Metody numeryczne w elektrotechnice

Ćwiczenie 13 — Sieci neuronowe

Dariusz Borkowski

9 czerwca 2012

Wstęp

Zapoznasz się ze sposobem projekowania i uczenia podstawowych sieci neuronowych.

1 Aproksymacja liniowa — 2 pkt

Najprostszą siecią liniową będzie pojedynczy neuron z liniową funkcją przejścia posiadający jedno wejście x, jedną wagę w, jedną stałą b i jedno wyjście, opisany równaniem

$$\hat{y} = wx + b \tag{1}$$

Widać więc, że realizuje on funkcję liniową, więc nie nadaje się do aproksymacji wyszukanych kształtów. Sprawdź co będzie jeśli nauczysz taką sieć na podstawie trzech par liczb (x, y) nie leżących na jednej linii.

1.1 Aproksymacja liniowa funkcji nieliniowej 1D

Napisz program, który aproksymuje funkcję $y = 0.05x^2 + 0.3x + 2$, za pomocą prostej sieci złożonej z jednego neuronu. Narysuj przebieg aproksymowanej funkcji dla wektora liczb `xx = 1:0.1:8`.

Wybierz 3 punkty o współrzędnych x równych 2; 4 i 7. Współrzędne x wybranych punktów będą tworzyć wierszowy wektor uczący o nazwie `inputs`, a wartości funkcji aproksymowanej w tych punktach będą tworzyć wektor zadanych wyjść z sieci (cel) o nazwie `targets`.

Za pomocą polecenia `net = newlind(inputs,targets)` wygeneruj sieć neuronową. W przypadku tak prostej sieci liniowej podczas generacji zostaje od razu przeprowadzony proces jej nauki. Jest to wyjątek jeśli chodzi o sieci neuronowe, wynikający z ogromnej prostoty tego typu sieci. Zobacz jakiego typu jest nowo utworzona zmienna `net` i jakie informacje na jej temat możesz uzyskać w Command window. Wyświetl na ekranie wartości wag oraz stałych (są one jednymi z ostatnich elementów struktury `net`). Waga w i stała b są współczynnikami liniowej funkcji aproksymującej danej wzorem (1). Porównaj je ze współczynnikami funkcji aproksymowanej (nie da się tego zrobić dosłownie, bo funkcje są różnych rzędów).

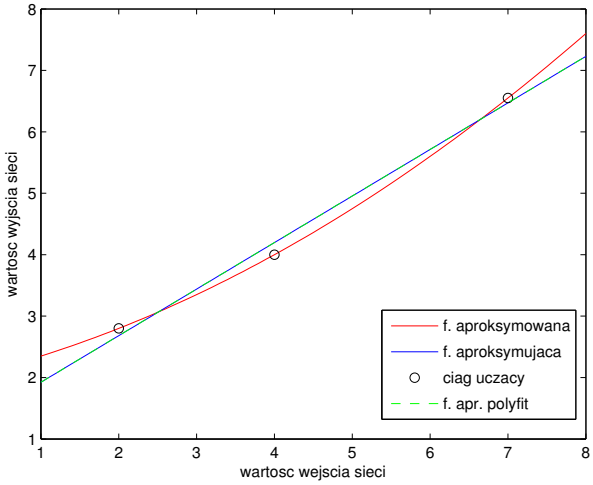
Poleceniem `sim` zastosowanym na strukturze `net` opisującej sieć wygeneruj wartości wyjścia sieci dla poprzednio użytego wektora `xx` i narysuj je na wspólnym wykresie z funkcją aproksymowaną. Narysuj dodatkowo na wykresie położenia punktów uczących. Widać, że odpowiedź sieci jest jedynie przybliżeniem kształt funkcji aproksymowanej y.

Uczenie sieci liniowych (podobnie jak perceptronów) polega na minimalizacji błędu średniokwadratowego:

$$e = \frac{1}{K}\sum_{k=1}^{K} e_k^2 = \frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2 = \frac{1}{K}\sum_{k=1}^{K}(y_k - wx_k - b)^2 \tag{2}$$

gdzie K jest długością ciągu uczącego, tu ilością par liczb (x, y), y jest zadaną wartością wyjścia sieci, a $\hat{y}$ faktyczną wartością wyjścia sieci np. podczas nauki lub po jej zakończeniu. Tak zdefiniowany błąd przy liniowej funkcji przejścia neuronu powoduje, że minimalizowana funkcja błędu jest funkcją kwadratową n zmiennych (tutaj dwóch), dlatego minimalizacja może być zrealizowana w jednym kroku poprzez rozwiązanie układu równań liniowych (liniowy estymator LS — nie wymaga iteracji).

