Data-driven Transient Stability Assessment Model Considering Network Topology Changes via Mahalanobis Kernel Regression and Ensemble Learning

Xianzhuang Liu, Xiaohua Zhang, Lei Chen, Fei Xu, and Changyou Feng

***Abstract*—Transient stability assessment (TSA) is of great importance in power system operation and control. One of the usual tasks in TSA is to estimate the critical clearing time (CCT) of a given fault under the given network topology and pre-fault power flow. Data-driven methods try to obtain models describing the mapping between these factors and the CCT from a large number of samples. However, the influence of network topology on CCT is hard to be analyzed and is often ignored, which makes the models inaccurate and unpractical. In this paper, a novel data-driven TSA model combining Mahalanobis kernel regression and ensemble learning is proposed to deal with the problem. The model is a weighted sum of several sub-models. Each sub-model only uses the data of one topology to construct a kernel regressor. The weights are determined by both the topological similarity and numerical similarity between the samples. The similarities are decided by the parameters in Mahalanobis distance, and the parameters are to be trained. To reduce the model complexity, sub-models within the same topology category share the same parameters. When estimating CCT, the model uses not only the sub-model which the sample topology belongs to, but also other sub-models. Thus, it avoids the problem that there may be too few data under some topologies. It also efficiently utilizes information of data under all the topologies. Moreover, its decision-making process is clear and understandable, and an effective training algorithm is also designed. Test results on both the IEEE 10-machine 39-bus and a real system verify the effectiveness of the proposed model.**

***Index Terms*—Transient stability assessment, critical clearing time, network topology change, Mahalanobis kernel regression, ensemble learning, data-driven.**

NOMENCLATURE

$\boldsymbol{x}$	Numerical variables of pre-fault power flow
$\boldsymbol{\zeta}$	Topological variables of pre-fault power flow
$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$	Arbitrary vectors describing power flow
γ	Smoothing parameter in Mahalanobis kernel regression (MKR)
$\boldsymbol{M}$	Smoothing parameter in MKR
$\kappa_{\mathrm{MD}}(\cdot)$	Mahalanobis kernel function
$[\cdot]^X$	Parameter or function for the numerical part, $[\cdot]$ can be one of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, γ, $\boldsymbol{M}$ and $\kappa_{\mathrm{MD}}(\cdot)$
$[\cdot]^Z$	Parameter or function for the numerical part
$\mathcal{M}$	Parameter set of $\boldsymbol{M}_t$ for the advanced model
Γ	Parameter set of γ_t for the advanced model
N^{train}	Number of train samples
$\boldsymbol{T}_i^{\mathrm{train}}$	Topological part of the i^{th} training sample (a binary vector)
$\boldsymbol{T}^{\mathrm{TRAIN}}$	Matrix consisting of $\boldsymbol{T}_i^{\mathrm{train}}$
T_s^{uni}	The s^{th} unique topology pattern
$\mathbb{T}_t$	The t^{th} topology category
$\mathbb{T}$	Set of all topology categories
$\boldsymbol{X}_i^{\mathrm{train}}$	Numerical part of the i^{th} training sample (a numerical vector)
$\boldsymbol{X}^{\mathrm{TRAIN}}$	Matrix consisting of $\boldsymbol{X}_i^{\mathrm{train}}$
Y	Value of critical clearing time (CCT)
Y_i^{train}	CCT of the i^{th} training sample

I. INTRODUCTION

TRANSIENT stability assessment (TSA) plays an important role in power system operation and control [1]. TSA analyzes the stability or stability index of a power system under some credible faults given in the form of a "critical fault set" [2]. The stability index is often expressed by the so-called critical clearing time (CCT) [3]-[5]. CCT under a given fault is related with the following two factors: the network topology and the pre-fault power flow. Mathematically, the core of TSA is to find the mapping or function between CCT and both the network topology and the pre-fault power flow [6], [7].

Manuscript received: May 28, 2020; accepted: October 12, 2020. Date of CrossCheck: October 12, 2020. Date of online publication: November 26, 2020.

This work was supported by National Key R&D Program of China (No. 2018YFB0904500) and State Grid Corporation of China (No. SGLNDK00KJJS1800236).

X. Liu and C. Feng are with State Grid Corporation of China, Beijing 100031, China (e-mail: liu-xianzhuang@sgcc.com.cn; changyou-feng@sgcc.com.cn).

X. Zhang is with State Grid Jibei Electric Power Company, Beijing 100053, China (e-mail: zhang-xiaohua@sgcc.com.cn).

L. Chen (corresponding author) and F. Xu are with the State Key Laboratory of Control and Simulation of Power System and Generation Equipment, Department of Electrical Engineering, Tsinghua University, Beijing 100084, China (e-mail:chenlei08@tsinghua.edu.cn; xufei@tsinghua.edu.cn).

DOI: 10.35833/MPCE.2020.000341

Commonly, TSA is completed by the following three methods: the time-domain simulation (TDS) method, the direct method, and the data-driven method. The TDS method calculates the CCT with the help of simulation softwares. The results are generally considered to be accurate. However, it is time-consuming and computationally burdensome [8], [9]. The direct method, such as the transient energy function method, analytically solves the problem and directly gives the results without simulation [10], [11]. However, it highly depends on the system model and can only adopt simple models at present, which limits its further application. In recent years, data-driven methods have developed rapidly to deal with the above issues [12], [13]. These methods exploit massive simulation data and machine learning tools to construct assessment models, which describe the aforementioned mapping between pre-fault power flow or network topology and the CCT. When the model has been constructed at the offline stage, the system status is then input into the model and the CCT can be obtained very quickly at the online stage. Although its interpretability is to be further studied, it shows good performance in practice [14]-[17].

Compared with the pre-fault power flow, it is more significant yet harder to analyze the influence of network topology. Usually the network topology can be expressed by a series of 0-1 variables [18]. For example, we can use a 0-1 variable to describe the status of an AC line. The variable is 1 when it is working, and is 0 when it is not working due to maintenance or outage.

Network topology changes have a greater influence on the CCT than the pre-fault power flow. The reason is as follows. The stability of the system is determined by a set of differential algebraic equations (DAEs) [19]. When the pre-fault power flow changes, it only influences the initial value of the DAE. While the topology changes, both the initial value and the parameters of the DAE will be influenced. So it is harder yet more important to analyze the impact of the network topology on CCT. In the data-driven method, various issues emerge when the topology changes are taken into consideration. Firstly, the model introduces 0-1 variables, which are hard to deal with. Therefore, the model becomes more complex, and the accuracy will decrease. Secondly, the requirement for data volume explodes, since we need more data to train a more refined model. For example, assume the data requirement is N for the model that does not consider topology changes. Then, the model considering 10 topology changes raises the data requirement to $10N$, since it needs to train 10 separate models. If the data are not enough, there may be only a few data under some network topologies, which makes the model not substantial. Thirdly, since the model complexity increases, the parameters within the model also boom, so the training for the model will be much harder.

In most data-driven methods, the impact of the network topology changes is not considered. With those methods which take the network topology changes into consideration, the issue is usually solved through the following several ways.

1) The most direct method is to train a separate model for each possible topology. The defects are obvious. Firstly, it is computationally burdensome, since a large number of models need to be trained, and it is nearly impossible in real practice. Secondly, it is rough when the data under some topologies are few, and then the model will be rather inaccurate. Thirdly, it is hard to analyze the relationship between the topology and the CCT. Finally, it wastes the information within the data of other topologies.

2) The second method is to take the statistical variables as inputs, such as percentiles and variances [20], [21]. In this manner, the influence of the topology changes is contained in the statistical variables. The method is usually combined with transfer learning, and is capable of working in two or more different systems (or say it is self-adaptive). There are two main disadvantages of this method. The first is that it is relatively rough. It usually focuses on how a trained model can be applied to a totally different system, whereas with one system, the topology changes usually do not make such a huge difference, so the model may not be able to distinguish the difference well. Secondly, it usually needs a large number of training data to yield a fine model. Effective data generation methods has been invented, but not been verified on a real system yet.

3) The third idea is to use the numerical variables to indicate the on-off states of the elements. For example, if the line flow equals to zero, the line will be off, otherwise the line will be on. In this way, we only need to include the numerical variables as inputs, just like models that do not consider topology changes. The topology information is implicitly contained in the values of the numerical variables. However, this method ignores some special situations, such as the line is not transmitting power (the line flow is zero) but still in operation. Moreover, since the topology changes influence the system stability significantly, the function may change very dramatically around 0 if we do not include 0-1 variables, and thus bring the overfitting problem.

