Stable Soft Sensor Modeling for Industrial Systems

1st Liang Cao
Chemical and Biological Engineering
University of British Columbia
Vancouver, BC, V6T 1Z3, Canada
clubc19@mail.ubc.ca

2nd Yankai Cao
Chemical and Biological Engineering
University of British Columbia
Vancouver, BC, V6T 1Z3, Canada
yankai.cao@ubc.ca

3rd R. Bhushan Gopaluni
Chemical and Biological Engineering
University of British Columbia
Vancouver, BC, V6T 1Z3, Canada
bhushan.gopaluni@ubc.ca

***Abstract*—Learning trustworthy models is essential for machine learning tasks, as many researchers have revealed the vulnerability of machine learning models, especially when the fundamental independent and identically distributed (IID) assumption is not satisfied. Building a trustworthy model is promising when training on big representative data but fails to work with limited data. In this paper, we focus on solving small sample problems and unstable prediction problems in machine learning. First, to deal with small sample problems, we propose using a uniform manifold approximation and projection (UMAP) algorithm to generate high-quality virtual samples. Then, with the generated big data and original small data, we use the stable learning method to achieve stable predictions. In addition to a detailed description of the UMAP algorithm and the stable learning algorithm, we also discuss the corresponding theoretical explanations and implementation details. Finally, several comparison studies are implemented on the Tennessee Eastman benchmark process to validate the effectiveness of the proposed method.**

***Index Terms*—trustworthy model, small sample, UMAP, stable learning, soft sensor**

I. Introduction

The performance of a machine learning model is largely dependent on the quality and quantity of data [1]. A common practice is to feed large volumes of data to an algorithm and hope that it provides the best possible model. However, in real applications, data shortage is inevitable due to missing values, high-dimension space, noise, and many other reasons [2]. Data shortage makes it difficult to build an accurate model and this is often referred to as the small sample problem [3]. Few Shot Learning (FSL) is a common approach that builds models with limited data [4]. FSL uses a pre-trained/pre-tuned model to rapidly generalize to new tasks that contain only a few samples with supervised information. However, in a large number of real applications, the pretrained/pretuned model is not readily available.

Virtual Sample Generation (VSG) is another effective approach for small sample problem [5], [6]. VSG is an approach that generates virtual samples according to the distribution of available samples. There are several ways to generate virtual samples, including resampling [7], generative model [5], and dimension reduction [8]. The dimension reduction method is very powerful for high-dimensional virtual sample generation and we choose dimension reduction to generate virtual samples. The reason is that the distribution of data in high-dimensional space is usually very complex and it is difficult to extract accurate distribution information in high-dimensional space to generate virtual samples.

As shown in Figure 1, the dimension reduction is mainly classified into two groups, matrix factorization [9] and neighbor graphs [8], [10]. The basic idea of matrix factorization is to find two (or more) matrices whose products best approximate the original matrix. The neighbor graphs are based on the nearest-neighbor search to find a low-dimensional representation of the data, in which the distances in the low-dimensional space follow the distances in the original high-dimensional space. UMAP is a state-of-the-art neighbor graph algorithm and has excellent feature extraction performance [8]. In this work, we use UMAP to obtain the representation of the samples in a low-dimensional space, then generate virtual samples in a low-dimensional space, and finally project to a high-dimension space to get virtual samples.

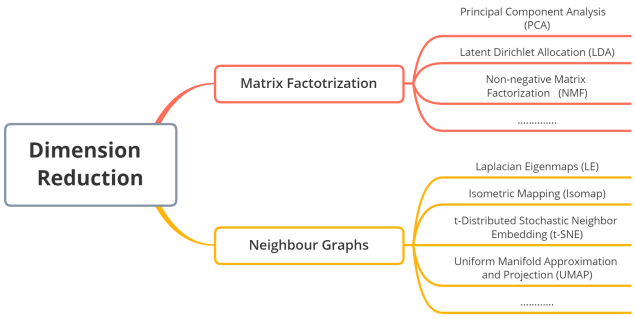

Fig. 1. Classification of dimension reduction algorithms

In machine learning algorithms, the same distribution of training data and test data ensures the predictive performance of the model for unknown test data sets. However, in practical applications, the data distribution of the unknown test set cannot be the same as that of the training set, which violates the fundamental assumption that training and testing data are IID [11]–[13]. This issue usually leads to inconsistent predictions of the model on different test sets. Although modern machine learning techniques have achieved significant success in many application areas, the drawback of not being able to guarantee prediction performance limits their utility. To address this challenge, some scholars proposed the concept

of stable learning that uses causality to guarantee prediction consistency even with out-of-distribution (OOD) test data sets [14]. Since causality is invariant across different distributions, it is obvious that prediction using causal variables is stable.

