Systemy Inteligentnego Przetwarzania
— Projekt —
Wahadło odwrócone (sieć neuronowa)

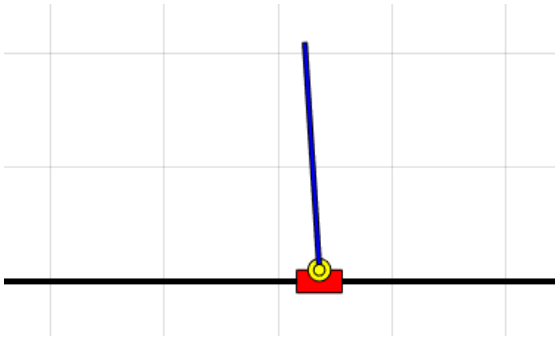
Adrian Frydmański, Dawid Gracek

19 grudnia 2017

1 Wstęp teoretyczny

Odwrócone wahadło niewiele różni się od swojego „zwykłego" odpowiednika. Jest swobodnie wiszącym prętem przymocowanym jednym końcem do wózka. Wózek z kolei ma możliwość poruszania się wzdłuż osi (w jednym wymiarze, acz nic nie stoi na przeszkodzie, żeby rozszerzyć problem wahadła do dwóch wymiarów). Układ ten:

- posiada dwa punkty równowagi: stabilny, kiedy wahadło spoczywa w położeniu dolnym i niestabilny, kiedy wahadło skierowane jest pionowo ku górze,
- jest tzw. obiektem niedosterowanym, ponieważ wielkości sterowanych możemy wyróżnić więcej niż jest wejść w układzie.



Rysunek 1: Model wahadła

Jedyną wielkością, która wpływa na stan układu jest siła przyłożona do wózka, którego przemieszczanie się wprawia w ruch wahadło. Taki układ regulacji może mieć kilka celów:

- stabilizacja wahadła w położeniu górnym,
- regulacja położenia wózka w odniesieniu do całego stanowiska,
- realizacja algorytmów umożliwiających rozbujanie wahadła z pozycji dolnej i doprowadzenie go to pozycji górnej.

Ów projekt zakłada realizację 2. pierwszych celów poprzez generowanie odpowiednich nastaw regulatora kontrolującego ruch wózka i wprowadzenie go do położenia górnego ze stanu początkowego, gdy wahadło jest nachylone względem ziemi pod kątem mniejszym, niż 90°. Nastawy mają być generowane przez wyuczoną sieć neuronową.

Matematyczny model wahadła jest opisany na stronie jtjt.pl[1] i stamtąd właśnie zaczerpnięte są obliczenia, z których korzysta symulacja.

[1] http://jtjt.pl/www/pages/odwrocone-wahadlo/LMIP.pdf

2 Implementacja

Środowisko, w jakim została przeprowadzona symulacja i tworzenie sieci neuronowej, to Matlab R2017a. Dane testowe do uczenia sieci zostały pozyskane przez funkcję[2] do strojenia regulatora liniowo kwadratowego w zależności od „parametrów środowiska":

- masa wózka,
- masa wahadła,
- długość od mocowania do środka ciężkości wahadła,
- współczynnik tarcia wózka.

W głównej funkcji, `lqr_training`, zostało wygenerowane 100.000 losowych zestawów danych i znaleziono dla każdego z nich odpowiednie nastawy regulatora.[3] Następnie w narzędziu `nntraintool` została wygenerowana sieć neuronowa, która na owych zestawach nauczyła się dobierać parametry regulatora. Sprawdzono, że dla 4 neuronów — tylu, co wyjść — sieć działa wystarczająco dobrze. Sieć posiada jedną warstwę ukrytą. Przyjęto następujące wielkości podzbiorów zbioru 100.000 zestawów testowych:

- 70% — zbiór uczący
- 15% — zbiór walidacyjny
- 15% — zbiór testowy

Warto przyjrzeć się dokładniej kilku aspektom, które udało się zauważyć podczas tworzenia niniejszego projektu.

