Regrese dat z Forexu

Petr Červíček, učo: 500328

Červen 2021

1 Úvod

Tento projekt se zaměřuje na regresi dat z Forexu. Forex, jehož jméno vychází z anglického Foreign Exchange, je mezinárodní obchodní systém pro směnu základních a vedlejších měnových párů. Jedná se o nejlikvidnější trh na světě a zároveň i největší světový trh. Forex je známý i pod názvy Forex Trading, Currency Trading, Foreign Exchange Market (zkráceně FX). Pro regresi byly použity přístupy ze strojového učení, převážně LSTM a také na data byl využit preprocessing.

2 Seznam podobných prací

1. Deep learning with long short-term memory networks for financial market predictions – autoři: T. Fischer, C. Krauss [3]
2. Algorithmic Trading Using Deep Neural Networks on High Frequency Data – autoři: A. Arévalo, J. Nino, G. Hernandez, J. Sandoval [1]
3. Algoritmické obchodování na burze s využitím umělých neuronových sítí – autor: Bc. Michal Chlud [2]
4. Algoritmické obchodování na burze s využitím umělých neuronových sítí – autor: Bc. Vítězslav Slavík [5]

3 Teorie a implementace

3.1 Data

Data byla stažena z Dukascopy (švýcarská banka) pomocí aplikace *tickstory*, která dokáže stáhnout všechny měnové páry, indexy i komodity. V tomto projektu byla snaha o regresi měnového páru USDCHF, k tomu byly staženy další páry, které jsou sepsány níže.

- USDCHF

- EURUSD
- GBPUSD
- USDJPY
- XAUUSD
- USA30IDXUSD
- USA500IDXUSD

Celkově byly data stažena z období od 2013-04-01 do 2021-04-01. Jedná se tedy o posledních 8 let. Tyto data pak byla následně zarovnána, aby všechny obsahovaly stejné množství řádků (výsledně to je cca 598 000 řádků). Obsah každého měnového páru je viditelný na obrázku 3.1.

```
Time,Open,High,Low,Close,Volume
2020-03-24 16:00:00,0.98204,0.98242,0.98172,0.98206,1548.29000151157
2020-03-24 16:05:00,0.98206,0.98278,0.98189,0.9826,1473.18000686169
2020-03-24 16:10:00,0.98261,0.98279,0.98225,0.9824,1210.40000593662
2020-03-24 16:15:00,0.98239,0.98264,0.98154,0.98157,1766.17000973225
2020-03-24 16:20:00,0.98155,0.9818,0.98085,0.98109,1407.50000309944
2020-03-24 16:25:00,0.98106,0.9816,0.9806,0.98116,1902.93000209332
2020-03-24 16:30:00,0.98115,0.98263,0.98086,0.98213,1718.21999847889
2020-03-24 16:35:00,0.98212,0.98284,0.98206,0.98273,1825.21999752522
2020-03-24 16:40:00,0.98277,0.98327,0.98262,0.98299,1370.03000044823
2020-03-24 16:45:00,0.98301,0.98306,0.98213,0.98263,1885.99000060558
```

Figure 1: Příklad obsahu jednoho měnového páru.

3.2 Převedení času

Čas je pro regresi velmi užitečnou informací, ačkoliv ne ve tvaru *string*. Proto byl čas převeden na signály ve tvaru *sinus* a *cosinus* pro dny, týdny a měsíce. To dává modelu přístup k důležitým frekvenčním vlastnostem.

3.3 Log-return

Dataset byl transformován na *logarithmic return*, díky čemuž došlo k odstranění šumu v těchto datech. Implementováno to bylo pomocí klouzavého okna, který vždy vydělil aktuální hodnotu hodnotou předešlou a na tuto výslednou hodnotu byl použit logaritmus.

