Wstęp do inteligencji komputerowej – zajęcia nr 3

Jarosław Stańczak
WSISiZ

Uczenie maszynowe:

- uczenie z nadzorem i bez nadzoru
- uczenie się ze wzmocnieniem
- przykład zastosowania uczenia ze wzmocnieniem
- uczenie Bayesowskie

Uczenie maszynowe

Uczenie maszynowe (UM, *ang. machine learning*, ML) jest dość szerokim pojęciem, gdyż właściwie można pod nie podciągnąć wszystkie dziedziny sztucznej inteligencji, a już na pewno te, w których następuje pozyskiwanie nowych faktów, danych, wiedzy, adaptacja do zmiennych warunków i wszelkie przejawy modyfikacji i poprawy sposobu działania na skutek interakcji z otoczeniem.
Zacznijmy więc od definicji uczenia się (na podst. P. Cichosz „Systemy uczące się”):

„Uczeniem się systemu jest każda autonomiczna zmiana w systemie zachodząca na podstawie doświadczeń, która prowadzi do poprawy jakości jego działania.”

Nie jest to jedyna definicja uczenia, powstało ich dużo więcej.

Uczenie maszynowe

R. Michalski w 1986 r. podał taką definicję uczenia:

"Uczenie się to konstruowanie i zmiana reprezentacji doświadczanych faktów. W ocenie konstruowania reprezentacji bierze się pod uwagę: wiarygodność – określa stopień, w jakim reprezentacji odpowiada rzeczywistości, efektywność – charakteryzuje przydatność reprezentacji do osiągania danego celu, poziom abstrakcji – odpowiada zakresowi szczegółowości i precyzji pojęć używanych w reprezentacji; określa on tzw. moc opisową reprezentacji. Reprezentacja jest rozumiana jako np. opisy symboliczne, algorytmy, modele symulacyjne, plany, obrazy."

Uczenie się

W systemach uczących się mamy pewien element, który zmienia sposób swojego działania w miarę zachodzenia interakcji z otoczeniem.
Jak to wynika z przedstawionej wcześniej definicji uczenia się, procesem uczenia nie będzie każda sytuacja zmiany w sposobie działania, ale tylko te, które prowadzą do poprawy jego działania, bo np. czy urządzenie, które właśnie się zepsuło i na pewno zmieniło swój sposób działania w stosunku do urządzenia sprawnego, zrobiło to jako efekt uczenia się? Raczej nie. Nie można zatem traktować każdej zmiany jako efektu uczenia się, choć oczywiście przypadkowe sytuacje też mogą być podstawą uczenia się. Uczenie się będzie z pewnością zmianą, która w sposób trwały (pamięć!) pozytywnie zmieni zachowanie systemu w wyniku zadziałania wbudowanych weń mechanizmów do analizy i oceny otoczenia.
Z tego wynika, że system uczący się musi mieć jakąś funkcję oceny, która kierunkuje proces uczenia i sprawia, że system po procesie uczenia działa lepiej zgodnie z tą funkcją oceny.

Metody zapisu nauczonej wiedzy

Uczenie się wymaga pamięci. Pozyskaną wiedzę należy zmagazynować. Istotna jest nie tylko ilość, ale też forma w jakiej tę wiedzę się przechowuje. Sposób przechowywania wiedzy zależy w dużej mierze od metody uczenia i późniejszego jej wykorzystania. Zazwyczaj wiedzę pamięta się jako:

- zbiory reguł składających się z części warunkowej i konkluzji
- drzew decyzyjnych, składających się z decyzji i stanów
- klauzul zapisanych w logice predykatów (uogólnienie reguł)
- rozkładów prawdopodobieństwa pewnych zdarzeń
- funkcji przejść automatów skończonych
- gramatyk formalnych
- sieci semantycznych
- wag w sieciach neuronowych
- strojonych parametrach algorytmów

Metody te były wykorzystywane m.in. w zapisie faktów w systemach ekspertowych, omawianych na poprzednim wykładzie.

Uczenie z nadzorem i bez

W przypadku **uczenia z nadzorem** posiadamy pewne przykłady wraz z ich interpretacją, pochodzącą np. od eksperta w danej dziedzinie, a interpretacja układu uczącego się podlega ocenie, przy czym ocena ta jest tym wyższa (lepsza), im odpowiedź układu jest bliższa interpretacji narzuconej w przykładzie trenującym. Sytuacja jest tu podobna do układu nauczyciel-uczeń.

