**Prognozowanie kierunku wzrostu aktywów przy użyciu sztucznych sieci neuronowych**

Projekt PSZT 15Z, KZ.SN.1

Autorzy:

**Paweł Andruszkiewicz**

**Marta Bandarzy**

**Paweł Srokosz**

Prowadzący:

**Kamil Żbikowski**

Spis treści

[Treść zadania 1](#_Toc441369645)

[Decyzje projektowe i realizacja 2](#_Toc441369646)

[Wstęp 2](#_Toc441369647)

[Model sieci 3](#_Toc441369648)

[Implementacja 5](#_Toc441369649)

[Instrukcja uruchomieniowa 5](#_Toc441369650)

[Eksperymenty, metodyka testowania, wnioski 6](#_Toc441369651)

[Wnioski 8](#_Toc441369652)

[Źródła 8](#_Toc441369653)

# Treść zadania

**KZ.SN.1.** Na podstawie danych notowań giełdowych z [www.stooq.pl](http://www.stooq.pl) zbudować aplikację, która służy do prognozowania kierunku wzrostu danego aktywa (tj. dla chwili t prognozuje czy dzienna stopa zwrotu definiowana jako Rt,t+1 = Pt+1/Pt, gdzie Pt to cena w chwili t >= 0, jest dodatnia czy ujemna). Przedstawić dyskusję dotyczącą wyników i metodyki badań. Powyższe zadanie klasyfikacji należy rozwiązywać z użyciem wielowarstwowej sieci neuronowej uczonej metodą online. Liczba warstw oraz neuronów w każdej z nich jest przedmiotem eksperymentów. Należy opisać metodykę testowania.

# Decyzje projektowe i realizacja

## Wstęp

Do rozwiązania zadania klasyfikacji kierunku wzrostu postanowiliśmy skorzystać z RNN – rekurencyjnych sieci neuronowych. Sieci rekurencyjne przeżywają ostatnio drugą młodość – i są bardzo powszechnie używane, ponieważ pozwalają uzyskiwać niespodziewanie dobre rezultaty (zwłaszcza w takich dziedzinach jak np. generowanie tekstu – polecamy wpis na blogu [1]).

Postanowiliśmy wykorzystać sieć **LSTM**, czyli ***Long Short-Term Memory***. Jest to specyficzna architektura sieci, która wykorzystuje tzw. komórki pamięci (memory cell), które charakteryzują się „bramką zapominania” (*forget gate*), która pozwala komórce zapamiętać lub zapomnieć swój poprzedni stan.

|  |
| --- |
| _images/lstm_memorycell.png |
| ***LSTM memory cell*** |

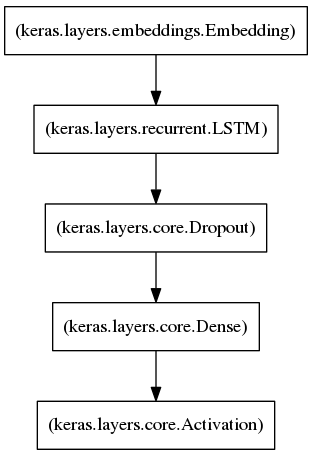
Sieć typu LSTM pozwala na lepsze modelowanie zależności długoterminowych i rozwiązuje problem z niekorzystnym zachowaniem gradientu w sieciach rekurencyjnych – zarówno jego zaniku (*vanishing gradient*) jak i eksplozji (*exploding gradient*).

Więcej informacji o LSTM można zaleźć w: [2], [3], [4] i [5].

Postanowiliśmy również sprawdzić jak będzie się sprawować nowe podejście (użyte po raz pierwszy w 2014 roku) – GRU, *Gated Recurrent Unit*. W dużym skrócie jest to trochę uproszczona wersja LSTM ([6], [7], [8]).

