LSTM: A Search Space Odysseys

Mușat Bogdan-Adrian

February 2017

Recurrent Neural Networks - RNNs

- Used to deal with sequential data, where there is a temporal dependence from a time instance to another
- Mathematically, they model a conditional distribution of the form $P(x_t | x_{t-1}, ..., x_2, x_1)$, where x_t is the current input at time t
- The output of a vanilla RNN cell at each time step is computed using the current input x_t and also the previous cell state h_{t-1}[1] as:

$$y_t = tanh(Ux_t + Wh_{t-1} + b),$$

where $U \in R^{M \times N}$, $W \in R^{N \times N}$ are the shared parameter matrices for the RNN cells and $b \in R^N$ is the bias

[1]Meant to encompass a summary of the past information

Recurrent Neural Networks Unrolled

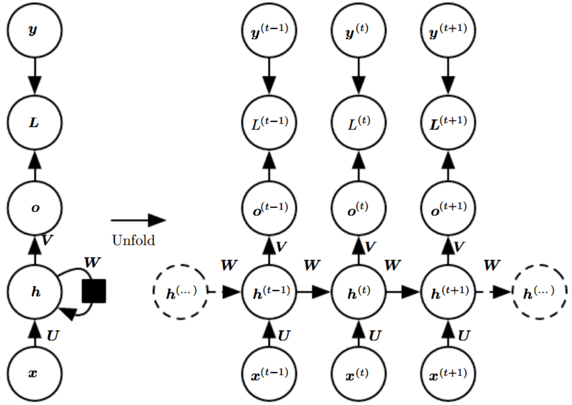

Figure 1: RNN unrolled[2]

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

Backpropagation Through Time

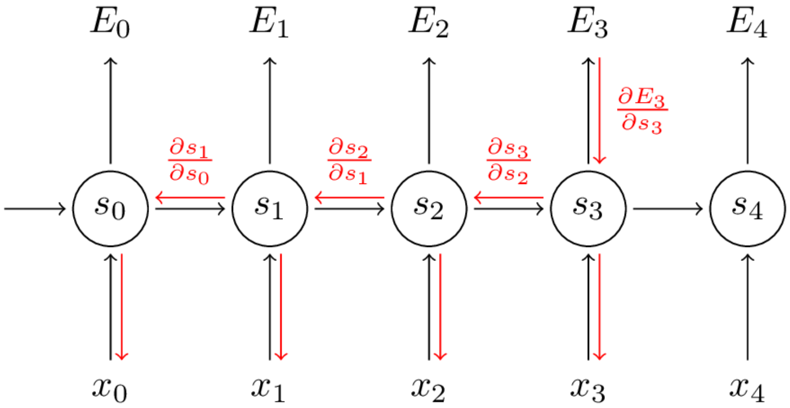
E_0
E_1
E_2
E_3
E_4
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
s_0
s_1
s_2
s_3
s_4
x_0
x_1
x_2
x_3
x_4

Figure 2: Backpropagation through time

Vanishing/Exploding Gradient

- The main algorithm for learning the weights of an RNN is called Backpropagation Through Time (BPTT)
- When computing the gradients with respect to the weights of the network, W depends on a recurrent connection from a previous timestep and so the partial derivative w.r.t. W becomes:

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial W},$$

  where $L^{(t)}$ is the error and $o^{(t)}$ is the predicted value at the t^{th} output step
- If the number of timesteps is big, the product from the formula above will either diminish to 0 or explode to ∞, thus prohibiting the learning process further on

Long Short-Term Memory - LSTM

- LSTM[3] solves the problem of vanishing/exploding gradients
- The LSTM architecture is a memory cell, which can maintain its state over time, and nonlinear gating units, which regulate the information flow into and out of the cell
- Further experiments have proven that no other variants that differ from the vanilla LSTM by adding, removing or modifying exactly one aspect can obtain a much better performance[4]

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: 9 (Dec. 1997), pp. 1735–80.

[4] Klaus Greff et al. "LSTM: A Search Space Odyssey". In: *CoRR* abs/1503.04069 (2015). arXiv: 1503.04069. URL: http://arxiv.org/abs/1503.04069.