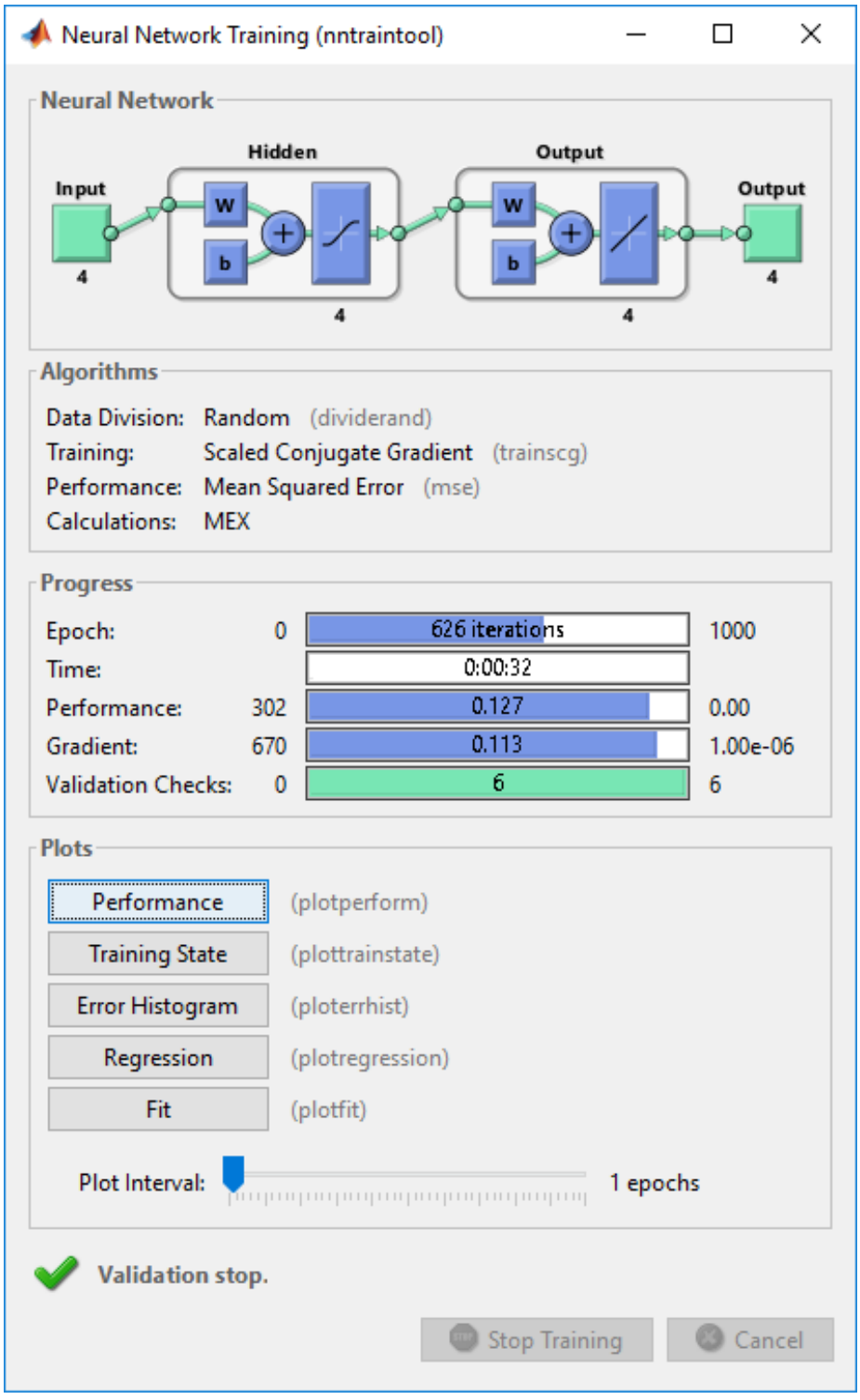
Mając 4 neurony wejściowe i wyjściowe sieci neuronowej, jej dokładność, czyli wielkość błędu średnio kwadratowego, zależy od przedziału wartości neuronów wejściowych. Zwiększając ten zbiór o jeden rząd dla dwóch neuronów (masy wózka i wahadła), liczbę danych uczących powinno się zwiększyć o 2 rzędy — jeśli granice badanej przestrzeni zwiększymy w 2 wymiarach 2-krotnie, to by zachować to samo zagęszczenie, należy zwiększyć liczbę elementów w niej 2-krotnie dla każdego wymairu, zatem 4-krotnie. Nie zwiększając go, błąd ulegnie powiększeniu ze względu na zbyt mały rozmiar zbioru danych uczących.

