Szymon Durak, PSI 2016-2017, sprawozdanie z projektu

Opis problemu

Jako problem do rozważenia wybrano zagadnienie orientowania się w przestrzeni przez pojazd autonomiczny przy mocno ograniczonych instrumentach do zbierania informacji o otoczeniu, jak to ma miejsce w przypadku np. robotów sumo/minisumo. Celem projektu było sprawdzenie, czy różne rodzaje sieci neuronowych mogłyby posłużyć do interpretacji i wyciągania wniosków o otoczeniu na podstawie danych wejściowych z 16 rozłożonych w równych odstępach na obwodzie okręgu czujników odległości. Warto zauważyć, że wspomniane roboty sumo często dysponują zaledwie 3-4 czujnikami skierowanymi tylko do przodu i na boki, choć istnieją roboty wyposażane przez konstruktorów w układ czujników podobny do rozważanego w projekcie. Do uczenia sieci neuronowych wykorzystano kilka zbiorów uczących, liczących ok. 10000 rekordów każdy. Podczas uczenia, zbiór zawsze był losowo dzielony w proporcji 70:30 na podzbiór uczący i walidujący. Dane wejściowe przechowywano w formacie JSON.

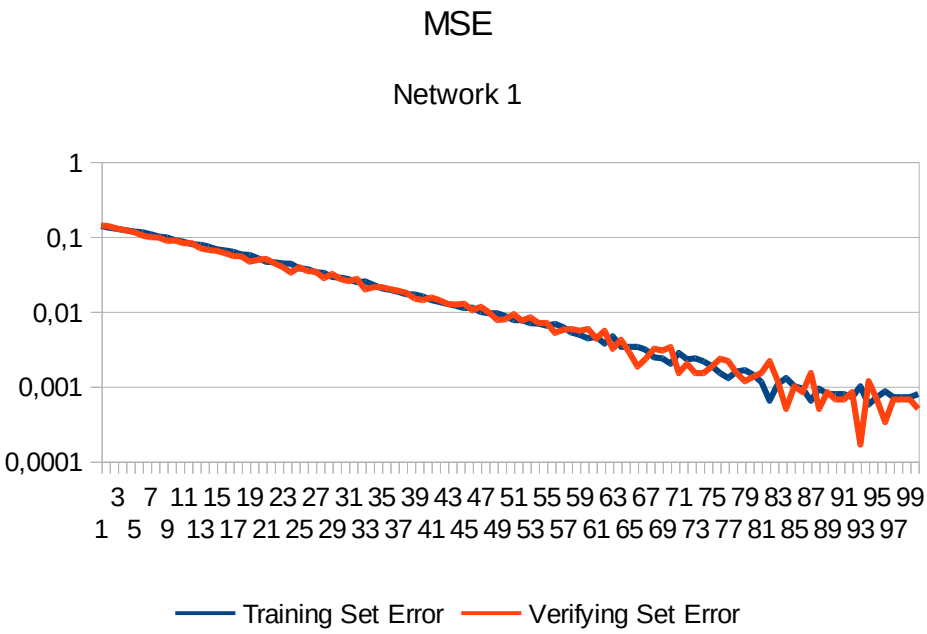
W ramach prodproblemów rozważano następujące zagadnienia:

1. Wykrywanie występowania kolizji oraz istnienia drogi wolnej od przeszkód – **perceptron**
2. Szacowanie odległości od najbliższej przeszkody – **sieć MCP**
3. Szacowanie kierunku w którym biegnie tunel – **sieć wielowarstwowa z propagacją wsteczną**
4. Grupowanie danych – rozpoznawanie ścian i tuneli – **sieć Hebba** i **Oji** w różnych wariantach
5. Grupowanie danych - klasyfikacja danych wejściowych z p.4 – **sieć WTA**
6. Uczenie sieci z p. 2 na danych sklasyfikowanych przez **sieć WTA**
7. Klasyfikacja przeszkód ustawionych w różnych miejscach wokół pojazdu – **sieć SOM z regułą WTM**

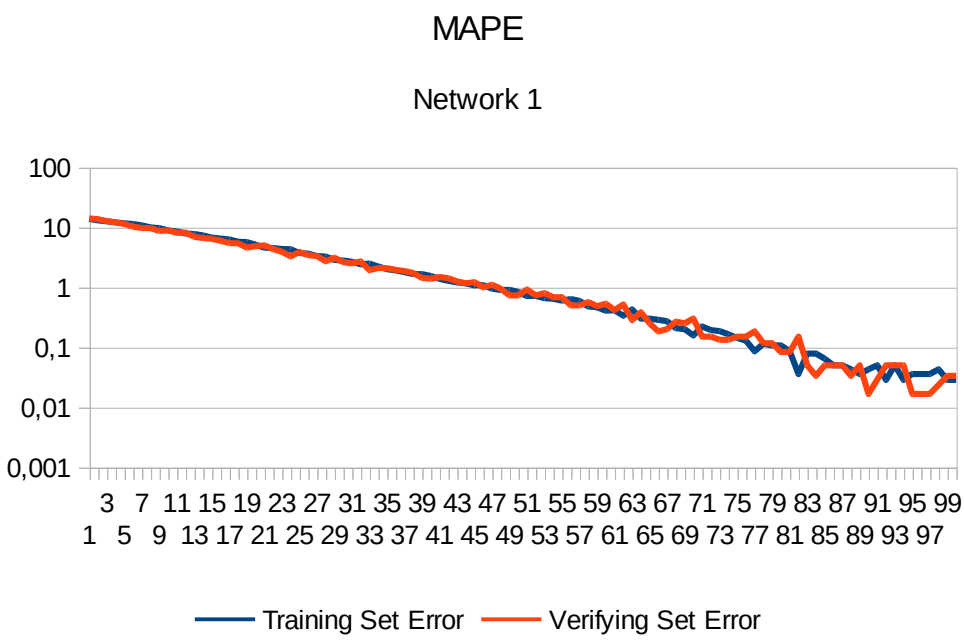
Perceptron

W przypadku perceptronu, założono, że kolizja występuje, jeśli odległość zmierzona przez dowolny czujnik okaże się mniejsza od progowej. Analogicznie rozwiązano problem istnienia drogi wolnej – musi istnieć pomiar wskazujący na odległość przekraczającą progową. Za progi przyjęto odpowiednio 0.35 i 0.65. Sieć posiadała zatem 16 wejść i 2 wyjścia. Pojedynczą daną wejściową jest w naszym przypadku wektor 16 sygnałów.

