Gliwice, 08.09.2017r.

Instytut Informatyki Politechniki Śląskiej

Rok akademicki	Rodzaj studiów*: SSI/NSI/NSM	Przedmiot: (Języki Asemblerowe/SMIW)	Grupa	Sekcja
2016/2017	SSI	**BIAI**	**GKiO3**	**2**
Prowadzący przedmiot:	dr inż. Grzegorz Baron		**Termin: (dzień tygodnia godzina)**	
Imię:	**Kacper**		**Wtorek**	
Nazwisko:	**Kowalik**		**11:45-13:15**	
Email:	kacpkow521@student.polsl.pl			

Raport końcowy

Temat projektu:

System klasyfikowania grzybów

Link do repozytorium github

Pod poniżej podanym linkiem znaleźć można pliki z kompletną dokumentacją, pliki źródłowe, dane testowe uporządkowane wg instrukcji zamieszczonej na stronie ZMITAC.

https://github.com/kacpkow/Mushroom-neural-network

Założenia projektowe:

1. System, który dla danych wejściowych opisujących wybrany grzyb ma stwierdzić, czy dany grzyb jest grzybem jadalnym lub trującym.
2. Baza danych opisująca grzyby zawiera ponad 8000 rekordów, a każdy rekord składa się z 23 atrybutów.
3. Projekt będzie obejmował implementację własnej sieci neuronowej oraz przeprowadzenie szeregu testów służących analizie napisanego klasyfikatora.
4. Aplikacja zostanie napisana w języku JAVA.

Realizacja założonych celów

1. Aplikacja została napisana przeze mnie w języku JAVA, sieć neuronowa została stworzona przeze mnie od podstaw.
2. Całość funkcjonalna składa się z dwóch części:
    - Programu serwera
    - Programu klienta
3. Na potrzeby testowania sieci neuronowej zostały stworzone 2 wersje aplikacji serwera:
    1) I wersja z następującą strukturą sieci neuronowej warstwa wejściowa – I warstwa ukryta – warstwa wyjściowa
    2) II wersja z następującą strukturą sieci neuronowej warstwa wejściowa – I warstwa ukryta – II warstwa ukryta - warstwa wyjściowa
4. Do nauczenia sieci neuronowej wykorzystałem algorytm wstecznej propagacji błędów ze współczynnikiem momentum.

Baza danych

Jako baza danych została wykorzystana baza mushrooms.csv pobrana z serwisu kaggle.com.

Każdy rekord bazy zawiera atrybut świadczący o klasyfikacji grzyba przyjmujący wartość edible(e) dla grzyba jadalnego i wartość poisonous(p) dla grzyba trującego. Rekord oprócz tego zawiera jeszcze 22 cechy grzyba opisane przez następujące atrybuty(niestety nie zostały przeze mnie przetłumaczone na j. polski):

Attribute Information: (classes: edible=e, poisonous=p)

- cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
- cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
- cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r,pink=p,purple=u,red=e,white=w,yellow=y
- bruises: bruises=t,no=f
- odor: almond=a,anise=l,creosote=c,fishy=y,foul=f,musty=m,none=n,pungent=p,spicy=s
- gill-attachment: attached=a,descending=d,free=f,notched=n
- gill-spacing: close=c,crowded=w,distant=d
- gill-size: broad=b,narrow=n
- gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e,white=w,yellow=y
- stalk-shape: enlarging=e,tapering=t
- stalk-root: bulbous=b,club=c,cup=u,equal=e,rhizomorphs=z,rooted=r,missing=?
- stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
- stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o,pink=p,red=e,white=w,yellow=y
- veil-type: partial=p,universal=u
- veil-color: brown=n,orange=o,white=w,yellow=y
- ring-number: none=n,one=o,two=t
- ring-type: cobwebby=c,evanescent=e,flaring=f,large=l,none=n,pendant=p,sheathing=s,zone=z
- spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r,orange=o,purple=u,white=w,yellow=y
- population: abundant=a,clustered=c,numerous=n,scattered=s,several=v,solitary=y
- habitat: grasses=g,leaves=l,meadows=m,paths=p,urban=u,waste=w,woods=d

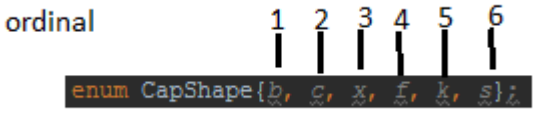
	A	B	C	D	E	F	G
1	class,cap-shape,cap-surface,cap-color,bruises,odor,gill-attachment,gill-sp						
2	p,x,s,n,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,k,s,u						
3	e,x,s,y,t,a,f,c,b,k,e,c,s,s,w,w,p,w,o,p,n,n,g						
4	e,b,s,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,n,m						
5	p,x,y,w,t,p,f,c,n,n,e,e,s,s,w,w,p,w,o,p,k,s,u						
6	e,x,s,g,f,n,f,w,b,k,t,e,s,s,w,w,p,w,o,e,n,a,g						
7	e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,n,g						
8	e,b,s,w,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,n,m						
9	e,b,y,w,t,l,f,c,b,n,e,c,s,s,w,w,p,w,o,p,n,s,m						
10	p,x,y,w,t,p,f,c,n,p,e,e,s,s,w,w,p,w,o,p,k,v,g						
11	e,b,s,y,t,a,f,c,b,g,e,c,s,s,w,w,p,w,o,p,k,s,m						
12	e,x,y,y,t,l,f,c,b,g,e,c,s,s,w,w,p,w,o,p,n,n,g						
13	e,x,y,y,t,a,f,c,b,n,e,c,s,s,w,w,p,w,o,p,k,s,m						
14	e,b,s,y,t,a,f,c,b,w,e,c,s,s,w,w,p,w,o,p,n,s,g						
15	p,x,y,w,t,p,f,c,n,k,e,e,s,s,w,w,p,w,o,p,n,v,u						
16	e,x,f,n,f,n,f,w,b,n,t,e,s,f,w,w,p,w,o,e,k,a,g						

Powyżej zaprezentowany został fragment bazy danych.

Plik z bazą danych pobrany ze strony kaggle.com zawiera ponad 8000 rekordów, jednakże znaczna część obarczona jest brakiem reprezentacji niektórych cech grzyba (niektóre pola rekordów mają wartość ‘’?”). Rekordy te zostały przeze mnie odrzucone. W procesie uczenia sieci neuronowej używałem dane testowe składające się z 900 rekordów (450 rekordów zawierających grzyby sklasyfikowane jako jadalne i 450 jako trujące) zapisane w pliku ‘’mushrooms11.csv’’ (znajdujące się w repozytorium github w folderze Plik_exe/Dane_testowe). W procesie testowania sieci neuronowej używałem pliku .csv z rekordami, które nie brały udziału w procesie uczenia. Plik ten zawierał 300 rekordów (150 rekordów opisuje grzyby jadalne, 150 grzyby trujące). Plik ten również dołączam do archiwum .zip projektu (plik ‘’mushrooms2.csv’’).

Mapowanie danych wektora wejściowego

```
enum CapShape{b, c, x, f, k, s};
private CapShape capShape;
public Integer getCapShapeOrdinalValue() { return capShape.ordinal(); }
```



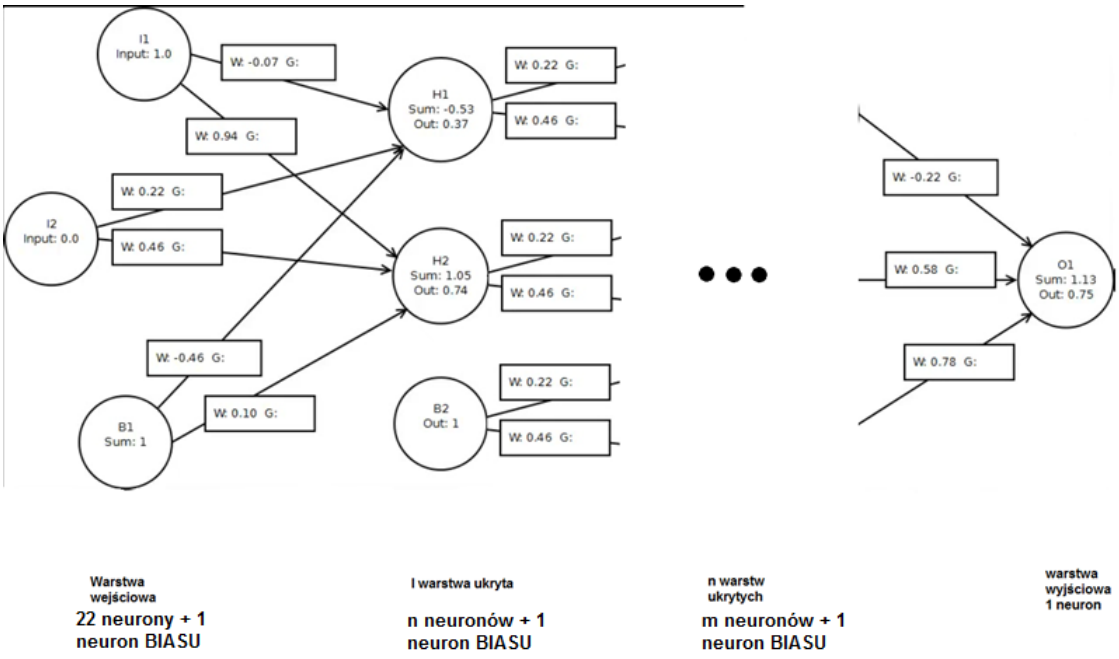
Dla przykładu powyższa funkcja ‘’getCapShapeOrdinalValue’’ zwróci numer porządkowy dla ustawionego parametru Cap Shape. Przykładowo, jeżeli parametr capShape będzie elementem c typu wyliczeniowego CapShape to funkcja zwróci wartość: 2. Następnie zanim wartość dla neurona wejściowego zostanie ustawiona zostanie poddana normalizacji min-max opisanej wzorem:

1. Normalizacja min-max

Ta metoda przeprowadza liniową transformację pierwotnych danych najczęściej do przedziału [0,1] według wzoru:

$$V' = \frac{(V - \min)}{\max - \min} * (new\_\max - new\_\min) + new\_\min$$

Struktura napisanej sieci neuronowej



Sieć neuronowa napisana przeze mnie występuje w dwóch wersjach:

- warstwa 0: 22 neurony wejściowe + 1 neuron BIASu - warstwa 1 ukryta: n neuronów wejściowych + 1 neuron BIASu – warstwa 2 wyjściowa: 1 neuron
- warstwa 0: 22 neurony wejściowe + 1 neuron BIASu - warstwa 1 ukryta: n neuronów wejściowych + 1 neuron BIASu – warstwa 2 ukryta: m neuronów wejściowych + 1 neuron BIASu – warstwa 2 wyjściowa: 1 neuron

Są to 2 odmienne aplikacje programu serwera, pierwszy wariant znajduje się w folderze Mushroom_server_1hidden_layer. Drugi wariant znajduje się w folderze Mushroom_server_2hidden_layers.

