Zaawansowane uczenie maszynowe: *podstawowe algorytmy uczenia się*

(rozszerzony przegląd)

Paweł Cichosz

Semestr 20Z

Przedstawiony tutaj przegląd podstawowych algorytmów uczenia się ma służyć jako pomocniczy materiał uzupełniający skrócony przegląd prezentowany na wykładach 2 i 3 z przedmiotu *Zaawansowane uczenie maszynowe*. Zawiera on zwięzłe podsumowanie najważniejszych treści w punktach oraz dodatkowy szerszy komentarz w ramkach.

1 Drzewa decyzyjne

Drzewa decyzyjne są prostą metodą reprezentacji modeli do uczenia się pojęć dostosowaną do wymogów praktycznych zastosowań. Ponieważ algorytmy używające tej reprezentacji – nazywane algorytmami indukcji drzew decyzyjnych – zwykle używają również podobnych mechanizmów tworzenia modeli, terminem „drzewa decyzyjne" określa się potocznie często nie tylko modele, lecz także te algorytmy. Należą one do najczęściej używanych algorytmów uczenia się pojęć.

1.1 Reprezentacja modeli

Węzły: podziały (testy) na podstawie warunków dotyczących wartości atrybutów $t : X \to R_t$.

Gałęzie: dla każdego wyniku $r \in R_t$ podziału t w węźle prowadzą z tego węzła do jego węzłów potomnych.

Liście: klasy lub prawdopodobieństwa klas.

Drzewo decyzyjne jest rekurencyjną strukturą złożoną z węzłów i liści. Węzły reprezentują podziały (zbioru trenującego i dziedziny) dokonywane na podstawie pewnych warunków, nazywanych także testami. Test możemy traktować jako funkcję t, która każdemu przykładowi przypisuje jeden z możliwych wyników podziału R_t. Szczególnie popularne są podziały binarne o dwóch wynikach, lecz mogą być także stosowane podziały wielowartościowe. Każdemu możliwemu wynikowi podziału odpowiada jedna gałąź wychodząca z węzła i prowadząca do węzła lub liścia na niższym poziomie.

O liściach założymy, że zawierają etykiety klas, przypisywane przykładom w czasie predykcji. Zwykle są tam również przechowywane prawdopodobieństwa poszczególnych klas, umożliwiające predykcję probabilistyczną.

Proces predykcji: przykład x propagowany od korzenia drzewa do liścia ścieżką wyznaczaną przez wyniki podziałów w kolejnych odwiedzonych węzłach $t_1(x), t_2(x), \ldots$, klasa z osiągniętego liścia jest wartością $h(x)$.

Predykcja za pomocą drzewa decyzyjnego polega na propagacji przykładu od korzenia drzewa do pewnego liścia, którego klasa staje się wynikiem predykcji. W każdym węźle do przykładu stosowany jest podział, którego wynik wyznacza węzeł lub liść, do którego przykład trafia schodząc niżej.

Dekompozycja dziedziny/zbioru danych: z każdym węzłem lub liściem $\mathbf{n}$ jest związany odpowiedni podzbiór $X_{\mathbf{n}}$ dziedziny X i odpowiedni podzbiór $S_{\mathbf{n}}$ dowolnego zbioru danych S, zawierający przykłady, które docierają do tego węzła lub liścia, jeśli są propagowane z korzenia drzewa ścieżkami wyznaczanymi przez wyniki kolejnych podziałów.

Drzewo decyzyjne można interpretować jako reprezentację dekompozycji zbioru trenującego i dziedziny na podzbiory (regiony), wyznaczaną przez podziały w węzłach. Z każdym węzłem można związać odpowiedni podzbiór zbioru trenującego i dziedziny, złożony z przykładów, które do tego węzła docierają, gdy propaguje się je w dół od korzenia.

Typy podziałów dla atrybutów dyskretnych:

wielowartościowe na podstawie wartości atrybutu: $t(x) = a(x)$

binarne na podstawie równości:

$$t(x) = \begin{cases} 1 & \text{jeżeli } a(x) = v \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

wielowartościowe na podstawie podzbiorów wartości atrybutu:

$$t(x) = \begin{cases} 1 & \text{jeżeli } a(x) \in V_1 \\ 2 & \text{jeżeli } a(x) \in V_2 \\ \dots & \\ k & \text{jeżeli } a(x) \in V_k \end{cases}$$

binarne na podstawie przynależności do zbioru:

$$t(x) = \begin{cases} 1 & \text{jeżeli } a(x) \in V \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

Typy podziałów dla atrybutów ciągłych:

binarne na podstawie nierówności:

$$t(x) = \begin{cases} 1 & \text{jeżeli } a(x) \leq v \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

wielowartościowe na podstawie przedziałów wartości atrybutu:

$$t(x) = \begin{cases} 1 & \text{jeżeli } a(x) \in (-\infty, v_1] \\ 2 & \text{jeżeli } a(x) \in (v_1, v_2] \\ \dots & \\ k & \text{jeżeli } a(x) \in (v_{k-1}, \infty) \end{cases}$$

Podział jest funkcją definiowaną na podstawie wartości atrybutów opisujących przykłady. Tutaj ograniczamy się do podziałów opartych na pojedynczych atrybutach, jakie są stosowane przez praktycznie wszystkie implementacje będące w powszechnym użyciu. Podziały używające wielu atrybutów mogą uprościć drzewo, lecz zwiększają złożoność algorytmu budowy drzewa.

Możliwe typy podziałów zależą od typów atrybutów. Najczęściej stosowane z nich są wymienione wyżej. Zwykle stosuje się albo wyłącznie podziały binarne, albo wielowartościowe – nie mieszając ich w ramach tego samego drzewa.

Wymiar VC: nieskończony dla atrybutów ciągłych, równy liczbie wszystkich możliwych różnych wektorów wartości atrybutów dla atrybutów dyskretnych – ryzyko nadmiernego dopasowania.

Przestrzeń modeli reprezentowanych za pomocą drzew decyzyjnych jest „pojemna". Budując odpowiednio głębokie drzewo można wyrazić za pomocą sekwencji podziałów dowolne warunki przynależności przykładów do klas, oparte na sprawdzaniu wartości atrybutów. Oznacza to zatem, że dowolnie wiele różnych przykładów (o różnych wartościach atrybutów) można poetykietować na wszystkie możliwe sposoby. Wymiar VC tej przestrzeni modeli jest więc równy liczbie wszystkich różnych wektorów wartości atrybutów, jeśli używane są wyłącznie atrybuty dyskretne, a staje się nieskończony w przypadku użycia atrybutów ciągłych. Jest to więc bardzo pojemna przestrzeń modeli, co zwiększa ryzyko nadmiernego dopasowania. Jednak dzięki mechanizmom ograniczającym to ryzyko efektywny wymiar VC staje się niższy, gdyż algorytmy budowy drzew decyzyjnych dzięki stosowanym ograniczeniom nie wykorzystują całej „pojemności" przestrzeni modeli.

Zauważmy przy okazji, że w przypadku dziedziny na której określone są dwa atrybuty ciągłe oraz dla zbioru klas $C = \{0, 1\}$ przestrzeń modeli reprezentowanych przez drzewa decyzyjne jest nadzbiorem rozważanej w przykładach dotyczących obliczeniowej teorii uczenia się przestrzeni modeli reprezentowanych przez prostokąty o bokach równoległych do osi układu współrzędnych. Faktycznie, dla dowolnego takiego prostokąta bez trudu można wskazać reprezentujące go drzewo, zawierające cztery podziały. Zademonstrowanie tego pozostawiamy jako ćwiczenie.

