Lawrence Berkeley National Laboratory
LBL Publications

Title

Neural Networks for Prediction of Complex Chemistry in Water Treatment Process Optimization

Permalink

https://escholarship.org/uc/item/9wm7g6m8

Authors

Dudchenko, Alexander V
Amusat, Oluwamayowa O

Publication Date

2024-07-09

DOI

10.69997/sct.107047

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-ShareAlike License, available at https://creativecommons.org/licenses/by-sa/4.0/

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Systems & Control Transactions

Research Article
Foundations of Computer Aided Process Design (FOCAPD 2024)
Breckenridge, Colorado, USA. July 14-18, 2024
Peer Reviewed Conference Proceeding

PSE PRESS

Neural Networks for Prediction of Complex Chemistry in Water Treatment Process Optimization

Alexander V. Dudchenko[a,*], Oluwamayowa O. Amusat[b]

[a] SLAC National Accelerator Laboratory, 2575 Sand Hill Rd, Menlo Park, CA 94025, USA
[b] Lawrence Berkeley National Laboratory (LBNL), 1 Cyclotron Rd, Berkeley, CA 94720, USA
* Corresponding Author: avd@slac.stanford.edu

ABSTRACT

Water chemistry plays a critical role in the design and operation of water treatment processes. Detailed chemistry modeling tools use a combination of advanced thermodynamic models and extensive databases to predict phase equilibria and reaction phenomena. The complexity and formulation of these models preclude their direct integration in equation-oriented modeling platforms, making it difficult to use their capabilities for rigorous water treatment process optimization. Neural networks (NN) can provide a pathway for integrating the predictive capability of chemistry software into equation-oriented models and enable optimization of complex water treatment processes across a broad range of conditions and process designs. Herein, we assess how NN architecture and training data impact their accuracy and use in equation-oriented water treatment models. We generate training data using PhreeqC software and determine how data generation and sample size impact the accuracy of trained NNs. The effect of NN architecture on optimization is evaluated by optimizing hypothetical black-box desalination processes using a range of feed compositions from USGS brackish water data set, tracking the number of successful optimizations, and testing the impact of initial guess on the final solution. Our results clearly demonstrate that data generation and architecture impact NN accuracy and viability for use in equation-oriented optimization problems.

Keywords: Machine Learning, Water, Technoeconomic Analysis, Pyomo, Wastewater

INTRODUCTION

Water chemistry plays a critical role in the design and operation of water treatment processes. Detailed chemistry modeling tools use a combination of advanced thermodynamic models and extensive databases to accurately predict phase equilibrium and reaction processes, such as those done by open-source PhreeqC software [1]. The complexity, formulation, and extensive database of these models preclude their direct integration in equation-oriented modeling platforms, making it difficult to use their capabilities for rigorous water treatment process optimization.

Neural networks (NN) can provide a pathway for integrating the predictive capability of chemistry software into equation-oriented models and enable optimization of complex water treatment processes across a broad range of conditions and process designs. A key challenge in developing a broadly applicable surrogate model for aqueous chemistry is the high non-linearity of the phenomena and high problem dimensionality. NNs have the potential to learn chemical phenomena and provide accurate estimates over a broad range of ion and reactant compositions [2].

NNs developed for water treatment optimization must accurately predict three critical chemistry phenomena: scaling tendencies, precipitation fractions, and changes in pH.

Modeling scaling tendency in desalination processes is critical for predicting solid formation, which occurs as ion concentration increases and results in process failure [3]. Typically, scaling is mitigated by reducing water recovery of desalination process, adding acids, which reduce scaling potential, and adding antiscalants that increase maximum allowable scaling tendency [4].

Modeling chemically driven precipitation is important for estimating the performance of pretreatment and softening processes, such as lime (CaO) and soda-ash (Na_2CO_3) softening [5]. These processes are commonly used to drive the formation of solids that remove divalent ions and reduce water hardness. Removing divalent ions minimizes the potential for scale formation in downstream processes.

