Preprints, 10th IFAC International Symposium on
Advanced Control of Chemical Processes
Shenyang, Liaoning, China, July 25-27, 2018

Data analytics for oil sands subcool prediction – a comparative study of machine learning algorithms

Chaoqun Li, Nabil Magbool Jan, Biao Huang ⋆

⋆ *University of Alberta, Edmonton, T6G 2R3, Canada*

Abstract: Steam Assisted Gravity Drainage (SAGD) is an efficient and widely used technology to extract heavy oil from a reservoir. The accurate prediction of subcool plays a critical role in determining the economic performance of SAGD operations since it influences oil production and operational safety. This work focuses on developing a subcool model based on industrial datasets using deep learning and several other widely-used machine learning methods. Furthermore, this work compares and discusses the out-of-sample performance of different machine learning algorithms using industrial datasets. In addition, we also show that care has to be taken when using machine learning algorithms to solve engineering problems. Data quality and a priori process knowledge play a role in their performance.

Keywords: data analytics, deep learning, process application, machine learning, SAGD, subcool

1. INTRODUCTION

Steam Assisted Gravity Drainage (SAGD) is an efficient, in situ, enhanced oil recovery technique to produce heavy crude oil and bitumen from reservoirs (Butler et al., 1998). The SAGD operation involves a well pair consisting of two wells: an injection well, and a production well. The high-temperature steam, generated from the steam generation system, is injected into the reservoir through the injection well which heats up and reduces the viscosity of the heavy bitumen in the reservoir, and forms the steam chamber underground. The heated bitumen and condensed liquid then flow towards the production well due to gravity. The bitumen collected in the producer is then pumped to the surface for further processing (Butler et al., 1998; Vander Valk et al., 2007).

One of the most important variables in SAGD operations is subcool, which is the temperature difference between steam at the injector and fluid at the producer (Vander Valk et al., 2007). It is a key parameter which reflects the liquid level at the producer and has a significant impact on SAGD reservoir performance (Vander Valk et al., 2007). Yuan et al. (2013) studied the relationship between subcool, wellbore drawdown, fluid productivity and liquid level. Moreover, a model for the SAGD liquid pool above the production well was studied using heat balance and mass balance equations (Gotawala et al., 2012).

Ito et al. (1999) conducted a study on reservoir dynamics and subcool optimization for steam trap control. Gotawala et al. (2009) proposed a subcool control method with smart injection wells. In their study, they divided the SAGD injector into several intervals, and controlled subcool by changing the steam pressure at each interval. In addition, the study on the optimization of subcool in SAGD bitumen process has been carried out (Stone and Bailey, 2014). Furthermore, the Model Predictive Control technique has been used to stabilize subcool temperature and automate well operations in SAGD industry (Patel et al., 2014).

Subcool not only influences reservoir and oil production performance but also has a significant effect on operational safety, since it can reflect the liquid level of the producer. An inappropriate liquid level can result in steam breakthrough thus damaging equipment. Therefore, predicting the subcool value is necessary, and it is beneficial in monitoring, control, and optimization of the process, since the prediction model of the subcool can provide useful information to process and operations engineers. Since subcool is a temperature difference, several factors which have an effect on the temperature at injector and producer will influence the subcool. For example, pump frequency will influence the liquid level trapped at the bottom of the producer, therefore, it has an effect on the temperature at the producer. Also, the heterogeneity of the reservoir properties hinders us from developing a first principle model of subcool. Researchers, therefore, often resort to developing data-driven models in this work.

This paper is organized as follows. First, the general SAGD process, subcool and problem description are introduced. Second, a brief description of deep learning and other selected machine learning methods are introduced. Third, the subcool model development and corresponding hyper parameter settings are discussed. Furthermore, model performances using industrial datasets are analysed, and finally, conclusions are presented.

2. PROBLEM DESCRIPTION

SAGD technology has been used extensively in the oil sands industry in recent decades, and a large volume of historical industrial datasets are available. With the advent of novel machine learning methods and data analytics,

⋆ Corresponding author, e-mail: bhuang@ualberta.ca

these historical process data can be efficiently used to improve the process performance. The stored data contains a wide variety of information such as seismic images of the steam chamber which are of image types and conventional process variables that are stored as floating point variables. Further, the enhancement in the instrumentation of the operations has increased the speed at which the data is stored. In addition, the data includes a lot of inconsistent measurement and missing values, which can be caused by the hardware sensor faults. It also has noisy measurement due to the hardware sensors and varying environment. Data in SAGD process also contains a lot of useful information that can be used efficiently to improve the process operation and increase profitability.