1.2 Zstępująca budowa drzewa

- Sekwencja decyzji:

  kryterium stopu: węzeł czy liść?

  wybór klasy dla liścia: jaka predykcja najlepsza w liściu?

  wybór podziału dla węzła: jaki podział najlepszy w węźle?

- Ten sam schemat powtarzany dla korzenia drzewa, jego węzłów potomnych, ich węzłów potomnych itd.

Proces budowy drzewa stosowany w typowych algorytmach jest zstępujący: zaczyna się od korzenia, któremu odpowiada cały zbiór trenujący, i rozbudowuje drzewo stosując sekwencję operacji podejmowania decyzji: czy utworzyć węzeł czy liść, jaką klasę umieścić w liściu, oraz jaki podział zastosować w węźle. W przypadku decyzji o utworzeniu węzła wybierany jest dla niego podział, a każdemu wynikowi tego podziału odpowiada gałąź prowadząca do kolejnego węzła lub liścia (przy czym zbiór przykładów odpowiadających temu węzłowi lub liściowi stanowi podzbiór zbioru z węzła macierzystego zawężony do przykładów, dla których podział daje wynik odpowiadający tej gałęzi). Dla każdego z nowo utworzonych węzłów/liści ponownie stosuje się kryterium stopu (aby rozstrzygnąć, czy faktycznie będzie węzłem czy liściem), itd.

Chociaż jest popularne formułowanie algorytmów budowy drzewa w sposób rekurencyjny, równie dobrze można przyjąć inne sformułowanie (np. iterację z kolejką węzłów).

1.3 Kryterium stopu

Kontekst:

- węzeł $\mathbf{n}$,
- zbiór przykładów trenujących $T_{\mathbf{n}}$.

Jednolita klasa: $T_{\mathbf{n}}$ zawiera przykłady dokładnie jednej klasy (wybieranej jako klasa dla liścia).

Brak przykładów: $T_{\mathbf{n}} = \emptyset$ (klasa dla liścia z węzła macierzystego).

Brak możliwości podziału: każdy możliwy podział osiąga dla $T_{\mathbf{n}}$ tylko jeden wynik (nie dzieli).

Warianty rozluźnione: dominacja jednej klasy, mała liczba przykładów, najlepszy podział zbyt słaby.

Wpływ na właściwości drzewa: kryteria stopu (w nierozluźnionej postaci) gwarantują minimalizację błędu na zbiorze trenującym.

Kryterium stopu odpowiada za podjęcie decyzji, czy dla pewnego zbioru przykładów należy utworzyć liść (czyli przypisać mu przewidywaną klasę), czy węzeł (czyli dalej go podzielić). Wybór klasy dla tworzonego liścia wiąże się bezpośrednio z warunkami, przy których jest on tworzony, więc może być przedstawiony łącznie z kryteriami stopu.

Wymienione wyżej kryteria mają postać najbardziej restrykcyjną – tworzą liść wtedy, gdy stosowanie dalszych podziałów jest ponad wszelką wątpliwość bezzasadne. Zatem węzeł $\mathbf{n}$ powinien niewątpliwie stać się liściem, jeśli odpowiadający mu podzbiór zbioru trenującego $T_{\mathbf{n}}$ zawiera przykłady dokładnie jednej klasy (nie ma sensu dzielić, gdyż ewentualny podział nie może zmienić predykcji drzewa), jeśli jest zbiorem pustym lub zbiorem, którego za pomocą dostępnych podziałów nie da się podzielić (tzn. każdy dostępny podział przypisuje wszystkim przykładom ten sam wynik, czyli ich nie rozdziela). Ta ostatnia sytuacja w istocie oznacza, że dostępny zestaw atrybutów nie jest wystarczający do pełnej separacji klas, tzn. w zbiorze trenującym występują identycznie „wyglądające" przykłady o różnych klasach. Jest to zjawisko zupełnie normalne, a nawet częste w rzeczywistych zbiorach danych.

Każde z tych restrykcyjnych kryteriów stopu można rozluźnić, aby działało wcześniej i ograniczało rozrost drzewa. Zwykle się to robi, aby ograniczyć ryzyko nadmiernego dopasowania (a także zaoszczędzić na obliczeniach). Można więc dobór kryteriów stopu

traktować jako jeden z mechanizmów zapobiegania nadmiernemu dopasowaniu drzew decyzyjnych.

Warto zauważyć, że kryteria stopu w restrykcyjnej postaci zapewniają maksymalny możliwy poziom dopasowania do danych trenujących (tzn. minimalny możliwy do osiągnięcia błąd na zbiorze trenującym), bez względu na to, jak wybierane są podziały. Błędnie klasyfikowane przykłady trenujące pojawią się tylko w liściach tworzonych ze względu na brak możliwości podziału, czyli w przypadku nieodróżnialnych przykładów o różnych klasach. Jeśli więc rolą kryteriów stopu jest kontrolowania poziomu dopasowania drzewa do danych trenujących, to wybór podziału musi pełnić inną rolę, o której będzie mowa dalej.

Niekiedy stosowanym uzupełniającym kryterium stopu jest ograniczenie maksymalnej głębokości drzewa. Podobnie jak rozluźnione warianty wcześniej dyskutowanych kryteriów, ogranicza ono poziom dopasowania do danych trenujących.

1.4 Kryterium wyboru podziału

Jak widzieliśmy, kryteria stopu są odpowiedzialne za poziom dopasowania drzewa decyzyjnego do danych i w restrykcyjnej postaci zapewniają minimalny możliwy błąd na zbiorze trenującym bez względu na sposób wybierania podziałów. Rolą kryterium wyboru podziału nie jest więc kontrolowanie poziomu dopasowania do danych, lecz wpływanie na strukturę drzewa przy określonym (przez kryteria stopu) poziomie dopasowania.

Motywacja: preferencja dla prostych drzew ograniczająca ryzyko nadmiernego dopasowania.

Istnieją powody, aby spośród różnych jednakowo dopasowanych drzew preferować drzewa prostsze, co jest zastosowaniem zasady brzytwy Ockhama. Podczas budowy prostszych drzew – o mniejszej liczbie poziomów i o krótszych ścieżkach – podejmowana jest mniejsza liczba decyzji, a więc mniejsze jest ryzyko, że niektóre z nich mogą być niewłaściwe. Podobnie, mniejsza liczba decyzji jest podejmowana podczas predykcji za pomocą prostszych drzew. Uważa się, że zazwyczaj sprzyja to redukcji ryzyka nadmiernego dopasowania, a więc – przy jednakowym błędzie na zbiorze trenującym – zmniejsza błąd rzeczywisty (oczekiwany błąd na nowych danych). Nie zawsze musi tak być i można znaleźć kontrprzykłady, ale dla większości rzeczywistych zbiorów danych można uznać preferencję dla prostszych drzew za skuteczną.

Realizacja: ocena jakości podziałów na podstawie nieczystości rozkładu klas po podziale, wybór najlepszego podziału w każdym węźle (ze skończonego zbioru kandydatów).

