Master Thesis: *Natural gas price forecasting using recurrent neural networks*

Christian Koopmann

School of Business and Economics
Chair of Information Systems
Humboldt–Universitaet zu Berlin
http://https://www.wiwi.hu-berlin.de/de/professuren/quantitativ/wi

Outline

Motivation

- *Methodology:* Evaluate the value of Long Short Term Memory Recurrent Neural Networks for time series prediction
  - Compare performance to simple RNN, Feed Forward Neural Networks (FFNN), and linear reference models
- *Application:* Support natural gas traders in choosing the optimal trading strategy
  - Trading as part of industrial procurement to meet physical demand
  - No speculative trading

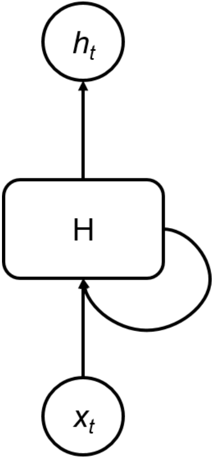
Description of the target variable / gas prices

- Different types of natural gas futures differ in various ways
- The **Virtual Trading Point** describes in which part of the European natural gas transport network the gas is delivered
- The **Delivery Period** describes during which time frame the gas is delivered at a constant rate
- The target variable is the future price of gas traded at the **TTF** VTP for delivery in the **next calendar month**.

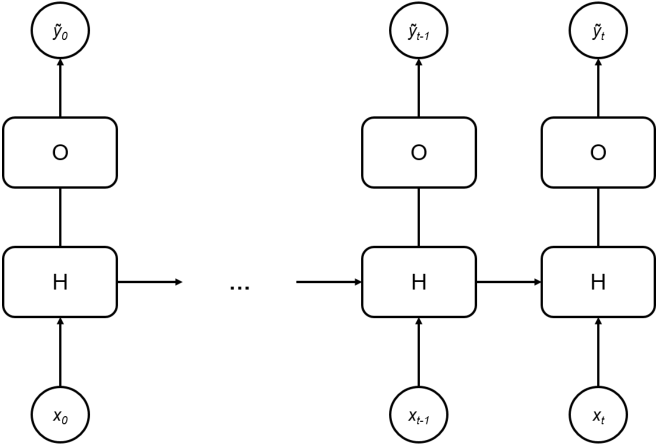
Why RNN ?

- FFNN can only learn static input-output mappings
- For machine learning problems based on sequential data the input-output mapping should be dynamic
- Examples of sequential data: Text, Speech, Videos, Financial Time Series
- RNN's are able to learn dependencies of arbitrary length, which does not need to be specified.
- Main idea: use hidden layer output at one time point as input to the hidden layer at the next input

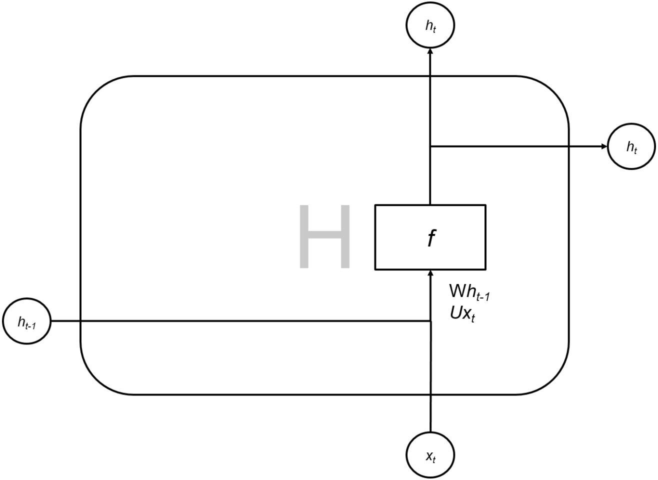
RNN Layout



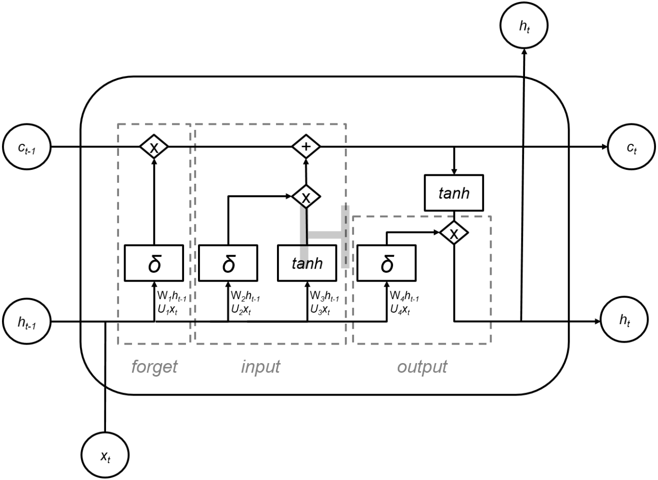
RNN Layout



Hidden Layer of Simple RNN



Hidden Layer of LSTM



Data Overview

- All Data was downloaded from the *Thomson Reuters Eikon* data base.
- **Energy Commodity Prices:** gas futures, Brent Oil Futures, Coal Futures, Electricity Base / Peak futures
- **Exchange Rates:** EUR/GBP, EUR/USD
- **Gas Market Fundamentals:** Storage Levels, Pipeline Flows, National Consumption / Production Data
- All data as daily values (Closing prices) starting between 2010 - 2014.

