CS230

Deep Learning Application in Well Production Problems

Tita Ristanto
Department of Energy Resources Engineering
Stanford University
ristanto@stanford.edu

Abstract

Getting an accurate flow rate forecast is important in oil and gas, CO_2 sequestration, and geothermal industry. This project focused on deep learning architecture search to accurately predict flow rate. Our task was to design and train a deep learning model/network; that is, given pressure from multiple wells, predicts the flow rate in these wells. This project explored the performance of different types of recurrent layers and the analysis showed that GRU performed better than LSTM and Vanilla RNN in our setting.

1 Introduction

In oil and gas, CO_2 sequestration, and geothermal fields, it is important to understand the behavior of the reservoir, often by seeing the dynamics of bottom hole pressure (p_{wf}) and flow rate (q) in each well. Bottom hole pressure and flow rate are commonly modeled using physical model, for example, reservoir numerical simulator. This approach requires physical parameters, including rock properties and reservoir geometry, which we do not always know. In some cases, collecting those physical parameters and building the physical model might not be practical. Applying this approach in a field that has thousands of wells with a very dynamic environment requires a lot of effort.

This study introduced an alternative approach using the sequence model in deep learning. A reservoir model was developed by learning the historical pattern of bottom hole pressure and flow rate without the physics of the flow being programmed explicitly. Using this model, given bottom hole pressure from n wells, we can predict flow rate of n wells, where n is the number of wells in the reservoir. We can also utilize this model as a diagnostic tool. For example, if liquid loading or condensate banking or wax/asphaltene deposition starts to happen, the actual pressure and flow rate response will be different than that suggested to the model, hence flagging that the reservoir performance has changed in character.

This study is an extension of my research work[7]. The Python implementation can be found on my Github repository: `https://github.com/titaristanto/LSTM_GRU_RNN_production`

2 Related work

There have been some implementations of AI in the field of oil and gas production. Boomer (1995) used Neural Network to forecast oil production in Vacuum Field, Texas and the model outperformed the professionals 93% of the time. Cao et al. (2016) found that Neural Network performed better than conventional decline curve methods (Arps, Duong, and Stretched Exponential Production Decline).

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

Suhag et al. (2017) implemented Neural Network to predict oil rate in Bakken shale wells and yielded better results than Duong's method.

Previous efforts by Liu (2013) have been successful in applying machine learning method in single-phase production cases. Important handcrafted features were identified and gave satisfactory results in several different cases. However, the computation of handcrafted features takes a long time and can be an issue if the data size is large. Chuan (2017) implemented Recurrent Neural Network (RNN) and nonlinear autoregressive exogenous model (NARX) in single-phase and single-well cases. This study explored the application of Vanilla RNN, Gated Recurrent Unit (GRU), and Long-Short Term Memory (LSTM) layers in our proposed network architecture. Unlike previous study, this project aims to build the sequence model that outputs flow rates of multiple wells. In other words, the proposed network architecture represents all interactions between wells in the entire reservoir.

3 Dataset and Features

The flow rate and pressure dataset were generated using the ECLIPSE reservoir simulator. In this study, as we had three wells producing oil and water from the reservoir, there were three pairs of bottom hole pressure and oil flow rate variables. The total length of the data set was 1800 hours with 1 hour interval. The data were split into training, development, and test sets, accounting for 70%, 15%, and 15% of the total data, respectively (see Fig. 1). Before feeding the input into the model, we performed min-max scaling, such that the range of input and output is from 0 to 1. Min-max scaling has been proven to be useful in various cases to speed up the optimization process.

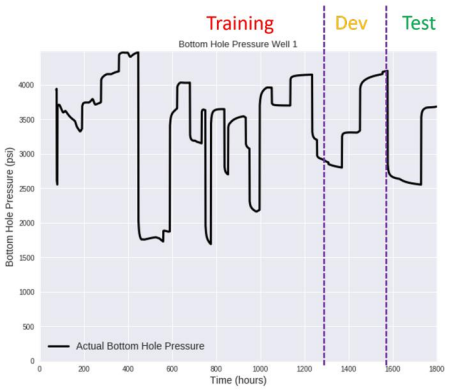

Figure 1: Training, development, and test sets visualization. This only shows pressure and rate in well 1. The data in well 2 and 3 are not shown.

