RAFAŁ JAN TRYBUS - 274356

Laboratorium Rozpoznawania Obrazów -
Rozpoznawanie cyfr pisanych ręcznie - klasyfikatory liniowe.

1. WSTĘP
Celem ćwiczenia było stworzenie klasyfikatora do rozpoznawania cyfr pisanych ręcznie za pomocą klasyfikatorów liniowych. Do tego zadania wykorzystano ogólnodostępny zbiór znormalizowanych danych MNIST dostępny pod adresem: http://yann.lecun.com/exdb/mnist.

Klasyfikatory liniowe, zgodnie z nazwą służą do klasyfikacji danych, jedynie liniowo separowalnych. Metody te polegają na wyznaczeniu hiperpłaszczyzny pomiędzy dwoma zbiorami punktów różnych klas. Klasyfikację taką można przeprowadzić na kilka sposobów m.in. wyznaczając granicę (hiperpłaszczyznę) decyzyjną pomiędzy każdą klasą zbioru (OVO: one-vs-one) lub pomiędzy klasą zbioru a pozostałymi klasami (OVR: one-vs-rest).

W ramach laboratorium zbudowano klasyfikatory w oparciu o model OVO oraz OVR.

2. Prace początkowe
W pierwszej kolejności należało zredukować wymiar danych, za pomocą transformacji PCA. Metoda ta polega na znalezieniu takiej osi (lub zestawu osi), dla których punkty pomiarowe są najbardziej na niej rozproszone. Zabieg ten pozwolił na redukcję zbioru uczącego i jednoczesne zachowanie jego charakteru separowalności podzbiorów.

3. Model OVO (one-vs-one)
W pierwszej kolejności podjęto próbę stworzenia klasyfikatora typu 1vs1. Do tego celu napisano funkcję obliczającą parametry płaszczyzny decyzyjnej, wyznaczono je z wykorzystaniem algorytmu uczącego perceptronu o stałej uczenia:

$r = \frac{1}{\sqrt{k}}$, k - numer kroku (iteracji)

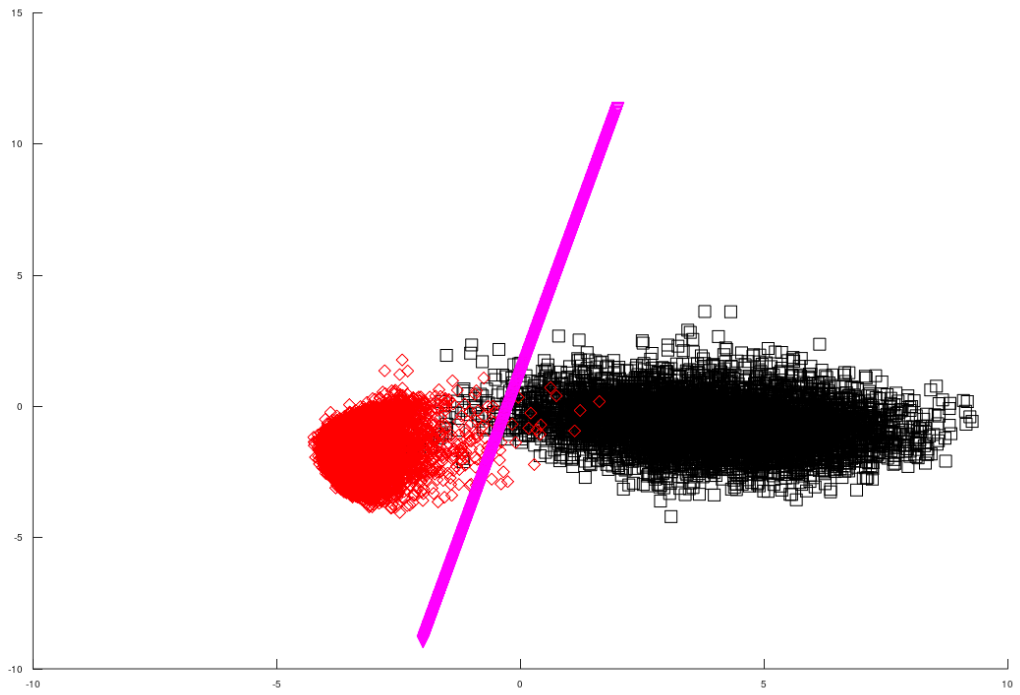
Funkcja ucząca zawarta w pliku perceptron.m ma następującą postać:

```
i = 1;
 do

 res = tset * sepplane' ;
 correct = sum(tset(res<0,:))/i;
 sepplane = sepplane + correct;

    ++i;
 until norm(correct) < 0.001 || i > 20;
```