Warstwa	Liczba neuronów
Wejściowa	16
Wyjściowa	2



Jak widać, udało się uzyskać dość niski błąd MSE – co wskazuje na to, że sieć nauczyła się radzić sobie z wyznaczonym dla niej podproblemem.

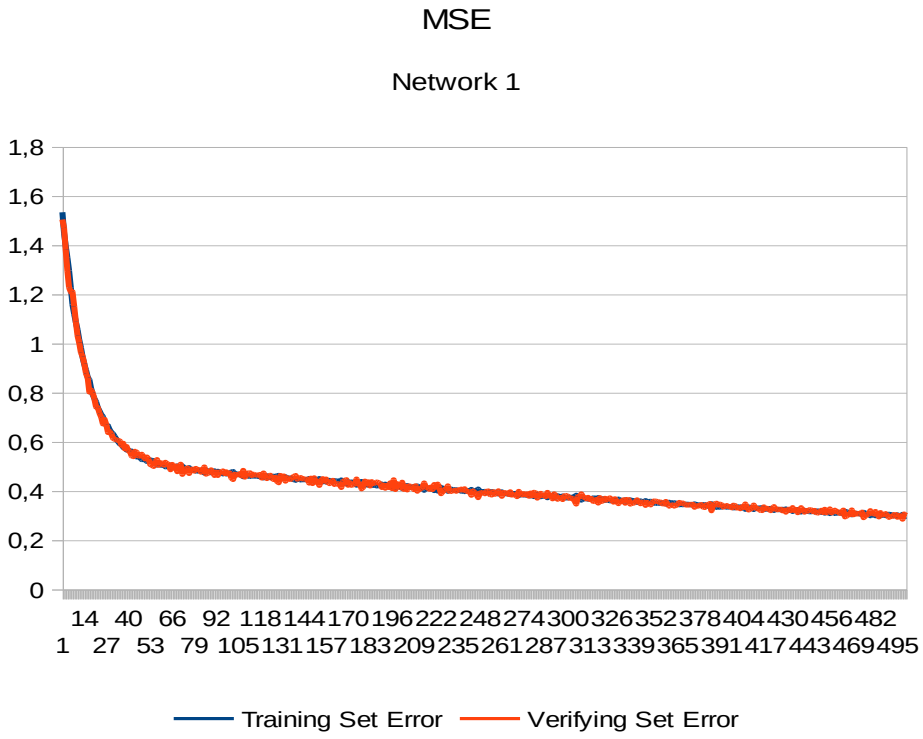


Wnioski wysnute z wykresu przedstawiającego błąd MSE potwierdza również błąd MAPE – który spadł do poziomów rzędu 0,1%. Co ciekawe, im dłużej sieci były uczone, tym bardziej błąd spadał i nie zaobserwowano zjawiska przeuczenia – jedynie stopniowy spadek tempa uczenia sieci.

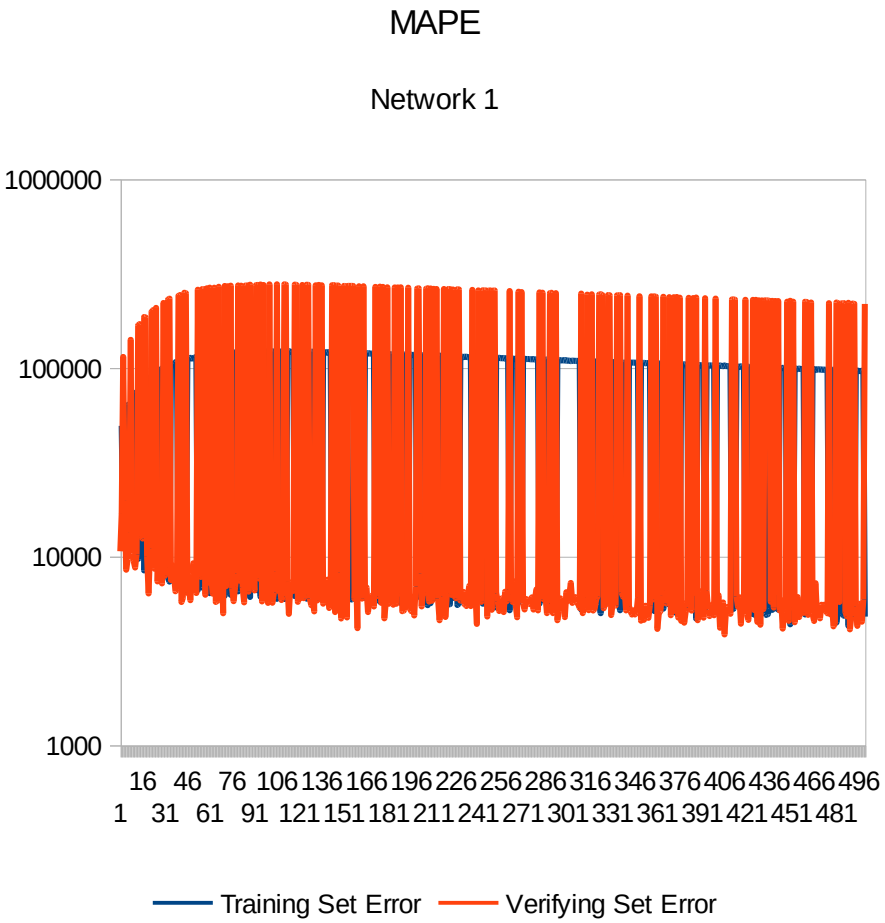
Sieć MCP

Sieć McCullocha-Pittsa miała posłużyć do wyznaczania odległości od najbliższej przeszkody. Jednak ze względu na nieliniowość problemu względem danych wejściowych oraz prostotę sieci, uzyskane wyniki okazały się całkowicie błędne i sieć nie nabyła zdolności udzielania prawidłowej odpowiedzi.