W przypadku **uczenia bez nadzoru** system uczący otrzymuje tylko przykłady, bez ich interpretacji i sam musi je odpowiednio zinterpretować (najczęściej pogrupować). Funkcja oceny jest tu więc włączona w algorytm przetwarzania danych przetwarzanych/trenujących. Algorytm analizuje różnice i podobieństwa w danych trenujących i na tej podstawie dzieli je na różne kategorie.

Wiele różnych algorytmów sztucznej inteligencji posiada wbudowane metody uczenia się z nadzorem (np. sieć neuronowa typu BP) lub bez (np. sieć neuronowa Kohonena, metody klasteryzacji).

Metody uczenia maszynowego

Metody uczenia maszynowego najłatwiej jest sklasyfikować na podstawie wykorzystywanych metod uczenia:

- uczenie ze wzmocnieniem,
- uczenie Bayesowskie (metody probabilistyczne),
- uczenie na podstawie przykładów,
- uczenie się (indukcja) zbioru reguł,
- uczenie się (indukcja) drzew decyzyjnych,
- uczenie się funkcji logicznych na podstawie przykładów,
- i wiele innych, włączając także sieci neuronowe, metody ewolucyjne i heurystyki obliczeniowe.

Cele uczenia maszynowego

Możemy zdefiniować następujące cele uczenia maszynowego:

- tworzenie lub odkrywanie nowych pojęć,
- wykrywanie regularności, prawidłowości lub zależności w danych,
- formułowanie/generowanie reguł decyzyjnych (np. na podstawie zbioru faktów),
- przyswajanie nowych pojęć i struktur przy pomocy uogólnienia i analogii,
- modyfikowanie, uogólniania i precyzowania danych,
- zdobywania wiedzy poprzez interakcję z otoczeniem,
- formułowania wiedzy w postaci zrozumiałej dla człowieka.

Programy uczące się

Uczenie maszynowe w każdym rozpatrywanym przypadku i metodzie sprowadzi się w końcu do pewnego ogólnego algorytmu, który na podstawie danych doświadczalnych będzie starał się zmienić tak swoje parametry, aby posiadać lepszą wartość jakiejś funkcji oceny. Najczęściej polega to na strojeniu pewnych parametrów algorytmu (często są to pewne parametry liczbowe) – jest to przykład pozyskiwania tzw. **wiedzy deklaratywnej**, opisującej pewne obiekty, sytuacje i związki (pojęcie to pojawiło się też na poprzednim wykładzie o SE). Można także wyobrazić sobie algorytmy (są one trudniejsze, ale realizowalne), które modyfikują swój sposób działania, strategię działania. Mamy wtedy do czynienia z tzw. **wiedzą proceduralną**, określającą czynności. To drugie podejście jest zdecydowanie rzadziej spotykane z uwagi na trudności teoretyczne i praktyczne modyfikacji działającego programu (tę możliwość wykorzystują niektóre wirusy komputerowe).

Uczenie ze wzmocnieniem

Uczenie ze wzmocnieniem to powszechnie przyjęta nazwa metody uczenia, w której agent (program, maszyna...) uczy się na podstawie interakcji z otoczeniem. Interakcje polegają na tym, że w dyskretnych chwilach czasowych wykonuje on pewne akcje (działania) z pewnego skończonego zbioru możliwych akcji, a następnie bada rekcję otoczenia na nie. Reakcja otoczenia nie musi być natychmiastowa, może być odsunięta w czasie. Ta reakcja otoczenia jest właśnie owym wzmocnieniem (czasem zwanym wypłatą), czyli nagrodą lub karą za podjęte w przeszłości akcje. Jest to więc forma oceny działania agenta za podjęte działania. Akcje agenta wywołują zmiany zarówno w otoczeniu, jak i w nim samym.

Uczenie ze wzmocnieniem

Generalnie idea metody polega na tym, że wzmocnieniu podlegają akcje lub strategie wykonywania akcji, które przynoszą większe wypłaty. Jest to heurystyka dość powszechnie stosowana przez organizmy żywe i ludzi także, np. jeśli twórca wyda płytę, która się dobrze sprzeda, to potem wydaje kolejną, dość podobną – oczywiście jeśli przesadzi, to może osiągnąć wynik odwrotny od zamierzonego...
Podobna sytuacja jest np. z filmami kinowymi: gdy osiągną sukces, to powstają sequele, prequele, wersje 1,5 i wieloma innymi dziedzinami życia.