Sprawdź czy rezultat jest faktycznie aproksymacją zadanej funkcji w sensie najmniejszych kwadratów przez porównanie z wynikami aproksymacja wielomianem 1–go stopnia (funkcja `polyfit` lub obliczonym ręcznie wg. (1)) dla użytych wcześniej trzech punktów czyli węzłów aproksymacji.

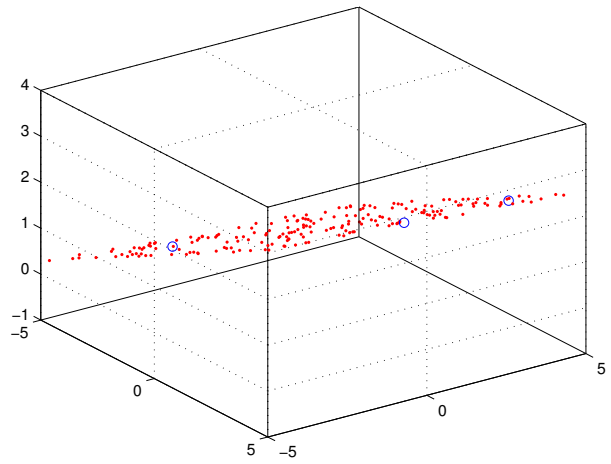


Rysunek 1: Wyniki aproksymacji za pomocą sieci liniowej z jednym neuronem.

1.2 Aproksymacja liniowa danych w przestrzeni 3D

Teraz wygeneruj zestaw trzech punktów (x_1, x_2, y) w przestrzeni 3D. Te trzy punkty (trzy ciągi uczące) definiują jednoznacznie płaszczyznę w przestrzeni 3D czyli funkcję liniową dwóch zmiennych (x_1, x_2).

Analogicznie jak wcześniej utwórz ciągi uczące: macierz `Inputs` o 3 kolumnach i dwóch wierszach (każda kolumna zawiera dwie współrzędne x_1 i x_2) oraz wektor wierszowy `Targets` zawierający trzy zadane wartości y. Utwórz nową sieć liniową o nazwie `net2` za pomocą polecenia `linearlayer` podając wartość 0 za wektor opóźnień i wartość 0.01 jako współczynnik nauki. W tym wypadku sieć ta nie będzie nauczona, a jej początkowe wagi oraz stałe są ustawione na zero. Przygotuj sieć do nauki poleceniem `preparenets`, które przyjmuje nazwę sieci, nazwę macierzy wejść i wektora zadanych wyjść. Zadaj dopuszczalny błąd nauki równy 0.5 wpisując tą wartość do pola `trainParam.goal` struktury `net2`. Następnie naucz sieć poleceniem `train` podając ciągi uczące `Inputs` i `Targets` i dokonaj jej symulacji (polecenie `sim`) dla zbioru 200 punktów o wartościach (x_1, x_2) wylosowanych z zakresu -5 do 5. Narysuj je na wykresie razem z punktami uczącymi za pomocą polecenia `plot3`, które jest trójwymiarową wersją polecenia `plot`.



Rysunek 2: Wyniki aproksymacji funkcji dwóch zmiennych za pomocą sieci liniowej z jednym neuronem dwuwejściowym.

Sprawdź obracając kursorem trójwymiarowy wykres czy trzy punkty ciągu uczącego leżą w tej samej płaszczyźnie co wyniki symulacji. Następnie zmień dopuszczalny błąd nauki na 1e-6 i powtórz naukę sieci,

symulację, a w końcu rysowanie wyników. Sprawdź czy teraz ciągu uczącego leżą w tej samej płaszczyźnie co wyniki. Czy zauważyłeś różnice w czasie nauki sieci?