Warstwa	Liczba neuronów
Wejściowa	16
Wyjściowa	1



Błąd MSE zdaje się wskazywać na to, że sieć neuronowa powoli, ale jednak uczy się szacować odległość, jednak błąd MAPE, ukazujący procentowe błędy szacunków, daje zgoła inny obraz rzeczywistości:

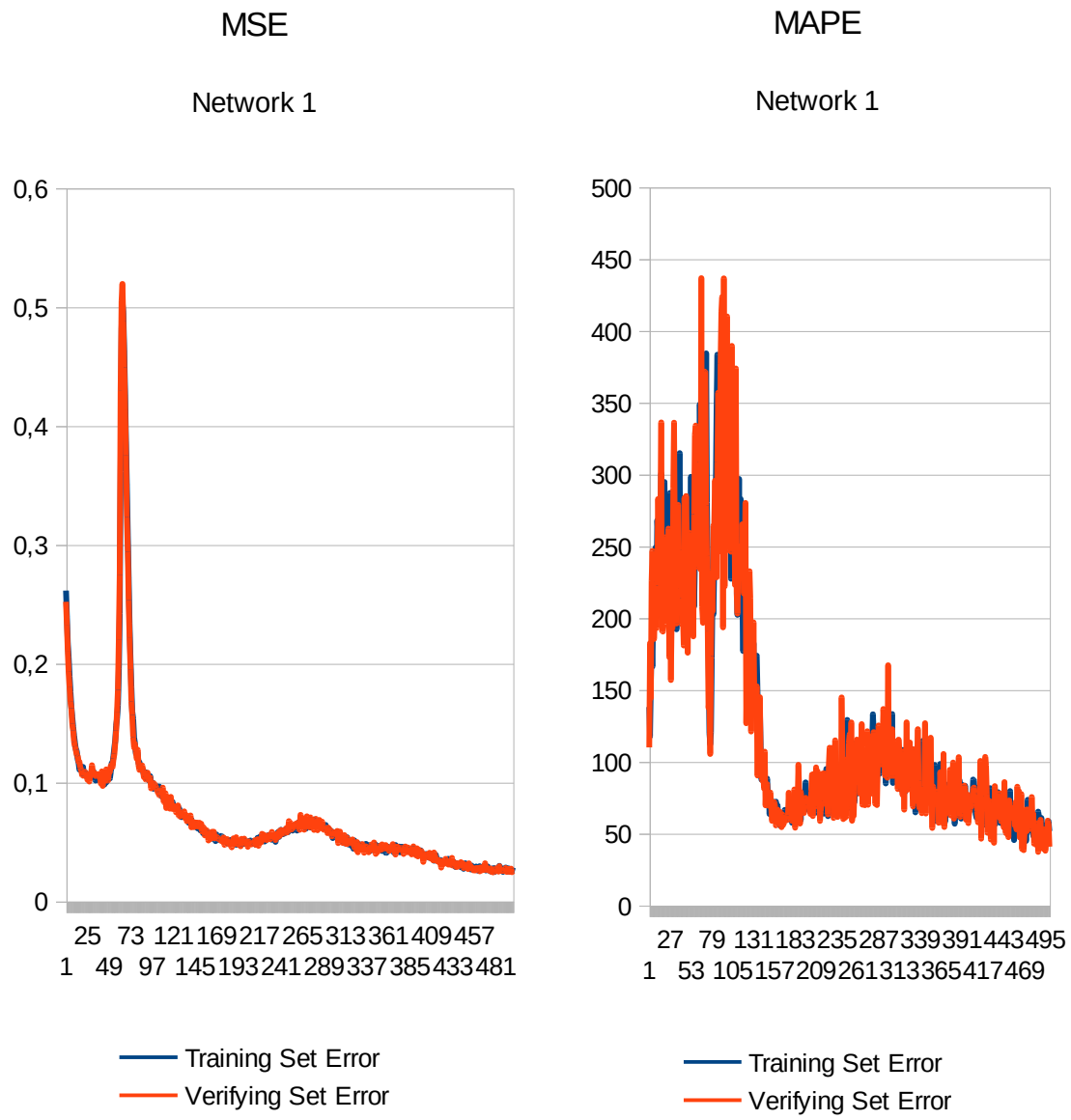


Błąd na poziomie 10000-100000% jest całkowicie nie do zaakceptowania, zatem próbę wykorzystania sieci MCP do szacowania odległości od najbliższej przeszkody możemy uznać za całkowitą porażkę.

Sieć wielowarstwowa z algorytmem wstecznej propagacji

W tym przypadku rozważono możliwość wykorzystania sieci neuronowej do określania kierunku, w którym biegnie tunel, w którym znajduje się pojazd. Ponownie pojazd ma do dyspozycji jedynie 16 czujników. Sieć wykorzystywała bipolarną funkcję aktywacji. Okazuje się, że we wspomnianym zagadnieniu sieć z 4 warstwami ukrytymi nauczyła się całkiem nieźle rozwiązywać podane zagadnienie, aczkolwiek pojawiały się błędy – jeśli tunel biegł niemal idealnie w prawo lub w lewo, sieć oczywiście nie umiała określić właściwego kierunku z powodu nierozróżnialności danych.

