Uczenie maszynowe

Skrypt na podstawie wykładu Profesora Andrew Ng
na Wydziale Informatyki Stanford University

Autor: Elżbieta Jowik
Konsultacje merytoryczne: Anna Wróblewska

Spis treści

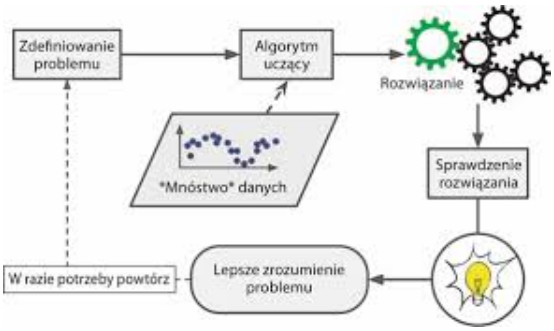
1 Wprowadzenie

Uczenie maszynowe (ang. *Machine Learning, ML*) jest dziedziną trudną do jednoznacznego zdefinowania. Jego główną ideą jest konstruowanie programów, które w oparciu o zdobywane doświadczenie mają dawać coraz lepsze rezulaty względem pewnej miary.

w pewnym uproszczeniu, uczenie maszynowe jest zbiorem narzędzi, którym przyświeca filozofia znajdowania schematów, bez oczekiwania odnalezienia w nich znaczenia (rys. 1).

Rysunek 1: Ogólna idea uczenia maszynowego

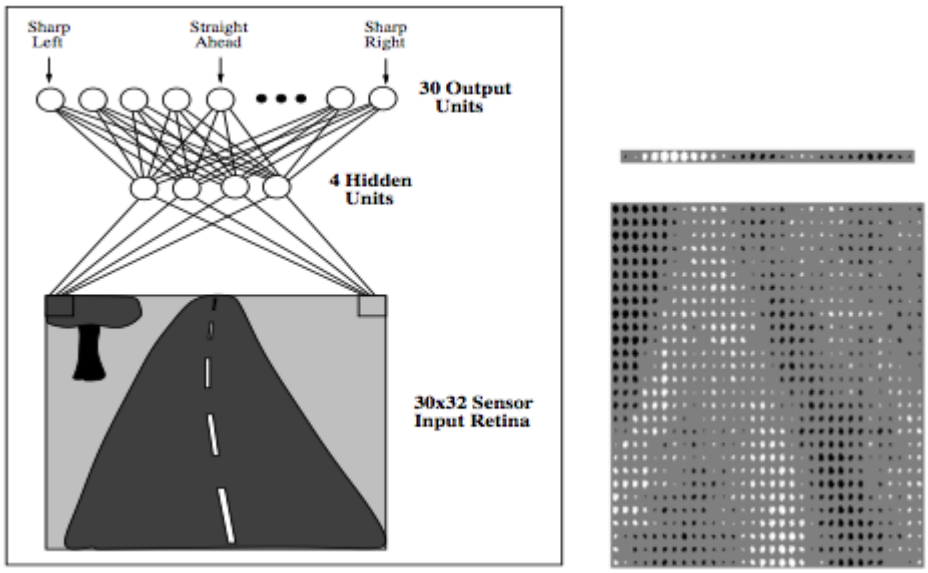
W praktyce ML to zestaw gotowych algorytmów popartych dowodami matematycznymi, spośród których większość została opracowana już w latach 70-tych. Jednak ze względu na brak dostatecznej mocy obliczeniowej w tamtym okresie, zastosowanie znajdują dopiero teraz.



Rysunek 2: Realizacja ML - cykl uczenia i testowania modelu

Ciekawostka

Uczenie maszynowe wydaje się być młodą dziedziną, a autonomiczne samochody objawieniem współczesnej nauki. Warto jednak wiedzieć, że już w 1989 roku w Stanach Zjednoczonych testowano samochód bazujący na sieciach neuronowych, który uczył się samodzielnej jazdy. Proces nauki (rys. 3) polegał na tym, że prowadzony przez człowieka ALVINN (ang. *Autonomous Land Vehicle In a Neural Network*) raz na 2 sekundy fotografował drogę przed sobą, a następnie do tego obrazu przypisywał aktualny kąt skrętu kierownicy. Jest to przykład uczenia nadzorowanego, a konkretnie problem regresji (do cechy, jaką jest obraz zapewniamy odpowiedź w postaci kąta, będącego zmienną ciągłą).



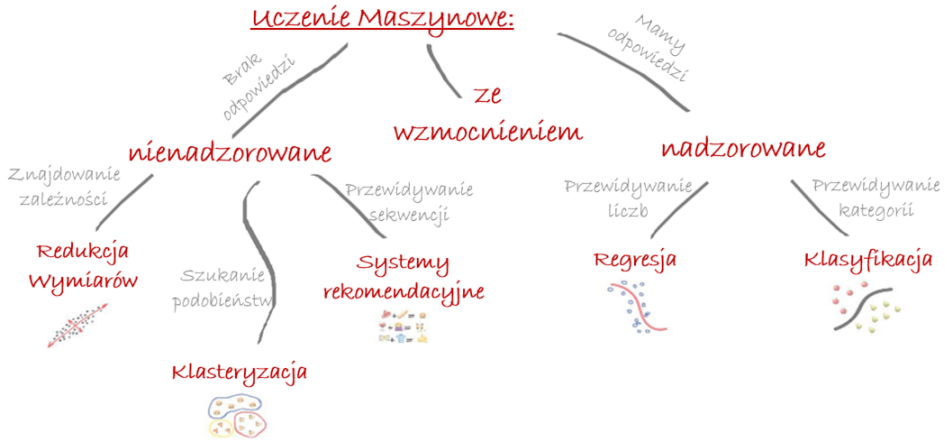
Rysunek 3: ALVINN - sieć neuronowa przewiduje przypisanie do obrazu drogi kąta skrętu kierownicy, np. mocno w lewo (ang. *Sharp Left*), prosto (ang. *Straight Ahead*) czy mocno w prawo (ang. *Sharp Right*). Rozważana sieć składała się z czterech warstw ukrytych (ang. *Hidden Units*)

2 Sposoby uczenia

Proces uczenia maszynowego polega na zbieraniu danych przez systemy komputerowe. Dane te następnie są agregowane przez ten sam system i wykorzystywane do ulepszania własnego działania.

Podobnie jak ludzie, którzy wraz z upływem lat zdobywają nowe doświadczenia, tak algorytmy samouczące się, im więcej danych zbiorą, tym trafniejsze decyzje podejmą.

w uproszczeniu, zanim algorytm będzie mógł służyć do predykcji, w pierwszej kolejności należy go wytrenować odpowiednią ilością danych. Istnieją trzy główne metody uczenia się modeli – przedstawione na rys. 4.



Rysunek 4: Podział modeli uczenia maszynowego

2.1 Uczenie nadzorowane (ang. *Supervised Learning*)

System na wstępie otrzymuje dane zarówno wejściowe (zmienne objaśniające), jak i wyjściowe (zmienną celu, zmienną objaśnianą). Jego zadaniem jest utworzenie odpowiednich reguł (generalizacja), które mapują wejście x na wyjście y na podstawie próby uczącej $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{m}$, gdzie m jest ilością danych – par wejść i wyjść $(x^{(i)}, y^{(i)})$ – w zbiorze uczącym.
po odpowiednim wytrenowaniu system taki powinien móc prawidłowo przypisać wyjście dla obiektu, którego dotychczas nie było na wejściu – wykonać predykcję.

Wartość y może być wartością dyskretną, czyli przyjmującą wartości z pewnego skończonego zbioru, np. {kobieta, mężczyzna}, albo wartością liczbową. w pierwszym przypadku problem uczenia nazywamy problemem klasyfikacji albo rozpoznawania wzorców, a w drugim problemem regresji.

2.2 Uczenie przez wzmacnianie (ang. *Reinforcement Learning*)

Ten sposób zakłada, że system działa w środowisku nieznanym. Brak jest określonych danych wejściowych i wyjściowych. Jedyną informacją, jaką otrzymuje algorytm jest sygnał wzmocnienia, który może być pozytywny w przypadku podejmowania trafnych decyzji lub negatywny w przypadku pomyłki. Celem jest takie działanie, aby maksymalizować pozytywne wzmocnienia (nagrody), a minimalizować negatywne (kary).

2.3 Uczenie nienadzorowane (ang. *Unsupervised Learning*)

W tym przypadku nieznana jest ani funkcja nagrody, ani właściwe dane wyjściowe (etykiety). Algorytm na podstawie zbioru wejść $\mathcal{D} = \{x^{(i)}\}_{i=1}^{m}$
musi samodzielnie znaleźć odpowiednią regułę, która cechuje wejście i w miarę możliwości ją zgeneralizować.

Zadanie to jest gorzej określone niż uczenie nadzorowane, ponieważ nie ma jasnego kryterium oceny jakości uczenia.

3 Rodzaje problemów uczenia maszynowego

3.1 Klasyfikacja

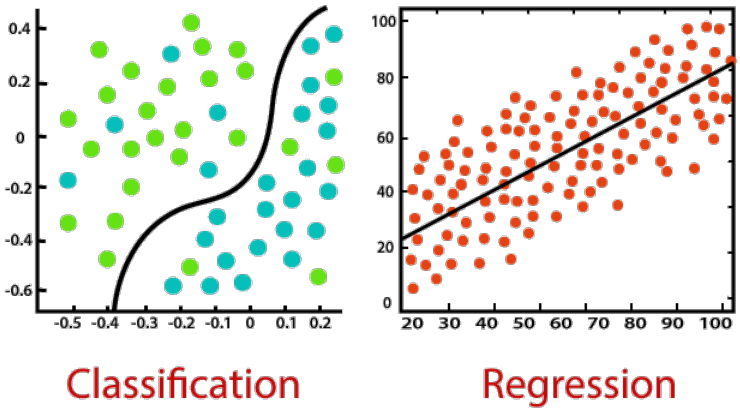
Realizacja zadania klasyfikacji (ang. *Classification*) (rys. 5) polega na automatycznym określaniu przynależności jakiegoś reprezentanta do danej klasy na podstawie wcześniej zgromadzonych informacji. Dane uczące są oznaczone właściwymi etykietami określającymi, do jakiej klasy należą. w trakcie nauki model modyfikuje swoją strukturę tak, aby jego odpowiedź była poprawna.

W ogólności, z klasyfikacją mamy do czynienia wówczas, gdy mamy dyskretny zbiór możliwych predykcji. O klasyfikacji binarnej (ang. *Binary Classfication*) mówimy wtedy, gdy klasy docelowe - zmienne celu są dwie, np. {0, 1}, klasy pozytywna i negatywna. Problem klasyfikacji wieloklasowej (ang. *Multi-Class*) występuje wtedy, gdy mamy więcej niż 2 rozłączne klasy. Klasyfikacja wieloetykietowa (ang. *Multi-Label*) występuje wtedy, gdy klasy nie są rozłączne, tzn. każdej obserwacji (danej wejściowej) możemy przypisać wiele etykiet.

3.2 Regresja

Zadaniem regresji jest przewidywanie wartości ciągłych, a nie skokowych jak w klasyfikacji. Regresja (ang. *Regression*) jest sposobem aproksymacji danych

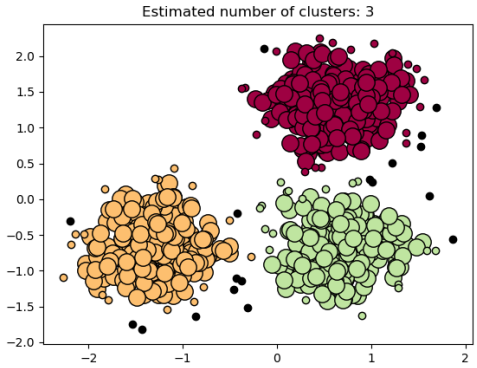
(rys. 5), czyli ich przybliżania. Klasyczne działanie algorytmu regresji polega na stworzeniu modelu w taki sposób, aby zminimalizować błąd średniokwadratowy dla wielkości, które są znane dla algorytmu (czyli dla zbioru uczącego).



Rysunek 5: Klasyfikacja i regresja

3.3 Klasteryzacja

Klasteryzacja (ang. *Clustering*) polega na automatycznym dzieleniu zbiorów danych na grupy (rys. 6). w odróżnieniu od problemu klasyfikacji, grupy te nie są znane — to algorytm jest odpowiedzialny za ich znalezienie. Najczęstsze realizacje zadania klasteryzacji oparte są na odnajdowaniu cech różniących dane i analizie skupień.



Rysunek 6: Klasteryzacja - kolorami zaznaczone są odzielne grupy obserwacji/punktów danych

3.4 Redukcja wymiarowości (ang. *Dimensionality Reduction*)

Redukcja wymiarowości jest to zbiór technik pozwalających na usuwanie ze zbioru cech, które nie są informatywne z punktu widzenia predykcji zmiennej celu. Polega to najczęściej na usuwaniu tych, które są od siebie zależne, ponieważ wtedy jedna z nich wnosi taką samą informację jak druga.

Obowiązująca notacja:

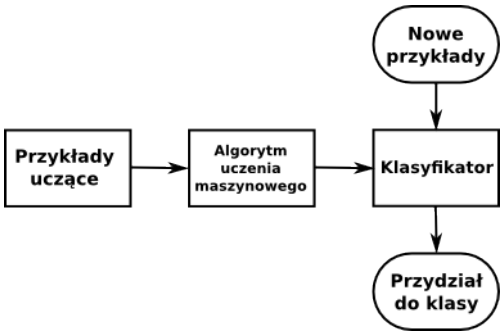
m	liczba obserwacji treningowych
x	zmienne objaśniające
y	zmienna celu
(x, y)	obserwacja treningowa
$(x^{(i)}, y^{(i)})$	i-ta obserwacja
#	liczność
$\mathcal{H}$	przestrzeń hipotez

4 Hipoteza i jej reprezentacja

Living area ($feet^2$)	$ (1000s)
2104	400
1416	232
1539	315
852	178
1940	240
⋮	⋮

Tabela 1: Próbka przykładowego zbioru danych. Wiersze reprezentują obserwacje (ang. *data points*), a kolumny zmienne objaśniające (ang. *features*)

Rozważmy problem predykcji ceny mieszkań na podstawie ich metraży. Załóżmy, że dysponujemy pewnymymi danymi z tabeli **??**.
Chcemy zachować metodologię przedstawioną poniżej (rys. 7):
Klasyfikator z powyższego schematu (rys. 7) można utożsamiać z pewną hipotezą. Hipoteza ta jest wypadkową (w tym przypadku kombinacją liniową) zmiennych objaśniających, wyznaczoną na podstawie przykładów uczących $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{m}$.
Oznaczmy hipotezę przez h.



Rysunek 7: Modelem jest system, który tworzy predykcje. Parametry, określane na podstawie dostarczanych do systemu danych uczących, to czynniki na podstawie których model formułuje swoje decyzje. Oczekujemy, że w momencie gdy pojawi się dana z poza zbioru uczącego model będzie w stanie ją odpowiednio sklasyfikować.

W powyższym przykładzie (1), w którym metraż mieszkania jest jedyną zmienną objaśniającą, liniowa reprezentacja hipotezy jest następująca:

$$h(x) = \theta_0 + \theta_1 x$$

gdzie x - cecha, θ_i - parametry.

W sytuacji, gdy mielibyśmy do czynienia z większą ilością cech np.:

Living area ($feet^2$)	$ (1000s)	# of bedrooms
2104	400	3
1416	232	2
1539	315	3
852	178	2
1940	240	4
⋮	⋮	⋮

Tabela 2: Próbka przykładowego zbioru danych. Wiersze reprezentują obserwacje (ang. *data points*), a kolumny zmienne objaśniające (ang. *features*)

mielibyśmy:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

A zatem w ogólności, przy założeniu $x_0 = 1$ liniowa reprezentacja hipotezy wygląda jak następuje:

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T \mathbf{x}$$

gdzie θ - parametry, x - zmienne objaśniające, n - # cech

5 Problem optymalnej parametryzacji

W powyższych rozważaniach θ-y oznaczają parametry.

Przyjmuje się, że wybór parametryzacji jest zdeterminowany przez lokalne minimum funkcji błędu (funkcji straty, ang. *Loss*):

$$J(\theta) = \frac{1}{2} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

gdzie zgodnie z przyjętą notacją m oznacza liczbę obserwacji treningowych.

Innymi słowy parametry θ dobieramy w taki sposób, aby osiągnąć:

$$\min_{\theta} J(\theta) = \min_{\vec{\theta}} \frac{1}{2} \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

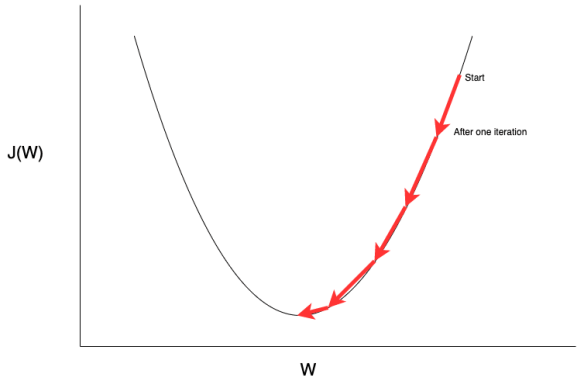
czyli jak najmniejszą sumę kwadratów różnic predykcji i wartości faktycznych.

Uwagi:

- Jest to minimum lokalne, a nie globalne, więc ostateczne wartości parametrów mogą różnić się w zależności od wyboru wektora początkowego.
- $\frac{1}{2}$ przed sumą służy ułatwieniu obliczeń, nie ma większego znaczenia z punktu widzenia znajdowania optymalnych parametrów.

5.1 Metoda najszybszego spadku gradientu (ang. *Gradient Descent*)

Iteracyjny algorytm numeryczny mający na celu znalezienie minimum zadanej funkcji celu to algorytm największego spadku gradientu (rys. 8).



Rysunek 8: Ilustracja działania metody najszybszego spadku dla dwuwymiarowej funkcji celu. W każdym kroku, w zadanym kierunku wyszukiwana jest najmniejsza wartość funkcji celu.

Jako wektor początkowy przyjmujemy

$$\theta_0 = \vec{0}$$

natomiast w i-tej iteracji:

$$\theta_i := \theta_{i-1} - \alpha \frac{\partial}{\partial \theta_{i-1}} J(\theta)$$

gdzie α jest współczynnikiem długości kolejnych kroków (ang. *Learning Rate*).

