Maciej Oziębły
Nr indeksu: 184147

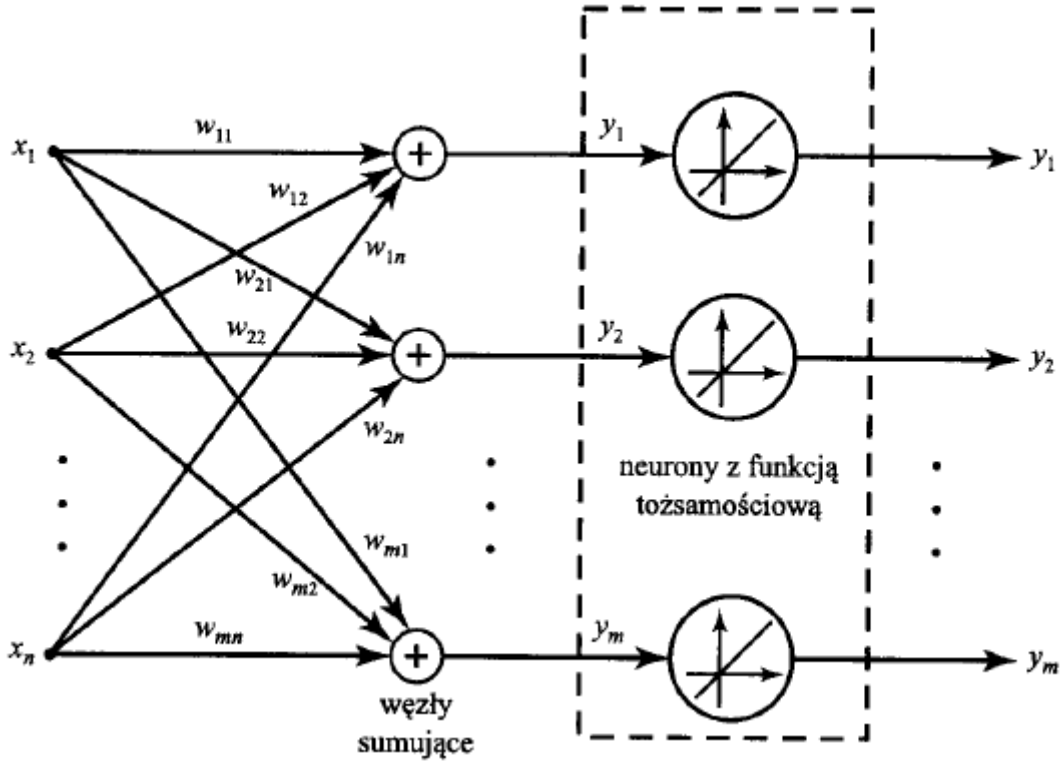
Programowanie sieciowe
Sprawozdanie 2

Zakres sprawozdania:

- asocjacyjne sieci neuronowe
- rozpoznawanie małych i wielkich 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 („association_recogniser.m” i „`calc_weights_matrix.m`”):

```
function [ recognised_letter] = association_recogniser( letter )

    % get output vector
    y = calc_weights_matrix(load_letters_definitions) * letter;

    % find and return best match (bigest val index in discrimination
vector)
    recognised_letter = find(ismember(y, max(y)));

end

function [ weights_matrix] = calc_weights_matrix( letters )
    % F matrix is eye matrix for now coz i have
    % just 1 pattern for each letter
    F = eye(35);

    % addding small letters (from 36 to 70)
    % there is exactly 1 small letter for each capital letter
    F = [F;eye(35)];

    weights_matrix = F' * letters';
end
```

