk, problem ten sprowadza się do problemu uczenia się gramatyk z przykładów (tzw. wywodzenie gramatyk, *grammatical inference*).

Uczenie w podejściach syntaktycznych

Załóżmy że

- wszystkie wzorce należące do klasy, którą chcemy rozpoznawać, generowane są przez nieznaną gramatykę G,
- mamy do dyspozycji skończony zbiór uczący (przykładów pozytywnych)

$$R^+ \subseteq \{\alpha \mid \alpha \in L(G)\}$$

O zbiorze R^+ powiemy że jest *strukuralnie kompletny* jeżeli każda produkcja z G jest wykorzystywana do wygenerowania przynajmniej jednego przykładu uczącego z R^+.

Cel: synteza skończonego automatu A_f który będzie akceptował ciągi z R^+ oraz (być może) pewne ciągi *podobne* do R^+ (indukcja!).

Algorytm

Zachodzi:

$$R^+ \subseteq \Sigma^*$$

Niech $z \in \Sigma^*$ będzie stringiem takim, że $zw \in R^+$ dla pewnego $w \in \Sigma^*$.

Dla naturalnego *k* zdefiniujmy k-ogon (*k tail*) ciągu *z* ze względu na R^+ jako

$$h(z, R^+, k) = \{w \mid zw \in R^+, |w| \leq k\}$$

(zbiór wszystkich ciągów z w/w właściwościami).

Algorytm

$$Q = \{q \mid q = h(z, R+, k) \; dla \; z \in \Sigma^*\}$$

$$\forall a \in \Sigma : \delta(q, a) = \{q' \in Q \mid q' = h(za, R^+, k), q = h(z, R+, k)\}$$

$$q_0 = h(\lambda, R^+, k)$$

$$F = \{q \mid q \in Q, \lambda \in q\}$$

Przykład

Krok 1: Generowanie stanów

$R^+ = \{a, ab, abb\}, \; k = 1$

$z = \lambda \quad h(\lambda, R^+, 1) = \{q \mid \lambda w \in R^+, |w| \leq 1\} = \{a\} = q_0$

$z = a \quad h(a, R^+, 1) = \{q \mid aw \in R^+, |w| \leq 1\} = \{\lambda, b\} = q_1$

$z = ab \quad h(ab, R^+, 1) = \{\lambda, b\} = q_1$

$z = abb \quad h(abb, R^+, 1) = \{\lambda\} = q_2$

plus dodatkowy stan $q_\varnothing$ odpowiadający sytuacji, gdy *h* jest zbiorem pustym. Zatem: $Q = \{q_0, q_1, q_2, q_\varnothing\}$

Krok 2: Funkcja przejść (nie omówiona)

Podejścia syntaktyczne - zalety i wady

Zalety:

- elegancki formalizm,
- podejście 'całościowe'.

Wady:

- problemy z uwzględnianiem zaszumienia danych (obrazów) i danymi brakującymi,
- problemy z automatycznym generowaniem gramatyk z przykładów, w tym:
  - problemy z aktualizacją gramatyki/automatu gdy pojawiają się nowe przykłady.
- złożoność obliczeniowa analizy syntaktycznej: NP (dla większości klas gramatyk),

Konkurencja:

- Ukryte modele/łańcuchy Markowa (*Hidden Markov Models*, HMM)

Przykłady zastosowań

- analiza scen naturalnych,
- analiza pisma ciągłęgo (ang. *dynamic handwritting*),
- analiza trajektorii cząstek elementarnych w komorze pęcherzykowej (Wilsona),
- kontrola jakości w przemyśle (*machine vision*)

41

Podejścia relacyjne

Podejścia relacyjne

Idea: składowe pierwotne w obrazie pozostają ze sobą w pewnych relacjach, ale nie wymagamy, aby relacje te dały się opisywać gramatykami formalnymi.

Dominujące podejście:

- opis scen rzeczywistych w językach logiki (Prolog, Progol, etc.)
- uczenie w tego typu podejściach: ILP (inductive logic programming); automatyczna indukcja programów w logice.

43

Podejścia relacyjne

Ciekawostka:

- specjalizowane sztuczne sieci neuronowe do przetwarzania informacji strukturalnej
- *neural folding architecture* (Schmitt, Goller),
  - *backpropagation through structure*,
  - zastosowanie w QSAR (*quantitative structure-activity relationship*), istotny etap w projektowaniu leków.
- Frasconi, Gori, Sperdutti
  - *backpropagation through structure* jako analogia do *backpropagation through time*

44

Podejścia relacyjne

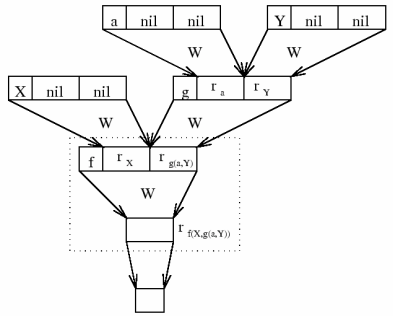


Fig. 3: The Encoder unfolded by the structure $f(X, g(a, Y))$.

45