2 Aproksymacja nieliniowa — 3 pkt

Do aproksymacji nieliniowej jedno lub wielowymiarowych funkcji nieliniowych należy użyć sieci o nieliniowych (np. sigmoidalnych) funkcjach przejścia neuronów w warstwie (lub warstwach) ukrytej oraz o liniowej funkcji przejścia neuronów w warstwie wyjściowej.

Aby taka sieć działała dobrze dane uczące powinny być spójne czyli powinny reprezentować jakąś sensowną zależność (wzór), a liczba neuronów warstwy ukrytej nie może być zbyt mała.

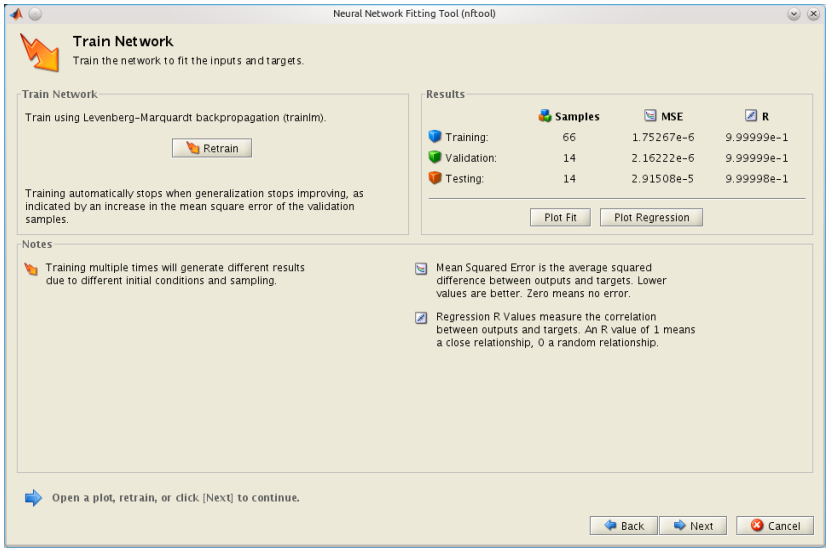
Na początek zapoznasz się z graficznym narzędziem rozwiązywania problemów dopasowania funkcji (function fitting) za pomocą sieci neuronowej. Wykorzystasz gotowe zestawy uczące dostępne w Matlabie. Następnie sprawdzisz możliwości aproksymacji funkcji skoku 1D o kształcie trudnym do aproksymacji metodami klasycznymi czyli np. za pomocą wielomianów.

2.1 Wpływ liczby neuronów na wyniki aproksymacji 1D

Uruchom w Command window narzędzie `nftool`. Kliknij `Next` i na kolejnym ekranie naciśnij przycisk `Load Example Data Set`. Na początek zaimportuj zestaw danych nazwany „Simple Fitting Problem", który zawiera 67 punktów (x, y) przebiegu pewnej funkcji $y = f(x)$ jednej zmiennej, która ma być aproksymowana za pomocą sieci neuronowej. Zauważ, że w polu `Inputs` pojawiła się nazwa zmiennej `simplefitInputs`, w której podane są ciągi uczące wejścia sieci czyli wartości x funkcji aproksymowanej. W polu `Targets` pojawiła się nazwa zmiennej `simplefitTargets`, która zawiera pożądane wartości wyjścia sieci przy zadanych pobudzeniach czyli wartości y funkcji aproksymowanej odpowiadające wartościom x wejść.

