

<b>UNIwersytet Zielonogórski</b>	<b>WYKONALI:</b>	<b>GRUPA</b>		<b>OCENA:</b>
<i>Rozpoznawanie obrazów</i>				
<b>ĆWICZENIE NUMER: 8</b>	<b>TEMAT:</b> Algorytm KNN	<b>DATA WYKONANIA:</b>	<b>DATA ODDANIA:</b>	<b>PODPIS:</b>

## 1. Cel ćwiczenia

Celem ćwiczenia jest zaznajomienie się ze skutecznością klasyfikowania obiektów przy pomocy algorytmu „k najbliższych sąsiadów”. Zadanie nasze będzie polegało na rozpoznawaniu wcześniej przygotowanych znaków alfabetu za pomocą algorytmu KNN.

## 2. Klasyfikacja znaków za pomocą algorytmu KNN.

Sprawdzimy jak działa metoda KNN na podstawie zadania polegającego na zaklasyfikowaniu nowego obiektu (graficzny znak alfabetu) spośród pewnej liczby obiektów znanych. W tym celu skorzystamy z przygotowanych wcześniej zestawów liter różnych alfabetów. Ponieważ posiadamy aż 116 różnych alfabetów, ich część przeznaczymy na uczenie sieci, a pozostałe znaki będziemy wykorzystywali do testowania zaproponowanego rozwiązania. Tak rozwiązanie pozwoli nam stwierdzić czy program będzie miał dobre właściwości uogólniające, podczas badań zbadamy też jak szybko działa metoda klasyfikacji przy użyciu algorytmu KNN, oraz postaramy się oszacować jak wiele pamięci operacyjnej wymaga.

Pierwszym krokiem będzie stworzenie części programu odpowiedzialnego za uczenie. Postanowiliśmy wykorzystać 70 pierwszych alfabetów do nauki, a pozostałą część przeznaczyć do testowania. Kod napisany w MATLAB-ie który realizuje naukę, wraz z komentarzami zamieszczamy poniżej.

```

clc;
clear all;
path = '\literki\';
tab_danych = [];
alfabet = [];
%pętla przeskakiwania po kolejnych znakach alfabetu
for i = 0:25
    sign = char('a' + i);
    dir = strcat(strcat(path,sign),'\');
    %pętla przeskakiwania po kolejnych modyfikacjach danego znaku
    for j=1:70
        path = '\literki\';
        %kiedy nazwa pliku ma numerację poniżej 10
        if j < 10
            name = strcat(sign,'_00',int2str(j),'.bmp');
        end
    end
end

```

```
    full_path = strcat(dir,name);
end
%kiedy nazwa pliku ma numeracje powyżej 10
if j >= 10
    name = strcat(sign,'_0',int2str(j),'.bmp');
    full_path = strcat(dir,name);
end
%odczyt danych z pliku graficznego
obraz = imread(full_path,'bmp');
obraz = obraz(:);
%wczytanie danych do tablicy
tab_danych = [tab_danych;obraz];
alfabet = [alfabet;sign];
end
end
```

Kiedy posiadamy już część programu która zapamiętała część wzorców możemy przejść dalej. W tym kroku napiszemy kod który z pozostałych 46 alfabetów, wylosuje 1000 znaków i zapisze je jako zbiór testowy. Jak wcześniej kod odpowiedzialny za realizację tego zadani zamieszczamy poniżej.

```
path = '\literki\';
%wylosowanie 1000 znaków z których kazdy składa sie z 576 pixeli
zb_t = zeros(1000,576);
alfabet_t = [];
%pętla losowania kolejnego znaku do zbioru testującego
for i = 1:1000
    %wylosowanie znaku alfabetu
    sign = char('a'+round(rand(1)*25));
    %wylosowanie numeru znaku
    sign_number = 70+round(rand(1)*46);
    %przekształcenie numeru do łańcucha znaków
    ktory_znak = int2str(sign_number);
    %dodanie cyfr według wartości
    if sign_number < 100,
        ktory_znak = strcat('0',ktory_znak);
    end
    %ustawienie pełnej ścieżki to wylosowanego pliku
    dir = strcat(path,sign,'\');
    name = strcat(sign,'_',ktory_znak,'.bmp');
    full_path = strcat(dir,name);
    %wczytanie pliku graficznego
    obraz = imread(full_path);
    obraz = obraz(:);
    obraz = double(obraz);
    zb_t(i,:) = obraz;
    alfabet_t = [alfabet_t;sign];
end
```