3.4 Exponenciální vyrovnávání

Exponenciální vyrovnávání (nebo také exponenciální vyhlazování) je metoda pro vyhlazování a krátkodobou predikci časových řad. Název pochází z toho,že význam datového bodu pro hodnotu predikce exponenciálně klesá s časovou

vzdáleností od predikce. Exponenciální vyrovnání působí jako filtr typu dolní propust a odstraní z dat vysokofrekvenční šum. V případě regrese dat z burz, je exponenciální vyrovnání vhodná metoda k odstranění náhodný jevů na burze.

3.5 Normalizace dat

Je důležité škálovat *features* dat před samotným trénováním neuronové sítě. Běžným způsobem jak toto provést je takzvaná normalizace. Jedná se o odečtení střední hodnoty a vydělením směrodatnou odchylkou od každé *feature*.

```
train_mean = train_df.mean()
train_std = train_df.std()

train_df = (train_df - train_mean) / train_std
val_df = (val_df - train_mean) / train_std
test_df = (test_df - train_mean) / train_std
```

Figure 2: Implementace normalizace.

Střední hodnota i směrodatná odchylka by měly být vypočteny z trénovací sady, aby model neměl přístup k hodnotám z validační nebo testovací sady.

3.6 Model

V tomto projektu jsou využívané LSTM neuronové sítě. *Long short term memory networks* – obvykle zkracováno pouze na *LSTM* – jsou speciálním typem rekurentních neuronových sítí. Jak jejich název napovídá, jsou schopny si zapamatovat dlouhodobé závislosti.

LSTMs byly navrženy, aby vyřešily problém dlouhodobých závislostí systému. Zapamatování si informací po delší dobu je jejich hlavním znakem chování.

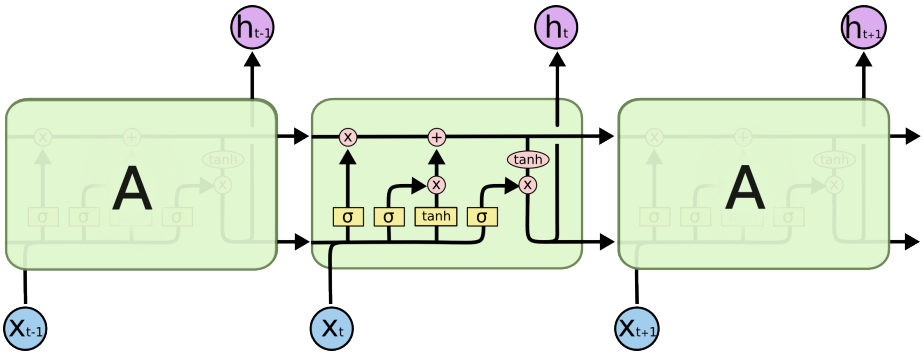
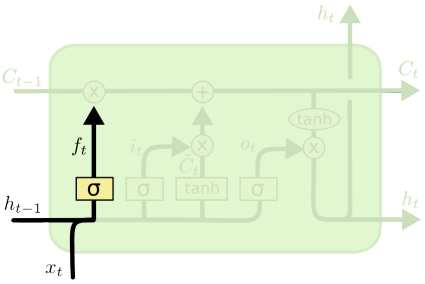


Figure 3: Opakující se perceptron v LSTM, obsahující 4 vrstvy, zdroj http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Na obrázku 3.6 je vidět, že vstupem do uzlu sítě je celý vektor vystupující z předchozího uzlu. Růžová kolečka značí operace nad vektory, kdežto žluté

čtverečky značí jednotlivé vrstvy neuronové sítě.