Ponieważ algorytm ten liczył macierze wag za każdym razem, kiedy chciał rozpoznać literę postanowiłem napisać też przyspieszony algorytm, w którym macierz wag podajemy z zewnątrz (znacznie przyspieszyło to program przy rozpoznawaniu dużej ilości znaków):

```
function [ recognised_letter] = association_recogniser_optimized( letter , weights_matrix)

    % get output vector
    y = weights_matrix * letter;

    % find and return best match
    recognised_letter = find(ismember(y, max(y)));

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:

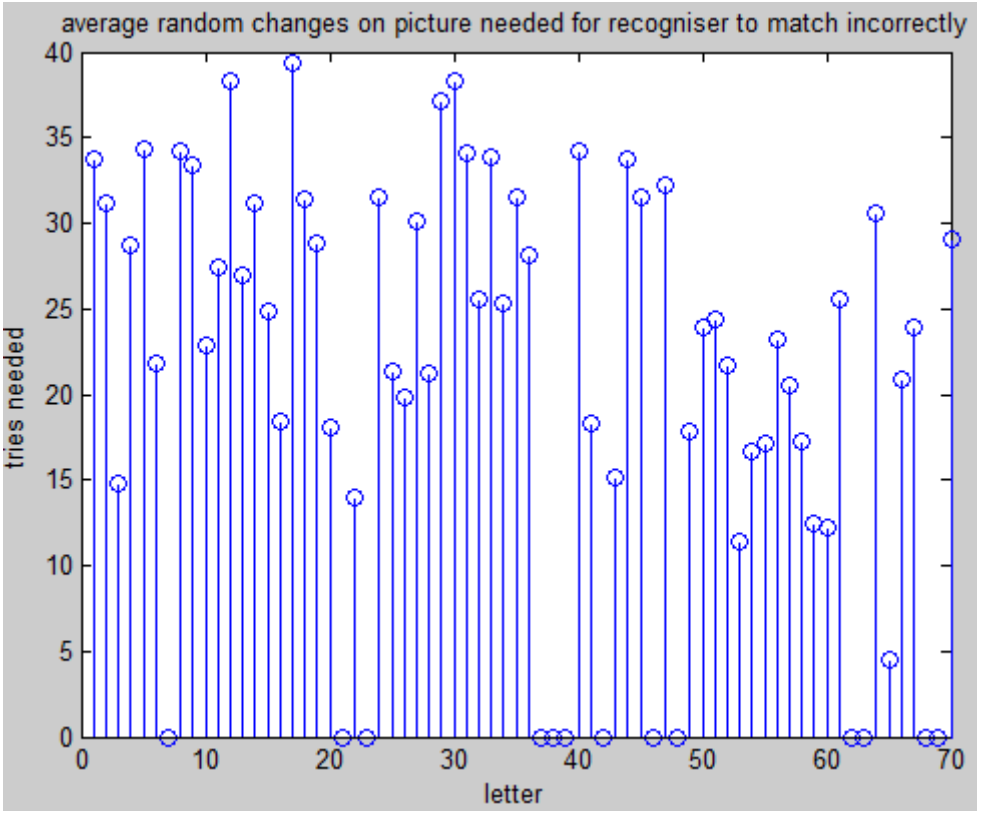
W celu przetestowania działania sieci stworzyłem skrypt „testerka.m”:

```
letters = load_letters_definitions();


letters = load_letters_definitions();


tries_per_letter = 300;
avg_tries_per_letter = zeros(size(letters,2),1);
weights = calc_weights_matrix(load_letters_definitions);
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(100,1);
        while(match_failed == 0 && random_changes_before_fail ~= 100)
            %change 36 -> 1, 2 -> 2, 70 -> 35 etc...
            if(association_recogniser_optimized(letter, weights) ~=
mod((letter_no-1), 35)+1)
            %if(association_recogniser(letter) ~= mod((letter_no-1), 35)+1)
                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) = letter(rand_index) * -1;
                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.