Accounting for pH change due to chemical addition,

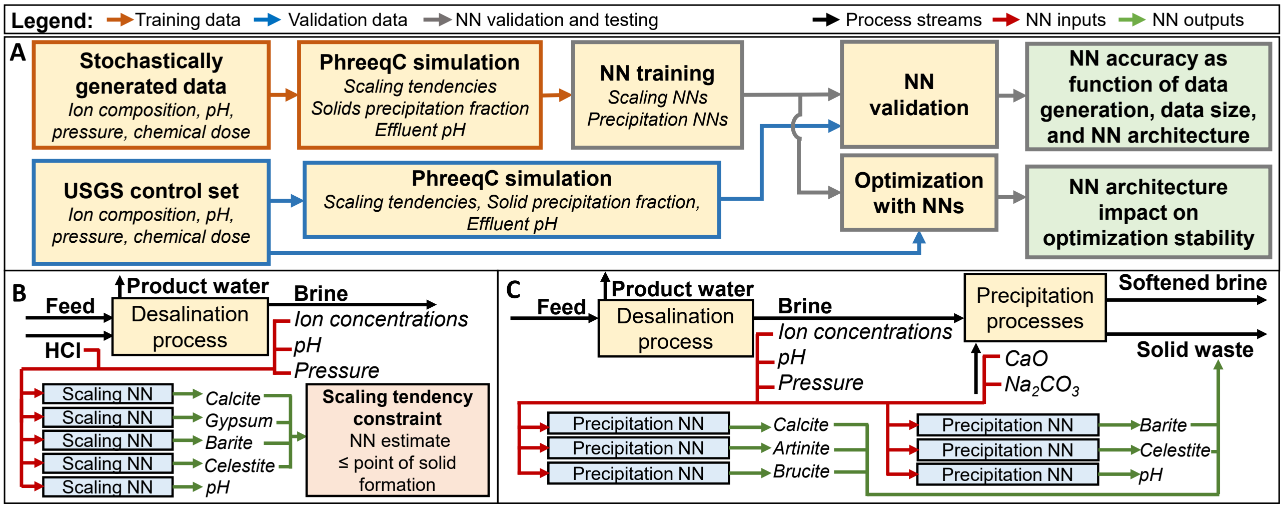
Legend: Training data Validation data NN validation and testing Process streams NN inputs NN outputs
A
Stochastically generated data
Ion composition, pH, pressure, chemical dose
PhreeqC simulation
Scaling tendencies
Solids precipitation fraction
Effluent pH
NN training
Scaling NNs
Precipitation NNs
NN validation
NN accuracy as function of data generation, data size, and NN architecture
USGS control set
Ion composition, pH, pressure, chemical dose
PhreeqC simulation
Scaling tendencies, Solid precipitation fraction, Effluent pH
Optimization with NNs
NN architecture impact on optimization stability
B
Feed
Product water
Desalination process
Brine
Ion concentrations
pH
Pressure
HCl
Scaling NN
Calcite
Gypsum
Barite
Celestite
pH
Scaling tendency constraint
NN estimate ≤ point of solid formation
C
Feed
Product water
Desalination process
Brine
Ion concentrations
pH
Pressure
Precipitation processes
Softened brine
Solid waste
CaO
Na_2CO_3
Precipitation NN
Calcite
Artinite
Brucite
Barite
Celestite
pH

Figure 1: (A) Workflow schema for generation and testing NNs. (B) Schematic for scaling tendency NN use in desalination processes, where estimated scaling tendency must be less than point where solids form (can be 1 if no anti-scalants are used and exceed 80 with anti-scalants use for certain solids). (C) Schema for precipitation NN use in precipitation optimization problem, where precipitation fractions estimate amount of solids that form.

such as HCl and lime, is critical for modeling full treatment trains. The addition of HCl and lime will change the effluent stream pH and change requirements for chemical addition in downstream processes.

Herein, we investigate how the training data and NN architecture affect NN performance for the three chemistry phenomena (Figure 1). We train NNs using stochastically generated data, and validate NN accuracy and stability in equation-oriented optimization models using USGS data set [6] (Figure 1A). Stochastic data generation is used to simulate a broad range of chemistries and chemical additions, which are used as inputs for PhreeqC to predict chemical phenomena of interest. We investigate the impact of training data by considering the effect of data skewing to sample lower ion concentration distributions preferentially, which better captures real water compositions represented by the USGS brackish water data set. Finally, the developed NNs are integrated into equation-oriented models, and their stability is evaluated by tracking the number of successful solutions and the sensitivity to the initial guess.

METHODS

Data generation

We use PhreeqC with the included Pitzer database to estimate scaling tendencies, precipitation fractions, and effluent pH based on feed composition and chemical addition (Figure 1A). Feed is charge neutralized by adjusting Cl ion concentration, and charge neutral composition is used as input for NN.

For NN training, we stochastically generated ion compositions and chemical addition amounts (Table 1). We store PhreeqC result for feed composition without and with chemical addition (Table 2). The precipitation fraction is the fraction of primary ion in the tracked phase that precipitates from the solution (e.g. precipitate fraction for $CaCO_3$ is the ratio of precipitated Ca to total Ca in solution (feed and added reactant) before reaction.