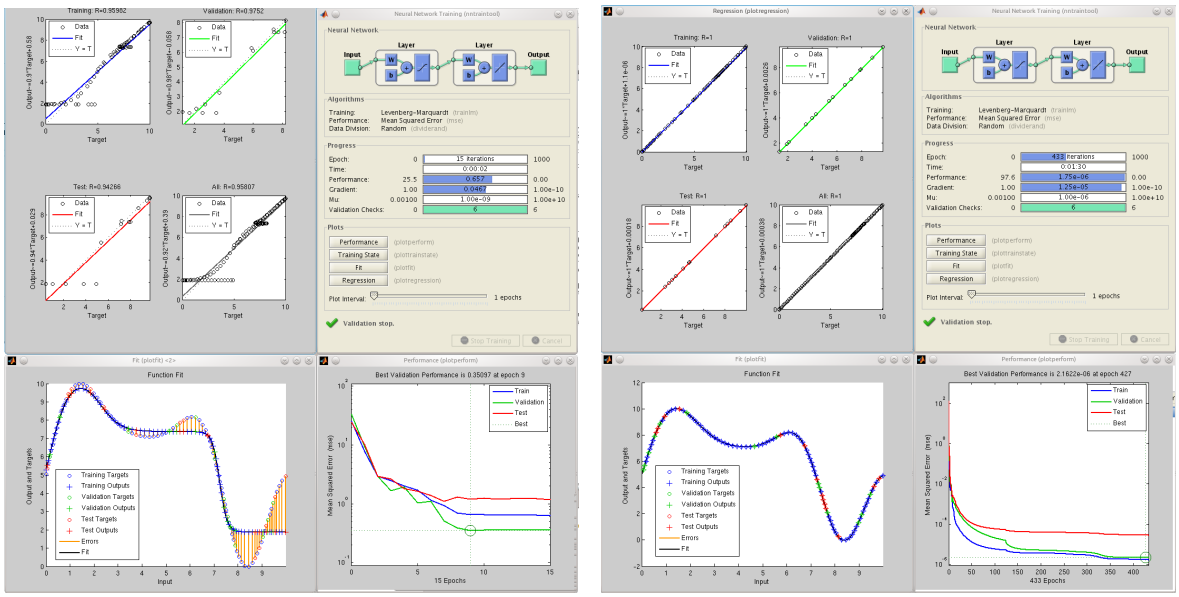
Przejdź do kolejnego kroku klikając `Next`. Teraz można ustalić podział danych uczących na trzy zestawy: uczący (domyślnie 70% wszystkich danych), weryfikujący (domyślnie 15% danych), testujący (domyślnie 15% danych). Obok wyjaśniono zastosowania poszczególnych rodzajów danych. Pozostaw domyślne wartości.

Przejdź do kolejnego kroku klikając `Next`. Ten ekran prezentuje strukturę projektowanej sieci i pozwala na określenie liczy neuronów w warstwie ukrytej (domyślnie 10). Na razie pozostaw wartość domyślną i przejdź dalej, klikając `Next` do ekranu uczenia i kliknij `Train`, aby rozpocząć naukę sieci.



Rysunek 3: Jeden z ekranów narzędzia `nftool`, pokazane wyniki etapów uczenia sieci.

W nowo otwartym oknie (rys. 4) pokazany jest przebieg nauki: numer epoki, czas, ocenę skuteczności nauki (wartość kryterium jakości nauczania), wartość normy gradientu (przy metodach gradientowych),



Rysunek 4: Wyniki uczenia sieci aproksymującej funkcję nieliniową: 3 neurony w warstwie ukrytej (po lewej); 10 neuronów w warstwie ukrytej (po prawej). W każdej części są pokazane 4 okna: lewe–górne pokazuje dopasowania sieci w różnych etapach nauki i testowania do danych jako liniową zależność faktycznego wyjścia Y od wartości zadanej T; prawe–górne to okno nauki sieci pokazujące wyniki w kolejnych iteracjach, lewe–dolne to wynik aproksymacji funkcji (ten wykres da się otworzyć tylko dla aproksymacji funkcji 1D); prawe–dolne to zmiany wskaźnika jakości nauki w kolejnych iteracjach nauki.

wartość stałej uczenia oraz liczbę kroków walidacji sieci. Podane są także rodzaj algorytmu uczenia, zastosowane kryterium jakości nuczenia oraz sposób podziału danych do nauki. Jak widać w narzędziu graficznym `nftool` struktura sieci, algorytm uczenia, kryterium jakości nie mogą być zmieniane przez użytkownika.

Kliknij przycisku `Fit` (rysowanie w trakcie nauki dopasowania do funkcji aproksymowanej). Sprawdź do jak niskiej liczby neuronów w warstwie ukrytej można dojść, aby aproksymacja funkcji była w miarę dokładna . Rzetelne sprawdzenie tego może wymagać np. kilkukrotnego powtórzenia nauki dla tej samej liczby neuronów. Wynika to z faktu, że wagi neuronów na początku iteracji są inicjowane losowymi wartościami, dlatego powtarzanie nauki z losowaniem wag, za każdym razem da nieco inaczej pracującą sieć neuronową. Czasami po procesie nauki sieć będzie źle nauczona, a po ponownym losowaniu wag i powtórzeniu nauki może działać dobrze.