4) Another idea is to take the 0-1 variables as inputs. However, without efficient models, there are problems of lacking enough data and model complexity.

As we can see, the two main issues here are the introduction of 0-1 variables and the exploding data requirements. To address the issues, this paper proposes a novel data-driven TSA model via Mahalanobis kernel regression (MKR) and ensemble learning [22], [23]. The model originates from the reformulation of MKR, and the decision-making procedure is understandable. It introduces 0-1 variables in MKR to describe network topology and utilizes ensemble learning to combine different sub-models under different topology changes. The weights of the sub-models are determined by the topological similarities and total numerical similarities. It makes efficient use of data under different network topologies, and thus enhances the estimation accuracy and reduces the need for training samples. Furthermore, the results given by the model are informative and can enlighten further analysis, such as the relationship between similar topologies.

The rest of the paper is organized as follows. In Section II, the basic model is introduced, which is able to yield a result considering topology changes, but not practical in real practice. In Section III, the advanced model is introduced which is based on the basic model and ensemble learning.

Further explanation about how the advanced model works is also given. In Section IV, the training algorithm of the proposed model is given. In Section V, the practical application flowchart of the proposed method is presented. In Section VI, the proposed model is verified on two systems, and shows good performances. Section VII concludes the paper and introduces future works.

II. Basic Model

A. Problem Formulation

In this paper, we use CCT as the stability index for TSA. In the power grid, once the dynamic parameters of the elements and the faults are given, then CCT is a function of both the pre-fault power flow (represented by the numerical variables $\boldsymbol{x}$) and the network topology (represented by the topological variables $\boldsymbol{\zeta}$).

We use Y to denote the value of CCT under given $\boldsymbol{x}$ and $\boldsymbol{\zeta}$. Then Y can be expressed as:

$$Y = F(\boldsymbol{x}, \boldsymbol{\zeta}) \tag{1}$$

Each element of $\boldsymbol{x}$ or $\boldsymbol{\zeta}$ represents a feature (dimension). Typical features are listed in Table I.

TABLE I
Typical Variables Describing Pre-fault Power Flow

Element type	Numerical variable		Topological variable	
	Symbol	Description	Symbol	Description
Bus	e_k, f_k	Real and image part of the voltage of bus k	τ_k	On-off state of bus k
Branch (including line and transformer)	$P^i_{k,j}, Q^i_{k,j}$	Active and reactive power injected from bus k through line $L^i_{k,j}$	$\tau^i_{k,j}$	On-off state of branch $L^i_{k,j}$
	Q^c_k	Capacitive charging power of L^i at the k, j side of bus k		
Generator	P^g_k, Q^g_k	Active and reactive power generations of the generator connected to bus k	τ^g_k	On-off state of the generator connected to bus k
Load	P^l_k, Q^l_k	Active and reactive power consumptions of the load connected to bus k	τ^l_k	On-off state of the load connected to bus k

The features in $\boldsymbol{x}$ are continuous, and we apply normalization to them to yield values ranging from −1 to 1. The features in $\boldsymbol{\zeta}$ are discrete, and we use −1 and 1 to denote the on-off state of the corresponding element, −1 for off and 1 for on, respectively. Note that the normalization range is $[-1, 1]$, but not $[0, 1]$. The reason is as follows. In the calculation of MKR, rotation and stretch are applied to the data. Therefore, we prefer the data evenly distributed on both sides of the axis. So we tailor the data into $[-1, 1]$, and in order to maintain consistency, we use −1 and 1 to denote the on-off state, respectively.

In fact, CCT is also related with the contingencies. Under different contingencies, the CCT is different even the power flow and the network topology are the same. However, in real practice, the so-called severe contingency set is often employed to consider the severest faults. It is generally agreed that if all the contingency constraints in the severe contingency set are satisfied, the system will be secure.

Hence the problem is clear: our target is to predict Y with certain $\boldsymbol{x}$ and $\boldsymbol{\zeta}$. In other words, we need to find a new mapping to make a best approximation of the real mapping, i.e., $F(\cdot)$:

$$\hat{Y} = f(\boldsymbol{x}, \boldsymbol{\zeta}) \tag{2}$$

The mapping is highly non-linear and complex, so the analytical solution can hardly be found. Therefore, a variety of data-driven methods are utilized to solve the problem. Besides, the influence of $\boldsymbol{\zeta}$ on Y is more critical and complicated. That is to say, potential topology changes will raise a challenge in terms of both model accuracy and model complexity. Therefore, topology changes are usually needed to be specially treated in data-driven models.

As mentioned in Section I, there are usually four ways to deal with the issue. The first is to train a sub-model $g(\boldsymbol{x})$ with each possible value $\boldsymbol{\zeta}$. The second is to take the statistical variables as inputs. The third is to ignore the topological variables and contain the network topology into the numerical variables. The fourth is to take the topological variables (0-1 variables) as inputs directly.

In this paper, we propose a novel data-driven model to solve the issue. Generally speaking, it combines the first and last ideas mentioned above. The model is advantageous in the following aspects.

1) It is a non-parametric model, namely MKR, which is more capable of shaping nonlinear complex mappings. Non-parametric models have been utilized in TSA in many studies and show good performances [17], [24]. However, most of them employ additive models and the practical explanation is not quite clear. MKR used in this paper is more comprehensible and has a clearer probabilistic meaning, which makes it more practical [25].

2) It is reformed by ensemble learning to deal with the topology problem, which is more practical in usage and more understandable. Ensemble learning has been exploited in recent researches and works as well [26], [27]. But this method has yet not been tested in the situation with topology changes. In this paper, we tailor the ensemble learning method to improve the MKR model, thus addressing the issue of topology change.

B. Basic Model

In [25], we have already proposed a non-parametric TSA model, however, it does not take topology changes into consideration. In fact, even if we include topological variables into the so-called the basic model, it still cannot handle the topology changes very well.

In this part, we first briefly introduce the basic conception of MKR. Then, we include the topological variables in the model to construct a decoupled-distance model, which is the basic model.

MKR is based on Nadaraya-Watson kernel regression (NWKR) and Mahalanobis distance. NWKR uses the weighted sum of all the samples to yield an estimated value of CCT. The weight of a sample is determined by the distance between itself and the data to be assessed. MKR further substitutes the Euclidean distance with Mahalanobis distance in the Gaussian kernel. This will lead to a better model since it considers the correlation between the features.

The basic form of MKR is:

$$\hat{Y}(\boldsymbol{x})=\frac{\sum Y_i^{\text{train}}\kappa_{\text{MD}}\left(\boldsymbol{x},\boldsymbol{X}_i^{\text{train}}\right)}{\sum\kappa_{\text{MD}}\left(\boldsymbol{x},\boldsymbol{X}_i^{\text{train}}\right)} \tag{3}$$

$$\kappa_{\text{MD}}(\boldsymbol{\alpha},\boldsymbol{\beta})=\exp\left\{-\gamma(\boldsymbol{\alpha}-\boldsymbol{\beta})\boldsymbol{M}(\boldsymbol{\alpha}-\boldsymbol{\beta})^{\text{T}}\right\} \tag{4}$$

As we can see, different from Gaussian kernel, Mahalanobis kernel uses the distance instead of L2 norm. When $\boldsymbol{M}$ is an identity matrix, (4) degrades into Gaussian kernel. The parameters γ and $\boldsymbol{M}$ are those we need to train.

In (3), we does not take the influence of $\boldsymbol{\zeta}$ into consideration. In fact, we can regard $[\boldsymbol{x},\boldsymbol{\zeta}]$ as a whole input for (3). However, in real practice, we find that the overfitting problem arises. Besides, the model is not well explainable.

In the light of this, we decouple the numerical and topological variables in the calculation of distance in (4):

$$\begin{aligned}\kappa\left(\boldsymbol{\alpha}^X,\boldsymbol{\beta}^X,\boldsymbol{\alpha}^Z,\boldsymbol{\beta}^Z\right)&=\exp\left\{-\gamma^X\left(\boldsymbol{\alpha}^X-\boldsymbol{\beta}^X\right)\boldsymbol{M}^X\left(\boldsymbol{\alpha}^X-\boldsymbol{\beta}^X\right)^{\text{T}}\right\}\cdot\\ &\exp\left\{-\gamma^Z\left(\boldsymbol{\alpha}^Z-\boldsymbol{\beta}^Z\right)\boldsymbol{M}^Z\left(\boldsymbol{\alpha}^Z-\boldsymbol{\beta}^Z\right)^{\text{T}}\right\}=\kappa_{\text{MD}}^X\left(\boldsymbol{\alpha}^X,\boldsymbol{\beta}^X\right)\kappa_{\text{MD}}^Z\left(\boldsymbol{\alpha}^Z,\boldsymbol{\beta}^Z\right)\end{aligned} \tag{5}$$

In fact, (5) computes the distances of the numerical parts and the topological parts, respectively, and makes a weighted sum of them to yield a synthesized distance.