Warstwa	Liczba neuronów
Wejściowa	16
Ukryta	32
Ukryta	16
Ukryta	8
Ukryta	4
Wyjściowa	1



Interesujące jest, że mimo występowania etapów nawet znacznego przeuczenia, sieć po każdym takim etapie schodziła do poziomów błędu MSE poniżej minimum sprzed przeuczenia. Niepokoi jednak fakt wysokiego błędu MAPE – dynamika zmian dla kolejnych epok odpowiada tej dla błędu MSE, ale ustrzymuje się na wysokim poziomie. Z tego względu należy uznać, że sieć wielowarstwowa mogłaby posłużyć jedynie do orientacyjnego, a nie dokładnego wyznaczania kierunków.

Sieci Hebba i Oji

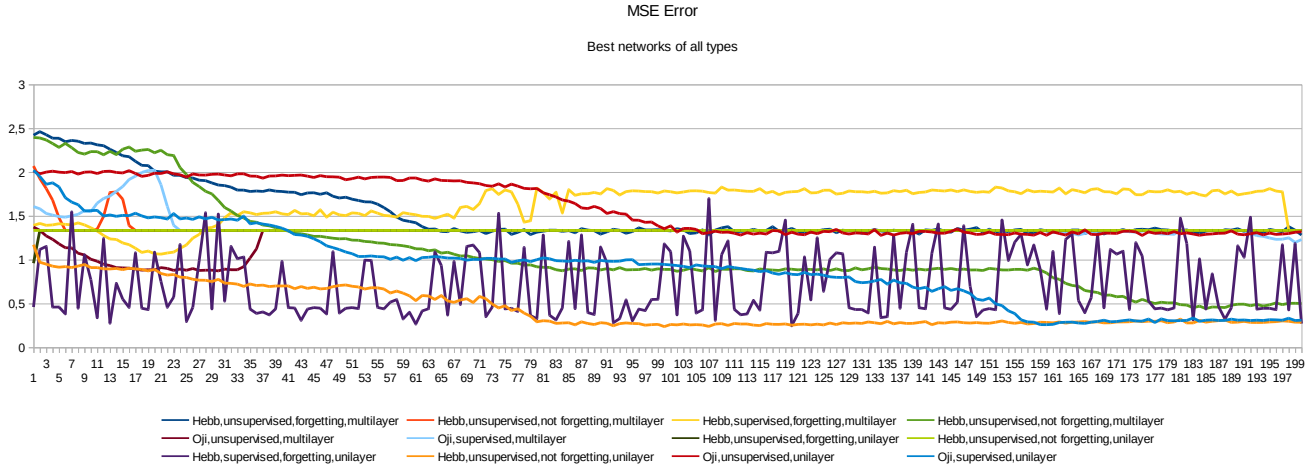
W tym wariancie próbowano wykorzystać sieci Hebba i Oji w wariantach z pojedynczą lub wieloma warstwami, nauczycielem i bez, zapominaniem i bez, do nauki rozpoznawania otoczenia – tunelu, ściany z prawej i lewej. Okazuje się, że sieci te mogą w miarę dobrze nauczyć się radzić sobie z tym zagadnieniem, i to, przynajmniej na pierwszy rzut oka, niezależnie od przyjętej konfiguracji.

Warianty z siecią jednowarstwową

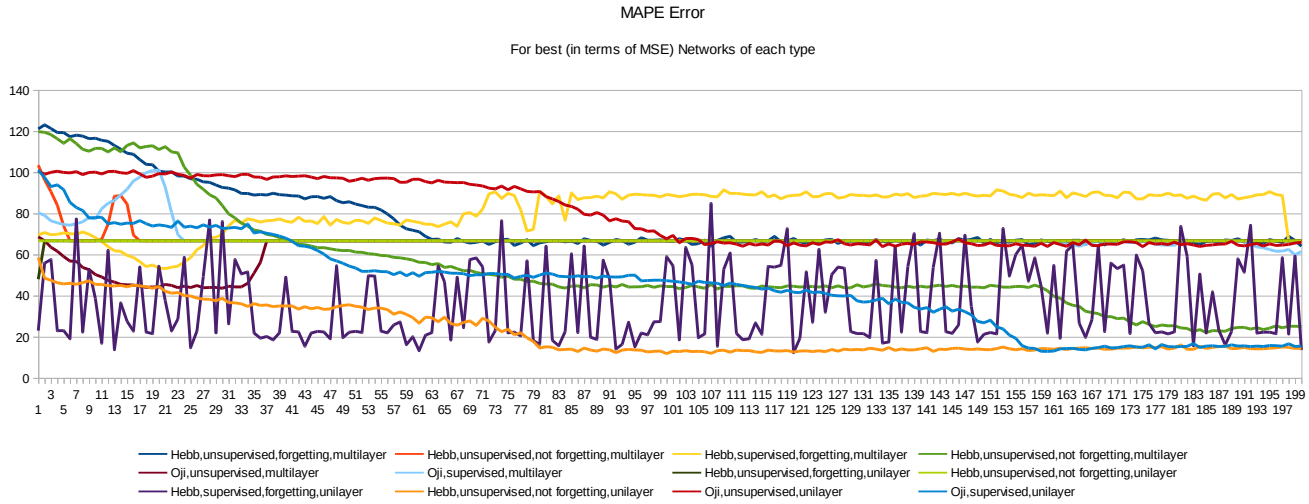
Warstwa	Liczba neuronów
Wejściowa	16
Wyjściowa	3

Szymon Durak, PSI 2016-2017, sprawozdanie z projektu
Warianty z siecią wielowarstwową