LSTM Cell

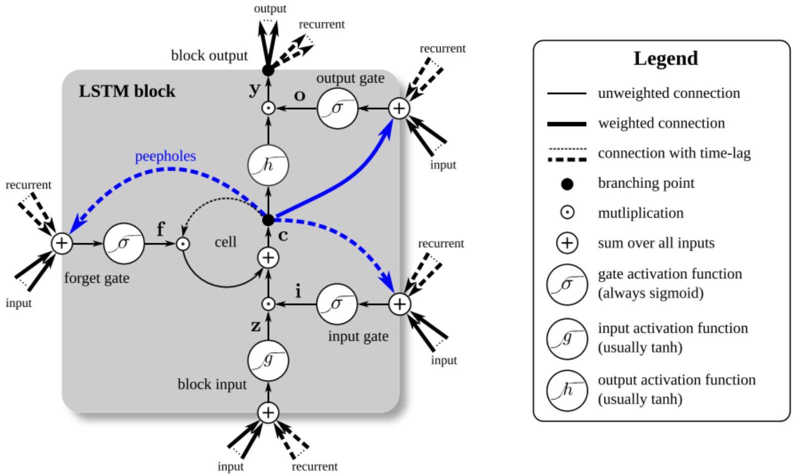

Figure 3: LSTM cell

LSTM Cell Computation

- The formulas for a vanilla LSTM layer forward pass can be written as:

$$
\begin{aligned}
\bar{z}^t &= W_z x^t + R_z y^{t-1} + b_z && \\
z^t &= g(\bar{z}^t) && \text{block input} \\
\bar{i}^t &= W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i && \\
i^t &= \sigma(\bar{i}^t) && \text{input gate} \\
\bar{f}^t &= W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f && \\
f^t &= \sigma(\bar{f}^t) && \text{forget gate} \\
c^t &= z^t \odot i^t + c^{t-1} \odot f^t && \text{cell} \\
\bar{o}^t &= W_o x^t + R_o y^{t-1} + p_o \odot c^{t-1} + b_o && \\
o^t &= \sigma(\bar{o}^t) && \text{output gate} \\
y^t &= h(c^t) \odot o^t && \text{block output}
\end{aligned}
$$

LSTM Gates

- Each gate is computed in terms of the current input, the previous output, a peephole connection to the previous cell state and a bias
- i^t, f^t and o^t are called the input, forget and output gates and their role is to determine how much information flows from previous and current timesteps. This effect is achieved by squashing the gates through a sigmoid function. Then an element wise multiplication is performed such that the input gate determines how much of the current information is used, the forget gate deals with the information flow from the previous cell state and finally, the output gate specifies how much information to send to the output y
- The peephole connections were added such that the network can learn precise timings easier[5]

[5] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning Precise Timing with Lstm Recurrent Networks". In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 115–143. ISSN: 1532-4435. DOI: 10.1162/153244303768966139. URL: https://doi.org/10.1162/153244303768966139.

Evaluation Setup

- The vanilla LSTM is used as a baseline and evaluated together with eight of its variants
- Random search was used to obtain good-performing hyperparameters
- Three datasets were used to evaluate the performance
  - The TIMIT Speech corpus
  - The IAM Online Handwriting Database
  - JSB Chorales
- The interaction between hyperparameters was observed using the fANOVA framework[6]

[6] F. Hutter, H. Hoos, and K. Leyton-Brown. "An Efficient Approach for Assessing Hyperparameter Importance". In: *Proceedings of International Conference on Machine Learning 2014 (ICML 2014)*. June 2014, 754762.