Iteracje powtarzamy do uzyskania zbieżności.

Ponieważ:

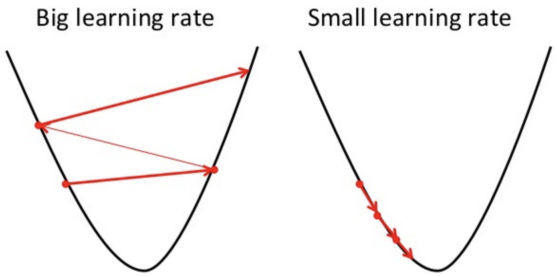
$$\begin{aligned} \frac{\partial}{\partial \theta_i} J(\theta) = \frac{\partial}{\partial \theta_i} \frac{1}{2} (h_\theta(x) - y)^2 = 2 \cdot \frac{1}{2} \cdot (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_i} (h_\theta(x) - y) = \\ = (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_i} (\theta_0 x_0 + \ldots + \theta_n x_n - y) = (h_\theta(x) - y) \cdot x_i \end{aligned} \tag{1}$$

to równoważnie:

$$\theta_i := \theta_{i-1} - \alpha(h_\theta(x) - y) \cdot x_{i-1}$$

$$\theta_i := \theta_{i-1} - \alpha \sum_{j=0}^{m} (h_\theta(x^{(j)}) - y^{(j)}) \cdot x_{i-1}^{j}$$

Dygresja
Oto jak współczynnik α wpływa na działanie metody (rys. 9):



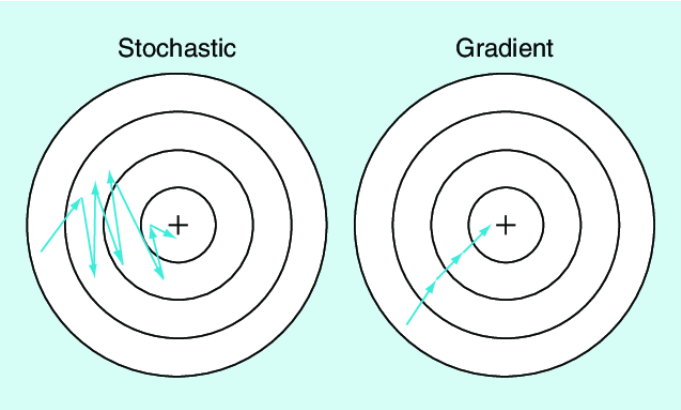
Rysunek 9: Wielkość kroku jest dodatnim skalarem. Wartość ta jest dobierana tak, aby otrzymać największą wartość spadku wartości funkcji, w każdym kolejnym punkcie.

Powyższa metoda to **Batch Gradient Descent**. Wykorzystuje ona w każdym kroku cały zbiór treningowy i w ogólności działa bardzo dobrze. w praktyce jednak zdarza się, że gdy zbiór treningowy jest bardzo duży, np. $m = 10^6$, metoda ta okazuje się zbyt obciążająca obliczeniowo.
Alternatywnie wykorzystuje się wówczas mniej wymagający algorytm **Stochastic Gradient Descent** (alg. **??**), który zmienia parametry modelu po przeliczeniu dla każdej próbki danych (obserwacji).

For j = 1 to m:

$$\theta_i := \theta_{i-1} - \alpha(h_\theta(x^{(j)}) - y^{(j)}) \cdot x_{i-1}$$

dla każdego i do uzyskania zbieżności.



Rysunek 10: Porównanie metod standardowej i stochastycznej. W środku znajduje się lokalne minimum funkcji celu. Metoda stochastyczna (ang. *Stochastic gradient descent*) dokonuje zmian w każdym kroku (po predykcji dla każdej obserwacji) - stąd zmiany są bardziej chaotyczne. Metoda standardowa (ang. *Batch gradient descent*) oblicza gradient po wykonaniu predykcji dla całego zbioru treningowego

5.2 Równanie normalne (ang. *Normal Equation*)

Stanowi ono inne podejście do problemu parametryzacji. Umożliwia bezpośrednie znalezienie θ bez potrzeby stosowania algorytów iteracyjnych typu gradient descent.

$$\theta = (x^T x)^{-1}(x^T y)$$

gdzie x - zmienne objaśniające, y - zmienne celu.

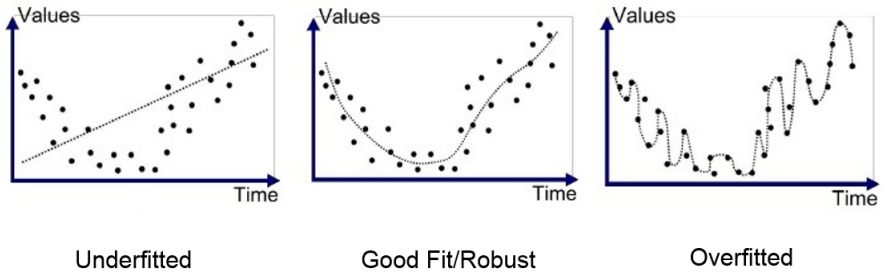
6 Problem dopasowania modelu do danych

6.1 Przeuczenie (ang. *Overfitting*)

Zjawisko nadmiernego dopasowania modelu do danych treningowych objawia się wykrywaniem pozornych prawidłowości w dużej ilości danych, gdzie faktyczne prawidłowości są prostsze, słabsze lub nie istnieją (rys. 11). Jest to możliwe, gdy bogata przestrzeń hipotez $\mathcal{H}$ zawiera, między innymi hipotezy h, dużo bardziej złożone niż poszukiwana funkcja f. w takim przypadku mówi się, że przeuczony model nie posiada zdolności generalizacji.

6.2 Niedouczenie (ang. *Underfitting*)

Stanowi przeciwieństwo przetrenowania i występuje ono wtedy, gdy model jest zbyt prosty, aby wyuczyć się struktur danych uczących (rys. 11). Może to wynikać z: niewystarczającej liczby próbek uczących lub ze zbyt uproszczonego modelu zastosowanego w uczeniu (zbyt ubogiej przestrzeni hipotez $\mathcal{H}$).



Rysunek 11: Problem dopasowania modelu do danych. Przeuczenie (ang. *Overfitting*) polega na nadmiernym dopasowaniu danych przez program, niedouczenie (ang. *Underfitting*) natomiast polega na zbyt słabym dopasowaniu do danych.

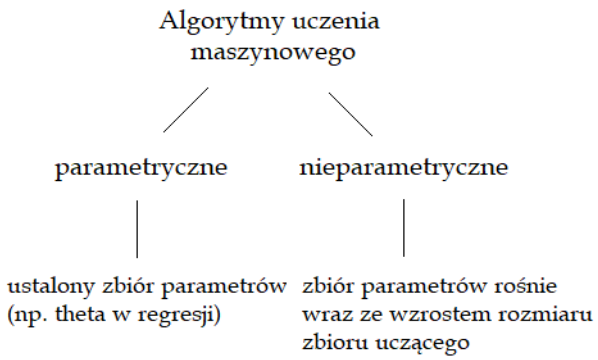
Metody oparte na modelach wykorzystują dane do zbudowania sparametryzowanego modelu. po wytrenowaniu, model służy do predykcji, a dane treningowe przestają być potrzebne.

Metody oparte na pamięci stanowią reprezentację podejścia nieparametrycznego. Jawnie przechowują one dane treningowe i wykorzystują je za każdym razem, gdy trzeba dokonać predykcji.

7 Regresja lokalnie ważona (ang. *Locally Weighted Regression, LWR*)

W poprzednim wykładzie część rozważań zorientowana była na wyznaczenie h dla konkretnego x. w problemie regresji liniowej chcieliśmy zwrócić $\theta^T x$, gdzie dobór θ był zdeterminowany chęcią minimalizacji wyrażenia $\sum_{i=0}^{n}(y^i - \theta^T x^i)^2$. Dobierając odpowiednie parametry rozważaliśmy wówczas dopasowanie h do całego zbioru treningowego.

W przypadku regresji lokalnie ważonej rozważamy nie cały zbiór uczący, a jedynie sąsiedztwo obserwacji, dla której mamy dokonać predykcji i stosujemy regresję liniową tylko na tym podzbiorze obserwacji (rys. 13).



Rysunek 12: Podział algorytmów ML ze względu na parametryzację

Regresja Lokalnie Ważona, ang. *Locally Weighted Regression* (LWR) jest nieparametryczą metodą, która wykonuje regresję w otoczeniu interesującej nas obserwacji przy użyciu tylko danych treningowych, które są "lokalne" dla tego punktu.

W regresji lokalnie ważonej waga punktów określana jest, za pomocą jądra, na podstawie ich bliskości względem rozważanej obserwacji x. Następnie przy użyciu punktów ważonych rozważamy problem regresji liniowej.
Formalnie, waga danego punktu wyraża się wzorem

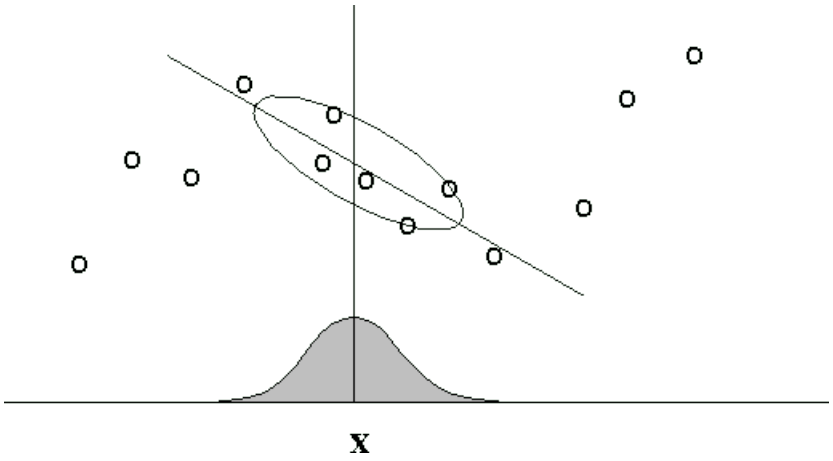
$$w^{(i)} = exp\{-\frac{(x^{(i)} - x)^2}{2\tau^2}\}$$

gdzie parametr τ odpowiada za to, jak szybko wraz z odległością maleje waga.

Zauważmy, że:
jeśli punkty są sobie bliskie - $|x^{(i)} - x|$ jest małe, wówczas $w^{(i)} \approx 1$
natomiast jeśli punkty są odległe, tzn. $|x^{(i)} - x|$ jest duże, wówczas $w^{(i)} \approx 0$

Analogicznie do klasycznej LR, w przypadku procedury LWR dobór parametrów θ regresji liniowej również zorientowany jest na minimalizację wartości pewnej funkcji. Wyraża się ona następującym wzorem:

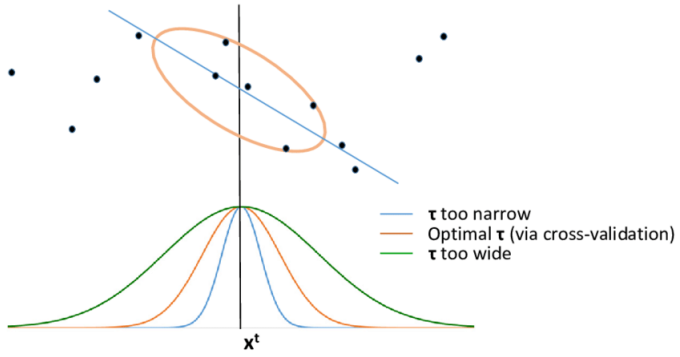
$$\sum_{i=0}^{n} w^{(i)}(y^{(i)} - \theta^T x^{(i)})^2$$



Rysunek 13: Regresja lokalnie ważona (ang. *Locally Weighted Regression*), dla sąsiedztwa rozważanaego punktu budujemy regresję liniową

Uwaga:

Wpływ parametru τ na funkcję wagi przedstawiony jest na rys. 14.



Rysunek 14: Ważnym parametrem w definiowaniu sąsiedzwa w algorytmie LWR jest funkcja jest szerokość pasma *tau* (optymalizowana zazwyczaj w procesie kroswalidacji). Odpowiada ona zakresowi danych wokół punktu zainteresowania, używanemu do przeprowadzenia regresji liniowej. Duża szerokość pasma powoduje włączenie dużej części zbioru treningowego do regresji liniowej, podczas gdy mniejsza szerokość pasma powoduje znacznie bardziej lokalną regresję ważoną wokół punktu zainteresowania x^t

8 Probabilistyczna interpretacja problemu parametryzacji

Załóżmy, że

$$y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$$

gdzie $\varepsilon^{(i)}$ - błąd.

Ponieważ, w praktyce, często błędy są dobrze przybliżalne za pomocą rozkładu Gaussa, przyjmijmy, że:

$$\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$$

Wówczas:

$$\mathbf{P}(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\}$$

Stąd:

$$\mathbf{P}(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\}$$

Zatem:

$$y^{(i)}|x^{(i)};\theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$$

Jeśli $\varepsilon^{(i)}$ są i.i.d., to funkcja wiarygodności:

$$L(\theta) = P(\vec{y}|X;\theta) = \prod_{i=1}^{m} \mathbf{P}(y^{(i)}|x^{(i)};\theta)$$

Metoda największej wiarygodności (ang. *Maximum Likelihood*) polega na wyborze takich θ, dla których $L(\theta) = P(\vec{y}|X;\theta)$ jest maksymalne. Maksymalizując L, zwykle szuka się maksimum dla funkcji $l(\theta) = \log L$, ponieważ osiągają one maksimalną wartość w tym samym punkcie.

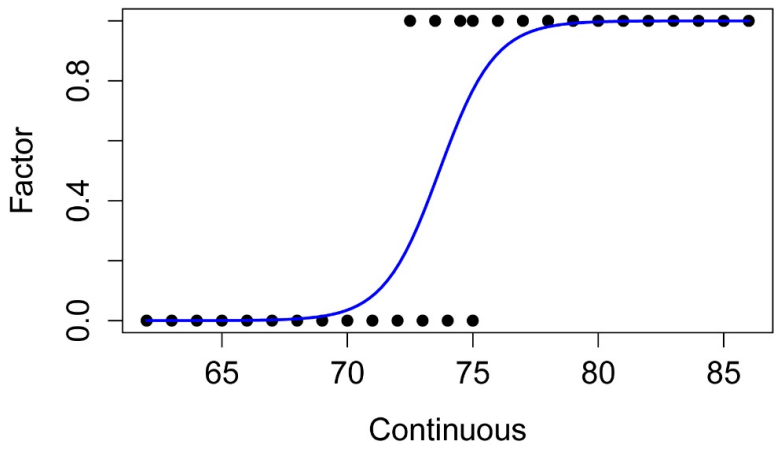
Niech:

$$\begin{aligned} l(\theta) = \log L(\theta) &= \log(\prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\}) = \\ &= \sum_{i=1}^{m} \log(\frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\}) = \\ &= m \log(\frac{1}{\sqrt{2\pi}\sigma}) + \sum_{i=1}^{m} -\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} \end{aligned}$$

Zatem maksymalizacja $l(\theta)$ jest równoważna minimalizacji

$$\frac{1}{2} \sum_{i=1}^{m} (y^i - \theta^T x)^2 = J(\theta)$$

9 Regresja logistyczna (ang. *Logistic Regression*)



Rysunek 15: Krzywa sigmoidalna

Regresja logistyczna:

- służy klasyfikacji binarnej $y \in \{0, 1\}$, a nie predykcji wartości ciągłych,
- do danych dopasowuje funkcję logistyczną/sigmoidalną,
- krzywa mówi o prawdopodobieństwie, z jakim dana obserwacja należy do klasy pozytywnej.

Predykcja modelu spełnia zatem warunek:

$$0 \leqslant h_\theta(x) \leqslant 1$$

i wyraża się wzorem:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{(1 + e^{-\theta^T x})} = \mathbf{P}(y = 1|x; \theta)$$

Stąd:

$$\mathbf{P}(y = 0|x; \theta) = 1 - h_\theta(x)$$

$$\mathbf{P}(y|x; \theta) = h_\theta(x)^y (1 - h_\theta(x))^{(1-y)}$$

$$L(\theta) = \mathbf{P}(\vec{y}|X; \theta) = \prod_i \mathbf{P}(y^{(i)}|x^{(i)}; \theta) = \prod_i h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)})^{y^{(i)}} + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

Funkcja L odpowiada prawdopodobieństwu $\mathbf{P}(\vec{y}|X;\theta)$, zatem przy doborze parametrów dąży się do jej maksymalizacji. Stosuje się, do tego omówiony wcześniej algorytm gradient descent, z tym, że z modyfikacją znaku względem wersji znajdującej się powyżej, mianowicie:

$$\theta_j := \theta_{j-1} + \alpha \frac{\partial}{\partial \theta_{j-1}} l(\theta)$$

gdzie

$$\frac{\partial}{\partial \theta_j} l(\theta) = \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$$

Zatem:

$$\theta_j := \theta_{j-1} + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$$

10 Dygresja: Perceptron prosty

Jest najprostszą siecią jednokierunkową, zbudowaną jest jedynie z warstwy wejściowej i warstwy wyjściowej. Elementem składowym perceptronu jest sztuczny neuron, którego model matematyczny może być opisany funkcją aktywacji:

$$g(z) = \begin{cases} 1 & z \geqslant 0 \\ 0 & w.p.p. \end{cases}$$

W algorytmie tym:

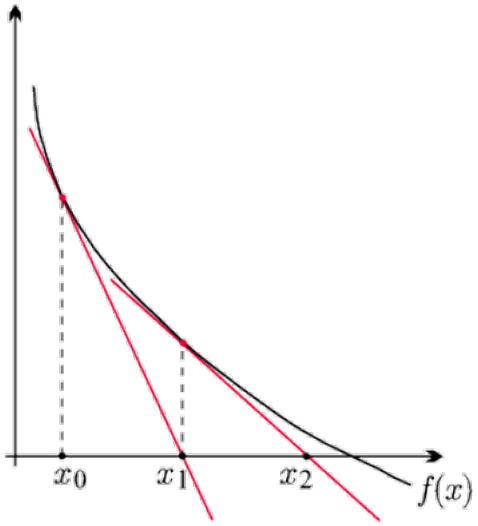
$$h_\theta(x) = g(\theta^T x)$$

$$\theta_j := \theta_{j-1} + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_{j-1}^{(i)}$$

Zwykły neuron z pigmoidalną funkcją aktywacji jest równoważny z regresji logistycznej.