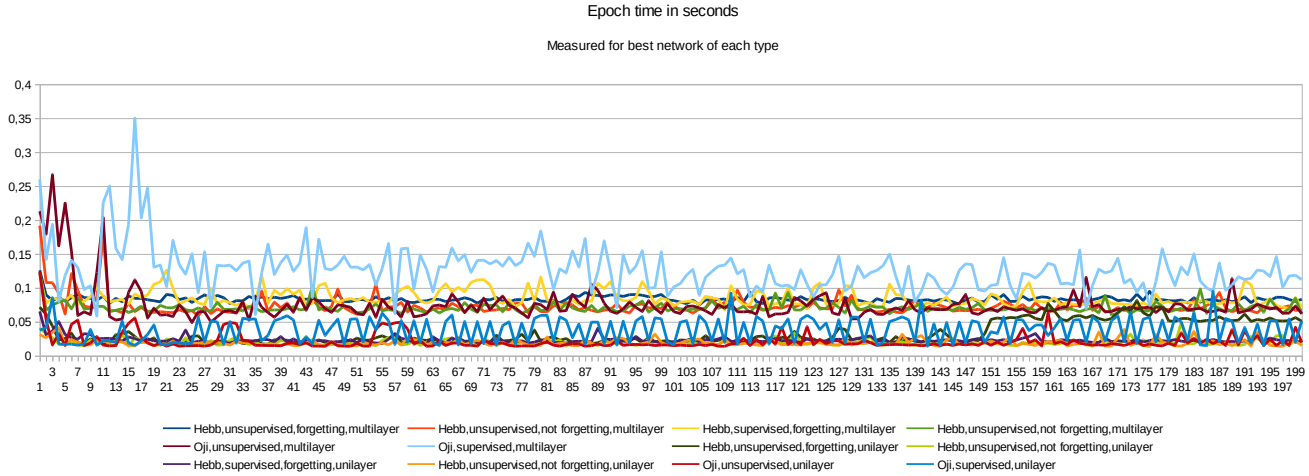
Warstwa	Liczba neuronów
Wejściowa	16
Ukryta	8
Wyjściowa	3



Niestety, wszystkie rodzaje sieci wykazały tendencję do przyjmowania „błędnego” kierunku uczenia – zamiast rozpoznawać jakiś konkretny rodzaj otoczenia, nieraz nie wykrywały żadnego, przez co błąd zbiegał do 1,(3).



Z drugiej strony, okazuje się, że sieci, które nie wpadły w ową „pułapkę”, osiągnęły poziom błędów rzędu 20%. Niestety, błąd ten nadal jest zbyt wysoki, by znaleźć dla nich praktyczne zastosowanie.



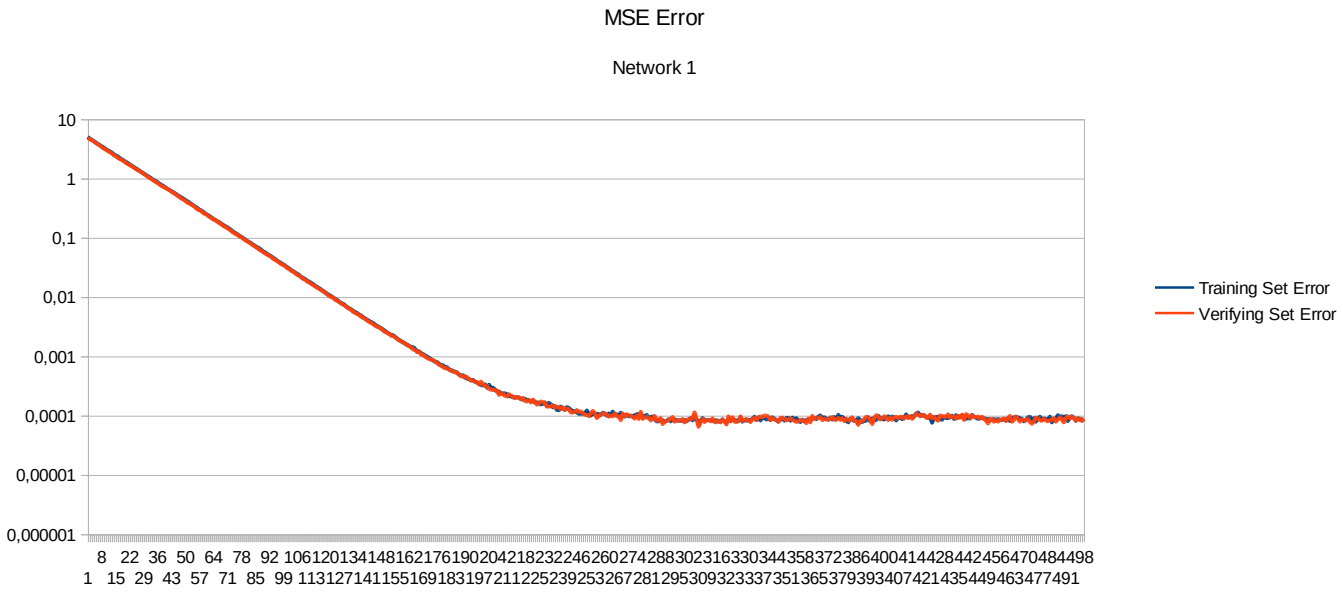
Jak widać, sieci jednowarstwowe uczyły się w znacznie krótszym czasie – nakłady obliczeniowe na uczenie sieci wielowarstwowej są zdecydowanie wyższe. Należy zauważyć, że wprowadziliśmy tylko jedną warstwę ukrytą, a mimo to znacznie wzrósł czas uczenia.