Replacing (4) with (5), (3) is changed into:

$$\hat{Y}(\boldsymbol{x},\boldsymbol{\zeta})=\frac{\sum Y_i^{\text{train}}\kappa\left(\boldsymbol{x},\boldsymbol{X}_i^{\text{train}},\boldsymbol{\zeta},\boldsymbol{T}_i^{\text{train}}\right)}{\sum\kappa\left(\boldsymbol{x},\boldsymbol{X}_i^{\text{train}},\boldsymbol{\zeta},\boldsymbol{T}_i^{\text{train}}\right)} \tag{6}$$

The model is able to yield a result considering topology changes. However, it is rough and inaccurate in real practice. The reason is that once we include topology changes, the model will be too simple and underfitting. In the next section, we construct it by several sub-models, and these sub-models share the same parameters. For systems that contain a large number of topology changes, the parameters are too few. We should add new parameters in order to shape more complex mappings. This is why we introduce ensemble learning to improve the model.

III. Advanced Model Considering Network Topology Change

In this section, we first reformulate the basic model. We will rebuild it with the idea of ensemble learning in Section III-A. In Section III-B, we will further explain how the model works. The detailed training procedure of the model will be proposed in Section IV.

A. Model Improvement via Ensemble Learning

In fact, the values of topological variables are limited in practice. In other words, the possible unique values (patterns) of $\boldsymbol{\alpha}^Z$ and $\boldsymbol{\beta}^Z$ in (5) are finite. Assume the number of the possible values is S, and the possible values are $\left\{T_s^{\text{uni}}|s=1,2,\ldots,S\right\}$, and the row numbers of the corresponding data in $\boldsymbol{X}^{\text{TRAIN}}$ and $\boldsymbol{T}^{\text{TRAIN}}$ are $\left\{\Omega_s|s=1,2,\ldots,S\right\}$.

That is to say, in the matrix $\boldsymbol{T}^{\text{TRAIN}}$, there are only S unique rows.

Then, the basic model (6) can be reformulated as:

$$\hat{Y}\left(\boldsymbol{x},\boldsymbol{\zeta}\right)=\sum_{s=1}^{S}P_s\left(\boldsymbol{x},\boldsymbol{\zeta}\right)\hat{Y}_s\left(\boldsymbol{x}\right)=\sum_{s=1}^{S}\left(P_s\left(\boldsymbol{x},\boldsymbol{\zeta}\right)\sum_{w\in\Omega_s}p_w^s\left(\boldsymbol{x}\right)Y_w^{\text{train}}\right) \tag{7}$$

$$P_s\left(\boldsymbol{x},\boldsymbol{\zeta}\right)=\frac{\kappa_{\text{MD}}^Z\left(\boldsymbol{\zeta},T_s^{\text{uni}}\right)\sum_{w\in\Omega_s}\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)}{\sum_{r=1}^{S}\left(\kappa_{\text{MD}}^Z\left(\boldsymbol{\zeta},T_r^{\text{uni}}\right)\sum_{w\in\Omega_r}\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)\right)} \tag{8}$$

$$\hat{Y}_s\left(\boldsymbol{x}\right)=\frac{\sum_{w\in\Omega_s}\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)Y_w^{\text{train}}}{\sum_{w\in\Omega_s}\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)}=\sum_{w\in\Omega_s}p_w^s\left(\boldsymbol{x}\right)Y_w^{\text{train}} \tag{9}$$

$$p_w^s\left(\boldsymbol{x}\right)=\frac{\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)}{\sum_{b\in\Omega_s}\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_b^{\text{train}}\right)} \tag{10}$$

$$\begin{cases}\sum_{w\in\Omega_s}p_w^s\left(\boldsymbol{x}\right)=1 \quad s=1,2,\ldots,S\\ \sum_{s=1}^{S}P_s\left(\boldsymbol{x},\boldsymbol{\zeta}\right)=1\end{cases} \tag{11}$$

Based on the above derivation, the basic model (6) is formulated through the following two steps.

Step 1: we construct several sub-models $\hat{Y}_s(\boldsymbol{x})$. Each $\hat{Y}_s(\boldsymbol{x})$ is established using the data in Ω_s (but without topological variables).

Step 2: we make a weighted sum of the estimated values of the sub-models. The weight $P_s\left(\boldsymbol{x},\boldsymbol{\zeta}\right)$ is given by (8).

In the light of this, the basic model (6) can be regarded as the ensemble of several sub-models, which coincides with the idea of ensemble learning. But still, there are two main discrepancies between the ensemble learning model and our basic model.

1) Firstly, the weights are invariable in ensemble learning, but are variable in basic model. In other words, with different inputs $\boldsymbol{x}$ and $\boldsymbol{\zeta}$, the weights take different values, and the model tends to favor different sub-models. These weights are related with topological and total numerical similarities (detailed explanations can be found in Section III-B), which makes the model able to recognize different network topologies, and thus intelligently weigh the sub-models.

2) Secondly, in the basic model (6), the sub-models share the same parameters, i.e., $\boldsymbol{M}^X$ and γ^X. However, in ensemble learning, the sub-models are usually trained to have different parameters.

We can modify the basic model through the second point. The idea is to use different kernels in the sub-models. In simpler terms, we do not use the same $\boldsymbol{M}^X$ and γ^X in all the sub-models, but train different parameters to improve the accuracy. Therefore, in the advanced model, the sub-models in (10) can be changed as:

$$\hat{Y}_s(\boldsymbol{x})=\frac{\sum_{w\in\Omega_s}\kappa_s\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)Y_w^{\text{train}}}{\sum_{w\in\Omega_s}\kappa_s\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)} \tag{12}$$

where the kernel κ_s is not fixed, and it changes with s.

The intuitive idea is to assign a different $\boldsymbol{M}^X$ and γ^X to each kernel. However, it is unpractical in real use. Firstly, although the topology changes (or the possible values of $\boldsymbol{\zeta}$) are finite, the number is still very large in real systems. So, the computation of off-line training and on-line estimating is so much that it can hardly be fulfilled. Secondly, many sub-models are constructed with very few data, thus causing the overfitting problem.

Therefore, we do not need to assign different $\boldsymbol{M}^X$ and γ^X to all the kernels. We merge similar topologies to formulate a "topology category". Finally, there are only I topology categories left, i.e., $\{\mathbb{T}\}=\left\{\mathbb{T}_l|l=1,2,\ldots,I\right\}$ $(I\le S)$.

The kernels in (12) can be expressed as:

$$\kappa_s\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)=\exp\left\{-\frac{\gamma(s)}{2}\left(\boldsymbol{x}-\boldsymbol{X}_w^{\text{train}}\right)\boldsymbol{M}(s)\left(\boldsymbol{x}-\boldsymbol{X}_w^{\text{train}}\right)^{\text{T}}\right\} \tag{13}$$

where $\boldsymbol{M}(s)$ and $\gamma(s)$ are functions that map s to the parameters. They are not injective mappings, and some sub-models share the same parameters.

By combining (7), (12) and (13), the advanced model is yielded.

B. Further Explanation of Advanced Model

In (8), the weight $P_s(\boldsymbol{x},\boldsymbol{\zeta})$ can be rewritten as:

$$P_s(\boldsymbol{x},\boldsymbol{\zeta})=\frac{P_s^Z(\boldsymbol{\zeta})P_s^X(\boldsymbol{x})}{\sum_{r=1}^{S}P_r^Z(\boldsymbol{\zeta})P_r^X(\boldsymbol{x})} \tag{14}$$

$$\begin{cases}P_s^Z(\boldsymbol{\zeta})=\dfrac{\kappa_{\text{MD}}^Z\left(\boldsymbol{\zeta},T_s^{\text{uni}}\right)}{\sum_{a=1}^{S}\kappa_{\text{MD}}^Z\left(\boldsymbol{\zeta},T_a^{\text{uni}}\right)}\\ P_s^X(\boldsymbol{x})=\dfrac{\sum_{w\in\Omega_s}\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_w^{\text{train}}\right)}{\sum_{b=1}^{N^{\text{train}}}\kappa_{\text{MD}}^X\left(\boldsymbol{x},\boldsymbol{X}_b^{\text{train}}\right)}\end{cases} \tag{15}$$

Therefore, the weight can be regarded as the production of the following two factors.