LSTM Variants

- **NIG:** No Input Gate: $i^t = 1$
- **NFG:** No Forget Gate: $f^t = 1$
- **NOG:** No Output Gate: $o^t = 1$
- **NIAF:** No Input Activation Function: $g(x) = x$
- **NOAF:** No Output Activation Function: $h(x) = x$
- **CIFG:** Coupled Input and Forget Gate: $f^t = 1 - i^t$
- **NP:** No Peepholes:

$$\begin{aligned}
\bar{i}^t &= W_i x^t + R_i y^{t-1} + b_i \\
\bar{f}^t &= W_f x^t + R_f y^{t-1} + b_f \\
\bar{o}^t &= W_o x^t + R_o y^{t-1} + b_o
\end{aligned}$$

LSTM Variants

- **FGR:** Full Gate Recurrence:

$$\begin{aligned}
\bar{i}^t =& W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i \\
& + R_{ii} i^{t-1} + R_{fi} f^{t-1} + R_{oi} o^{t-1} \\
\bar{f}^t =& W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f \\
& + R_{if} i^{t-1} + R_{ff} f^{t-1} + R_{of} o^{t-1} \\
\bar{o}^t =& W_o x^t + R_o y^{t-1} + p_o \odot c^{t-1} + b_o \\
& + R_{io} i^{t-1} + R_{fo} f^{t-1} + R_{oo} o^{t-1}
\end{aligned}$$

- The CIFG variant is also known as Gated Recurrent Unit (GRU)[7]
- The FGR variant adds recurrent connections between all the gates, thus significantly increasing the number of parameters

[7] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Results

- Figure 3 represents the test set performance for all 200 trials (top) and for the best 10% (bottom) trials (according to the validation set) for each data set and variant. Boxes: range between the 25th and 75th percentiles of the data. Whiskers: whole range. Red dot: mean. Red line: median of the data. Thick blue lines: boxes of variants that differ significantly from the vanilla LSTM. Gray histogram in the background: average number of parameters for the top 10% performers of every variant.

Results

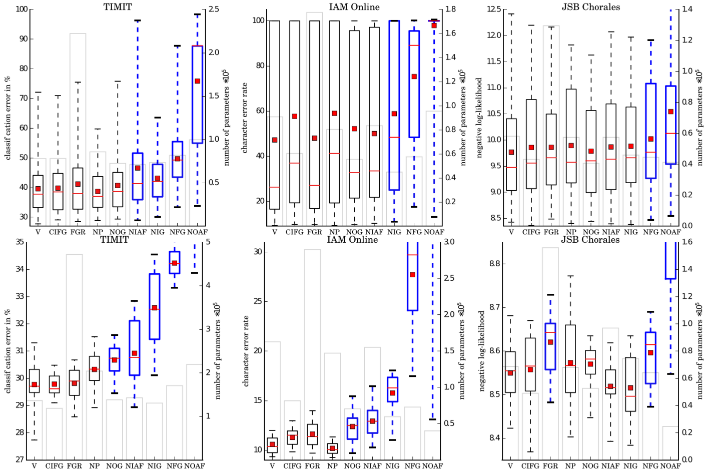

Figure 4: Box plots for the test performance

Observations

- Removing the output activation function (NOAF) or the forget gate (NFG) significantly hurt performance on all three data sets
- Input and forget gate coupling (CIFG) did not significantly change the mean performance on any of the data sets
- Removing the peephole connections (NP) also did not lead to significant changes
- Adding FGR does not significantly change performance, but increases the number of parameters
- For supervised learning on continuous real-valued data the input gate, output gate, and input activation function are crucial

Impact of Hyperparameters

- The fANOVA framework for assessing hyperparameters importance is based on the observation that marginalizing over dimensions can be done efficiently in regression trees. This allows predicting the marginal error for one hyperparameter while averaging over all the others
- The hyperparameter space is sampled at random
- The learning rate is determined as being the most important hyperparameter. A practical advice would be to tune it on a small network and then usee it to train a large one
- The hidden layer size is the second in order of importance. Larger networks perform better, but with diminishing returns.
- The additive Gaussian noise on the inputs almost always hurt performance and slightly increases training times
- Momentum affects neither performance nor training time in any significant way

