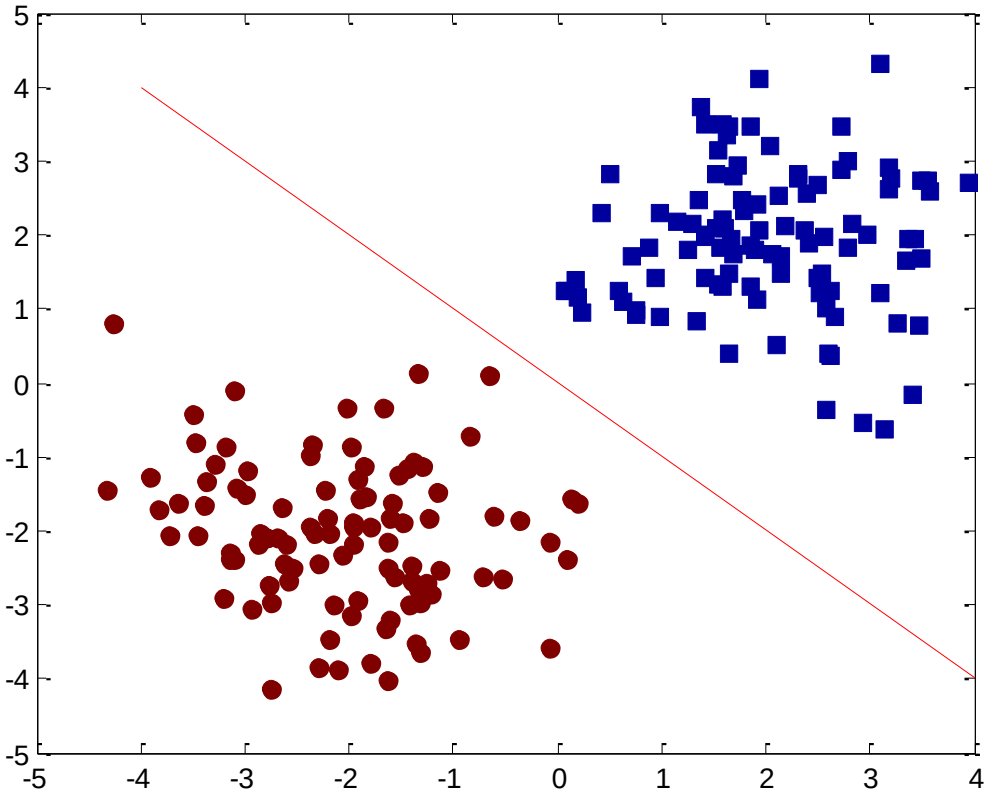
Klasyfikator liniowy

Wstęp

Klasyfikator liniowy jest najprostszym możliwym klasyfikatorem. Zakłada on liniową separację – liniowy podział dwóch klas między sobą. Przedstawia to poniższy rysunek:



Gdzie dwie klasy oddzielone są hiperpłaszczyzną, która dla przestawionego przypadku redukuje się do linii prostej w n-wymiarowej przestrzeni. Funkcje taką nazywamy funkcją decyzyjną

$$g(\mathbf{x}) = \mathbf{w}\mathbf{x}^{\mathbf{T}} + \mathrm{b} = w_1 x_1 + w_2 x_2 + \ldots + w_{\mathrm{n}} x_n + b$$