Stable learning algorithms can be thought of as feature selection mechanisms based on regression coefficients [15]. Assuming that one wants to predict a target variable with a set of variables, it is well known that some variables in the set may not provide useful information for modeling and in some cases degrade the model. Usually, a subset of variables contains all the useful information and that subset is called the Markov blanket [16]. The definition of Markov blanket comes from the structural causal model framework, which is a conceptual framework that describes the causal mechanisms of a system [17]. Stable learning algorithms are essentially aiming to find a Markov blanket to guarantee the generalization ability of the model in OOD cases.

Several efficient stable learning algorithms already exist in the literature. [13] proposed a framework to decorrelate every two features by minimizing the covariance matrix. [11] proposed a sample reweighting method to decorrelate all features by fitting a binary probabilistic classifier. [12] developed a deep learning model, called StableNet, which extends linear stable learning frameworks to nonlinear frameworks by adopting random Fourier features. However, these stable learning methods are generally affected by the sample size. They typically perform well with big data but fail when data are limited. Therefore, we need to propose a new method to improve the stable learning algorithm when the sample size is small.

The present work aims to introduce stable learning into the analysis of industrial process modeling with a small sample size. With few informative samples, we propose to use UMAP to generate virtual samples according to the distribution of original data. Then, stable learning is utilized with the enlarged dataset to achieve consistent prediction. Unlike traditional stable learning methods, the proposed algorithm performs well even when data are limited. The illustrative case study shows that the proposed stable learning model performs well even when the distributions of training and test data are different.

This article is organized as follows. In section 2, we propose new approaches for generating virtual samples and provide detailed implementation procedures with theoretical explanations. In section 3, we provide a review of the stable learning algorithm and discuss the theoretical aspects of stable learning. In section 4, we present case studies using the well-known Tennessee Eastman benchmark process to verify the effectiveness of the proposed methodology, and finally, in section 5 we present a brief summary.

Notation: Throughout the paper, we define X as a $n \times p$ matrix, n as the number of training samples, p as the number of features, x as the input vector, x_{ij} as the feature j in the training sample i, y_i as the label of a sample i, y as the list of n labels that containing the label y_i in position i, x_i as the list of all features for a sample i with p elements $x_{i1}, x_{i2}, ..., x_{ip}$. For virtual sample generation, we define $\tilde{X}$ as $m \times p'$ matrix, m as the number of virtual samples (normally, $m \gg n$), $\tilde{X}_{ij}$ as feature j in the training sample i, $\tilde{y}$ as the list of m labels that containing the label $\tilde{y}_i$ in position i.

II. Virtual Sample Generation

The virtual sample generation task can be defined as follows: Given the original data with input X and label y, the task is to generate virtual sample input $\tilde{X}$ and label $\tilde{y}$ according to the distribution of the original data.

A. Generation of low-dimensional virtual samples inputs based on UMAP

1) Introduction of UMAP: The theoretical basis of UMAP is mainly manifold theory and topological analysis. The two steps in the UMPA algorithm are graph construction and graph projection. The first step of UMAP is to build a weighted k-neighbour graph in high-dimensional space. For input X with a metric (dissimilarity measurement) d, given a hyper-parameter k, we compute the k nearest neighbors set $\{x_{i_1}, \cdots, x_{i_k}\}$ of each x_i under the metric d. If k is big, the global structure is better preserved. If k is small, the local structure is better preserved. Therefore, k can provide the balance between the preservation of local and global structures.

For each x_i, we define ρ_i as the distance from each $i-th$ data point to its first nearest neighbor (minimum distance) and the expression can be given as follows:

$$\rho_i = \min\left\{d\left(x_i, x_{i_j}\right) \middle| 1 \leq j \leq k, d\left(x_i, x_{i_j}\right) > 0\right\} \tag{1}$$

ρ_i varies from point to point, which guarantees the local connectivity of the manifold.

The normalization factor in UMAP is used to scale the distances between data points in the high-dimensional space to the distances in the low-dimensional space. This helps to preserve the relative distances between data points as much as possible, while also reducing the dimensionality of the data. Here, define σ_i as a normalization factor and set σ_i to be the value such that: $\sum_{j=1}^{k} \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right) = \log_2(k)$.