Na koniec przygotuj 1000-elementowy ciąg uczący zawierający w zmiennej `myTargets` wartości funkcji podwójnego skoku danej wzorem

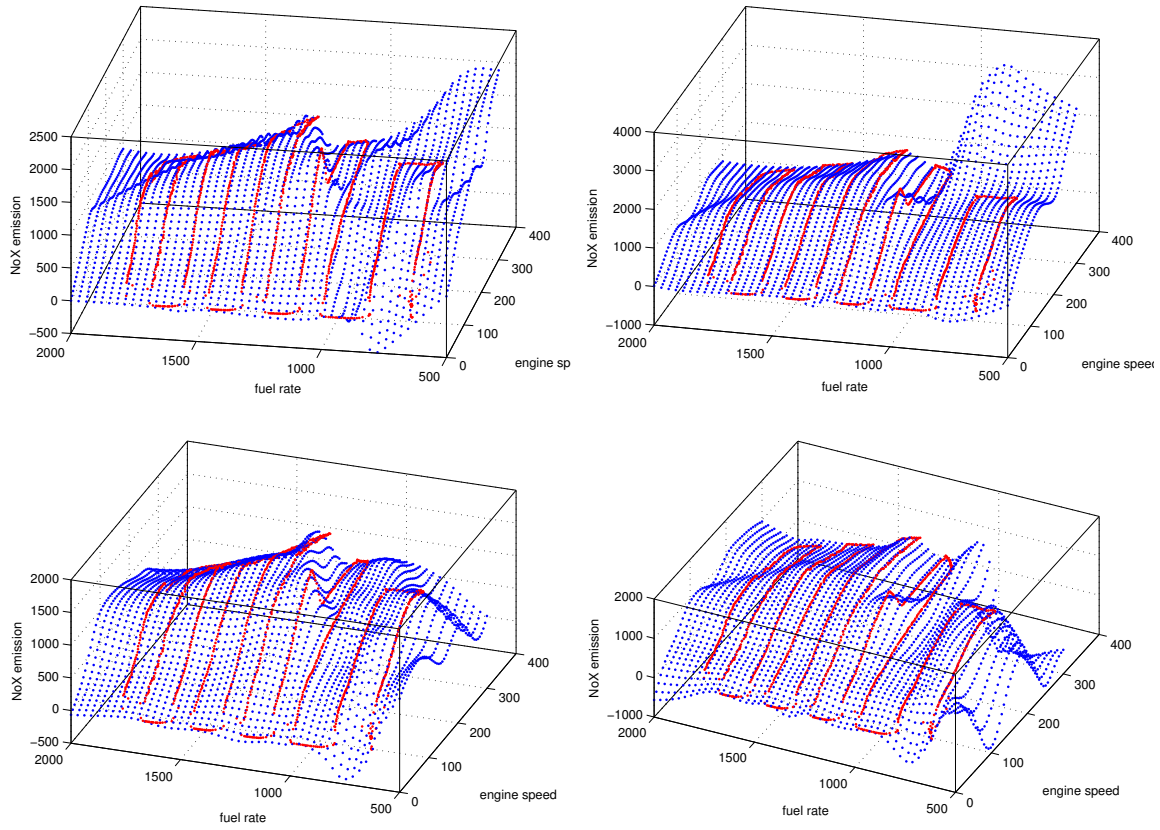
$$y = \begin{cases} 1 & -1 < x < 0 \\ 0 & x \leqslant -1 \vee 0 \leqslant x \end{cases} \tag{3}$$

w przedziale x od -2 do 2, a w `myInputs` wartości x. Do wygenerowania wartości funkcji wykorzystaj operator porównania $<$ zwracający tylko wartości 0 lub 1 (prawda lub fałsz). Następnie przeprowadź naukę sieci i sprawdź jaka jest najmniejsza liczba neuronów w warstwie ukrytej dająca sensowne wyniki.