11 Metoda Newtona

Iteracyjna metoda służąca znajdowaniu miejsc zerowych funkcji o następującej metodologii (rys. 16).



Rysunek 16: Ilustracja działania metody Newtona, pokazane zostały 2 pierwsze kroki.

Metoda Newtona przedstawiona jest poniższym algorytmem na listingu alg. **??**.

$$\Delta = \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$$

$$\theta^{(1)} = \theta^{(0)} - \frac{f(\theta^{(0)})}{f'(\theta^{(0)})}$$

$$\theta^{(t+1)} = \theta^{(t)} - \frac{f(\theta^{(t)})}{f'(\theta^{(t)})}$$

W celu maksymalizacji funkcji różniczkowalnej często stosowaną praktyką jest przyrównanie jej pochodnej do 0. Zatem w naszym przypadku, aby zmaksymalizować funkcję $l(\theta)$ chcemy znaleźć parametr θ t.że: $l'(\theta) = 0$.
Stosując metodę Newtona otrzymujemy:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{l'(\theta^{(t)})}{l''(\theta^{(t)})}$$

A w przypadku wielowymiarowym:

$$\theta^{(t+1)} = \theta^{(t)} + H^{-1}\nabla_\theta l$$

gdzie H-macierz hessego:

$$H_{ij} = \frac{\partial^2 f}{\partial \theta_i d\theta_j}$$

Metoda Newtona jest bardzo szybko zbieżna (kwadratowo).

12 Uogólnione modele liniowe (ang. *Generalized Linear Model, GLR*)

Dotychczas rozważaliśmy przykłady regresji gdzie $y \in \mathcal{R}$ oraz $y \sim \mathcal{N}(\mu, \sigma^2)$ lub binarnej klasyfikacji, gdzie $y \in \{0, 1\}$ oraz $y \sim Bern(\phi)$. w pierwszym przypadku wykorzystywaliśmy metodę najmniejszych kwadratów, w drugim regresję logistyczną. Okazuje się jednak, że obydwa te problemy są szczególnymi przypadkami szerszej klasy modeli, tzw. uogólnionych modeli liniowych.

Fakt

Postać gęstości świadcząca o przynależności do wykładniczej rodziny rozkładów:

$$p(y; \eta) = b(y) exp\{\eta^T T(y) - a(\eta)\}$$

Dowód przynależności rozkładu Bernoulliego do wykładniczej rodziny rozkładów

$Ber(\phi) \Rightarrow p(y = 1; \phi) = \phi$

$$\mathbf{P}(y, \phi) = \phi^y (1 - \phi)^{(1-y)} = exp\{\log(\phi^y (1 - \phi)^{(1-y)})\} =$$

$$= exp\{y \log(\phi) + (1 - y) \log(1 - \phi)\} = exp\{log(\frac{\phi}{1 - \phi})y + \log(1 - \phi)\}$$

Stąd:

$b(y) = 1, \eta = log(\frac{\phi}{1-\phi}), T(y) = y, -a(\eta) = \log(1 - \phi)$

Zatem $Ber(\phi)$ należy do wykładniczej rodziny rozkładów.

Dowód przynależności rozkładu Gaussa do wykładniczej rodziny rozkładów

Niech $\sigma^2 = 1$.

Wówczas:

$p(y, \mu) = \frac{1}{\sqrt{2\pi}} exp\{-\frac{1}{2}(y - \mu)^2\} = \ldots = \frac{1}{\sqrt{2\pi}} exp\{-\frac{1}{2}y^2\} exp\{\mu y + \frac{1}{2}\mu^2\}$

Stąd:

$b(y) = \frac{1}{\sqrt{2\pi}} exp\{-\frac{1}{2}y^2\}$, $\eta = \mu$, $T(y) = y$, $a(\eta) = -\frac{1}{2}\mu^2 = -\frac{1}{2}\eta^2$

Zatem $(\mu, 1)$ należy do wykładniczej rodziny rozkładów.

Założenia modelu Uogólnione Modele Liniowe, ang. *Generalized Linear Model* (GLM):

- $y|x;\theta \sim ExpFamily(\eta)$
- mając x, celem jest predykcja $E[T(y)|x]$; dążymy do wyznaczenia $h(x) = E[T(y)|x]$, T - statystyka dostateczna
- $\eta = \theta^T x$, gdy $\eta \in \mathcal{R}$;
  $\eta_i = \theta_i^T x$, gdy $\eta \in \mathcal{R}^k$

Model GLM obejmuje wiele znanych modeli statystycznych, dzięki ogólnej postaci wzoru modelu.

Przykład

Niech $y \in \{1, 2, \ldots, k\}$.
Tym razem nasz problem polega na przydzieleniu zmiennych niezależnych do jednej z k klas, czyli zmienna zależna nadal jest dyskretna, ale może przyjmować jedną z k wartości. Mówimy, że zmienne y podlegają rozkładowi wielorakiemu (ang. *Multinomial*).

Załóżmy, że:
$\phi_1, \ldots, \phi_k$ - parametry rozkładu, $\mathbf{P}(y = i) = \phi_i$, $\phi_k = 1 - (\phi_1 + \ldots + \phi_{k-1})$
Aby wyrazić rozkład wieloraki w języku rodziny rozkładów wykładniczych zdefiniujmy $T(y) \in \mathcal{R}^{k-1}$ w następujący sposób:

$T(y)_i = \mathbb{1}\{y = i\}$ gdzie $T(y)_i$ - i-ty element wektora T(y)

$$\text{Zatem: T(1)} = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}, \text{T(2)} = \begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix}, \ldots T(k-1) = \begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix}, T(k) = \begin{bmatrix}0\\0\\\vdots\\0\end{bmatrix}$$

$$\mathbf{P}(y) = \phi_1^{\mathbb{1}\{y=1\}}\phi_2^{\mathbb{1}\{y=2\}}\ldots\phi_k^{\mathbb{1}\{y=k\}} = \phi_1^{T(y)_1}\phi_2^{T(y)_2}\ldots\phi_{k-1}^{T(y)_{k-1}}\phi_k^{1-\sum_{j=1}^{k-1}T(y)_j} = \ldots$$
$$\ldots = b(y)exp\{\eta^T T(y) - a(\eta)\}$$

$$\text{gdzie } \eta = \begin{bmatrix} \log(\frac{\phi_1}{\phi_k}) \\ \vdots \\ \log(\frac{\phi_1}{\phi_k}) \end{bmatrix} \in \mathcal{R}^{k-1},\ a(\eta) = -\log(\phi_k),\ b(y) = 1$$

Powyżej mamy zdefiniowany obiekt η jako funkcję parametrów ϕ. z definicji tej można wyprowadzić następującą równość:

$$\phi_i = \frac{e^{\eta_i}}{1 + \sum_{j=1}^{k-1} e^{\eta_i}} = \frac{e^{\theta_i^T x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}}$$

Zatem:

$$h_\theta(x) = E[T(y)|x;\theta] = E \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix} = \begin{bmatrix} \frac{e^{\theta_1^T x}}{1+\sum_{j=1}^{k-1} e^{\theta_j^T x}} \\ \vdots \\ \frac{e^{\theta_{k-1}^T x}}{1+\sum_{j=1}^{k-1} e^{\theta_j^T x}} \end{bmatrix}$$

13 Dygresja: regresja wieloraka (ang. *Softmax Regression*)

W dotychczasowych modelach klasyfikacji rozpatrywaliśmy jedynie problem klasyfikacji binarnych. Istnieje jednak możliwość klasyfikacji również w sytuacji, gdy mamy więcej niż dwie klasy. Regresja wieloraka to model GLM stanowiący uogólnienie regresji logistycznej z binarnej klasyfikacji do klasyfikacji k-klasowej.

$$y \in \{1, ..., k\}$$
$$L(\theta) = \prod_{i=1}^{m} \mathbf{P}(y^{(i)}|x^{(i)};\theta) = \prod_{i=1}^{m} \phi_1^{\mathbb{1}\{y^{(i)}=1\}} \phi_2^{\mathbb{1}\{y^{(i)}=2\}} \dots \phi_k^{\mathbb{1}\{y^{(i)}=k\}}$$

Ze względu na naturę predykcji wyróżnia się dwa typy modeli uczenia maszynowego: generatywny (ang. *Generative Learning Models*) i dyskryminacyjny (ang. *Discriminative Learning Models*).

Klasyfikatory dyskryminujące (rys. 17), modelują $p(y|x)$ lub uczą się bezpośredniego mapowania danych wejściowych x na etykiety klas. Dotychczas rozważana regresja logistyczna jest jednym z przykładów modelu dyskrymunującego.

Do tego samego problemu klasyfikacji możemy zastosować również inne podejście. Rozważmy problem klasyfikacji binarnej, w którym mamy dwie klasy,

klasę pozytywną A (y = 1) i klasę negatywną B (y = 0). Tak jak wcześniej, chcemy umieć odróżnić obserwacje należące do tych klas w oparciu o niektóre cechy. Teraz bierzemy wszystkie przykłady etykiety A, próbujemy nauczyć się jego cech i zbudować model dla klasy A. Następnie bierzemy wszystkie przykłady oznaczone etykietą B i próbujemy zbudować osobny model dla klasy B. na koniec, aby sklasyfikować nowy element, porównujemy go do każdego modelu i sprawdzamy, do którego pasuje lepiej. Takie podejście nazywa się generatywnym.

Modele generatywne (rys. 17), opierają się na łącznym prawdopodobieństwie p(x, y) danych wejściowych x i etykiety y, i z wykorzystaniem reguły Bayesa:

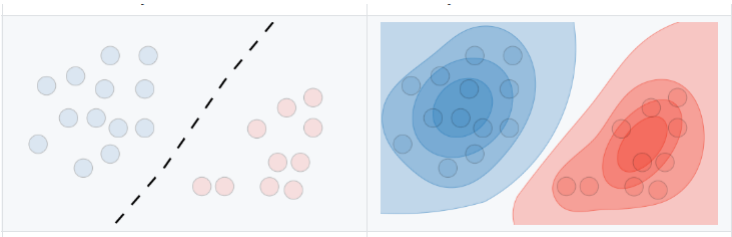
$$p(y = 1|x) = \frac{p(x|y = 1)p(y = 1)}{p(x)}$$

$$p(y = 0|x) = \frac{p(x|y = 0)p(y = 0)}{p(x)}$$

oraz faktu:

$$p(x) = p(y = 0|x)p(x) + p(y = 1|x)p(x)$$

dokonują predykcji $p(y|x)$, a następnie wybiera najbardziej prawdopodobną etykietę y. Model generatywny zakłada, że wszystkie cechy są warunkowo niezależne, podczas gdy model dyskryminacyjny nie zakłada niczego związanego z niezależnością cech.



Rysunek 17: Modele: dyskryminujący (lewy) i generatywny (prawy). Model dyskryminacyjny modeluje granicę decyzyjną między klasami, a generatywny jawnie modeluje faktyczny rozkład każdej klasy

Zasadniczo, decyzja, który klasyfikator należy zastosować, opiera się na rozmiarze zbioru danych i ich niezależności. Jeśli cechy są warunkowo niezależne, stosuje się modele generatywne, w przeciwnym razie dyskryminacyjne.
Ponieważ model generatywny zależy od łącznego prawdopodobieństwa, działa dobrze z mniejszymi zbiorami danych, podczas gdy modele dyskryminacyjne, ze względu na tendencję do overfittingu, cechują się słabą wydajnością dla niewielkich zbiorów.

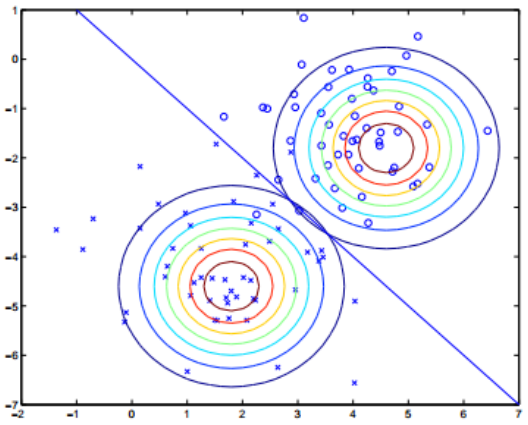
14 Ogólna analiza dyskryminacyjna (ang. *Gaussian Discriminant Analysis*, GLM)

Założenia modelu:

$x|y = 0 \sim \mathcal{N}(\vec{\mu_0}, \Sigma)$
$x|y = 1 \sim \mathcal{N}(\vec{\mu_1}, \Sigma)$
$y \sim Bernoulli(\phi) \Rightarrow p(y) = \phi^y(1-\phi)^{1-y}$

gdzie:
Σ - macierz
$z \sim \mathcal{N}(\vec{\mu}, \Sigma) \Rightarrow p(z) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(x-\vec{\mu})^T\Sigma^{-1}(x-\vec{\mu})\}$
Zatem:
$p(x|y=0) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(x-\vec{\mu_0})^T\Sigma^{-1}(x-\vec{\mu_0})\}$
$p(x|y=1) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} exp\{-\frac{1}{2}(x-\vec{\mu_1})^T\Sigma^{-1}(x-\vec{\mu_1})\}$

W pewnym uproszczeniu metoda polega na dopasowaniu jednego rozkładu Gaussa do obserwacji klasy pozytywnej i drugiego do obserwacji klasy negatywnej. Te dwa rozkłady wyznaczają podział (rys. 18):



Rysunek 18: Przybliżenie rozkładów Gaussa dla dwóch klas, niebieska linia - granica decyzyjna

Otrzymany podział może się różnić od rezultatu przeprowadzenia regresji logistycznej na tym samym zbiorze.

W Ogólna Analiza Dyskryminacyjna, ang. *Gaussian Discriminant Analysis* (GDA) funkcja l wyraża się wzorem:

$$l(\phi, \mu_0, \mu_1, \Sigma) = \log(\prod_{i=1}^{m} p(x^{(i)}, y^{(i)})) = \log(\prod_{i=1}^{m} p(x^{(i)}|y^{(i)})p(y^{(i)}))$$

Szukamy takich parametrów, aby zmaksymalizować l
Wynoszą one odpowiednio:

$$\phi = \frac{\sum_i y^{(i)}}{m} = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 1\}}{m}$$

$$\mu_0 = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 0\}x^{(i)}}{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 1\}x^{(i)}}{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 1\}}$$

Predykcja
Dążymy do wyznaczenia:

$$arg\max_y \mathbf{P}(y|x) = arg\max_y \frac{\mathbf{P}(x|y)\mathbf{P}(y)}{\mathbf{P}(x)} = arg\max_y \mathbf{P}(x|y)\mathbf{P}(y)$$

Uwaga:

jeśli klasy pozytywna i negatywna są równie prawdopodobne, wówczas:

$$arg\max_y \mathbf{P}(y|x) = arg\max_y \frac{\mathbf{P}(x|y)\mathbf{P}(y)}{\mathbf{P}(x)} = arg\max_y \mathbf{P}(x|y)\mathbf{P}(y) = arg\max_y \mathbf{P}(x|y)$$

W ogólności, im mocniejsze założenia nałożymy na dane, tym mniej jest ich potrzebnych GDA do stworzenia dobrego modelu.

Dygresja

Okazuje się, że jeśli

$$\begin{cases} x|y = 0 \sim ExpFamily(\eta_0) \\ x|y = 1 \sim ExpFamily(\eta_1) \end{cases}$$

to $\mathbf{P}(y = 1|x)$ jest funkcją logistyczną

15 Klasyfikator naiwny bajesowski (ang. *Naive Bayes*)

Załóżmy, że mamy klasyfikator y, taki że:

$$y = \begin{cases} 1 & e-mail\ jest\ spamem \\ 0 & w.p.p. \end{cases}$$

oraz wektor x, świadczący o obecności danego słowa w wiadomości:

$$x = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Wartość 1 na i-tej pozycji wektora x oznacza, że w rozważanej wiadomości występuje jest i-te słowo z pewnego słownika.

Dążymy do wyznaczenia $p(x|y)$, wektor $x \in \{0,1\}^n$ a liczba parametrów wynosi $2^n - 1$, gdzie n oznacza długość słownika.

W algorytmie nakłada się silne założenia na $p(x|y)$. Przyjmuje się bowiem, że x_i są warunkowo niezależne pod warunkiem y, tzn. że:

$$\mathbf{P}(x_1, ..., x_n|y) = \mathbf{P}(x_1|y)...\mathbf{P}(x_n|y) = \prod_{i=1}^{m} \mathbf{P}(x_i|y)$$

gdzie $\mathbf{P}(x_i|y) \sim Bernoulli$.

To właśnie z powodu założenia niezależności, algorytm nazywa się naiwnym. Okazuje się jednak, że mimo iż w ogólności jest ono fałszywe, pozwala na otrzymywanie dobrych rezultatów w rozważanym problemie klasyfikacji tekstu.

Parametry modelu:
$\phi_{i|y=1} = \mathbf{P}(x_i = 1|y = 1)$
$\phi_{i|y=0} = \mathbf{P}(x_i = 1|y = 0)$
$\phi_y = \mathbf{P}(y = 1)$

Szukamy:
$arg\max_y \mathbf{P}(y|x) = arg\max_y \mathbf{P}(x|y)\mathbf{P}(y)$

$L(\phi_y, \phi_{i|y=0}, \phi_{i|y=1}) = \prod_{i=1}^{m} \mathbf{P}(x^{(i)}, y^{i)})$

Okazuje się, że optymalne parametry wynoszą:

$$\phi_{i|y=1} = \frac{\sum_{i=1}^{m} \mathbb{1}\{x_j^{(i)} = 1, y^{(i)} = 1\}}{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 1\}}$$

$$\phi_y = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = 1\}}{m}$$

16 Wygładzanie Laplace'a

Klasyfikator naiwny bajesowski może sprawiać problemy w sytuacji, gdy jeden ze składników iloczynu: $\prod_{i=1}^{m} \mathbf{P}(x_i|y)$ jest równy 0. Wówczas pomimo, że m-1 składników iloczynu jest różna od 0, zeruje się całość. w skrajnym przypadku może dojść do sytuacji: $\mathbf{P}(y = 1|x) = \frac{0}{0+0}$. Prawdopodobieństwo wystąpienia tego problemu jest znikome, nie mniej jednak istnieje.