To describe the probability of the existence of directed edges (connections) between samples, we define a weighted matrix $\overline{H} = (V, E, w)$ where V is the number of vertices, E is the set of directed edges between x_i and its k nearest neighbors of x_i, $E = \left\{\left(x_i, x_{i_j}\right) | 1 \leq j \leq k, 1 \leq i \leq N\right\}$, w is the weight function and is defined as follows: $w\left(x_i, x_{i_j}\right) = \exp\left(\frac{-\max\left(0, d\left(x_i, x_{i_j}\right) - \rho_i\right)}{\sigma_i}\right)$. Intuitively, one can think of the weight of an edge as the probability of the existence of an edge between x_i and its neighbor x_{i_j}. After UMAP glues points together with locally varying metrics (via the parameter ρ_i), symmetrization is necessary because it may happen that the probability of the existence of directed edges between x_i and x_{i_j} is not equal to x_{i_j} and x_i. Define $w_h(i, j)$ as the

probability that there is at least one of the two directed edges and it can be given as follows:

$$w_h(i,j) = w\left(x_i, x_{i_j}\right) + w\left(x_j, x_{j_i}\right) - w\left(x_i, x_{i_j}\right) w\left(x_j, x_{j_i}\right) \tag{2}$$

The new UMAP graph H (different from $\overline{H}$) is then an undirected weighted graph whose adjacency matrix element is given by $w_h(i,j)$.

The second phase of UMAP involves projecting the graph H in high dimensions to the graph L in low dimensions. We would like to find low-dimensional position points $\{l_i\}_{i=1\cdots n}$ and its weighted graph L such that the graph L induced by those points most closely approximates the graph H, which allows to recover the important topology and retain the information of high-dimensional space to a large extent.

To construct a low-dimensional topological representation, the first question is how to determine the topological structure in low-dimensional space. UMAP uses $w_l(i,j)$ to model the probability of edge existence in low dimensions: $w_l(i,j) = \left(1 + a\left(\|l_i - l_j\|_2^2\right)^b\right)^{-1}$, where a and b are UMAP hyperparameters to control the topological structure in low-dimensional space.

The second question is how to find a good metric to measure the difference between two weighted graphs H and L, and then optimize the layout of the data representation in the low-dimensional space. UMAP uses total cross entropy on all edges as a cost function: $C = \sum_i \sum_j w_h(i,j) \log\left(\frac{w_h(i,j)}{w_l(i,j)}\right) + (1 - w_h(i,j)) \log\left(\frac{1-w_h(i,j)}{1-w_l(i,j)}\right)$.

Overall, UMAP constructs a high-dimensional graph representation of the data and then optimizes a low-dimensional graph to be as structurally similar as possible. We can perform a mapping from high dimension to low dimension to effectively extract rich information in the data. In this work, for sample $x_i = \{x_{i1}, x_{i2}, ..., x_{ip}\}$ with p features, UMAP is adopted to embed data into space with p' dimensions $l_i = [l_{i1}, l_{i2}, ..., l_{ip'}]$.

2) Interpolation of virtual samples in low-dimensional space: In the low-dimensional space, k nearest neighbors set $\{l_{i_1}, \cdots, l_{i_k}\}$ of l_i are selected to calculate the average value and this value is assigned to the virtual sample $\tilde{l}_i$: $\tilde{l}_i = \frac{1}{k}\sum_{j=1}^{k} l_{i_j}$. After repeating the interpolation step m times, m virtual samples are obtained in a low-dimensional space. Since the UMAP algorithm is stochastic, different runs with the same hyperparameter may yield different virtual samples; it would be very useful to evaluate the similarity score between real and virtual data. The Kullback-Leibler (KL) divergence [18] is designed to measure the similarity between two distributions. For distributions P and Q of a random continuous variable, the KL divergence from Q to P is defined as: $D_{KL}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$, where p and q denote the probability density of P and Q.

