Judgment Prediction Based on Tensor Decomposition With Optimized Neural Networks

Xiaoding Guo, Lei Zhang, and Zhihong Tian, *Senior Member, IEEE*

***Abstract*— In the field of smart justice, handling legal cases through artificial intelligence technology is a research hotspot. Traditional judgment prediction methods are mainly based on feature models and classification algorithms. The former is difficult to describe cases from multiple angles and capture the correlation information between different case modules, while requires a wealth of legal expertise and manual labeling. The latter is unable to accurately extract the most useful information from case documents and produce fine-grained predictions. This article proposes a judgment prediction method based on tensor decomposition with optimized neural networks, which consists of OTenr, GTend, and RnEla. OTenr represents cases as normalized tensors. GTend decomposes normalized tensors into core tensors using the guidance tensor. RnEla intervenes in a case modeling process in GTend by optimizing the guidance tensor, so that core tensors represent tensor structural and elemental information, which is most conducive to improving the accuracy of judgment prediction. RnEla consists of the similarity correlation Bi-LSTM and optimized Elastic-Net regression. RnEla takes the similarity between cases as an important factor for judgment prediction. Experimental results on real legal case dataset show that the accuracy of our method is higher than that of the previous judgment prediction methods.**

***Index Terms*— Judgment prediction, legal cases, neural networks, tensor decomposition.**

I. Introduction

IN THE field of intelligent justice, the use of artificial intelligence technology to assist judicial organs in handling cases is a current research hotspot. Judgment prediction algorithms provide judges with decision recommendations, such as fines or sentencing. When judgment results deviate significantly from similar cases, for example, one case was sentenced to six months in prison and a $2000 fine, but similar cases were sentenced to more than five years in prison, judgment prediction algorithms give warnings and corrections. Judgment prediction algorithms can assist judges in judging cases and speed up processes of law enforcement and case handling. The misjudgment risk and occurrence of unjust cases are reduced, which is of great significance to promoting the legalization of the whole society.

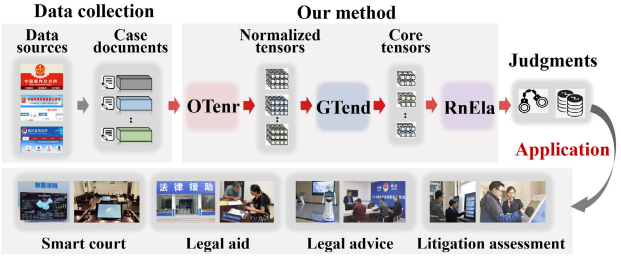

Fig. 1. Composition and applications of our method.

The focus of judgment prediction methods is to achieve a unified and standardized case representation and to accurately predict judgment results. Previous judgment prediction methods are mainly based on feature models and classification algorithms. Feature models rely on legal expertise and manual annotation, which requires a lot of manpower resources and leads to dimensional explosion. Feature models are difficult to describe cases in a multilevel and fine-grained manner. Judgment results predicted by classification algorithms are coarse in granularity, which is unqualified for needs of practical applications. Traditional judgment prediction methods cannot establish the relationship between case modeling and judgment prediction processes while ignoring the background and similarity information between cases.

This article proposes a method based on tensor decomposition with optimized neural networks for the standardized representation of legal cases and accurate judgment prediction. Fig. 1 shows the composition and applications of our method, which consists of OTenr, GTend, and RnEla. RnEla achieves judgment prediction using the similarity correlation Bi-LSTM and optimized Elastic-Net regression. Our method effectively addresses challenges faced by current judgment prediction services. Our method is applicable to many justice scenarios, such as smart court, legal aid, legal advice, and litigation assessment.

OTenr is based on tensor models. OTenr describes cases as normalized tensors while taking into account character backgrounds and special vocabulary. Under the action of

Manuscript received 23 December 2021; revised 21 July 2022 and 10 January 2023; accepted 20 February 2023. Date of publication 7 March 2023; date of current version 6 August 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021YFB2012402, in part by the Key Project of National Natural Science Foundation under Grant U20B2046, and in part by the Key Research Projects of Henan Higher Education Institutions under Grant 23A520026. *(Corresponding author: Zhihong Tian.)*

Xiaoding Guo and Lei Zhang are with the School of Computer and Information Engineering and the Henan Key Laboratory of Big Data Analysis and Processing, Henan University, Kaifeng 450001, China.

