Sieci neuronowe

Sieci jednokierunkowe wielowarstwowe typu perceptron wielowarstwowy

Matlab

Proces tworzenia i posługiwania się siecią neuronową w środowisku Matlab można podzielić na trzy etapy:

1. Tworzenie modelu sieci
2. Uczenie sieci
3. Symulacja/Testowanie sieci

Pierwszy etap obejmuje proces definiowania parametrów sieci, wynikiem czego jest stworzenie odpowiedniej struktury danych zawierającej opis poszczególnych składników sieci. W etapie tym możliwe jest określenie

1. Liczby neuronów w poszczególnych warstwach
2. Kształtu neuronów w poszczególnych warstwach (domyślnie w Matlabie cała warstwa powinna posiadać identyczną funkcję aktywacji)
3. Funkcji uczącej sieć (algorytm uczenia)
4. Ilości epok uczenia
5. itp.

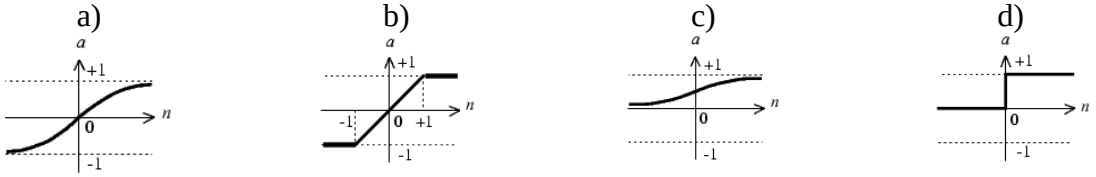
Proces ten realizowany jest poleceniem:

```
net = newff(X,Y,S,T,BTF);
```

gdzie

- net – struktura opisująca sieć
- X –wejścia sieci (uwaga w poziomie cechy, w pionie wektory)
- Y – wyjścia sieci (etykiety wektorów wej.) każdy wiersz opisuje pojedyncze wyjście
- S_i – współczynnik określający ilość neuronów w poszczególnych warstwach oprócz ostatniej (ostatnia warstwa jest zdeterminowana przez liczbę wyjść)
- T – tablica komórkowa zawierająca informację w postaci stringów opisujących kształt funkcji aktywacji dla poszczególnych warstw. Możliwe funkcje to:
  a. tansig – sigmoidalna funkcja aktywacji (+1,-1)
  b. satlins - liniowa funkcja aktywacji z ograniczeniami
  c. logsig - logarytmiczno sigmoidalna sunkcja aktywacji (0, +1)
  d. hardlim – binarna funkcja aktywacji typu y=1(x)

a) b) c) d)



- BTF – algorytm uczenia sieci. Możliwe opcje to:
  - traingd - największego spadku z propagacją wsteczną
  - traingda - największego spadku z adaptacją wsp. uczenia i propagacją wsteczną
  - traingdm - Największego spadku z funkcją momentum z propagacją wsteczną

- traingdx - Największego spadku z momentum i adaptacyjnym wsp. uczenia i propagacją wsteczną
- trainlm - algorytm Levenberg-Marquardt

Po stworzeniu struktury sieci wartości wag w sposób automatyczny są inicjalizowane wg. określonego algorytmu.
Kolejnymi istotnymi parametrami posiadającymi istotny wpływ na proces uczenia sieci są:

```
net.trainParam.epochs = 200; - liczba epok/iteracji uczenia
net.trainParam.showWindow = false; - czy ma być wyświetlane okno uczenia
net.performFcn='mse'; - typ funkcji kosztu
```

Etap drugi to proces uczenia sieci, w którym następuje dopasowanie wag neuronów. Zadanie to – kluczowe realizowane jest za pomocą funkcji:

```
nnet = train(net,X,Y);
```

gdzie

- net – struktura opisująca zainicjowaną sieć
- X – zbiór danych wejściowych (patrz wyżej)
- Y – zbiór danych wyjściowych – informacje które sieć ma się nauczyć (patrz wyżej)
- nnet – struktura sieci opisująca nauczoną sieć nauronową