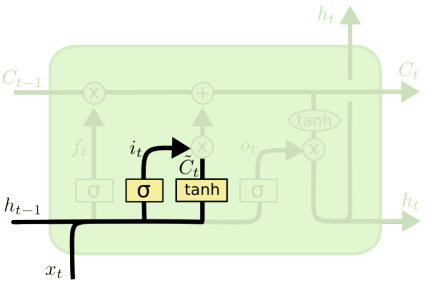
Prvním krokem ve vyobrazeném LSTM je rozhodování, které informace se mohou zahodit, a naopak, které jsou zapotřebí. Toto je realizováno pomocí *sigmoid* vrstvy, která je též nazývána jako *forget gate layer*. Funkce přijímá na vstupu výstup *h_t-1* a x_t a na výstupu je poté vypočteno číslo v rozmezí 0 až 1 pro každý perceptron C_{t-1}. 0 značí *kompletně se zbavit vstupu* a naopak 1 značí *zcela zachovat vstup*.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 4: Výpočet zahození vstupu, zdroj http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Dalším krokem je rozhodování, kterou novou informaci uchováme ve stavu. Tuto fázi lze rozdělit do dvou částí. První je *sigmoid* vrstva, též nazývána *input gate layer*, jejímž úkolem je rozhodnout, které hodnoty nahradíme. Druhá vrstva je *tanh* vytvářející vektor nových kandidátů $\tilde{C}$, kteří by mohli být přidáni do stavu. Spojením těchto dvou kroků sestavíme nový stav.

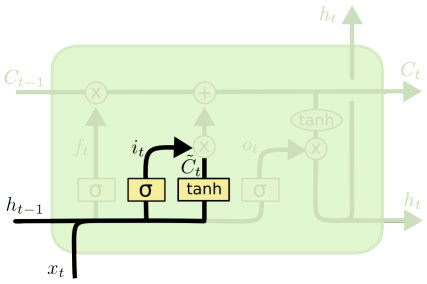


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 5: Vytváření nového stavu.

Nyní je potřeba aktualizovat starý stav C_{t-1} novým stavem C_t. Předchozí krok určil, jak nový stav bude vypadat, teď je potřeba toto rozhodnutí uskutečnit.

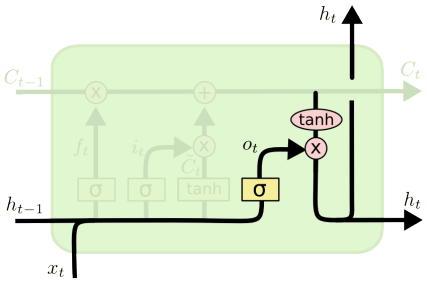
Vynásobíme stav C_{t-1} pomocí f_t, což způsobí *zapomenutí* informací, které jsme rozhodli v předchozím kroku. Poté přičteme $i_t * \tilde{C}$. Získáme tak nového kandidát upraveného podle naší potřeby na aktualizaci každé hodnoty stavu.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 6: Aktualizace stavu.

V poslední fázi je potřeba rozhodnout, co bude na výstupu. Tento výstup je daný na stavu perceptronu, ale bude ještě filtrovaný. Nejdříve projde vstup do *sigmoid* vrstvy. Ta rozhodne, která část daného stavu bude dána na výstup perceptronu. Poté *proženeme* upravený stav funkcí *tanh* (nastaví hodnoty v rozmezí -1 až 1) a vynásobíme tuto funkci takzvanou *sigmoid branou*, která určí požadovaný výstup.



$$o_t = \sigma\left(W_o \; [\, h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 7: Filtrování stavu.

V tomto projektu je možné natrénovat dva typy modelů – One-Step model nebo Multi-Step model. Oba využívají výše zmíněný LSTM, rozdíl je v jejich predikci. One-Step model dělá predikci pouze jeden časový úsek, který je velký podle velikosti okno zvoleného před samotným trénováním modelu. Pokud velikost tohoto okna je rovna 1, tak tento model bude predikovat vývoj na burze 5 minut dopředu. Pokud bychom chtěli mít regresi například hodinu dopředu, bylo by nutné nastavit toto okno na velikost 12. Z toho vyplývá, že velikost regrese tohoto modelu je $velikostokna * 5$.

Multi-Step model se liší od One-Step modelu tím, že jeho predikce není pouze jeden časový úsek, ale více pětiminutových úseků. Celková velikost (počet pětiminutových úseků) závisí stejně jako u One-Step modelu na velikosti trénovacího okna.