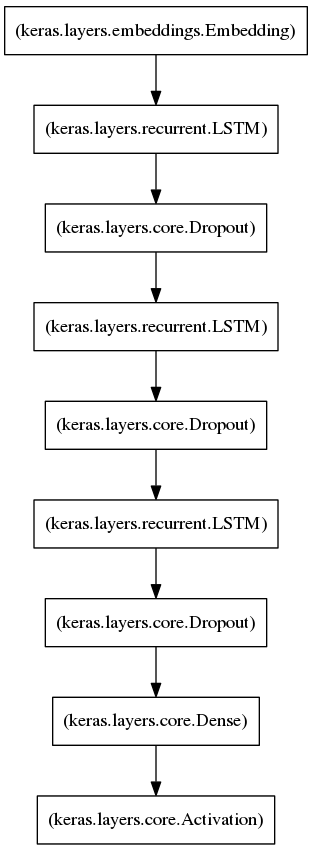
## Model sieci

Wykorzystaliśmy sieć zbudowaną w następujący sposób:

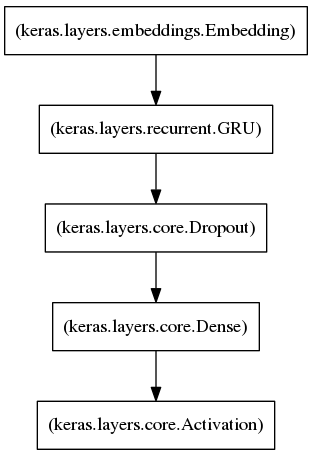


* warstwa Embedding – warstwa wykonująca pewnego rodzaju mapowanie wartości na wektor specyficzny wektor gęstości (optymalizacja danych wejściowych)
* warstwa LSTM – najważniejsza warstwa modelu
* warstwa Dropout – warstwa dodająca dropout do danych (warstwa zerująca wartość losowych neuronów, jej celem jest zmniejszenie podatności na overfitting)
* warstwa Dense – klasyczna warstwa fully-connected
* - warstwa Activation – warstwa dodająca funkcję aktywacji na danych wyjściowych

Testowaliśmy też sieć zawierającą więcej (3) warstwy LSTM zbudowaną w następujący sposób:



A także sieć opartą o GRU:



Parametry używanych modeli sieci dobraliśmy za pomocą dostępnych materiałów oraz przeprowadzonych eksperymentów (więcej w punkcie *Eksperymenty, metodyka testowania, wnioski*).

## Implementacja

Projekt został napisany w Pythonie. Korzystaliśmy z freamworku Keras [<http://keras.io>], który jest świetnym narzędziem do prototypowania i eksperymentowania z sieciami neuronowymi. Freamwork ten opiera się o bibliotekę Theano [<http://deeplearning.net/software/theano/>] bardzo dobrze sprawującą się w zagadnieniach deep-learningowych.

Trenujemy sieć na historycznych danych wartości aktywów konkretnych akcji. Jako wejście sieci podawana jest cena zamknięcia z ostatnich 20 dni (na podstawie artykułów dot. przewidywania giełdy najlepsze wyniki dawały wartości 15-25 dni). Problem jest klasyfikatorem binarnym – sieć ma stwierdzić, czy kolejnego dnia cena wzrośnie czy zmaleje.

# Instrukcja uruchomieniowa

[TODO]

# Eksperymenty, metodyka testowania, wnioski

Przeprowadziliśmy szereg eksperymentów i testów w celu dobrania najlepszych parametrów dla naszej sieci neuronowej.

Pierwsze eksperymenty miały na celu stwierdzić **ile iteracji trenowania** powinno być przeprowadzonych. Po kilku testach stwierdziliśmy, że najlepsze wyniki daje sieć po ok. 5-20 iteracjach. W końcu zdecydowaliśmy się trenować w 10 iteracjach. Biorąc pod uwagę, że ilość danych trenujących dla naszego problemu (rząd kilku tysięcy – historyczne dane dot. wartości akcji) są stosunkowo małe dla sieci LSTM, tak mała liczba iteracji jest wystarczająca.

Przy większej liczbie iteracji zachodzi zjawisko tzw. overfittingu sieci – nadmiernego dopasowania się do danych trenujących – objawia się to wysokim accuracy dla danych trenujących, a bardzo niskim dla danych testujących.

Przykładowe wartości accuracy dla zbioru testującego po kilku iteracjach przedstawiono na wykresach (testy na modelu z jedną warstwą LSTM zawierającą 128 neuronów ukrytych, dane trenujące i testujące – zbiór danych od 1994 do 2015 podzielony w stosunku 70:30):

Kolejne eksperymenty były podyktowane wybraniem **odpowiedniej liczby neuronów ukrytych w warstwie LSTM**. Przetestowaliśmy modele z 64, 128 i 256 neuronami ukrytymi.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Trafność przewidywań** | | | | | |
| **Liczba neuronów** | **Dębica – DBC** | | | **KGHM Polska Miedź - KGH** | | |
| **Po 5 iteracjach** | **Po 10 iteracjach** | **Po 15 iteracjach** | **Po 5 iteracjach** | **Po 10 iteracjach** | **Po 15 iteracjach** |
| **64** | 0,5210 | 0,5311 | 0,5127 | 0,5224 | 0,5224 | 0,5216 |
| **128** | 0,5222 | 0,5203 | 0,5083 | 0,5188 | 0,5209 | 0,5144 |
| **256** | 0,5318 | 0,5006 | 0,5095 | 0,5209 | 0,5224 | 0,5065 |

