

Sieć neuronowa jako pomoc w podejmowaniu decyzji inwestycyjnych

Tomasz Guźniczak

Paweł Nocoń

Hanna Mazurkiewicz

GKiO2

Dokumentacja użytkownika

1. Informacje ogólne

1.1. Autorzy:

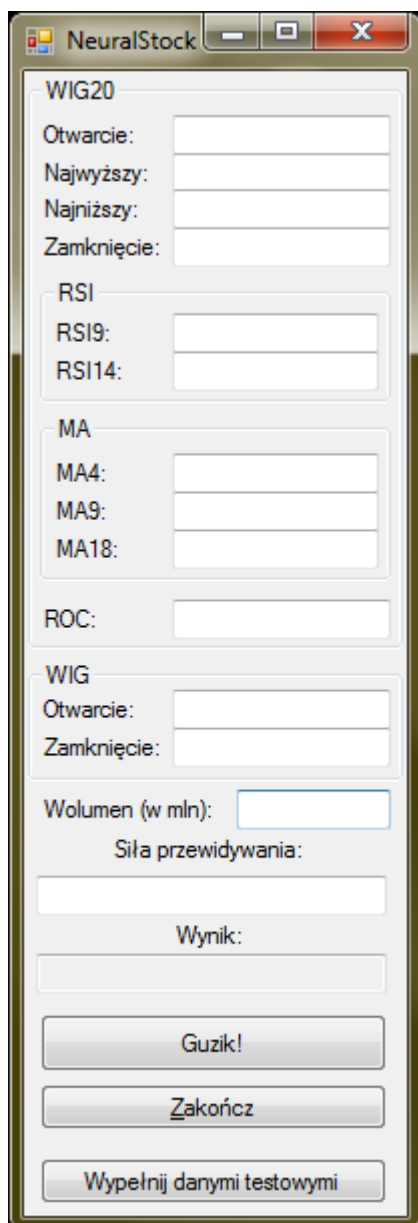
- a) Tomasz Guźniczak
- b) Paweł Nocoń
- c) Hanna Mazurkiewicz

1.2. Opis programu

Aplikacja ma za zadanie wspomagać podejmowanie decyzji dotyczących inwestycji giełdowych.

2. Środowisko programu:

2.1. Interfejs użytkownika



The screenshot shows the 'NeuralStock' application window. It contains several input fields and buttons for stock analysis:

- WIG20** section:
 - Otwarcie:
 - Najwyższy:
 - Najniższy:
 - Zamknięcie:
- RSI** section:
 - RSI9:
 - RSI14:
- MA** section:
 - MA4:
 - MA9:
 - MA18:
- ROC**:
- WIG** section:
 - Otwarcie:
 - Zamknięcie:
- Wolumen (w mln)**:
- Siła przewidywania**:
- Wynik**:
- Guzik!** button
- Zakończ** button
- Wypełnij danymi testowymi** button

2.2. Dane wejściowe

Do odpowiednich pól należy wprowadzić kluczowe wartości indeksów WIG20 i WIG, wartość obrotu na giełdzie i wskaźniki liczone na indeksie WIG20.

Pole "Siła przewidywania" służy do określenia jak bardzo prognozowana zmiana musi odbiegać od 0, aby została zaklasyfikowana jako "Kupuj" lub "Sprzedawaj".

2.3. Wyniki

Wynikiem działania programu jest komunikat "Kupuj", "Sprzedawaj" lub "Wstrzymaj się od działań" zależny od wyniku obliczeń.

2.4. Przykłady danych wejściowych i wyników programu

The image displays three instances of the NeuralStock application window, each showing a different set of input data and the resulting trading signal. The windows are titled "NeuralStock" and contain the following fields:

- WIG20:** Otwarcie, Najwyższy, Najniższy, Zamknięcie, RSI (RSI9, RSI14), MA (MA4, MA9, MA18), ROC.
- WIG:** Otwarcie, Zamknięcie.
- Wolumen (w mln):**
- Siła przewidywania:** (empty field)
- Wynik:** (displayed as KUPUJ, SPRZEDAWAJ, or KUPUJ)
- Buttons:** Guzik!, Zakończ, Wypełnij danymi testowymi.

Example 1 (Left):

WIG20	
Otwarcie:	2368,03
Najwyższy:	2395,13
Najniższy:	2366,21
Zamknięcie:	2384,34
RSI	
RSI9:	48,4
RSI14:	64,23
MA	
MA4:	2386,723
MA9:	2367,31
MA18:	2362,02
ROC:	0,0069
WIG	
Otwarcie:	40871,6
Zamknięcie:	41069,7
Wolumen (w mln):	24,03
Siła przewidywania:	
Wynik:	KUPUJ

Example 2 (Middle):

WIG20	
Otwarcie:	2283,29
Najwyższy:	2290,49
Najniższy:	2259,67
Zamknięcie:	2271,03
RSI	
RSI9:	29,12
RSI14:	29,927
MA	
MA4:	2305,49
MA9:	2334,886
MA18:	2349,70
ROC:	-0,0054
WIG	
Otwarcie:	39561,7
Zamknięcie:	39392,5
Wolumen (w mln):	32,0750
Siła przewidywania:	
Wynik:	SPRZEDAWAJ

Example 3 (Right):

WIG20	
Otwarcie:	2223,16
Najwyższy:	2246,45
Najniższy:	2219,61
Zamknięcie:	2242,15
RSI	
RSI9:	74,825
RSI14:	76,229
MA	
MA4:	2254,86
MA9:	2234,60
MA18:	2166,06
ROC:	0,0085
WIG	
Otwarcie:	39898,82
Zamknięcie:	40187,43
Wolumen (w mln):	28,54
Siła przewidywania:	
Wynik:	KUPUJ

3. Sytuacje niepoprawne

3.1. Wykaz komunikatów:

- "Chwilka, sieć w trakcie nauki..." - należy poczekać aż aplikacja zakończy uczenie sieci neuronowej.