Aby go rozwiązać wprowadzono
następującą metodologię.
Dotychczas obliczenia przeprowadzaliśmy w następujący sposób:

$$P(y = 1) = \frac{\#\text{"1"}}{\#\text{"0"} + \#\text{"1"}},$$

gdzie: $\#\text{"0"}, \#\text{"1"}$ - liczności obserwacji, odpowiednio, negatywnych i pozytywnych.
Dla zabezpieczenia przed $\frac{0}{0+0}$, wprowadzono następującą modyfikację:

$$P(y = 1) = \frac{\#\text{"1"} + 1}{\#\text{"0"} + 1 + \#\text{"1"} + 1}$$

Praktyka określana jest wygładzaniem Laplace'a (ang. *Laplace Smoothing*), a jej uogólnienie dla $y \in \{1, ..., k\}$ ma następującą postać:

$$P(y = j) = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = j\} + 1}{m + k}$$

Zastosowanie wygładzania Laplace'a dla algorytmu Naiwnego Bayesa:

$$\phi_{i|y=1} = \frac{\sum_{i=1}^{m} \mathbb{1}\{x_j^{(i)} = 1, y_j^{(i)} = 1\} + 1}{\sum_{i=1}^{m} \mathbb{1}\{y_j^{(i)} = 1\} + 2}$$

17 Multinomial event model

Dla $x \in \{1, 2, ..., k\}$ przyjmujemy następujący wariant Naiwnego Bayesa:

$$\begin{cases} \mathbf{P}(x|y) \sim Multinomial \\ \mathbf{P}(x|y) = \prod_{i=1}^{m} \mathbf{P}(x_i|y) \end{cases}$$

Wróćmy do przykładu klasyfikacji widomości. E-mail będziemy teraz reprezentować nie jako binarny wektor, a jako wektor cech $(x_1^{(i)}, ..., x_{n_i}^{(i)})$ gdzie n_i - liczba słów w wiadomości, $x_j \in \{1, ..., n\}$ - indeks w słowniku.
Zwróćmy uwagę na fakt, że przy tej reprezentacji, w przeciwieństwie do poprzedniej bierzemy pod uwagę ilość powtórzeń poszczególnych słów w danej wiadomości.
Warto zauważyć, że w tak skonstruowanym modelu prawdopodobieństwa $\mathbf{P}(x_i|y)$ mogą mieć różne rozkłady dla różnych klas y.
Nie mniej jednak:

$P(x, y) = (\prod_{i=1}^{n_i} \mathbf{P}(x_i|y))\mathbf{P}(y)$

$\phi_{k|y=1} = \mathbf{P}(x_j = k|y = 1)$
$\phi_{k|y=0} = \mathbf{P}(x_j = k|y = 0)$
$\phi_u = \mathbf{P}(y)$

$l(\phi_{k|y=1}, \phi_{k|y=0}, \phi_y) = \log(\prod_{j=1}^{n_i} \mathbf{P}(x^{(j)}, y^{(j)}; \phi_{k|y=1}, \phi_{k|y=0}, \phi_y)) =$

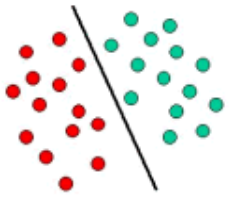
$\log(\prod_{j=1}^{m} \prod_{j=1}^{n_i} \mathbf{P}(x^{(j)}|y^{(j)}; \phi_{k|y=1}, \phi_{k|y=0})\mathbf{P}(y^{(i)}; \phi_y)$

a optymalny parametr $\phi_{k|y=1}$ wyraża się wzorem:

$\phi_{k|y=1} = \frac{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)}=1\} \sum_{i=1}^{n_i} \mathbb{1}\{x_j^{(i)}=k\}+1}{\sum_{i=1}^{m} \mathbb{1}\{y^{(i)}=1\} n_i + n}$

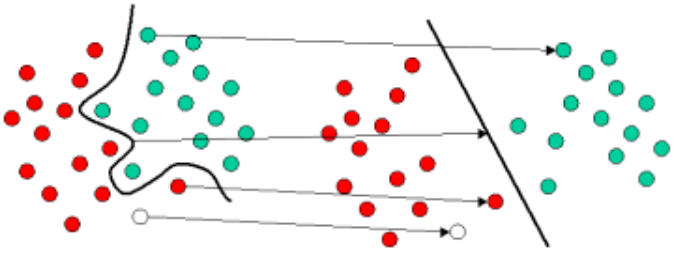
18 Metoda wektorów nośnych (ang. *Support Vector Machine*, SVM)

U podstaw metody wektorów nośnych leży koncepcja przestrzeni decyzyjnej, którą dzieli się budując granice separujące obiekty o różnej przynależności klasowej, czego przykład widzimy na rysunku 19.



Rysunek 19: Ilustracja bardzo prostego przykładu klasyfikatora liniowego, dzielącego obszar prób na dwie części za pomocą prostej. Przedstawione obserwacje są liniowo separowalne

Jest to ilustracja bardzo prostego przykładu klasyfikatora liniowego, dzielącego obszar prób na dwie części za pomocą prostej. Większość praktycznych zadań klasyfikacyjnych jednak nie jest tak oczywista. Do poprawnego klasyfikowania potrzebne są bardziej skomplikowane struktury niż linia prosta. Rysunek 20 ilustruje główną ideę metody wektorów nośnych. Oryginalne obiekty z lewej strony rysunku zostały "zmapowane" (przetransformowane) za pomocą funkcji jądrowych (ang. *Kernels*) na przestrzeń ilustrowaną po prawej. Co ważne, w nowej przestrzeni dwie klasy są liniowo separowalne, co pozwala uniknąć skomplikowanej postaci granicy klas.



Rysunek 20: Rysunek ilustruje główną ideę metody wektorów nośnych. Oryginalne obiekty z lewej strony rysunku zostały "zmapowane" (przetransformowane) za pomocą funkcji jądrowych (ang. *kernels*) na przestrzeń ilustrowaną po prawej.Od tej chwili rozważamy obserwacje liniowo separowalne.

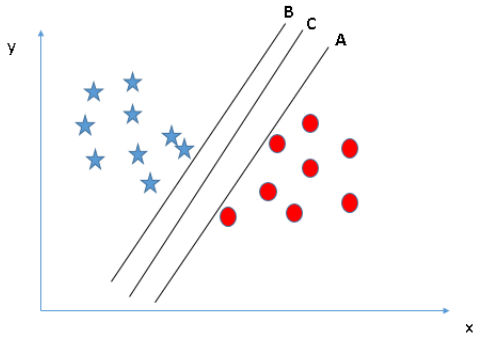
Intuicja marginesu funkcyjnego

Celem regresji logistycznej jest obliczenie $\theta^T x$ i predykcja
"1" $\Leftrightarrow \theta^T x \geqslant 0$
"0" $\Leftrightarrow \theta^T x < 0$

Jeśli $\theta^T x >> 0$ to duża pewność, że y=1,
jeśli $\theta^T x << 0$ to duża pewność, że y=0.

Optymalna sytuacja występuje, gdy:
$\forall_i$ t.że $y^{(i)} = 1$ mamy $\theta^T x^{(i)} >> 0$
$\forall_i$ t.że $y^{(i)} = 0$ mamy $\theta^T x^{(i)} << 0$

Intuicja marginesu geometrycznego



Rysunek 21: Margines geometryczny - na rysunku każda z linii (płaszczyzn) poprawnie odseparowuje klasę pozytywną i negatywną.

Ale która jest najlepsza? Względem jakiej miary określamy, czy dana prosta jest lepsza od drugiej?
Otóż w SVM chcemy zachować jak największą odległość pomiędzy obserwacjami i płaszczyzną odseparowującą. Tym samym, na powyższym rysunku najlepsza spośród prostych A, B i C, jest prosta C.

Notacja

Od tej chwili:
$y \in \{-1, 1\}$,
$h \in \{-1, 1\}$
$\mathrm{g(z)} = \begin{cases} 1 & z \geqslant 0 \\ 0 & w.p.p. \end{cases}$

$h_{w,b}(x) = g(w^T x + b)$ gdzie w $= \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_0 \end{bmatrix}$, b $= \theta_0$

18.1 Margines funkcyjny

Definicja
Margines funkcyjny hiperpłaszczyzny (w,b) dla obserwacji $(x^{(i)}, y^{(i)})$ dany jest wzorem:

$$\hat{\gamma}^{(i)} = y^{(i)}(w^T x^{(i)} + b)$$

Dążymy do zmaksymalizowania marginesu, zatem:
gdy $y^{(i)} = 1$, chcemy aby:

$$w^T x^{(i)} + b >> 0$$

natomiast gdy $y^{(i)} = 0$, aby:

$$w^T x^{(i)} + b << 0$$

Jeśli

$$y^{(i)}(w^T x^{(i)} + b) > 0$$

wówczas obserwacja $(x^{(i)}, y^{(i)})$ jest poprawnie sklasyfikowana.

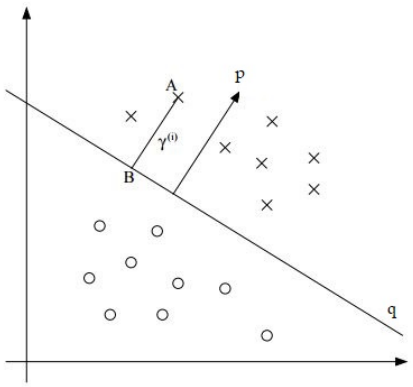
Dla całego zbioru treningowego:

$$\hat{\gamma} = \min_i \hat{\gamma}^{(i)}$$

Uwaga

na margines funkcyjny powinny być narzucone warunki normalizacji, np. $\|w\| = 1$, ponieważ jest on podatny na skalowanie parametrów. Wystarczy pomnożyć parametry przez odpowiednie stałe, aby znacznie zwiększyć margines.

18.2 Margines geometryczny



Rysunek 22: Margines geometryczny

Płaszczyzna p dana jest wzorem:

$$w^T x + b = 0$$

wektor p prostopadły do tej płaszczyzny:

$$p = \frac{w}{\|w\|}$$

a $\gamma^{(i)}$ oznacza odległość obserwacji $(x^{(i)}, y^{(i)})$ od płaszczyzny q

Zauważmy, że współrzędne obserwacji $(x^{(i)}, y^{(i)})$ odległej o $\gamma^{(i)}$ od płaszczyzny q muszą spełniać równość:

$$w^T(x^{(i)} - \gamma^{(i)}\frac{w}{\|w\|}) + b = 0$$

Zauważmy, że

$$w^T x^{(i)} + b = \gamma^{(i)}\frac{w^T w}{\|w\|} = \gamma^{(i)}\|w\|$$

Stąd:

$$\gamma^{(i)} = (\frac{w}{\|w\|})^T x^{(i)} + \frac{b}{\|w\|}$$

Ogólna postać marginesu geometrycznego:

$$\gamma^{(i)} = y^{(i)}[\frac{w^T}{\|w\|}x^{(i)} + \frac{b}{\|w\|}]$$

Z powyższego wzoru wynika, że pomiędzy marginesami: funkcyjnym i geometrycznym jest duże podobieństwo. Różnica polega na normalizacji $\|w\|$. Ponownie chcemy, aby ten margines był jak największy.

Fakty:

1. $\|w\| = 1 \Rightarrow \hat{\gamma}^{(i)} = \gamma^{(i)}$

2. $\gamma^{(i)} = \frac{1}{\|w\|}\hat{\gamma}^{(i)}$

Analogicznie jak w przypadku marginesu funkcyjnego, margines geometryczny dla całego zbioru uczącego definiujemy następująco:

$$\gamma = \min_i \gamma^{(i)}$$

19 Klasyfikator maximum margin (ang. *Max Margin Classifier*)

Algorytm uczenia wyznaczający parametry w i b tak, aby zmaksymalizować margines geometryczny. Innymi słowy wyznacza:

$$\max_{\gamma,w,b} \gamma \tag{2}$$

tak, aby:

$$\begin{cases} y^{(i)}(w^T x^{(i)} + b) \geqslant \gamma \\ \|w\| = 1 \end{cases}$$

Jest to pierwsza z rozważanych dzisiaj formuł problemu optymalizacji.

Margines geometryczny nie jest podatny na skalowanie parametrów w i b, i nie zmienia się pod wpływem $\|w\|$. Zatem w celu ułatwienia obliczeń możemy manipulować wartościami w i b. w szczególności można przyjąć $\|w\| = 1$ lub $|w| = 1$.

20 Problemy optymalizacji

W formule (2.) występuje ograniczenie na miarę wektora w. Organiczenie do zostało określone jako bezkontekstowe i kolejne formuły tego samego problemu optymalizacji zorientowane są na jego eliminację.

Alternatywna formuła problemu optymalizacji:

$$\max_{\hat{\gamma},w,b} \frac{\hat{\gamma}}{\|w\|} \tag{3}$$

tak, aby:

$$y^{(i)}(w^T x^{(i)} + b) \geqslant \hat{\gamma}$$

Ponieważ $\frac{\hat{\gamma}}{\|w\|} = \gamma$, to jasne jest, że jest to inne ujęcie tego samego problemu optymalizacji, z tym, że nie ma w nim kłopotliwego ograniczenia na normę wektora w.

20.1 Ograniczenie skalowania

Załóżmy, że na $\hat{\gamma}$ chcemy nałożyć ograniczenie

$$\hat{\gamma} = 1$$

czyli

$$\min_i y^{(i)}(w^T x^{(i)} + b) = 1$$

Warunek ten jest łatwy do zrealizowania poprzez odpowiednie skalowanie parametrów w i b. Okazuje się, że dodanie powyższego założenia do problemu optymalizacji (3.) prowadzi nas do sformułowania:

$$\max_{w,b} \frac{1}{\|w\|^2} = \min_{w,b} \|w\|^2 \tag{4}$$

tak, że:

$$y^{(i)}(w^T x^{(i)} + b) \geqslant 1$$

Jest to złożony problem optymalizacji, w którym stosuje się metodę gradient descent.

21 Dygresja: Mnożnik Lagrange'a

Metoda obliczania ekstremum funkcji, wykorzystywana w teorii optymalizacji. Pozwala na znalezienie estremów warunkowych funkcji $f : \mathcal{R}^n \to \mathcal{R}$

pod warunkiem zerowania funkcji $h : \mathcal{R}^n \to \mathcal{R}$.

Innymi słowy szukamy: $\min_\omega f(\omega)$ takiego że: $h(\omega) = \begin{bmatrix} \mathrm{h}_1(\omega) \\ \mathrm{h}_2(\omega) \\ \vdots \\ \mathrm{h}_n(\omega) \end{bmatrix} = \vec{0}$

Metoda Lagrange'a

$$\mathcal{L}(\omega, \beta) = f(\omega) + \sum_i \beta_i h_i(\omega)$$

gdzie β_i - mnożnik Lagrange'a

Dla każdego ω^* będącego rozwiązaniem, musi istnieć β^* t.że:

$$L(\omega^*, \beta^*)\partial\omega = 0$$
$$L(\omega^*, \beta^*)\partial\beta = 0$$

Zatem, aby rozwiązać powyższy problem należy zatem ustalić, że:
$L\partial\omega = 0$, $L\partial\beta = 0$ i w efekcie jesteśmy w stanie wyznaczyć rozwiązanie. Szukamy:

$$\min_\omega f(\omega)$$

takiego że:
$g_i(\omega) \leqslant 0$, i = 1,...,k $\quad (g(\omega) \leqslant \vec{0})$
$h_i(\omega) = 0$, i = 1,...,l $\quad (h(\omega) = \vec{0})$

Metoda Lagrange'a:

$$\mathcal{L}(\omega, \alpha, \beta) = f(\omega) + \sum_{i=1}^{k} \alpha_i g_i(\omega) + \sum_{i=1}^{l} \beta_i h_i(\omega)$$

21.1 Primal problem

Zdefiniujmy:

$$\theta_p(\omega) = \max_{\substack{\alpha,\beta \\ \alpha_i \geqslant 0}} \mathcal{L}(\omega, \alpha, \beta)$$

Rozważmy:

$$p^* = \min_\omega \max_{\substack{\alpha,\beta \\ \alpha_i \geqslant 0}} \mathcal{L}(\omega, \alpha, \beta) = \min_\omega \theta_p(\omega)$$

$\theta_p(\omega)$:

- jeśli $g_i(\omega) \geqslant 0$, to $\theta_p(\omega) = \infty$
- jeśli $h_i(\omega) \neq 0$, to $\theta_p(\omega) = \infty$
- w przeciwnych przypadkach $\theta_p(\omega) = f(\omega)$

Zatem:

$$\theta_p(\omega) = \begin{cases} f(\omega) & gdy\ g_i(\omega) \leqslant 0\ i\ \ h_i(\omega) = 0 \\ \infty & w.p.p. \end{cases}$$

Stąd oryginalny problem wyraża się następująco:

$$\min_{\omega} \theta_p(\omega)$$

21.2 Dual problem

Zdefiniujmy:

$$\theta_{\mathcal{D}}(\alpha, \beta) = \min_{\omega} \mathcal{L}(\omega, \alpha, \beta)$$

Rozważmy:

$$d^* = \max_{\substack{\alpha \geqslant 0 \\ \beta}} \min_{\omega} \mathcal{L}(\omega, \alpha, \beta) = \max_{\substack{\alpha \geqslant 0 \\ \beta}} \theta_p(\omega)$$

Różnica polega na zamianie kolejności max i min.
Z tej zamiany właśnie wynika fakt nierówności :

$$d^* \leqslant p^*$$

bo $\max\min(...) \leqslant \min\max(...)$
Częściej rozważanym problemem jest dual problem.

22 Optymalizacja SVM

Niech f będzie funkcją wypukłą (hesjan $H \geqslant 0$).

Załóżmy, że: $h_i(\omega) = a_i^T \omega + b_i$ oraz $\exists_\omega$ t.że $\forall_i g_i(\omega) < 0$

Zatem istnieją $\omega^*, \alpha^*, \beta^*$ będące rozwiązaniami problemów optymalizacji oraz $p^* = d^* = \mathcal{L}(\omega^*, \alpha^*, \beta^*)$.