2.2 Jednoczesna aproksymacja dwóch funkcji 2D

Uruchom ponownie narzędzie `nftool` i wczytaj zestaw uczący o nazwie Engine. Zawiera on w ciągu uczącym 2 zmienne: w pierwszym wierszu zużycie paliwa, a w wierszu drugim prędkość obrotową. Dane wyjściowe również zawierają 2 zmienne: moment obrotowy w pierwszym wierszu oraz emisję tlenków azotu w drugim wierszu. Oznacza to, że nauczona sieć będzie podwójnym aproksymatorem jednocześnie dwóch funkcji dwóch zmiennych czyli momentu i emisji NoX. Sieć taka oprócz dwóch wejść, pewnej liczby neuronów warstwy ukrytej posiada dwa neurony w warstwie wyjściowej. Jeden z nich zwraca przybliżenie momentu, a drugi zwraca przybliżenie emisji NoX. Aby dało się generować wartości z całego przedziału liczb rzeczywistych, neurony warstwy wyjściowej mają liniowe funkcje przejścia.

Przejdź przez wszystkie ekrany narzędzia `nftool`, wykonując po drodze naukę (trening) sieci, aż do ostatniego. Na ostatnim ekranie zatytułowanym Save result wybierz przycisk `Generate M-file`. Matlab wygeneruje funkcje, która realizuje kroki wykonane w narzędziu `nftool` razem z wybranymi opcjami. Zmień ten plik z funkcji na zwykły skrypt, w którym na początku załadujesz zestaw Engine poleceniem `load engine_dataset`. Dopisz do niego rysowanie wszystkich punktów ciągu uczącego (t.j. wartości emisji NoX w funkcji prędkości i zużycia paliwa) za pomocą czerwonych kropek w przestrzeni 3D. Realizuje się to poleceniem `plot3`. Potem wygeneruj ciągi po 50 równomiernie rozłożonych wartości prędkości obrotowej w zakresie od 500 do 2000 w zmiennej `es` i zużycia paliwa od 0 do 300 w zmiennej `fr` za pomocą polecenia `linspace`. Następnie (np. za pomocą podwójnej pętli `for`) wygeneruj ciąg wejściowy sieci zawierający wszystkie kombinacje wygenerowanych wartości `es` i `fr` (pełna siatka punktów wejściowych) i zapisz je jako dwuwierszową macierz `IN`. Oblicz wartości wyjść sieci `OUT` dla wszystkich punktów na siatce za pomocą polecenia `sim(net,IN)`. Narysuj te punkty na poprzednim wykresie innym kolorem (jak na rysunku 5). Zobacz jak przy kolejnych powtórzeniach nauki zmiania się kształt funkcji aproksymującej dane (ciąg uczący).



Rysunek 5: Przybliżenie emisji NoX w funkcji predkości obrotowej i szybkości zużycia paliwa. Czerwone punkty to wartości ciągu uczącego (jednakowe na każdym wykresie). Niebieskie punkty to aproksymacja emisji NoX obliczona przez sieć we wszystkich punktach przygotowanej siatki. Oba górne oraz lewy dolny to wyniki dla 16 neuronów warstwy ukrytej, lecz różne realizacje sieci (wartości wag) uzyskane podczas niezależnych procesów uczenia. Prawy dolny to wynik dla 25 neuronów warstwy ukrytej.

Wpisz w Command window nazwę zmiennej zawierającą sieć (prawdopodobnie nazywa się `net`) i zobacz jak jest zbudowana. Wyświetl na ekranie wagi neuronów i wartości stałe (pola IW, LW i b). Wyjaśnij znaczenie i rozmiary dwóch pierwszych.

3 Rozpoznawania wzorców (klasyfikacja) — 2 pkt

W tym ćwiczeniu nauczysz kilka rodzajów sieci automatycznej klasyfikacji wyników badań medycznych.

3.1 Klasyfikacja siecią uczącą pod nadzorem (z nauczycielem)

Do klasyfikacji stosuje się sieci o wielu wyjściach (tyle wyjśc ile klas). Zazwyczaj sieci te są perceptronami czyli sieciami posiadającymi w warstwie wyjścia neurony o nieliniowej funkcji przejścia z nasyceniem (limits). Mogą to być perceptrony progowe (funkcja przejscia hard limiting czyli skokowa) lub zwykłe (funkcj przejścia sigmoidalna np. logsig).

W percpetronie progowym zawsze tylko jedno wyjście powinno przyjmować wartość 1, a pozostałe wartości 0. Oznacza to, że dany wzorzec (wektor wejściowy x został zaliczony do klasy związanej z tym jednym wyjściem. Przynależność wzorca do klasy jest zatem jednoznaczna. Jest w tej klasie lub nie.