Przedstawioną wyżej funkcję poddano testowi poprzez wyznaczenie parametrów płaszczyzny decyzyjnej pomiędzy próbkami klasy reprezentującymi cyfry 0 oraz cyfry 1. Poniżej przedstawiono wykres zawierający próbki obu tych klas wraz z wyznaczoną granicą decyzyjną (ograniczono się również do 2 cech w przestrzeni PCA):



Wykres przedstawiający granicę decyzyjną pomiędzy klasą reprezentującą cyfrę ‚0' oraz klasą reprezentującą cyfrę ‚1' (one-vs-one).

Następnie przeprowadzono test na danych wielowymiarowych przyjmując liczbę 40 wymiarów w PCA. W tym celu wytrenowano zestaw 45 klasyfikatorów typu 1vs1. Kolejno wykonano test klasyfikacji poprzez jednomyślne głosowanie klasyfikatorów (tzn. Aby zaklasyfikować daną próbkę do konkretnej klasy, wszystkie klasyfikatory musiały zgodnie ją zaklasyfikować - w przeciwnym wypadku podejmowano decyzję wymijającą). Poniżej przedstawiono tabelę wyniku klasyfikacji dla zbioru uczącego oraz testowego:

- **Zbiór uczący**

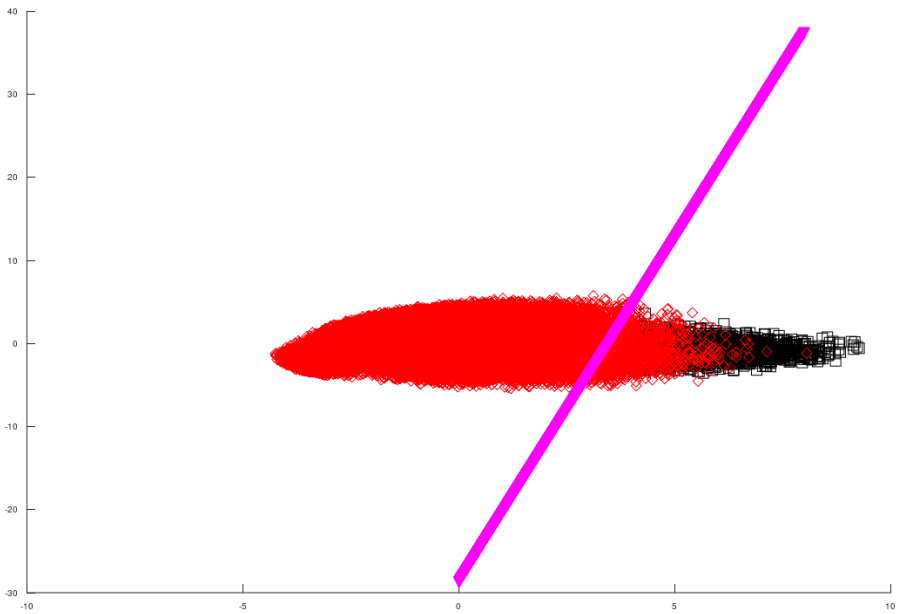
Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
87,0%	9,7%	3,2%

- **Zbiór testowy**

Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
87,7%	9,0%	3,2%

4. Model OVR (one-vs-rest)

Aby dokonać klasyfikacji w modelu OVR poczyniono pewne modyfikacje kodu, względem poprzedniej wersji. Ponownie rozpoczęto od pracy na zbiorze dwóch klas - w tym wypadku klasy reprezentującą cyfrę 0 oraz klasy ‚pozostałe cyfry'. Użyto niezmienionej postaci algorytmu perceptronu do wyznaczenia płaszczyzny decyzyjnej i naniesiono na wykres:



Wykres przedstawiający granicę decyzyjną pomiędzy klasą reprezentującą cyfrę ‚0' oraz klasą reprezentującą wszystkie pozostałe cyfry. (one-vs-rest)

Napisano również funkcję głosującą zawartą w pliku *trainOVRenssemble.m,* jest ona jedynie niewielką modyfikacją wcześniej użytej *trainOVOenssamble.m.* W tym podejściu generuje ona dokładnie 9

klasyfikatorów przyjmujących ‚1' jako etykietę danej klasy oraz ‚-1' dla klasy każdej pozostałej:

```
sets = 0:9;
posSamples = tset(tlab == sets(i), :); % 1 dla danej klasy
    negSamples = tset(tlab ~= sets(i), :); % 0 dla całej reszty
```

