Information Retrieval and DataMining: Memory Models for Time Series Analysis

Mark Neumann, Alexander Chapovskiy and Zheng Tian
UCL
mark.neumann.15@ucl.ac.uk
alexander.chapovskiy.15@ucl.ac.uk
zheng.tian.11@ucl.ac.uk@ucl.ac.uk

Abstract

This is a report for group project in the Information Retrieval and Data Mining course. In this report we investigate different deep neural network architectures on the time series. In particular, we apply neural networks to energy load and household energy consumption datasets.

1 Introduction

1.1 Memory Models

Long Short-Term memory[7] has been utilised widely across many disciplines in Machine Learning, with particularly notable performance in domains such as Machine Translation[4], Constituency Parsing[6], Speech Recognition and even generative models, such as human level handwriting generation.[5]

Particularly key in the success of LSTM and other variants, such as the GRU, has been their perceived ability to selectively encode long-term dependencies and learn precise timings. In this work, we investigate the effect of precise timings in accurately modelling time-series data, an area which should particularly benefit from the accuracy features provided by Long Short-Term memory.

1.2 Other Model Architectures

One perceived failure of neural networks until recently was their inability to deal with arbitrary sequence lengths. The input dimensions to a standard LSTM classifier/regressor are fixed, meaning that data which intrinsically contains irreducible sequences of different lengths was difficult or impossible to process. This clearly is a problem with time series data, as the noisy nature of real world data means that we are rarely going to achieve perfectly timed and sanitised data.

In [4], Sutskever introduces sequence to sequence neural networks, which utilise two separate networks, one to encode a sentence into a vector representation and the other to decode a vector representation into the model ouptut. These are trained jointly, enabling arbitrary length sequences and particularly enabling non-monotonic transformations. We note that although we do not explore sequence to sequence models in this paper, the underlying cell structure we explore can be used in this architecture and that this is a very interesting area of development for time series models given their nature.

2 Datasets

2.1 Energy Load

!!!Description of the dataset

This dataset was used in the Kaggle competition. After the competition the winners of the competition published their approaches, [1]. Mostly the approach is linear regression with non-linear features.

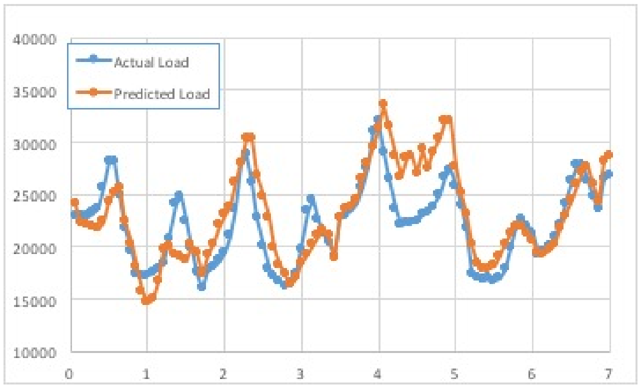
Neural networks are pplied to this dataset in [**?**]

!!!! Before we apply out models to the dataset we perform the following preprocessing.

2.2 Household Energy Consumption

3 Regression with Non-linear Features

In order to construct a baseline to examine our results, we performed linear regression with non-linear features on batches of one week's worth of data. We use square root mean squared error divided by average load, achieving 12% error using this metric. The figure below shows a representative week for one of the zones with actual and predicted loads shown.



4 Reccurent Neural Networks

Below we describe the models applied to the dataset, for which performance can be found in the next section. In addition to vanilla Long Short-Term Memory, we explore several recent innovations over the last few years, including Gated Recurrent Units and peephole connections.