Teraz możemy przejść do napisania kodu klasyfikatora. W tym momencie musimy zdecydować się na zastosowanie jednej z wielu miar odległościowych. Na zajęciach zdecydowaliśmy się na miarę Czebyszewa, ale po implementacji jej w naszym programie doszliśmy do wniosku że nie nadaje się ona zupełnie do tego zadania. Miara Czebyszewa bierze pod uwagę tylko maksymalne wartości, a przy takich wzorcach wejściowych jak dostaliśmy do ręki w każdym przypadku miara maksymalna wyszła nam 228. Wynika to z tego iż literki w kolejnych obrazkach są nieco przesunięte lub posiadają inny kształt co pociąga za sobą fakt iż zawsze znajdzie się piksel który w jednym obrazie będzie posiadał wartość 255 (kolor biały) a w drugim 46 (kolor czarny) co w wyniku odejmowania da 228. Z tego powodu zdecydowaliśmy się na zastosowanie innej miary, a konkretnie kwadratu odległości euklidesowej.

$$\text{odległość}(x,y) = \sum_i (x_i - y_i)^2$$

Miara ta nie posiada wady poprzedniej metody.

Poniżej zamieszczamy część kodu odpowiedzialną za klasyfikację. Pozostawiliśmy jednak miarę Czebyszewa, ale daliśmy ją w komentarz.

```
clc;
sukces=0; porazka=0;
%przez S oznaczmy liczbę sąsiadów
S=3;
%petla sprawdzania kolejnych znaków
for i=1:1000,
    %odczyt pierwszego znaku testowego
    obraz = zb_t(i,:);
    %przekształcenie zbioru uczacego do double
    tab_danych = double(tab_danych);
    [X,Y]=size(tab_danych);
    test = zeros(X,Y);
    %zapisanie do t w kazde miejsce wartosci literki testowanej
    for j=1:X, test(j,:)=obraz; end
    %zastosowanie miary czebyszewa
    %buf = abs(tab_danych-test)';
    %odleglosci = [];
    %for m=1:1000
    % maximum=0;
    % for n=1:576
    %     if buf(n,m)>maximum
    %         maximum=buf(n,m);
    %     end
    % end
    % odleglosci = [odleglosci, maximum];
    %end
    %zastosowanie kwadratowej odległości euklidesowej
    odleglosci = sum(((tab_danych-test).^2)');
    %posortowanie według najmniejszej odległości
    [odleglosci,index] = sort(odleglosci);
    ulozyny_alfabet = alfabet(index);
```

```
%wybór tego znaku, którego wystapień było najwięcej
max=1; ktory=1;
for k=1:S
    wybrany=ulozyny_alfabet(k);
    ile=0;
    for l=1:S
        if wybrany==ulozyny_alfabet(l)
            ile = ile+1;
        end
    end
    if ile>max
        max = ile;
        ktory = k;
    end
end
wybrany = ulozyny_alfabet(ktory);
%sprawdzenie czy wybrany zwyczajem jest poprawnie skojarzonym znakiem
if wybrany == alfabet_t(i) sukces = sukces+1; end
if wybrany ~= alfabet_t(i) porazka = porazka+1; end
end
```

Badania przeprowadzimy dla ilości sąsiadów: 1, 3, 5, 15, 31, 51.  
Wyniki Badań zamieszczamy w tabeli poniżej.

Ilość sąsiadów	Poprawnie rozpoznanych znaków	Niepoprawnie rozpoznanych znaków	Wynik procentowy poprawnie rozpoznanych znaków
1	789	211	78,9 %
3	790	210	79 %
5	786	214	78,6 %
15	774	226	77,4 %
31	716	284	71,6 %
51	647	353	64,7 %

### 3. WNIOSKI

Z przeprowadzonych badań wynika że metoda ta nie daje najlepszych rezultatów ponieważ rozpoznawalność na poziomi niecałych 80% nie jest dużym gwarantem sukcesu. Wynik jest ściśle zależny od zastosowanej miary. W naszym przypadku, przy obraniu kwadratowej odległości euklidesowej wynik wydaje się być dość stabilny i dopiero przy zmianie liczby sąsiadów powyżej 15, odbiega od normy. Kwadratowa miara odległości ma to do siebie, że podnosi wartość do kwadratu, aby przypisać większą wagę obiektom, które są bardziej oddalone. Właśnie ta cecha decyduje o stabilności tej metody. Metoda k najbliższych sąsiadów stosuje pamięciowy model zdefiniowany przez zestaw przykładowych wzorców. Wynika stąd iż posiada ona ogromne zapotrzebowanie pamięciowe, aby działała w zadowalającym stopniu. Osobnym problemem jest czas potrzebny na działanie algorytmu. O ile uczenie programu przebiegało względnie szybko, to znacznie gorzej było już przy klasyfikowaniu wzorców.

Co prawda w naszym programie klasyfikowaliśmy 1000 znaków jednorazowo, jednak jeśli chcielibyśmy na podstawie tego algorytmu zbudować program do rozpoznawania znaków, to 1000 sztuk jest bardzo małą ilością. Ta wada kompletnie dyskwalifikuje tą metodę do rozpoznawania znaków alfanumerycznych.