Jak widać, najlepszy okazał się model z warstwą LSTM zawierającą 64 neurony. Mniejsza liczba neuronów to też dużo krótszy (nawet 7x w stosunku do modelu z 256 neuronami) czas trenowania modelu.

**Porównanie z modelem 3-warstowym:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **KGHM Polska Miedź - KGH** | | |
| **Model** | **Po 5 iteracjach** | **Po 10 iteracjach** | **Po 15 iteracjach** |
| **1 warstwa LSTM**  **(64 neurony)** | 0,5224 | 0,5224 | 0,5216 |
| **3 warstwy LSTM**  **(64, 64, 32)** | 0,5202 | 0,5152 | 0,5224 |

Jak widać, zastosowanie modelu z trzema warstwami nie daje dodatkowych korzyści w naszym przypadku.

**Zastosowanie GRU**:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Dębica – DBC** | | |
| **Model** | **Po 5 iteracjach** | **Po 10 iteracjach** | **Po 15 iteracjach** |
| **1 warstwa LSTM**  **(64 neurony)** | 0,5210 | 0,5311 | 0,5127 |
| **1 warstwa GRU**  **(64 neurony)** | 0,5426 | 0,5280 | 0,5280 |

Możemy zauważyć, że wyniki, które otrzymaliśmy używając GRU są bardzo zbliżone do wyników otrzymywanych przy użyciu LSTM. W związku z tym udostępniamy możliwość wyboru modelu opartego o GRU w naszym programie (chociaż w przypadku tego problemu różnica między LSTM a GRU będzie bardzo mała).

## Wnioski

Może się wydawać, że osiągnięte przez nas wyniki (poprawność predykcji na poziomie średnio 52-53%) są wynikami bardzo słabymi. Zauważmy jednak, że giełda jest tworem bardzo złożonym, na który wpływa niezliczona ilość czynników. My staramy się przewidzieć kierunek zmiany tylko na podstawie danych historycznych – co z góry skazane jest na porażkę. Na podstawie kilku artykułów i prac doktorskich (np. [9], [10]) dotyczących tematu przewidywań giełdowych dowiedzieliśmy się, że często, nawet na dużo bardziej skomplikowanych modelach rzeczywistości, uzyskiwana skuteczność jest niewiele większa niż 60-70%. Na samych danych historycznych najlepsze osiągane wyniki jakie widzieliśmy nie przekraczały 58%.

W związku z tym osiągane przez nas wyniki na poziomie 52-53% należy uznać za całkiem niezłe (mimo, że na pierwszy rzut oka takie accuracy nie różni się wiele od rzutu monetą – to jednak statystyka jest po naszej stronie i nawet dysponując taką dokładnością możemy się spodziewać długoterminowych zysków – w końcu na podobnej zasadzie działa gra w ruletkę w kasynie…).

# Źródła

[1] The Unreasonable Effectiveness of Recurrent Neural Networks <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

[2] LONG SHORT-TERM MEMORY <http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf>

[3] LSTM Networks for Sentiment Analysis <http://deeplearning.net/tutorial/lstm.html>

[4] Understanding LSTM Networks <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[5] Supervised Sequence Labelling with Recurrent Neural Networks <http://www.cs.toronto.edu/~graves/preprint.pdf>

[6] Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation <http://arxiv.org/pdf/1406.1078v3.pdf>

[7] Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling <http://arxiv.org/pdf/1412.3555v1.pdf>

[8] On the Properties of Neural Machine Translation: Encoder–Decoder Approaches <http://www.aclweb.org/anthology/W14-4012>

[9] Investigation of financial market prediction by recurrent neural network <http://www.kolegija.lt/dokumentai_img/Maknickiene-3-8-Pages-from-IITSBE-2011-2-Nr11.pdf>

[10] Financial Market Time Series Prediction with Recurrent Neural Networks <http://cs229.stanford.edu/proj2012/BernalFokPidaparthi-FinancialMarketTimeSeriesPredictionwithRecurrentNeural.pdf>