**Maciej Oziębły**

**Nr indeksu: 184147**

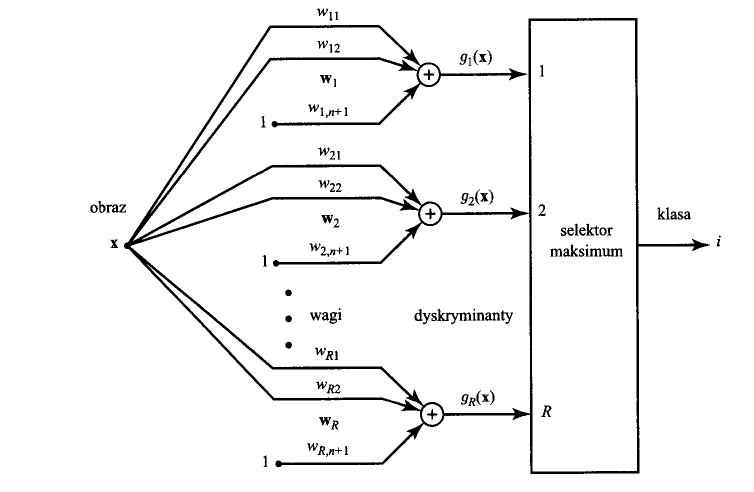
Programowanie sieciowe

Sprawozdanie 1

Zakres sprawozdania:

* liniowe sieci neuronowe
* wykorzystanie klasyfikatora minimalnoodległościowego
* rozpoznawanie liter

## Stworzenie sieci neuronowej

W celu stworzenia liniowej sieci neuronowej do do rozpoznawania znaków wykorzystałem następujący schemat:  


Opierając się na schemacie i teorii z zajęć stworzyłem sieć za pomocą kodu w matlabie („letter\_recogniser.m”):

function [ recognised\_letter ] = letter\_recogniser( input )

%% load letters

letters = load\_letters\_definitions();

%% weight calculation

w\_n\_plus\_1 = -1\*sum(letters'.\*letters',2)/2;

%% dicrimination foo vector calculation

discrimination = letters' \* input + w\_n\_plus\_1;

%% find and return best match (bigest val index in discrimination vector)

recognised\_letter = find(ismember(discrimination, max(discrimination)));

end

Podając na wejściu funkcji literę w formie wektora 100 wartości dostajemy na wyjściu numer litery z bezy danych. Jako bazę danych wykorzystałem funkcję „load\_letters\_definitions.m”, w której znajdują się wektory zawierający wygląd wszystkich kolejnych liter

## Testy:

W celu przetestowania działania sieci stworzyłem skrypt „testerka.m”:

letters = load\_letters\_definitions();

tries\_per\_letter = 300;

avg\_tries\_per\_letter = zeros(size(letters,2),1);

for letter\_no = 1:size(letters,2)

tries = 0;

letter\_no

for i = 1:tries\_per\_letter

letter = letters(:, letter\_no);

random\_changes\_before\_fail = 0;

match\_failed = 0;

changed\_indexes = zeros(200,1);

while(match\_failed == 0)

if(letter\_recogniser(letter) ~= letter\_no)

match\_failed = 1;

else

% generate random index not changed yet

rand\_index = randi(100);

while(max(ismember(changed\_indexes, rand\_index)) ~= 0)

rand\_index = randi(100);

end

changed\_indexes(length(changed\_indexes)+1) = rand\_index;

letter(rand\_index) = mod(letter(rand\_index)+1,2);

random\_changes\_before\_fail = random\_changes\_before\_fail+1;

end

end

tries = tries + random\_changes\_before\_fail;

end

avg\_tries\_per\_letter(letter\_no) = tries / tries\_per\_letter;

end

stem(avg\_tries\_per\_letter)

title('average random changes on picture needed for recogniser to match incorrectly')

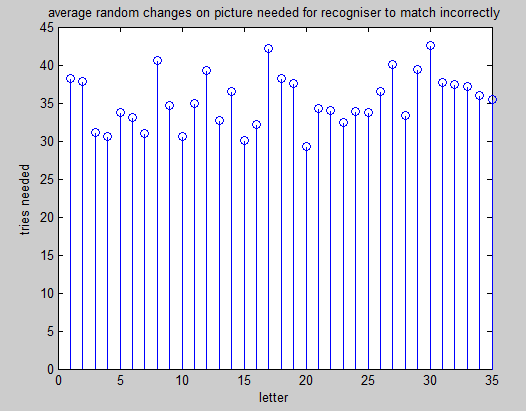
xlabel('letter')

ylabel('tries needed')

Skrypt ten wrzuca po kolei każdą z liter do sieci rozpoznającej znaki, a następnie zmienia 1 losowy piksel (nie zmieniony wcześniej) na jego przeciwieństwo. Zmiany trwają tak długo, aż sieć popełni błąd w rozpoznawianiu znaku. Dla każdej litery wykonane jest 300 prób, a następnie jest wyciągany z nich średnia ilość zmian potrzebnych do wystąpienia błędu.

## Wyniki testów:

Używając swojego skryptu testującego wygenerowałem wykres średniej ilości losowych zmian potrzebnych, do wystąpienia błędu w sieci rozpoznającej znaki:



Liczba zmian potrzebnych do wywołania błędu mieści się w zakresie [30,44] więc rozrzut wynosi   
ok 20%, a sam zakres zawiera zaskakująco wysokie wartości (średnio trzeba zmienić ponad 1/3 obrazka, aby algorytm rozpoznający zawiódł).

## Wnioski:

* Liniowa sieć neuronowa polegająca na klasyfikatorze minimanoodległościowym służąca do rozpoznawania znaków jest bardzo prosta do zaimplementowania – w moim przypadku zajmuje tylko 3 linię kodu w Matlabie (a możliwe jest zaposanie jej w 1 linii)
* Zastosowanie takiej sieci neuronowej daje bardzo dobrze wyniki – średnio trzeba zmienić ponad 1/3 obrazka, aby sieć dała zły wynik. Warto zauważyć, że po takiej ilości zmian rozpoznanie litery jest już bardzo trudne dla człowieka.