Uczenie ze wzmocnieniem

Celem uczącego się agenta jest wypracowanie strategii decyzyjnej, która zmaksymalizuje otrzymywane przez niego wypłaty w długim horyzoncie czasowym. Najczęściej stosowane jest tu kryterium o postaci:

$$Q = \max_{a_0, a_1, \ldots} E\left[\sum_{t=0}^{\infty} \gamma^t * r_t\right]$$

czyli maksymalizujemy po wykonywanych akcjach (a_0, a_1,...) wartość oczekiwaną (agent i/lub środowisko mogą być stochastyczne) ze zdyskontowanej (γ - współczynnik dyskontowania, wybierany z przedziału (0,1)) sumy otrzymanych wypłat – wzmocnień. Współczynnik dyskonta jest odpowiedzialny za różnicowanie wpływu wypłat oddalonych w czasie, im będzie ona bliższa, tym większy będzie miała wpływ, gdyż γ wykładniczo maleje wraz z upływem czasu.

Uczenie ze wzmocnieniem

algorytm czasowego przypisania zasługi (Q-learning)

Wzór podany na poprzednim slajdzie nie jest wygodny do bezpośredniego stosowania, jest on zbyt ogólny. Jedną z metod uszczegółowiających ten ogólny schemat są metody czasowego przypisania zasługi, a między innymi metoda Q-learning, w której wyboru akcji dokonuje się na podstawie jak najwyższych wartości funkcji oceny Q:

$$Q_{t+1}(x,a)=\begin{cases}(1-\eta)*Q_t(x_t,a_t)+\eta*(r_t+\gamma*\max\limits_{a'}Q_t(x_{t+1},a')) & \text{dla } x=x_t, a=a_t \\ Q_t(x,a) & \text{dla } x\neq x_t, a\neq a_t\end{cases}$$

η - współczynnik szybkości uczenia, γ - współczynnik dyskontowania, r_t – wzmocnienie (wypłata), t – chwila czasowa, x – stan agenta, a – akcja agenta, $\max\limits_{a'}Q_t(x_{t+1},a')$ - oszacowanie maksymalnej wartości funkcji oceny.

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki

Załóżmy, że chcemy rozwiązać problem komiwojażera: odwiedzić po najkrótszej drodze *n* miast, oczywiście pamiętając o odwiedzeniu wszystkich miast i nieodwiedzaniu żadnego wielokrotnie. Zadanie to jest trudne do rozwiązania, gdyż przestrzeń poszukiwań ma wielkość rzędu *n*!, więc dla liczby miast większej od 100 właściwie niemożliwe jest pełne jej przejrzenie i ocena rozwiązań. Dlatego też do rozwiązywania tego typu zadań stosuje się najczęściej metody heurystyczne (więcej o nich na kolejnych wykładach), a nie metody dokładne.
Metodę Q-learning przetestujemy na takiej najprostszej, uniwersalnej metodzie heurystycznej, służącej do rozwiązywania szerokiej klasy zadań optymalizacyjnych.

Uczenie ze wzmocnieniem

przykład prostego zastosowania w tzw. algorytmie wzrostu

Jedną z najprostszych heurystyk jest tzw. algorytm wzrostu, w którym wylosowane rozwiązanie początkowe (rozwiązanie to np. lista miast w kolejności odwiedzania) poddaje się losowej modyfikacji, a otrzymane rozwiązania (może być ich więcej niż 1) ocenia się i jeśli wśród nich jest lepsze od posiadanego bazowego, akceptuje się jako nowe bazowe, podlegające kolejnej modyfikacji. Tak powtarza się sekwencje modyfikacji i ewentualnych akceptacji, aż osiągniemy kryterium stopu, czyli np. określoną liczbę iteracji.

Uczenie ze wzmocnieniem

przykład prostego zastosowania

podstawowy algorytm metody (największego) wzrostu

begin
  t:=0
  wylosuj rozwiązanie początkowe x_s
  oceń x_s
  x_0:=x_s
  repeat
    wygeneruj n nowych rozwiązań x_1...x_n przez modyfikację x_0 operatorem perturbacji
    oceń rozwiązania x_1...x_n
    wybierz najlepsze rozwiązanie x^* z x_1...x_n
    jeśli x^* jest lepsze od x_0 to x_0:=x^*
    t:=t+1
  until t=MAX
end

Uczenie ze wzmocnieniem

przykład prostego zastosowania

W najprostszej wersji stosuje się jeden wariant losowej modyfikacji (perturbacji) rozwiązania. Jednakże algorytmy heurystyczne dość łatwo jest rozwijać i ulepszać. Takim ulepszeniem może być np. zastosowanie kilku róznych operatorów perturbacji, mogą być nawet nie tylko losowe. Który z nich wybrać oraz jak i kiedy stosować?
Do tego celu możmy wykorzystać uczenie maszynowe.