Metodą realizacji preferencji dla prostszych drzew jest wybieranie podziałów, które „skracają ścieżki" w drzewie, a więc przybliżają moment spełnienia kryteriów stopu i utworzenia liści. Oczywiście wiodącym kryterium stopu jest to, które dotyczy uzyskania jednolitej klasy, i to jemu podporządkowany jest wybór podziału. Standardową praktyką jest przy tym preferowanie podziałów, które pozwalają uzyskać najmniejszą „nieczystość" klas po podziale, gdyż to oznaczana przybliżenie sytuacji, w której zostanie spełnione kryterium stopu. Duża „nieczystość" lub nierównomierność rozkładu klas oznacza sytuację, w której klasa dominująca ma znaczną przewagę i pozostałe klasy występują rzadko.

Kontekst:

- węzeł $\mathbf{n}$,
- zbiór przykładów trenujących $T_{\mathbf{n}}$,
- rozważany podział $t : X \to R_t$,
- dla każdego wyniku podziału $r \in R_t$ odpowiedni podzbiór przykładów trenujących $T_{\mathbf{n},t=r} = \{x \in T_{\mathbf{n}} \mid t(x) = r\}$.

Entropia warunkowa: dla węzła $\mathbf{n}$, podziału t:

$$E_{T_{\mathbf{n},t=r}}(c) = \sum_{d \in C} -P_{T_{\mathbf{n},t=r}}(c = d) \log P_{T_{\mathbf{n},t=r}}(c = d)$$

$$E_{T_{\mathbf{n}}}(c|t) = \sum_{r \in R_t} \frac{|T_{\mathbf{n},t=r}|}{|T_{\mathbf{n}}|} E_{T_{\mathbf{n},t=r}}(c)$$

gdzie

$$P_{T_{\mathbf{n},t=r}}(c = d) = \frac{|T_{\mathbf{n},t=r,c=d}|}{|T_{\mathbf{n},t=r}|}$$

Oceniając podział t przeznaczony do użycia w węźle $\mathbf{n}$, któremu odpowiada podzbiór przykładów trenujących $T_{\mathbf{n}}$, powinniśmy rozważyć wszystkie wyniki tego podziału $r \in R_t$ i zmierzyć „nieczystość" klas w odpowiadających im podzbiorach $T_{\mathbf{n},t=r}$.

Jedną z najczęściej stosowanych miar „nieczystości" jest znana z teorii informacji entropia, obliczana jako suma zanegowanych iloczynów prawdopodobieństw klas i ich logarytmów. Podstawa logarytmu nie ma znaczenia (jej zmiana sprowadza się do skalowania), o ile pozostaje ustalona. W obliczeniach przyjmuje się $0 \log 0 = 0$.

Wyznaczając entropię dla każdego z podzbiorów wyznaczonych przez podział, a następnie średnią ważoną uzyskanych wyników (z wagami określonymi jako liczba przykładów w podzbiorach, $|T_{\mathbf{n},t=r}|$), otrzymujemy w istocie tzw. entropię warunkową klas pod warunkiem zastosowania podziału. Im mniejsza jest jej wartość, tym średnio mniejsza „nieczystość" klas po podziale, a więc tym lepszy podział.

Redukcja nieczystości w wyniku podziału:

$$\Delta E_{T_\mathrm{n}}(c|t) = E_{T_\mathrm{n}}(c) - E_{T_\mathrm{n}}(c|t)$$

Wybór podziału: minimalizacja nieczystości po podziale (równoważnie: maksymalizacja redukcji nieczystości w wyniku podziału).

Pożądany efekt „skracania ścieżek" w drzewie przybliża wybieranie podziałów o minimalnej „nieczystości" klas po podziale (a więc np. o minimalnej entropii warunkowej lub o minimalnym warunkowym indeksie Giniego). Równoważnie, można wziąć pod uwagę redukcję „nieczystości" jako różnicę „nieczystości" przed podziałem i po podziale. W przypadku entropii byłaby to różnica między bezwarunkową i warunkową entropią (i analogicznie dla indeksu Giniego). Taka redukcja powinna być maksymalizowana. Dodatkową zaletą wyznaczenia redukcji „nieczystości" jest możliwość sprawdzenia, czy i w jakim stopniu jest ona większa od 0, tzn. czy oceniany podział faktycznie wydaje się przynosić poprawę. Warunek, że dla najlepszego podziału redukcja „nieczystości" klas jest zerowa (lub mniejsza od pewnej ustalonej minimalnej wartości), mógłby stanowić dodatkowe kryterium stopu (rezygnacja z dalszego dzielenia, jeśli najlepszy dostępny podział jest zbyt słaby).

Przykład: pogoda

Budowa drzewa decyzyjnego na podstawie zbioru trenującego:

x	*outlook*	*temperature*	*humidity*	*wind*	*play*
1	*sunny*	*hot*	*high*	*normal*	*no*
2	*sunny*	*hot*	*high*	*high*	*no*
3	*overcast*	*hot*	*high*	*normal*	*yes*
4	*rainy*	*mild*	*high*	*normal*	*yes*
5	*rainy*	*cold*	*normal*	*normal*	*yes*
6	*rainy*	*cold*	*normal*	*high*	*no*
7	*overcast*	*cold*	*normal*	*high*	*yes*
8	*sunny*	*mild*	*high*	*normal*	*no*
9	*sunny*	*cold*	*normal*	*normal*	*yes*
10	*rainy*	*mild*	*normal*	*normal*	*yes*
11	*sunny*	*mild*	*normal*	*high*	*yes*
12	*overcast*	*mild*	*high*	*high*	*yes*
13	*overcast*	*hot*	*normal*	*normal*	*yes*
14	*rainy*	*mild*	*high*	*high*	*no*

Załóżmy, że w roli podziałów będą używane bezpośrednio atrybuty (podziały wielowartościowe na podstawie wartości atrybutu). Mamy więc podziały `outlook` i *temperature* o trzech możliwych wynikach oraz podziały *humidity* i *wind* o dwóch możliwych

wynikach. Ponadto przyjmijmy, że stosowane są restrykcyjne kryteria stopu, zgodnie z którymi liście tworzy się po uzyskaniu jednolitych klas (o ile to możliwe).

Rozważmy wybór podziału dla korzenia drzewa, w którym mamy $T_{\mathbf{n}} = T$. Dla podziału *outlook* mamy, stosując logarytmy dwójkowe:

$$\begin{aligned} E_{T_{outlook=sunny}}(c) &= -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \approx 0.971 \\ E_{T_{outlook=overcast}}(c) &= -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0 \\ E_{T_{outlook=rainy}}(c) &= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} \approx 0.971 \end{aligned}$$

Na tej podstawie można obliczyć entropię warunkową:

$$E_T(c|\,outlook) = \frac{5}{14}\cdot 0.971 + \frac{4}{14}\cdot 0 + \frac{5}{14}\cdot 0.971 \approx 0.694$$

W podobny sposób należy ocenić pozostałe podziały i wybrać najlepszy z nich (o minimalnej entropii warunkowej), a następnie zastosować wybrany podział i przejść do sprawdzenia kryterium stopu oraz ewentualnie wyboru podziału dla utworzonych węzłów potomnych. Zakładając, że w korzeniu drzewa będzie wybrany podział *outlook*, otrzymamy liść potomny dla gałęzi *outlook=overcast* (są tam wyłącznie przykłady klasy *yes*) oraz węzły potomne dla gałęzi *outlook=sunny* i *outlook=rainy*, dla każdej z których trzeba będzie przeprowadzić wybór podziału. Kontynuacja przykładu pozostaje do dokończenia samodzielnego jako ćwiczenie.