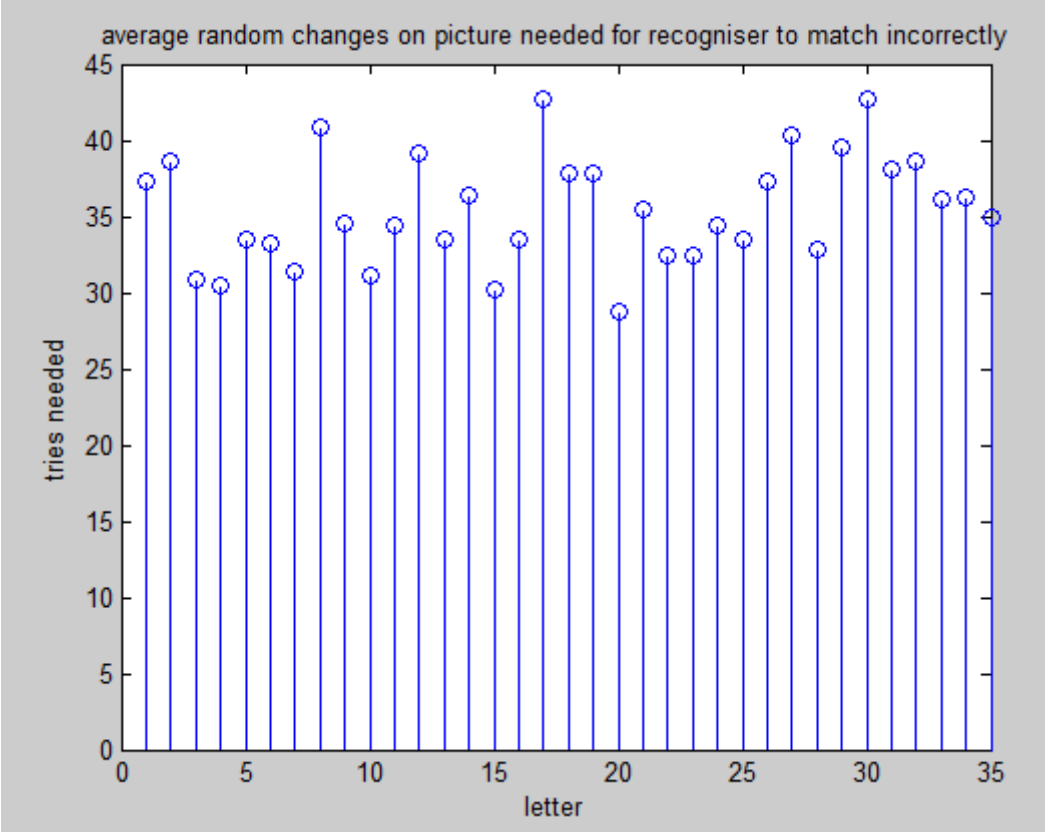
Wyniki testów:

Używając swojego skryptu testującego wygenerowałem wykres średniej ilości losowych zmian potrzebnych, do wystąpienia błędu w sieci rozpoznającej znaki (znakli dla indeksów >35 to małe litery odpowiadające wielkim literom o indeksie mniejszym o 35 od nich).



Liczba zmian potrzebnych do wywołania błędu mieści się w zakresie [0, 40]. Algorytm nie radzi sobie z rozpoznawaniem niektórych znaków, co wynika z tego, że niektóre znaki są do siebie bardzo podobne.

Dla porównania przeprowadziłem testy bez małych liter:



Tutaj wyniki są o wiele lepsze – warto jednak zauważyć, że dla takich danych wejściowych sieć ta działa niemal identycznie jak liniowa sieć neuronowa z poprzedniego zadania.

Wnioski:

Asocjacyjna sieć neuronowa radzi sobie bardzo dobrze z rozpoznawaniem wielkich liter, jednak po dodaniu do bazy wiedzy małych liter algorytm zaczął działać o wiele gorzej i często popełniać błędy – w wielu przypadkach nie rozpoznawał poprawnie litery, która była wzorcem. Działo się tak w przypadkach, kiedy mała litera bardzo różniła się od wielkiej litery, ponieważ w takim przypadku macierz wag dla tej asocjacji była bardzo zaszumiona. W przypadku posiadania jeszcze większej ilości wzorców dla każdej asocjacji własność ta pogłębiłaby się w znacznym stopniu przez co nie uważam sieci tego typu za odpowiednie do stosowania w praktyce.