Training / Tuning Approach

1. Parameter Tuning of univariate models
2. Variable Selection of multivariate models
3. Parameter Tuning of multivariate models
4. Model evaluation

Predicting Price Levels

- Predict tomorrows closing price of the TTF Front Month future based on all data available up to the current day
- The MSE loss function is minimized using stochastic gradient descent
- Linear Reference Model: AR(1)

Price Level Prediction Results

	Model	Variables	MSE	MSEReference
1	lstm		0.143	0.151
2	rnn		0.150	0.151
3	ffnn_long		0.151	0.151
4	ffnn_long	EURGBPFX_EURUSDFX_TradeIUK_TradeRUNWE	0.152	0.151
5	rnn	TTFDA	0.154	0.151
6	ffnn_short		0.162	0.151
7	ffnn_short	EURGBPFX_TTFDA_ProdNL_TradeBBL_NBPFM	0.162	0.151
8	lstm	TradeIUK	0.181	0.151

Table 1: Test Results using monthly cross validation of tuned models for data 01 - 07/2017

Binary Prediction

- **Problem:**
  - Decision Problem: Buy today or at any other day before the end of the month to cover physical demand
  - Once bought, futures can't be sold again (Company Policy, Regulation).
  - Therefore optimal trading strategy can't be derived directly from tomorrows price level
- **Solution:**
  - New binary target variable: *Is today's closing price minimal among all closing prices for the rest of the month ?*
  - Yes - 1, No - 0
  - Loss Function: Binary Cross Entropy
  - Naive reference model: $\tilde{y}_i = \frac{1}{N_i}$ with N_i the remaining trading days for this month

Binary Prediction Results

	Model	Variables	CrossEntropy	CrossEntropyReference
1	mlp_long	EURUSDFX	0.447	0.497
2	rnn	EURGBPFX	0.454	0.497
3	ffnn_short	ElectricityBaseFM_EURUSDFX	0.454	0.497
4	lstm		0.468	0.497
5	lstm	ProdUKCS	0.469	0.497
6	rnn		0.486	0.497
7	ffnn_short		0.487	0.497
8	ffnn_long		0.517	0.497

Table 2: Test Results using monthly cross validation of tuned models for data 01 - 07/2017

Conclusions

- Among univariate models LSTM outperforms alternative models in both prediction problems
- Among multivariate problems opposite seems to be the case
- Univariate LSTM shows best relative performance in Price Level Prediction where it is the only model to significantly outperform the Linear Reference
- Univariate models seem to be better in Price Level Prediction where the opposite is true for the binary case.

Outlook

There are several ways for possible extension / improvement

- Extend parameter tuning to choice of optimizer, activation, length, batch size etc.
- Extend parameter tuning to include multi-level architectures
- Use a finer tuning grid for the current parameters
- Use Cross Validation during parameter tuning

Comments

- Model Training / Tuning: Python (Keras, Tensorflow), Result Analysis: R
- Running models on Amazon Web Services, is easier / cheaper than expected and frees valuable resources on local machine
- Valuable Resources on LSTMs and implementation in Python:
  - BlogPost *Understanding LSTM Networks*: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
  - ML Blog *Machinelearningmastery*: https://machinelearningmastery.com/

Simple RNN

- Different types of RNN differ in the way they connect the hidden layers between time steps
- The simplest variant treats the previous output h_t in the same way as the other inputs x_t
- $h_t = f(Wh_{t-1} + Ux_t)$

Vanishing Gradient Problem

- Recursive definition of the hidden layer can be expanded:
  - $h_t = f(Wh_{t-1} + Ux_t)$
  - $h_t = f(Wf(Wh_{t-2} + Ux_{t-1}) + Ux_t)$
- Repeated application of chain rule:
  - $\frac{h_t}{dW_{ij}} = \frac{df(Wh_t)}{dWh_t}(\frac{dW}{dW_{ij}}h_{t-1} + W\frac{dh_{t-1}}{dW_{ij}})$
  - $\frac{h_t}{dW_{ij}} = \frac{df(Wh_t)}{dWh_t}(\frac{dW}{dW_{ij}}h_{t-1} + W\frac{df(Wh_{t-1})}{dWh_{t-1}}(\frac{dW}{dW_{ij}}h_{t-2} + W\frac{dh_{t-2}}{dW_{ij}}))$
  - $\frac{dh_t}{dW_{ij}} = \sum_{k=1}^{t}(\prod_{l=1}^{k}\frac{df(Wh_{t-l})}{dWh_{t-l}})W^{k-1}\frac{dW}{dW_{ij}}h_{t-k}$