Gdzie

w – szukany wektor wag

x – wektor wejściowy/wektor sygnału wejściowego

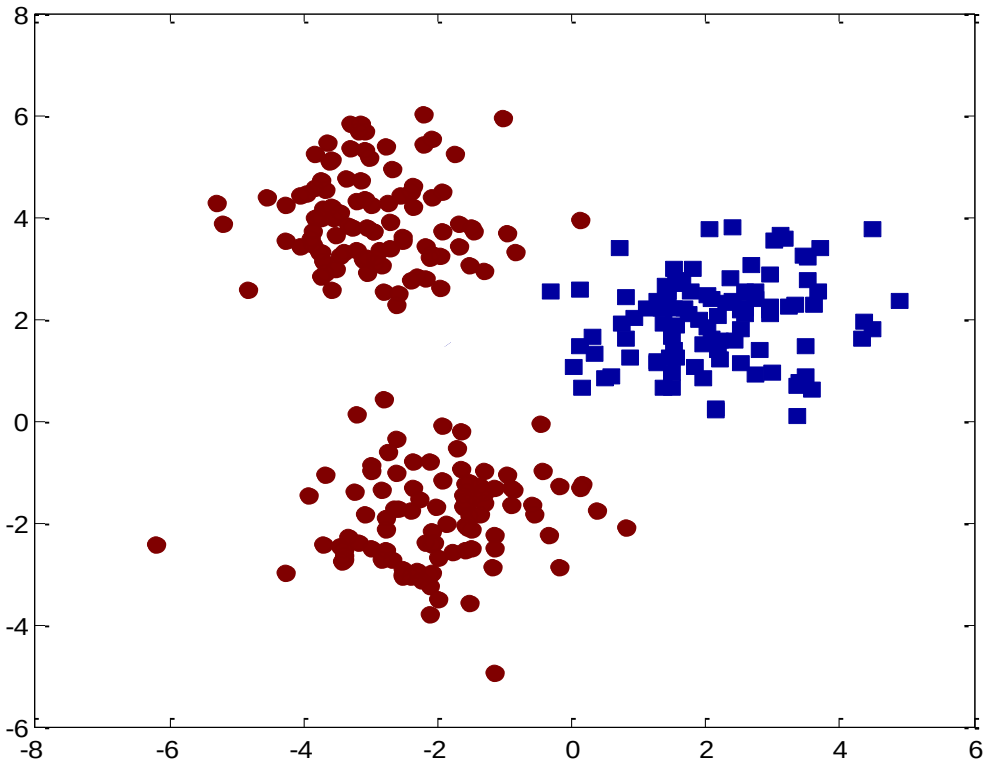
b – wyraz wolny

Funkcja decyzyjna rozdziela więc obydwie klasy i określa wyjście modelu wg. zależności

$g(\mathbf{x}) > 0$ - klasa c1

$g(\mathbf{x}) < 0$ - klasa c2

W pewnych sytuacjach dyskryminacja liniowa nie jest możliwa do osiągnięcia, jak na poniższym obrazku,



dlatego też część modeli liniowych uwzględnia taką możliwość poprzez minimalizację liczby popełnianych błędów.

Do typowych modeli dyskryminacji liniowej można zaliczyć:

- algorytm Rosenblatta
- algorytm Fishera
- algorytm minimalizacji MSE czyli wykorzystanie regresji liniowej

Algorytm Rosenblatta

Algorytm Rosenblatta zakłada liniową separowalność klas. Stara się on minimalizować funkcję kosztu zdefiniowaną jako:

$$J(\mathbf{w}) = \sum_{\mathbf{x} \in \Omega} (-\mathbf{w}^T \mathbf{x})$$

Gdzie Ω - zbiór wektorów błędnie klasyfikowanych. Algorytm Rosenblatta można opisać w postaci minikodu

t – numer iteracji
w – wektor wag
y_i – etykieta (klasa) wektora $\mathbf{x}_i$
m – liczba wektorów uczących
α – współczynnik uczenia -> funkcja malejąca w czasie np.

$$\alpha = 0.5t/\max\_iter$$

```
t = 0
w = generuj losowo
b = generuj losowo
for t=max_iter  :  1
        for i=1:m
                g = w^T x_i+b;
                if (y_i=1)  and (g<0) =>
                        w=w+α*x_i
                        b = b – α*g;
                if (y_i =-1) and (g>0) =>
                        w=w-α*x_i
                        b = b – α*g;
        end
end
```

Algorytm k-NN

Algorytm k najbliższych sąsiadów jest prostym ale niezwykle ważnym algorytmem. Jego idea działania polega na odnalezieniu w zbiorze danych najbardziej podobnych przypadków i zaklasyfikowaniu ich dokonując głosowania owych k najbliższych przypadków.