The feed compositions, pressure, pH, and amount of chemical added is sampled using the standard Latin hypercube (LHS) method [7]. The LHS samples are scaled 0-1 and are rescaled to ion concentration, pressure, and chemical addition using the exponential skew function (eq. 1) and for pH using the log skew function (eq. 2). In these equations, x is the LHS sample, S is the skew magnitude, R_{high} is the high range of absolute value, and R_{low} is the low range for absolute value.

$$f_{exp-skew}(x) = (S^x - 1) * \frac{R_{high}-R_{low}}{S-1} + R_{low} \quad (1)$$

$$f_{log-skew}(x) = \log_{10}(S * x + 1) * \frac{R_{high}-R_{low}}{\log_{10} S+1} + R_{low} \quad (2)$$

The exponential skew function rescales the input value to be exponentially lower, such that an LHS sample of 0.1 (range 0-1) would be rescaled to 0.028, 0.006, 0.001, 0.0002 for skew magnitudes of 10, 100, 1,000, and 10,000, respectively. This skewing increases the sampling of low ion concentration distribution typically observed in real waters. The log skew function has an inverse relationship; for an LHS sample of 0.1 (range 0-1) the rescaled samples would be 0.15, 0.34, 0.5, and 0.6 for skew magnitudes of 10, 100, 1,000, and 10,000, respectively. The log skew function is used to increase the number of samples with basic pH (higher pH), where pH sensitive scalants have high scaling tendencies and are filtered out during data generation (described below).

We generate 2 million total samples using PhreeqC for NN training using a range of skew magnitudes only applied to ion composition, pressure, and chemical addition, while pH skew magnitude is maintained at 10. The data is generated iteratively by taking 10,000 LHS samples at every step, applying the appropriate skew functions to the LHS samples, and keeping samples with total solids concentrations below 360 g per kg of water (g/kgw) and scaling tendency below 100. The procedure is repeated until 2 million samples are generated.

We further use the USGS brackish water data set to test NNs for real water conditions. The data set was filtered to isolate samples that exist in water-stressed regions using the method described by Ahban et al.[8]. This data set was filtered to remove water composition with high silica scaling potential by removing samples with Si concentrations above 50 ppm and H_4SiO_2 concentration above >2 mM at 90% water recovery and pH of 6.5 [9]. This data is referred to as the USGS control set and contains 782 feed compositions.

Table 1: Feed composition and inputs in the stochastic data set. Temperature is fixed at 20°C.

Feed composition in stochastic data

Input	*Low range*	*High range*	*Unit*
Na	0	136	g/kgw
Cl	0	180	
Ca	0	10	
Mg	0	10	
HCO_3	0	10	
SO_4	0	100	
K	0	40	
Sr	0	10	
Ba	0	0.1	
pH	4	12	pH

Inputs for scaling tendency prediction

Input	*Low range*	*High range*	*Unit*
Pressure	1	401	atm
HCl	0	2000	PPM

Inputs for precipitation fraction prediction

Input	*Low range*	*High range*	*Unit*
CaO	0	2000	PPM
Na_2CO_3	0	2000	

The USGS control data set is used to generate test data by imitating a black box desalination process within PhreeqC. The test data set for scaling tendencies includes process operating with water recoveries ranging from 0 to 90% in 10% steps and with HCl addition of 10, 50, 100, 1,000, and 1,500 ppm (N=38,357). The water recoveries for the precipitation fraction test data set were 0, 20, 60, and 90%, and CaO and Na_2CO_3 addition was 10, 100, and 1,000 PPM (N=27,460). All data sets excluded samples that resulted in scaling tendencies above 100.

Neural network training and validation

NNs were built and trained using Pytorch 2.0.0 on NVIDIA GPUs using standard CUDA implementation [10]; only stochastically generated data was used for training, with 10,000 samples set aside for testing. Throughout the paper, we only present results that use the USGS control set to quantify errors in NNs.

We build dense networks using 3 and 5 deep layers, with 30, 60, and 90 neurons with either sigmoid or tanh activation function applied to all layers except the output layer. All NNs use the ion composition and feed pH as inputs. Additionally, the scaling tendency NNs include HCl addition and pressure as inputs, while the precipitation NNs include CaO and Na_2CO_3 as inputs. Each NN predicts only a single output, resulting in five scaling NNs and six precipitation NNs per architecture type. Additionally, each network is trained using two different weight decays of 0.001 and 10^{-6}. Thus, a total of 24 NN architectures are evaluated (2 deep layers x 3 neuron types x 2 activation function x 2 weight decays).

Table 2: Output scaling tendencies ranges and precipitation fractions.