For the multivariate case, KL divergence of two random vectors that have multivariate Gaussian distributions f_z and $\tilde{f}_z$ can be given as follows:

Lemma 1 [19]: Under the assumption that $\tilde{f}_z$ and f_z are the densities of d_z dimensional normal random vectors $\tilde{z} \sim N\left(\tilde{\mu}_z, \tilde{\Sigma}_z\right)$ and $\tilde{z} \sim N(\mu_z, \Sigma_z)$, the KL divergence of $\tilde{f}_z$ with respect to f_z is given by:

$$\begin{aligned} D_{KL}\left(\tilde{f}_z \| f_z\right) &= \frac{1}{2}\left\{(\tilde{\mu}_z - \mu_z)^T {\Sigma_z}^{-1} (\tilde{\mu}_z - \mu_z)\right\} \\ &+ \frac{1}{2}\ln\left(\frac{|\Sigma_z|}{\left|\tilde{\Sigma}_z\right|}\right) + \frac{1}{2}\left\{tr\left({\Sigma_z}^{-1}\tilde{\Sigma}_z\right) - d_z\right\} \end{aligned} \tag{3}$$

Here, $ln()$ is the natural logarithm, $tr()$ is the trace of a square matrix. Lemma 1 will be used to evaluate the quality of virtual samples. Bad virtual dataset will be removed if $D_{KL}\left(P(virtual)\|P(original)\right)$ is larger than a predetermined threshold T.

B. Generation of virtual samples based on regression

It is important to note that UMAP warps the high-dimensional structure of the data when projecting to low dimensions; the distance in low dimensions is not directly reflected in the distance in high dimensions. Therefore, it is essential to use a regression model to predict virtual samples $\tilde{X}$ and $\tilde{y}$. For better visualization, we choose the dimension of the low-dimensional space as 3. First, p regression models are established with 3 low-dimensional input l and p high-dimensional output X in the original data. Then, we will use these established p regression models to predict the virtual dataset $\tilde{X}$ with KNN-interpolated $\tilde{l}$. Second, the output regression model is established with X as input and target y as output in the original data. Then, the output regression model will be used to predict the virtual sample output $\tilde{y}$ with the virtual sample input $\tilde{X}$. In this work, random forest regression is used for input regression and output regression.

III. Stable Learning

A. Stable learning and preliminaries

Stable learning can be defined as follows: given the target y and the input X, the objective is to find a robust model that can achieve consistent predictions in different distributions. As mentioned above, the core idea of stable learning is to eliminate the correlation of variables by reweighting. Consider the following linear regression model:

$$y = X^T \overline{\beta}_{1:p} + \overline{\beta}_0 + b(X) + \varepsilon \tag{4}$$

where $b(X)$ is a bias term with bound δ, i.e., $|b(X)| \leq \delta$, and ε is zero-mean noise with variance σ^2. Define $\bar{\beta} = \left[\bar{\beta}_0, \bar{\beta}_{1:p}\right]$ as the model parameter and $\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} \left({x_i}^T \beta_{1:p} + \beta_0 - y_i\right)^2$ as the least-squares solution. Two assumptions are needed for stable learning [11]: 1. There is a stable structure between y and X that remains invariant throughout different distributions. 2. There are spurious correlations which lead to unstable predictions across different distributions. With the assumptions mentioned above, the following propositions hold:

Proposition 1 [11]: Define γ^2 as the smallest eigenvalue of the covariance matrix $\operatorname{cov}(X,X) = E\left[(X-E(X))(X-E(X))^T\right]$, then the estimation error in the coefficients caused by the bias term $b(X)$ can be as bad as:

$$\left\|\hat{\beta}-\bar{\beta}\right\| \leq 2\left({}^{\delta}/_{\gamma}\right)+\delta \tag{5}$$

For proposition 1, we observe that the worst-case estimation error goes to infinity when γ goes to 0. This means that when the variables are highly correlated (the smallest eigenvalue γ is 0 when there are perfectly collinear variables existing), the error in parameter estimation will be amplified, resulting in unstable prediction results if these estimated parameters are used. To solve the collinearity problem, a sample reweighting method is proposed to reduce collinearity between input variables. If we can find $\hat{\beta}$ such that the estimation error $\left\|\hat{\beta}-\bar{\beta}\right\|_2 = O(\delta)$ (independent of γ), then the stability of the model can be guaranteed. Proposition 2 will provide a theoretical guarantee of the existence of sample weights that can reduce the correlation between the input variables.