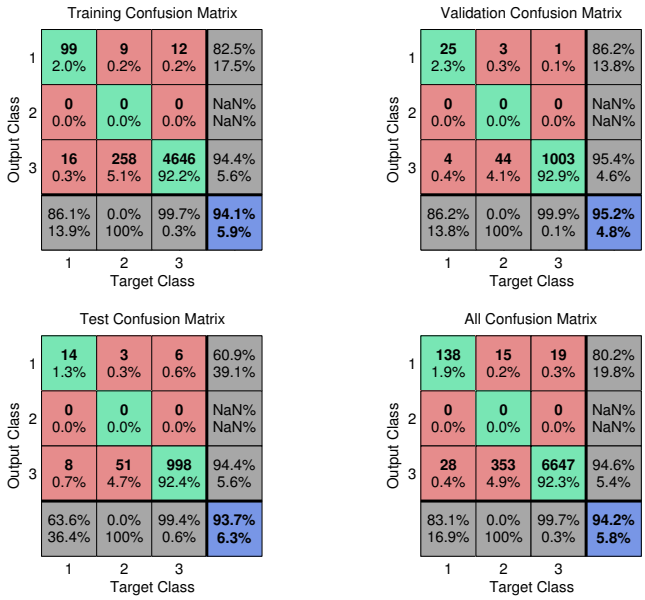
Percpetrony o sigmoidalnych funkcjach przejścia mogą na wyjściach zwracać wszystkie wartości np. z zakresu od 0 do 1. Zatem liczba bliska 1 (np. 0.96) na wyjściu n–tym , podczas gdy na pozostałych są wartości bliskie 0 oznacza, że sieć uznała ten wektor wejściowy za najlepiej pasujący do klasy numer n, a znacznie mniej pasujący do klas pozostałych. Są to właściwości zbliżone do idei logiki rozmytej, gdzie przynależność elementu do zbioru określa się nie binarnie ale w sposób ciągły (np. procentowy). Uwaga, w Matlabie perceptronem nazywane są sieci o skokowych funkcjach przejścia!

Uruchom narzędzie `nprtool` służące do tworzenia i nauki sieci jednokierunkowe do klasyfikacji danych na podstawie ciągów uczących i zadanych wartości wyjść. Jeżeli podczas nauki używamy znanych wartości żądanych wyjść sieci to jest to tzw. nauka po nadzorem lub z nauczycielem (supervised learning).

Zaimportuj zestaw danych nazwany Thyroid. Zestaw ten zawiera wyniki badań 7200 pacjentów (każdy pacjent opisany jest przez 15 danych binarnych i 9 wartości ciągłych) pod kątem funkcjonowania tarczycy, podzielone na 3 klasy: prawidłowa praca tarczycy, nadczynność tarczycy i niedoczynność tarczycy. Wartości zadane czyli wyniki diagnozy lekarskiej zawierają zatem 3 wiersze odpowiadające trzem opisanym klasom. Każdy rekord 3 elementowy (związany z pojedyczym pacjentem) składa się z dwóch zer i jedynki. Położenie jedynki oznacza przynależność do danej klasy. Najwięcej rekordów opisuje przypadki niedoczynności tarczycy.

Przejdź dwa ekrany dalej, do ekranu obrazującego strukturę sieci. Wyjaśnij liczbę wejść i wyjść sieci. Zwróć uwagę na wybrane funkcje przejścia. Co może wynikać z kształtu przyjętej funkcji przejścia w warstwie wyjściowej?

Przejdź do następnego ekranu i przeprowadź naukę (trening) sieci, a następnie zobacz postać macierzy pomyłek (przycisk `Plot Confusion`). Wyjaśnij znaczenie poszczególnych pól.



Rysunek 6: Macierze błędów uczenia.

Przejdź do ostatniego ekranu i wyeksportuj sieć do Matlab Workspace za pomocą Save Results po zaznaczeniu odpowiednich zmiennych do eksportu. Wyjdź z `nprtool`.