4 Methods

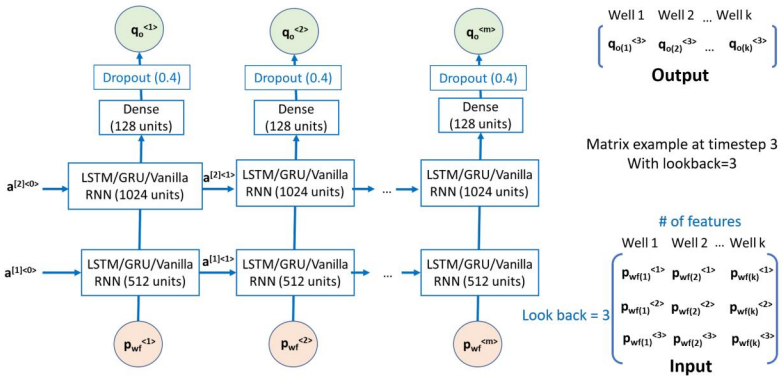

Figure 2: The architecture implemented in this study.

First, we pass feed the three-dimensional matrix of bottom hole pressure input to two recurrent layers with 512 and 2013 units, respectively. First dimension indicates the number of samples, second dimension denotes the 'look back' or time step, and third one is the number of features (which is equal to the number of the wells). The recurrent layer is either Vanilla RNN, GRU, or LSTM layer in this study. Next, the information is passed to a fully-connected layer with 128 units and 'ReLU' activation function. This layer is connected to an output layer with 0.4 dropout rate. The structure as well as input and output matrices can be seen in Fig. 2.

Adam was used as the optimizer and mean squared error loss function was used to evaluate the prediction result (see Eq. 1).

$$MSE = \frac{1}{N}\sum_{n=1}^{N}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 \tag{1}$$

where N is the number of samples, y is the actual flow rate (ground truth), and $\hat{y}$ is the prediction.

Vanilla RNN is the simplest form of recurrent layer. In Vanilla RNN, the output at time i ($\hat{y}^{(i)}$) is a result of tanh activation function and is influenced by the predictor at the same ($x^{(i)}$) time step the as well as the activation output from previous layer. LSTM has more complex structure with four different gates: forget, update, tanh, and output gates. GRU is a simplification of LSTM, where the tanh, forget, and output gates are replaced by update and reset gate (Cho et al. (2014)). The illustration is shown in Fig. 3.

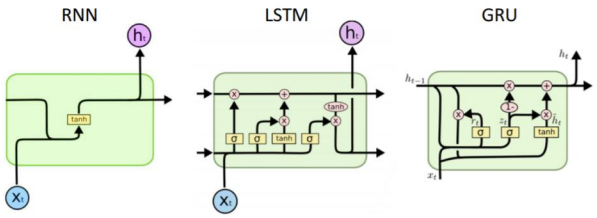

Figure 3: The types of recurrent layers used in this study. Taken from [5].

5 Experiments/Results/Discussion

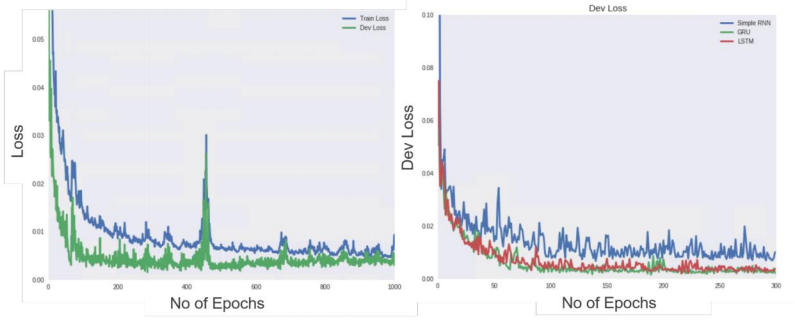

Figure 4: Left: Training and development losses using GRU. Right: Development loss comparison of Vanilla RNN, GRU, and LSTM.