Uczenie:

Uczenie polega na zapamiętaniu zbioru uczącego – zarówno zmiennych opisujących jak i wartości przewidywanych

Testowanie:

Testowanie polega na policzeniu odległości od danego wektora testowego do wszystkich wektorów zapamiętanych podczas uczenia

Posortowaniu odległości w celu znalezienie najbardziej podobnych wektorów w zbiorze uczącym

Wybraniu k-najbardziej podobnych wektorów

Dokonaniu głosowania k najbardziej podobnych wektorów

Zliczeniu głosów i wybraniu najczęściej głosowanego przypadku

Problem

Zad 1) Zaimplementuj klasyfikator liniowy Rosenblatta.

W celu realizacji zadania 1 otwórz skrypt skryptLin2D, zawiera on gotowy skrypt matlaba wywołujący funkcję uczącą klasyfikatora liniowego: *klasyfikatorLiniowy* Jego wyniki są przekazywane do funkcji *klasyfikatorLiniowyTest* która dokonuje predykcji.

Do poprawnego uruchomienia skryptu niezbędne jest uzupełnienie funkcji *klasyfikatorLiniowy, klasyfikatorLiniowyTest* – dokonaj niezbędnych modyfikacji tak aby klasyfikator zaczął działac

Wykonując skrypt skryptLin2D zwróć uwagę na sposób modyfikacji działania klasyfikatora. Skrypt nanosi na wykres Figure 1 kolejne uzyskane rozwiązania.

Zwróć uwagę na uzyskiwane dokładności klasyfikatora

Zad 2) Wykonaj skrypt skryptLin. Skrypt ten służy do analizy działania klasyfikatora na realnych problemach. Skrypt pozwala na załadowanie dwóch różnych zbiorów danych (jeden należy zakomentować). Ocena jakości klasyfikatora polega tutaj na uruchomieniu tego samego skryptu na dwóch różnych konfiguracjach podzbiorów zbioru danych. Tzn każdorazowo zbiór danych dzielony jest *v* =10 razy na część treningowa na której klasyfikator jest uczony, oraz część testową która służy ocenie jakości klasyfikatora. Uzyskane wyniki gromadzone są w zmiennej *acc*. Wartość średnia z *acc* obrazuje średnią dokładność klasyfikatora, choć skrypt wyświetla również wartości indywidualne Uruchom skrypt dwukrotnie zmieniając wykorzystywany zbiór danych. Zanotuj uzyskane dokładności.

Co możemy powiedzieć o uzyskanych wynikach?

W szczególności zwróć uwagę na wyniki z poszczególnych iteracji

Zad 3) Uruchom skrypt skryptkNN2D.m słuzy on do wizualizacji działania klasyfikatora kNN. Jest to klasyfikator nieliniowy. Aby móc uruchomić skrypt należy dokończyć implementacji tego algorytmu zawartego w funkcji klasyfikatorkNNTest. Zwróć uwagę na komentarze.

Uruchom i zaobserwuj uzyskane wyniki

Zad 4) Powtórz zadanie 2 dla klasyfikatora kNN, w tym celu wykonaj skrypt *skryptkNN*. Skrypt jest identyczny z zad2 z tą różnica iż wykorzystano inny klasyfikator.

Przydatne funkcje matlaba

y' – transpozycja wektora

*x*y – mnożenie wektorów (uwaga należy pamiętać o wymiarach, gdy x i y są wektorami „poziomymi" wówczas konieczna jest transpozycja x*y')*

rand(n,m) – generuje macierz losową o rozkładzie jednostajnym o wymiarach n x m.

[s,id] = sort(a); - funkcja sortuje wartości w a, a wynik sortowania zapisuje w s. Sortowanie odbywa się od liczby najmniejszej do największej. Zmienna id zawiera indeksy wektorów od najbliższego do najdalszego, tak iż s = a(id);

Y = sign(x) badanie znaku dla x. Jeśli x > 0 => y=1, jeśli x<0 => y=-1

D = Odleglosc2(x,y); - funkcja liczy odległość między wektorem lub wektorami z x, a y, w efekcie jeśli x i y są wektorami wynikiem D będzie liczba, jeśli x jest wektorem a y jest macierzą wówczas D będzie wektorem odległości od x do każdego wiersza z y. Jeśli x i y są macierzami wówczas wyliczona odległość będzie macierzą.