In this study, we aim to solve a problem of estimating an underground state variable, subcool, using some of the manipulated variables as inputs. Figure 1 presents the schematic of SAGD operation. The injected steam plays a significant role in subcool, and the liquids produced are lifted up by the pump. Therefore, the input variables used are those related to injector flowrate, injector pressure, and pump frequency. In this study, we will build a prediction model of subcool as follows:

$$Y = f(X_1, X_2, ..., X_p) \tag{1}$$

where Y denotes subcool at a certain location and X_1, X_2, ..., X_p denote selected input variables. As described, we have only selected manipulated variables as influential features for the subcool prediction. The developed data-driven model is beneficial when underground hardware sensor measurement is unavailable or unreliable, and can be utilized as an alternative subcool measurement.

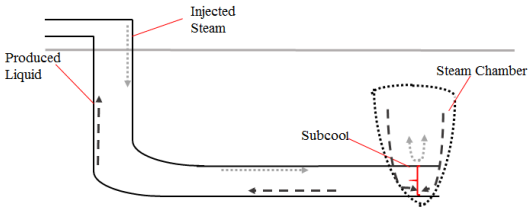

Fig. 1. Schematic of a SAGD process

3. REVISITING MACHINE LEARNING METHODS

Machine learning includes a wide range of algorithms, such as supervised learning, unsupervised learning, reinforcement learning, transfer learning, etc. In order to deal with the problem described in Section 2, we resort to advanced data-driven modelling techniques, such as Deep Learning, ensemble tree-based methods, kernel methods and linear methods. We introduce them briefly in this section.

3.1 Deep Learning

Deep Learning includes a wide range of algorithms, such as Deep Neural Networks, Auto Encoders, Restricted Boltzmann Machines, Deep Belief Networks, etc (Goodfellow et al., 2016). There are many Deep Neural Networks types, such as Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM), which have profound applications in Image Processing and Natural Language Processing, respectively (LeCun et al., 2015). In this study, we consider Deep feedforward neural networks. "Deep feedforward networks, also called feedforward neural networks, or multilayer perceptrons (MLPs), are the quintessential deep learning models" (Goodfellow et al., 2016).

We next introduce some key aspects of deep feedforward network which are crucial to its performance. One of the important factors is weight initialization. There are several methods to do the initialization, via randomly sampling from a uniform or normal distribution over an interval or generating a random orthogonal matrix (Saxe et al., 2013; Bottou, 2012; Glorot and Bengio, 2010).

Another important factor that affects the performance of a deep learning model is the choice of the activation function. Sigmoid and hyperbolic tangent function were the popular choices of activation function in the past, and Rectified Linear Unit (ReLU) has become popular recently (LeCun et al., 2015). The mathematical form of ReLU, which is proved to improve Restricted Boltzmann Machines, can be expressed as follows (Nair and Hinton, 2010):

$$f(x) = max(0, x) \tag{2}$$

Other types of activation function are introduced in (Goodfellow et al., 2016).

Another important component in Deep Learning is the optimization algorithm. The widely used algorithms are mini-batch Stochastic gradient descent and its multiple variants (Bottou, 2010; Duchi et al., 2011). In this work, we will use Adam (Kingma and Ba, 2014). For detailed introduction to deep learning, the reader is referred to the works of (Goodfellow et al., 2016; LeCun et al., 2015).

3.2 Gradient Boosted Decision Trees

Gradient Boosted Decision Trees (GBDT) is one of the most widely used machine learning algorithms today. It iteratively develops an additive regression model sequentially. At each iteration, it assumes that the model is imperfect and constructs a new model to be added to the existing model (Freund et al., 1999; Friedman et al., 2001). Gradient Boosting constructs weak predictors by fitting a gradient at each iteration. This gradient is that of the loss function to be minimized with respect to the model values (Friedman, 2001, 2002).

The gradient tree boosting algorithm is briefly introduced as follows (Friedman et al., 2001): It starts with building a constant regression model. To illustrate this, assume we build M trees, so there are M outer loop iterations. In each iteration, the gradient of the loss function with respect to the function values at each training data point is calculated, and a regression tree is fitted between training data points and residuals. Then, the current predictor is updated. After M iterations, the final predictor is constructed.