3.7 Window generator

Modely predikují *output* na základě okna po sobě jdoucích vzorků z dat. Metoda *__init__* zahrnuje veškerou potřebnou logiku pro inputs a labels. Dále jsou vstupem této metody dataframy train, test a eval. Metoda *split_window* dělí vstup na *input* a *labels*. Nakonec metoda *make_dataset* konvertuje *dataframe* pomocí funkce *preprocessing.timeseries_dataset_from_array* na *tf.data.Dataset* [4].

3.8 Constants

Soubor obsahuje veškeré globální proměnné, které jsou potřeba k nastavení parametrů před trénováním dat. Je zde například nastavení velikosti okna, velikost dat, velikost trénovací/evaluační/testovací sady, typ modelu, název modelu a mnoho dalšího. V rámci zlepšování tohoto projektu je v plánu převést většinu těchto proměnných, tak aby se generovali automaticky a uživatel je nemusel zadávat.

3.9 Testování

Pro testování byly napsány vlastní testovací, které měří přesnost modelů. Jelikož je možnost trénování dvou modelů, bylo nutné implementovat více testovacích funkcí – celkem tři. První je společná pro oba typy, druhá a třetí jsou pouze pro Multi-Step modely.

FAILS in direction slope vector accuracy Udává chybovost v procentech s jakou síť špatně predikovala záporný resp. kladný sklon linearní regresse vektoru.

Ratio slope vector accuracy Vyhodnocuje přesnost neboli poměr vyjádřený v procentech sklonu linearní regrese mezi realným a predikovaným vektorem.

FAILS in last value accuracy Udává chybovost v procentech s jakou síť špatně predikovala pokles resp. růst pro poslední hodnotu vektorů.

4 Instalace a spuštění

Seznam potřebných knihoven k instalaci:

Python Vysokoúrovňový skriptovací jazyk. Nabízí dynamickou kontrolu datových typů a podporuje různá programovací paradigmata, včetně objektově orientovaného, imperativního, procedurálního nebo funkcionálního.

Tensorflow Open-source knihovna pro matematické výpočty využívající grafy pro tok dat.

SciPy Software pro matematické, vědecké a technické práce.

Pandas Pro analýzu a manipulaci s daty.

Scikit-learn Jendoduchý nástroj pro analýzu prediktivních dat.

Matplotlib Knihovna pro vykreslování dat. Je možné vygenerovat křivky, histogramy,spektra, tabulky a mnoho dalších užitečných věcí.

4.1 Spuštění

Projekt je možné pustit ve čtyrech různých módech. Jedná se o zavolání příkazu *python main.py [MÓD]*. Níže je uveden seznam módů spolu s jejich popisem.

"-l", "--load" Tento mód načte data ze složky *data*, provede jejich úpravu (data preprocessing) a následně tyto upravená data uloží do tří *pickle* souborů. Musí být spuštěn jako první.

"-r", "--train" Druhý mód provede natrénování modelu z uložených dat a uloží tento model do složky *models*.

"-p", "--predict" Třetí mód provede predikci na natrénovaném modelu a výsledky predikce uloží do *pickle* souboru.

"-t", "--test" Poslední mód načte vypredikované data ze souboru a provede na nich testování.

Ve složce *models* jsou již předtrénovaný modely. Čtyři One-step a tři Multi-step modely. Je možné pustit testování na těchto modelech, stačí pouze v souboru *constants.py* změnit hodnotu proměnné *MODEL_NAME* na název chtěného modelu a proměnnou *IS_MULTI_STEP* nastavit na *True* nebo *False* podle druhu modelu. True pro Multi-step model a false pro One-step model. Poté je nutné pustit projekt v módu *predict* a poté je možné pustit testování.

5 Ukázky z běhu

V této sekci jsou ukázky (screenshots) z aplikace. Například ukázky trénování, testování apod.