Proposition 2 [11]: Let $p_u(x)$ be the uniform distribution on $\chi = \chi_1 \times \chi_2 \times \cdots \times \chi_p \subset \mathbb{R}^p$, and assume that $E_{x\sim p_u(x)}\|x\|_2^2 < \infty$ for each variable $x_j \in \chi_j$, and that the vector x has a density $p(x)$ on χ such that $0 < 2\gamma_0 \leq p_u(x)/p(x) \leq \gamma_1/2$. For all $\xi > 0, \varsigma > 0$, with probability larger than $1-\varsigma$, there exists w such that $\|w\|_1 = 1, \gamma_0/n \leq \|w\|_\infty \leq \gamma_1/n$ and

$$\left| n^{-1}\sum_{i=1}^{n} w_i\left(x_{ij}-\bar{x}_j\right)\left(x_{ik}-\bar{x}_k\right)\right| \leq \xi \tag{6}$$

where $\bar{x}_j = n^{-1}\sum_i x_{ij}$ is the mean of each variable j and $j \neq k$. The left side of equation 6 shows the off-diagonal elements of the covariance matrix. Assuming that we standardize all variables (all variables will have a mean of 0, the standard deviation of 1), the covariance matrix becomes a correlation matrix since $corr(X,X) = \frac{\operatorname{cov}(X,X)}{\sigma_X \sigma_X}$. Proposition 2 demonstrates that the elements of the correlation matrix can be constrained arbitrarily small (no larger than ξ) with the sample weight w_i. As $\gamma^2 \geqslant 1-(p-1)\xi$, by reducing the pairwise correlation between variables, we can adjust the smallest eigenvalue to be not close to 1. In this way, the problem of unstable parameter estimation mentioned in proposition 1 is solved.

B. Stable learning algorithms

The framework of typical stable learning algorithms is two steps, sample reweighting, and weighted least squares. In step 1, for training data, the sample weights are learned by the sample reweighted decorrelation operator (SRDO) to ensure statistical independence between the features [11]. Figure 2 is an example of SRDO. First, we use the matrix X to generate a column-decorrelated $\hat{X}$ by performing a random resampling column-wise, where $i, j, k, r, s, t, u, v, w$ are drawn from $1, 2, \cdots, n$ at random. Random resampling can break down the joint distribution D of X into p independent marginal distributions $\hat{D}$ of $\hat{X}$. Since $\hat{X}$ is completely independent of columns, which means we can transfer the original X to the decorrelated $\hat{X}$ by SRDO.

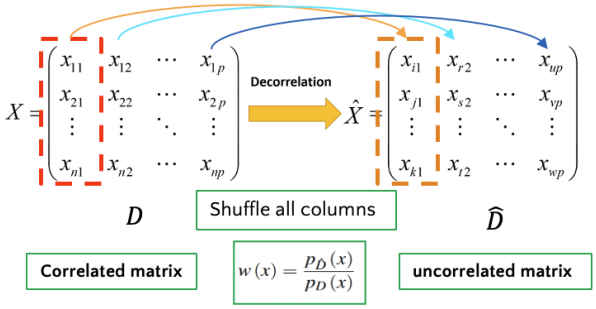

Fig. 2. A graphical representation of the sample reweighted decorrelation operator

Specifically, we set the samples in $\hat{X}$ as positive samples $(Z=1)$ while the samples in X as negative samples $(Z=0)$ and fit a binary probabilistic classifier. The decorrelated weight can be given as follows:

$$w(x) = \frac{p_{\hat{D}}(x)}{p_D(x)} = \frac{p(Z=1|x)}{p(Z=0|x)} \tag{7}$$

where $p(Z=1|x)$ is the estimated probability that the sample x is drawn from $\hat{D}$ and $p(Z=0|x)$ is the estimated probability of sample x being drawn from D. For example, for the first sample $x_1 = [x_{11}, x_{12}, \cdots, x_{1p}]$ in X, if the binary classifier outputs $p(Z=1|x_1) = 0.5$ and $p(Z=0|x_1) = 0.5$, the weight $w(x_1) = 1$. It means that x_1 is very similar to $\hat{x}_1 = [x_{i1}, x_{r2}, \cdots, x_{up}]$, the classifier cannot tell that x_1 comes from D or $\hat{D}$, so it is a good sample in X and gets a high weight. If the binary classifier outputs $p(Z=1|x_1) = 0.01$ and $p(Z=0|x_1) = 0.99$, the weight $w(x_1) \approx 0.01$. This means that x_1 is very different from $\hat{x}_1$, it is not a good sample in X, and we get a very small weight.

In step 2, learned sample weights are incorporated into the weighted regression method to obtain the solution $\hat{\beta}_w = \arg\min_{\beta_w} \sum_{i=1}^{n} w(x_i)\left(x_i{}^T\{\beta_{1:p_w} + \beta_{0w} - y_i\right)^2$.