1) $P_s^Z(\boldsymbol{\zeta})$ represents the "topological similarity" of $\boldsymbol{\zeta}$ and T_s^{uni}.

2) $P_s^X(\boldsymbol{x})$ represents the "total numerical similarity" of $\boldsymbol{x}$ and Ω_s, i.e., the sum of the similarities of $\boldsymbol{x}$ and the numerical data in Ω_s.

If the topological similarity $P_s^Z(\boldsymbol{\zeta})$ is too small, or the total numerical similarity $P_s^X(\boldsymbol{x})$ is too small (due to lack of data or other reasons), the sub-model is not likely to be chosen in the process of determining CCT.

Figure 1 further explains how the model works. In Fig. 1, each row represents one type of network topology, and each topology corresponds to a weak learner. It is clearly shown that the model tends to give weak learner $\boldsymbol{E}_s$ a very high weight and give the other weak learners low weights. The reasons are listed as below.

1) The model does not favor weak learner $\boldsymbol{E}_r$. Although the input sample is topologically very similar to T_r^{uni}, the total numerical similarity between $\boldsymbol{x}$ and Ω_r is too low. This is usually caused by too few data with in Ω_r, so the sub-model will not be given a high weight in order to avoid overfitting problem.

2) The model does not favor weak learner $\boldsymbol{E}_S$. Although the numerical similarity between $\boldsymbol{x}$ and Ω_S is relatively high, $\boldsymbol{\zeta}$ is too different from T_S^{uni}. Thus, the sub-model is not capable of estimating the CCT under this topology.

3) The model does not favor weak learners $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$, since they are undesirable in terms of both $P_s^Z(\boldsymbol{\zeta})$ and $P_s^X(\boldsymbol{x})$.

4) The weak learner $\boldsymbol{E}_s$ is given the highest weight, since it is topologically very similar to the input data, and the data in Ω_s are enough to make a nice predictor.

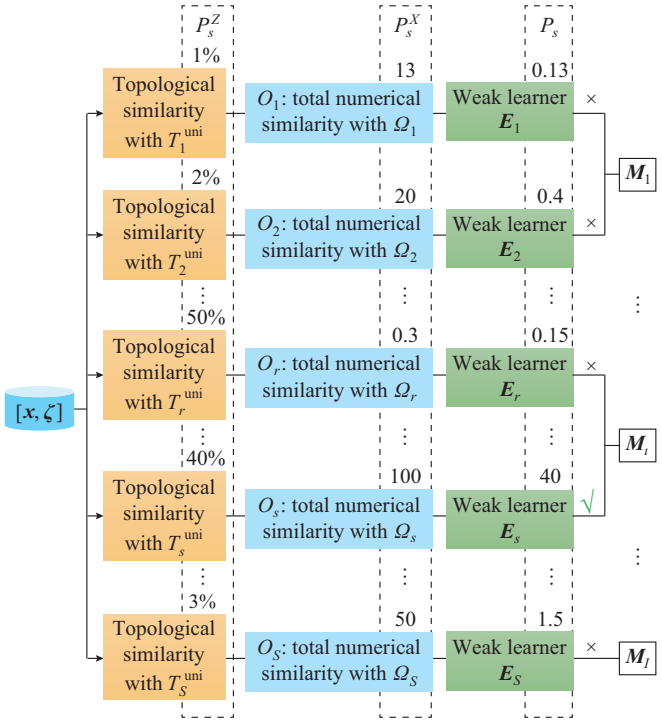

Fig. 1. Explanation of advanced model.

Another thing to notice is that the parameters are shared among the weak learners. For instance, in Fig. 1, the weak learners $\boldsymbol{E}_1$ and $\boldsymbol{E}_2$ share the same parameters. This ensures that the complexity of the model is controlled within a certain level.

The advantages of the advanced model can be concluded as follows.

1) The model is a strong learner based on several weak learners, which makes it more robust. Besides, it makes full

use of all the data, thus reduces the data requirement.

2) The decision-making process is clear and understandable.

3) The similarity between topologies give further information about the system. Topologies within the same category are similar in terms of transient stability, therefore further analysis can be launched based on the results.

IV. Model Training

In the advanced model, the parameters to be trained are the matrices $\boldsymbol{M}^Z$ and $\mathcal{M}=\{\boldsymbol{M}_\iota\}$, and the smoothing parameters γ^Z and $\Gamma=\{\gamma_\iota\}$.

Therefore, we decouple the training process into two parts: the training of the sub-models ($\mathcal{M}$ and Γ), and the training of the topology-related parameters ($\boldsymbol{M}^Z$ and γ^Z).

A. Training Algorithm for Sub-model

The training of the sub-models are actually that of $\mathcal{M}$ and Γ. In this step, the training procedure does not involve $\boldsymbol{M}^Z$ and γ^Z, so it is comparatively simple. The least mean square error loss is usually employed as the target function, and the stochastic gradient descent is adopted as the solving algorithm. There are various studies focusing on this area, so we do not repeat it in this article. Detailed training procedures can be found in [3], [25].

B. Overall Training Algorithm for Advanced Model

Another part is the training of $\boldsymbol{M}^Z$ and γ^Z. We train these parameters through the optimization of the weights $P_s(\boldsymbol{x},\boldsymbol{\zeta})$. However, as we can see from (14) and (15), the weights are also related with $\mathcal{M}$ and Γ. So, an alternating iterative training procedure is needed to train all the parameters. First, we fix the weights $P_s(\boldsymbol{x},\boldsymbol{\zeta})$ and train the sub-models. Then, we fix the sub-models and train the weights $P_s(\boldsymbol{x},\boldsymbol{\zeta})$, and the cycle goes on until the stopping criterion is met.

The detailed procedure of the overall training algorithm is listed in Algorithm 1.

The algorithm alternatively trains the parameters of the numerical part (step 15 to step 19) and the topological part (step 20). The "merging" steps are from step 9 to step 11, where the old kernels are shared when they satisfy the accuracy requirement.

In fact, the algorithm is enlightened by the gradient boost algorithm in ensemble learning [28]. The main difference is that we replace the line search step with the training of $\boldsymbol{M}^Z$ and γ^Z, and we embed the training of $\mathcal{M}$ and Γ into the algorithm.

The training of the overall model usually takes a long time, especially when the scale of the system is large. But once the inicial training is completed, incremental training can be applied and will not take much time. For incremental training, we just need to skip the initialization step (step 1). Detailed analysis of the computational cost for the training stage can be found in Section VI.

V. Practical Application

We design an overall flowchart of the proposed method in practical application, as shown in Fig. 2.

Algorithm 1 Training algorithm for advanced model

Require: $\boldsymbol{X}^{\text{TRAIN}}$, $\boldsymbol{T}^{\text{TRAIN}}$, $\boldsymbol{Y}^{\text{TRAIN}}$, ϵ_Z

Ensure: parameters for the numerical part $\mathcal{M}$, Γ; parameters for the topological part $\boldsymbol{M}^Z$, γ^Z; the mappings $\boldsymbol{M}(s)$ and $\gamma(s)$.

1: Initialize parameters:
$\mathcal{M}=\{\cdot\},\Gamma=\{\cdot\},I=0;\boldsymbol{M}^Z=\boldsymbol{M}_0^Z,\gamma^Z=\gamma_0^Z$.
2: for $e=1$ to N_E do
3: Update all the weights using (14): $\alpha_i=P_s\left(\boldsymbol{X}_i^{\text{train}},\boldsymbol{T}_i^{\text{train}}\right)$, $i\in\Omega_s$.
4: $f_{\text{stop}}=\text{true}$
5: for $s=1$ to S do
6: $f_Z=\text{true}$
7: for $\iota=1$ to I do
8: Assuming the sub-model adopts $\boldsymbol{M}_\iota$ and γ_ι as parameters, calculate its loss function:
$$L_\iota=\sum_{s=1}^{S}\sum_{i\in\Omega_s}\lambda_i\left(Y_i^{\text{train}}-\hat{Y}_s\left(\boldsymbol{X}_i^{\text{train}}\right)\right)^2$$
9: if $L_\iota<\epsilon_Z$ then
10: The sub-model does not need to set a new kernel:
$\boldsymbol{M}(s)=\boldsymbol{M}_\iota,\gamma(s)=\gamma_\iota,\ f_T=\text{false}$.
break
11: end if
12: end for
13: if f_T then
14: The sub-model needs to set a new kernel, which means that the training process should continue: $f_{\text{stop}}=\text{false}$.
15: Train the sub-model with loss function (16) and stochastic gradient descent (mentioned in Section IV-A), and we get $\boldsymbol{M}^*$ and γ^*. Note that λ_i is set to be constant in this step.
16: $I=I+1,\ \boldsymbol{M}_I=\boldsymbol{M}^*,\ \gamma_I=\gamma^*$
17: $\mathcal{M}=\mathcal{M}+\boldsymbol{M}_I,\ \Gamma=\Gamma+\gamma_I$
18: end if
19: end for
20: Adjust $\boldsymbol{M}^Z$: fix $\boldsymbol{M}(s)$, and train the weak learner in (7) using stochastic gradient descent, and we get $\boldsymbol{M}^{Z^*}$ and γ^{Z^*}.
21: $\boldsymbol{M}^Z=\boldsymbol{M}^{Z^*},\ \gamma^Z=\gamma^{Z^*}$
22: end for
23: return $\mathcal{M},\Gamma,\boldsymbol{M}^Z,\gamma^Z,\{\boldsymbol{M}(s)\},\{\gamma(s)\}$.