Stosujemy metodę mnożników Lagrange'a, zatem:

$$\frac{\partial}{\partial \omega} \mathcal{L}(\omega^*, \alpha^*, \beta^*) = 0$$

$$\frac{\partial}{\partial \beta} \mathcal{L}(\omega^*, \alpha^*, \beta^*) = 0$$

oraz spełnione są warunki Karusha-Kuhna-Tuckera (KKT), tj.:

$$\alpha^* g_i(\omega) = 0$$
$$g_i(\omega^*) \leqslant 0$$
$$\alpha^* \geqslant 0$$

Okazuje się, że $\alpha \neq 0$ tylko dla wektorów nośnych (obserwacji o marginesie funkcyjnym równym 1)
Z KKT wynika, że:

$$\alpha_i \geqslant 0 \Rightarrow g_i(\omega^*) = 0$$

czyli $g_i(\omega)$ jest aktywnym ograniczeniem.

W praktyce najczęściej okazuje się, że

$$\alpha_i^* \neq 0 \Leftrightarrow g_i(\omega^*) = 0$$

Notacja:
mnożniki $\alpha_i, \beta_i \rightarrow \alpha_i$
dla ułatwienia obliczeń zamiast $\min \|w\|^2$ będziemy rozważać $\min \frac{1}{2}\|w\|^2$ Szukamy $\min \frac{1}{2}\|w\|^2$ t.że: $y^{(i)}(w^T x^{(i)} + b) \geqslant 1$
Niech

$$g(\omega, b) = y^{(i)}(w^T x^{(i)} + b) - 1$$

wówczas

$$y^{(i)}(w^T x^{(i)} + b) \geqslant 1 \Leftrightarrow g(\omega, b) \leqslant 0$$

$\mathcal{L}(\omega, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^m \alpha_i(y^i(w^T x^{(i)} + b) - 1)$

Zdefiniujmy: $\theta_{\mathcal{D}}(\alpha) = \min_{w,b} \mathcal{L}(\alpha, b, \omega)$

$$\begin{cases} \nabla_w \mathcal{L}(\alpha, b, \omega) = \omega - \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} = 0 \Rightarrow w \ = \sum_{i=1}^m \alpha_i y^{(i)} x^{(i)} \\ \frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^m y^{(i)} \alpha_i = 0 \end{cases}$$

Stąd:

$\mathcal{L}(\omega, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^m \alpha_i(y^i(w^T x^{(i)} + b) - 1) =$

$\frac{1}{2}(\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)})^T (\sum_{i=1}^m \alpha_i y^{(i)} x^{(i)}) - \sum_{i=1}^m \alpha_i(y^i(w^T x^{(i)} + b) - 1) =$

$\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - \sum_{i=1}^m \sum_{j=1}^m y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} + \sum_{i=1}^m \alpha_i =$

$-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y^{(i)}y^{(j)}\alpha_i\alpha_j(x^{(i)})^T x^{(j)} + \sum_{i=1}^{m}\alpha_i := W(\alpha)$

$$\theta_{\mathcal{D}}(\alpha) = \begin{cases} W(\alpha) & gdy\ \sum_i y_i\alpha_i = 0 \\ -\infty & gdy\ \sum_i y_i\alpha_i \neq 0 \end{cases}$$

Chcemy wyznaczyć

$$\max_{\alpha \geqslant 0} W(\alpha)$$

t.że $\alpha_i \geqslant 0$, $\sum_i y_i\alpha_i = 0$

Równania do wyznaczenia α i b:

$$\begin{cases} w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \\ b = \frac{1}{2}(\max_{i:y^(i)=-1} w^T x^{(i)} + \min_{i:y^(i)=1} w^T x^{(i)}) \end{cases}$$

Ten sam algorytm w języku iloczynów skalarnych:

$$\begin{cases} w \ = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} & \leftarrow input \\ h_{w,b}(x) = g(w^T x + b) & \leftarrow hipoteza \end{cases}$$

Należy wyznaczyć $w^T x + b$.
Okazuje się, że $w^T x + b = \sum_{i=1}^{m} \alpha_i y^{(i)} (x^{(i)})^T x + b$
Powyższy algorytm pozwala na rozpatrywanie wielowymiarowych x
i prowadzenie dla nich efektywnych obliczeń.

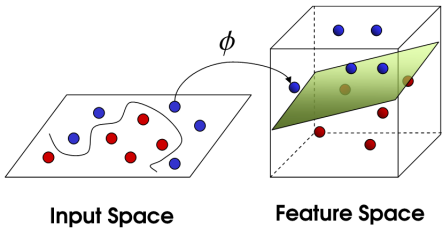
23 Jądra (ang. *Kernels*)

Załóżmy, że mamy pewną cechę $x \in \mathcal{R}$ np. metraż mieszkania, rozważany w pierwszym wykładzie. Często stosowaną praktyką jest poddanie oryginalnej cechy ciągłej, mapowaniu do zbioru zmiennych o wyższych wymiarach na przykład w następujący sposób:

$$x \xrightarrow{\phi} \begin{bmatrix} x \\ x^2 \\ x^3 \\ x^4 \end{bmatrix} = \phi(x)$$

Funkja ϕ przeprowadza przestrzeń wektorów wejściowych do nowej przestrzeni, o większej liczbie wymiarów (rys. 23), w której dane będą separowalne.

Wykorzystanie takiej transformacji w algorytmie jest bardzo proste.
Wystarczy zastąpić iloczyn skalarny $< x^{(i)}, x^{(j)} >$ przez $< \phi(x^{(i)}), \phi(x^{(j)}) >$, gdzie $< x, y >= x^T y$



Rysunek 23: Obiekty z lewej strony (ang. *Input Space*) rysunku zostały przetransformowane za pomocą funkcji jądrowych (ang. *kernels*) na przestrzeń ilustrowaną poprawej (ang. *Feature Space*).

Intuicja

Niech $x \to \phi(x)$, $z \to \phi(z)$. Wówczas, jeżeli x i z są 'podobne', wówczas wektory $\phi(x)$, $\phi(z)$ będą wskazywać ten sam kierunek i ich iloczyn $< \phi(x), \phi(z) >$ będzie duży, z drugiej strony jeżeli x i z znacznie się różnią, wówczas iloczyn $< \phi(x), \phi(z) >$ będzie mały. Zdarza się, że $\phi(x)$ ma wiele, a nawet nieskończenie wiele wymiarów. w tych szczególnych przypadkach, obliczenie powyższego iloczynu jest problematyczne, a zdarza się, że wręcz niemożliwe ze względu na ograniczoność mocy obliczeniowej. Okazuje się jednak, że nawet w tego typu sytuacji jesteśmy w stanie, niewielkim kosztem obliczyć

$$\mathcal{K}(x^{(i)}, x^{(j)}) =< \phi(x^{(i)}), \phi(x^{(j)}) >$$

Wówczas realizacja SVM jest analogiczna do zasygnalizowanej powyżej i polega na zastąpieniu iloczynu skalarnego $< x^{(i)}, x^{(j)} >$ przez $\mathcal{K}(x^{(i)}, x^{(j)})$.

Przykład

Załóżmy, że mamy dwie cechy: x = $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, które chcemy przetransformować do następującej przestrzeni czterowymiarowej: $(x_1^2, x_1x_2, x_2x_1, x_2^2)$.
Procedura polegająca na zmapowaniu obserwacji do nowej przestrzeni $\mathcal{R}^2 \to \mathcal{R}^4$ oraz obliczeniu iloczynu $\phi(x^{(i)})\phi(x^{(j)})$, gdzie $\phi(x^{(i)}) = \begin{bmatrix} (x_1^{(i)})^2 \\ x_1^{(i)}x_2^{(i)} \\ x_2^{(i)}x_1^{(i)} \\ (x_2^{(i)})^2 \end{bmatrix}$, ma złożoność rzędu $\mathcal{O}(n^2)$. Weźmy na przykład: $x^{(i)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $x^{(j)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}$, wówczas zgodnie z powyższą metodologią: $\begin{bmatrix} 1 \\ 2 \\ 2 \\ 4 \end{bmatrix} \cdot \begin{bmatrix} 9 \\ 15 \\ 15 \\ 25 \end{bmatrix} = 9 + 30 + 30 + 100 = 169.$

Zauważmy, że:
$\mathcal{K}(x^{(i)}, x^{(j)}) = ((x^{(i)})^T x^{(j)})^2 = (3+10)^2 \ = \ 169$
A zatem obliczenie tego samego iloczynu z wykorzystaniem jądra ma koszt $\mathcal{O}(n)$

Wniosek
Do pracy w nowej przestrzeni nie jest wymagana znajomość funkcji ϕ,
znajomość jądra jest wystarczająca.

Twierdzenie
Niech $K \in \mathcal{R}^{m \times m}$ będzie macierzą jądra t.że

$$K_{i,j} = \mathcal{K}(x^{(i)}, x^{(j)}) = (\phi(x^{(i)}))^T \phi(x^{(j)})$$

oraz niech $\mathcal{K}(x, z)$ będzie dane.
$\mathcal{K}$ jest jądrem (tzn. $\exists \phi$ t.że $\mathcal{K}(x, z) =< \phi(x), \phi(z) >$) wtedy i tylko wtedy, gdy dla każdego zbioru $\{x^{(1)}, ..., x^{(m)}\}$, (m$< \infty$) macierz $K \in \mathcal{R}^{m \times m}$
jest **symetryczna i dodatnio określona**.

24 Zmienne osłabiające (ang. *Soft Margin*)

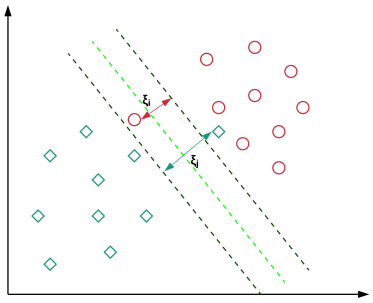
W dotychczas rozważanym algorytmie SVM naszym celem było utrzymanie poprawności klasyfikacji, przy zachowaniu jak najszerszych marginesów. Innymi słowy:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

tak, że:

$$y^{(i)}(w^T x^{(i)} + b) \geqslant 1$$

Idea zmiennych objaśniających opiera się na prostej przesłance: pozwól, aby SVM popełnił pewną liczbę błędów i utrzymuj jak najszerszy margines (rys. 24), tak, aby inne punkty mogły nadal być poprawnie klasyfikowane.



Rysunek 24: Dla każdego punktu danych x_i wprowadzamy zmienną ξ_i. Jeśli x_i znajduje się po niewłaściwej stronie marginesu, to ξ_i jest tożsama z odległością x_i od marginesu odpowiedniej klasy, w przeciwnym razie zero. W ten sposób punkty znajdujące się daleko od marginesu po niewłaściwej stronie generują większą karę.

Pomysł ten można wdrożyć w prosty sposób modyfikując cel SVM.

Mianowicie:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i$$

tak, że:

$$y^{(i)}(w^T x + b) \geqslant 1 - \xi_i$$

gdzie $\xi_i \geqslant 0$

$$\mathcal{L}(w,b,\xi,\alpha,r) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i(y^{(i)}(\omega^T x^{(i)} + b) - 1 + \xi_i) - \sum_{i=1}^{n} r_i\xi_i$$

Zmienne $\xi_i \geqslant 0$ (ang. *Soft Margin*) dobiera się dla każdego przykładu uczącego. Jej wartość zmniejsza margines separacji.

Jeżeli $0 \leqslant \xi_i \leqslant 1$, to punkt danych $x^{(i)}$ leży wewnątrz strefy separacji, ale po właściwej stronie.

Jeżeli $\xi_i > 1$, punkt $x^{(i)}$ leży po niewłaściwej stronie hiperpłaszczyzny i wystąpi błąd klasyfikacji.

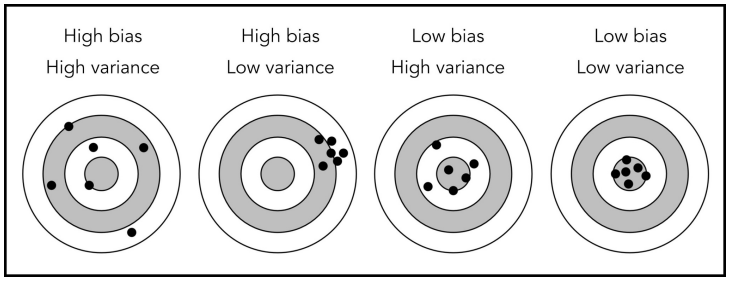
Takie podejście do SVM jest powszechnie stosowane, ponieważ pozwala na uniezależnienie podziału (położenia hiperpłaszczyzny) od pojedynczych obserwacji.

25 Błędy obciążenia i wariancji

Błąd nieredukowalny: wynika z zaszumienia samych danych. Jedynym sposobem na jego zmniejszenie jest ich oczyszczenie.

Błąd obciążenia (ang. *Bias*): tendencja do systematycznego uczenia się z tych samych błędnych rzeczy. Wynika z niewłaściwych założeń, takich jak założenie, że dane są 'liniowe', gdy w rzeczywistości są 'kwadratowe'. O modelach niedouczonych mówi się, że mają wysoki błąd obciążenia (rys. 25).

Błąd wariancji (ang. *Variance*): tendencja do uczenia się losowych rzeczy bez względu na realny trend. Wynika z nadmiernej wrażliwości modelu na małe różnice w danych trenujących. Wysokim błędem wariancji charakteryzują się modele przeuczone (rys. 25).



Rysunek 25: Niskie błędy obciążenia i wariancja powoduji, że model przewiduje wartości bardzo zbliżone do celu. Wysoki błąd wariancji i niski błąd obciążenia powoduje dużą zmienność prognoz. Niski błąd wariancji i wysoki błąd obciążenia oznaczają, że wyniki są bardziej skoncentrowane, jednak dalekie od celu.

Z dobrym modelem mamy do czynienia wówczas, gdy zarówno obciążenie, jak i wariancja są niskie (rys. 25). Zwiększenie złożoności modelu zazwyczaj zwiększa jego wariancję i zmniejsza jego obciążenie. Zmniejszenie złożoności modelu zwiększa jego obciążenie i zmniejsza jego wariancję.

Celem wykładu jest zrozumienie problemu niedouczenia i przeuczenia modelu, w terminach błędu obciążenia i błędu wariancji.

26 Problem klasyfikacji liniowej

Niech:
$h_\theta(x) = g(\theta^T x)$,
$g(z) = \mathbf{1}\{z \geqslant 0\}$ (zauważmy, że $y \in \{0, 1\}$),
$\mathcal{S} = \{(x^{(i)}, y^{(i)})\}$ - zbiór treningowy, z dodatkowym założeniem, że:
$(x^{(i)}, y^{(i)}) \sim_{i.i.d.} \mathcal{D}$, gdzie $\mathcal{D}$ jest oznaczeniem pewnego rozkładu.
Zdefiniujmy dodatkowo błąd treningowy hipotezy h_θ (ryzyko) jako:
$\hat{\varepsilon}(h_\theta) = \hat{\varepsilon}_{\mathcal{S}}(h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{h_\theta(x^{(i)}) \neq y^{(i)}\}$
Uproszczony algorytm, będący przedmiotem dalszych rozważań, określany jest jako **ERM** (ang. *Empirical Risk Minimization*) i parametryzowany przez:

$$\hat{\theta} = arg \min_{\theta} \hat{\varepsilon}_{\mathcal{S}}(h_\theta)$$

Dalsze rozważania będą zorientowane na analizę właśnie tego, podstawowego, algorytmu, ponieważ okazuje się, że regresja logistyczna oraz SVM, formalnie mogą być uznawane jako jego przybliżenia.
Informacją przydatną na przestrzeni wykładu jest fakt, że o algorytmach ML, często dogodniej jest myśleć nie jak o wyborze zbioru parametrów, ale jak o doborze funkcji. Jako funkcję postrzegamy wówczas h_θ należącą do klasy hipotez $\mathcal{H} = \{h_\theta : \theta \in \mathcal{R}^{n+1}\}$, taką, że wraz z modyfikacją θ, h_θ staje się inną funkcją oraz $h_\theta : X \to \{0, 1\}$.
Stosując takie podejście należy również zredefiniować **ERM**:

$$ERM: \ \hat{h} = arg \min_{h \in \mathcal{H}} \hat{\varepsilon}_{\mathcal{S}}(h)$$

Należy zwrócić uwagę na fakt, że pomimo występowania w definicji ERM błędu treningowego, to nie jego minimalizacja jest ostatecznym celem procesu uczenia. Finalnie dążymy bowiem do tego, aby model potrafił dokonywać trafnych predykcji dla danych, których 'wcześniej nie widział', czyli aby miał zdolność **generalizacji**. Zdefiniujmy więc wielkość, tożsamą z **błędem generalizacji**:

$$\varepsilon(h) = \mathbf{P}_{(x,y) \sim_{i.i.d} \mathcal{D}}(h(x) \neq y)$$

Lemat 1.
Niech A_1, A_2,...,A_k oznacza k, niekoniecznie niezależnych, zdarzeń.

Wówczas $\mathbf{P}(A_1 \cup A_2 \cup ... \cup A_k) \leqslant \mathbf{P}(A_1) + \mathbf{P}(A_2) + ... + \mathbf{P}(A_k)$

Lemat2.
Niech $Z_1, Z_2, ..., Z_m \sim_{i.i.d.} Bernoulli(\phi) \rightarrow \mathbf{P}(Z_i = 1) = \phi$.
Niech $\hat{\phi} = \frac{1}{m} \sum_{i=1}^{m} Z_i$ i niech dowolne $\gamma > 0$ będzie ustalone.
Wówczas:

$$\mathbf{P}(|\hat{\phi} - \phi| > \gamma) \leqslant 2exp\{-2\gamma^2 m\}$$

26.1 Przypadek skończonego $\mathcal{H}$

$$\mathcal{H} = \{h_1, h_2, ..., h_k\}$$

$$\hat{h} = arg \min_{h_i \in \mathcal{H}} \varepsilon_{\mathcal{S}}\hat{(}h_i)$$

Metodologia:
(1.) Dowód, że błąd treningowy jest dobrym przybliżeniem błędu generalizacji, tj. $\hat{\varepsilon} \approx \varepsilon$.
(2.) Dowód, że z powyższego stwierdzenia wynika ograniczenie na $\hat{\varepsilon}(h)$.

Uwaga: Punkt (1.) implikuje, że minimalizacja błędu treningowego, jest tożsama z minimalizacją błędu generalizacji.

Dowód
Ustalmy dowolne $h_j \in \mathcal{H}$. Zdefiniujmy $Z_i := \mathbf{1}\{h_j(x^{(i)}) \neq y^{(i)}\} \in \{0, 1\}$.