Analysis of Variance

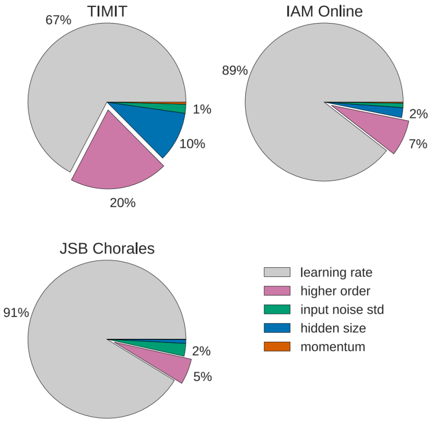

Figure 5: Pie charts showing which fraction of variance of the test set performance can be attributed to each of the hyperparameters

Interaction of Hyperparameters

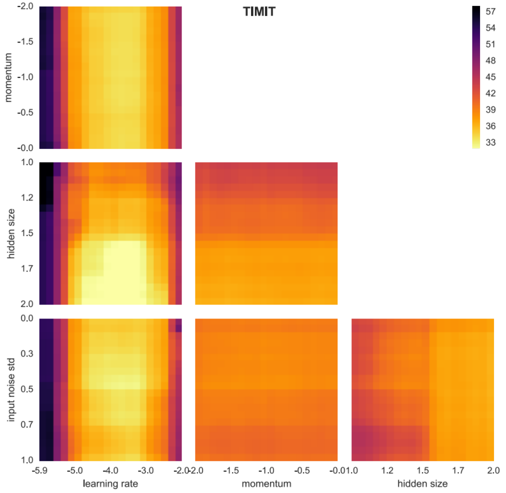

Figure 6: Hyperparameter interaction on TIMIT

Interaction of Hyperparameters

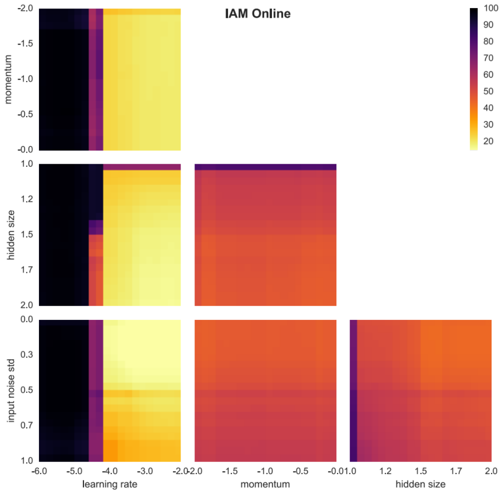

Figure 7: Hyperparameter interaction on IAM Online

Interaction of Hyperparameters

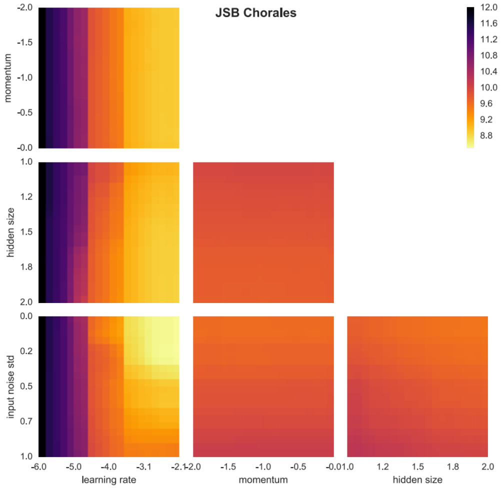

Figure 8: Hyperparameter interaction on JSB Chorales

Conclusions

- Vanilla LSTM performs reasonably well on a various data sets, none of the eight investigated modifications significantly improving the performance
- Simpler LSTM variants like CIFG and NP might be attractive because of the reduction in the number of parameters
- The forget gate and the output activation function are the most critical components of the LSTM block
- The learning rate is the most important hyperparameter, followed by the network size
- The analysis of hyperparameters interaction revealed no apparent structure, meaning that they can be treated as approximately independent

Technical Slides - Vanishing/Exploding Gradients Derivations

$$\frac{\partial L}{\partial W} = \sum_{t=0}^{S} \frac{\partial L^{(t)}}{\partial W}$$

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(k)}} \frac{\partial h^{(k)}}{\partial W}$$

$$\frac{\partial h^{(t)}}{\partial h^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial h^{(i)}}{\partial h^{(i-1)}}$$

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial W}$$