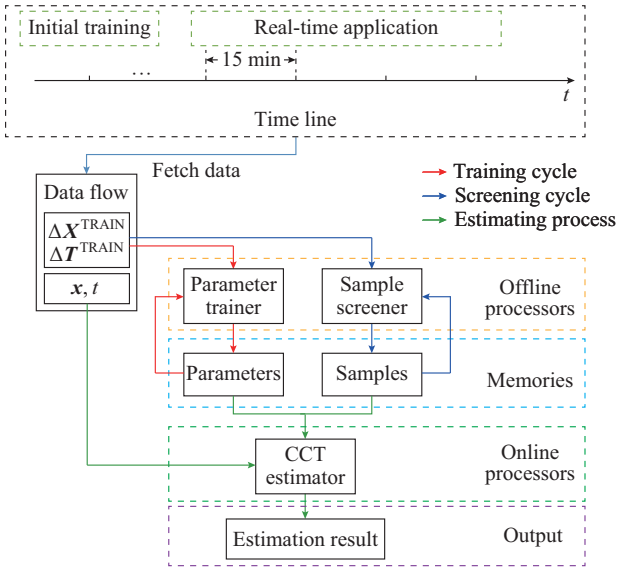

Fig. 2. Overall flowchart of proposed method.

In the training cycle, the incremental data (historical data for the initial training) are sent to the model trainer, and the parameters are refined and yielded through Algorithm 1.

In the screening cycle, we use certain algorithms to update and delete samples. This work is for maintaining the size of the training set, since the training set cannot grow

without limitation. The algorithms are what we are now working on.

In the estimation process, (12) is employed to obtain the final estimation result.

These three threads run simultaneously. The training cycle and the screening cycle run on offline processors, and the estimation process runs on the online processor.

VI. Case Study

We do experiments on two systems, and verify the effectiveness of our model, as elaborated behind.

A. IEEE 10-machine 39-bus System

The single line diagram of the IEEE 10-machine 39-bus (IEEE-10M39B) system is shown in Fig. 3.

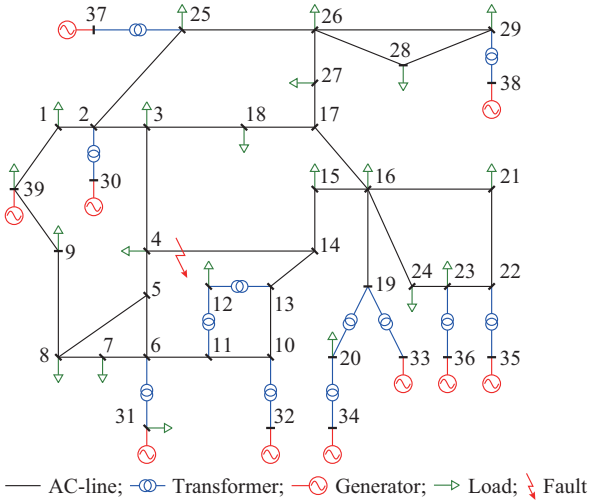

Fig. 3. Single line diagram of IEEE-10M39B system.

We apply a three-phase short-circuit fault on line 4-14 near bus 4. This fault is to show the case where the instability of the system is caused by the tripping-off of the tie-line interconnecting two areas. Governors, exciters, and stabilizers are applied to the generators.

There are 34 AC lines in the system. We consider all the $N-1$ and $N-2$ outages, and get 562 topologies in total. We randomly generate 100000 samples, and calculate their CCT. We use 80000 training samples and 20000 testing samples.

We employ the following data-driven models to make a comparison: ① artificial neural network (ANN); ② convolutional neural network (CNN); ③ least absolute shrinkage and selection operator (LASSO); ④ Bayesian linear regression (BLR); ⑤ classification and regression tree (CART); ⑥ the basic model proposed in Section II, namely MKR; ⑦ the advanced model proposed in Section III, namely "MKR+"; ⑧ the advanced model with input disturbances, namely "MKR+(d)".

We add some input disturbances to the model. For numerical variables, we add a Gaussian white noise. For each feature, the variance is set to be 0.02; for topological variables, we randomly choose 0.1% of the variables to take wrong values. This is to simulate the situation where there are measurement errors, and to prove the robustness of our method.

The frequency densities of the estimate error of CCT with these methods are shown in Fig. 4.

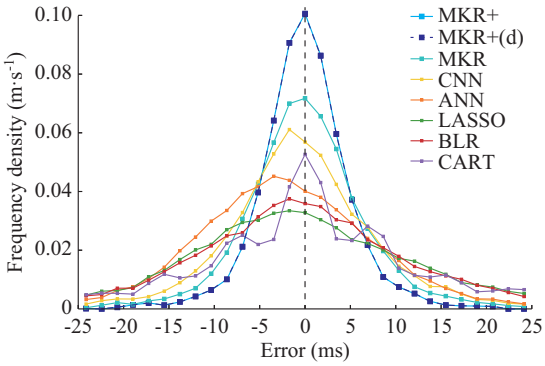

Fig. 4. Frequency density of estimate error under multiple data-driven methods in IEEE-10M39B system.

Further, some statistics are listed in Table II, including the mean error (ME), the root mean square error (RMSE), the mean error rate (MER) and the mean accuracy rate (MAR).

TABLE II
Statistical Indices of Estimate Error in IEEE-10M39B System

Method	ME (ms)	RMSE (ms)	MER (%)	MAR (%)
MKR+	3.66	5.00	1.32	98.68
MKR+(d)	3.67	5.02	1.33	98.67
MKR	5.47	8.39	1.97	98.03
CNN	6.80	9.85	2.42	97.58
ANN	7.97	10.35	2.92	97.08
LASSO	12.16	17.17	4.37	95.63
BLR	11.27	16.16	4.06	95.94
CART	13.48	21.20	4.80	95.20

We also compute some statistical indices of the absolute errors, as shown in Table III. These indices include the 1st, 2nd and 3rd quartiles of the absolute errors, and the index P, which is the proportion of the testing samples where the absolute error of our method is smaller than others. In other words, P can be regarded as the probability that our method is better than others.

TABLE III
Statistical Indices of Estimate Error and Probability Index in IEEE-10M39B System

Method	1st quartile (ms)	2nd quartile (ms)	3rd quartile (ms)	P (%)
MKR+	1.27	2.72	4.96	
MKR+(d)	1.27	2.73	4.97	
MKR	1.78	3.80	7.06	81.88
CNN	1.93	4.19	7.75	85.07
ANN	2.19	4.83	8.96	75.26
LASSO	3.07	6.50	11.20	74.34
BLR	3.99	8.74	15.96	83.37
CART	3.51	7.96	14.96	81.20

As shown in Table III, the advanced model MKR+ shows the best performance under all the indices. The performance

of the proposed method is significantly better than other methods in terms of these statistical indices. Besides, in the situation where there are input disturbances, the model still performs well, thus demonstrating the robustness of our model. In Fig. 4, the curves of MKR+ and MKR+ (d) are so close that it is hard to distinguish them.

Furthermore, we can excavate more information from the results. Through the procedure, we get 18 topology categories in total. The topology category of each $N-1$ outage is shown in Fig. 5.

The color of each line represents the topology category of the corresponding $N-1$ outage. The regularities are shown as follows.