Scaling tendency data set outputs

Output	*Low range*	*High range*
Calcite	0	98.19
Gypsum	0	14.61
Barite	0	99.03
Celestite	0	97.7
pH	0.56	11.96

Precipitation fraction data set outputs

Output	*Low range*	*High range*	*Primary ion*
Calcite	0	1	Ca
Artinite	0	0.96	Mg
Brucite	0	1	Mg
Barite	0	1	Ba
Celestite	0	0.99	Sr
pH	4	12.8	N/A

We use Pytorch implementation of AdamW optimizer using cosine annealing with warm restart to train NNs [11]. We train with five cycles, that switch based on the number of gradient updates, performing ~2M gradient updates in total. The learning rates for sigmoid and tanh activation functions were 0.01 and 0.001, respectively, with the final learning rate of 10^{-6}. We use minibatches with a size of 4096 samples. At each epoch, the data is shuffled and sampled without replacement. Finally, we linearly scale all input and output data for training between 0 and 1.

To test the effect of skew magnitude and data size on NN accuracy, we use NN with 5 layers and 60 neurons (5x60) with a sigmoid activation function and trained with a weight decay of 10^{-6} on Calcite scaling tendency, pH after HCl addition, and Calcite precipitation fraction.

We validate NN accuracy by comparing their predictions against the USGS control data sets and presenting errors in all figures as statistical distributions, showing 5th, 25th, median, 75th, and 95th percentile errors. The error for scaling tendencies and pH is the percent difference between NN prediction and ground truth (*GT*) generated using PhreeqC, as shown in equation 3.

$$\text{error} = \frac{\text{NN}_{\text{prediciton}} - \text{GT}}{\text{GT}} * 100\% \qquad (3)$$

We exclude any errors for scaling tendencies below 0.5, as in water treatment no scaling occurs below scaling tendencies of 1. For these samples, the absolute errors remain below 0.2 but can significantly shift error distribution (e.g. for a scaling tendency of 0.1, the prediction could be 0.3, resulting in 200% error but having no implication for optimization of the water treatment process). For precipitation fractions, we present the absolute difference between NN and ground truth in percent, as shown in equation 4.

$$error = NN_{prediciton} - ground_{truth} * 100\% \qquad (4)$$

Neural network integration into waterTAP

The two desalination processes we consider (Figure 1B and 1C) are modeled in WaterTAP, a framework for technoeconomic assessment of water treatment systems. We integrate developed NNs into WaterTAP using the Optimization and Machine Learning Toolkit (OMLT) [12, 13]. We use the reduced smooth formulation in OMLT, which loads the NN as a single large expression into the model. Additionally, we add constraints that convert absolute values to scaled NN inputs and outputs.

We formulate two optimization problems that emulate black box desalination processes with scaling and precipitation NNs (Figure 1B and 1C). The desalination process is emulated using a single feed block that specifies the feed mass flow of ions and water. The ion mass flow is fixed to ion concentrations as specified by a sample from USGS data set. The mass flow of water is unfixed during optimization, imitating a desalination process that removes pure water from the feed block and increases ion concentration in the remaining brine. The difference between the initial and the optimized flow mass of water is equal to the amount of product water, while the optimized water flow mass is equal to mass flow of waste brine.

The scaling tendency problem, as shown in Figure 1B, is where scaling NNs are added to the desalination problem to predict scaling tendencies due to increased ion concentration in feed block, with HCl added as a scaling control mechanism. The scaling tendencies are constrained to remain below 60 for Calcite, 2.3 for Gypsum, 60 for Barite, and 8 for Celestite, imitating a desalination process operating with anti-scalants [3].

The precipitation problem, as shown in Figure 1C, is where precipitation NNs are added to the desalination problem to predict the removal of solids that form due to increased concentration of ions and chemical addition [5]. Here an additional constraint is added to calculate the final brine hardness after solid formation, which is constrained to be below or equal to 50 ppm of $CaCO_3$ as shown in equation 5, where $Ca_{f\text{-}total}$ is mole flow rate of Ca in feed and Ca added from CaO addition, Mg_f is mole flow rate of Mg in feed, and $rm_{Calcite}$, $rm_{artinite}$, and $rm_{brucite}$ are removal fraction for Calcite, Artinite, and Brucite respectively.

$$Total - hardness = \left(\left|Ca_{f-total} - Ca_{f-total} * rm_{Calcite}\right| * 100.1 + \left|Mg_f - Mg_f * (rm_{Artenite} + rm_{Brucite})\right| * 100.1\right) * 1000 \qquad (5)$$

The optimization problems are formulated to maximize the value of produced water from black box desalination processes. In the optimization, the product water can be sold with a value of 0.5 units/kg, while the remaining waste brine is penalized with a cost of 10 units/kg, and the addition of chemicals (HCl, CaO, and Na_2CO_3) is penalized with the value of 0.19 units/kg. For the scaling tendency problem, there are 2 degrees of freedom: the amount of water to remove and the amount of HCl to add. For the removal fraction problem, there are 3 degrees of freedom: amount of water to remove, amount of CaO to add, and amount of Na_2CO_3 to add.