Algorytm backpropagation z momentum

Sztuczna sieć neuronowa zaimplementowana w programie korzysta z wstecznej propagacji błędów z uwzględnieniem współczynnika momentum.

Klasyczny algorytm wstecznej propagacji błędów modyfikuje wagę w_{kj} o wartość (Δw_{kj}) , która jest proporcjonalna do pochodnej cząstkowej funkcji celu (wagi są modyfikowane zgodnie z regułą delty):

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} = \eta \delta_k y_j$$

Gdzie η jest parametrem nauczania algorytmu wstecznej propagacji błędów.

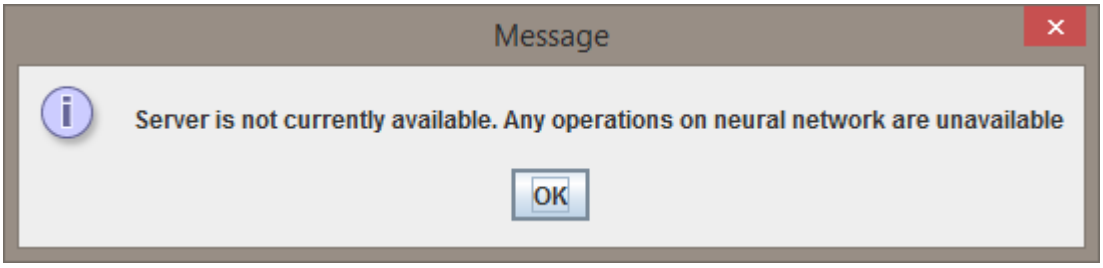
Jednym z rozwiązań umożliwiających bezpieczne zwiększenie efektywnego tempa uczenia sieci bez pogarszania stabilności procesu jest zastosowanie momentowej metody wstecznej propagacji błędu (ang. Momentum BackPropagation). Istotą metody jest wprowadzenie do procesu uaktualniania wagi pewnej bezwładności tzw. "momentu", proporcjonalnego do zmiany tej wagi w poprzedniej iteracji:

$$w_{ji}(t+1) = w_{ji}(t) + \eta \delta_j(t) y_i(t) + \underbrace{\mu\left(w_{ji}(t) - w_{ji}(t-1)\right)}_{\text{Momentum}}$$

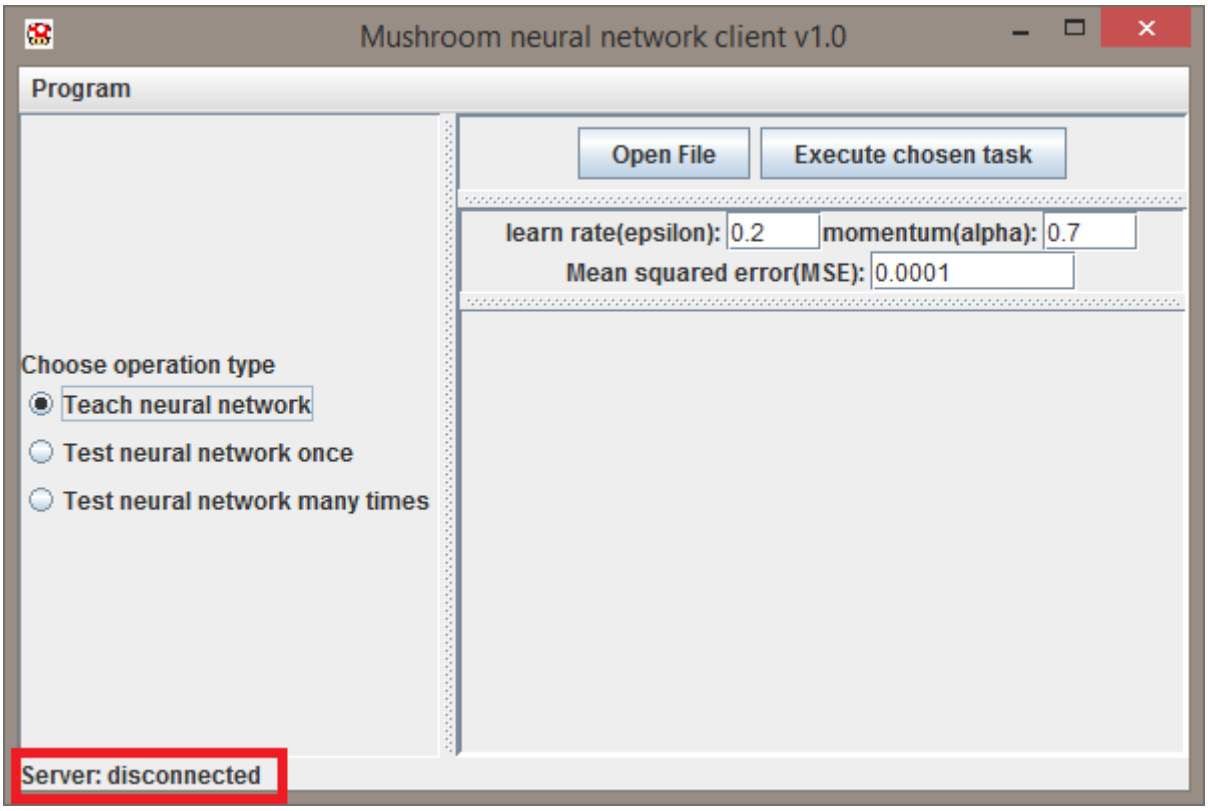
gdzie $\mu \in (-0,1]$ - współczynnik momentu.
Pierwsze dwa składniki po prawej stronie wyrażenia są analogiczne, jak w klasycznej metodzie BP, natomiast ostatni uwzględnia poprzednią zmianę wagi i jest niezależny od aktualnej wartości gradientu.

Instrukcja obsługi programu

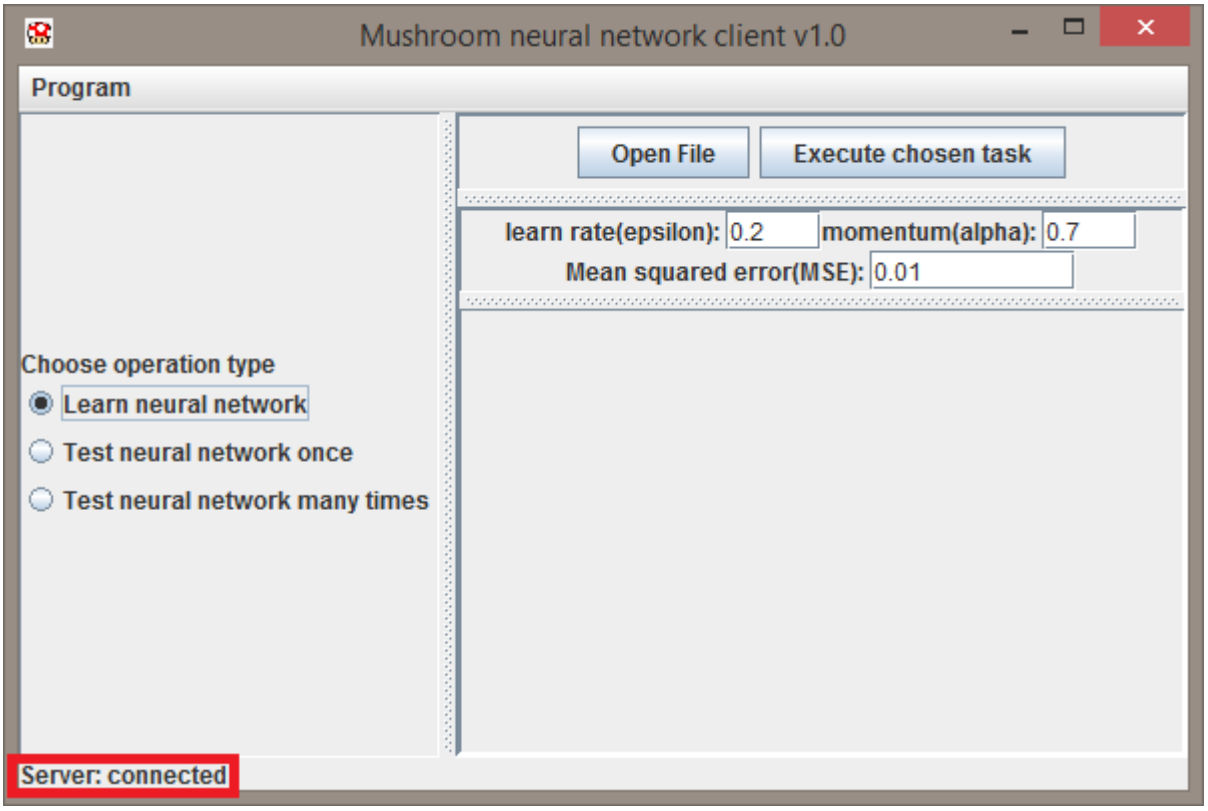
1. Uruchomić wybraną wersję serwera drogą kompilacji projektu lub uruchomienie pliku wykonywalnego z rozszerzeniem .jar znajdującego się w ścieżce ''Plik_exe/Mushrooms_server_1hidden_layer" lub ''Plik_exe/Mushrooms_server_2hidden_layers"
2. Uruchomić program klienta drogą kompilacji projektu lub uruchomienie pliku wykonywalnego z rozszerzeniem .jar znajdującego się w ścieżce ''Plik_exe/Mushroom_client''
3. Po uruchomieniu klienta w przypadku braku wykrycia połączenia z aplikacją serwera zostanie wyświetlony komunikat:



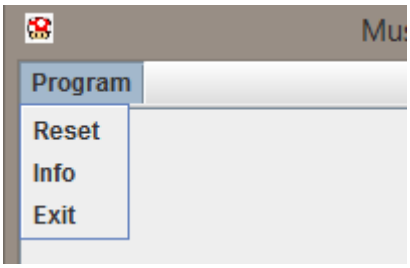
Następnie program klienta uruchomi się ale będzie niezdolny do wykonania wszelkich operacji na sieci neuronowej. Brak połączenia sygnalizuje napis ''Server:disconnected"



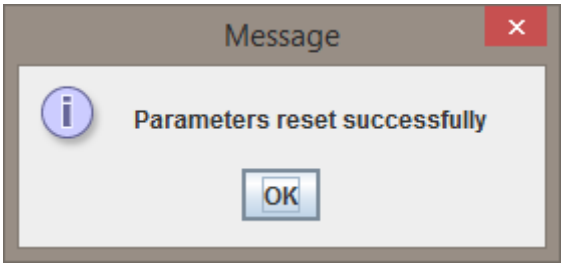
4. W przypadku wykrycia połączenia z serwerem program wyświetli status ‘’Server:connected’’



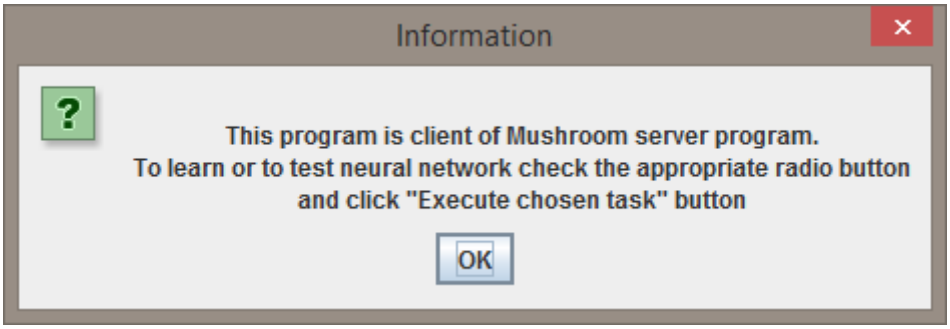
5. Użytkownik ma do wyboru z menu kontekstowego następujące opcje:



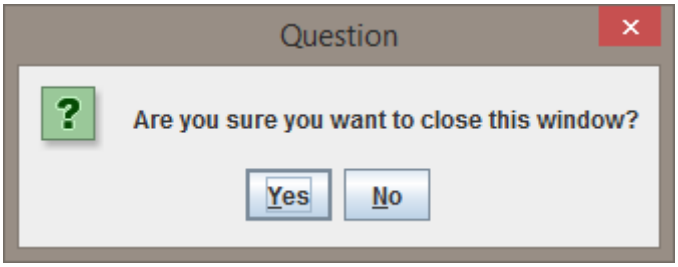
- Opcja ‘’Reset’’ – resetuje sieć neuronową na serwerze i powoduje zainicjalizowanie nowej sieci neuronowej z losowo dobranymi wagami poszczególnych węzłów sieci neuronowej. Po kliknięciu użytkownik zostanie poproszony o potwierdzenie wybranej opcji w oknie dialogowym. Po pomyślnym zresetowaniu sieci neuronowej użytkownikowi zostanie wyświetlone okno dialogowe:



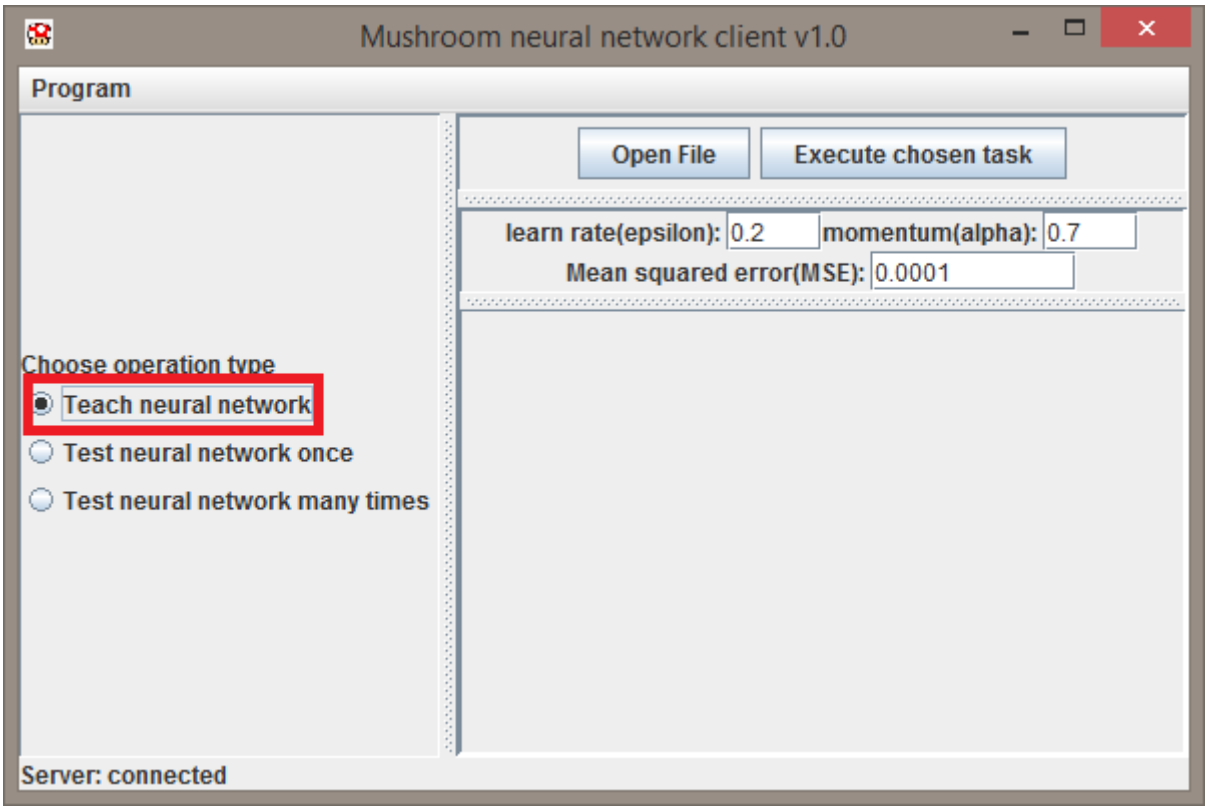
- Opcja ‘’Info’’ – prezentuje użytkownikowi krótką wskazówkę na temat działania programu.



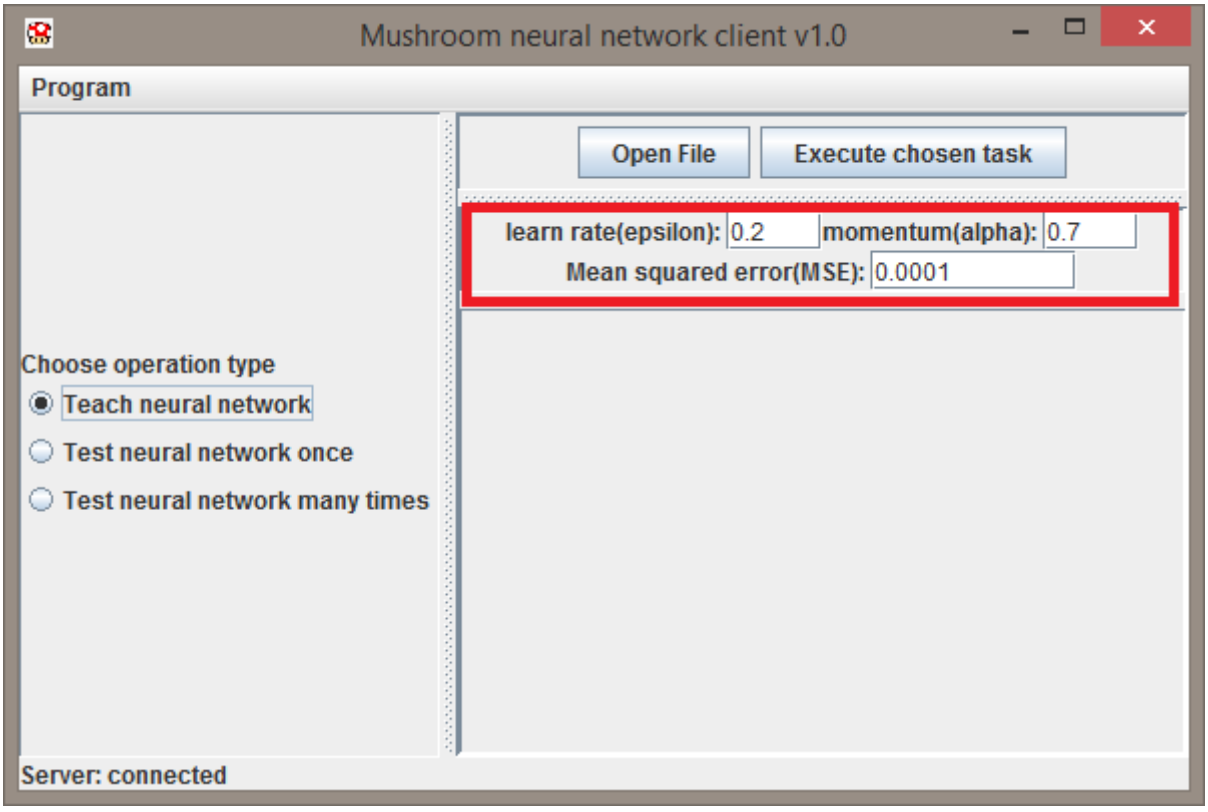
- Opcja ‘’Exit’’ – po kliknięciu program zapyta użytkownika czy ten na pewno chce wyjść z programu.