1.5 Właściwości drzew decyzyjnych

- Zwykle dobra jakość predykcji (chociaż często inne algorytmy dają nieco lepsze modele).
- Reprezentacja modeli czytelna dla człowieka.
- Podatne na nadmierne dopasowanie – konieczne staranne dobranie kryteriów stopu lub zastosowanie przycinania.

Drzewa decyzyjne osiągają zazwyczaj dość dobrą jakość predykcji, chociaż znane są algorytmy tworzące często lepsze modele. Jednak, w przeciwieństwie do nich, drzewa mają prostą strukturę, nadającą się do bezpośredniej interpretacji przez człowieka.

Drzewa decyzyjne są dość efektywne obliczeniowo i stosunkowo łatwe w użyciu. Szczególnej uwagi wymaga jednak, ze względu na ich dużą podatność na nadmierne dopasowanie, dostrojenie parametrów kontrolujących kryteria stopu lub zastosowanie przycinania.

W głównym nurcie wykładów ZUM będzie mowa właśnie o metodach przycinania drzew, a także o obsłudze brakujących wartości atrybutów i o wariantach drzew służących do zadania regresji.

2 Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski jest najprostszym reprezentantem algorytmów uczenia się opartych na probabilistycznym wnioskowaniu wykorzystującym dobrze znany wzór Bayesa z rachunku prawdopodobieństwa, pochodzący z pracy Thomasa Bayesa z 1763 roku.

Wzór Bayesa:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

Zastosowanie do predykcji prawdopodobieństw klas:

$$P(d|x) = P(c = d \mid a_1 = a_1(x), a_2 = a_2(x), \ldots, a_n = a_n(x))$$

Wzór Bayesa które opisuje związek między prawdopodobieństwem warunkowym pewnego zdarzenia a jego prawdopodobieństwem bezwarunkowym. Może być on traktowany jako zapis mechanizmu wnioskowania, w którym zdarzenie A reprezentuje pewną hipotezę, a zdarzenie B – fakty bądź obserwacje, które tę hipotezę mogą uprawdopodabniać lub unieprawdopodabniać. Wówczas prawdopodobieństwo $P(A)$ nazywane jest prawdopodobieństwem *a priori*, a prawdopodobieństwo $P(A|B)$ – prawdopodobieństwem *a posteriori*.

Zastosowanie wzoru Bayesa do zadania klasyfikacji może polegać na wyznaczeniu prawdopodobieństwa pewnej klasy d dla pewnego przykładu x, czyli $P(d|x)$. W bardziej rozwiniętym zapisie, przedstawionym wyżej, w jawny sposób przedstawiona jest jego interpretacja: chodzi o prawdopodobieństwo tego, że klasą pewnego przykładu jest klasa d, jeśli wiadomo, że wartości atrybutów tego przykładu są równe $a_1(x), a_2(x), \ldots, a_n(x)$, czyli takie, jak dla przykładu x.

Prawdopodobieństwo klasy na podstawie wartości atrybutów:

$$\begin{aligned} &P(c = d \mid a_1 = v_1, \ldots, a_n = v_n) \\ &\quad = \frac{P(c = d)P(a_1 = v_1, \ldots, a_n = v_n \mid c = d)}{P(a_1 = v_1, \ldots, a_n = v_n)} \end{aligned}$$

Pisząc dalej krótko $v_1, v_2, \ldots, v_n$ w miejsce $a_1(x), a_2(x), \ldots, a_n(x)$ oraz korzystając ze wzoru Bayesa możemy interesujące nas prawdopodobieństwo przedstawić w postaci ułamka jak wyżej. Przyjrzymy się teraz poszczególnym występującym w nim prawdopodobieństwom i rozważymy możliwość ich wyznaczenia na podstawie pewnego zbioru trenującego T.

Prawdopodobieństwo *a priori* klasy:

$$P(c = d) = P_T(c = d) = \frac{|T_{c=d}|}{|T|}$$

Prawdopodobieństwo *a priori* klasy może być bezpośrednio estymowane na podstawie zbioru trenującego w zwykły sposób. Klasy mogą być niezrównoważone, lecz można założyć, że wszystkie są reprezentowane w zbiorze trenującym wystarczająco licznie, aby taka estymacja była wystarczająca.

Założenie o niezależności:

$$P(a_1 = v_1, \ldots, a_n = v_n \mid c = d) = \prod_{i=1}^{n} P(a_i = v_i | c = d)$$

Łączne prawdopodobieństwo warunkowe wartości wszystkich atrybutów przy danej klasie nie może być już bezpośrednio estymowane ze zbioru trenującego. Nie można realistycznie zakładać, że ten zbiór jest na tyle duży, aby znalazła się w nim wystarczająco liczna reprezentacja każdej kombinacji wartości atrybutów w ramach każdej klasy do wiarygodnej estymacji ich prawdopodobieństw. W rzeczywistych zbiorach danych można się spodziewać, że wiele możliwych kombinacji wartości atrybutów w ogóle nie wystąpi, niektóre wystąpią jednokrotnie, i tylko nieliczne więcej niż raz. Aby wyznaczyć to prawdopodobieństwo przyjmuje się zatem inne założenie – o warunkowej niezależności atrybutów względem klas (tzn. że w ramach każdej klasy atrybuty są niezależne). Założenie to pozwala wyznaczyć łączne prawdopodobieństwo warunkowe dla wszystkich atrybutów jako iloczyn warunkowych prawdopodobieństw dla pojedynczych atrybutów. Założenie to jest zazwyczaj niespełnione, a jego „naiwne" przyjmowanie jest wyjaśnieniem nazwy algorytmu.

Przyjmując niespełnione założenie powodujemy, że wyznaczane na jego podstawie prawdopodobieństwa nie są poprawne. Nie musi to jednak wykluczać, że okażą się one wystarczające do skutecznej klasyfikacji.

Prawdopodobieństwo warunkowe wartości atrybutu:

$$P(a_i = v_i | c = d) = P_{T_{c=d}}(a_i = v_i) = \frac{|T_{c=d, a_i=v_i}|}{|T_{c=d}|}$$

Prawdopodobieństwo wartości jednego atrybutu przy danej klasie, może być już bezpośrednio estymowane na podstawie zbioru trenującego. Załóżmy na razie, że będzie to standardowa estymacja częstościowa, jak zapisana wyżej. W ten sposób został określony sposób wyznaczenia licznika ułamka uzyskanego po zastosowaniu wzoru Bayesa.

Mianownik:

$$P(a_1 = v_1, \ldots, a_n = v_n) = \sum_{d \in C} P(c = d) P(a_1 = v_1, \ldots, a_n = v_n \mid c = d)$$

W przypadku mianownika wzoru Bayesa należy powstrzymać się przed pokusą ponownego skorzystania z założenia o niezależności, tym razem w mocniejszej formie (już bezwzględnej a nie względem klas), które także zazwyczaj byłoby niespełnione i którego przyjęcie byłoby dodatkowym czynnikiem sprzyjającym niepoprawnej predykcji. Lepszym pomysłem jest skorzystanie z mianownika jako okazji do wymuszenia, aby prawdopodobieństwa wyznaczone dla tego samego przykładu (wektora wartości atrybutów), ale różnych klas, sumowały się do 1. Wówczas mianownik staje się po prostu stałą normalizującą, którą – jak zapisano wyżej – można wyznaczyć jako sumę liczników dla wszystkich klas. Oczywiście w przypadku, gdy predykcja polega na przypisaniu przykładom najbardziej prawdopodobnych klas, wyznaczanie mianownika można całkowicie pominąć, gdyż do ich wyboru wystarczy porównanie wartości licznika.