In these optimization problems, removing water increases ion concentration, which increases scaling tendencies and hardness. The scaling tendencies can be decreased for some scalants with HCl addition. Hardness can be reduced by adding CaO and Na_2CO_3, which increase precipitation fractions of Ca and Mg containing solids. Thus, the optimization balances the cost of producing water, disposing of brine, and satisfying scaling and hardness constraints through chemical addition.

We optimize the two problems by randomly drawing 500 feed compositions from the USGS control set, and providing two initial guesses. We provide one guess that is poorly posed where 90% water is removed and 1 PPM of chemicals is added, resulting in a feed composition that is likely violating the scaling tendencies and hardness constraints. The second guess is well-posed, where feed is diluted by 5 times and 1,000 PPM of chemicals are added, resulting in a feed composition that is unlikely to violate scaling tendencies and hardness constraints. The total solvability is evaluated by tracking the percent of total solved samples and guesses (N=1000). The NN propensity to solve to a local minimum is quantified by tracking the number of solutions with different objective values caused by different initial guesses for the same feed composition and where one guess failed to solve. The

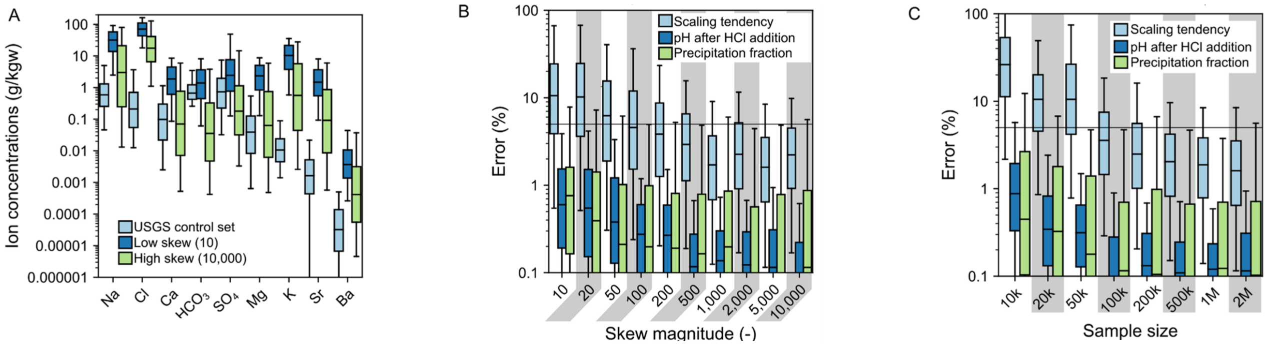

Figure 2: Effect of data generation on NN accuracy. (A) Effect of skew magnitude on ion sampling, (B) Effect of skew magnitude on NN accuracy, (C) Effect of sample size on NN accuracy. Error is shown for Calcite scaling tendency and precipitation fraction.

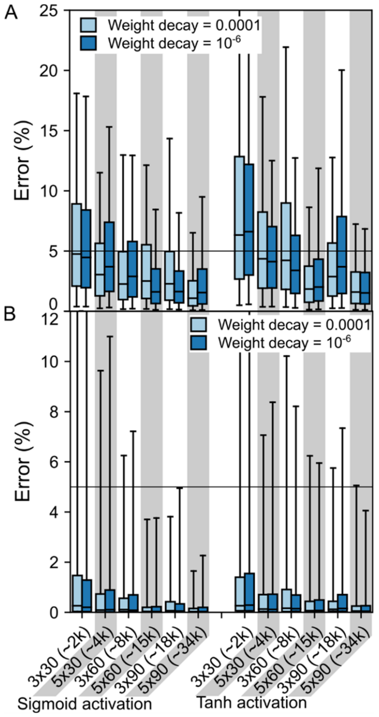

Figure 3: Effect of network architecture on NN error in prediction of (A) scaling tendencies and (B) removal fractions of Calcite. The network size is labeled as depth x width with values in parenthesis showing a rounded number of trainable parameters (e.g. 3x30 (~2k) indicates an NN with 3 deep layers and 30 neurons, with ~2,000 trainable parameters).