1) The distribution of the categories shows a "regional effect". For example, the $N-1$ topologies corresponding to lines 16-24, l6-21, 21-22, 22-23, and 23-24 fall into the same category. This is because they formulate a loop, and tripping off each line within this loop will cause similar effects on the transient stability of the system.

2) Some topologies are treated specially. For example, the $N-1$ outage corresponding to line 16-19 does not share the same category with any other lines. The reason is that the tripping-off of this line will directly result in the loss of two generators, thus bringing a relatively big impact on the system.

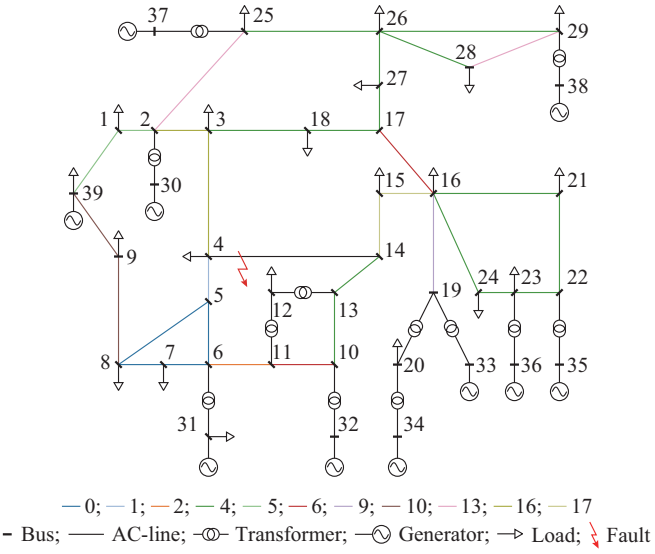

Fig. 5. Topology category of each $N-1$ outage.

For $N-2$ topologies, although they cannot be reflected in one single line diagram as for $N-1$ topologies, some critical information can be found in Fig. 6.

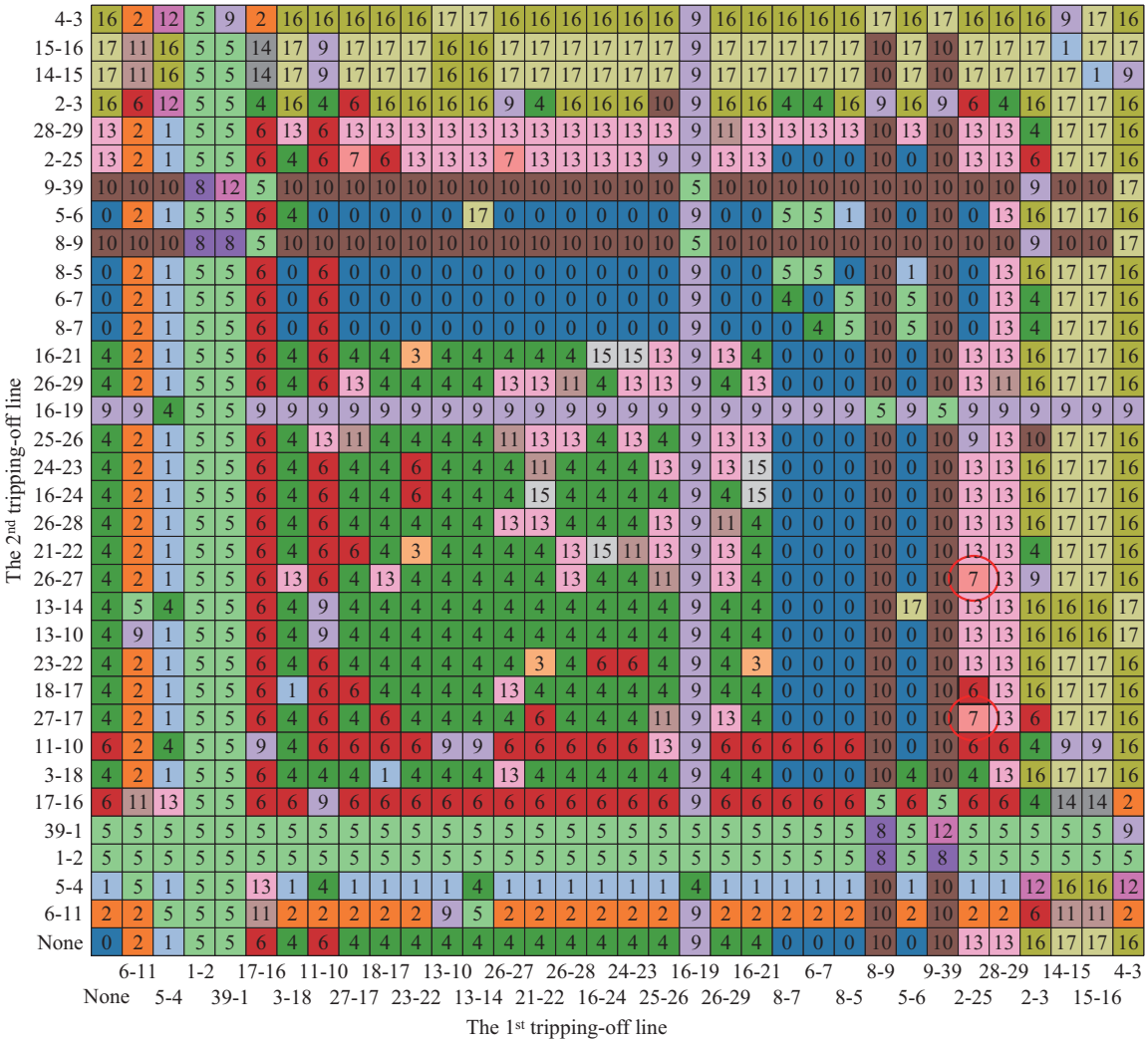

Fig. 6. Topology category of each $N-2$ outage.

The two axes represents two tripping-off lines in the $N-2$ topologies, respectively. In fact, the figure also reflects situations under $N-1$ topologies.

As we can see, the regularities are also obvious.

1) In some $N-2$ outage changes, there exists a "dominant line". With this line tripped off, the influence of another trip-off line is insignificant. For example, in the 3rd and 4th column, there is a "main color", suggesting that the tripping-off of line 1-2 or line 39-1 is of greater importance in the $N-2$ outages corresponding to them.

2) The two red-circled topologies are classified to the same topology category. This is because they actually refer to the same interface, as shown in Fig. 7. In this case, our model precisely identifies the "similar topologies". The result is also supported by intuitive knowledges.

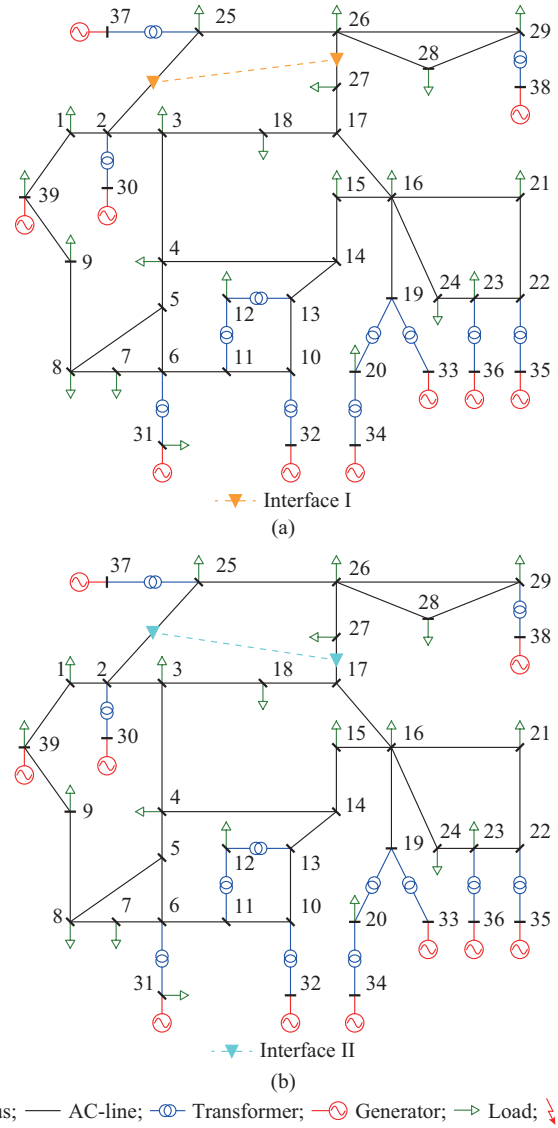

Fig. 7. Two $N-2$ topologies with same topology category refer to same interface. (a) Interface I. (b) Interface II.