Poleceniem `sim` przeprowadź symulację sieci podając na wejścia wszystkie elementy ciągu uczącego `thyroidInputs`. Wyniki klasyfikacji pacjentów (wyjścia sieci) zapisz w zmiennej `Outputs`. Wyrysuj pole-

ceniem `plot` poszczególne wiersze macierzy `Outputs`. Zauważ, że nie zawierają one wartości binarnych. Co według Ciebie oznaczają zatem wartości zwracane przez sieć?

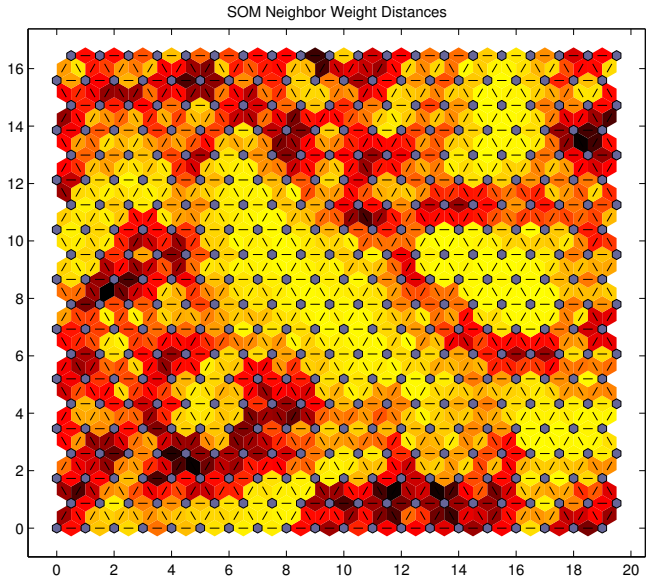
3.2 Sieć samoorganiuzująca uczona bez nadzoru

Możliwe jest też przeprowadzenie automatycznego grupowania danych na podstawie wzajemnego podobieństwa (clustering) jeżeli nie znamy zadanych wartości wyjść sieci. Jest to także zagadnienie klasyfikacji zbliżone rozpoznawania wzorców. Stosuje się do tego sieci Kohonena czyli tak zwane samoorganiuzujące się mapy (SOM — self organizing map).

Nauka takiej sieci, o neuronach rozłożonych na siatce 2D (prosokątnej lub heksagonalnej) polega na takiej modyfikacji wag neuronów o wartościach wag najbardziej podobnych do aktualnie podanego na wejścia wzorca, żeby jeszcze bardziej wzmocnić wartość wyjścia tego neuronu przy pobudzeniu danym wzorcem. W ten sposób neurony zaczynają się łączyć w grupy pomiędzy którymi są mniejsze odległości niż między neuronami należącymi do różnych grup. Przez odległość między neuronami rozumie się pewną metrykę różnicy wektorów wag dwóch neuronów czyli np. $||\boldsymbol{w}_i - \boldsymbol{w}_j||$.

Po procesie uczenia sieci można wyróżnić grupy rozpoznane w trakcie uczenia i granice między nimi na wykresie obrazującym odległości między neuronami. Następnie można użyć sieci do grupowania nowych wzorców. Po podaniu wzorca na wejścia sieci należy sprawdzić, do której z wcześniej wyróżnionych grup należy neuron o najsilniejszym wyjściu, do tej grupy sieć przypisuje wzorzec.

Wykonaj polecenie `load thyroid_dataset` w celu załadowania tego samego zestawu danych co w poprzednim punkcie. Tym razem do nauki sieci będą użyte tylko wejścia (Inputs) bez zadanych wartości wyjścia (Targets). Uruchom narzędzie `nctool` służące do projektowania sieci neuronowych typu SOM (samoorganiuzujących) i wybierz ciąg wejściowy o nazwie thyroidInputs. Przejdź do kolejnego ekranu zmień rozmiar mapy neuronów na 20x20 (wpisz wartość 20). Przejdź do następnego etapu i przeprowadź naukę sieci. Następnie kliknij w oknie `nctool` na przycisku `Plot SOM Neighbor Distances` i zinterpretuj wyniki. Jak mają się one do nauki pod nadzorem z poprzedniego punktu?



Rysunek 7: Wykres odległości między neuronami nauczonej sieci. Ciemniejsze kolory oznaczają większe odległości między neuronami, a więc są to granice rozpoznanych klas. Jasne obszary to klasy. W zależności od spostrzegawczości można wyróżnić około sześciu klas.