Biorąc po uwagę szybkość generacji danych uczących oraz szybkość uczenia się sieci neuronowej, należy zauważyć, że przedział wartości masy wózka i masy wahadła oraz liczba próbek danych uczących muszą być dostosowane do mocy obliczeniowej i czasu, jaki chcemy przeznaczyć na cały proces.

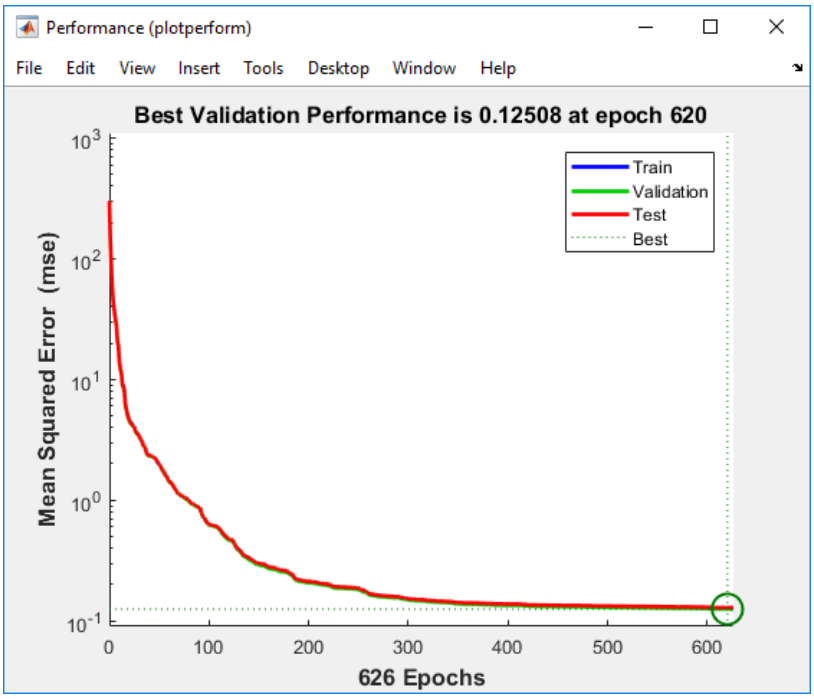
Dla 100 000 próbek wyznaczono eksperymentalnie, że wartości ograniczające przedział masy wózka i wahadła muszą od siebie odstawać maksymalnie o 3 rzędy. Np. 1 – 100, 0.1 – 10 itp. Liczba wartości w przedziale zależy od dokładności funkcji losującej *rand()*, a ta w Matlabie ma dokładność 0,0001.