3.3 Random Forest

Random Forest was proposed by Breiman (Breiman, 2001). As in Bagging, it trains each tree on a sampled bootstrap dataset from the training data. However, it considers a random subset of the variables to split an internal

node, rather than all the variables. Each tree of Random Forest is a decision tree, and for a regression problem, the average value is applied as the prediction.

The training process of Random Forest can be summarized as follows (Breiman, 2001):

1. Draw M bootstrap datasets from the training dataset.

2. Grow a decision tree for each bootstrap dataset. A randomly selected q features are considered to split an internal node. The tree is grown until the maximum depth.

3. Aggregate M predictions of the new data point to compute the final single prediction. The majority vote and the average value are applied for classification and regression as final prediction, respectively.

Note that some important properties of Random Forest, such as, Out-of-Bag (OOB) error, Variable Importance, and Intrinsic Proximity Measure (Breiman, 2001) are not discussed here in the interest of brevity.

3.4 Support Vector Regression

Cortes and Vapnik at AT&T Labs proposed Support Vector Network as a learning machine (Cortes and Vapnik, 1995). The main idea here is to do feature expansion: First, inputs are mapped to a high dimensional space non-linearly; then, a linear surface is built in the new space. The solution of SVM is sparse, and only a subset of training data points are utilized to form the solution (Friedman et al., 2001).

In order to map the inputs from the original space to a high dimensional space, a kernel function is used. Also, Support Vector Regression applies an ϵ-insensitive loss function, which is proposed by Vapnik, and the optimization algorithm aims at minimizing the error bound, rather than the observed training errors (Cortes and Vapnik, 1995; Friedman et al., 2001; Smola and Schölkopf, 2004). Equation 3 shows its form (Cortes and Vapnik, 1995):

$$L_\epsilon(y, f(x)) = \begin{cases} 0 & \text{if } |y - f(x)| \leq \epsilon \\ |y - f(x)| & otherwise \end{cases}. \quad (3)$$

3.5 Linear Regression and Ridge Regression

Linear Regression is the simplest regression model. Ridge Regression is a variant of linear regression, considering shrinkage of the coefficients (Friedman et al., 2001). By adding an l_2 norm penalty to the loss function of linear regression, we obtain the loss function of Ridge Regression.

4. MODEL DEVELOPMENT

In this section, we focus on developing data-driven models using deep learning and other machine learning methods. Description of dataset, data pre-processing, hyperparameter exploration, settings and software are discussed.

4.1 Data Description

In this study, we use an industrial dataset to develop and investigate the performance of different predictive models for subcool prediction. The dataset contains 5 inputs variables, which we use as influential input variables, to predict an output variable. See Table 1.

Table 1. Process variables description

Variables	Description	Units
Output	Reservoir Subcool	°C
Input 1	Injector tubing pressure	kPa
Input 2	Injector casing pressure	kPa
Input 3	Injector tubing flowrate	sm^3/hr
Input 4	Injector casing flowrate	sm^3/hr
Input 5	Pump frequency	Hz

The dataset contains 25,000 samples of measured data after data cleaning as described in Section 4.2. These measurements are taken over a time period of nine months. The data is divided chronologically for training and testing purposes. The first 20,000 are considered as training dataset and for tuning hyperparameters. The next 5000 samples are used for testing the out-of-sample performance of the developed predictive models. The sampling time is 10 mins and the sample value is the average value of this interval. It is considered to have achieved the steady state, thus, we build static models.

4.2 Data Pre-processing

The raw industrial dataset collected from the historical database might include inconsistency, missing values, and outliers. Therefore, they cannot be used directly in the model development. Hence, data pre-processing is performed prior to model development. Data cleaning involves removing inconsistent measurement, missing values, and outliers. Then, we normalize the data for scaling issues, and also, for the convergence and better performance of machine learning algorithms.

4.3 Hyperparameter exploration

The goal of this subsection is to explore hyperparameter settings for each of the algorithms under investigation. For this purpose, only the first 20,000 data points are used. Test data are not used in this subsection. The first 20,000 data points are divided chronologically into training and validation datasets. In this study, we evaluate the following statistics: Mean Absolute Error (MAE), Mean Square Error (MSE), Pearson correlation coefficient, and the trend of plots to compare the performance of different hyperparameter settings of each model on the validation set. We applied Grid Search, Random Search, and Greedy Search as hyperparameter search strategies. Once the hyperparameter settings are determined, we train the model again on all of the 20,000 data points to obtain the final model. This model will be used later to analyze the performance on test data.