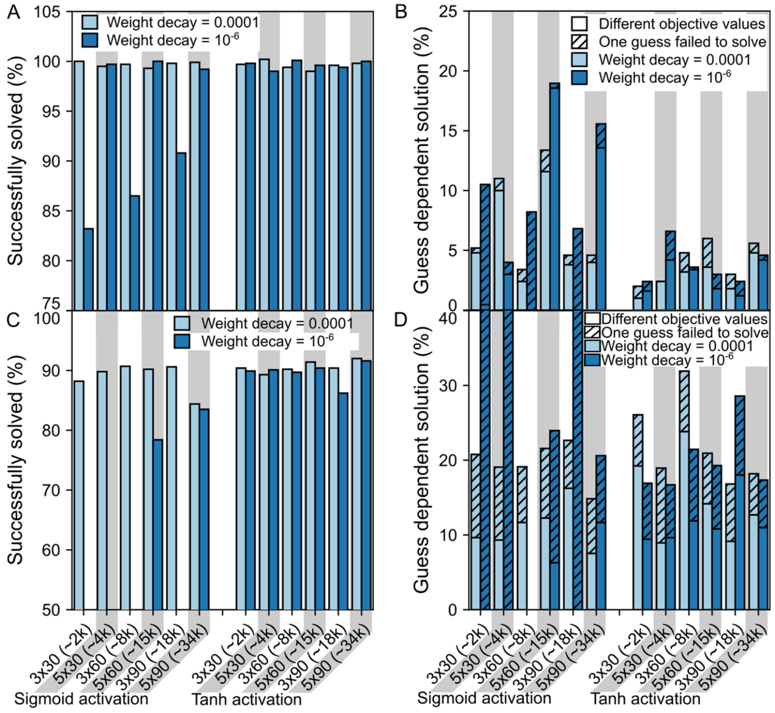

Figure 4: Effect of network architecture on optimization using IPOPT. (A) Percent of successful solves and (B) guess dependence for scaling problems. (C) Percent of successful solves and (D) guess dependence for precipitation problem. The network size is labeled as depth x width with values in parenthesis showing a rounded number of trainable parameters (e.g. 3x30 (~2k) indicates an NN with 3 deep layers and 30 neurons, with ~2,000 trainable parameters.

problems were solved using IPOPT with MA27 linear solver [14].

RESULTS AND DISCUSSION

Effect of data on Neural Network accuracy

The ultimate accuracy of NNs in predicting phenomena of interest depends on the statistical distribution of underlying training data and data size. In the case of aqueous chemistry used for water treatment applications, a common objective is to predict the propensity of ions in solution to form solids as a function of ion composition, pH, temperature, and pressure. Due to the sheer number of dimensions (12 dimensions are considered herein (Table 1)), a stochastic generation methodology must be used to generate data sets that enable the training of surrogate models applicable to different feed water composition and water treatment processes.

The nonlinear nature of aqueous chemistry requires a nonuniform sampling of ion compositions to ensure relevant data is generated from chemistry software. For example, Calcium is an ion of great interest in water treatment and will form Gypsum ($CaSO_4$) in the presence of SO_4, which has solubility below 2.5 g/kgw and limits Ca concentrations. However, in the absence of SO_4, calcium concentrations can exceed 10 g/kgw, although such waters are uncommon, as SO_4 is ubiquitous in real waters. Thus, uniformly sampling across ion concentrations will generally form ion compositions, scaling tendencies, and precipitation fractions that are not representative of real waters and conditions under which water treatment processes operate.

Skewing the sampling of ion concentrations to prioritize lower concentrations generates ion composition distributions that are more representative of real water compositions (Figure 2A). Increasing skewing magnitude from 10 to 10,000 results in a sampling of much lower ion concentrations on the median that approach USGS control set values but still sample high ion concentrations that occur in desalination processes.

Testing NNs against real ion composition and chemical additions that would be encountered by desalination and precipitation water treatment processes (USGS control set) demonstrates that accuracy increases with an increase in the skew magnitude (Figure 2B). At skew magnitude below 100, the median error for Calcite scaling tendency approached 10%. Increasing the skew magnitude above 200 reduced 75th percentile errors to below 5% for Calcite scaling tendency and reduced errors for precipitation fractions and pH.

Testing NNs using stochastically generated test data demonstrated a slight increase in prediction errors of less 0.5 % on the median with an increase in skew magnitude from 10 to 10,000. These test errors were significantly lower than those for USGS control set, with 75th percentile errors being less than <1% across all skew magnitudes. This clearly shows that NNs are learning chemistry in underlying training data, and skewing is shifting training to learning chemistry relevant to real water composition, as found in USGS control data set.

Increasing the data size used for training improves NN accuracy (Figure 2C). Training with less than 100,000 samples produces poor NN accuracy for Calcite scaling tendency, with median errors exceeding 10%. Increasing the sample size to above 200,000 samples reduced errors to below 5% in the 75th percentile for Calcite scaling tendency and the 95th percentile for pH and Calcite precipitation fraction. Increasing the sample size from 200,000 to 2,000,000 provided minimal improvements in overall network accuracy, suggesting that data sizes of around 500,000 could be sufficient for predicting scaling tendency. However, in the case of pH and precipitation fractions, data sizes of 50,000-100,000 samples could be adequate. The data size requirement is expected to increase with the number of inputs.