[2] Model i funkcja dostępne na https://github.com/Jarczyslaw/Inverted-Pendulum

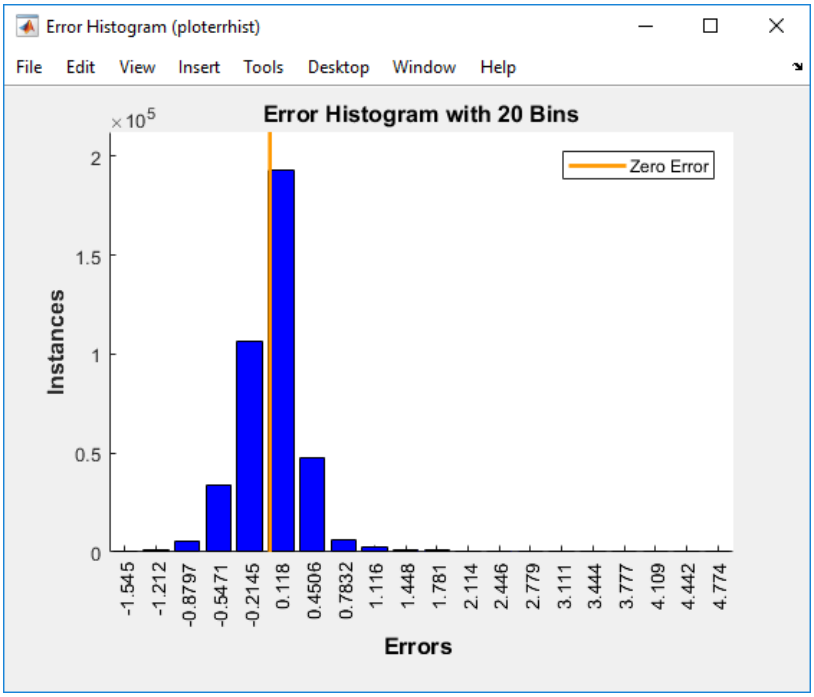
[3] Cały kod dostępny w repozytorium https://github.com/Adrian94F/SIP