The model developments and evaluations are performed in Python 2.7, in Mac OS X 10.11.5 environment. Deep Learning model is developed via Keras 2.0.6 (Chollet et al., 2015), using TensorFlow 1.0.0 as its backend, which is developed by Google (Abadi et al., 2016). Other machine learning methods investigated in this study, such as Random Forest, Gradient Boosted Decision Trees, Support Vector Regression, Ridge Regression and Multiple Linear

Regression are performed via scikit-learn version 0.18.1 (Pedregosa et al., 2011; Buitinck et al., 2013). Next, we present the hyperparameter settings of different models.

Deep Learning As mentioned previously, we choose Deep Feed Forward Neural Networks in Deep Learning. One of the core ideas of deep learning is to have deeper architectures (Bengio et al., 2009). Therefore, we tried multiple layers on training dataset, and finally selected 4 hidden layers. Dropout is a way to avoid overfitting (Srivastava et al., 2014). Our hidden layer is not wide, and we do not apply dropout in this case. We apply l_2 norm for regularization and avoid overfitting. An epoch means a complete pass through the whole training dataset while training the model (Goodfellow et al., 2016; Chollet et al., 2015). The network is fully connected.

Gradient Boosted Decision Trees Learning rate shrinks the contribution of each tree when a tree is added to the model, and could be considered as a weighting factor of the additive sequentially learned models (Friedman, 2001; Friedman et al., 2001). Each decision tree has a maximum tree depth. The deeper a tree grows, the more complex a model becomes.

Random Forest Minimum number of samples required to be at a leaf node controls the depth of a regression tree. While a tree is being trained, a randomly selected number of features are considered to split an internal node. The maximum number to consider equals to the square root of the total number of features in this setting.

Support Vector Regression We choose Radial Basis Function as the kernel in this case after testing different kernel choices. Penalty parameter controls the trade-off between bias and variance.

Ridge Regression Regularization parameter controls the bias and variance trade-off.

Linear Regression There is no hyperparameter in Linear Regression.

The hyperparameter settings of different models under consideration are presented in Table 2.

Table 2. Hyperparameter settings

Hyperparameter	Value
Deep Learning	
Number of hidden layers	4
Number of neurons in each hidden layers	32, 128, 256, 128
Hidden activation function	ReLU
Output layer	1 linear neuron
ℓ_2 regularization parameter	1e-4
Optimizer	Adam learning rate: 0.001
Number of epochs	300
GBDT	
Number of regression trees	250
Learning rate	0.01
Maximum depth of regression tree	5
Random Forest	
Number of regression trees	500
Maximum features to consider in each split	Square root of the number of features
Minimum number of samples required to be at a leaf node	0.01× number of samples
Support Vector Regression	
Kernel function	Radial Basis Function
Penalty parameter for the error term in loss function	0.5
Kernel Parameter	10
ϵ	0.1
Ridge Regression	
Regularization strength parameter	2.5

Table 3. Test results

Methods	Test MAE	Test MSE	Correlation Coefficient
Deep Learning	0.28615	0.16440	0.78320
GBDT	0.22425	0.10475	0.80857
Random Forest	0.23073	0.11927	0.77248
SVR	0.27219	0.17925	0.65551
Ridge Regression	0.53630	0.38927	0.26273
Linear Regression	0.53633	0.38931	0.26270

5. RESULTS AND DISCUSSIONS

In this section, we report prediction performance on test data. Only normalized data results are shown for data proprietary reasons.

5.1 Performance on test data

In this subsection, we show the predictive performance of each model (see Figure 2 and Figure 3), using hyperparameter settings discussed in Section 4.3. Linear Regression shows the same trend plot as Ridge Regression. We only show the plot of Ridge Regression. The performance statistics of different models such as MAE, MSE, Pearson correlation coefficient are presented in Table 3.