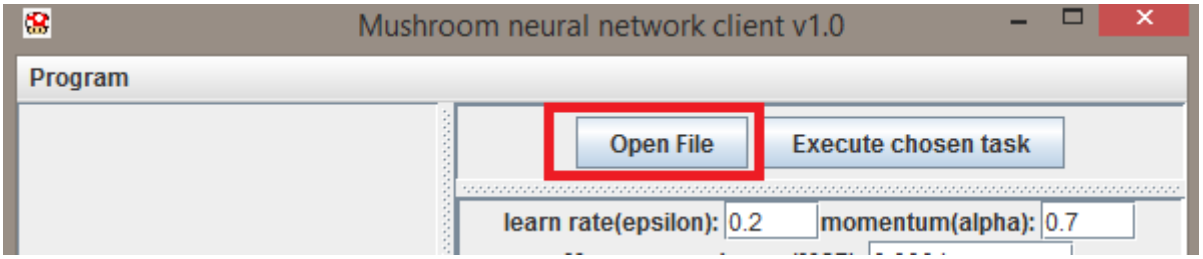
6. Program udostępnia użytkownikowi następujące funkcjonalności:
   1) Nauczenie sieci neuronowej
      - By nauczyć sieć neuronową użytkownik zaznacza następujący Radio Button



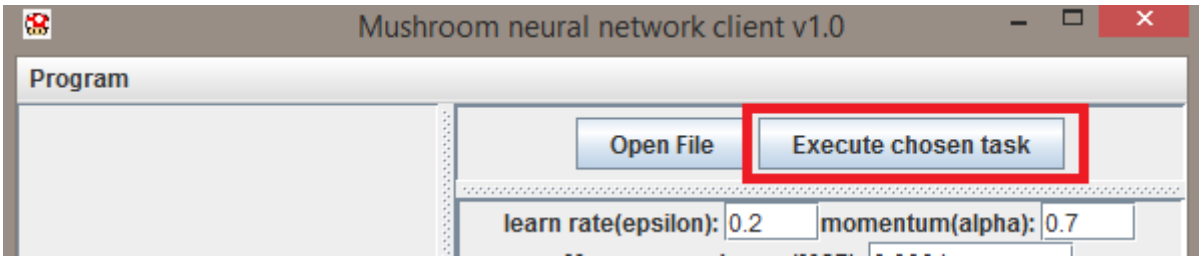
- Następnie podaje wartość współczynnika uczenia(learn rate), wartość momentum oraz wartość błędu średniokwadratowego epoki, po którego osiągnięciu program ma uznać, że sieć została nauczona i zaprzestać uczenia sieci znajdującej się po stronie serwera



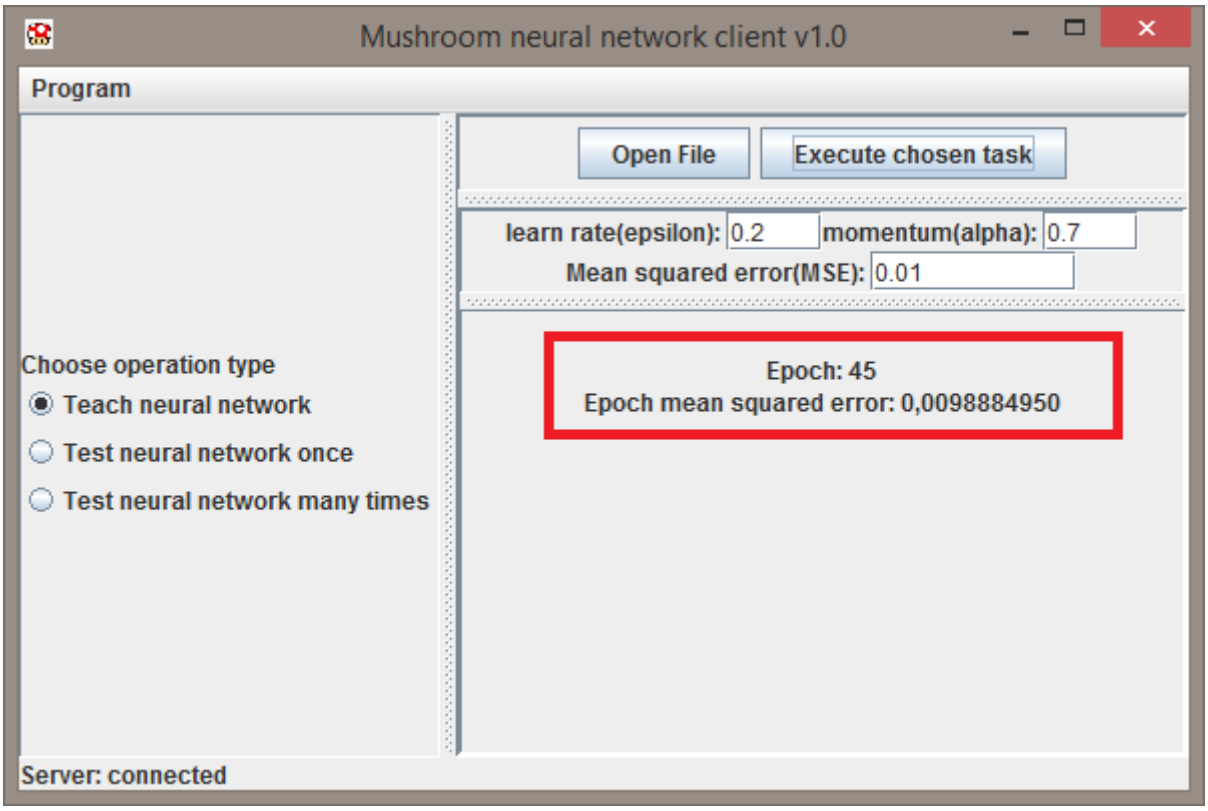
- Następnie użytkownik wybiera plik z rozszerzeniem .csv, w którym znajdują się rekordy służące do trenowania sieci neuronowej. Wyboru tego pliku użytkownik dokonuje po kliknięciu w przycisk ‘’Open File’’



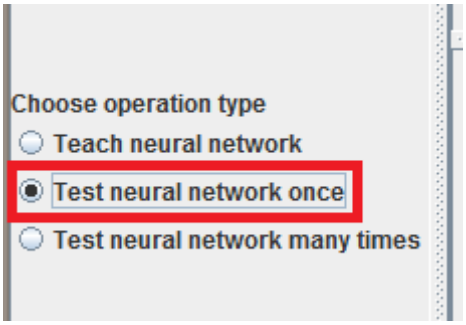
- Program wyświetli okno wyboru pliku. Po wybraniu pliku użytkownik klika w przycisk ‘’Execute chosen task”



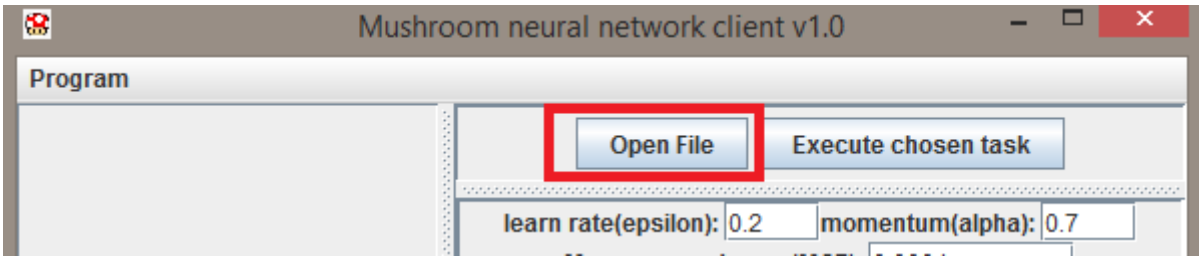
- Teraz następuje uczenie sieci neuronowej znajdującej się po stronie serwera. Proces ten w zależności od wybranych przez użytkownika wartości współczynników learn rate, momentum i MSE może potrwać od kilku sekund nawet do kilkunastu, kilkudziesięciu minut. Program klienta zatrzyma proces uczenia sieci kiedy otrzyma od serwera informację że średni błąd średniokwadratowy epoki jest mniejszy od zdefiniowanego przez użytkownika i wyświetli liczbę epok, które były potrzebne do nauki oraz aktualny błąd średniokwadratowy epoki



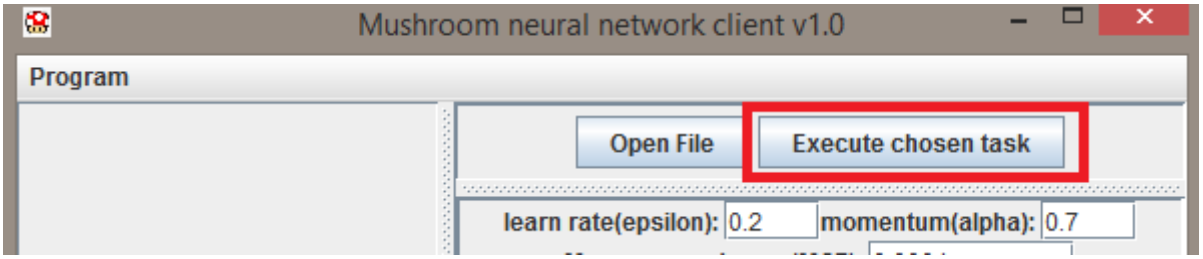
2) Klasyfikację pojedynczego grzyba pod względem jadalności
   - Użytkownik wybiera opcję



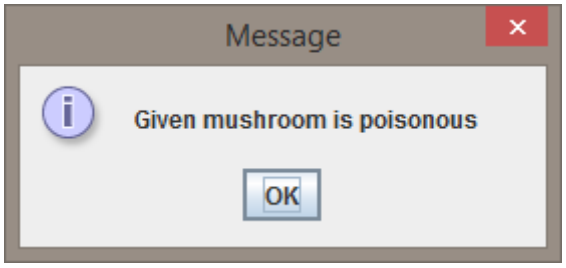
   - Następnie użytkownik wybiera plik z rozszerzeniem .csv, w którym znajduje się 1 rekord opisujący grzyba, którego użytkownik chce sklasyfikować. Uwaga! Rekord ten nie powinien opisywać pola clasification, ponieważ to serwis ma określić klasyfikację grzyba. W związku z tym rekord powinien zaczynać się od pola opisującego atrybut: ''cap-shape''. Wyboru tego pliku użytkownik dokonuje po kliknięciu w przycisk ''Open File''



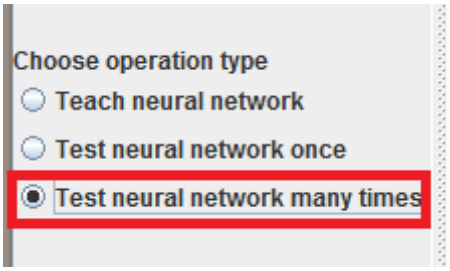
- Następnie klika w przycisk ‘’Execute chosen task”



- Wynik klasyfikacji rekordu wybranego przez użytkownika program wyświetli w oknie dialogowym

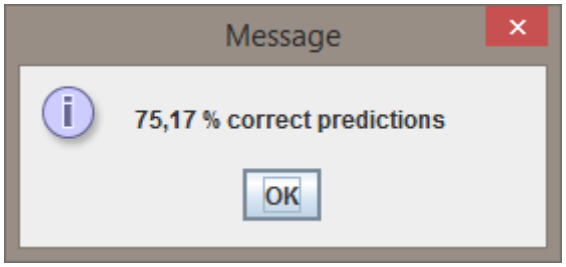


3) Na potrzeby testowania optymalnej struktury sieci neuronowej zaimplementowana została opcja wielokrotnego testowania sieci neuronowej i stwierdzeniu procentu prawidłowych klasyfikacji sieci neuronowej



- Sekwencja wykonywanych instrukcji prowadzących do wykonania tej operacji jest prawie taka sama jak dla opcji jednorazowego testowania sieci neuronowej z tą różnicą, że plik, który wskazuje użytkownik zawiera dowolną ilość rekordów opisujących grzyby. Rekordy te jednak muszą zawierać atrybuty stwierdzające czy dany grzyb jest jadalny lub trujący po to, by możliwe stało się oszacowanie procentu prawidłowych klasyfikacji sieci neuronowej.