Ostatnim etapem działania sieci jest proces jej symulacji, w którym następuje wykorzystanie nauczonej sieci w celu predykcji czyli wyznaczenia wartości wyjściowych dla określonych wejść. Etap ten realizowany jest poprzez polecenie:

```
D = sim(nnet,Xt);
```

Gdzie

- nnet – jest strukturą opisującą nauczoną sieć
- Xt – jest macierzą opisującą dane dla których nasza sieć ma dokonać predykcji, czyli odpowiedzieć jakie powinno być wyjście z sieci
- D – odpowiedź sieci

Zadania

Zad 1

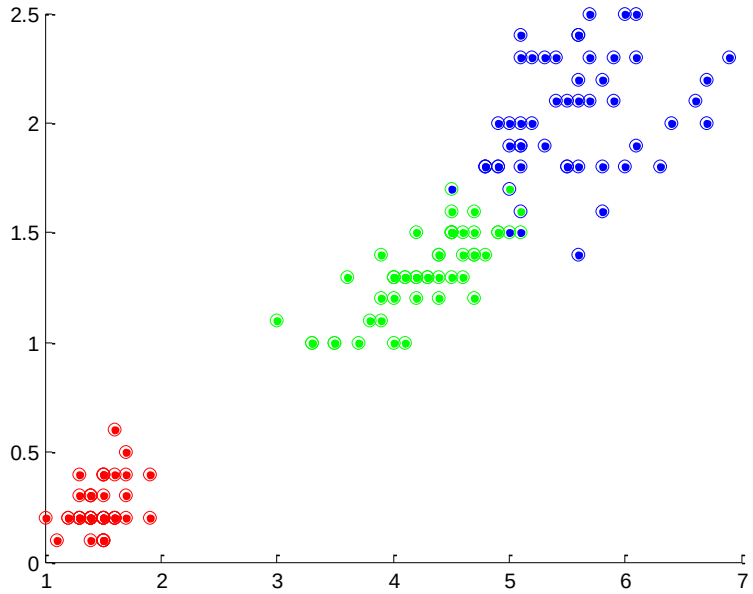
Celem zadania jest analiza wpływu procesu uczenia na proces predykcji sieci neuronowej. Wczytaj zbiór danych iris34.csv, stwórz zmienne: X opisującą wejście sieci (pierwsze dwie kolumny) pamiętaj że toolbox matlaba wymaga by dane były zapisane w postaci kolumn, gdzie jedna kolumna opisuje jeden przypadek oraz Y opisującą wyjście sieci.

- Uwaga kolumny [3 4 5] zbioru danych opisują wyjścia sieci. Każde z kolumn przyjmuje wartość 1, jeśli dany obiekt należy do danej kategorii. Liczba kategorii jest więc równa liczbie kolumn. Inny sposób zakodowania wyjścia to stworzenie jednej kolumny, która przyjmuje wartości 1 dla kategorii nr 1, 2 dla kategorii ne 2 i 3 dla ostatniej kategorii.
  Pytanie: Jak myślisz, który sposób zakodowania wyjścia sieci jest lepszy – czy za pomocą n zmiennych binarnych, czy może jednej zmiennej z wieloma wartościami. Uzasadnij odpowiedź.

- Na podstawie X i Y stwórz sieć neuronową o jednej warstwie ukrytej i 3 neuronach w warstwie wyjściowej (UWAGA: patrz wcześniejszy opis dot. tworzenia struktury sieci)
- Dokonaj symulacji sieci nie nauczonej na danych X i wynik zapisz do zmiennej D1.
- Dokonaj procesu uczenia sieci na danych X i Y, a następnie dokonaj ponownie symulacji sieci na danych X, a wynik zapisz do D2.