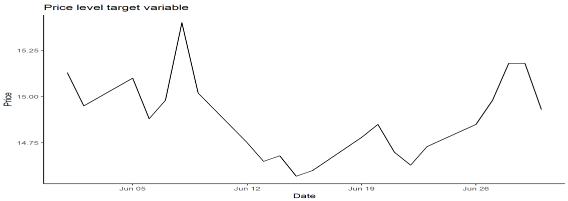
Vanishing Gradient Problem

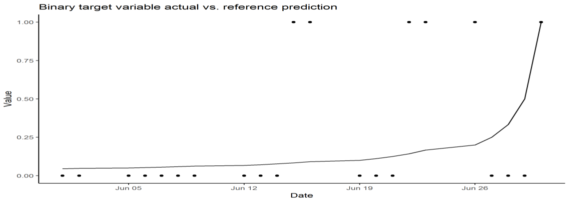
- Contribution of $\frac{dh_{t-k}}{dW_{ij}}$ is multiplied by $(\prod_{l=1}^{k} \frac{df(Wh_{t-l})}{dWh_{t-l}})W^{k-1}$
  - Exponential behaviour leads to either *exploding* or *vanishing gradient problem*
  - *Exploding* case can be controlled relatively easily by clipping the gradient
  - No solution of *vanishing gradient problem* in this model set-up
  - RNNs unable to learn long term dependencies

LSTM

- Long Short Term Memory Networks try to overcome this problem by introducing the cell state c_t
  - c_t is manipulated in different *gates*
  - In each gate c_t is multiplied by or added to the output of a layer of neurons with trained weights
- Formal definition:
  - $h_t = o_t * tanh(c_t)$
  - $c_t = f_t * c_{t-1} + i_t$
  - $o_t = \sigma(W_4 h_{t-1} + U_4 x_t)$
  - $i_t = \sigma(W_2 h_{t-1} + U_2 x_t) * tanh(W_3 h_{t-1} + U_3 x_t)$
  - $f_t = \sigma(W_1 h_{t-1} + U_1 x_t)$
- LSTM does not suffer from *vanishing gradient problem* but has four times as many parameters to train with same input data.

Binary Prediction





Training / Tuning Approach

1. Parameter Tuning of univariate models
   - Single Train / Test split
   - Training Data 2010 - 2015 / Test Data: 2016
   - Tuning of: Network Architecture, Dropout, Learning Rate
2. Variable Selection of multivariate models
   - Use tuned parameters from respective univariate model
   - Forward variable selection based on MSE
   - Same Train / Test split as above
3. Parameter Tuning of multivariate models
   - Use previously selected input variables
   - Same parameters / data as in univariate case
4. Model evaluation
   - Month wise cross validation with testing months selected from 01 - 08/2017

Binary Prediction Univariate parameter tuning

	Model	Architecture	Dropout	LearningRate	binary_crossentropy
1	lstm	16	0.250	0.001	0.450
2	rnn	8	0.250	0.001	0.449
3	mlp_long	32	0.000	0.001	0.468
4	mlp_short	8	0.000	0.010	0.473

Table 3: Best parameter combinations for each binary model

Binary Prediction Variable Selection

	Model	Variables	binary_crossentropy
1	lstm	ProdUKCS	0.45
2	rnn	EURGBPFX	0.44
3	mlp_long	EURUSDFX	0.47
4	mlp_short	ElectricityBaseFM_EURUSDFX	0.47

Table 4: Best variable combinations for each binary model

Binary Prediction Multivariate parameter tuning

	Model	Variables	Architecture	Dropout	LR	CE
1	lstm	ProdUKCS	32	0.500	0.001	0.452
2	rnn	EURGBPFX	8	0.250	0.001	0.444
3	mlp_long	EURUSDFX	32	0.000	0.001	0.478
4	mlp_short	ElectricityBaseFM_EURUSDFX	16	0.000	0.010	0.461

Table 5: Best parameter combinations for each binary model

Price Level Prediction Univariate parameter tuning

	Model	Architecture	Dropout	LearningRate	mse
1	lstm	8	0.000	0.001	0.117
2	rnn	8	0.000	0.001	0.122
3	mlp_long	8	0.000	0.100	0.736
4	mlp_short	16	0.000	0.100	0.459

Table 6: Best parameter combinations for each binary model

Price Level Prediction Variable Selection

	Model	Variables	mse
1	lstm	EURUSDFX	0.12
2	rnn	ElectricityBaseFM	0.11
3	mlp_long	TradeRUNWE_ConLDZNL	0.16
4	mlp_short	TTFDA_StorageEU_LNGStockEU_ProdNL	0.22

Table 7: Best variable combinations for each level prediction model

Price Level Prediction Multivariate parameter tuning

	Model	Variables	Architecture	Dropout	LR	ms
1	lstm	EURUSDFX	32	0.250	0.001	0.11
2	rnn	ElectricityBaseFM	8	0.000	0.001	0.14
3	mlp_long	TradeRUNWE_ConLDZNL	16	0.000	0.100	0.18
4	mlp_short	TTFDA_StorageEU_LNGStockEU_ProdNL	8	0.000	0.100	0.17

Table 8: Best parameter combinations for each binary model