Zhihong Tian is with the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China (e-mail: tianzhihong@gzhu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2023.3248275, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2023.3248275

the guidance tensor, GTend converts normalized tensors into tensors of smaller dimension. The guidance tensor captures the relationship between case modeling and judgment prediction processes. The latter intervenes in tensor decomposition by optimizing the guidance tensor, so that core tensors carry the most effective information for judgment prediction. RnEla combines the similarity correlation Bi-LSTM and optimized Elastic-Net regression. RnEla treats the similarity between cases as a determinant for judgment prediction. RnEla improves classification and regression algorithms using similarity gates and guidance tensor optimization layers. RnEla coordinates prediction results obtained by classification and regression algorithms using judgment weights and fuzzy boundaries.

Our method can effectively achieve the unified and standardized representation of cases while accurately predicting judgments. Core tensors derived by GTend greatly reduce the dimensionality of normalized tensors obtained by OTenr. Core tensors remove inaccurate, redundant, and meaningless information from normalized tensors, while reducing the interference of noise information on RnEla, which decreases the computational complexity of RnEla. Furthermore, RnEla intervenes in GTend by optimizing the guidance tensor, so that core tensors carry information, which is most beneficial to improve the accuracy of judgment prediction algorithms. GTend greatly improves the computation speed and prediction accuracy of judgment prediction methods. GTend provides technical support for establishing the relationship between case modeling and prediction algorithms.

The key of our method is the construction of normalized tensors and the initialization and iterative of the guidance tensor. This article sets up multiple method strategies and compares our method with various machine learning and deep neural network algorithms. Under different initial values of normalized tensors and the guide tensor, the prediction accuracy and convergence time of our method and the comparison algorithms are given. This article comprehensively analyzes the roles of OTenr, GTend, and RnEla and further reflects the advantages of our method over previous methods.

The main contributions of this article are as follows.

1) This article proposes a case representation method OTenr, which is based on tensor models and represents cases as unified, standardized tensors. Considering special vocabulary and party background in cases, OTenr describes cases as normalized tensors using multiple perspectives. OTenr automatically obtains important features of cases and is less dependent on manual tagging, thus greatly avoiding the occurrence of data sparsity.
2) In response to challenges faced by auxiliary judgment services, this article proposes the guided tensor decomposition in order to establish the correlation between case modeling and judgment prediction processes. With the intervention of the guidance tensor, GTend transforms normalized tensors into core tensors, which have smaller dimensions and carry the most favorable information for judgment prediction.
3) This article proposes a prediction algorithm RnEla, which constructs similarity gates and new regularization terms in order to determine judgments while considering the similarity between legal cases. RnEla intervenes in case modeling process by optimizing the guidance tensor, so that obtained core tensors can be interpreted as the most effective information for judgment prediction.

Section II provides research status of judgment prediction algorithms. Section III gives an introduction of our method. Section IV provides experimental results and analysis. Section V gives the conclusion.

II. Previous Work

Previous judgment prediction methods mainly focus on case modeling and prediction algorithms. The former is based on feature models [1], [2], [3], [4], [5], [6], [7], while the latter is based on classification models, such as machine learning and deep neural networks [8], [9], [10], [11], [12], [13]. Xu et al. [14] use graph neural networks to distinguish easily confused cases, while extracting striking features through the attention mechanism. Zhong et al. [15] propose an alternate question answering model, which obtains facts of cases using the reinforcement learning model. Branting et al. [16] highlight important features according to attention weights, which enhances the interpretability of prediction algorithms. Bibal et al. [17] propose the limitations of judgment prediction models. Yang et al. [18] obtain the relationship between cases using recurrent attention networks.