- Po wykonaniu operacji program wyświetli okno dialogowe prezentujące oszacowanie procentu prawidłowych klasyfikacji



7. Należy pamiętać, że program jest skonstruowany w taki sposób, że:
    - To aplikacja klienta przeprowadza nadzorowany proces uczenia wysyłając do serwisu rekordy opisujące grzyby.
    - Aplikacja serwera odsyła aplikacji klienta wartość błędu średniokwadratowego dla danej epoki.
    - Aplikacja klienta na podstawie błędu średniokwadratowego epoki decyduje czy zakończyć proces uczenia czy rozpocząć nową epokę.
    - Każdorazowe zamknięcie aplikacji klienta przy wytrenowanej sieci neuronowej znajdującej się po stronie serwera nie powoduje utraty danych dla wytrenowanej sieci neuronowej. Utrata następuje tylko po zakończeniu działania aplikacji serwera lub po wybraniu przez użytkownika opcji ‘’Reset’’ z menu kontekstowego programu.
    - Jeśli aplikacja nie odpowiada na zdarzenia użytkownika, np. kliknięcia przycisków, a ponadto program symuluje stałe wciśnięcie przycisku ‘’Execute chosen button’’ oznacza to, że program w aktualnym momencie przeprowadza proces uczenia sieci neuronowej i stanie się zdolny do użytku po wytrenowaniu sieci i zwróceniu komunikatu o liczbie epok, które minęły do momentu nauczenia sieci i wartości błędu średniokwadratowego epoki.

Inne informacje o programie

- Programy serwera skompilowane są tak, że wyjścia neuronów obliczane są przy użyciu fukcji aktywacji f(x) = tanh(x). Można je uruchamiać i testować. W celu skorzystania z sigmoidalnej funkcji aktywacji konieczne jest naniesienie kilku poprawek w kodzie i ponowne skompilowanie projektu. Poprawki te opatrzone są komentarzami w klasach zaczynającymi się od słów ‘’sigmoid change’’, np.

```
error = normaliseOutput(currentMushroom.getClasificationOrdinalVal
ue(), 0, 1, -1.0, 1.0) - outputNeuron.getOutValue();
//sigmoid change: comment line above and uncomment following line
```

```
//error = normaliseOutput(currentMushroom.getClasificationOrdinalValue(),
0, 1, 0.0, 1.0) - outputNeuron.getOutValue();
```

- Program klienta działa zarówno z jedną jak i z drugą wersją programu serwera.
- Zarówno w programie klienta jak i w 2 programach serwera zostały zawarte komentarze dotyczące większości zmiennych i metod. Ponadto zostały wygenerowane 3 dokumenty JavaDoc opisujące zmienne, klasy i metody. Pliki JavaDoc znajdują się w repozytorium githuba w folderze ''Dokumentacja" w folderach dotyczących odpowiednich aplikacji klienta i serwera.

Analiza i testowanie napisanego klasyfikatora

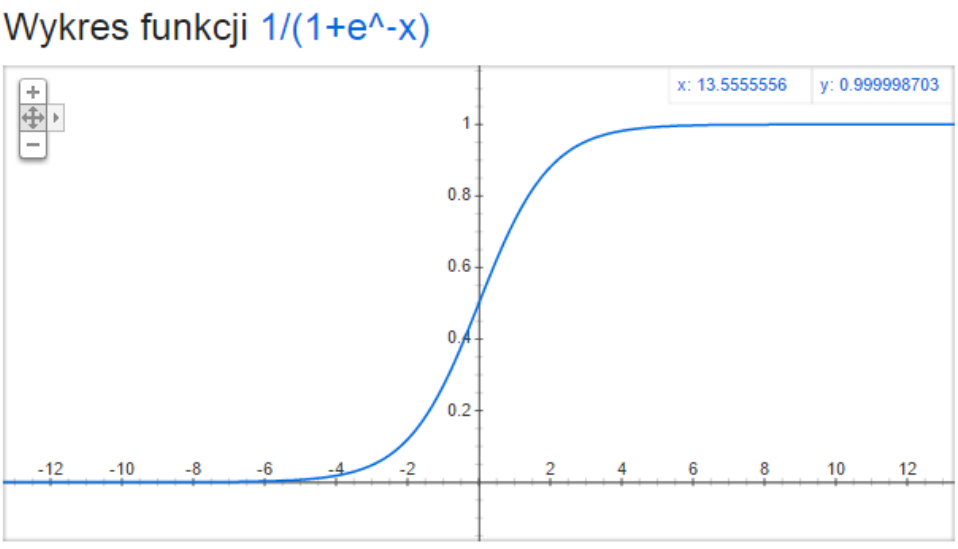
Dla napisanego klasyfikatora został przeprowadzony szereg testów służących wyłonieniu najlepszej struktury sieci dla problemu klasyfikacji grzybów pod względem jadalności. Testy przeprowadziłem dla dwóch funkcji aktywacji:

1. Dla sigmoidalnej funkcji aktywacji opisanej wzorem

$$s(x) = \frac{1}{1 + e^{-x}}$$

I jej pochodnej:

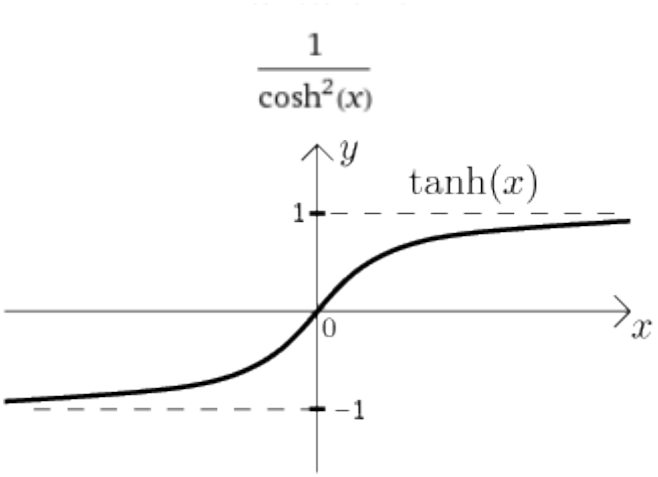
$$\frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{e^x}{(e^x+1)^2}$$



2. Dla funkcji aktywacji opisanej wzorem

$$f(x) = \tanh(x)$$

I jej pochodnej:

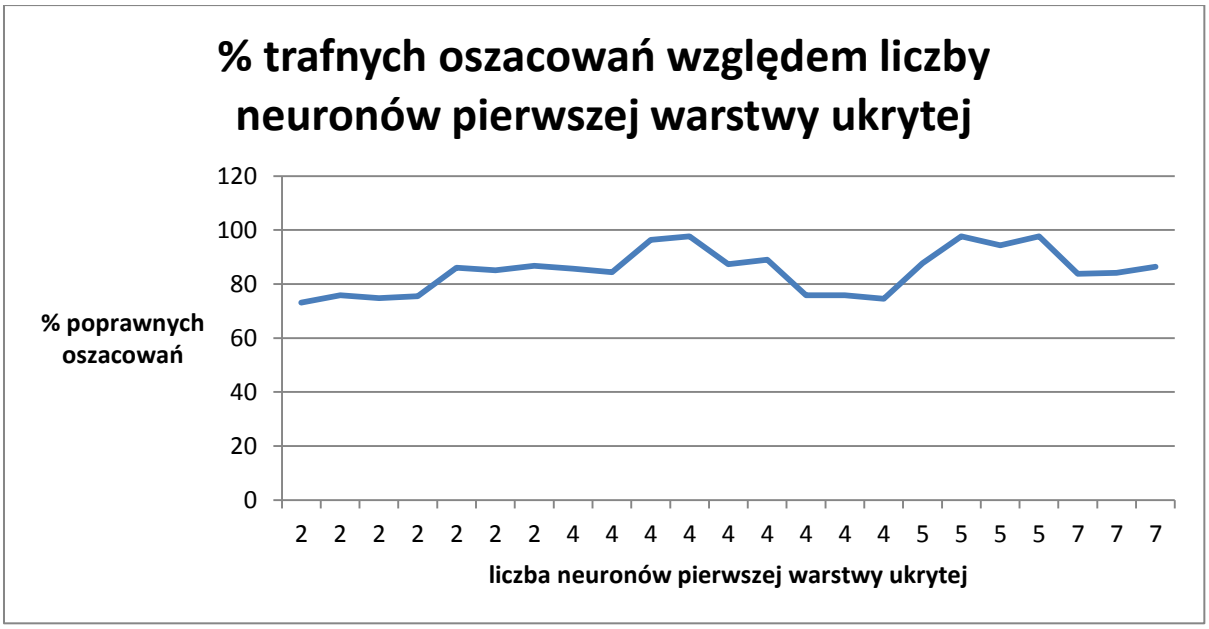


Wyniki analizy przeprowadzonej dla sigmoidalnej funkcji aktywacji

1. Sigmoidalna funkcja aktywacji z jedną warstwą ukrytą neuronów

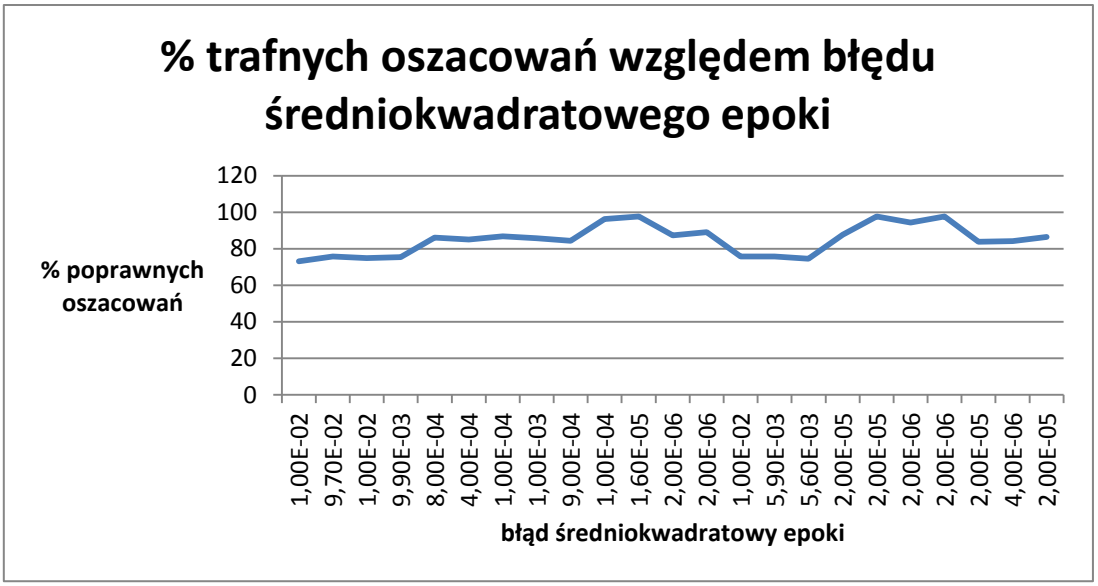
Nr. Próby	MSE	Liczba epok	% poprawnych oszacowań	learn rate	momentum	liczba neuronów pierwszej warstwy ukrytej	liczba neuronów drugiej warstwy ukrytej
1.	1,00E-02	59	73.18	0.2	0.8	2	0
2.	9,70E-02	206	75.83	0.2	0.5	2	0
3.	1,00E-02	63	74.83	0.5	0.0	2	0
4.	9,90E-03	81	75.50	0.2	0.7	2	0
5.	8,00E-04	189	86.09	0.2	0.7	2	0
6.	4,00E-04	167	85.10	0.2	0.7	2	0
7.	1,00E-04	177	86.75	0.2	0.7	2	0
8.	1,00E-03	210	85.76	0.2	0.7	4	0
9.	9,00E-04	202	84.44	0.2	0.7	4	0
10.	1,00E-04	360	96.36	0.2	0.7	4	0
11.	1,60E-05	889	97.68	0.2	0.7	4	0
12.	2,00E-06	2366	87.42	0.2	0.7	4	0
13.	2,00E-06	336	89.07	0.2	0.7	4	0
14.	1,00E-02	54	75.83	0.2	0.7	4	0
15.	5,90E-03	78	75.83	0.2	0.7	4	0
16.	5,60E-03	51	74.50	0.2	0.8	4	0
17.	2,00E-05	742	87.75	0.2	0.7	5	0
18.	2,00E-05	582	97.68	0.2	0.7	5	0
19.	2,00E-06	1548	94.37	0.2	0.7	5	0
20.	2,00E-06	929	97.68	0.2	0.7	5	0
21.	2,00E-05	506	83.77	0.2	0.7	7	0
22.	4,00E-06	1668	84.11	0.2	0.7	7	0

23.	2,00E-05	567	86.42	0.2	0.7	7	0



Najlepsze wyniki uzyskałem (97,68% trafnych oszacowań) dla liczby neuronów pierwszej warstwy ukrytej równej 5 (nie wliczając neuronu z BIASem).

Nie dostrzegłem ścisłego związku pomiędzy liczbą epok potrzebnych do wytrenowania sieci neuronowej przy założonym poziomie błędu epoki od liczby neuronów pierwszej ukrytej warstwy neuronów.

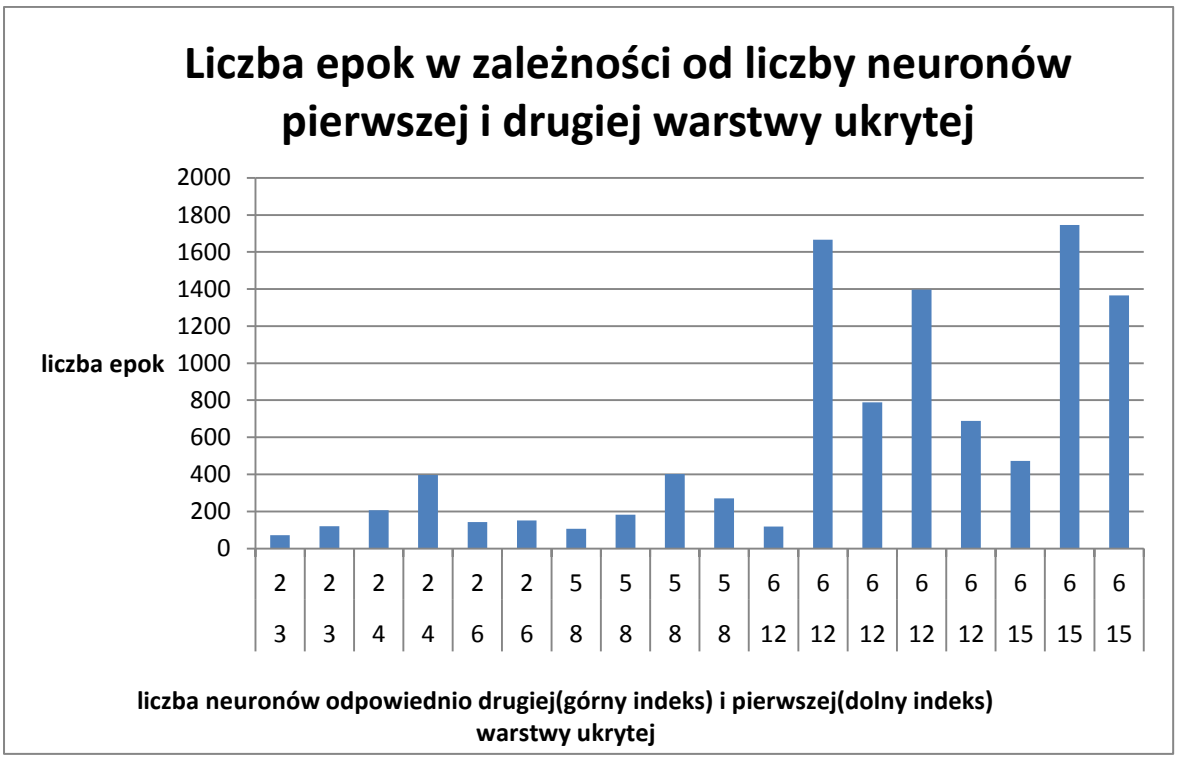


Gdy błąd średniokwadratowy epoki maleje, wzrasta % poprawnych oszacowań, zgodnie z oczekiwaniami.

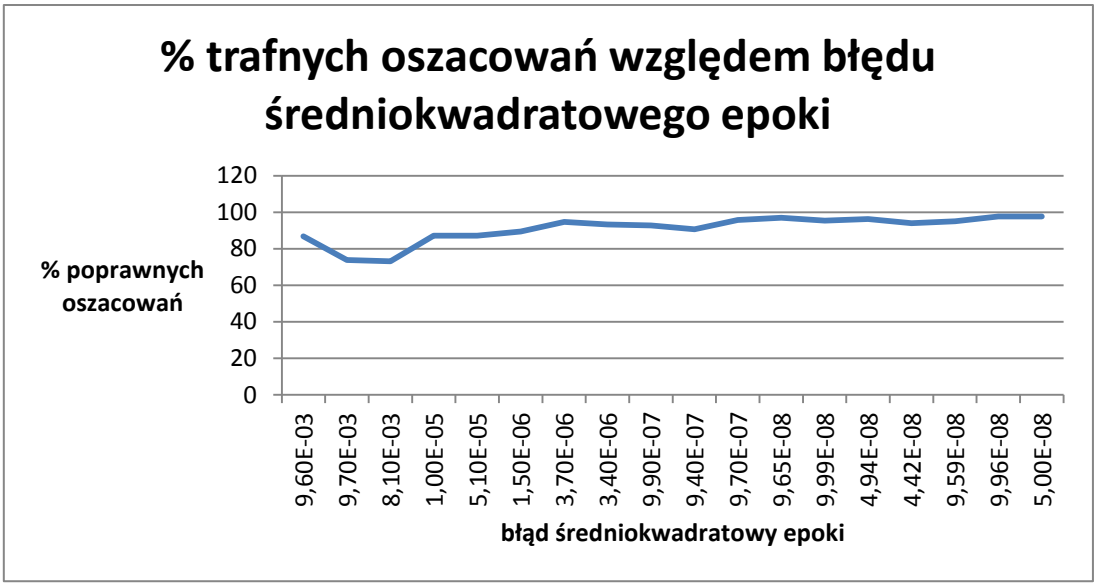
2. Sigmoidalna funkcja aktywacji z dwoma warstwami ukrytymi neuronów

Nr Próby	MSE	Liczba epok	% poprawnych oszacowań	learn rate	momentum	liczba neuronów pierwszej warstwy ukrytej	liczba neuronów drugiej warstwy ukrytej
1	9,60E-03	72	86,75	0,2	0,7	3	2
2	9,70E-03	120	73,84	0,2	0,7	3	2
3	8,10E-03	206	73,18	0,2	0,7	4	2
4	1,00E-05	396	87,09	0,2	0,7	4	2
5	5,10E-05	142	87,09	0,2	0,7	6	2
6	1,50E-06	151	89,4	0,2	0,7	6	2
7	3,70E-06	106	94,7	0,2	0,7	8	5
8	3,40E-06	183	93,38	0,2	0,7	8	5
9	9,90E-07	401	92,72	0,2	0,7	8	5
10	9,40E-07	270	90,73	0,2	0,7	8	5
11	9,70E-07	119	95,7	0,2	0,7	12	6
12	9,65E-08	1667	97,02	0,2	0,7	12	6
13	9,99E-08	789	95,36	0,2	0,7	12	6
14	4,94E-08	1397	96,36	0,2	0,7	12	6
15	4,42E-08	689	94,04	0,2	0,7	12	6
16	9,59E-08	473	95,03	0,2	0,7	15	6
17	9,96E-08	1745	97,68	0,2	0,7	15	6
18	5,00E-08	1366	97,68	0,2	0,7	15	6

Większa liczba neuronów w warswie I i warstwie II przyczyniała się do poprawy liczby poprawnych kwalifikacji.



