Opis programu do uczenia sieci neuronowych `sieci`

Program `sieci` służy do uczenia modeli dynamicznych realizowanych przez sieci neuronowych typu perceptronowego (ang. Multi Layer Perceptron, MLP) o jednej warstwie ukrytej, w której stosuje się funkcje tangensa hiperbolicznego. Modelowany proces może mieć tylko jedno wejście u oraz wyjście y, ale możliwy jest dowolny rząd dynamiki, opóźnienie, liczba neuronów ukrytych nie może przekraczać 50. Możliwe są dwa tryby uczenia: predyktor ARX (bez rekurencji, jeden krok do przodu) oraz OE (rekurencyjny, wielokrokowy). W trybie ARX równanie modelu ma postać

$$\hat{y}(k) = f(u(k-\tau), \ldots, u(k-n_\mathrm{B}), y(k-1), \ldots, y(k-n_\mathrm{A}))$$

gdzie $\hat{y}(k)$ jest sygnałem wyjściowym modelu dla chwili k, symbole $u(k-\tau), \ldots, u(k-n_\mathrm{B})$ oznaczają sygnał wejściowy dla chwil $k-\tau, \ldots, k-n_\mathrm{B}$, symbole $y(k-1), \ldots, y(k-n_\mathrm{A})$ oznaczają sygnał wyjściowy procesu dla chwil $k-1, \ldots, k-n_\mathrm{A}$. Sieć ma więc $n_\mathrm{B} - \tau + 1 + n_\mathrm{A}$ wejść. W trybie OE równanie modelu ma postać

$$\hat{y}(k) = f(u(k-\tau), \ldots, u(k-n_\mathrm{B}), \hat{y}(k-1), \ldots, \hat{y}(k-n_\mathrm{A}))$$

gdzie symbole $\hat{y}(k-1), \ldots, \hat{y}(k-n_\mathrm{A})$ oznaczają sygnał wyjściowy modelu dla chwil $k-1, \ldots, k-n_\mathrm{A}$.

Dane do uczenia zapisane są w pliku `dane.txt`, przy czym w pierwszej kolumnie są kolejne próbki sygnału wejściowego proces (u), w drugiej kolumnie są kolejne próbki sygnału wyjściowego procesu (y). Maksymalna liczba próbek wynosi 5000.

Uczenie (optymalizacja wag sieci) odbywa się za pomocą algorytmu najszybszego spadku lub algorytmu zmiennej metryki typu BFGS. Błąd sieci (minimalizowana funkcja celu) ma postać

$$E = \sum_{k=S}^{P} (\hat{y} - y(k))^2$$

gdzie: $S = \max(n_\mathrm{A}, n_\mathrm{B}) + 1$, P – liczba próbek danych, $\hat{y}(k)$ – wyjście modelu dla chwili k, $y(k)$ – wzorzec uczący dla chwili k (ze zbioru danych uczących). Początkowe wartości wag (inicjalizacja uczenia) są losowe (z zakresu $-1 \ldots 1$). Zatrzymanie algorytmu uczącego następuje po przekroczeniu maksymalnej liczby iteracji lub wówczas, gdy błąd spadnie poniżej wartości granicznej.

Konfiguracja programu jest zawarta w pliku `ustawienia.txt`. Kolejność parametrów jest następująca:

τ
n_B
n_A
K (liczba neuronów ukrytych, maksymalnie 50)
maksymalna liczba iteracji uczących
błąd graniczny, przy którym następuje przerwanie uczenia
algorytm uczący: 1 – najszybszy spadek, 2 – BFGS
tryb uczenia: 1 – ARX, 2 – OE

Skrypt `uczenie.m` przedstawia w sposób graficzny zmiany funkcji celu (dla predyktora ARX i OE) w kolejnych iteracjach uczących, a także zmiany kroku optymalizacji i normy gradientu. Parametry (wagi) nauczonego modelu zapisywane są w pliku `model.m`. Wagi pierwszej warstwy oznaczone są symbolem $w1(i,j)$, gdzie $i = 1, \ldots, K$, $j = 1, \ldots, n_\mathrm{B} - \tau + 1 + n_\mathrm{A}$, wagi drugiej warstwy symbolem $w2(i)$, gdzie $i = 1, \ldots, K$, polaryzacja pierwszej warstwy symbolem

$w10(i,1)$, gdzie $i = 1, K$, polaryzacja drugiej warstwy symbolem $w20$. Wyjście modelu można wyznaczyć w zależności od wejść modelu i wag za pomocą wzoru (dla predyktora OE)

$$\hat{y}(k) = w20 + \sum_{i=1}^{K} w2(i)\,\mathrm{tgh}\left(w10(i,1) + \sum_{j=1}^{n_\mathrm{B}-\tau+1} w1(i,j)u(k-\tau+1-j) + \sum_{j=1}^{n_\mathrm{A}} w1(i,n_\mathrm{B}-\tau+1+j)\hat{y}(k-j)\right)$$

Wyjście modelu można również bardzo łatwo korzystając z faktu, że wagi są wektorami i macierzami o odpowiednich wymiarach

$$\hat{y}(k) = w20 + w2\,\mathrm{tgh}\,(w10 + w1q(k))$$

gdzie wektor wejściowy sieci w chwili k ma postać

$$q(k) = [u(k-\tau)\ldots u(k-n_\mathrm{B})\ \hat{y}(k-1)\ldots\hat{y}(k-n_\mathrm{A})]^\mathrm{T}$$