Long et al. [19] obtain the interactive information among case facts, litigation claims, and laws using inference rules, in order to predict judgments of cases. Kumar et al. [20] apply expert knowledge to the construction of case elements, while predicting judgments through the Naive Bayes method. Das et al. [21] predict case charges and verdicts through machine learning algorithms. Kadar et al. [22] propose a prediction method for sparse and unbalanced data, which uses spatiotemporal data and machine learning models to predict judgments. Agrawal et al. [23] propose a judgment prediction method, which combines manual judgment and prediction algorithm. Expert experience is used to assist prediction algorithms to predict judgments.

Yang et al. [24] obtain the topological dependencies among subtasks, such as prediction of legal provisions, crimes, and judgments using the Bi-feedback network based on attention mechanisms. Yi et al. [25] apply continuous conditional random field to the calculation of similarity between multiregion cases, while predicting judgments using neural networks. Chen et al. [26] filter out fine-grained case features using the aggregated deep gating network. Combined with case charges, neural networks are used to predict judgments. Chen et al. [27] calculate correlation between words using dynamic word embedding methods in order to solve the problem of imbalance in judicial data. Hu et al. [28] use dual-robust reinforcement learning network to reduce the influence of noise on judgment prediction.

Previous methods have natural flaws of feature models: 1) a lot of legal expertise and manual annotation is required; 2) data explosion is easy to occur; and 3) case is difficult to describe in a fine-grained manner at multiple levels [29], [30],

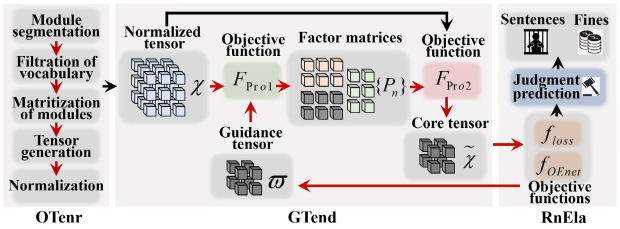

Fig. 2. Framework of our method.

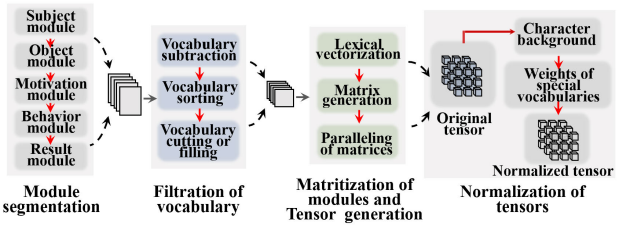

Fig. 3. Method composition of OTenr.

[31], [32], [33]. Classification algorithms cannot obtain fine-grained judgment prediction results, which are not suitable for actual needs [34], [35], [36]. Previous methods are difficult to establish the connection between case modeling and prediction processes [37], while ignoring similarity information between cases, which is not conducive to the improvement of method performance.

III. Our Approach

Our approach aims to solve problems of auxiliary judgment services, including unified and standardized representation of cases, correlation between case modeling and judgment prediction processes, and application of similarities between cases in judgment prediction. As is shown in Fig. 2, our method is divided into the following steps.

1) *Description of Legal Cases:* According to background information of the parties and weights of special vocabularies, OTenr describes legal cases as normalized tensors using tensor models.
2) *Construction of Guidance Tensor Decomposition:* GTend figures out core tensors with smaller dimensions using normalized tensors and the guidance tensor. Core tensors remove redundant and meaningless information from normalized tensors. GTend provides technical support for the intervention of prediction algorithms on case modeling.
3) *Accurate Prediction of Judgments:* RnEla combines the similarity correlation Bi-LSTM and optimized Elastic-Net regression. RnEla intervenes in a case modeling process by optimizing guidance tensor, so that core tensors represent the most effective information for judgment prediction.

A. Preliminaries

This article introduces tensor decomposition to solve problems faced by auxiliary judgment services. This section presents calculation rules related to our method. I is used to represent the unit matrix. For matrix A, $A \in \mathbb{R}^{I\times J}$, A_{ij} represents the element of A in position (i, j). Trace(A) is the trace norm of matrix A, and $\|A\|_F$ is the Frobenius norm of matrix A, so as tensors and vectors [38].