Uczenie ze wzmocnieniem

przykład prostego zastosowania

Przykładowe sposoby modyfikacji (perturbacji) rozwiązania:

1. Losowe przestawienie 2 miast w posiadanej liście.
2. Wylosowanie 2 miast i odwrócenie kolejności miast między nimi.
3. Wylosowanie 2 par sąsiednich miast i sprawdzenie, czy nie poprawimy wyniku, prowadząc drogę „na krzyż” między nimi.
4. Wylosowanie pewnego miasta na trasie i przestawienie obok niego miasta najbliższego (w sensie przyjętej odległości).
5. Wylosowanie pewnego miasta na trasie i przestawienie obok niego miasta wylosowanego z pewnej grupy najbliższych (odległościowo).
6. Kilkukrotne powtórzenie (liczba powtórzeń może być losowana) jednego z wymienionych wyżej operatorów – traktowane jako osobny operator.
7. … - z pewnością można wymyślić jeszcze sporo innych sposobów modyfikacji rozwiązań w tym problemie.

Posiadamy wobec tego 6 możliwych rodzajów perturbacji do zastosowania!

Uczenie ze wzmocnieniem

przykład prostego zastosowania

ulepszony algorytm metody (największego) wzrostu z UM

begin
  t:=0
  wylosuj rozwiązanie początkowe x_s
  ustaw (jednakowe?) **oceny startowe** $o_1(0)$... $o_k(0)$ **operatorów** v_1... v_k
  oceń x_s
  x_0:=x_s
  repeat
    wylosuj operator v_i z v_1... v_k **na podstawie ocen operatorów** $o_1(t)$... $o_k(t)$
    wygeneruj n nowych rozwiązań x_1...x_n przez modyfikację x_0 **wybranym operatorem** v_i
    oceń rozwiązania x_1...x_n
    oceń operatory v_1... v_k **odpowiednio modyfikując** $o_1(t)$... $o_k(t)$
    wybierz najlepsze rozwiązanie x^* z x_1...x_n
    jeśli x^* jest lepsze od x_0 to x_0:=x^*
    t:=t+1
  until t=MAX
end

Uczenie ze wzmocnieniem

przykład prostego zastosowania

Skoro nie wiemy, które z nich będą lepsze, a które gorsze (sytuacja ta może się też zmieniać w trakcie obliczeń), to może zastosujemy je wszystkie, tylko z odpowiednim mechanizmem wyboru w każdej iteracji jednego z nich, wzorowanym na uczeniu maszynowym, zgodnie z heurystyką „im większą poprawę rozwiązania daje operator, tym częściej go stosujmy”, czyli będziemy maksymalizować wypłaty.
Zastosujemy wobec tego operatory perturbacji z pewnymi prawdopodobieństwami p_1, .. p_k, które będą proporcjonalne do ich osiągnięć o_1,.., o_k. W każdej iteracji będzie losowany 1 operator, a osiągnięta dzięki jego zadziałaniu „nagroda” podniesie prawdopodobieństwo jego wyboru w kolejnych krokach.

Uczenie ze wzmocnieniem

przykład prostego zastosowania

Prawdopodobieństwa te będą się zmieniać w trakcie poszukiwania rozwiązania problemu, gdyż jest bardzo prawdopodobne, że wymagania metody mogą się zmieniać w trakcie działania, np. operator 3 bardzo dobrze rozplątuje zapętlenia trasy - zapętlenia są zawsze nieoptymalne i przez zmianę dróg między 2 parami miast można je łatwo wyeliminować. Takich zapętleń jest na trasie pewna liczba, a gdy operator je wyeliminuje, nie będzie już potrzebny (przynajmniej dopóki nie powstaną nowe na skutek działania innych operatorów). Widać tu więc, że częstości wywoływania operatorów powinny się zmieniać: jedne maleć, gdy nie przynoszą (na jakimś etapie obliczeń) poprawy, inne rosnąć, gdy przynoszą.

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki z uczeniem

Dlatego też zaprojektujemy i zastosujemy tu mechanizm zarządzania operatorami. Naszym agentem będzie opisana wcześniej metoda obliczeniowa – algorytm wzrostu, która wykonuje akcje – wybiera i uruchamia któryś z operatorów modyfikujących rozwiązanie. Każdy wybrany i uruchomiony operator podlega ocenie. Jeśli któreś z nowych rozwiązań jest lepsze od posiadanego bazowego, otrzymuje nagrodę, jeśli są gorsze lub bez poprawy – karę.
Te nagrody i kary powinny układać się w jakąś prawidłowość, dzięki której przy zastosowaniu opisanej metody, powinniśmy przyspieszyć obliczenia w stosunku do metody bez oceny operatorów.