Rysunek 2: Wygenerowana sieć i proces uczenia

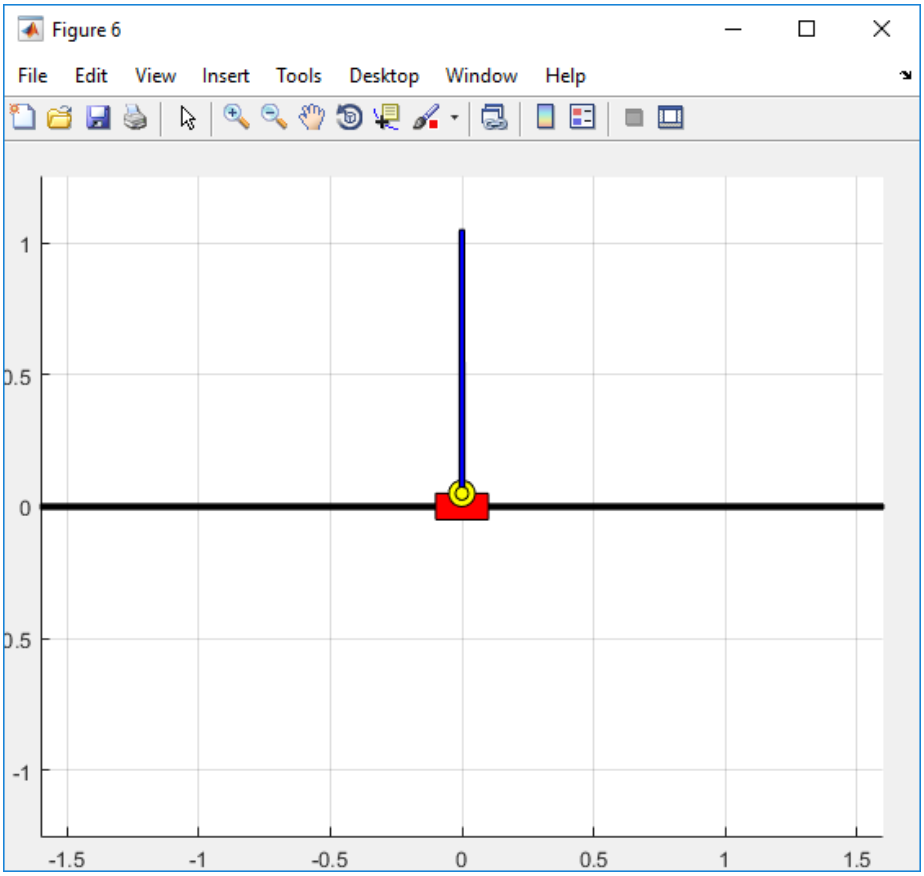


Rysunek 3: Wykres błędów od epok treningowych

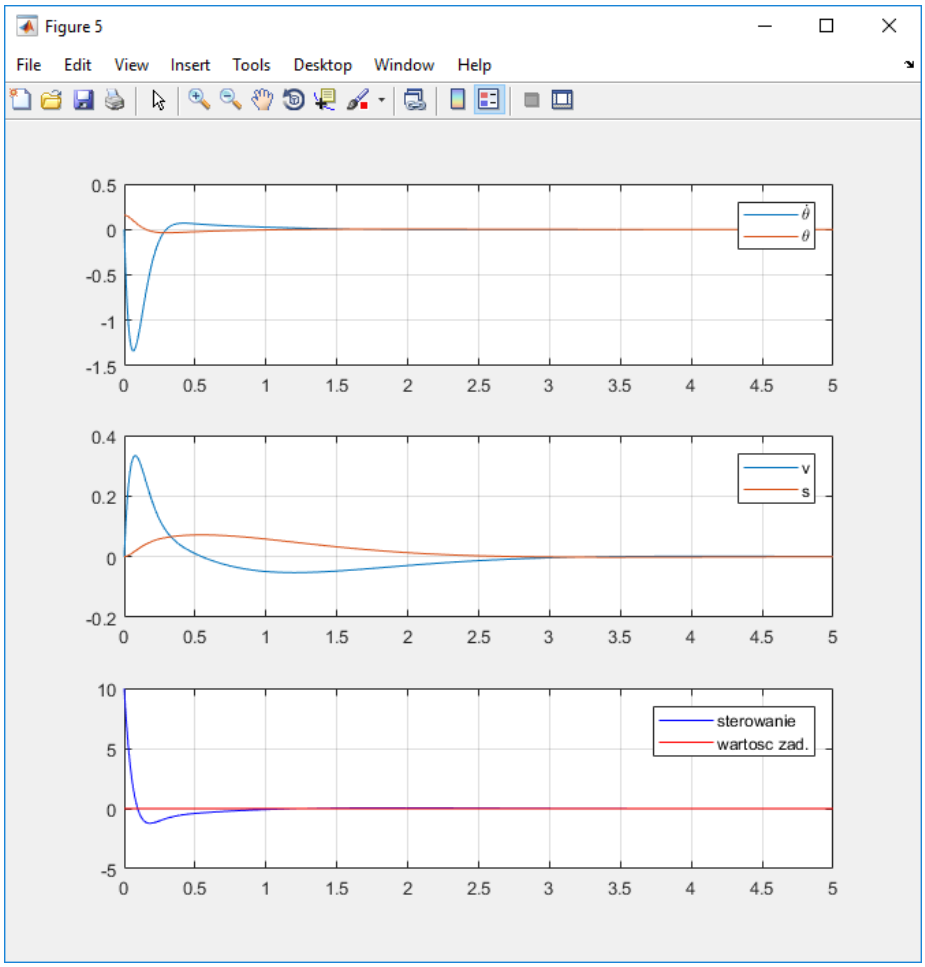


Rysunek 4: Histogram błędów

Po wygenerowaniu sieci został ponownie wylosowany zbiór zmiennych środowiska, na podstawie których sieć znalazła odpowiednie nastawy regulatorów. Została uruchomiona symulacja i przebiegła ona pomyślnie.



Rysunek 5: Osiągnięty stan równowagi podczas symulacji



Rysunek 6: Wyniki symulacji (od góry): kąt i pochodna kąta pomiędzy wahadłem, a pionem, prędkość i pozycja wózka, przyłożona siła do wózka i zadana wartość zadana

3 Podsumowanie

Projekt pokazał, że sieć neuronową można zaprząc do przeróżnych zadań, nawet do dostosowywania nastaw regulatora w wahadle odwróconym. Jest to możliwe do zrobienia i proste w implementacji w takim środowisku, jak Matlab, aczkolwiek zdaniem autorów projektu zbędne, gdyż funkcja strojąca regulator, dzięki której uzyskano dane do nauki sieci, istnieje i działa wystarczająco szybko. Nie widać więc potrzeby wymiany jej na sieć neuronową, której wynik zawsze będzie obarczony pewnym błędem w stosunku do wyniku działania owej funkcji.