The left plot of Fig. 4 shows that GRU model was able to learn the dataset and generalize well. From the same figure, we can also see that beyond 300 epochs, the development loss started to grow up. Hence, it is reasonable to perform early stopping at 300 epochs to avoid overfitting. On the right figure, Vanilla RNN had the highest dev loss. GRU performed the best and LSTM closely followed behind it.

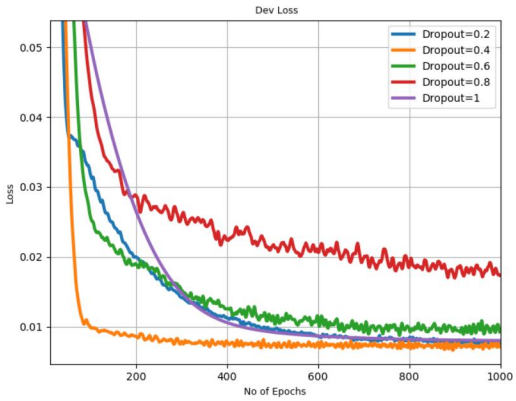

Figure 5: Dev loss curves for different dropout rates.

In order to achieve the best possible model, hyperparameter values must be tuned. We performed dropout sensitivity using GRU to obtain the most optimum drop out value. It turns out that dropout around 0.4 is the best choice, as seen in Fig. 5. Removing some neurons randomly helps the classifier generalize well. However, at some point, removing too much neurons can also be bad as the dev set score starts to flip. We applied the same procedure to other hyperparameters and the optimum results are summarized in Table 5.

Table 1: Optimum hyperparameter setup

Hyperparameters	Values
Learning rate	0.01
Dropout rate	0.4
Number of epochs	300
Batch size	300
Look back	25

With high learning rate, the optimization might not converge. On the other hand, low learning rate is better but requires longer time with shorter optimization step. In this case, 0.01 was the ideal value of learning rate. The optimum look back or time step was 25. Setting too small look back value can lead to some minor fluctuations in the prediction, while too high look back can put more weight on the data points that are more backwards in time and ignore the latest ones.

We then re-run three different recurrent models using the optimum hyperparameter configuration in Table. 5 and the result is shown in Table 5. Vanilla RNN came up as the fastest model but had very high mean squared error (MSE). LSTM yielded much lower MSE but took 4 times as long as Vanilla RNN. GRU had the lowest MSE and run twice as fast as LSTM. It is not surprising that LSTM is slower as LSTM is more complex and has more parameters than GRU. Overall, GRU outperformed Vanilla RNN and LSTM in this setting.

Table 2: MSE of Vanilla RNN, GRU, and LSTM.

Method	Training MSE	Dev MSE	Test MSE	Runtime (s/epoch)
Vanilla RNN	228710.4	56484.4	68027.23	0.48
GRU	1489.8	725.5548	1072.54	1
LSTM	1823.45	1486.16	2295.92	2

The visualization of input and output using GRU model is shown in Fig. 6. We can see that the model captured the trend and variation well and the predicted flow rate had acceptable difference less than 10%. On the other hand, Vanilla RNN failed to capture the extreme variations in pressure and that was the main reason of its high MSE. This study also found that for the model to perform well, training set must contain enough data points and variations in pressure.