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki z uczeniem

Zastosujemy tu wzór na ocenę operatorów, analogiczny do pokazanego wcześniej:

$$o_n(t+1)=\begin{cases}(1-\eta)*o_k(t)+\eta*(r_t+\gamma*o^*(t+1)) \text{ dla } n=k\\ o_n(t) \text{ dla pozostałych } n\end{cases}$$

$o_n(t)$ – ocena operatora, r_t – nagroda lub kara, γ – współczynnik dyskonta, η – współczynnik nauki, $o^*(t+1)$ – oszacowanie maksymalnej wartości oceny, k – indeks operatora wybranego do modyfikacji rozwiązania,
oraz normalizację wypracowanych ocen, aby przekształcić je w prawdopodobieństwa wyboru operatorów:

$$p_k(t)=\frac{o_k(t)}{\sum_{i=1}^{n} o_i(t)}$$

Uczenie ze wzmocnieniem

przykład prostego zastosowania heurystyki z uczeniem

W taki sposób zastosowaliśmy metodę uczenia ze wzmocnieniem do automatycznego uczenia się sposobu doboru operatorów modyfikujących rozwiązanie w prostej heurystyce obliczeniowej – metodzie wzrostu. Tym samym przekształciliśmy ją w całkiem wyrafinowane narzędzie obliczeniowe, które może rozwiązywać także inne problemy, o ile zmienimy reprezentację rozwiązania, funkcję celu i zapewne także operatory.

Uczenie ze wzmocnieniem

Zastosowania

- generowanie strategii on-line na podstawie danych o akcjach i reakcjach środowiska;
- uczenie strategii w grach (np. wspomniane programy grające w go, szachy,...)
- pozyskiwanie metod sterowania obiektami, np. dronami, autonomicznymi samochodami.

Uczenie Bayesowskie

twierdzenie Bayesa

Uczenie zwane bayesowskim służy do pozyskiwania wiedzy o środowisku w postaci rozkladów prawdopodobieństw i oparte jest na znanym ze szkoły wzorze na prawdopodobieństwo warunkowe:

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

Jeśli teraz weźmiemy pod uwagę podstawową wersję twierdzenia Bayesa (można je dość łatwo wyprowadzić ze wzoru podstawowego powyżej):

$$P(A/B)=\frac{P(B/A)*P(A)}{P(B)}$$

gdzie A i B są zdarzeniami oraz $P(B)>0$.

to będziemy już blisko otrzymania narzędzia, dzięki któremu powstają różne algorytmy wnioskowania i uczenia się, wykorzystywane w SI, szczególnie jeśli weźmiemy pod uwagę wariant twierdzenia dla wielu zdarzeń.

Twierdzenie Bayesa

dla wielu zdarzeń

Niech $B, T_1, \ldots, T_n$ będą zdarzeniami takimi że:

$P(B) \geq 0, B \subset \bigcup_{i=1}^{n} T_i$ i $T_i \cap T_j = \emptyset$ dla $i \neq j$

to

$$P(T_i/B) = \frac{P(B/T_i) * P(T_i)}{\sum_{j=1}^{n} P(B/T_j) * P(T_j)}$$

Twierdzenie Bayesa

krótkie zadanie z prawdopodobieństwem warunkowym

Wykonano 3 rzuty symetryczną kostką do gry. W pierwszym rzucie wypadły 4 oczka. Jakie jest prawdopodobieństwo wyrzucenia w sumie więcej niż 10 oczek.
Jakie jest prawdopodobieństwo wyrzucenia więcej niż 10 oczek, gdy nic nie wiadomo o pierwszym rzucie?
Jak sytuacja będzie wyglądać, gdy w pierwszym rzucie wypadnie 1 oczko, a jak gdy 6?
Jakie byłoby prawdopodobieństwo wyrzucenia 4 oczek w pierwszym rzucie, gdy wiemy, że suma oczek w 3 rzutach jest większa od 10 (prawdopodobieństwo a posteriori)?

Twierdzenie Bayesa

krótkie zadanie z prawdopodobieństwem warunkowym

Wykonano 3 rzuty symetryczną kostką do gry. W pierwszym rzucie wypadły 4 oczka. Jakie jest prawdopodobieństwo wyrzucenia w sumie więcej niż 10 oczek. Odp.: $P(A/B)=7/12$, B – wyrzucenie 4 w pierwszym rzucie $P(B)=1/6$, $P(A \cap B)=21/216$, $P(A/B)= P(A \cap B)/P(B)=(21/216)/(1/6)=7/12$

Jakie jest prawdopodobieństwo wyrzucenia więcej niż 10 oczek, gdy nic nie wiadomo o pierwszym rzucie? Odp.: $P(A)=1/2$, A – wyrzucenie więcej niż 10 oczek w 3 rzutach.