Effect of architecture on accuracy

We explore the effect of NN architecture by training 24 different NN types using the 2M sample data set with a skew magnitude of 5,000. Each NN predicts a single output, resulting in five scaling NNs per architecture that predict scaling tendencies for the four scalants and pH, and six precipitation NNs per architecture that estimate precipitation fractions for the five solids and pH, as shown in Table 1 and Table 2. Calcite scaling tendencies and calcite precipitation fractions always have higher errors and are thus used as benchmarks for the performance of scaling tendencies and precipitation fraction prediction.

Prediction error decreased with an increase in the number of NN trainable parameters (Figure 3). The smallest NNs with 3 deep layers and 30 neurons (3x30) have 2,281 trainable parameters and exhibited median error in scaling tendencies on the order of ~5% and ~7% for sigmoid and tanh activation functions, respectively. In contrast, the largest 5x90 networks with 34,021 trainable parameters had errors <3% in 75th percentile (Figure 3A). Similar improvements were observed for precipitation networks (3B).

In general, we observed that deeper networks had slightly lower errors than shallow networks with a similar number of trainable parameters. This is demonstrated by the 3x90 network with 17,641 trainable parameters typically exhibiting a higher or similar level of error compared to their 5x60 network counterparts with 15,481 trainable parameters. In theory, the three-layer deep and five-layer deep network should be able to encode the same degree of nonlinear behavior, but it appears that higher depth improves encoding performance.

Activation function type and weight decay had a small effect on prediction error relative to network size impact. On average, NNs with sigmoid activation function had lower errors than NNs with tanh activation function, but the difference was less than 0.5% for median errors. The weight decay did impact the accuracy of NNs, but the impact depends on their architecture, implying that each architecture has an optimal weight decay value.

Effect of Neural Network architecture on optimization

Surrogate models developed for equation-oriented programming models must faithfully capture underlying phenomena and provide smooth, continuous functions that ensure a successful optimization and avoid local solutions when solved with a local solver such as IPOPT. We test the NNs stability in optimization by integrating them in two black box desalination problems, the scaling tendency problem (Figure 1B), and the precipitation problem (Figure 1C).

The optimization results demonstrate that optimization problems that use NNs with tanh activation function solve more often, are less likely to fall into a local minimum, and solve more frequently regardless of initial guess (Figure 4). The use of the tanh activation function resulted in >98% of problems successfully solving for the scaling tendency problem and >88% for the removal fraction problem (Figure 4A and 4C). In contrast, the use of the sigmoid activation function in small and shallow networks that use low weight decay (10^{-6}) resulted in a large number of failed solves.

The optimization problems that used NNs with tanh activation function were not strongly impacted by initial guess relative to problems that used NNs with sigmoid activation function (Figure 4B and 4D). For the scaling tendency problem, using the tanh activation function resulted in less than 5% of solutions having different objective values caused by different initial guesses across all NNs. This contrasts the results for optimization problems that used NNs with the sigmoid activation function, even when it had a comparable number of solves to tanh, the sigmoid activation function produced a higher number of local solves (exceeding 10% in 4 of the NNs). This trend was not observed in the removal fraction problem, where a similar number of local minimums (~10% of samples) were observed for both activation functions. Similarly, for cases where the number of successful solves is comparable between tanh and sigmoid, the initial guess resulted in a similar number of failed solves.

We hypothesize that tanh function provides a more mathematically stable formulation than the sigmoid activation function, resulting in a higher likelihood that a problem is solved successfully, even as the use of the sigmoid function resulted in more accurate NNs. The tanh function provides larger gradients than the sigmoid activation function, improving the potential for IPOPT in avoiding local minimums. In addition, the sigmoid formulation utilizes an exponential in its denominator, which is prone to overflow errors during computation. The probability of overflow decreases as the network size increases and the weight decay parameter gets higher. Larger network size and weight decay regularize weights and reduce the potential for extreme values being passed into the activation function. This results in a lower likelihood of an exponential overflow occurring when changing NN input values, explaining why the sigmoid function performs better with the increase in trainable parameters and weight decay. With the tanh formulation, however, overflow can not occur, and optimization algorithms can freely vary the inputs.

NN architecture did not impact the number of iterations required to solve an optimization problem, but it did increase computational time. On average, the scaling problem required 20 iterations to solve, with 95th percentile of problems requiring 40 iterations, regardless of NNs used. The removal fraction problems required on average 40 iterations to solve, with 95th percentile requiring ~100 iterations, regardless of NNs used. The largest impact of increasing network size was on solving time, with the use of the 5x90 NNs requiring upto ten times more time per iteration than for 3x30 NNs. The activation function type and weight decay did not impact the solve time.