3.2. Opis błędów i warunków powstania

- a) "Popraw dane wejściowe!" - jedno lub więcej pól nie zostało wypełnione lub jego zawartość nie jest liczbą. Należy skorygować wprowadzone dane.
- b) "Brak pliku "teaching.txt" z danymi uczącymi!" - na dysku nie znaleziono pliku "teaching.txt". Należy umieścić plik z danymi uczącymi w folderze z aplikacją i upewnić się, że ma odpowiednią nazwę.
- c) "Błąd w pliku z danymi uczącymi!" - plik "teaching.txt" zawierał nieprawidłowe znaki.

Dokumentacja techniczna

1. Struktura programu

1.1. Opis plików zewnętrznych:

a) "teaching.txt":

Format zapisu:

Struktura dzieli się na linie, w których wartości indeksów giełdowych podane w odpowiedniej kolejności są oddzielone znakiem tabulatora.

Użycie:

Plik zawiera dane wykorzystane do uczenia sieci neuronowej.

1.2. Wykaz modułów systemowych i dodanych bibliotek.

a) NeuralNetwork – główna klasa zawierająca logikę aplikacji

b) AForge.Net (AForge.Neuro) – biblioteka wspomagająca tworzenie aplikacji wykorzystujących sieci neuronowe.

2. Moduł NeuralNetwork

2.1. Przeznaczenie

Główna logika aplikacji. Zawiera metody uczące sieć neuronową i pozwalające na wykonanie przewidywania zmian na giełdzie indeksu WIG20.

2.2. Opis metod klasy

Konstruktory publiczne:

- `public NeuralNetwork()`

Tworzy instancję obiektu z domyślnymi parametrami

- `public NeuralNetwork(int alpha, int hidden, double learning, double momentum)`

Tworzy instancję obiektu o podanych parametrach.

Parametry:

int alpha – współczynnik alfa sigmoidy

int hidden – ilość neuronów ukrytych

double learning – siła uczenia zestawu danych

double momentum – wartość określająca jak silny wpływ na proces uczenia ma poprzedni zestaw danych

Metody publiczne:

- `public void Teach()`

Rozpoczyna proces uczenia sieci neuronowej.

- `public void PrepareData(string filename)`

Wczytuje dane z pliku tekstowego i parsuje je do tablicy.

Parametry:

string filename – nazwa pliku zawierającego dane uczące

- `public double[] Think(double[] input)`

Wywołuje funkcję wykonującą obliczenia nauczonej sieci neuronowej.

Parametry:

double[] input – tablica danych wejściowych algorytmu

Zwraca:

double[] – tablica zawierająca dane wyjściowe algorytmu

- `public double Test()`

Testuje dokładność algorytmu.

Zwraca:

double – współczynnik udanych testów.

2.3. Sposób wykorzystania

Po utworzeniu obiektu klasy NeuralNetwork konieczne jest wywołanie metody Teach(), która wykona proces uczenia sieci neuronowej. Dopiero po zakończeniu tej operacji można wywołać metodę Think(double[]) z parametrem zawierającym odpowiednio przygotowaną tablicę z danymi wejściowymi algorytmu. Zwrócony wynik należy odpowiednio zinterpretować.

2.4. Sytuacje niepoprawne

- a) FileNotFoundException – brak pliku zawierającego dane uczące
- b) FormatException – błąd konwersji zawartości pliku na typ double

3. Testy

3.1. Metoda Test()

Metoda Test sprawdza wyniki działania algorytmu dla wczytanych wcześniej z pliku danych i zwraca stosunek słusznych przewidywań do całości. Przy ustawieniach domyślnych, które są dobrane tak, aby były jak najbardziej optymalne, wynosi on nawet 86%.

4. Wnioski

Moglibyśmy zaoszczędzić mnóstwo czasu, gdybyśmy wcześniej zrozumieli, że dane wejściowe i wyjściowe muszą być zmapowane na odpowiednie wartości (przy użyciu BipolarSigmoidFunction były to wartości z zakresu $[-1, 1]$). Po zrozumieniu tego poszło już z górki. Zgodnie z przewidywaniami najlepsza ilość neuronów ukrytych to połowa lub niewiele więcej neuronów wejściowych. Okazało się też, że momentum lepiej nie ruszać – przynajmniej przy tym problemie. Im większe było momentum, tym mniej pewnym było czy sieć zdoła się w miarę sensownie nauczyć, za to osiągnięte maksimum trafnych przewidywań było coraz mniejsze. Wynik (86% trafnych przewidywań na danych uczących) pozytywnie nas zaskoczył i nie omieszkamy przetestować działania

programu na danych obecnych – będzie to zapewne nieco uciążliwe, gdyż nie udało nam się znaleźć darmowego źródła danych, ale może nam się uda to kiedyś jakoś obejść – im większa okaże się trafność prawdziwych prognoz, tym nasza motywacja ku temu na pewno będzie rosła.