Jak sytuacja będzie wyglądać, gdy w pierwszym rzucie wypadnie 1 oczko? Odp.: 1/6.

A jakie jest gdy 6? Odp.:5/6.

Jakie byłoby prawdopodobieństwo wyrzucenia 4 oczek w pierwszym rzucie, gdy wiemy, że suma oczek w 3 rzutach jest większa od 10? Odp.: $P(B/A)=P(A/B)*P(B)/P(A)=7/36$, odpowiednio dla 6 w pierwszym rzucie to 10/36 a dla 1 w pierwszym rzucie to 2/36.

Klasyfikacja bayesowska

Jeśli teraz zdarzenia T_i staną się hipotezami h_i ze zbioru hipotez H, a B będzie oznaczać zbiór danych trenujących D, to otrzymamy podstawę do metody zwanej klasyfikacją bayesowską.

$$P(h_i/D)=\frac{P(D/h_i)*P(h_i)}{\sum_{j=1}^{n} P(D/h_j)*P(h_j)}$$

Klasyfikacja bayesowska

Zakładając, że posiadamy dane trenujące z odpowiednimi etykietami (jak widać jest to uczenie z nadzorem) oraz zestaw kilku hipotez, które grupują te dane, na podstawie posiadanych lub szacowanych prawdopodobieństw występujących w powyższym wzorze, to możemy odnaleźć hipotezę maksymalizującą prawdopodobieństwo poprawnej klasyfikacji.
Opisane etykietami dane trenujące stają się podstawą do obliczenia prawdopodobieństw, których „nauczy się” metoda i w ten sposób w oparciu o te prawdopodobieństwa będzie również mogła klasyfikować nieznane wcześniej dane do wyuczonych kategorii.

Klasyfikacja bayesowska
zasady ML i MAP

W zasadzie można z pokazanych wzorów wysnuć dwie zasady wyboru najlepszych hipotez:

- zasadę maksymalnej zgodności (ML), która maksymalizuje warunkowe prawdopodobieństwo danych trenujących (zaobserwowania danych zgodnych z przyjętą hipotezą):

$$h_{ML} = \underset{h_i}{arg\,max}\left(P\left(D/h_i\right)\right)$$

- zasadę maksymalnego prawdopodobieństwa a posteriori (MAP), czyli właściwą metodę Bayesa, dzięki której można wybrać hipotezę o maksymalnym prawdopodobieństwie a posteriori, czyli najlepiej dopasowana do danych trenujących:

$$h_{MAP} = \underset{h_i}{arg\,max}\left(P\left(h_i/D\right)\right)$$

Klasyfikacja bayesowska

optymalny klasyfikator bayesowski

Optymalny klasyfikator beyesowski bazuje na h_{MAP} . Ma on jednak poważną wadę (podobnie zresztą jak operator ML). Otóż aby go użyć do celów praktycznych należałoby rozważyć zbiór wszystkich możliwych hipotez dla danego problemu, a następnie policzyć dla nich odpowiednie prawdopodobieństwa. Jest to zazwyczaj niewykonalne, gdyż zbiór możliwych hipotez najczęściej jest bardzo duży a nawet nieskończony. Oszacowanie odpowiednich prawdopodobieństw też może być trudne lub mogą być one nieznane. Dlatego też do celów praktycznych (poza bardzo rzadkimi przypadkami o niewielkim rozmiarze przestrzeni hipotez) używa się innych metod, z których najciekawszy wydaje się tzw. naiwny klasyfikator bayesowski zwany czasem także niezależnym klasyfikatorem bayesowskim lub NBC.

Klasyfikacja bayesowska

naiwny klasyfikator bayesowski

Na wstępie oszacujemy prawdopodobieństwa przynależności przykładów do danych kategorii najprościej na podstawie częstości wystąpienia:

$$P(c(x)=d)=\frac{|D^d|}{|D|}$$

$c(x)=d$ – wartość przyjętej funkcji klasyfikującej, $|D^d|$ i $|D|$ - liczby danych odpowiednio zaklasyfikowanych do kategorii d i wszystkich.

i przykładów danych o rożnych wartościach atrybutów a_i należących do kategorii d:

$$P(a_i(x)=v/c(x)=d)=\frac{|D^d_{a_i,v}|}{|D^d|}$$

Pewne problemy występują, gdy napotkamy kategorie/atrybuty, których przedstawiciele nie wystąpili w danych trenujących. Często nadaje im się pewne oszacowane prawdopodobieństwa, w najprostszym wypadku może to być 1, często jednak są to wartości niewielkie, a w pewnych sytuacjach może być to 0.