Zestaw prawdopodobieństw $P(c = d)$ dla wszystkich klas oraz $P(a_i = v_i|c = d)$ dla wszystkich atrybutów, wartości i klas stanowi reprezentację modelu naiwnego klasyfikatora bayesowskiego. Jak łatwo zauważyć, do jej wyznaczenia wystarcza jeden przebieg przez zbiór trenujący, w trakcie którego aktualizowane są odpowiednie liczniki. Jest to więc algorytm o niskiej złożoności obliczeniowej, łatwy do implementacji i możliwy do zrównoleglenia w trywialny sposób.

Użycie modelu do predykcji jest jeszcze prostsze. Dla przykładu do klasyfikacji x wystarczy wybrać prawdopodobieństwa $P(a_i = v_i|c = d)$ dla $v_i = a_i(x)$ oraz wyznaczyć na podstawie ich iloczynu $P(c = d \mid a_1 = v_1, \ldots, a_n = v_n)$.

2.1 Prawdopodobieństwa zerowe i prawie zerowe

Stała wartość dodatnia: $P(a_i = v_i|c = d) = \epsilon$ jeśli $T_{c=d,a_i=v_i} = \emptyset$.

Wygładzanie Cestnika (m-estymacja):

$$P(a_i = v_i|c = d) = \frac{|T_{c=d,a_i=v_i}| + mp}{|T_{c=d}| + m}$$

(zwykle $p = \frac{1}{|A_i|}$)

Wygładzanie Laplace'a:

$$P(a_i = v_i|c = d) = \frac{|T_{c=d,a_i=v_i}| + l}{|T_{c=d}| + l|A_i|}$$

Łatwo zauważyć, że uzyskanie zerowej wartości któregokolwiek z prawdopodobieństw $P(a_i = v_i|c = d)$ pociąga za sobą uznanie, że klasa d ma zerowe prawdopodobieństwo dla klasyfikowanego przykładu. Jest to radykalny i pochopny wniosek. W dodatku w sytuacji, gdy taka zerowa wartość pojawi się dla każdej klasy (a nie byłoby niczym dziwnym, gdyby dla każdej klasy okazało się, że jakaś wartość atrybutu ani razy nie występuje w zbiorze trenującym), jakakolwiek sensowna klasyfikacja przykładu stałaby się niemożliwa. W związku wartości zerowe prawdopodobieństw muszą wyeliminowane. Najprostszą techniką byłoby ich zastąpienie pewną małą stałą dodatnią, ale lepszą możliwość dają znane nam z wykładu 2 wygładzone estymatory prawdopodobieństwa, w wariancie Cestnika lub Laplace'a. Parametr kontrolujący wygładzanie staje się parametrem algorytmu.

Logarytm prawdopodobieństwa: mniejsze ryzyko błędów numerycznych przy mnożeniu małych prawdopodobieństw:

$$\log\left(P(c = d)\prod_{i=1}^{n} P(a_i = a_i(x)|c = d)\right)$$
$$= \log P(c = d) + \sum_{i=1}^{n} \log P(a_i = a_i(x)|c = d)$$

Często stosowaną techniką jest wyznaczanie logarytmu prawdopodobieństwa klas, co pozwala zastąpić mnożenie dodawaniem logarytmów i zmniejszyć ryzyko problemów numerycznych.

2.2 Atrybuty ciągłe

Funkcja gęstości: zastąpienie $P(a_i = v_i|c = d)$ przez $g_{d,i}(v_i)$, gdzie $g_{d,i}$ jest funkcją gęstości atrybutu a_i w klasie d – najczęściej zakładany rozkład normalny, parametry estymowane jako $m_{T_{c=d}}(a_i)$ i $s_{T_{c=d}}(a_i)$.

Dyskretyzacja: często lepsze, chociaż bardziej pracochłonne podejście.

Naiwny klasyfikator bayesowski w podstawowym wariancie zakłada, że atrybuty są dyskretne i ma sens estymowanie prawdopodobieństw $P(a_i = v_i|c = d)$. Dla atrybutów ciągłych można to prawdopodobieństwo zastąpić wartością funkcji gęstości. Wymaga to ustalenia pewnej rodziny rozkładów prawdopodobieństwa (w praktyce – zawsze lub niemal zawsze rozkład normalny) oraz estymacji jego parametrów (średniej i odchylenia standardowego wartości atrybutu a_i dla przykładów klasy d). Podejście to może się sprawdzać dobrze, jeśli rozkład atrybutu jest faktycznie taki, jak zakładany. W przeciwnym przypadku zwykle bardziej skuteczne jest przeprowadzenie dyskretyzacji atrybutów ciągłych (o metodach, które mogą być w tym celu zastosowane, będzie mowa w drugiej części semestru).

2.3 Obsługa brakujących wartości

Tworzenie modelu: pomijanie brakujących wartości przy estymacji prawdopodobieństw $P(a_i = v_i|c = d)$,

Predykcja: pomijanie prawdopodobieństw $P(a_i = v_i|c = d)$ jeśli wartość a_i nie jest znana dla klasyfikowanego przykładu.

Omawiając drzewa decyzyjne prezentowaliśmy stosowane w nich metody obsługi brakujących wartości atrybutów. W przypadku naiwnego klasyfikatora bayesowskiego można sobie z nimi poradzić w dużo prostszy sposób. Przy tworzeniu modelu, tzn. estymacji prawdopodobieństw $P(a_i = v_i|c = d)$, wystarczy przykłady o brakujących wartościach atrybutu a_i pominąć. Z kolei stosując model należy w obliczanym iloczynie prawdopodobieństw pominąć czynnik $P(a_i = v_i|c = d)$, jeśli wartość atrybutu a_i nie jest znana.

2.4 Właściwości

- Prosty koncepcyjnie, implementacyjnie i obliczeniowo.
- Odporny na nadmierne dopasowanie.
- Niewymagający strojenia parametrów.
- Naruszenie założenia o niezależności nie wyklucza wartości predykcyjnej, chociaż predykcje prawdopodobieństw klas nie są skalibrowane.
- Skuteczny jeśli:
  - trzeba uwzględnić nieznaczny wpływ znacznej liczby atrybutów (np. klasyfikacja tekstu),
  - liczba przykładów jest stosunkowo mała w porównaniu z liczbą atrybutów,

– występują liczne brakujące wartości.

Naiwny klasyfikator bayesowski może być uznany za najprostszy algorytm klasyfikacji, który jest w stanie dostarczać użytecznej jakości predykcji. Prosta zasada działania, łatwość implementacji i efektywność obliczeniowa mogą zachęcać do jego używania, przynajmniej po to, aby porównać go z innymi bardziej wyrafinowanymi algorytmami. Jego dużą zaletą jest znaczna odporność na nadmierne dopasowanie i brak konieczności podejmowania jakichkolwiek środków w celu jego uniknięcia. Modele tworzone tym algorytmem mają raczej tendencję do niewystarczającego niż nadmiernego dopasowania. Wygodną cechą jest także brak konieczności strojenia parametrów algorytmu.

Zwykle niestety modele uzyskiwane za pomocą naiwnego klasyfikatora bayesowskiego nie mogą dorównać modelom uzyskiwanym za pomocą bardziej złożonych algorytmów, co można uznać za jedną z konsekwencji przyjęcia niespełnionego założenia o niezależności. Predykcje probabilistyczne naiwnego klasyfikatora bayesowskiego nie są skalibrowane, podobnie jak w przypadku drzew decyzyjnych, chociaż z innych przyczyn. Nie wyklucza jednak to ich wartości predykcyjnej – niepoprawne prawdopodobieństwa wciąż mogą być podstawą do dobrej klasyfikacji.