Zauważmy, że:

- $Z_i \sim Bernoulli$
- $\mathbf{P}(Z_i = 1) = \varepsilon(h_j)$
- $\hat{\varepsilon}(h_j) = \frac{1}{m} \sum_{i=1}^{m} Z_i$

Z lematu 2. wynika, że:

$$\mathbf{P}(|\hat{\varepsilon}(h_j) - \varepsilon(h_j)| > \gamma) \leqslant 2exp\{-2\gamma^2 m\}$$

Zatem dla dostatecznie dużego m, dla hipotezy h_j zachodzi: $\hat{\varepsilon} \approx \varepsilon$.
Zdefiniujmy A_j jako zdarzenie: $|\hat{\varepsilon}(h_j) - \varepsilon(h_j)| > \gamma$. Wówczas z lematu 1. oraz z powyższego dowodu, że: $P(A_j) \leqslant 2exp\{-2\gamma^2 m\}$ wynika, że:

$$\mathbf{P}(\exists_{h_j \in \mathcal{H}} |\hat{\varepsilon}(h_j) - \varepsilon(h_j)| > \gamma) = P(A_1 \cup A_2 \cup ... \cup A_k) \leqslant$$

$$\leqslant \sum_{i=1}^{k} \mathbf{P}(A_i) \leqslant \sum_{i=1}^{k} 2exp\{-2\gamma^2 m\} = 2kexp\{-2\gamma^2 m\}$$

Stąd:

$$\mathbf{P}(\sim (\exists_{h_j \in \mathcal{H}} |\hat{\varepsilon}(h_j) - \varepsilon(h_j)| > \gamma)) = \mathbf{P}(\forall_{h_j \in \mathcal{H}} |\hat{\varepsilon}(h_j) - \varepsilon(h_j)| \leqslant \gamma) \geqslant 1 - 2kexp\{-2\gamma^2 m\}$$

Zatem dla każdego $h \in \mathcal{H}$ i dostatecznie dużego m, $\hat{\varepsilon}(h)$ znajduje się "w odległości" co najwyżej γ od $\varepsilon(h)$, z prawdopodobieństwem $1-2kexp\{-2\gamma^2 m\}$. (zbieżność jednostajna/uniform convergence).

Na potrzeby powyższego dowodu, jako dane przyjęliśmy wielkości m i γ, natomiast szukaną była wartość prawdopodobieństwa (oznaczmy ją przez δ).

Przeformułowując problem w taki sposób, że jako szukaną przyjmujemy m, a jako dane γ i δ i wyznaczając wielkość zbioru treningowego z równania $\delta = 2kexp\{-2\gamma^2 m\}$ $(*)$ można otrzymać następujący wniosek:
jeśli $m \geqslant \frac{1}{2\gamma^2} \log \frac{2k}{\delta}$, to dla każdego $h \in \mathcal{H}$ z prawdopodobieństwem δ zachodzi nierówność: $|\hat{\varepsilon}(h_j) - \varepsilon(h_j)| \leqslant \gamma$. (ang. *"Sample Complexity" Bound*)

Okazuje się, że ustalenie wartości m i δ, a za niewiadomą przyjmując γ, z równania $(*)$ wynika, że: $|\hat{\varepsilon}(h_j) - \varepsilon(h_j)| \leqslant \underbrace{\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}}_{\gamma}$. (ang. *"Error Bound"*)

Zauważmy, że jeśli:
(1.) $\forall_{h_j \in \mathcal{H}} |\hat{\varepsilon}(h_j) - \varepsilon(h_j)| \leqslant \gamma$
(2.) $\hat{h} = arg\min_{h \in \mathcal{H}} \hat{\varepsilon}(h)$
(3.) $h^* = arg\min_{h \in \mathcal{H}} \varepsilon(h)$
to:

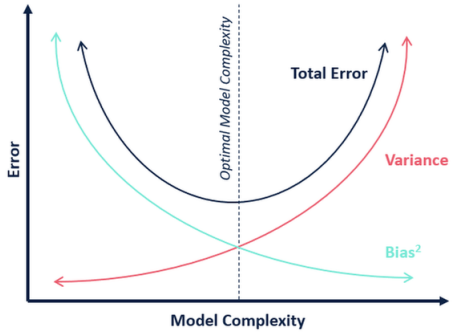
$$\varepsilon(\hat{h}) \leqslant^{(1)} \hat{\varepsilon}(\hat{h}) + \gamma \leqslant^{(2)} \hat{\varepsilon}(h^*) + \gamma \leqslant^{(1.)} \hat{\varepsilon}(h^*) + 2\gamma$$

Twierdzenie
Niech $|\mathcal{H}| = k$ oraz niech m i δ będą ustalone. Wówczas z prwdopodobieństwem $1 - \delta$ zachodzi nierówność:

$$\varepsilon(\hat{h}) \leqslant \min_{h \in \mathcal{H}} \varepsilon(h) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

Rozważmy dwie klasy: klasę funkcji liniowych i klasę funkcji kwadratowych, odpowiednio $\mathcal{H}$, $\mathcal{H}'$. Wówczas faktem jest, że $\mathcal{H} \subseteq \mathcal{H}'$. Im bardziej złożoną klasę hipotez rozważamy, tym lepszą wartość, w sensie błędu generalizacji , przyjmie czynnik $\min_{h \in \mathcal{H}} \varepsilon(h)$. Jednocześnie, należy mieć na uwadze, że im bardziej złożona klasa hipotez tym większe k i tym samym $2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$. na podstawie tej intuicji można stwierdzić, że wyrażenie nierówności z powyższego twierdzenia w terminach błędów obciążenia i wariancji ma następującą postać (rys. 26):

$$\varepsilon(\hat{h}) \leqslant \underbrace{\min_{h \in \mathcal{H}} \varepsilon(h)}_{"bias"} + \underbrace{2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}}_{"variance"}$$



Rysunek 26: Krzywe pokazują, że zwiększając złożoność modelu, zmniejszymy błąd obciążenia, ale błąd wariancji wzrośnie, w wyniku czego całkowita strata będzie wysoka. Nie powinno się przyjmować zbyt prostego modelu, ze względu na wysoki błąd obciążenia i nie można też wziąć zbyt dużego, ponieważ charakteryzować się będzie wysokim błędem wariancji.

Twierdzenie
Niech $|\mathcal{H}| = k$ oraz niech γ i δ będą ustalone.
Wówczas dla $\varepsilon(\hat{h}) \leqslant \min_{h \in \mathcal{H}} \varepsilon(h) + 2\gamma$ z prawdopodobieństwem $1 - \delta$ zachodzi:

$$m \geqslant \frac{1}{2\gamma^2} \log \frac{2k}{\delta} = O(\frac{1}{\gamma^2} \log \frac{k}{\delta})$$

Definicja
Rozbicie zbioru dla niepustego zbioru $\mathcal{A}$ to taka rodzina Π niepustych podzbiorów tego zbioru, że każdy element zbioru $\mathcal{A}$ należy do dokładnie jednego podzbioru tej rodziny.

Definicja
Zbiór przykładów $\mathcal{X}$ (o n elementach) jest rozbijany przez przestrzeń hipotez $\mathcal{H}$, jeśli dla dowolnej (z 2^n możliwych) klasyfikacji elementów zbioru istnieje hipoteza $h \in \mathcal{H}$ dla której $(h(x_1), \ldots, h(x_n))$ jest taką klasyfikacją.
Wymiar Wapnika-Czerwonienkisa (ozn. $VC(\mathcal{X})$) jest zdefiniowany jako liczność największego zbioru, który jest rozbijany przez przestrzeń hipotez.

Twierdzenie
Niech $\mathcal{H}$ będzie ustalone i niech $VC(\mathcal{X}) = d$. Wówczas dla każdego $h \in \mathcal{H}$, z prawdopodobieństwem $1 - \delta$, zachodzi nierówność:

$$|\hat{\varepsilon}(h_j) - \varepsilon(h_j)| \leqslant O(\sqrt{\frac{d}{m} \log \frac{m}{d} + \frac{1}{m} \log \frac{1}{\delta}})$$

Stąd mamy, że również w prawdopodobieństwem $1 - \delta$, zachodzi:

$$\varepsilon(\hat{h}) \leqslant \varepsilon(h^*) + O(\sqrt{\frac{d}{m} \log \frac{m}{d} + \frac{1}{m} \log \frac{1}{\delta}})$$

Wniosek
Aby nierówność: $\varepsilon(\hat{h}) \leqslant \varepsilon(h^*) + 2\gamma$ zachodziła z prawdopodobieństwem $1 - \delta$, powinna być spełniona równość: $m = O_{\gamma,\delta}(d)$.
Zatem w przypadku algorytmów, minimalizujących błąd treningowy, zależność pomiędzy liczbą obserwacji w zbiorze uczącym i liczbą wymiarów klasy hipotez powinna być liniowa.

27 Wybór modelu

Oznaczy zbiór modeli przez: $\mathcal{M} = \{M_1, M_2, M_3, ...\}$.
Spsobem wyboru optymalnego modelu, może być przeprowadzenie kroswalidacji dla każdego z nich i wybór tego, o najmniejszej wartości błędu predykcji na zbiorze walidacyjnym.

K-krotna walidacja krzyżowa

Rysunek 27: K-krotna walidacja krzyżowa (ang. *K-Fold Cross-Validation*) dokonuje losowego podziału zbioru danych na k podzbiorów, gdzie k-1 podzbiorów wykorzystywanych jest do trenowania modelu .
Proces powtarzany jest k-krotnie, a w efekcie otrzymujemy k modeli i oszacowań skuteczności. Ostateczny wynik oszacowania uzyskiwany jest poprzez uśrednienie wyników uzyskanych ze wszystkich iteracji.

K-krotna walidacja krzyżowa dokonuje próbkowania bez zwracania, co redukuje wariancję oszacowania modelu. Innymi słowy, każda próbka będzie występowała tylko raz w zestawie treningowym lub testowym.

Metoda leave-one-out
Metoda minus jednego elementu (ang. Leave-One-Out-Method) Jest to odmiana k-krotnej walidacji krzyżowej. w tym wypadku liczba podzbiorów jest równa liczbie próbek uczących (rys. 28). w związku z tym, metoda LOO wymaga znacznie

więcej mocy obliczeniowej w stosunku do k-krotnego sprawdzianu krzyżowego, zatem zaleca się jej stosowanie w przypadku małych zbiorów danych.

Rysunek 28: Leave-one-out jest odmianą walidacji K-krotnej, gdy N-elementowa próba jest dzielona na N podzbiorów, zawierających po jednym elemencie. Stosowana często dla małych zbiorów danych.

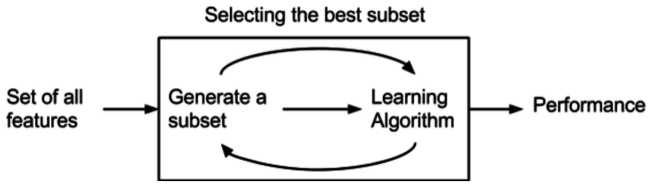
28 Selekcja cech

Przesłanki:

- wybór spośród wszystkich zmiennych objaśniających, podzbioru cech informatywnych z punktu widzenia rozważanego problemu, odrzucenie informacji redundantnych,
- redukcja wymiarowości przestrzeni danych,
- zniwelowanie ich zaszumienia.

Najlepszy podzbiór powinien zawierać minimalną liczbę cech, o największym wpływie na jakość modelu.

Idea (rys. 29):



Rysunek 29: Selekcja cech (ang. *Feature Selection*), w ogólności, polega na wyborze podzbioru cech, a następnie przeprowadzeniu uczenia modelu w oparciu o ten podzbiór. Jeśli rezutaty, zwrócone przez model są zadowalające, procedura zostaje zakończona. W przeciwnym przypadku podzbiór podlega modyfikacji, a model ponownemu uczeniu.

28.1 Forward search dla n cech

Jest iteracyjną metodą, w której zaczynamy od pustego zbioru cech w modelu. w każdej iteracji dodajemy zmienną objaśniającą, która najlepiej wpływa na jakość predykcji modelu w procesie kroswalidacji. Kroki iteracyjne powtarzane są dopóki, dopóty dodanie nowej zmiennej nie poprawi wydajności modelu.

Ustalenie początkowego $\mathcal{F} = \emptyset$.
Iteracyjnie{
(1.) Dla $i = 1, ..., n$ dodanie i-tej cechy to zbioru $\mathcal{F}$
i przeprowadzenie kroswalidacji.
(2.) Ustalenie $\mathcal{F} = \mathcal{F} \cup \{$najlepsza cecha znaleziona w kroku (1.)$\}$
}
Zwrócenie najlepszej znaleznionej hipotezy.

28.2 Backward search

W kroku początkowym jako $\mathcal{F}$ przymuje się zbiór wszystkich dostępnych cech i w kolejnych krokach dokonywana jest eliminacja tych najmniej znaczących. Usuwanie trwa tak długo, jak długo eliminacja kolejnych cech pozytywnie wpływa na jakość modelu.

28.3 "Filter" method

Dla każdej cechy i, obliczana jest wartość pewnej miary, względem której oceniana jest informatywność x_i w kontekście predykcji y. A następnie np. z użyciem kroswalidacji wybiera się k najbardziej istotnych cech.
Przykładem miary powszechnie wykorzystywanej w tym podejściu jest korela-

cja. w szczególności w przypadku klasyfikacji tekstu stosuje się:

$$\underbrace{MI(x_i, y)}_{Mutual\ information} = \sum_{x_i \in \{0,1\}} \sum_{y_i \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} = KL(p(x, y)||p(x)p(y))$$

29 Wnioskowanie bayesowskie na przykładzie regresji liniowej

W rozważaniach problemu optymalizacji modeli regresji liniowej, zastosowanie znalazła metoda największej wiarygodności (ang. *Maximum Likelihood*), za pomocą której znajdowaliśmy $\max_\theta \prod_{i=1}^m p(y^{(i)}|x^{(i)}, \theta)$. Alternatywą dla takiego podejścia jest wnioskowanie bayesowskie.
Niech $P(\theta)$ - prawdopodobieństwo a priori, $\mathcal{S} = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$ - zbiór uczący. Mając zbiór treningowy $\mathcal{S}$ jesteśmy w stanie policzyć rozkład a posteriori: $P(\theta|\mathcal{S})$.

$$P(\theta|\mathcal{S}) \sim \prod_{i=1}^{m} (P(y^{(i)}|x^{(i)}, \theta))P(\theta)$$

Dla nowej obserwacji x:

$$P(y|x, \mathcal{S}) = \int_\theta P(y|x, \theta)P(\theta|\mathcal{S})d\theta$$

$$\mathbf{E}[y|x, \mathcal{S}] = \int_y yP(y|x, \mathcal{S})dy$$

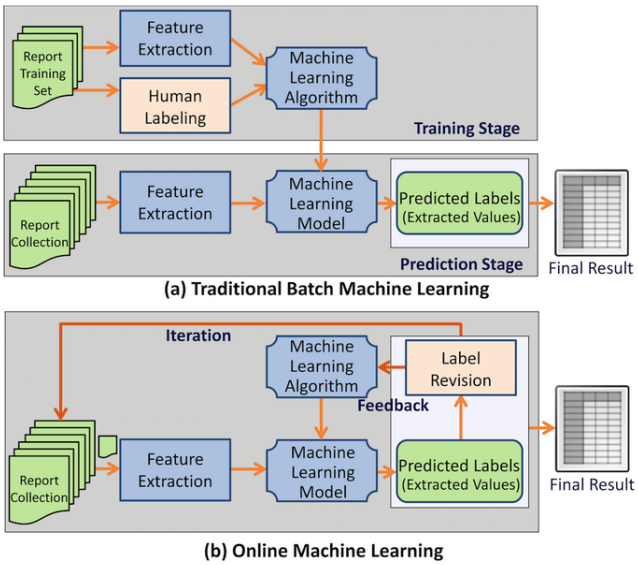
Często stosowaną praktyką, pozwalającą uniknąć obliczania pełnego rozkładu $P(\theta|\mathcal{S})$, jest maksymalizacja wyrażenia $\prod_{i=1}^m (P(y^{(i)}|x^{(i)}, \theta))P(\theta)$.
Niech $\hat{\theta}_{MAP} = arg \max_\theta P(\theta|\mathcal{S}) = arg \max_\theta \prod_{i=1}^m (P(y^{(i)}|x^{(i)}, \theta))P(\theta)$.
Wówczas w celu dokonania predykcji dla nowej obserwacji x, należy wyznaczyć $h_{\hat{\theta}_{MAP}}(x) = \hat{\theta}_{MAP}^T \cdot x$. Jest to efektywne podejście w szczególności w przypadku klasyfikatorów tekstu.

30 Online Learning

Metoda uczenia maszynowego, w której dane stają się dostępne w kolejności sekwencyjnej i są używane do bieżącej poprawy jakości predykcji modelu (rys. 30). Dąży się bowiem, do bieżącej minimalizacji wartości błędu, wyrażonego przez:

$$\sum_{i=1}^{m} \mathbf{1}\{\hat{y}^{(i)} \neq y^{(i)}\}$$

Jest to inne podejście niż w przypadku tradycyjnych technik (ang. *Batch Techniques*), w których szkolenie modelu odbywa się jednocześnie na całym zbiorze treningowym. Online Learning jest powszechnie stosowane w obszarach uczenia maszynowego, w których obliczeniowo niewykonalne jest trenowanie uczenie modelu na całym zbiorze danych w sytuacjach, w których algorytm musi dynamicznie dostosowywać się do nowych wzorców w danych lub gdy same dane są generowane dynamicznie, jako funkcja czasu.



Rysunek 30: Tradycyjne algorytmy uczenie maszynowe mają charakter dwuetapowy: szkolenie oparte na zbiorze danych szkoleniowych z etykietami (ang. *Batch Training*) i predykcja dla nowych obserwacji, oparta na modelu wygenerowanym w procesie uczenia. Algorytmy uczenia maszynowego online (ang. *Online Machine Learning*) wdrażają podejście iteracyjne. Algorytm uczy się na pewnej części zbioru uczącego i dokonuje predykcji dla następnej. Uczenie się odbywa się na podstawie poprawek dokonanych przez użytkownika.