Klasyfikacja bayesowska

naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski zakłada, że wartości poszczególnych atrybutów opisujących dane są od siebie warunkowo niezależne (co często nie jest spełnione). To „naiwne” założenie dało nazwę metodzie. Jednak praktyka pokazuje, że nawet przy niespełnieniu tego założenia algorytm działa dobrze. Na podstawie posiadanych sklasyfikowanych danych uczących można stworzyć klasyfikator zdolny klasyfikować nowe, nieopisane dane, znajdując kategorię do której należy z największym prawdopodobieństwem .

Klasyfikacja Bayesowska

naiwny klasyfikator bayesowski

Dla każdej nowej danej, podlegającej klasyfikacji, należy obliczyć prawdopodobieństwa przynależności do poszczególnych kategorii. Największa znaleziona wartość determinuje klasyfikację przykładu według wzorów:

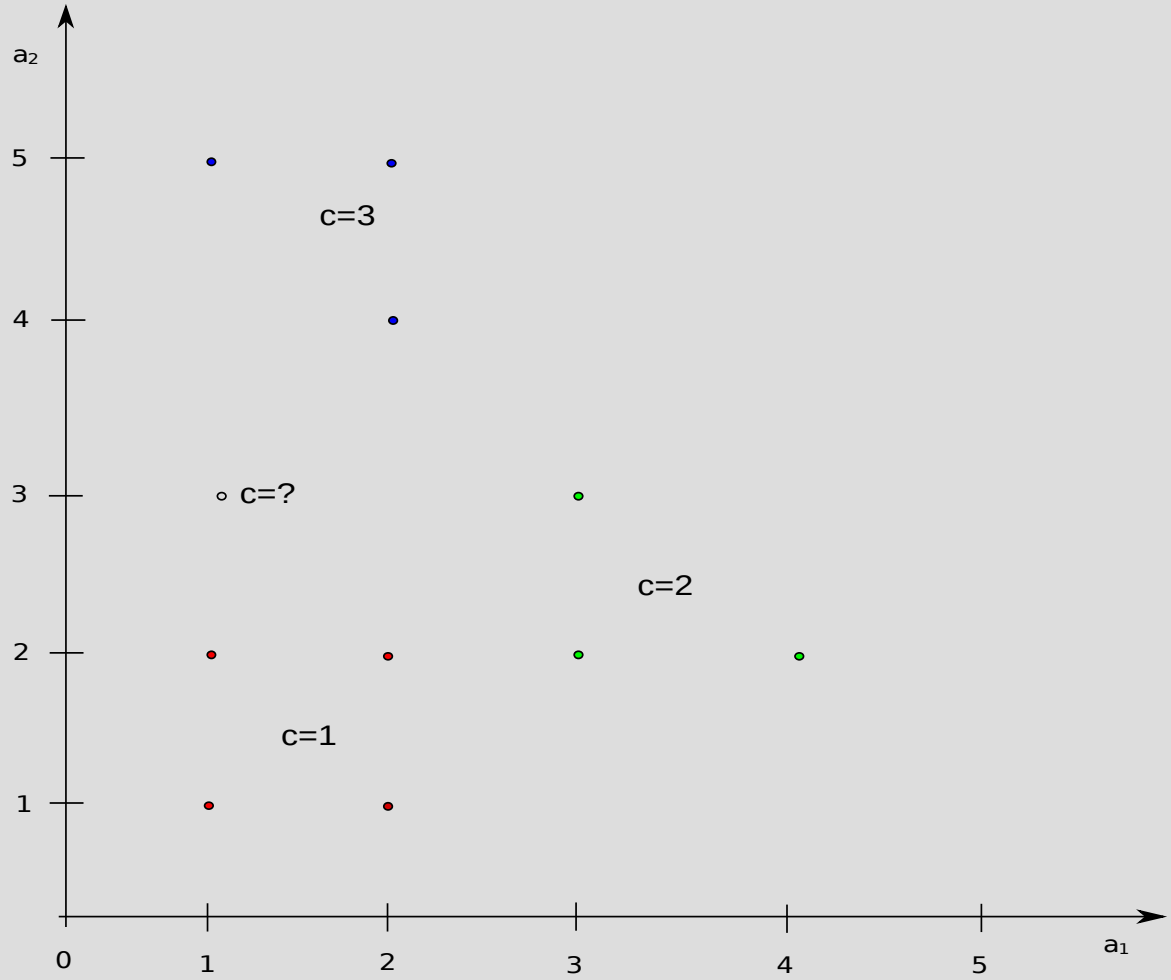
$$h_{(x^n)} = \underset{d}{arg\,max}\left(P\left(c(x)=d / a_1(x)=a_1(x^n) \wedge a_2(x)=a_2(x^n) \wedge \ldots \wedge a_l(x)=a_l(x^n)\right)\right)$$