Zastosowania, w których naiwny klasyfikator bayesowski okazuje się najbardziej skuteczny, charakteryzują się często z jednej strony dużą liczbą atrybutów, a z drugiej strony brakiem na tyle silnych zależności, aby do predykcji klasy dla pewnego przykładu wystarczyło się opierać na wartościach niewielkiej liczby spośród nich. W przypadku drzew decyzyjnych, nawet jeśli atrybutów są dziesiątki lub setki, przy klasyfikacji jednego przykładu na jednej ścieżce od korzenia do liścia zwykle wykorzystuje się ich nie więcej niż kilka. Nie zawsze jednak sprawdzenie wartości kilku atrybutów wystarczy do predykcji klasy. Jeśli tak nie jest i każdy lub prawie każdy z wielu atrybutów musi być uwzględniony, naiwny klasyfikator bayesowski może okazać się atrakcyjną alternatywą. Jest on szczególnie często z dobrymi efektami używany m.in. do klasyfikacji tekstu, gdzie atrybuty odpowiadają słowom występującym w zbiorze dokumentów i ich liczba sięga często od kilkuset nawet do wielu tysięcy.

W głównym nurcie wykładów ZUM będą przedstawione „nie całkiem naiwne” warianty naiwnego klasyfikatora bayesowskiego oparte na sieciach bayesowskich.

3 Modele liniowe

Algorytmy uczenia się wykorzystujące reprezentację modeli za pomocą liniowej kombinacji wartości atrybutów są szeroko stosowane w zadaniach regresji i klasyfikacji. Tutaj przyjrzymy się najprostszym algorytmom opartym na reprezentacji liniowej.

3.1 Regresja liniowa

W podstawowej postaci reprezentacja liniowa służy do zadania regresji i od tego przypadku zaczynamy.

Funkcja reprezentacji:

$$h(x) = \sum_{i=1}^{n} w_i a_i(x) + w_{n+1} = \sum_{i=1}^{n+1} w_i a_i(x) = \mathbf{w} \circ \mathbf{a}(x)$$

gdzie $\mathbf{w}$ – wektor parametrów $w_1, w_2, \ldots, w_n, w_{n+1}$, $\mathbf{a}(x)$ – wektor wartości (ciągłych) atrybutów $a_1(x), a_2(x), \ldots, a_n(x), a_{n+1}(x)$, przy czym $a_{n+1}(x) \equiv 1$, $\circ$ – symbol iloczynu skalarnego.

Dla liniowego model regresji predykcja jest obliczana jako liniowa kombinacja wartości atrybutów i parametrów modelu. O atrybutach zakładamy w tym przypadku, że są ciągłe. Liczba parametrów modelu liniowego jest o 1 większa od liczby atrybutów – parametr w_{n+1} jest składnikiem stałym (przesunięciem). Dla wygody i zwięzłości zapisu umówimy się teraz, że zestaw atrybutów będzie rozszerzony o dodatkowy atrybut a_{n+1} stale równy 1, co pozwala włączyć składnik stały modelu do sumy iloczynów wartości atrybutów i parametrów. Zauważmy także, że traktując wartości wszystkich atrybutów dla przykładu x, $a_1(x), a_2(x), \ldots, a_n(x), a_{n+1}(x)$, jako wektor $\mathbf{a}(x)$ i odpowiednio wartości parametrów $w_1, w_2, \ldots, w_n, w_{n+1}$ jako wektor $\mathbf{w}$, możemy predykcję modelu $h(x)$ przedstawić jako iloczyn skalarny wektorów $\mathbf{a}(x)$ i $\mathbf{w}$.

Estymacja parametrów: minimalizacja błędu na zbiorze trenującym:

$$E_{T,f}(h) = \frac{1}{2} \sum_{x \in T} (f(x) - h(x))^2$$

Rozważmy zagadnienie doboru parametrów dla liniowego modelu regresji, zakładając, że celem jest minimalizacja błędu średniokwadratowego na zbiorze trenującym. Dla technicznej wygody wykorzystamy jednak jego zmodyfikowaną postać, w której suma kwadratów residuów jest dzielona nie przez liczbę przykładów, lecz przez 2. Nie ma to oczywiście żadnego znaczenia z punktu widzenia tego, jaki model powstaje, gdyż zmiana polega wyłącznie na innym współczynniku skalującym, a prowadzi do większej wygody i elegancji zapisu.

Zadanie doboru parametrów minimalizujących błąd na zbiorze trenującym jest zadaniem optymalizacji numerycznej i istnieje szereg metod, które mogą być do tego zastosowane. Prostym standardowym podejściem jest np. zastosowanie metody spadku gradientu.

Reguła spadku gradientu:

$$\mathbf{w} := \mathbf{w} + \beta\left(-\nabla_{\mathbf{w}} E_{T,f}(h)\right)$$

gdzie $0 < \beta < 1$ – rozmiar kroku.

Metoda spadku gradientu w podstawowym wariancie polega na modyfikowaniu wektora parametrów minimalizowanej funkcji przez jego przesunięcie w kierunku, w którym spada jej wartość. W naszym przypadku minimalizowany ma być błąd, a wektorem jest wektor parametrów modelu. Kierunek spadku błędu wyznacza jego zanegowany gradient (wektor pochodnych cząstkowych) względem parametrów. Pojedynczą modyfikację wektora parametrów opisuje przedstawione wyżej reguła, w której rozmiar kroku β kontroluje wielkość przeprowadzanej modyfikacji.

Gradient błędu:

$$\nabla_{\mathbf{w}} E_{T,f}(h) = \sum_{x \in T} (f(x) - h(x))(-\nabla_{\mathbf{w}} h(x))$$

Gradient błędu względem parametrów modelu może być przez różniczkowanie funkcji kwadratowej przekształcony do powyższej postaci, w której występuje gradient predykcji modelu względem jego parametrów. Jego postać jest już oczywiście zależna od stosowanej dla modelu funkcji reprezentacji, o której w tym momencie trzeba założyć, że jest różniczkowalna. Przy okazji widać cel użycia mnożnika $\frac{1}{2}$ zamiast $\frac{1}{|T|}$ w zmodyfikowanym wzorze na błąd – uprościł się on z czynnikiem 2 uzyskanym z różniczkowania funkcji kwadratowej.

Gradient funkcji reprezentacji: $\nabla_{\mathbf{w}} h(x) = \mathbf{a}(x)$.

Reguła aktualizacji parametrów:

$$\mathbf{w} := \mathbf{w} + \beta(f(x) - h(x))\mathbf{a}(x)$$

Tryb stosowania: iteracyjnie, wielokrotnie dla wszystkich przykładów (aktualizacja przyrostowa po przetworzeniu każdego przykładu lub wsadowa po przetworzeniu zbioru przykładów).

Oczywiście pochodna cząstkowa $h(x)$ względem parametru w_i jest równa $a_i(x)$, a zatem gradient $h(x)$ względem $\mathbf{w}$ jest równy $\mathbf{a}(x)$. To pozwala zapisać regułę aktualizacji parametrów w podanej wyżej postaci. Reguła ta wymaga wielokrotnego stosowania, z odpowiednio dobranym rozmiarem kroku, w celu osiągnięcia wektora parametrów minimalizującego błąd średniokwadratowy.