Większa liczba neuronów I i II warstwy powoduje, że proces treningu wydłuża się.



Gdy błąd średniokwadratowy epoki maleje, wzrasta % poprawnych oszacowań, zgodnie z oczekiwaniami.

Sigmoidalna funkcja aktywacji – wnioski

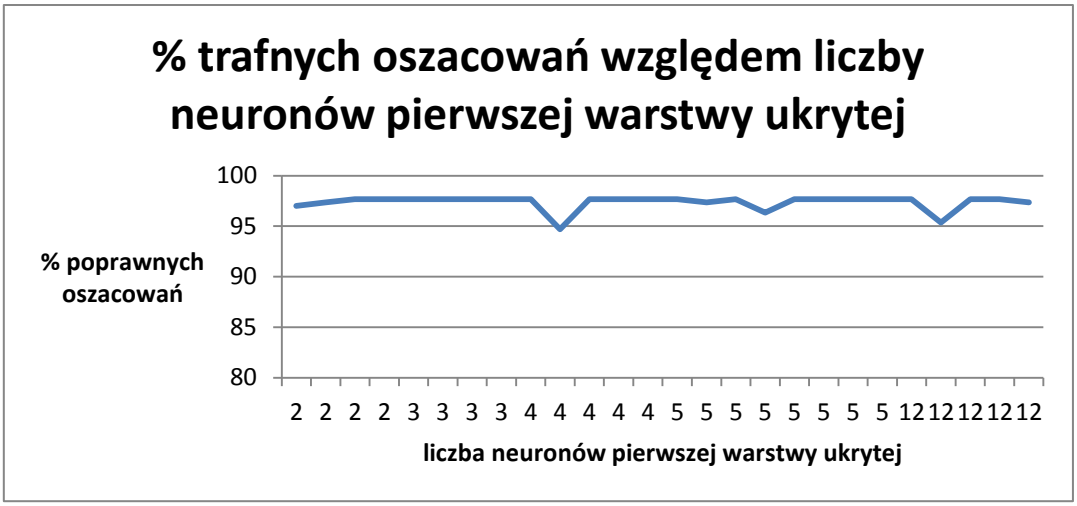
- Proces uczenia sieci neuronowej z sigmoidalną funkcją aktywacji wyjść jest nierzadko procesem długim, a przy nieumiejętnie dobranych współczynnikach uczenia i momentum często niemożliwy. Dobranie współczynnika uczenia większego od wartości 0.3 i mniejszego od 0.2 często kończyło się tym, że sieć nie została wytrenowana, stąd danych tych nie ma w tabeli.
- Dobór liczby neuronów w poszczególnych warstwach jest kwestią bardzo istotną. Zaobserwowałem, że sieci neuronowej z jedną warstwą ukrytą neuronów posiadającą w warstwie ukrytej 1 neuron (oprócz BIASu) program nie mógł wytrenować. Wytrenować nie mogłem także sieci o strukturze 12-4 (12 neuronów w I warstwie ukrytej i 4 w II warstwie ukrytej). Podobne zachowanie przejawiały sieci neuronowe z jedną lub dwoma warstwami ukrytymi posiadające dużą liczbę neuronów w poszczególnych warstwach. Generalnie przez to, że program nie mógł się wytrenować mam na myśli fakt, że nie został osiągnięty błąd średniokwadratowy dla epoki mniejszy niż 0.01.
- Najlepsze wyniki uzyskałem dla struktury sieci neuronowej z jedną warstwą ukrytą posiadającą 5 neuronów (+ 1 neuron BIASu) osiągając 97.68% trafnych klasyfikacji w 300 elementowej próbie.
- Nie zaobserwowałem, żeby zmiany wartości momentum przy sieci neuronowej z sigmoidalną funkcją aktywacji pełniły dużą rolę. Przy dużych wartościach momentum rzędu 0.9 sieć wykonywała tak duże oscylacje w poszukiwaniu optymalnych wag poszczególnych połączeń, że nie mogła znaleźć lokalnych wartości dla których błąd byłby najmniejszy.

- Korzystając z sigmoidalnej funkcji aktywacji wyjść wektor danych wejściowych musiałem poddać normalizacji do zakresu od 0.0 do 1.0. Próba zwiększenia zakresu, np. od 0.0 do 2.0 lub od -1.0 do 1.0 powodowała, że sieć nie mogła się nauczyć. Prawdopodobnie spowodowane jest to tym, że funkcja sigmoidalna generuje wartości z zakresu od 0.0 do 1.0.
- Sieć z dwoma warstwami ukrytymi i sigmoidalną funkcją aktywacji wyjść łatwiej jest nauczyć.

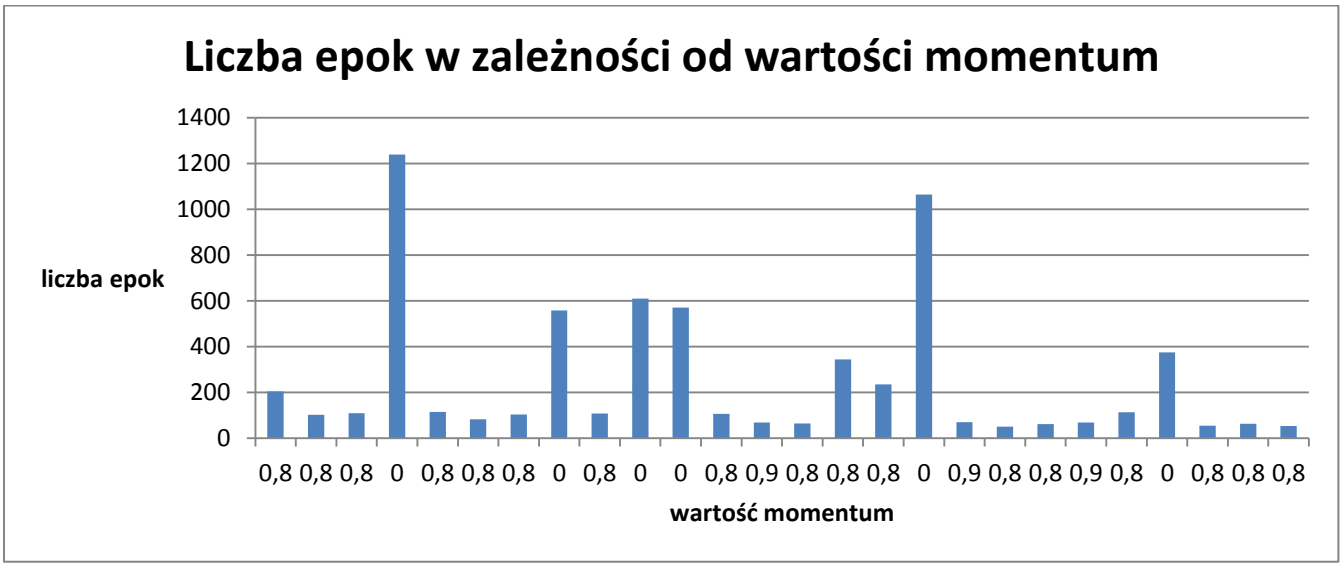
Wyniki analizy przeprowadzonej dla funkcji aktywacji tangens hiperboliczny tanh(x)

1. Funkcja aktywacji tangens hiperboliczny tanh(x) z jedną warstwą ukrytą neuronów

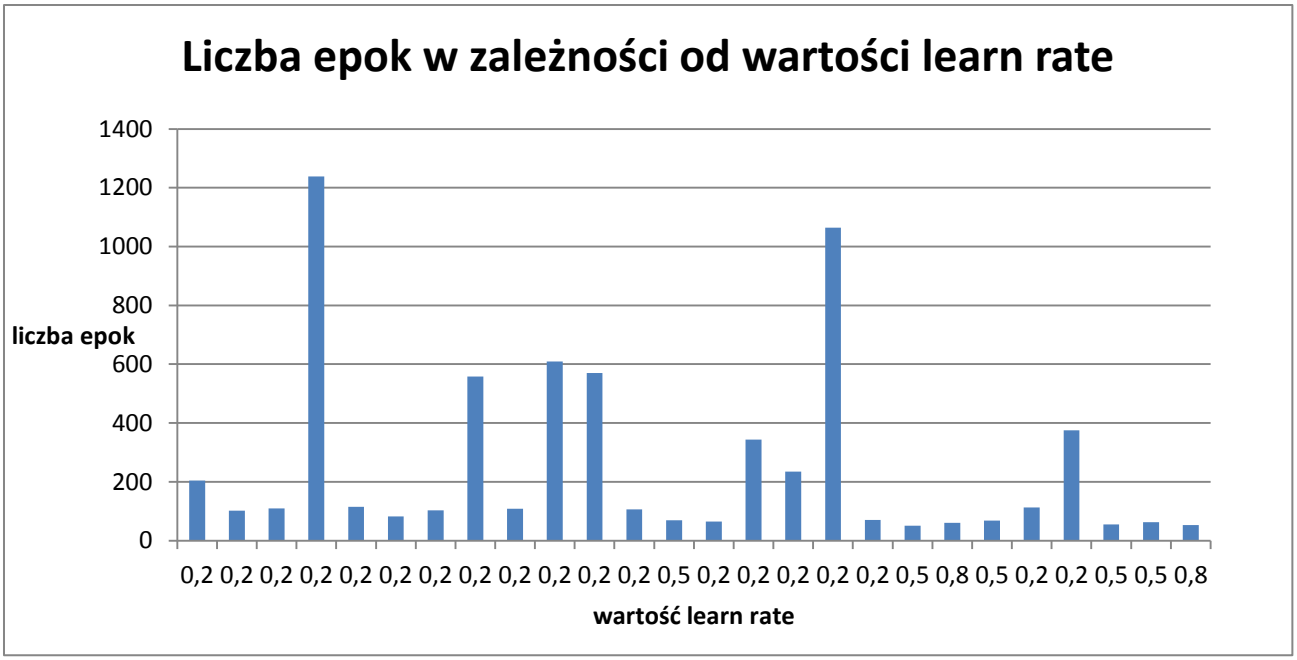
Nr Próby	MSE	Liczba epok	% poprawnych oszacowań	learn rate	momentum	liczba neuronów pierwszej warstwy ukrytej	liczba neuronów drugiej warstwy ukrytej
1	9,97E-09	204	97,02	0,2	0,8	2	0
2	9,65E-09	102	97,35	0,2	0,8	2	0
3	9,63E-09	109	97,68	0,2	0,8	2	0
4	9,99E-09	1239	97,68	0,2	0	2	0
5	9,79E-09	115	97,68	0,2	0,8	3	0
6	9,99E-09	82	97,68	0,2	0,8	3	0
7	9,68E-09	103	97,68	0,2	0,8	3	0
8	9,97E-09	558	97,68	0,2	0	3	0
9	9,98E-09	108	97,68	0,2	0,8	4	0
10	9,97E-09	609	94,7	0,2	0	4	0
11	9,98E-09	570	97,68	0,2	0	4	0
12	9,78E-09	106	97,68	0,2	0,8	4	0
13	9,95E-09	69	97,68	0,5	0,9	4	0
14	1,59E-06	65	97,68	0,2	0,8	5	0
15	9,98E-10	344	97,35	0,2	0,8	5	0
16	9,98E-10	235	97,68	0,2	0,8	5	0
17	9,99E-09	1064	96,36	0,2	0	5	0
18	9,54E-09	70	97,68	0,2	0,9	5	0
19	4,85E-17	51	97,68	0,5	0,8	5	0
20	3,72E-11	61	97,68	0,8	0,8	5	0
21	8,77E-24	68	97,68	0,5	0,9	5	0
22	9,96E-09	113	97,68	0,2	0,8	12	0
23	9,98E-09	375	95,36	0,2	0	12	0
24	3,31E-10	55	97,68	0,5	0,8	12	0
25	1,16E-14	63	97,68	0,5	0,8	12	0
26	6,37E-09	53	97,35	0,8	0,8	12	0