While developing deep feedforward neural network model in Keras with TensorFlow as its backend, we should note that the issue of randomness may lead to non-reproducible results, which comes from the random initialization weights, shuffling data, mini batch in optimization, and implementation of tools for parallel computing (Chollet et al., 2015; Abadi et al., 2016). We train the model with the same settings and data 10 times, separately. Each of the 10 models will produce a prediction. For brevity, we report the results with the best MAE of the 10 deep models only.

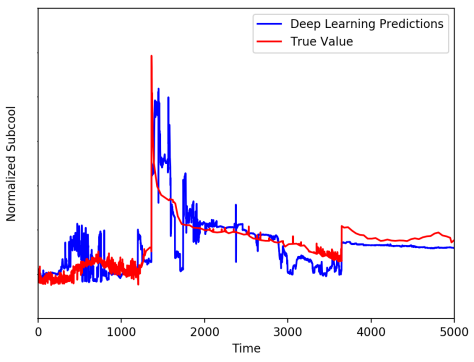

Fig. 2. Test results of Deep Learning/Deep Feedforward Neural Networks

5.2 Discussions

First, it is obvious that the non-linear methods have better performance compared to linear methods due to inherent nonlinearity of the dataset. Therefore, the model is underestimated with linear methods, and a more complex model is required.

Second, it can be observed that SVR outperforms linear methods. The kernel trick in SVR doing feature expansion not only deals with data nonlinearity, but also implies that there are some hidden features in the process from the 5 input features to subcool because of its better performance than linear methods. Also, we see a relatively smoother plot of SVR predictions in Figure 3. It could be explained by the threshold epsilon, which makes the prediction less sensitive to smaller noises.

Comparison between the Random Forest and GBDT indicates that GBDT performs better in terms of MAE, MSE and Pearson correlation coefficient, but both methods perform well. As introduced earlier, both methods apply multiple models to estimate the single final prediction. This is one reason why they both perform well. Moreover, they are both based on Regression Tree. Prediction is performed through comparison between the value of the corresponding feature and the internal nodes. As a result, the qualitative relationships between subcool and selected features are influential in this case. In addition, using the samples of the leaf node for prediction can decrease the effect of noise which is due to environment, sensors, etc.

Deep learning method does not show the best results in terms of MAE, MSE and Pearson correlation coefficient. However, the trend plot of the Deep Learning model can capture the peak in the trend very well while all other models more or less failed in capturing the peak. This is because of the flexibility of deep learning model and powerful optimization algorithm. First, we apply multiple layers and the number of neurons in each hidden layer can be changed. Therefore, the automatic latent feature expansion and reduction within multiple hidden layers imply the meaningful results of deep feedforward neural network. Second, the ReLU activation function does not suffer from saturation problem, which avoids gradient decays (Nair and Hinton, 2010). Also, a good model needs a powerful optimization algorithm to train parameters.

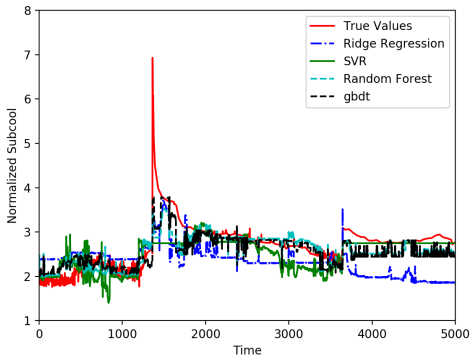

Fig. 3. Test results of GBDT, Random Forest, SVR and Ridge Regression

The Adam, a stochastic optimization algorithm, has the advantages of bias correction and individual adaptive learning rate, which makes it useful in solving non-convex optimization problem (Kingma and Ba, 2014). Hence, the flexible architecture, unsaturated activation function and powerful optimization algorithm explains why deep leaning can capture the peak.

5.3 Care Taken in Applying Machine Learning Tools

In this section, we show that care has to be taken when using machine learning tools to solve complex engineering problems. To this end, another set of data, containing 37,100 data points after data cleaning, is considered. Data normalization and model hyper parameter settings are explored again and poor performance has been reported on the new test data.