4.1 LSTM

Given an input vector $x_t \in \mathbb{R}^N$ at time t and the previous output of the LSTM $h_{t-1} \in \mathbb{R}^K$, the forget gate(f_t), the input gate(i_t), the output gate(o_t) and the cell input(z_t) are computed as follows:

$$\begin{aligned} H &= \begin{bmatrix} x_t \\ h_{t-1} \end{bmatrix} \in \mathbb{R}^{N+K} & o_t &= \sigma(W_o H + b_o) \\ & & f_t &= \sigma(W_i H + b_i) \\ z_t &= Tanh(W_z H + b_z) & c_t &= f_t \odot c_{t-1} + i_t \odot z_t \\ i_t &= \sigma(W_i H + b_i) & h_t &= o_t \odot Tanh(c_t) \end{aligned} \tag{1}$$

Where $W_z, W_i, W_o, W_f \in \mathbb{R}^{K\times(N+K)}$ and $b_z, b_i, b_o, b_f \in \mathbb{R}^K$ are trained weight matrices and bias vectors for N, the dimension of the input vector and K the hidden size of the LSTM. σ and $Tanh$ are element-wise activation functions representing the sigmoid and hyperbolic tangent functions, with $\odot$ representing element-wise multiplication.

Below we present a pictorial demonstration of the connections in the LSTM cell.

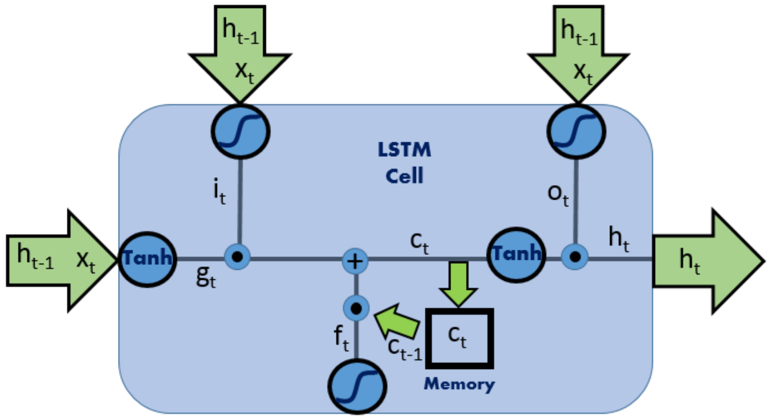


Figure 1: A demonstration of the interactions within an LSTM cell. Here we can see clearly how the input and output control the respective features of the cell: the closer to zero these vectors are, the smaller the amount of information which can travel through the dot product.

4.2 Gated Recurrent Unit

The Gated Recurrent Unit[2] can be seen as a simplified LSTM with a tied output and forget gate, merging two of the above equations.

Given an input vector $x_t \in \mathbb{R}^N$ at time t and the previous output of the GRU $h_{t-1} \in \mathbb{R}^K$, the forget gate(f_t), the input gate(i_t) and cell output(h_t) are computed as follows:

$$\begin{aligned} &Input : x_t && f_t = \sigma(W_f^x x_t + W_f^h h_{t-1} + b_f) \\ &PrevState : h_{t-1} && \tilde{h_t} = Tanh(W x_t + f_t \odot h_{t-1}) \\ &i_t = \sigma(W_i^x x_t + W_i^h h_{t-1} + b_i) && h_t = i_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h_t} \end{aligned} \tag{2}$$

Where $W_i^x, W_f^x, \in \mathbb{R}^{K\times(N)}, W_i^h, W_f^h \in \mathbb{R}^{K\times(K)}$ and $b_i \in \mathbb{R}^N, b_f \in \mathbb{R}^K$ are trained weight matrices and bias vectors for N, the dimension of the input vector and K the hidden size of the GRU. σ and $Tanh$ are again element-wise activation functions representing the sigmoid and hyperbolic tangent functions, with $\odot$ representing element-wise multiplication as above.

It should be noted that subsequent papers by Cho expand on the GRU, particularly to incorporate ”recurrent feedback”, intuitively described as a linear combination of memory storage across hidden layers. This would certainly be an interesting extension to the work in this paper.