Uzyskane wyniki narysuj za pomocą funkcji `plotClassificationResults`
Pamiętaj że wyniki sieci neuronowej są liczbami rzeczywistymi, a rzeczywiste wyjścia są wartościami binarnymi. Do uzyskania poprawnych wartości wykorzystaj funkcję `transformResults`

Przykład:



I wyświetl na dwóch różnych rysunkach wyniki działania sieci – przed uczeniem i po uczeniu.
Zbuduj funkcję obliczającą dokładność uczenia sieci jako ilość poprawnie odgadniętych wzorców w stosunku do całkowitej liczby wzorców.
Podaj dokładność nienauczonej sieci (wyniki uzyskane na podstawie analizy zmiennej D1) oraz sieci nauczonej (wyniki zapisane w D2). Wyniki zapisz w sprawozdaniu

Zad 2

Celem ćwiczenia jest wizualizacja wpływu struktury sieci neuronowej na podejmowane przez nią decyzje

a. Wczytaj zbiór danych class_data_noise_train.csv i naucz na nich sieć neuronową
b. Wczytaj zbiór danych class_data.csv i wyświetl wynik korzystając z przygotowanej wcześniej funkcji do rysowania, tak aby było widać czego sieć się nauczyła, oraz to czego powinna się nauczyć.
c. Dokonaj modyfikacji sieci zwiększając liczbę neuronów w warstwach oraz liczbę warstw oraz dokonaj identycznej wizualizacji jak w poprzednim punkcie oraz wylicz uzyskana dokładność klasyfikacji. Wyniki wizualizacji wraz z ustawieniami sieci i uzyskanymi dokładnościami klasyfikacji zapisz w sprawozdaniu
d. Odpowiedz na pytania:

e. Czy liczba neuronów i liczba warstw wpływa na jakość uzuskanych wyników? Czy zwiększanie liczby neuronów i liczy warstw zawsze prowadzi do poprawy wyników

Zad 3

Celem ćwiczenia jest zbadanie relacji między błędem/dokładnością predykcji sieci neuronowej na zbiorze uczącym i zbiorze testowym.

a. Korzystając ze zbioru class_data_noise_train.csv spróbuj dobrać najlepszą możliwą sieć neuronową zapewniającą największą możliwą dokładność klasyfikacji. W trakcie tego ćwiczenia dobieraj odpowiednio liczbę neuronów w warstwach i liczbę warstw. Do realizacji zadania nie używaj wykresu. Zadanie wykonaj jedynie na podstawie wartości funkcji błędu. Zapisz uzyskany najlepszy wynik klasyfikacji
b. Wybierz najlepszą sieć nauronową z poprzedniego podpunktu (a) – czyli taką która zapewniała najmniejszy błąd (największą dokładność klasyfikacji), następnie wczytaj zbiór class_data_noise_test.csv i dla tej najlepszej sieci dokonaj symulacji. Wyznacz wartość błędu/dokładności klasyfikacji uzyskanej dla zbioru class_data_noise_test.csv i porównaj tą wartość z błędem dokładnością uzyskaną w podpunkcie (a). Jeśli są różnice pomiędzy wynikami z poprzedniego podpunktu i obecnego to spróbuj je wyjaśnić – tzn jeśli występują to dlaczego te dwie dokładności/błędy sa różne, która/który jest większa?

Zad 4

Cele ćwiczenia jest identyczny z celem ćwiczenia nr 3, jednakże przy zmienionej metodzie oceny sieci.

a. Spróbuj powtórzyć zadanie 3, ale przy wyborze najlepszej sieci (wybierz jakie parametry konfiguracyjne sieci są najlepsze) wykorzystaj tzw. test krzyżowy – w tym celu skorzystaj z funkcji neuralNetTestKrzyzowy UWAGA: funkcja jako argumenty przyjmuje dane w reprezentacji takiej, gdzie wiersze odpowiadają rekordom.
b. Porównaj uzyskaną najlepszą sieć z zadania 3 i najlepszą z zadania 4 – czy najlepsze konfiguracje są identyczne czy się różnią?
c. Dla najlepszej sieci z podpunktu a powtórz operację z zadania 3.b Czy wyniki są podobne, czy się może różnią?