As we can see, the model can automatically explore some useful conclusions, which is helpful in the further analysis.

B. A Real System

We also test our model on a real system. The skeleton of the real system is shown in Fig. 8.

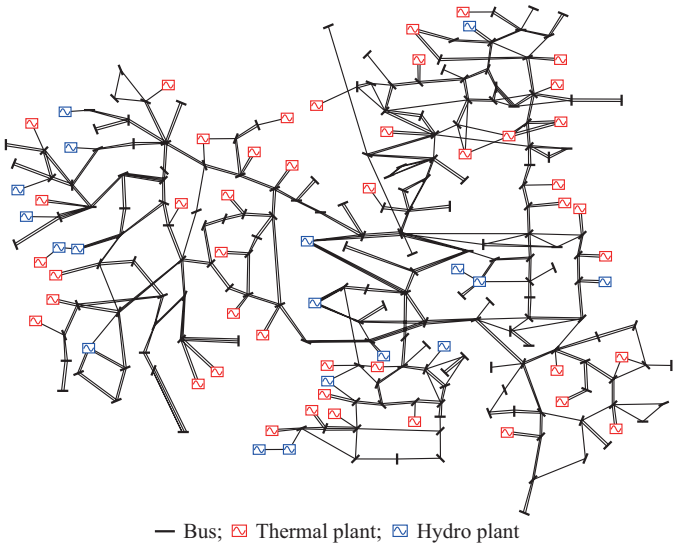

Fig. 8. Skeleton of real system.

The system consists of 6 areas which are interconnected with AC lines and DC lines. For confidentiality, we do not show the names of the nodes, and we apply some transformations to the geometrical shape.

Approximately, the system contains 3000 stations, 15000 AC lines, 6300 transformers, 4000 generators, and 25000 loads. The total capacity of the system is about 1000 GW. We collected about 13000 samples (data of half a year) to make an analysis. We set three-phase short-circuit faults on 4 main interfaces to calculate the CCT. For topology changes, we only take the main elements into consideration in Fig. 8. There are about 450 topologies in the original data.

We do experiments under the following two situations.

Situation 1: we train and test our model using the original 13000 samples, 3000 of which are used as the testing set.

Situation 2: we trip off one pair of lines in the original samples for five times to generate 65000 more samples (80000 samples in total), 20000 of which are used as the testing set.

The samples in situation 1 contain relatively small topology changes, whereas the samples in situation 2 contain large topology changes. This is to investigate how the performances of the models will change under large topology changes. The error distribution and the statistical indices are shown in Fig. 9, Tables IV, and V.

It shows that with relatively large topology changes, our model still remains a high level of accuracy (although a little decline compared with small topology changes), whereas some other models suffer from a significant degradation. Statistically, our model still shows a significantly better performance than other methods and a nice robustness against input disturbances.

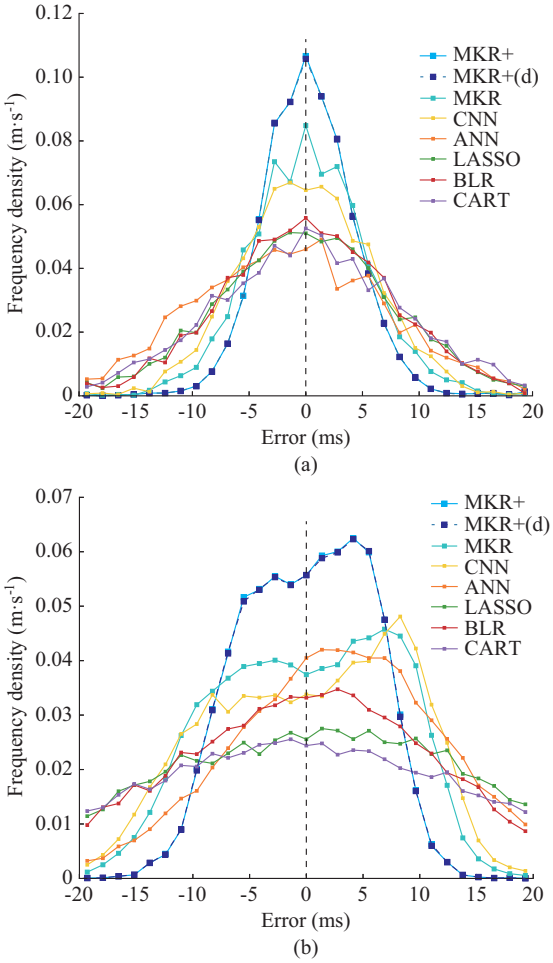

Fig. 9. Frequency densities of estimate error under multiple data-driven methods in real system. (a) Considering small topological changes. (b) Considering $N-2$ topological changes.

TABLE IV
STATISTICAL INDICES OF ESTIMATE ERROR IN REAL SYSTEM

Method	ME (ms)		RMSE (ms)		MER (%)		MAR (%)	
	Situation 1	Situation 2	Situation 1	Situation 2	Situation 1	Situation 2	Situation 1	Situation 2
MKR+	3.34	4.67	4.78	5.59	1.23	1.50	98.77	98.50
MKR+(d)	3.34	4.68	4.79	5.60	1.23	1.51	98.77	98.49
MKR	4.48	6.55	6.66	7.78	1.75	2.13	98.25	97.87
CNN	4.97	7.35	6.95	8.76	1.90	2.40	98.10	97.60
ANN	7.10	8.26	8.99	15.39	2.48	2.36	97.52	97.64
LASSO	6.74	11.61	9.15	15.05	4.09	4.59	95.91	95.41
BLR	5.99	9.21	7.56	11.39	2.02	2.75	97.98	97.25
CART	7.06	13.16	9.44	16.82	2.82	4.36	97.18	95.64

C. Computation Cost

We divide the computation cost into two parts: the cost at the training stage and the cost at the estimating stage. The following results are worked out on a workstation with 8 cores, each with a 3.50 GHz CPU (Intel Xeon CPU E5-2637). Python and TensorFlow are used for coding.

1) Training Efficiency

The training time depends on the system dimension. For the real system, the feature number is about 50000, and the convergence process is shown in Fig. 10.

TABLE V
STATISTICAL INDICES OF ESTIMATE ERROR AND PROBABILITY INDEX IN REAL SYSTEM

Method	1st quartile (ms)		2nd quartile (ms)		3rd quartile (ms)		P (%)	
	Situation 1	Situation 2	Situation 1	Situation 2	Situation 1	Situation 2	Situation 1	Situation 2
MKR+	1.18	2.22	2.60	4.37	4.46	6.67		
MKR+(d)	1.18	2.23	2.59	4.38	4.47	6.68		
MKR	1.68	3.23	3.42	6.24	5.82	9.35	73.91	83.42
CNN	1.90	3.67	3.90	7.08	6.61	10.34	78.69	88.01
ANN	2.76	3.20	5.99	6.79	10.21	11.53	82.84	76.64
LASSO	2.47	4.85	5.28	10.01	9.26	16.22	80.17	90.69
BLR	2.42	3.77	5.02	7.98	8.51	13.57	79.44	83.95
CART	2.58	5.12	5.67	10.98	9.85	18.73	81.34	88.30

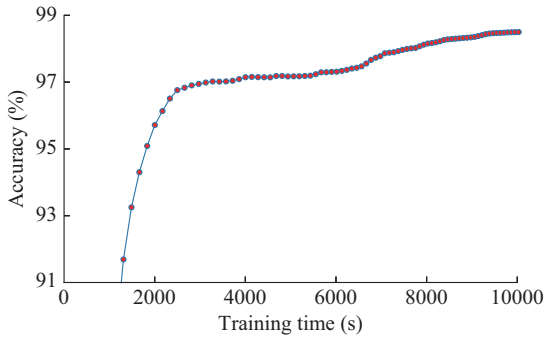

Fig. 10. Convergence process of real system.

In real practice, such a training time is enough to satisfy the real need of a large power system. In fact, since we use mini-batch algorithm in the training process and there are a large number of matrix operations, the training process can be accelerated by parallel computation, and the training time can be decreased to wihtin 1 h. Besides, this is the cost for the initial training. Once the initial training is finished, the cost for the incremental training will be much smaller.

2) Estimation Efficiency

The computation of (12) can be transformed into a series of matrix operations, so the estimation can be accelerated in most cases. Our test results show that the estimation time for each scenario is 0.5 to 0.6 s, which is enough for online usage.