The integration of NNs into optimization problems did not impact their accuracy. The results demonstrated similar error distributions to those observed in USGS control set, confirming the expectation that using NNs in an equation-oriented model does not affect their accuracy.

CONCLUSION

The results of this work have demonstrated that NNs can be effective surrogate models for accurately predicting scaling tendencies, pH change, and removal fraction across a broad range of real water compositions and use in the equation-oriented programming models of water treatment processes. Our results have extracted a set of generalizable guidelines for data generation and design of NNs for chemistry prediction in water treatment equation-oriented process optimization:

- Sampling of ion compositions and chemical addition should be skewed to lower values to provide good accuracy in real water compositions.
- The number of required stochastically generated data samples depends on the underlying chemistry being modeled.
- Deeper networks provide slightly better performance over shallower networks with the same number of trainable parameters.
- Tanh activation function provides better stability than sigmoid activation functions in equation-oriented models, and should be preferred over sigmoid even at the cost of accuracy. This is a fundamental mathematical limitation and applies to all NNs, regardless of the data they are predicting.
- Weight decay should be optimized to a specific network architecture to extract the highest accuracy.
- Large NNs do not increase the number of iterations but do increase the solving time required per iteration.

The methods presented herein, for the first time, enable optimization studies of water treatment processes to consider a broad range of real water compositions, capturing their performance while faithfully representing real and complex aqueous chemistry.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Alliance for Water Innovation (NAWI), funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), Advanced Manufacturing Office, under Funding Opportunity Announcement Number DE-FOA-0001905. The version of IPOPT implemented in this work was compiled using HSL, a collection of Fortran codes for large-scale scientific

computation. See http://www.hsl.rl.ac.uk.

REFERENCES

1. Description of Input and Examples for PHREEQC Version 3—A Computer Program for Speciation, Batch-Reaction, One-Dimensional Transport, and Inverse Geochemical Calculations. (2013)
2. Pirdashti, M., Curteanu, S., Kamangar, M.H., Hassim, M.H., Khatami, M.A.: Artificial neural networks: applications in chemical engineering. Reviews in Chemical Engineering. 29, 205–239 (2013).
3. Gabelich, C.J., Rahardianto, A., Northrup, C.R., Yun, T.I., Cohen, Y.: Process evaluation of intermediate chemical demineralization for water recovery enhancement in production-scale brackish water desalting. Desalination. 272, 36–45 (2011).
4. Hydranautics: Chemical Pretreatment For RO and NF. Nitto Denko, (2008)
5. Hoover, C.P.: Review of Lime-Soda Water Softening. Journal - American Water Works Association. 29, 1687–1696 (1937).
6. Qi, S.L., Harris, A.C.: Geochemical Database for the National Brackish Groundwater Assessment of the United States, (2017)
7. Mease, D., Bingham, D.: Latin Hyperrectangle Sampling for Computer Experiments. Technometrics. 48, 467–477 (2006).
8. Ahdab, Y.D., Thiel, G.P., Böhlke, J.K., Stanton, J., Lienhard, J.H.: Minimum energy requirements for desalination of brackish groundwater in the United States with comparison to international datasets. Water Research. 141, 387–404 (2018).
9. Sheikholeslami, R., Bright, J.: Silica and metals removal by pretreatment to prevent fouling of reverse osmosis membranes. Desalination. 143, 255–267 (2002).
10. Stevens, E., Antiga, L., Viehmann, T.: Deep learning with PyTorch. Manning Publications Co, Shelter Island, NY (2020)
11. Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization, http://arxiv.org/abs/1711.05101, (2019)
12. Beattie, Keith, Gunter, Daniel, Knueven, Ben, Lee, Andrew, Ladshaw, Austin, Drouven, Markus, Bartholomew, Tim, Bi, Xiangyu, Bianchi, Ludovico, Arnold, Travis, Atia, Adam, Wang, Chenyu, Miara, Ariel, Kurban, Sitterley, Allu, Srikanth, Dudchenko, Alexander, Amusat, Oluwamayowa, Kinshuk, Panda, Young, Ethan: WaterTAP v1.0.0, (2021)
13. Ceccon, F., Jalving, J., Haddad, J., Thebelt, A., Tsay, C., Laird, C.D., Misener, R.: OMLT: Optimization & Machine Learning Toolkit.
14. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106, 25–57 (2006).

© 2024 by the authors. Licensed to PSEcommunity.org and PSE Press. This is an open access article under the creative commons CC-BY-SA licensing terms. Credit must be given to creator and adaptations must be shared under the same terms. See https://creativecommons.org/licenses/by-sa/4.0/

cc BY SA