a ponieważ stosujemy naiwne założenie o warunkowej niezależności prawdopodobieństw opisujących atrybuty, to:

$$h_{(x^n)} = \underset{d}{arg\,max}\left(P(c(x)=d) * P(a_1(x)=a_1(x^n)/c(x)=d) * P(a_2(x)=a_2(x^n)/c(x)=d) * \ldots * P(a_l(x)=a_l(x^n)/c(x)=d)\right)$$

Klasyfikacja Bayesowska

prosty przykład



Klasyfikacja Bayesowska

prosty przykład

Mamy 10 danych o podanych etykietach. Należy znaleźć klasyfikację danej x^n oznaczonej na rysunku c=? o wartościach atrybutów $a_1=1$, $a_2=3$.

Obliczamy:

$P(c(x)=1)=4/10$, $P(c(x)=2)=3/10$, $P(c(x)=3)=3/10$

dla $c(x)=1$

$P(a_1=1/c(x)=1)=1/2$, $P(a_1=2/c(x)=1)=1/2$, $P(a_2=1/c(x)=1)=1/2$, $P(a_2=2/c(x)=1)=1/2$

dla $c(x)=2$

$P(a_1=3/c(x)=2)=2/3$, $P(a_1=4/c(x)=2)=1/3$, $P(a_2=2/c(x)=2)=2/3$, $P(a_2=3/c(x)=2)=1/3$

dla $c(x)=3$

$P(a_1=1/c(x)=3)=1/3$, $P(a_1=2/c(x)=3)=2/3$, $P(a_2=4/c(x)=3)=1/3$, $P(a_2=5/c(x)=3)=2/3$

wobec tego:

$P(c(x^n)=1)=P(c(x)=1)*P(a_1=1/c(x)=1)*P(a_2=3/c(x)=1)=4/10*1/2*1=2/10=1/5$

$P(c(x^n)=2)=P(c(x)=2)*P(a_1=1/c(x)=2)*P(a_2=3/c(x)=2)=3/10*1/3*1=1/10$

$P(c(x^n)=3)=P(c(x)=3)*P(a_1=1/c(x)=3)*P(a_2=3/c(x)=3)=3/10*1/3*1=1/10$

Czyli maksymalna wartość prawdopodobieństwa przynależności jest dla $c(x)=1$.

Proszę policzyć jak zaklasyfikuje metoda dane o atrybutach: $a_1=2$, $a_2=3$ oraz: $a_1=3$, $a_2=1$.

Sieć bayesowska

Sieć bayesowska ma za zadanie przedstawić zależności pomiędzy zdarzeniami opierając się na rachunku prawdopodobieństwa. Sieci reprezentują zależności przyczynowe pomiędzy przyczynami a skutkami jakichś działań lub sytuacji.

Sieć taka jest skierowanym grafem acyklicznym. Wierzchołki tego grafu reprezentują zdarzenia, a łuki pokazują związki przyczynowe pomiędzy tymi zdarzeniami. Jeśli od wierzchołka A do B prowadzi ścieżka, to B jest potomkiem A. W sieci bayesowskiej zdarzenia są niezależne od innych zdarzeń, które nie są jego potomkami.

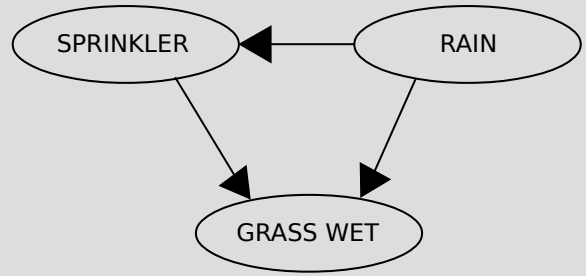
Sieć bayesowska

Można uznać, że sieć bayesowska zapisuje informacje o pewnej dziedzinie przy użyciu wykresu (grafu), pokazującego zależności między wierzchołkami, będącymi pewnymi zmiennymi losowymi, gdzie krawędzie pokazują zależności między nimi, mające najczęściej również charakter losowy.

Do opisu zależności i pomiędzy węzłami służą wzory analogiczne do występujących w Twierdzeniu Bayesa.

Sieć bayesowska
przykład

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99



	RAIN	
T	F	
0.2	0.8	

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Przykład na podst. Wikipedii.