Dokonano również niezbędnych modyfikacji w plikach oprogramowujących głosowanie klasyfikatorów tj. *voting.m* oraz *unamvoting.m* . Wspomniane modyfikacje pozwoliły na przeprowadzenie klasyfikacji typu OVR. Otrzymano następujące wyniki:

- **Zbiór uczący**

Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
74,4%	4,6	21,1%

- **Zbiór testowy**

Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
75,3%	4,5	20,2%

5. Zwiększnie liczby cech - funkcja expandFeatures()

W kolejnym kroku zwiększono liczbę cech z wykorzystaniem funkcji expandFeatures(). Funkcja ta tworzy nowe cechy jako iloczyn cech pierwotnych. Zgodnie ze wzorem:
$F'_i = F_j * F_k$, *gdzie F' oznacza nową cechę, F cechę pierwotną. (i <= j).* Z uwagi na wybraną wcześniej liczbę komponentów PCA, c = 40 po zastosowaniu funkcji zbiór poddany transformacji będzie posiadał ich 860, zgodnie ze wzorem:
$c' = c + \frac{c * c + 1}{2}$, *gdzie c' oznacza nową liczbę cech, c pierwotną ich liczbę.*
Po przeprowadzeniu transformacji, ponownie wytrenowano klasyfikatory OVO oraz OVR i przeprowadzono ich głosowanie. W tabeli poniżej przedstawiono wyniki tej klasyfikacji:

- **One-vs-rest (on expandFeature)**

X	Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
Zbiór uczący	80,6%	2,8%	16,5%
Zbiór testowy	81,0%	3,0%	14,9%

- **One-vs-one (on expandFeature)**

X	Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
Zbiór uczący	91,6%	6,4%	1,9%
Zbiór testowy	91,5%	6,5%	2,0%

6. Poprawa jakości klasyfikatora.

W ostatnim kroku podjęto próbę zwiększenia jakości klasyfikatora przedstawionego w poprzednim paragrafie. Zdecydowano się na zastosowanie w tym celu liniowej dyskryminancji Fishera (***FLD - Fisher Linear Discrimination)*** na zestawie danych o rozszerzonych cechach. Dyskryminacja Fishera skutkuje projekcją w której rozróżnienie między klasami jest najprostsze.
Kod funkcji obliczającej FLD został zaczerpnięty z witryny Mathworks i jest dostępny pod adresem:
https://www.mathworks.com/matlabcentral/fileexchange/33768-linear-discriminant-analysis-code

Schemat postępowania był następujący: *{lp - liczba próbek ; lc - liczba cech}*

I. Wczytanie „surowych” danych *{lp = 60k ; lc = 780}*
II. Przeprowadzenie transformacji PCA z wykorzystaniem 40 komponentów. *{lp = 60k ; lc = 40}*
III. Losowe 6x pomniejszenie zbioru uczącego. *{lp = 10k ; lc = 40}*
IV. Przeprowadzenie transformacji rozszerzającej liczbę cech - expandFeature. *{lp = 10k ; lc = 230}*
V. Przeprowadzenie transformacji Fishera *{lp = 10k ; lc = 230}*
VI. Uczenie klasyfikatorów.

Wybór liczby 40 komponentów w przestrzeni PCA (punkt II), zdał się być najlepszym balansem pomiędzy wynikającymi z jego wartości czasów kolejnych transformacji (expandFeature, FLD) oraz wymierną poprawą klasyfikacji. Z tego samego powodu dokonano również losowego zmniejszenia zbioru uczącego. Poniżej przedstawiono wyniki klasyfikacji po zastosowaniu wyżej opisanej metody:

- **One-vs-rest (on expandFeature)**

X	Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
Zbiór uczący	84,1%	1,2%	14,1%
Zbiór testowy	81,3%	2,0%	15,9%

- **One-vs-one (on expandFeature)**

X	Poprawna klasyfikacja	Błędna klasyfikacja	Odpowiedź wymijająca
Zbiór uczący	96,7%	2,6%	0,6%
Zbiór testowy	94,0%	4,9%	0,9%

Analiza powyższej tabeli wskazuje na wzrost jakości klasyfikatorów, można zatem stwierdzić, że użycie przedstawionej metody jest zasadne i uznać tak nauczony klasyfikator za produkt ostateczny.