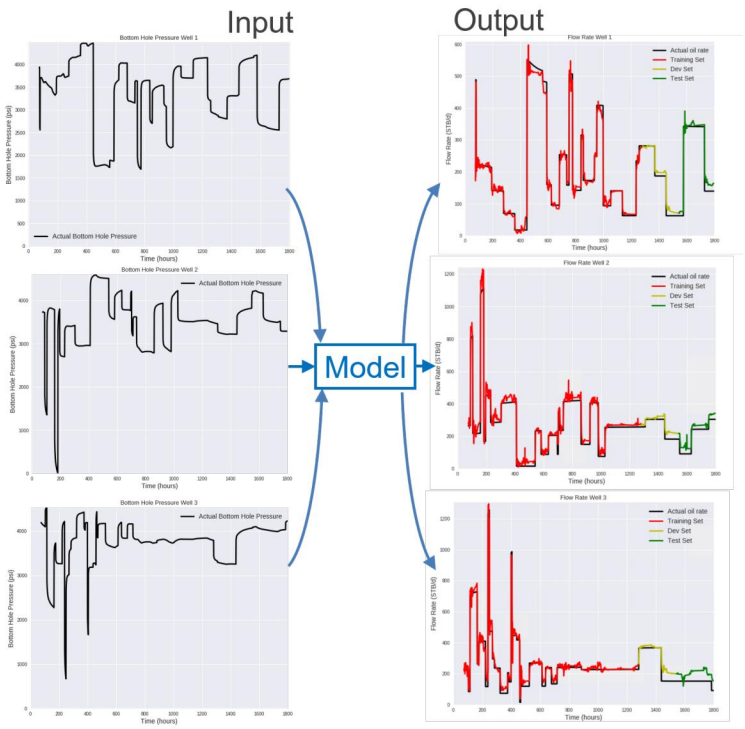

Figure 6: The input and output of the model using GRU.

6 Conclusion/Future Work

We have demonstrated that pressure-flow rate dynamics on the well can be represented with the sequence model in deep learning in this setting. We also implemented a deep learning model that performed many-to-many mapping; that is, given bottom hole pressure values from n wells, we can predict flow rate values on n wells. Architecture search found that the model with GRU layers performed better than the ones with Vanilla RNN and LSTM.

Further development is needed to develop a model that gives smoother prediction. Two possible options to address the issue include denoising or other type of post-processing techniques to the output as well as architecture modification. This research assumed homogeneous porosity and permeability. But in reality, almost no homogeneous permeability or isotropic reservoir exists. Another possible extension of this project is to investigate the application in heterogeneous and more complex reservoirs.

A typical oil and gas reservoir produces oil, water, and gas phases. Building a machine/deep learning model for three-phase problems would be more complicated, as gas phase exhibits different flow behaviors than water and oil at various pressure conditions.

Another potential improvement is needed to develop a machine/deep learning model that is robust with changing water-cut and reservoir pressure over time. One question to answer is whether pressure and flow rate are sufficient to model the complex post-water breakthrough behavior. If the answer is no, it would be interesting to see what other parameters are required to build a good model.

7 Contributions

All the work was done by Tita Ristanto under the guidance of Prof. Roland N. Horne (research adviser) and Cristian Bartolome (CS230 mentor).

References

[1] Liu, Y. & Horne, R.N., (2013) Interpreting Pressure and Flow Rate Data from Permanent Downhole Gauges Using Data Mining Approaches. PhD Thesis, Stanford University.

[2] Tian, C. & Horne, R.N., (2015b) Recurrent Neural Networks for Permanent Downhole Gauge Data Analysis. SPE ATCE.

[3] ECLIPSE Reservoir Simulator. Schlumberger. 2015.

[4] F. Chollet et al. Keras. https://keras.io. 2015.

[5] Olah, C. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] K. Cho, B. van Merienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2014, pp. 1724-1734.

[7] Ristanto, T. & Horne, R.N., (2018). Machine Learning Applied to Multiphase Production Problems. MS Thesis. Stanford University.

[8] Boomer, R.J. (1995). Predicting Production Using a Neural Network (Artificial Intelligence Beats Human Intelligence). Society of Petroleum Engineers (SPE 30202).

[9] Suhag, A., Ranjith, R., and Aminzadeh, F. (2017). Comparison of Shale Oil Production Forecasting using Empirical Methods and Artificial Neural Networks. University of Southern California. SPE ATCE (SPE-187112-MS).

[10] Cao, Q., Banerjee, R., Gupta, S., Li, J., Zhou, W., and Jeyachandra, B. (2016)Data Driven Production Forecasting Using Machine Learning. Schlumberger. SPE Argentina Exploration and Production of Unconventional Resources Symposium, 1-3 June, Buenos Aires, Argentina.