Powyższy wykres przedstawia zależność procentu trafnych oszacowań od liczby neuronów pierwszej warstwy ukrytej. Jak widać, dla tak małej liczby neuronów jak 2 + neuron BIASu sieć dokonała 97.68% trafnych klasyfikacji.



Na podstawie powyższego wykresu można stwierdzić, że duża wartość momentum znacząco minimalizuje liczbę epok potrzebnych do wytrenowania sieci neuronowej.

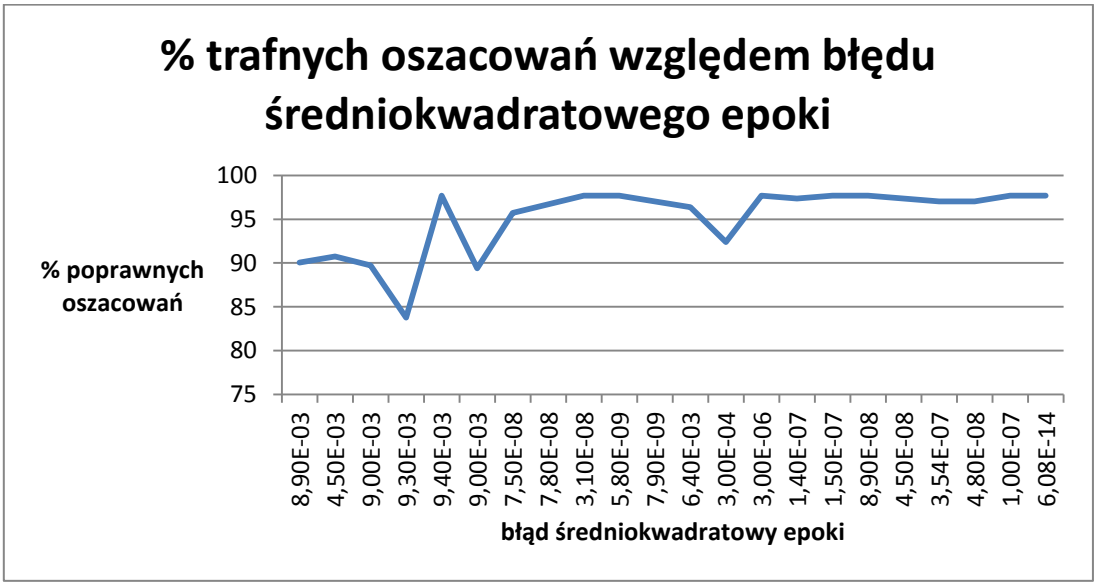


Na podstawie powyższego wykresu można stwierdzić, że wartość learn rate znacząco poprawiła szybkość treningu sieci neuronowej.

2. Funkcja aktywacji tangens hiperboliczny tanh(x) z dwoma warstwami ukrytymi neuronów

Nr Próby	MSE	Liczba epok	% poprawnych oszacowań	learn rate	momentum	liczba neuronów pierwszej warstwy ukrytej	liczba neuronów drugiej warstwy ukrytej
1	8,90E-03	135	90,07	0,2	0,7	3	2
2	4,50E-03	469	90,73	0,2	0,7	3	2
3	9,00E-03	249	89,74	0,2	0,7	3	2
4	9,30E-03	135	83,77	0,2	0,7	4	2
5	9,40E-03	253	97,68	0,2	0,7	4	2
6	9,00E-03	290	89,4	0,2	0,7	4	2
7	7,50E-08	103	95,7	0,2	0,7	4	3
8	7,80E-08	84	96,69	0,2	0,7	4	3
9	3,10E-08	119	97,68	0,2	0,7	4	3
10	5,80E-09	97	97,68	0,2	0,9	4	3
11	7,90E-09	138	97,02	0,2	0,9	4	3
12	6,40E-03	262	96,36	0,2	0,7	6	2
13	3,00E-04	481	92,38	0,2	0,7	6	2
14	3,00E-06	89	97,68	0,2	0,7	6	4
15	1,40E-07	69	97,35	0,2	0,7	6	4
16	1,50E-07	124	97,68	0,2	0,7	6	4
17	8,90E-08	78	97,68	0,2	0,7	6	4

18	4,50E-08	69	97,35	0,2	0,7	6	4
19	3,54E-07	65	97,02	0,2	0,7	8	5
20	4,80E-08	111	97,02	0,2	0,7	8	5
21	1,00E-07	102	97,68	0,2	0,7	8	5
22	6,08E-14	73	97,68	0,2	0,7	12	4



Na podstawie powyższego wykresu można zauważyć, że tylko dobrze nauczona sieć neuronowa z wartością błędu średniokwadratowego na bardzo niskim poziomie jest w stanie poprawnie klasyfikować wysoki odsetek danych testowych.

Funkcja aktywacji tangens hiperboliczny – wnioski

- Sieć neuronowa z funkcją aktywacji wyjść opisaną wzorem tanh(x) trenuje się całkiem szybko. Duże wartości współczynnika uczenia oraz momentum znacznie skracały proces uczenia sieci neuronowej. Korekcje wag połączeń sieci neuronowej dokonywane były w tak dokładny sposób, że możliwe było uzyskanie błędu średniokwadratowego epoki rzędu wartości 8,77E-24.
- Dobór liczby neuronów w poszczególnych nie grał tak istotnej roli jak przy sieci z sigmoidalną funkcją aktywacji wyjść neuronów. Sieci z jedną warstwą ukrytą neuronów posiadającą w warstwie ukrytej 1 neuron (oprócz BIASu), podobnie jak dla sigmoidalnej funkcji aktywacji program nie mógł wytrenować.
- Chociaż przy sigmoidalnej funkcji aktywacji wyjść neuronów mogłem wytypować najlepszą strukturę sieci neuronowej pod względem liczby warstw i liczby neuronów w poszczególnych warstwach, to dla rozpatrywanej funkcji aktywacji nie mam

faworyta. Zarówno sieci z jedną jak i dwoma warstwami ukrytymi neuronów stosunkowo szybko trenowały się i dokonywały ponad 90% poprawnych predykcji.

- Korzystając z funkcji aktywacji wyjść opisanej wzorem tanh(x) wektor danych wejściowych musiałem poddawałem normalizacji min-max do zakresu od -1.0 do 1.0. Próba zmiany zakresu, np. od 0.0 do 1.0 spowodowała znaczne obniżenie stopnia trafnych predykcji. Nie udało mi się również zminimalizować błędu średniokwadratowego do wielkości mniejszych niż 0.001. Poniżej zestawienie kilku wykonanych przeze mnie prób dla takiej sieci

Nr Próby	MSE	Liczba epok	% poprawnych oszacowań	learn rate	momentum	liczba neuronów pierwszej warstwy ukrytej	liczba neuronów drugiej warstwy ukrytej
1	8,50E-03	28	68,21	0,3	0,8	7	0
2	7,80E-03	65	73,84	0,3	0,8	7	0
3	7,80E-03	44	67,88	0,3	0,7	7	0
4	8,80E-03	95	65,89	0,3	0,9	7	0
5	9,90E-03	16	57,95	0,3	0,7	7	0
6	5,90E-03	84	58,94	0,3	0,7	7	0

Podsumowanie

- Klasyfikator oparty na sieci neuronowej bardzo dobrze sprawdza się w postaci klasyfikatora grzybów pod względem jadalności. Dla danych składających się z 300 rekordów opisujących grzyby, z których dokładnie połowa była zaklasyfikowana jako grzyby jadalne, a druga połowa jako grzyby niejadalne możliwa była taka korekcja wag, dla której program poprawnie klasyfikował aż 97.68% przypadków.
- W roli funkcji aktywacji wyjść neuronów znacznie lepiej niż sigmoidalna sprawdza się funkcja tangens hiperboliczny.
- Z problemem klasyfikacji grzybów radzi sobie bardzo dobrze już tak mała sieć jak o strukturze 22 – 2 – 1 (22 neurony wejściowe + 1 neuron BIASu - 2 neurony w pierwszej warstwie ukrytej + 1 neuron BIASu - 1 neuron wyjściowy) z funkcją aktywacji opisaną wzorem f(x) = tanh(x).

Źródła

https://www.kaggle.com/uciml/mushroom-classification

http://zsi.ii.us.edu.pl/~nowak/ed/cw4.pdf

http://www.heatonresearch.com/jeff/

http://www.neurosoft.edu.pl/media/pdf/jbartman/sztuczna_inteligencja/NTI%20cwiczenie_6.pdf