5.1 Čas jako sinus/cosinus

Na následujícím obrázku je zobrazen čas převedený na sinus a cosinus. Jedno je pro den, druhý pro týden a třetí pro měsíc. Důvodem proč v těchto grafech jsou skoky (tvar není přesně sinusovka/cosinusovka) je absence dat. Jedná se o to, že pro daný čas nejsou data k dispozici (například se v tuto dobu neobchodovalo).

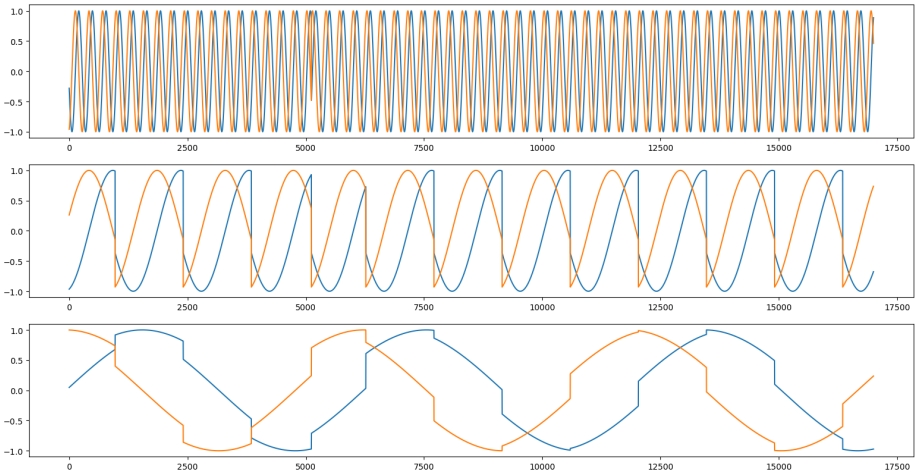


Figure 8: Čas převedený na sinus/cosinus.

5.2 Trénovaní

Zde je příklad toho, jak je v terminálu vidět průběh trénování.

```
Epoch 1/100
857/857 [==============================] - 648s 756ms/step - loss: 2.2545 - mean_absolute_error: 0.9913 - val_loss: 1.9045 - val_mean_absolute_error: 0.9438
Epoch 2/100
857/857 [==============================] - 599s 699ms/step - loss: 1.3066 - mean_absolute_error: 0.7700 - val_loss: 1.6889 - val_mean_absolute_error: 0.8983
Epoch 3/100
857/857 [==============================] - 664s 775ms/step - loss: 1.2309 - mean_absolute_error: 0.7429 - val_loss: 1.5183 - val_mean_absolute_error: 0.8589
Epoch 4/100
857/857 [==============================] - 657s 767ms/step - loss: 1.1937 - mean_absolute_error: 0.7293 - val_loss: 1.4068 - val_mean_absolute_error: 0.8187
Epoch 5/100
857/857 [==============================] - 661s 771ms/step - loss: 1.1728 - mean_absolute_error: 0.7207 - val_loss: 1.4141 - val_mean_absolute_error: 0.8240
Epoch 6/100
857/857 [==============================] - 627s 732ms/step - loss: 1.1590 - mean_absolute_error: 0.7138 - val_loss: 1.3369 - val_mean_absolute_error: 0.7952
Epoch 7/100
857/857 [==============================] - 645s 753ms/step - loss: 1.1111 - mean_absolute_error: 0.6969 - val_loss: 1.3366 - val_mean_absolute_error: 0.7946
Epoch 8/100
857/857 [==============================] - 495s 577ms/step - loss: 1.1150 - mean_absolute_error: 0.6987 - val_loss: 1.3756 - val_mean_absolute_error: 0.8076
Epoch 9/100
857/857 [==============================] - 522s 610ms/step - loss: 1.0992 - mean_absolute_error: 0.6901 - val_loss: 1.3166 - val_mean_absolute_error: 0.7833
Epoch 10/100
857/857 [==============================] - 634s 739ms/step - loss: 1.0803 - mean_absolute_error: 0.6840 - val_loss: 1.3258 - val_mean_absolute_error: 0.7872
Epoch 11/100
857/857 [==============================] - 631s 736ms/step - loss: 1.0712 - mean_absolute_error: 0.6807 - val_loss: 1.3236 - val_mean_absolute_error: 0.7908
```

Figure 9: Průběh trénování.