31 Porady dotyczące efektywnego stosowania algorytmów uczenia maszynowego

31.1 Debugowanie algorytmów uczenia

Przykład:
Rozważmy problem klasyfikacji anty-spam, znany z poprzednich wykładów.
Metodolodia:

- Spośród 50000+ angielskich słów wybieramy niewielki podzbiór, np. stu, które będziemy traktować jako cechy.
- Stosujemy wnioskowanie bayesowskie dla regresji logistycznej

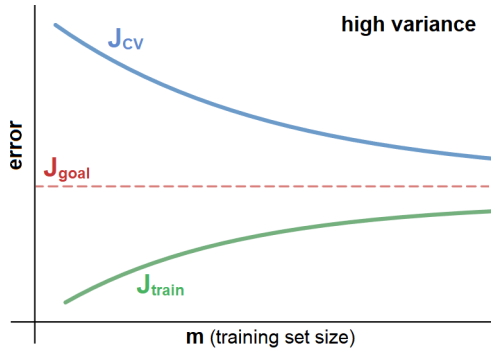
$$\max_{\theta} \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda\|\theta\|^2$$

  z metodą najszybszego spadku (ang. *Gradient Descent*), w efekcie otrzymując wysoki 20%-owy błąd. Jak go zniwelować?

- Popularnym podejściem jest próba poprawy algorytmu poprzez:
  - dostarczenie większej ilości obserwacji treningowych,
  - pomniejszenie zbioru zmiennych objaśniających,
  - powiększenie zbioru zmiennych objaśniających,
  - zmiana zmiennych objaśniających,
  - zwiększenie liczby iteracji algorytmu gradient descent,
  - zastosowanie metody Newtona,
  - zmiana wartości λ,
  - zastosowanie SVM.

  Takie postępowanie, pomimo że może doprowadzić do zniwelowania błędu, postrzegane jest jako naiwne. Wymaga sprawdzenia wszystkich powyższych możliwości, przez co okazuje się bardzo czasochłonne, a to, czy ostatecznie doprowadzi do rozwiązania faktycznego problemu w dużej mierze jest kwestią szczęścia.

- Lepsze podejście: **zbadanie modelu pod kątem błędów wariancji i obciążenia.**
  - w przypadku wysokiego błędu wariancji, błąd predykcji na zbiorze testowym maleje wraz ze wzrostem ilości obserwacji w tym zbiorze. Występuje znacząca różnica błędów predykcji na zbiorach treningowym i testowym (rys. 31).

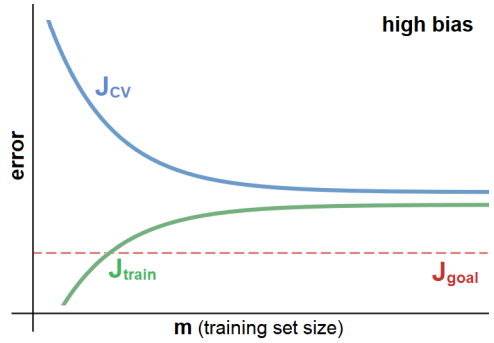


Rysunek 31: Wysoki błąd wariancji. Niewielka wartość m: J_{train} - niski, J_{CV} - wysoki. Wraz ze wzrostem m: J_{train} rośnie, J_{CV} maleje. Zwiększenie zbioru nie pomaga.

Możliwe rozwiązania:

* zwiększenie liczby obserwacji uczących,
* zmniejszenie liczby zmiennych objaśniających.

– w przypadku wysokiego błędu obciążenia, błędy predykcji na zbiorach uczącym i walidacyjnym będą wysokie, ich różnica będzie niewielka (rys. 32).



Rysunek 32: Wysoki błąd obciążenia. Niewielka wartość m: J_{train} - niski, J_{CV} - wysoki. Duże m: $J_{train} \approx J_{CV}$ - wysokie. Zwiększenie zbioru nie pomaga.

Możliwe rozwiązania:

* zwiększenie liczby zmiennych objaśniających,
* zmiana zmiennych objaśniających.

- Innymi aspektami wartymi rozważenia są:
  – zbadanie zbieżności metody najszybszego spadku,

– adekwatność doboru maksymalizowanej funkcji do celu, który chcemy osiągnąć.
Przykład:
odpowiedni dobór wag dla spamu i właściwych wiadomości w funkcji

$$a(\theta) = \sum_i w^{(i)} \mathbf{1}\{h_\theta(x^{(i)}) = y^{(i)}\}$$

– odpowiednie wartości stałych, np. λ w

$$\max_\theta \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda\|\theta\|^2$$

Przykład:
Rozważmy sytuację, w której dla tego samego problemu zastosowano dwa różne algorytmy: bayesowską regesję logistyczną i SVM. Okazało się, że lepsze rezultaty względem błędu predykcji otrzymano dla SVM, jednak ze względu na dostępną moc obliczeniową zależy nam na zastosowaniu regresji. Co zrobić w takiej sytuacji?
Niech θ_{SVM}, θ_{BLR} oznaczają zbiory optymalnych parametrów odpowiednio dla algorytmów SVM i bayesowskiej regresji logistycznej. Rozważamy dokładność ważoną

$$a(\theta) = \max_\theta \sum_i w^{(i)} \mathbf{1}\{h_\theta(x^{(i)}) = y^{(i)}\}$$

Dla SVM jakość modelu względem tej miary była lepsza, zatem:

$$a(\theta_{SVM}) > a(\theta_{BLR})$$

BLR maksymalizuje funkcję:

$$J(\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}, \theta) - \lambda\|\theta\|^2$$

Zatem możliwe są dwa przypadki:

(I)

$$\begin{cases} a(\theta_{SVM}) > a(\theta_{BLR}) \\ J(\theta_{SVM}) > J(\theta_{BLR}) \end{cases}$$

Wówczas wyznaczony przez BLR zbiór parametrów nie maksymalizuje funkcji J, a stąd wynika niezbieżność algorytmu.
Zatem przyczyną problemu jest algorytm optymalizujący.

Możliwe rozwiązania:

– zwiększenie liczby iteracji algorytmu gradient descent,

– zastosowanie metody Newtona.

(II)

$$\begin{cases} a(\theta_{SVM}) > a(\theta_{BLR}) \\ J(\theta_{SVM}) \leqslant J(\theta_{BLR}) \end{cases}$$

w tym przypadku BLR poprawnie maksymalizuje funkcję J, a mimo to lepszy wynik względem dokładności a, otrzymujemy dla algorytmu SVM. to oznacza, że funkcja celu J problemu optymalizacji nie jest odpowiednio dobrana.

Możliwe rozwiązania:

– zastosowanie metody Newtona,

– zmiana wartości λ.

31.2 Analiza błędu

Celem analizy błędu jest określenie różnicy pomiędzy rezultatami: otrzymanym i idealnym.
Celem analizy "ablacyjnej" (ang. *Ablative Analysis*) jest określenie różnicy pomiędzy rezultatami: wyjściowym (podstawowym) i aktualnym.

Załóżmy, że dzięki dodaniu do rozważanego klasyfikatora anty-spam komponentów takich jak:

- korekta pisowni,
- zmienna hosta nadawcy,
- zmienna tematu wiadomości,
- parsery tekstu,
- parser Javascript

jakość modelu, określona względem pewnej miary, wzrosła z 94% do 99.9%.
Aby określić jak poszczególne komponenty wpłynęły na jakość modelu i wybrać te najistotniejsze, agreguje się ablative analysis, tzn. kolejno usuwa się każdy z nich a następnie weryfikuje, jak poszczególne modyfikacje wpłynęły na jakość modelu, np.:

Component	Accuracy
Wydajność wyjściowa	99.9%
Korekta pisowni	99.0%
Zmienna hosta nadawcy	98.9%
Zmienna tematu wiadomości	98.9%
Parsery tekstu	95.0%
Parser Javascript	94.5%

Tabela 3: Dokładność modelu pozbawionego poszczególnych komponentów

Wniosek:
Parsery tekstu i Javascript najsilniej wpływają na poprawę jakości modelu.

32 Algorytm k-średnich (ang. *K-Means Clustering*)

Metoda k-średnich należy do grupy algorytmów analizy skupień, tj. analizy polegającej na szukaniu i wyodrębnianiu grup obiektów podobnych. Algorytm ten polega na przenoszeniu obiektów ze skupienia do skupienia tak długo, aż zostaną zoptymalizowane zmienności wewnątrz skupień oraz pomiędzy skupieniami. Podobieństwo w skupieniu powinno być jak największe, zaś osobne skupienia powinny się maksymalnie od siebie różnić.

32.1 Metodologia

Wejście: $\{x^{(1)}, ..., x^{(m)}\}$
Etap 1: Losowa inicjalizacja wstępnych środków skupień $\mu_1, ..., \mu_k \in \mathcal{R}^n$
Etap 2:

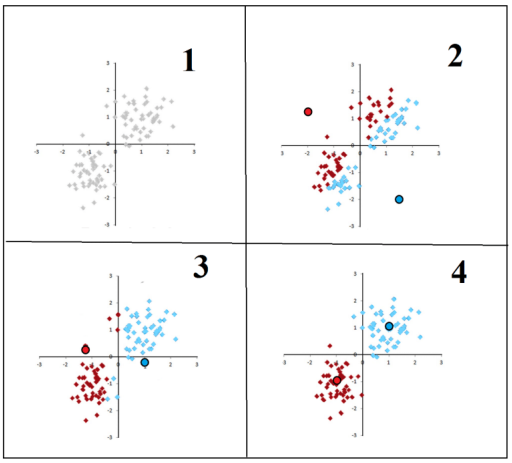
Krok 1: Przypisanie obiektów do skupień:

$$c^{(i)} = arg\min_j \|x^{(i)} - \mu_j\|$$

Krok 2: Ustalenie nowych centroidów:

$$\mu_j := \frac{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)} = j\}}$$

Iteracyjne powtarzanie kroków do uzyskania zbieżności.



Rysunek 33: Wizualizacja metodologii algorytmu k-średnich (ang. *K-Means Algorithm*). Algorytm losowo przypisuje dwa środki skupień dla punktów wejściowych. Każdy punkt danych jest przypisany do centroidu klastra zgodnie z predefiniowaną funkcją odległości. W następnym kroku centroidy są ponownie obliczane. Metoda wykonuje te kroki wielokrotnie, aż do braku zmiany centroidu.

32.2 Funkcja rozproszenia (distortion function)

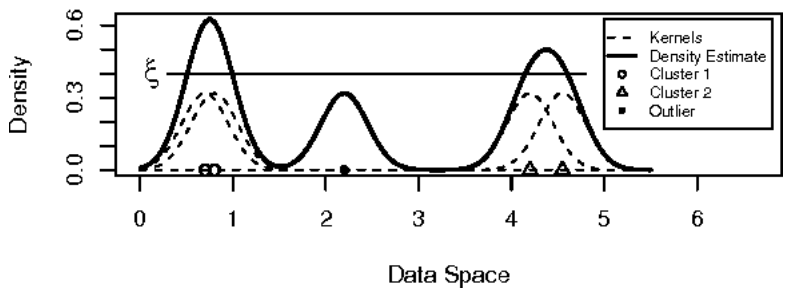
Naszym celem jest więc znalezienie centrów klastrów oraz przyporządkowanie obserwacji do klastrów tak, aby minimalizować sumę kwadratów odległości punktów do najbliższego centra.
Naszą funkcją celu jest więc tzw. funkcja rozproszenia:

$$\mathcal{J}(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

33 Modele klasteryzacji oparte na jądrowej estymacji gęstości (ang. *Clustering based on Kernel Density*)

33.1 Intuicja

Klaster jest zdefiniowany przez lokalne maksimum szacowanej funkcji gęstości. Punkty danych są przypisywane do klastrów w taki sposób, że obserwacje dążące do tego samego maksimum lokalnego są umieszczane w tym samym klastrze (rys. 34).



Rysunek 34: Klasteryzacja w oparciu o estymację gęstości (ang. *density estimation*) jądra (ang. *kernel*) i próg zaszumienia ξ

33.2 Algorytm EM (ang. *Expectation–Maximization Algorithm) w estymacji gęstości*

Załóżmy, że mamy dany zbiór obserwacji $\{x^{(1)}, ..., x^{(m)}\}$, ale ponieważ rozważamy problem uczenia nienadzorowanego, nie dysponujemy zmienną odpowiedzi. Chcielibyśmy modelować dane za pomocą łącznego rozkładu $p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})p(z^{(i)})$. w naszym przypadku $z^{(i)} \sim Multinomial(\phi)$ (gdzie $\phi_j > 0, \sum_{j=1}^{k} \phi_j = 1, \phi_j = p(z^{(i)} = j)$, $x(i)|z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$ oraz k oznacza liczbę wartości jakie mogą przyjąć poszczególne $z^{(i)}$). A zatem, nasz model zakłada, że każdy $x^{(i)}$ został wygenerowany przez losowe wybranie $z^{(i)}$ ze zbioru $1, ..., k$, a następnie $x^{(i)}$ zostało wyodrębnione z jednego z k rozkładów Gaussa, w zależności od wylosowanego $z^{(i)}$. Jest to tzw. **mixture of Gaussians model**. Zauważmy, że wartości $z^{(i)}$ są utajone, czyli ukryte lub nieobserwowane.
Parametrami rozważanego modelu są: ϕ, μ, Σ. w celu ich estymacji można spróbować wykorzystać, znaną już z wcześniejszych rozważań funkcję:

$$l(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}; \phi, \mu, \Sigma) = \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)}; \mu, \Sigma)p(z^{(i)}; \phi)$$

Okazuje się, że zróżniczkowanie powyższej funkcji względem odpowiednich parametrów i przyrównanie do 0 nie pozwala uzyskać prawidłowo wyestymowanych wartości.
Wartości losowe $z^{(i)}$ wskazują, z którego spośród k rozkładów pochodzą odpowiadające im obserwacje $x^{(i)}$. Znajomość ich wartości, znacznie uprościłaby problem największej wiarygodności. Wówczas otrzymalibyśmy funkcję:

$$l(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}|z^{(i)}; \mu, \Sigma) + p(z^{(i)}; \phi),$$

której rezultaty po maksymalizacji przyjęłyby postaci:

$$\phi_j = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}\{z^{(i)} = j\}$$

$$\mu_j = \frac{\sum_{i=1}^m \mathbf{1}\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^m \mathbf{1}\{z^{(i)} = j\}}$$

$$\Sigma_j = \frac{\sum_{i=1}^m \mathbf{1}\{z^{(i)} = j\} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^m \mathbf{1}\{z^{(i)} = j\}}$$

Jak wyestymować te parametry nie znając wartości $z^{(i)}$?
Algorytm EM jest algorytmem iteracyjnym, który składa się z dwóch głównych kroków. Zastosowany do naszego problemu, w kroku E, próbuje „odgadnąć" wartości $z^{(i)}$. w kroku M aktualizuje parametry naszego modelu na podstawie $z^{(i)}$, które nie jest już niewiadomą.
Iteracyjnie do uzyskania zbieżności{

(Krok E.) Ustalenie wartości z z odpowiedniego rozkładu i oszacowanie:

$$w_j^{(i)} := \mathbf{P}(z^{(i)}|x^{(i)}, \phi, \mu, \Sigma)$$

(Krok M.)

$$\phi_j := \frac{1}{m} \sum_{i=1}^m w_j^{(i)}$$

$$\mu_j = \frac{\sum_{i=1}^m w_j^{(i)} x^{(i)}}{\sum_{i=1}^m w_j^{(i)}}$$

$$\Sigma_j := \frac{\sum_{i=1}^m w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^m w_j^{(i)}}$$

}

W kroku E prawdopodobieństwo a posteriori obliczane jest z wykorzystaniem reguły Bayesa:

$$\mathbf{P}(z^{(i)}|x^{(i)}, \phi, \mu, \Sigma) = \frac{\mathbf{P}(x^{(i)}|z^{(i)} = j)\mathbf{P}(z^{(i)} = j)}{\sum_{l=1}^k \mathbf{P}(x^{(i)}|z^{(i)} = l)\mathbf{P}(z^{(i)} = l)}$$

34 Analiza czynników (ang. *Factor Analysis*)

Kiedy mamy dane $x^{(i)} \in \mathcal{R}^n$, które pochodzą z mieszaniny rozkładów Gaussa, w celu dopasowania modelu można zastosować algorytm EM. w tym podejściu zazwyczaj zakładamy, że w naszym problemie, dysponujemy wystarczającą ilością danych, aby rozpoznać w nich strukturę mieszaniny. na przykład byłby to przypadek, gdy w zbiorze uczącym liczba obserwacji m byłaby znacznie większa niż wymiar danych n.
Rozważmy sytuację, w której $n >> m$. w takim problemie modelowanie za pomocą jednego lub wielu rozkładów normalnych może okazać się problematyczne. w szczególności, jeśli dane obserwacje rozpinają niskowymiarową podprzestrzeń

$\mathcal{R}^n$, modelowanie gaussowskie, w którym wartość oczekiwana i wariancja szacowane są za pomocą standardowych estymatorów:

$$\mu = \sum_{i=1}^{m} x^{(i)}$$

$$\Sigma = \sum_{i=1}^{m} (x^{(i)} - \mu)(x^{(i)} - \mu)^T$$

nie ma sensu, ponieważ otrzymana macierz Σ będzie osobliwa i tym samym nieodwracalna. Innymi słowy wielkość $1/|\Sigma|^{\frac{1}{2}}$ niezbędna do obliczenia gęstości mieszaniny wynosi $1/0$. Bardziej ogólnie: jeśli, $n >> m$, to estymacja maksymalnego prawdopodobieństwa średniej i kowariancji może przynosić niewystarczające rezultaty. Nie mniej, jeśli nadal konsekwentnie, chcielibyśmy dopasować do danych model normalny, możemy to zrobić nakładając na przestrzeń macierzy Σ odpowiednie ograniczenia, np. że roważamy tylko macierze diagonalne, lub diagonalne z jednakowymi wartościami na przekątnej. to pozwoli na otrzymanie nieosobliwej macierzy kowariancji w przypadku, niespełnienia warunku $m \geqslant n + 1$.