Metoda najmniejszych kwadratów: wyznaczanie parametrów minimalizujących błąd średniokwadratowy za pomocą zamkniętej formuły, bez wielokrotnych aktualizacji.

Gradientowa aktualizacja nie jest jednak ani jedyną, ani (zazwyczaj) najlepszą metodą wyznaczania parametrów modelu liniowego minimalizujących błąd średniokwadratowy. Istnieje bardziej bezpośrednie podejście, umożliwiające ich wyznaczenie bez iteracyjnego procesu wielokrotnych aktualizacji. Jest to tzw. metoda najmniejszych kwadratów, która jest poza zakresem tego przeglądu, ale będzie przedstawiana w głównym nurcie wykładów ZUM.

3.2 Klasyfikacja liniowo-progowa

Klasyfikacja liniowo-progowa reprezentuje modele do uczenia się pojęć za pomocą hiperpłaszczyzn separujących klasy. Tego typu przestrzeń modeli pojawiała się już w przykładach dotyczących obliczeniowej teorii uczenia się (w najprostszym wariancie dla dwóch atrybutów – modele reprezentowane przez proste).

Wewnętrzna reprezentacja liniowa:

$$g(x) = \sum_{i=1}^{n} w_i a_i(x) + w_{n+1} = \mathbf{w} \circ \mathbf{a}(x)$$

Zewnętrzna funkcja progowa:

$$h(x) = \begin{cases} 1 & \text{jeśli } g(x) \geq 0 \\ 0 & \text{w przeciwnym przypadku} \end{cases}$$

Reprezentacja modeli za pomocą hiperpłaszczyzn separujących, którą teraz nazwiemy reprezentacją liniowo-progową, opiera się na złożeniu wewnętrznej funkcji liniowej i zewnętrznej funkcji progowej. Funkcja liniowa g oblicza kombinację liniową wartości atrybutów dokładnie tak samo jak model regresji liniowej.

Predykcja modelu h jest uzyskiwana przez „opakowanie" wewnętrznej funkcji liniowej g w zewnętrzną funkcję progową (skoku jednostkowego), która wartość swojego argumentu porównuje z ustalonym progiem (można nie zmniejszając ogólności rozważań przyjąć próg 0) i na podstawie tego porównania przypisuje przykładowi przewidywaną klasę 0 albo 1. Uzyskujemy w ten sposób model klasyfikatora liniowo-progowego.

Granica decyzyjna: hiperpłaszczyzna o równaniu $g(x) = 0$ (punkty po stronie dodatniej klasyfikowane do klasy 1, punkty po stronie ujemnej klasyfikowane do klasy 0).

W przypadku dziedziny o dwóch atrybutach a_1, a_2 wewnętrzna funkcja liniowa g wyznacza płaszczyznę, a jej przecięcie z płaszczyzną a_1-a_2 odpowiadające porównaniu $g(x)$ z 0 wyznacza prostą, którą nazwiemy prostą separującą lub granicą decyzyjną – przykłady położone po jej dodatniej stronie otrzymują przewidywaną klasę 1, a przykłady położone po jej ujemnej stronie – przewidywaną klasę 0.

Reprezentacja klasyfikatora liniowo-progowego dla przypadku dwóch atrybutów jest zilustrowana na rys. 1. Lewy górny wykres przedstawia płaszczyznę wyznaczoną przez wewnętrzną liniową funkcję reprezentacji. Prawy górny wykres przedstawia tę samą płaszczyznę z różnymi odcieniami szarości – ciemniejszym gdy wewnętrzna funkcja linowa przyjmuje wartości poniżej 0 i jaśniejszym w przeciwnym przypadku. Lewy dolny wykres przedstawia wynik zastosowania skoku jednostkowego do wartości wewnętrznej liniowej funkcji reprezentacji. Prawy dolny wykres przedstawia rzut na dwuwymiarową płaszczyznę wartości atrybutów, z użyciem ciemniejszego odcienia dla regionu, który otrzymuje klasę 0, oraz jaśniejszego dla regionu, który otrzymuje klasę 1, i prostą jako granicą decyzyjną separującą te regiony.

W przypadku n atrybutów g reprezentuje $(n+1)$-wymiarową hiperpłaszczyznę, a jej porównanie z 0 wyznacza n-wymiarową hiperpłaszczyznę separującą (granicę decyzyjną).

Wymiar VC: $n+1$ – ograniczone ryzyko nadmiernego dopasowania.

Przy okazji przykładów ilustrujących wymiar VC doszliśmy do wniosku, że dla przestrzeni modeli reprezentowanych przez proste (dla dwóch atrybutów) wynosi on 3, a w ogólności dla n-wymiarowych hiperpłaszczyzn separujących (n atrybutów) można pokazać, że wynosi on $n+1$. Liniowy wzrost wymiaru VC przy wzroście liczby atrybutów wskazuje na dość umiarkowane ryzyko nadmiernego dopasowania klasyfikatora liniowo-progowego.

Odległość od granicy decyzyjnej:

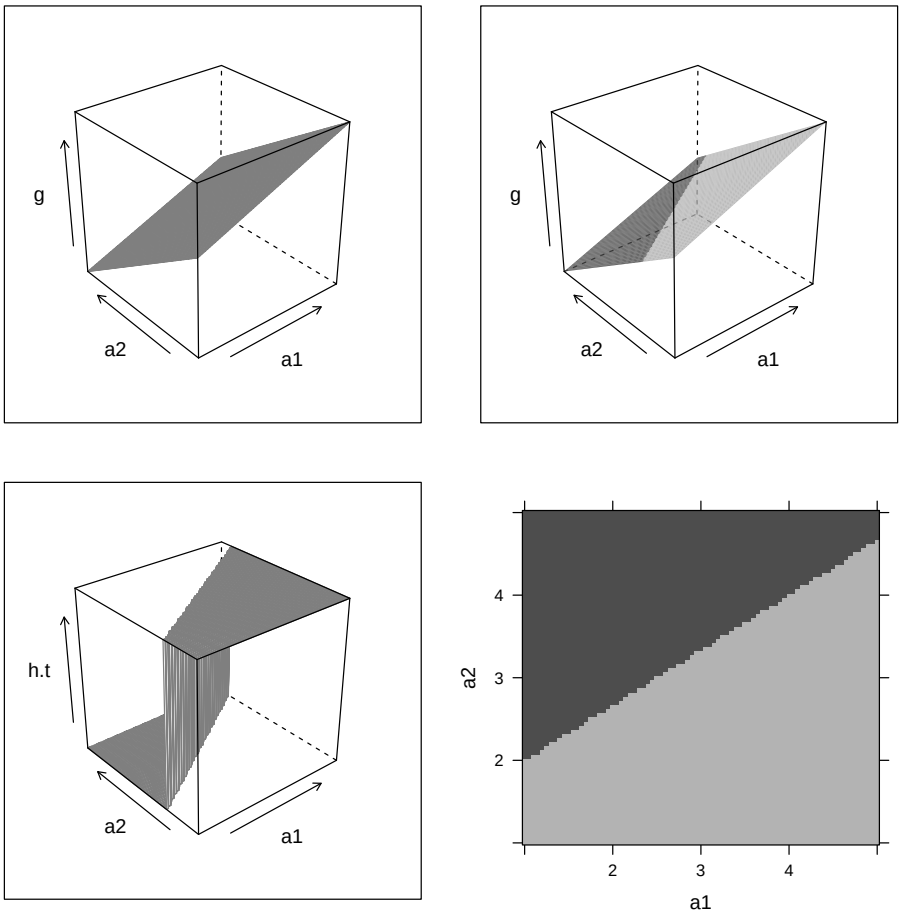
ze znakiem:

$$\delta_{\mathbf{w}}(x) = \frac{\sum_{i=1}^{n} w_i a_i(x) + w_{n+1}}{\sqrt{\sum_{i=1}^{n} w_i^2}} = \frac{\mathbf{w} \circ \mathbf{a}(x)}{\|\mathbf{w}_{1:n}\|}$$

gdzie $\mathbf{w}_{1:n}$ – wektor parametrów $w_1, w_2, \ldots, w_n$ (bez w_{n+1}),

bezwzględna dla przykładów niepoprawnie klasyfikowanych (tzn. jeśli $h(x) \neq c(x)$): $-c_-(x)\delta_{\mathbf{w}}(x)$, gdzie

$$c_-(x) = 2c(x) - 1 = \begin{cases} 1 & \text{jeśli } c(x) = 1 \\ -1 & \text{jeśli } c(x) = 0 \end{cases}$$