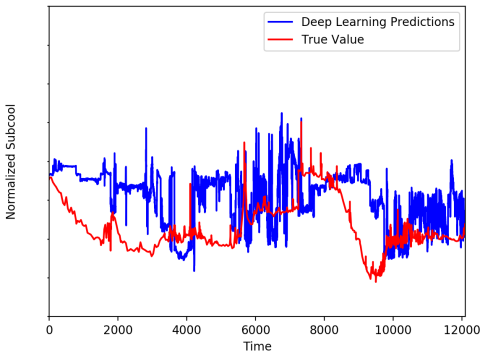

Fig. 4. Test results of Deep Learning/Deep Feedforward Neural Networks using new dataset

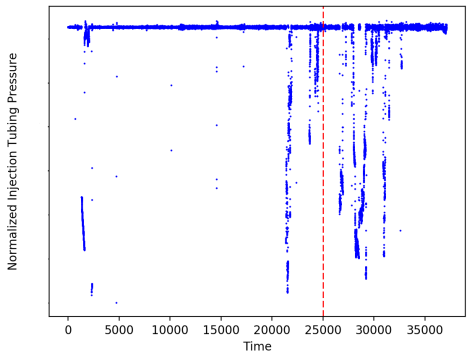

Fig. 5. Trend of Injection Tubing Pressure

Only the trend plot of Deep Feedforward Neural Networks is listed since it has shown the best performance, and other methods do not perform better than this in terms of any metrics on this new dataset. Pearsons correlation coefficient of a deep feedforward neural network in this new data set is 0.2169 whereas the test MAE and MSE were found to be 0.474 and 0.316, respectively.

We have investigated the original data to see what happened in the period of this set of data. We found there were obvious operation condition changes between the range of training data and testing data. Because of closed-loop control and cascade control, the operational condition changes

can be reflected by several variables. See Figure 5 for Injection Tubing Pressure, where training data period and testing data period is split by red dashed line. From this figure, one can see that much more significant downtrend oscillations of the injection tubing pressure occurred in the testing data period than the training data period, which has invalidated the model learned from the training data. Therefore, the data quality in engineering applications plays a critical role in the process data analytics and deep learning. Blind use of the machine learning tools can introduce problems. Meanwhile priori knowledge of the industrial process can also help in dealing with this type of problems.

6. CONCLUSIONS

Machine learning can make use of complex industrial data for building data driven models. The potential advantages of various machine learning methods under consideration have been discussed in this study. The Deep Feedforward Neural Network has shown good predictive performance in capturing process trends. Also, the performance of ensemble decision tree based regression models are comparable. Further, the model development task highlights the necessity of assessing data quality prior to model building.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, 265–283.

Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1), 1–127.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, 177–186. Springer.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, 421–436. Springer.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5–32.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., et al. (2013). Api design for machine learning software: experiences from the scikit-learn project. *arXiv preprint arXiv:1309.0238*.

Butler, R. et al. (1998). Sagd comes of age! *Journal of Canadian Petroleum Technology*, 37(07).

Chollet, F. et al. (2015). Keras. `https://github.com/keras-team/keras`.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3), 273–297.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul), 2121–2159.

Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780), 1612.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Friedman, J.H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4), 367–378.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Gotawala, D.R., Gates, I.D., et al. (2009). Sagd subcool control with smart injection wells. In *EUROPEC/EAGE Conference and Exhibition*. Society of Petroleum Engineers.

Gotawala, D.R., Gates, I.D., et al. (2012). A basis for automated control of steam trap subcool in sagd. *SPE Journal*, 17(03), 680–686.

Ito, Y., Suzuki, S., et al. (1999). Numerical simulation of the sagd process in the hangingstone oil sands reservoir. *Journal of Canadian Petroleum Technology*, 38(09).

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436.

Nair, V. and Hinton, G.E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Patel, K., Aske, E.M., Fredriksen, M., et al. (2014). Use of model-predictive control for automating sagd well-pair operations: A simulation study. *SPE Production & Operations*, 29(02), 105–113.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct), 2825–2830.

Saxe, A.M., McClelland, J.L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Smola, A.J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3), 199–222.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.

Stone, T. and Bailey, W. (2014). Optimization of subcool in sagd bitumen processes. In *Prepared for presentation at the 2014 World Heavy Oil Congress, New Orleans*, 5–7.

Vander Valk, P., Yang, P., et al. (2007). Investigation of key parameters in sagd wellbore design and operation. *Journal of Canadian Petroleum Technology*, 46(06), 49–56.

Yuan, J.Y., Nugent, D., et al. (2013). Subcool, fluid productivity, and liquid level above a sagd producer. *Journal of Canadian Petroleum Technology*, 52(05), 360–367.