Sieci WTA

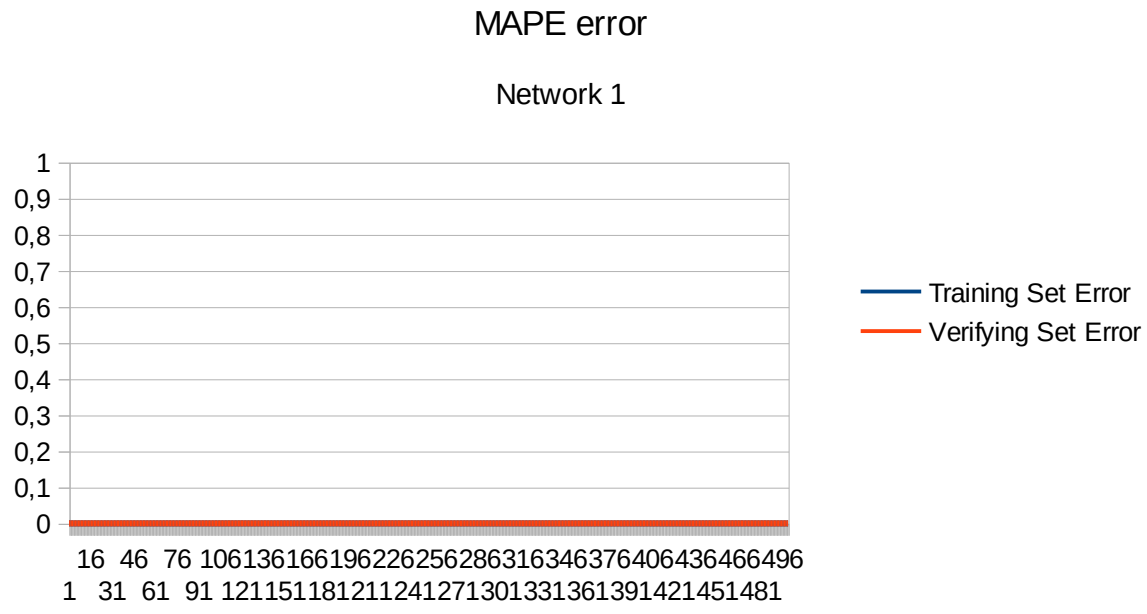
Po zaimpementowaniu sieci WTA dokonano próby rozwiązania powyższego problemu przy użyciu tej sieci. Stosując te same dane wejściowe i taką samą strukturę sieci:

Warstwa	Liczba neuronów
Wejściowa	16
Wyjściowa	3

Sprawdzono, czy sieć WTA pozwoli uzyskać pożądane wyniki grupowania:

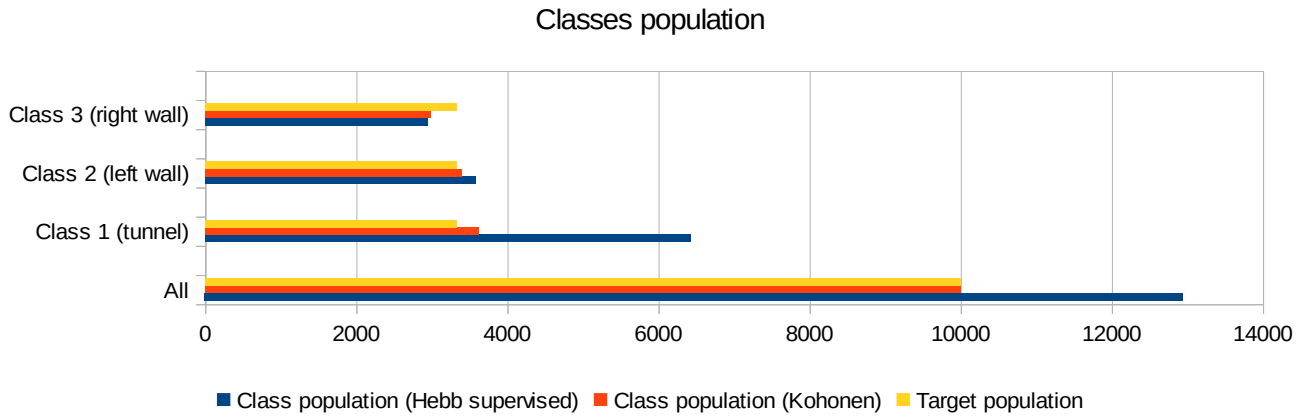


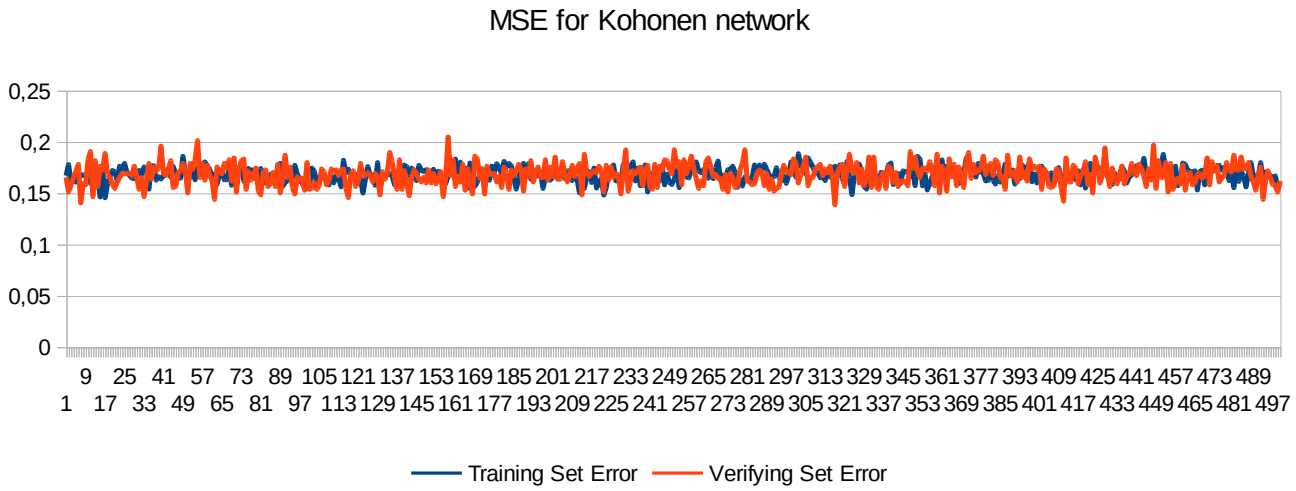
Jak się okazało, dla znormalizowanych wyników sieć uzyskała błąd MSE na poziomie 0,0001, o wiele lepszy, niż w przypadku sieci Hebba. Niestety, zaginęły informacje o błędzie MAPE, być może ze względu na sposób obliczania błędu:

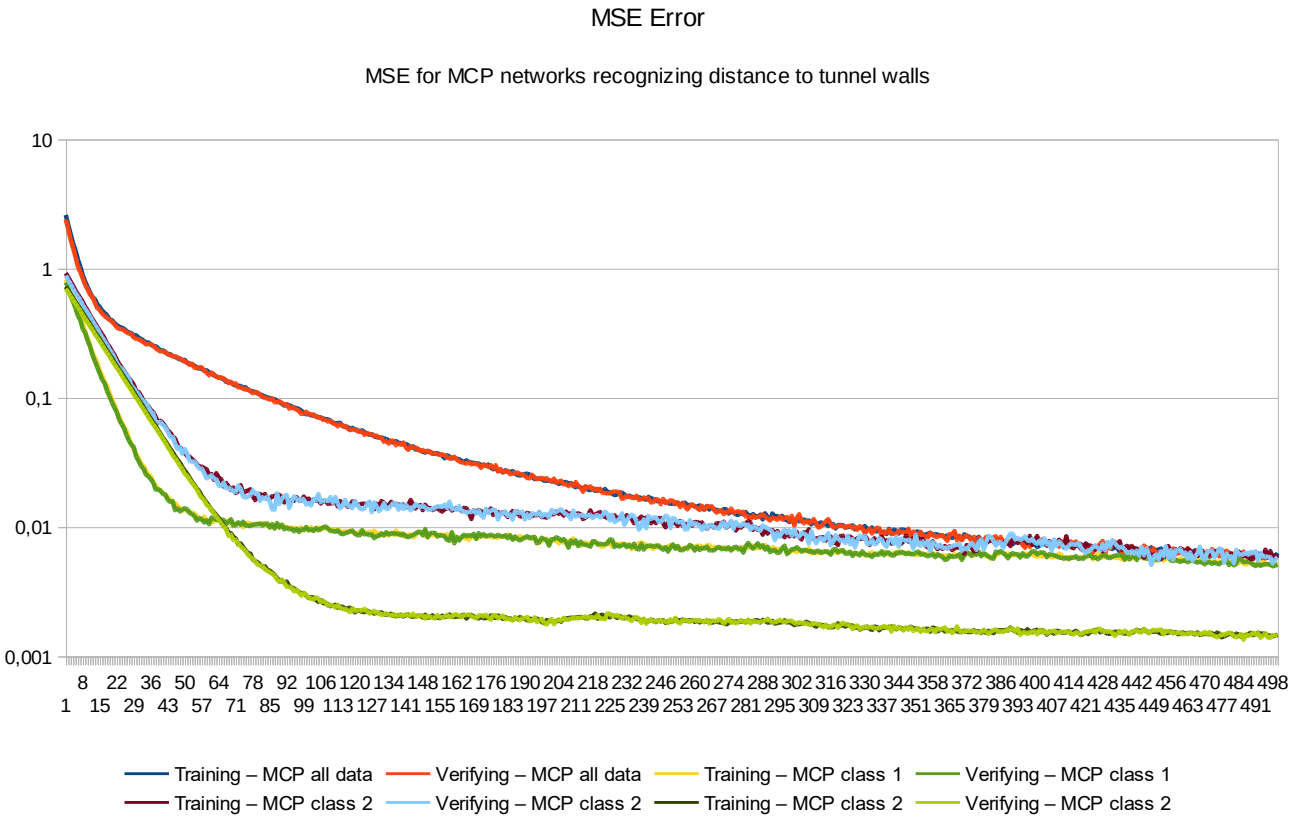


Trudno powiedzieć, co jest przyczyną takiego stanu rzeczy.

Następnie, wykorzystano sieć do pogrupowania danych na podzbiory, i dopiero na tych podzbiorach przeprowadzono uczenie sieci MCP i porównano dodatkowo z uczeniem Hebba:





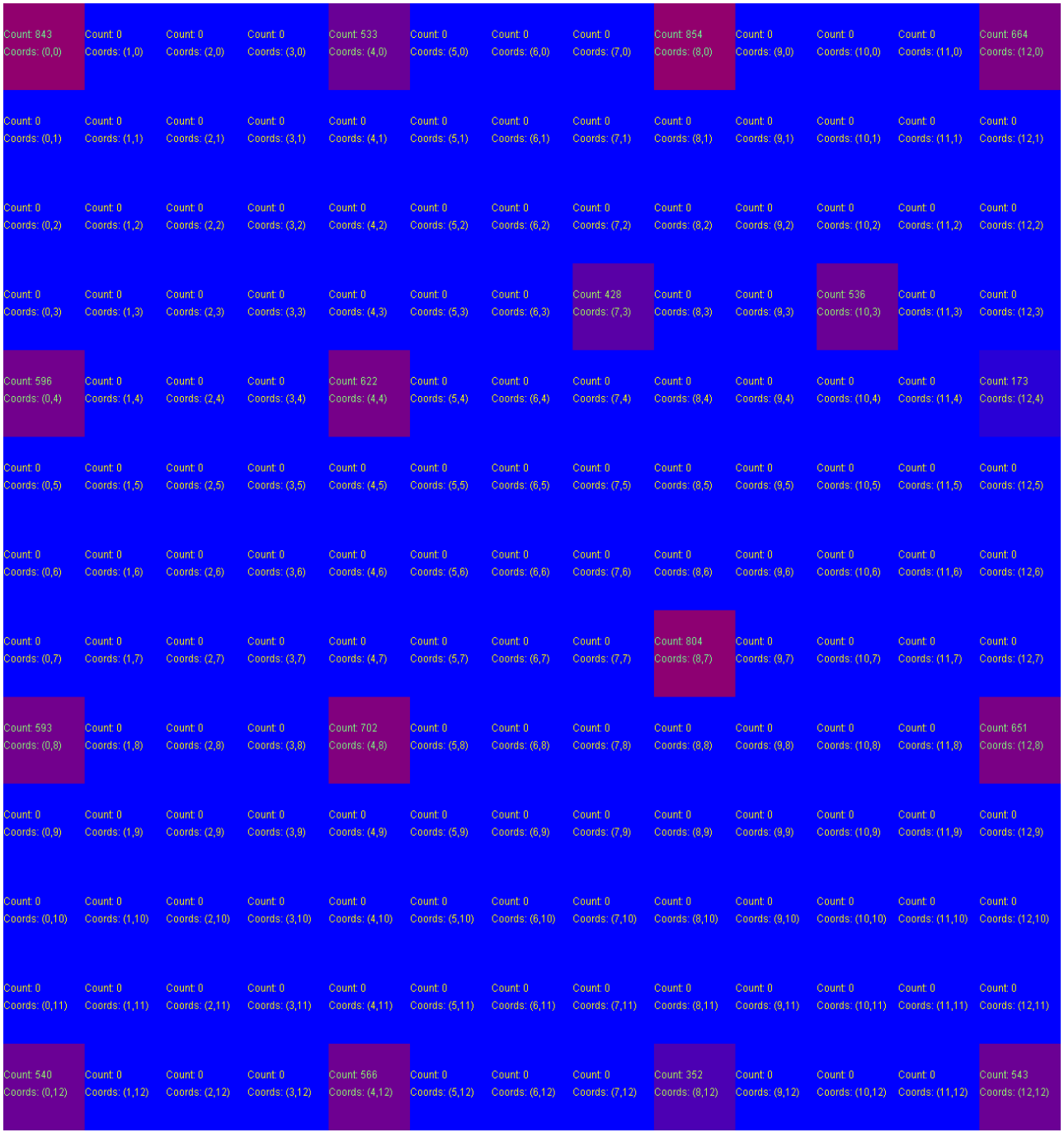


Szymon Durak, PSI 2016-2017, sprawozdanie z projektu
Jak widać, pogrupowanie danych wejściowych pozwoliło na uzyskanie nieco lepszych wyników niż dla wymieszanych danych, aczkolwiek wyniki nadal nie były zadowalająco precyzyjne.

Sieci SOM

W ostatnim ze zrealizowanych podprojektów rozważono problem klasyfikacji danych i organizacji zebranych danych w mapy neuronów reprezentujące zebrane informacje. Rozważono przypadek, w którym dane z 16 neuronów wejściowych chcemy odwzorować na mapę 13x13 neuronów sieci SOM. Dane wejściowe dzieliły się na 16 podzbiorów, z których każdy zawierał odczyty czujników dla przeszkód ustawionych przed wybranym z czujników – przeszkoda była na tyle szeroka, że więcej niż 1 czujnik emitował sygnał, w efekcie „sąsiednie" dane były do siebie podobne. Postanowiono sprawdzić, w jaki sposób i w jakiej liczbie klastrów zostaną odwzorowane takie dane wejściowe. Dla 10000 danych i 16 podzbiorów, w idealnej sytuacji każda klasa powinna liczyć 625 elementów.

Warstwa	Liczba neuronów
Wejściowa	16
Wyjściowa	13x13=169



Jak się okazało, uzyskana mapa rzeczywiście posiadała 16 wyraźnych skupień, przy czym jedno było rozbite na 2 niezbyt odległe od siebie, przy czym jedno z nich wyraźnie dominowało w kwestii liczebności. Liczebność klas na mapie wahała się między ok. 350-800, zatem możemy uznać, że przy danej specyfice danych wejściowych, uzyskano dość dobre wyniki.

Uwagi

- W projekcie zastosowano własną implementację wszystkich sieci neuronowych
- Projekt napisano w języku Java, z wykorzystaniem Maven i środowiska IntelliJ IDEA
- Ze względu na duże rozmiary logów (standardowo uczono po ok. 10-100 sieci przez 100-500 epok), nie zamieszczono ich w niniejszym sprawozdaniu. Można je znaleźć w katalogach *wyniki_1, wyniki_2*… itd.
- Dane wejściowe dla problemów znajdują się w katalogu *learning_set*. Jest ich wiele ze względu na fakt, że specyfika kolejno implementowanych sieci często wymagała określenia nowych danych wejściowych lub połączenia ich z wynikami oczekiwanymi. Dlatego w niektórych plikach dane wejściowe mają dodatkowe rekordy, np. 16+3
- Do logów należą przede wszystkim pliki MSE.txt, MAPE.txt i TIME.txt, zawierające pomiary czasu uczenia oraz błędy dla kolejnych epok
- Szersze wnioski do uzyskanych wyników można znaleźć w arkuszach kalkulacyjnych i/lub plikach wnioski.txt
- Czasy uczenia dla poszczególnych sieci dla tego samego zagadnienia mogą znacząco różnić się od siebie. Jest to spowodowane faktem, że ze względu na ciągle rosnące czasy uczenia wprowadzono współbieżne uczenie wielu sieci, natomiast liczba uczonych sieci na ogół nie była wielokrotnością liczby dostępnych procesorów logicznych. W efekcie niektóre sieci nie musiały współdzielić zasobów procesora z innymi i uzyskiwały około dwukrotnie krótsze czasy uczenia