Rysunek 1: Reprezentacja klasyfikatora liniowo-progowego.

W celu przedstawienia algorytmu uczenia się dla reprezentacji liniowo-progowej będziemy odwoływać się do odległości przykładu od hiperpłaszczyzny separującej. Odległość przykładu x od granicy decyzyjnej reprezentowanej przez wektor parametrów $\mathbf{w}$ można obliczyć jako $\delta_{\mathbf{w}}(x)$ zgodnie z podanym wyżej standardowym wzorem na odległość punktu od hiperpłaszczyzny (uogólnieniem „szkolnego" wzoru na odległość punktu od prostej). Występuje tam norma kwadratowa (pierwiastek z sumy kwadratów elementów) wektora parametrów modelu, z pominięciem jednak składnika stałego w_{n+1}. W dalszym ciągu przyjmujemy, że wektor $\mathbf{w}$ zawiera także ten składnik, a wektor pierwszych n parametrów będzie oznaczany przez $\mathbf{w}_{1:n}$. W podobny sposób, w razie potrzeby, będziemy odwoływać się do wartości pierwszych n atrybutów przykładu x (z pominięciem sztucznego atrybutu a_{n+1} stale równego 0) pisząc $\mathbf{a}_{1:n}(x)$.

W podanej postaci $\delta_{\mathbf{w}}(x)$ to odległość ze znakiem, zgodnym ze znakiem $g(x) = \mathbf{w} \circ \mathbf{a}(x)$ – dodatnim jeśli przykład leży po dodatniej stronie granicy decyzyjnej i ujemnym jeśli

leży po jej ujemnej stronie. Weźmy pod uwagę odległość bezwzględną (bez znaku) dla przykładów *niepoprawnie klasyfikowanych* – a więc leżących po stronie dodatniej przykładów klady 0 i leżących po stronie ujemnej przykładów klasy 1. Łatwo zauważyć, że uzyskamy ją stosując do odległości ze znakiem mnożnik $-c_-(x)$, gdzie $c_-(x) = 2c(x) - 1$ jest przekształconym wariantem pojęcia docelowego, którego etykietami klas są liczby -1 i 1 zamiast 0 i 1. (Zaczynamy tu dla wygody zapisu korzystać z etykiet klas jako liczb, przy czym w dalszym ciągu będziemy zakładać dla klasyfikacji binarnej klasy 0 i 1, a wersję pojęcia z etykietami -1 i 1 oznaczać przez c_-.)

Przykład leżący po dodatniej stronie granicy decyzyjnej, którego prawdziwą klasą jest 0, uzyska po zastosowaniu takiego mnożnika dodatnią odległość, podobnie jak przykład leżący po ujemnej stronie granicy decyzyjnej, którego prawdziwą klasą jest 1.

Gradient do minimalizacji $-c_-(x)\delta_{\mathbf{w}}(x)$:

$$\nabla_{\mathbf{w}}(-c_-(x)\mathbf{w} \circ \mathbf{a}(x)) = -c_-(x)\mathbf{a}(x)$$

(pominięty mianownik $\|\mathbf{w}_{1:n}\|$)

Reguła aktualizacji parametrów (*prosty perceptron*):

$$\mathbf{w} := \begin{cases} \mathbf{w} + c_-(x)\mathbf{a}(x) & \text{jeśli } h(x) \neq c(x) \\ \mathbf{w} & \text{w przeciwnym przypadku} \end{cases}$$

Inicjalizacja: dowolna (np. 0).

Efekt aktualizacji: zmniejszenie odległości niepoprawnie klasyfikowanych przykładów od granicy decyzyjnej.

Tryb stosowania: iteracyjnie wielokrotnie dla wszystkich przykładów, iteracyjnie wielokrotnie dla losowo wybieranych przykładów.

Kryterium stopu: poprawna klasyfikacja wszystkich przykładów.

Zbieżność: gwarantowana tylko jeśli w zbiorze trenującym klasy są liniowo separowalne.

Podstawowy algorytm uczenia się klasyfikatora liniowo-progowego, nazywany *prostym perceptronem*, opiera się na dążeniu do minimalizacji odległości od granicy decyzyjnej dla tych przykładów trenujących, które znajdują się po jej niewłaściwej stronie. Zmniejszając tę odległość możemy oczekiwać, że uda się doprowadzić do takiego przesunięcia granicy decyzyjnej, aby w końcu wszystkie przykłady znalazły się (o ile to możliwe) po jej poprawnej stronie.

Koncepcję minimalizacji odległości błędnie klasyfikowanych przykładów od granicy decyzyjnej realizuje przedstawiona wyżej reguła aktualizacji, oparta na zasadzie spadku gradientu. Wektor parametrów modelu przesuwany jest w kierunku, w którym następuje spadek minimalizowanej funkcji, wyznaczanym przez jej zanegowany gradient – wektor pochodnych cząstkowych względem parametrów modelu. Przy wyznaczaniu gradientu odległości pominięty został mianownik $\|\mathbf{w}_{1:n}\|$ jako niezależny od przykładów trenujących czynnik skalujący. Tymczasem oczywiście dowolne skalowanie parametrów modelu (mnożenie wszystkich $w_1, w_2, \ldots, w_n, w_{n+1}$ przez dowolny dodatni współczynnik) nie zmienia położenia granicy decyzyjnej.

Przedstawiona reguła aktualizacji stosowana iteracyjnie gwarantuje znalezienie w skończonym czasie parametrów modelu poprawnie klasyfikującego przykłady decyzyjne, o ile istnieje linowa granica decyzyjna poprawnie separujące klasy. O danych, dla których to jest możliwe, mówi się, że są *liniowo separowalne*. W przypadku braku liniowej separowalności algorytm nie daje żadnych gwarancji dotyczących poziomu dopasowania do danych – zawsze będą przykłady niepoprawnie klasyfikowane, więc kryterium stopu algorytmu – polegające na poprawnej klasyfikacji całego zbioru trenującego – nie zostanie osiągnięte.

Ze względu na ograniczenie możliwości użycia wyłącznie do danych separowalnych liniowo prosty perceptron jako algorytm uczenia się pojęć nie ma praktycznego znaczenia, ale stanowi dla nas wstęp do przedstawienia w głównym nurcie wykładów ZUM bardziej rozwiniętych i w pełni przydatnych praktycznie algorytmów regresji logistycznej oraz SVM opartych również na reprezentacji liniowej.