C. Reasons for combining virtual sample generation and stable learning

Recently, stable learning algorithms have been shown to be very effective in improving generalization in some machine learning tasks. However, it is important to note that stable learning algorithms are generally affected by the sample size. In many applications, decreased performance has been observed in a small sample dataset due to the effect of variance inflation (mentioned in Proposition 1) in parameter estimation. Inspired by the idea of virtual sample generation, it is natural to generate a large number of virtual samples to improve the performance of the stable learning algorithm. Figure 3 shows the framework of the proposed algorithm.

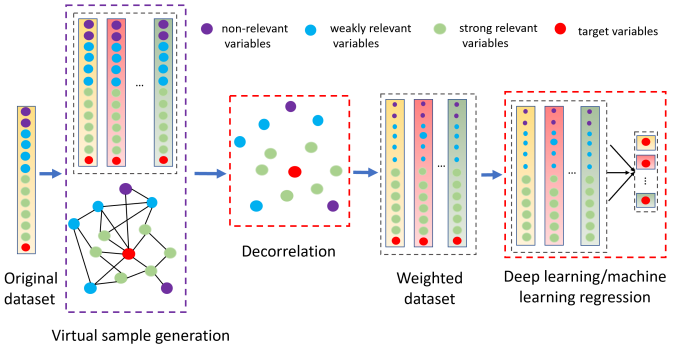

Fig. 3. Framework of stable learning with small samples

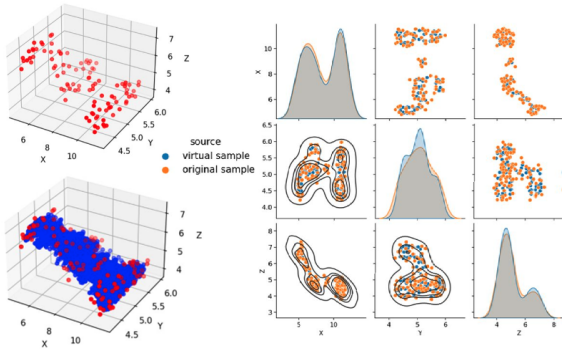

Fig. 4. Left: Low dimension representation of original data (red dots) and virtual generated data (blue dots) Right: Pairwise relationship between virtual data features and original data features in 3-dimension space

IV. Case Study

The effectiveness of the proposed methodology is illustrated in the Tennessee Eastman process benchmark (TEP) [20]. TEP provides a realistic industrial process monitoring benchmark. In this case study, 33 variables are chosen as the input variables; component C in the purge gas is chosen as the target variable to be predicted. To simulate different distributions on test data, normal operating condition data and 5 faulty conditions data are used. The detailed information about 5 faulty cases can be found in Table I. To simulate the small sample scenario, only 100 samples are used for training and 800 samples are used for testing. The threshold T of KL divergence is set at 0.5.

TABLE I
5 Process Faults

Fault case	Description	Type
Fault case 1	A Feed Loss (Stream 1)	Step
Fault case 2	A,B,C Feed Composition (Stream 4)	Random Variation
Fault case 3	D Feed Temperature (Stream 2)	Random Variation
Fault case 4	Reaction Kinetics	Slow Drift
Fault case 5	Reactor Cooling Water Valve	Sticking

In the virtual sample generation section, first, the dimensions of the input variables are reduced from 33 to 3 (define the 3-dimension as X, Y, Z) by UMAP graph projection; then, 5000 virtual samples are generated based on the low-dimension interpolation. As shown in Figure 4, the red dots are the original samples, and the blue dots are the virtual samples generated. As we can see, the blue dots fill the space between the red dots indicating that the generated virtual samples are a good approximation of the original data.

The KL divergence is 0.37 between the original distribution and the virtual distribution in a 3-dimensional space. Figure 4 shows multiple pairwise distributions of the virtual data subset and the original data in a 3-dimensional space. It can be seen that virtual samples and original samples have almost similar distributions in every low dimension X, Y, Z. After KNN interpolation, the input and output regression models are established to generate virtual samples.

In the stable learning section, stable regression models are built with weighted samples (generated 5000 virtual samples and 100 original samples). Here, we choose the linear regression method as the regression model. Four methods, linear regression with original data (small data), linear regression with virtual data and original data, stable learning linear regression with weighted original data (SL) and stable learning linear regression with weighted virtual generated data and weighted original data (VSGSL), are used to compare the performance in 6 different test data distributions.