35 Model analizy czynnikowej

W modelu analizy czynnikowej zakładamy następujący łączny rozkład, gdzie z jest utajoną zmienną losową:

$$z \sim \mathcal{N}(0, I)$$

$$x|z \sim \mathcal{N}(\mu + \Lambda z, \Psi)$$

Parametrami modelu są: wektor $\mu \in \mathcal{R}^n$, macierz $\Lambda \in \mathcal{R}^{n \times k}$ i macierz diagonalna $\Psi \in \mathcal{R}^{n \times n}$. Zazwyczaj przyjmuje się $k < n$. Zatem zakładamy, że każda obserwacja $x^{(i)}$ jest generowana przez próbkowanie wielowymiarowego rozkładu nomalnego $z^{(i)}$, a następnie jest odwzorowywana na k-wymiarową afiniczną przetrzeń R^n poprzez obliczenie $\mu + \Lambda z^{(i)}$ oraz dodanie do współrzędnych otrzymanego wektora szumu kowariancji Ψ. Równoważnie, możemy zatem zdefiniować model analizy czynnikowej jako:

$$z \sim \mathcal{N}(0, I)$$

$$\varepsilon \sim \mathcal{N}(0, \Psi)$$

$$x = \mu + \Lambda z \ + \varepsilon$$

gdzie ε i z są niezależne.

Zmienne losowe x i z mają łączny rozkład normalny $\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}(\mu_{zx}, \Sigma)$ Wyznaczmy μ_{zx} i Σ.

z powyższych własności wiadomo, że:

$$\mathbf{E}[z] = 0$$

$$\mathbf{E}[x] = \mathbf{E}[\mu + \Lambda z \ + \varepsilon] = \mathbf{E}[\mu] + \Lambda\mathbf{E}[z] + \mathbf{E}[\varepsilon] = \mu$$

Stąd: $\mu_{zx} = \begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}$

w celu wyznaczenia Σ, należy kolejno wyznaczyć:

$$\Sigma_{zz} = \mathbf{E}[(z - \mathbf{E}[z])(z - \mathbf{E}[z])^T]$$

$$\Sigma_{zx} = \mathbf{E}[(z - \mathbf{E}[z])(x - \mathbf{E}[x])^T]$$

$$\Sigma_{xx} = \mathbf{E}[(x - \mathbf{E}[x])(x - \mathbf{E}[x])^T]$$

Ponieważ $z \sim \mathcal{N}(0, I)$, to $\Sigma_{zz} = Cov(z) = I$. Ponadto:

$$\mathbf{E}[(z - \mathbf{E}[z])(x - \mathbf{E}[x])^T] = \mathbf{E}[z(\mu + \Lambda z \ + \varepsilon - \mu)^T] = \mathbf{E}[zz^T]\Lambda^T + \mathbf{E}[z\varepsilon^T] =$$

$$= Cov(z)\Lambda^T + \mathbf{E}[z]\mathbf{E}[\varepsilon^T] = I\Lambda^T + 0 = \Lambda^T$$

Analogicznie:

$$\Sigma_{xx} = \mathbf{E}[(\mu+\Lambda z +\varepsilon-\mu)(\mu+\Lambda z +\varepsilon-\mu)^T] = \mathbf{E}[\Lambda z z^T\Lambda^T+\varepsilon z^T\Lambda^T+\Lambda z\, \varepsilon^T+\varepsilon\varepsilon^T] =$$

$$= \Lambda\mathbf{E}[zz^T]\Lambda^T + \mathbf{E}[\varepsilon\varepsilon^T] = \Lambda\Lambda^T + \Psi$$

Stąd:

$$\begin{bmatrix} z \\ x \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \vec{0} \\ \mu \end{bmatrix}, \begin{bmatrix} i & \Lambda^T \\ \Lambda & \Lambda\Lambda^T + \Psi \end{bmatrix} \right)$$

Widzimy zatem, że rozkład brzegowy x: $x \sim \mathcal{N}(\mu, \Lambda\Lambda^T + \Psi)$. Zatem:

$$l(\mu, \Lambda, \Psi) = \log \prod_{i=1}^{m} \frac{1}{(2\pi)^{n/2}|\Lambda\Lambda^T + \Psi|^{1/2}} \exp\{-\frac{1}{2}(x^{(i)}-\mu)^T(\Lambda\Lambda^T+\Psi)^{-1}(x^{(i)}-\mu)\}$$

Analityczna maksymalizacja powyższej formuły względem poszczególnych parametrów jest skomplikowana obliczeniowo, dlatego w rozważanym problemie wykorzystuje się algorytm EM.

36 Analiza głównych składowych (ang. *Principal Component Analysis, PCA*)

W omówieniu analizy czynnikowej rozważaliśmy sposób modelowania danych $x \in \mathcal{R}^n$, które w przybliżeniu leżały w podprzestrzeni k-wymiarowej, gdzie $k < n$. w szczególności zakładaliśmy, że każdy punkt $x^{(i)}$ został utworzony przez wygenerowanie $z^{(i)}$ leżącego w przestrzeni afinicznej wymiaru k, a następnie dodanie szumu kowariancji. Analiza czynnikowa jest oparta o model probabilistyczny, do estymację parametrów wykorzystano iteracyjny algorytm EM. Metoda Principal Components Analysis (Analiza Głównych Składowych, ang.

Principal Component Analysis (PCA)), również ma na celu zidentyfikowanie podprzestrzeń, w której "w przybliżeniu" znajdują się dane. Analiza głównych składowych (PCA) to metoda redukcji wymiarów, która jest często stosowana w celu zmniejszenia wymiarowości dużych zbiorów danych, poprzez przekształcenie dużego zestawu zmiennych w mniejszy, który wciąż zawiera większość informacji znajdujących się w zbiorze wyjściowym.

36.1 Krok 1.: Standaryzacja

Celem tego kroku jest ujednolicenie zakresu ciągłych zmiennych w wyjściowym zbiorze, tak aby każda z nich miała równy udział w analizie. Powodem, dla którego niezwykle ważne jest przeprowadzenie standaryzacji przed PCA, jest fakt, że metoda ta jest dość wrażliwa na wariancje zmiennych początkowych. Oznacza to, że jeśli istnieją duże różnice między zakresami zmiennych początkowych, zmienne o większych zakresach będą dominować nad zmiennymi o małych zakresach co doprowadzi do przekłamania rezultatów.

36.2 Krok 2.: Wyznaczenie macierzy kowariancji

Zdarza się, że wysoce skorelowane zmienne nie niosą żadnych dodatkowych informacji, a jedynie zwiększają wymiarowość zbioru danych. Celem tego kroku jest identyfikacja korelacji pomiędzy zmiennymi, innymi słowy, sprawdzenie, czy istnieje między nimi jakaś zależność.

36.3 Krok 3.: Wyznaczenie wektorów i własności własnych

Wektory własne i wartości własne to pojęcia algebry liniowej, które musimy obliczyć z macierzy kowariancji, aby określić główne składowe.
Główne składowe to nowe zmienne, które są konstruowane jako kombinacje liniowe. Te kombinacje są budowane w taki sposób, że nowe zmienne (tj. główne składniki) są nieskorelowane, a większość informacji w zmiennych początkowych jest skoncentrowana w pierwszych składnikach. Tak więc pomysł polega na tym, że dane 10-wymiarowe dają 10 głównych składników, ale PCA stara się umieścić możliwie jak najwięcej informacji w pierwszym z nich, a następnie maksymalną pozostałą informację w drugim itd. Ważnym aspektem metody jest fakt, że główne elementy są mniej interpretowalne i nie mają żadnego rzeczywistego znaczenia, ponieważ są skonstruowane jako liniowe kombinacje zmiennych początkowych. Geometrycznie, główne komponenty reprezentują kierunki danych, to znaczy prostej, które przechwytują większość informacji w danych. Zależność między wariancją a informacją jest taka, że im większa wariancja, tym większe rozproszenie punktów danych wzdłuż prostej, a im większe rozproszenie wzdłuż prostej, tym bardziej informatywna jest dana zmienna. Mówiąc najprościej, wystarczy myśleć o głównych składowych jako o nowych osiach, które zapewniają najlepszy kąt widzenia i oceny danych. Pierwszy główny składnik to prosta, względem której rzut punktów (obserwacji) jest najbardziej rozproszony lub

matematycznie, jest to prosta, która maksymalizuje wariancję (średnią kwadratowych odległości od rzutowanych punktów. Drugi główny składnik oblicza się w ten sam sposób, pod warunkiem, że nie jest on skorelowany z pierwszym głównym składnikiem i że uwzględnia następną najwyższą wariancję.
Wektory własne macierzy kowariancji są w rzeczywistości kierunkami osi, w których występuje największa wariancja i które nazywamy głównymi składowymi, a wartości własne to współczynniki związane z wektorami własnymi, które dają wielkość wariancji przenoszonej w każdym głównym składniku. Obliczenie wektorów własnych i uporządkowanie ich według ich wartości własnych w porządku malejącym pozwala nam znaleźć główne składniki w kolejności istotności.

36.4 Krok 4: Wybór wektora cech

na tym etapie wybieramy, czy zachować wszystkie składniki, czy też odrzucić te o mniejszym znaczeniu (o niskich wartościach własnych) i utworzyć z pozostałymi macierz wektorów, które nazywamy wektorem cech. Tak więc wektor cech jest po prostu macierzą, która ma jako kolumny wektory własne komponentów, które postanowiliśmy zachować. Jest to pierwszy krok w kierunku zmniejszenia wymiarów, ponieważ jeśli zdecydujemy się trzymać tylko p wektorów własnych (składników) spośród n, końcowy zestaw danych będzie miał tylko p wymiarów.

36.5 Krok 5: Przekształcenie danych względem osi, wyznaczanych przez główne komponenty

W ostatnim kroku, celem jest użycie wektora cech utworzonego za pomocą wektorów własnych macierzy kowariancji, w celu zmiany orientacji danych z osi pierwotnych na te reprezentowane przez główne składniki. Można tego dokonać mnożąc transpozycję oryginalnego zestawu danych przez transpozycję wektora cech:

$$\text{Zbiór przetransformowany} = (\text{Wektor Cech})^T \times (\text{Standaryzowany zbiór wejściowy})^T$$

37 Analiza składowych niezależnych (ang. *Independent Component Analysis, ICA*)

Analiza składowych niezależnych (ICA - Independent Component Analysis) jest sprawdzoną i skuteczną metodą przeznaczoną do wydzielania sygnału z wielu zmiennych. Poszukiwanie w danych interesującego, pierwotnego sygnału jest częstym problemem, zasadniczym w ogólnym temacie statystycznego przetwarzania sygnałów, szeroko wykorzystywanym w takich dziedzinach jak analiza dźwięku, analiza obrazu.
Wyobraźmy sobie pełną słuchaczy salę, w której dwóch prelegentów jednocześnie wygłasza swoją prezentację. Słuchacze, komentują wypowiedzi i powodują szum. My, jesteśmy natomiast zainteresowani tym, co mówią prelegenci, a nie dźwiękami dochodzącymi z sali. w dwóch miejscach sali umieszczone są dwa

mikrofony, rejestrujące obu mówców jak i hałas pochodzący od sali. Naszym zadaniem jest wyodrębnienie z dwóch zapisów dźwięku dwóch mówców, ignorując przy tym szum tła. Jest to klasyczny przykład analizy składowych niezależnych, ugruntowanej, szeroko stosowanej metody statystycznej wyodrębniania zupełnie nieznanych, niezależnych sygnałów na podstawie kombinacji liniowej zmiennych, które podlegają pomiarom.

38 Problem uczenia się ze wzmocnieniem (ang. *Reinforcement Learning, RL*)

Uczenie się ze wzmocnieniem stanowi sposób rozwiązywania niektórych problemów, których nie da się w prosty sposób rozwiązać analitycznie albo gdy nie dysponujemy dobrym modelem. Nie jest to uczenie się indukcyjne, a więc oparte na generalizacji dużej liczby przykładów, nie jest to także żadna inna forma uczenia się pojęć ani w ogóle żadnej wiedzy deklaratywnej. Uczenie się ze wzmocnieniem ma na celu uczenie się wiedzy proceduralnej (umiejętności). Jest to obliczeniowe podejście do uczenia się celowego zachowania na podstawie interakcji. Źródłem informacji trenującej, która ma charakter wartościujący jakość działania agenta, jest środowisko. Interakcje polegają na obserwacji przez agenta kolejnych stanów środowiska oraz wykonywaniu wybranych zgodnie z jego obecną strategią decyzyjną akcji. Po wykonaniu akcji uczeń otrzymuje rzeczywistoliczbowe wartości wzmocnienia (nagrody lub kary), które stanowią pewną miarę oceny jakości jego działania. Wykonanie akcji może również powodować zmianę stanu środowiska. Uczenie się polega na modyfikacji strategii na podstawie obserwowanych sekwencji stanów, akcji i nagród.

38.1 Procesy decyzyjne Markowa

Modelem matematycznym problemu uczenia się ze wzmocnieniem (a właściwie modelem środowiska) jest proces decyzyjny Markowa (ang. *Markov Decision Process, MDP*), który definiujemy jako czwórkę:

$$\mathrm{MDP} = \langle X, A, \varrho, \delta \rangle,$$

gdzie:
X jest skończonym zbiorem stanów,
A jest skończonym zbiorem akcji,
ϱ jest funkcją nagrody (wzmocnienia),
δ jest funkcją przejścia stanów.

Dla każdej pary $\langle x, a \rangle \in X \times A$ wartość $\varrho(x, a)$ jest zmienną losową oznaczającą nagrodę otrzymywaną po wykonaniu akcji a w stanie x, a wartość $\delta(x, a)$ jest zmienną losową oznaczającą następny stan po wykonaniu akcji a w stanie x. Oznacza to, że r_t jest realizacją zmiennej losowej $\varrho(x_t, a_t)$ oraz x_{t+1} jest realizacją zmiennej losowej $\delta(x_t, a_t)$.

Dla ułatwienia posługiwania się funkcjami przejścia i wzmocnienia wygodnie jest zdefiniować dodatkowe oznaczenia:

$$R(x,a) = \mathbf{E}\big[\varrho(x,a)\big],$$

$$P_{xy}(a) = \Pr\big(\delta(x,a) = y\big).$$

38.2 Własność Markowa

Kluczowe znaczenie ma tzw. własność Markowa: ϱ i δ nie zależą od historii. w każdym kroku nagroda i następny stan zależą (probabilistycznie) tylko od aktualnego stanu i akcji.W praktyce często można spotkać środowiska niemarkowowskie. Jeśli własność Markowa nie zachodzi, agent nie ma dostępu do informacji umożliwiającej wybór najlepszej akcji – mówi się wówczas o ukrytym stanie, rezerwując termin stan dla stanów środowiska, które zapewniałyby własność Markowa.

38.3 Strategie i funkcje wartości

Dla procesów decyzyjnych Markowa możemy formalnie zdefiniować strategie oraz funkcje wartości, wykorzystywane dalej do określania i obliczania strategii optymalnych.

Definicja
Strategią dla procesu decyzyjnego Markowa jest dowolna funkcja $\pi : X \mapsto A$.

Definicja
Funkcja wartości ze względu na strategię π jest dla każdego stanu $x \in X$ określona następująco:

$$V^{\pi}(x) = \mathbf{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \;\middle|\; x_0 = x\right].$$

W powyższej definicji symbol $\mathbf{E}_{\pi}$ oznacza wartość oczekiwaną przy założeniu posługiwania się strategią π.

Definicja
Funkcja wartości akcji ze względu na strategię π jest dla każdej pary stan-akcji $\langle x, a\rangle \in X \times A$ określona następująco:

$$Q^{\pi}(x,a) = \mathbf{E}_{\pi}\left[\varrho(x,a) + \sum_{t=1}^{\infty} \gamma^t r_t \;\middle|\; x_0 = x, a_0 = a\right].$$

38.4 Optymalność strategii

Posługując się powyższymi definicjami można sformalizować pojęcie optymalności strategii.

Definicja
Mówimy, że strategia π' jest lepsza od strategii π jeśli $V^{\pi'}(x) \geqslant V^{\pi}(x)$ dla każdego $x \in X$ i przynajmniej dla jednego stanu nierówność jest ostra.

Definicja
Strategią optymalną jest każda strategia, dla której nie istnieje strategia od niej lepsza. Inaczej mówiąc, strategią optymalną jest (każda) strategia maksymalizująca wartość każdego stanu.

Definicja
Strategią zachłanną względem fukcji wartości V jest (każda) strategia określona następująco:

$$\pi(x) = \arg\max_a \Big[R(x,a) + \gamma \sum_y P_{xy}(a)V(y)\Big].$$

Definicja
Strategią zachłanną względem fukcji wartości akcji Q jest (każda) strategia określona następująco:

$$\pi(x) = \arg\max_a Q(x,a).$$

Dowolna strategia zachłanna względem optymalnej funkcji wartości lub optymalnej funkcji wartości akcji jest strategią optymalną.

Literatura

[1] https://www.youtube.com/watch?v=UzxYlbK2c7E&list=PLA89DCFA6ADACE599

[2] http://www.miroslawmamczur.pl/czym-jest-uczenie-maszynowe-i-jakie-sa-rodzaje/

[3] http://itcraftsman.pl/wstep-do-machine-learning/

[4] https://archive.bluesoft.com/pl/machine-learning-co-kryje-pojeciem/

[5] https://kcir.pwr.edu.pl/~witold/ai/ml_intro_s.pdf

[6] https://brain.fuw.edu.pl/edu/index.php/Uczenie_maszynowe_i_sztuczne_sieci_neuronowe

[7] https://medium.com/

[8] https://www.quora.com

[9] http://www.cs.put.poznan.pl/jstefanowski/mlteaching.html

[10] https://pl.wikipedia.org

[11] https://mlinpl.pl

[12] https://www.statystyka.az.pl

[13] https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[14] https://www.statsoft.pl/textbook/stathome_stat.html?https%3A%2F%2Fwww.statsoft.pl%2Ftextbook%2Fstindcomp.htmlprinc

[15] https://dloranc.github.io

[16] http://staff.elka.pw.edu.pl/~pcichosz/um/wyklad/arch/wyklad12/wyklad12.html

Spis skrótów

GDA Ogólna Analiza Dyskryminacyjna, ang. *Gaussian Discriminant Analysis*. 27

GLM Uogólnione Modele Liniowe, ang. *Generalized Linear Model*. 23, 24

LWR Regresja Lokalnie Ważona, ang. *Locally Weighted Regression*. 16

PCA Analiza Głównych Składowych, ang. *Principal Component Analysis*. 63, 64

SVM Metoda Wektorów Nośnyche, ang. *Support Vector Machine*. 2, 32, 38, 41, 43–45, 54, 56, 57