4.3 Peephole Connections

Peephole connections are a popular addition to the Long Short-Term memory model[3]. Originating in a paper by Schmidhuber and Gers whilst working at the IDSIA lab in Switzerland, they quickly became popular in an array of literature. The basic premise is that the gates of the LSTM mechanism become a function not only of the current input x_t and previous LSTM state h_{h-1}, but also a function of the current cell memory state which persists from the last timestep, c_{t-1}. The input, output and forget gates are therefore defined as follows:

$$H = \begin{bmatrix} x_t \\ h_{t-1} \\ c_{t-1} \end{bmatrix} \in \mathbb{R}^{N+2K} \qquad \begin{aligned} i_t &= \sigma(W_i H + b_i) \\ f_t &= \sigma(W_f H + b_f) \\ o_t &= \sigma(W_o^x x_t + W_o^h h_{t-1} + W_o^c c_t + +b_o) \end{aligned} \tag{3}$$

where the memory is simply appended onto the vector used in the LSTM mechanism for the calculation of the input and forget gates, with the output gate using a different format only because it uses the *current* state's memory, rather than the previous one as in the input and forget gates. The other functionality remains unchanged. Additionally, some practitioners choose to only apply peephole connections to gates selectively - often, they are only applied to the input and output gates and not the forget gate. Note the effect of peephole connections is difficult to measure accurately, as in a vanilla LSTM, all the gates are implicitly a function of the memory vector, as they are a function of the previous state which is informed by the memory at that point. Below we present a pictorial representation of how the addition of peephole connections changes the internal cell architecture.

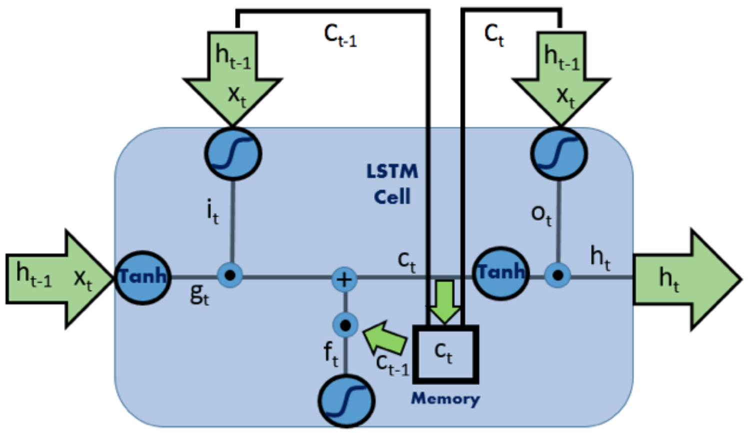


Figure 2: Here we can clearly see how all gates now have direct interaction with the memory state.

5 Results

6 Conclusions

Overall, we have examined the performance of peephole connections and neural network architectures in general for several tasks, drawing mixed results.

References

[1] Tao Hong, Pierre Pinson and Shu Fan, (2014), *Global Energy Forecasting Competition 2012*. International Journal of Forecasting, 30(2), 357;

[2] Junyoung Chung and Çaglar Gülçehre and KyungHyun Cho and Yoshua Bengio (2014) *Gated Recurrent Neural Networks*

[3] Gers, F. and Schmidhuber, J. (2000) *Recurrent Nets that time and Count*

[4] Sutskever, I., Vinyals, O., Le, Q.V. (2015)
Sequence to Sequence Learning with Neural Networks

[5] Graves, A. (2013) *Generating Sequences With Recurrent Neural Networks*

[6] Oriol Vinyals and Lukasz Kaiser and Terry Koo and Slav Petrov and Ilya Sutskever and Geoffrey E. Hinton, (2015) *Grammar as a Foreign Language*

[7] Sepp Hochreiter and Jrgen Schmidhuber(1995) Long Short-Term Memory
Charlton, N. and Singleton, C. (2014). A refined parametric model for short term load forecasting. International Journal of Forecasting, 30(2), 364-368.
Souhaib Ben Taieba and Rob J. Hyndmanb, (2014), A gradient boosting approach to the Kaggle load forecasting competition. International Journal of Forecasting, 30(2), 382;
Lloyd, J. R. (2014), GEFCom2012 hierarchical load forecasting: Gradient boosting machines and Gaussian processes. International Journal of Forecasting, 30(2), 369.