TABLE II
Performance Comparison of Different Regression Methods

		small data	VSG(5000)	SL	VSGSL(5000)
Normal	RMSE	**0.486**	0.364	0.472	**0.359**
	R^2	0.035	0.080	**0.033**	**0.106**
Fault 1	RMSE	**5.923**	0.680	2.449	**0.665**
	R^2	0.043	0.236	**0.028**	**0.273**
Fault 2	RMSE	1.684	1.131	**1.707**	**1.099**
	R^2	**0.028**	0.209	0.033	**0.251**
Fault 3	RMSE	**3.143**	2.089	3.049	**2.060**
	R^2	0.292	0.367	**0.222**	**0.405**
Fault 4	RMSE	**2.910**	1.640	2.417	**1.613**
	R^2	**0.002**	0.328	0.006	**0.353**
Fault 5	RMSE	**3.242**	1.678	2.69	**1.632**
	R^2	**0.003**	0.355	0.004	**0.390**

Table II lists the detailed comparison results of different methods in different cases. The numbers in bold blue means the worst performance, while the bold red means the best performance among all methods. From Table II, it can be seen that, for training data and test data with the same distribution (normal case), VSG and SL show an improvement in $RMSE$ and R^2 compared to using only small data, and the proposed VSGSL method has the best performance. For example, for fault case 1, the VSGSL method has an improvement of about 10 times in $rmse$ and 6 times improvement in r^2 compared to the small data case.

In this case study, we generated 5000 virtual samples to build the models. The number of virtual samples may have an impact on model performance. To evaluate the impact of virtual sample sizes on model performance, we test the performance of VSG and VSGSL with 100, 500, 1500, 5000, 7500, and 9000 virtual samples in six different scenarios.

TABLE III
PERFORMANCE COMPARISON OF VSG AND VSGSL WITH DIFFERENT NUMBER OF VIRTUAL SAMPLES

VSG		100	500	1500	5000	7500	9000
Normal	rmse	**0.395**	0.367	0.365	**0.364**	0.369	0.372
	r2	**0.023**	**0.082**	0.079	0.080	0.058	0.045
Fault 1	rmse	**0.752**	0.689	0.686	**0.680**	0.691	0.700
	r2	**0.133**	0.229	0.228	**0.236**	0.213	0.195
Fault 2	rmse	**1.315**	1.140	**1.113**	1.131	1.184	1.193
	r2	**0.073**	0.186	0.208	**0.209**	0.152	0.137
Fault 3	rmse	**2.221**	**1.929**	2.003	2.089	2.094	2.143
	r2	**0.177**	**0.451**	0.410	0.367	0.335	0.294
Fault 4	rmse	**1.894**	1.711	1.642	**1.640**	1.675	1.687
	r2	**0.194**	0.293	0.325	**0.328**	0.296	0.285
Fault 5	rmse	**2.067**	1.897	1.730	**1.678**	1.795	1.794
	r2	**0.180**	0.278	0.334	**0.355**	0.280	0.271
VSGSL		100	500	1500	5000	7500	9000
Normal	rmse	**0.378**	0.363	0.359	**0.329**	0.364	0.366
	r2	**0.060**	0.098	0.089	**0.106**	0.082	0.069
Fault 1	rmse	**0.711**	0.677	0.679	**0.665**	0.675	0.683
	r2	**0.204**	0.254	0.244	**0.273**	0.249	0.232
Fault 2	rmse	**1.241**	1.120	1.100	**1.099**	1.151	1.160
	r2	**0.120**	0.211	0.223	**0.251**	0.191	0.177
Fault 3	rmse	**2.073**	**1.874**	1.983	2.060	2.050	2.096
	r2	**0.301**	**0.494**	0.431	0.405	0.379	0.344
Fault 4	rmse	**1.854**	1.687	1.628	**1.613**	1.643	1.654
	r2	**0.231**	0.313	0.336	**0.353**	0.324	0.315
Fault 5	rmse	**2.072**	1.894	1.715	**1.632**	1.753	1.749
	r2	**0.222**	0.291	0.345	**0.390**	0.311	0.306

Table III gives the results of VSG and VSGSL with different numbers of virtual samples. Both VSG and VSGSL have the worst performance when the number of virtual samples is 100, and almost the best performance when the number is 5000. As the number of virtual samples increases, the performance of the model improves first; when the number of virtual samples reaches a certain level (5000 virtual samples in this work), the model performance is optimal; and then, as the number of virtual samples increases, the model performance slowly declines. The reason may be that more virtual samples would also lead to an increase in the number of low-quality virtual samples (that cannot represent the real-world problem being modeled), and these low-quality samples may have a significant impact on the performance of the model. In this case, having more virtual samples may actually decrease the performance of the model. It should be noted that the optimal number of virtual samples is determined by trial and error in this work, and an effective method to determine the optimal number of virtual samples will be studied in future work.