VII. CONCLUSION

In this paper, we proposed a data-driven model to fulfill TSA considering network topology changes. The model is based on MKR and ensemble learning. We include the topological variables in the kernel regression, and the model is constructed by several sub-models. The weights are determined by both the topological and total numerical similarities. We use ensemble learning to rebuild the model, thus the data under different topologies are utilized effectively. The model is advanced in the following aspects.

1) It is able to efficiently exploit the information within

the data of all the topologies, thus the model is robust and could make full use of the data.

2) The decision-making procedure of the model is clear and understandable.

3) The results of the model are informative and can support further researches.

The model is tested on two systems, and the experiments demonstrate its effectiveness and practicability.

Still there are some practical limitations for our method. For example, as is discussed in Section V, the training set cannot expand infinitely, so the strategies for sample update and deletion are needed. Besides, experiments show that when the number of topology changes is large, the converging speed of the proposed algorithm is slow, and the model performance decreases. In fact, all sorts of data-driven methods encounter with these issues; they need larger data with more information. In this paper, we proposed a refined model which is able to dig out more information from the data, thus decreasing the data requirement. However, there is still room for improvement.

Further work includes efficient data generation techniques linked to the proposed model, and a model capable of larger number of topology changes.

REFERENCES

[1] Y. Zhang and L. Xie, "A transient stability assessment framework in power electronic-interfaced distribution systems," *IEEE Transactions on Power Systems*, vol. 31, no. 6, pp. 5106-5114, Feb. 2016.

[2] Z. Li, J. Wang, H. Sun *et al.*, "Transmission contingency screening considering impacts of distribution grids," *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1659-1660, Mar. 2015.

[3] S. Abhyankar, G. Geng, M. Anitescu *et al.*, "Solution techniques for transient stability-constrained optimal power flow – part I," *IET Generation, Transmission & Distribution*, vol. 11, no. 12, pp. 3177-3185, Sept. 2017.

[4] A. Shamisa and M. Karrari, "Model free graphical index for transient stability limit based on on-line single machine equivalent system identification," *IET Generation Transmission & Distribution*, vol. 11, no. 2, pp. 314-321, Jan. 2017.

[5] Y. Kato and S. Iwamoto, "Transient stability preventive control for stable operating condition with desired CCT," *IEEE Transactions on Power Systems*, vol. 17, no. 4, pp. 1154-1161, Nov. 2002.

[6] N. Yorino, A. Priyadi, H. Kakui *et al.*, "A new method for obtaining critical clearing time for transient stability," *IEEE Transactions on Power Systems*, vol. 25, no. 3, pp. 1620-1626, Feb. 2010.

[7] P. Bhui and N. Senroy, "Real-time prediction and control of transient stability using transient energy function," *IEEE Transactions on Power Systems*, vol. 32, no. 2, pp. 923-934, May 2016.

[8] Y. Zhou, W. Hu, Y. Min *et al.*, "Active splitting strategy searching approach based on misocp with consideration of power island stability," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 3, pp. 475-490, May 2019.

[9] G. Geng, S. Abhyankar, X. Wang *et al.*, "Solution techniques for transient stability-constrained optimal power flow – part II," *IET Generation, Transmission & Distribution*, vol. 11, no. 12, pp. 3186-3193, Sept. 2017.

[10] H. Song and M. Kezunovic, "Stability control using PEBS method and analytical sensitivity of the transient energy margin," in *Proceedings of IEEE PES Power Systems Conference and Exposition (PSCE)*, New York, USA, Oct. 2004, pp. 1-6.

[11] V. Chadalavada and V. Vittai, "Transient stability assessment for network topology changes: application of energy margin analytical sensitivity," *IEEE Transactions on Power Systems*, vol. 9, no. 3, pp. 1658-1664, Aug. 1994.

[12] C. Liu, K. Sun, Z. H. Rather *et al.*, "A systematic approach for dynamic security assessment and the corresponding preventive control scheme based on decision trees," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 717-730, Oct. 2013.

[13] W. Hu, Z. Lu, S. Wu *et al.*, "Real-time transient stability assessment in power system based on improved SVM," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 1, pp. 26-37, Jan. 2019.

[14] Y. Liu, J. Zhao, L. Xu *et al.*, "Online TTC estimation using nonparametric analytics considering wind power integration," *IEEE Transactions on Power Systems*, vol. 34, no. 1, pp. 494-505, Aug. 2018.

[15] J. Lv, M. Pawlak, and U. D. Annakkage, "Prediction of the transient stability boundary using the LASSO," *IEEE Transactions on Power Systems*, vol. 28, no. 1, pp. 281-288, Jun. 2012.

[16] S. Wu, L. Zheng, W. Hu *et al.*, "Improved deep belief network and model interpretation method for power system transient stability assessment," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 1, pp. 27-37, Nov. 2019.

[17] J. Lv, M. Pawlak, and U. Annakkage, "Prediction of the transient stability boundary based on nonparametric additive modeling," *IEEE Transactions on Power Systems*, vol. 32, no. 6, pp. 4362-4369, Feb. 2017.

[18] D. H. Choi and L. Xie, "Impact of power system network topology errors on real-time locational marginal price," *Journal of Modern Power Systems and Clean Energy*, vol. 5, no. 5, pp. 797-809, Sept. 2017.

[19] J. Shu, W. Xue, and W. Zheng, "A parallel transient stability simulation for power systems," *IEEE Transactions on Power Systems*, vol. 20, no. 4, pp. 1709-1717, Oct. 2005.

[20] C. Ren and Y. Xu, "Transfer learning-based power system online dynamic security assessment: using one model to assess many unlearned faults," *IEEE Transactions on Power Systems*, vol. 35, no. 1, pp. 821-824, Oct. 2019.

[21] R. Zhang, J. Wu, B. Li *et al.*, "Self-adaptive power system transient stability prediction based on transfer learning," *Power System Technology*, vol. 44, no. 6, pp. 2196-2205, Nov. 2019.

[22] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge: MIT press, 2012.

[23] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer Science & Business Media, 2009.

[24] T. Liu, Y. Liu, L. Xu *et al.*, "Non-parametric statistics-based predictor enabling online transient stability assessment," *IET Generation, Transmission & Distribution*, vol. 12, no. 21, pp. 5761-5769, Nov. 2018.

[25] X. Liu, Y. Min, L. Chen *et al.*, "Data-driven transient stability assessment based on kernel regression and distance metric learning," *Journal of Modern Power Systems and Clean Energy*, 2020. doi: 10.35833/MPCE.2019.000581.

[26] I. B. Sulistiawati, A. Priyadi, O. A. Qudsi *et al.*, "Critical clearing time prediction within various loads for transient stability assessment by means of the extreme learning machine method," *International Journal of Electrical Power & Energy Systems*, vol. 77, pp. 345-352, May 2016.

[27] Y. Li and Z. Yang, "Application of EOS-ELM with binary Jaya-based feature selection to real-time transient stability assessment using PMU data," *IEEE Access*, vol. 5, pp. 23092-23101, Oct. 2017.

[28] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, p. 21, Dec. 2013.

Xianzhuang Liu received the B.S. and Ph.D. degrees in electrical engineering from Tsinghua University, Beijing, China, in 2014 and 2020, respectively. He is currently working at the State Grid Corporation of China, Beijing, China. His research interests include transient stability analysis and data-driven techniques in power grid.

Xiaohua Zhang received the B.S. and M.S. degrees in electrical engineering from Nanchang University, Nanchang, China, in 1984 and 2005, respectively. He is now a Professor Level Senior Engineer. He is currently working at the State Grid Jibei Electric power Company Limited, Beijing, China. His research interests include operation and scheduling in large scale power grid.

Lei Chen received the B. S. and Ph. D. degrees in electrical engineering from Tsinghua University, Beijing, China, in 2003 and 2008, respectively. Since 2008, he has been with the Department of Electrical Engineering, Tsinghua University, where he is now an Associate Professor. He was a Recipient of the Excellent Young Scientists Fund of National Natural Science Foundation of China in 2019. His research interests include dynamic analysis and control of power systems.

Fei Xu received the B.S. and Ph.D. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1999 and 2015, respectively. Since

1999, he has been with the Department of Electrical Engineering, Tsinghua University, where he is now an Associate Research Fellow. His research interests include integrated energy system and power system stability control.

Changyou Feng received the B.S. degree in material science and engineering and the Ph.D. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2004 and 2010, respectively. He is currently working at the State Grid Corporation of China, Beijing, China. His main research interests include power market and power system reliability.