5.3 Trénovaní

Příklady testování, jak pro One-Step, tak i pro Multi-Step.

Figure 10: Výsledek testu pro One-Step model.

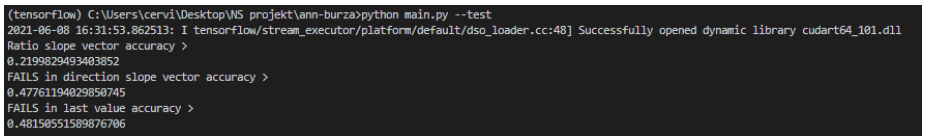


Figure 11: Výsledky testů pro Multi-Step model.

6 Výsledky testování

Vyhodnocení by se dalo rozdělit na tři části. Porovnání One-Step modelů, porovnání Multi-Step modelů a porovnání One-Step modelů a Multi-Step modelů mezi sebou.

Pro otestování One-Step modelů byly natrénovány čtyři modely s různými parametry. Výsledky těchto vyhodnocení jsou v tabulce 1 níže:

Model	Velikost dat	Počet epoch	Výsledek	Window
Onestep-1	50 000	5 000	0.48462393	288*5
Onestep-2	598 000	500	0.49451362	288
Onestep-3 (hour)	598 000	500	0.49967232	288
Onestep-4 (hour)	598 000	500	0.49743827	288*5

Table 1: Výsledky pro One-Step modely.

Dále byly natrénovány tři Multi-Step modely viz tabulka 2. Výsledek 1 je výsledek pro testovací funkci *FAILS in direction slope vector accuracy* a výsledek 2 je pro testovací funkci *Ratio slope vector accuracy*.

Model	Velikost dat	Počet epoch	Výsledek 1	Výsledek 2	Window
Multistep-1	50 000	5 000	0.47761194	0.21998295	288*5
Multistep-2	30 000	5 000	0.49981433	0.16004868	288*5
Multistep-3 (hour)	598 000	500	0.50681501	0.05847375	288

Table 2: Výsledky pro Multi-Step modely.

Při porovnání One-Step a Multi-Step modelů nebyly natrénovaný žádný nový modely, pouze byly porovnány One-Step modely pro hodinovou regresi a Multi-Step modely, pro které byla vypočtena přesnost pro hodinovou regresi. Výsledky jsou v tabulce 3. U One-Step modelů je výsledek shodný jako v tabulce 1, zatímco pro Multi-Step modely je výsledek vypočten pomocí testovací funkce *FAILS in last value accuracy*.

Model	Velikost dat	Počet epoch	Výsledek
Onestep-3 (hour)	598 000	500	0.49967232
Onestep-4 (hour)	598 000	500	0.49743827
Multistep-3	598 000	500	0.5027982
Multistep-1	50 000	5 000	0.4815055

Table 3: Porovnání One-Step a Multi-step modelů.

References

[1] Andrés Arévalo et al. "Algorithmic Trading Using Deep Neural Networks on High Frequency Data". In: Aug. 2017, pp. 144–155. ISBN: 978-3-319-66962-5. DOI: 10.1007/978-3-319-66963-2_14.

[2] Michal Chlud. "Algoritmické obchodování na burze s využitím umělých neuronových sítí". Bachelor's Thesis. Vysoké učení technické v Brně, 2016.

[3] Thomas Fischer and Christopher Krauss. "Deep learning with long short-term memory networks for financial market predictions". In: *European Journal of Operational Research* 270.2 (2018), pp. 654–669. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2017.11.054. URL: https://www.sciencedirect.com/science/article/pii/S0377221717310652.

[4] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[5] Vítězslav Slavík. "Algoritmické obchodování na burze s využitím umělých neuronových sítí". Bachelor's Thesis. Vysoká škola ekonomická v Praze, 2019.