V. CONCLUSION

This study addresses the problem of building trustworthy models when only a small amount of data is available and the underlying assumption of IID about the data distribution is not satisfied. Considering that current stable learning algorithms are generally affected by sample size, a stable learning algorithm based on UMAP virtual sample generation is proposed. KL-divergence is used to evaluate the quality of generated virtual samples and then select good virtual samples to build a trustworthy model. The effectiveness of the proposed method is validated in 6 different cases of the Tennessee Eastman process. This study shows that the integration of virtual sample generation and stable learning is very promising in terms of improving model accuracy and generalization.

REFERENCES

[1] J. Fan, F. Han, and H. Liu, "Challenges of Big Data analysis," *National Science Review*, vol. 1, no. 2, pp. 293–314, Feb 2014.

[2] A. R. T. Donders, G. J. van der Heijden, T. Stijnen, and K. G. Moons, "Review: A gentle introduction to imputation of missing values," *Journal of Clinical Epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.

[3] M. Wasikowski and X.-w. Chen, "Combating the small sample class imbalance problem using feature selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1388–1400, 2010.

[4] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, Jun 2020.

[5] Q.-X. Zhu, K.-R. Hou, Z.-S. Chen, Z.-S. Gao, Y. Xu, and Y.-L. He, "Novel virtual sample generation using conditional gan for developing soft sensor with small data," *Engineering Applications of Artificial Intelligence*, vol. 106, p. 104497, 2021.

[6] Y.-L. He, Q. Hua, Q.-X. Zhu, and S. Lu, "Enhanced virtual sample generation based on manifold features: Applications to developing soft sensor using small data," *ISA Transactions*, 2021.

[7] M. K. P. B. Kulesa, Anthony and N. Altman, "Sampling Distributions and the Bootstrap," *Nature Methods*, vol. 12, no. 6, p. 477–478, 2015.

[8] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *ArXiv e-prints*, Feb. 2018.

[9] I. S. Dhillon and S. Sra, "Generalized nonnegative matrix approximations with bregman divergences," in *Proceedings of the 18th International Conference on Neural Information Processing Systems*, ser. NIPS'05. Cambridge, MA, USA: MIT Press, 2005, p. 283–290.

[10] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[11] Z. Shen, P. Cui, T. Zhang, and K. Kunag, "Stable learning via sample reweighting," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5692–5699, April 2020.

[12] X. Zhang, P. Cui, R. Xu, L. Zhou, Y. He, and Z. Shen, "Deep stable learning for out-of-distribution generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5372–5382.

[13] K. Kuang, P. Cui, S. Athey, R. Xiong, and B. Li, "Stable prediction across unknown environments," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1617–1626.

[14] X. Zhang, P. Cui, R. Xu, L. Zhou, Y. He, and Z. Shen, "Deep stable learning for out-of-distribution generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 5372–5382.

[15] R. Xu, P. Cui, Z. Shen, X. Zhang, and T. Zhang, "Why stable learning works? A theory of covariate shift generalization," 2021. [Online]. Available: https://arxiv.org/abs/2111.02355

[16] K. Yu, L. Liu, and J. Li, "A unified view of causal and non-causal feature selection," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 4, April 2021. [Online]. Available: https://doi.org/10.1145/3436891

[17] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*, 1st ed. USA: Basic Books, Inc., 2018.

[18] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951. [Online]. Available: https://doi.org/10.1214/aoms/1177729694

[19] J. Zeng, U. Kruger, J. Geluk, X. Wang, and L. Xie, "Detecting abnormal situations using the kullback-leibler divergence," *Automatica*, vol. 50, no. 11, p. 2777–2786, Nov 2014.

[20] B. R. Russell E.L., Chiang L.H., *Data-driven Methods for Fault Detection and Diagnosis in Chemical Processes*. London: Springer, 2000.