Definition 1 (n-Mode Expansion): The n-mode matrix expansion of a tensor is to expanded its elements orderly in the nth dimension. Given a tensor χ, $\chi \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $n \in [1, N]$, the n-mode matrix expansion of χ is $\chi_{(n)}$, $\chi_{(n)} \in \mathbb{R}^{(I_1 I_2,\ldots,I_{n-1}I_{n+1},\ldots,I_N)\times I_n}$. The (p, i_n)th element of $\chi_{(n)}$ is $\chi_{i_1,i_2,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N}$, where p is the order of $(i_1, i_2, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N)$. In the same manner, χ_{vec} represents the vector expansion of χ.

Definition 2 (n-Mode Vector Product): Given a tensor χ and a vector a, $\chi \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $a \in \mathbb{R}^{I_n}$, $n \in [1, N]$, the n-mode product of χ and a is ν. $\nu \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_{n-1}\times I_{n+1}\times\cdots\times I_N}$. $\nu_{i_1,i_2,\ldots,i_{n-1},i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} \chi_{i_1,i_2,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} a_{i_n}$. $\nu = \chi \times_n a$.

Definition 3 (n-Mode Matrix Product): Given a tensor χ and a matrix A, $\chi \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $A \in \mathbb{R}^{I_n\times J_n}$, $n \in [1, N]$, the n-mode product of χ and A is ν, $\nu \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_{n-1}\times J_n\times I_{n+1}\times\cdots\times I_N}$. $\nu_{i_1,i_2,\ldots,i_{n-1},j_n,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} \chi_{i_1,i_2,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} A_{i_n,j_n}$. $\nu = \chi \times_n A$.

B. OTenr

With the generation of large amounts of judicial data from judicial devices, the unified and standardized representation of cases is crucial. Previous methods represent cases as 2-D matrices, which cannot describe cases in a multilevel and fine-grained manner, and exist natural defect of feature models. OTenr is a modeling algorithm based on tensor concept. As shown in Fig. 3, according to background information of the parties and weights of special vocabularies, OTenr represents cases in 3-D normalized tensors. OTenr consists of the following parts: 1) module segmentation; 2) filtration of vocabulary; 3) matritization of modules; 4) tensor generation; and 5) normalization of tensors.

Module segmentation refers to segmenting cases into different components. Cases break into subject, object, motivation, behavior, and result modules. Each module matches a specific ingredient. Filtration of vocabulary aims to select important words from case modules. Since tensor representation of cases has the same dimensions, each case module should have the same number of words. Filtration of vocabulary is divided into the following steps.

1) *Vocabulary Subtraction:* Build lists of specific words and filter vocabularies in each module, which removes meaningless words while retaining professional information.
2) *Vocabulary Sorting:* Sort words in case modules according to order, frequency, and weights. In legal case documents, words that appear repeatedly or firstly are more important. TF-IDF is used to select representative words.
3) *Vocabulary Cutting or Filling:* Case modules should contain the same number of words. Cut or fill words in order to make each module the same length.

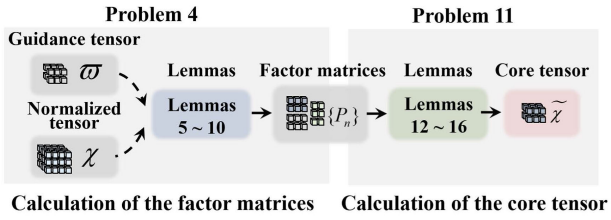

Fig. 4. Method composition of GTend.

Matritization of modules aims to convert case modules into matrices with the same dimension. Words in case modules are represented as vectors using word2vec models. Matrices represent case modules are united into a 3-D tensor through tensor generation. Normalization of tensors aims to mapping background of parties and weights of special vocabularies into normalized tensors, which are more suitable for specific judgment scenes. Tensor normalization consists of party background fusion and special vocabularies fusion. For similar cases, different types of persons produce different judgments. The background vector is set to weaken or enhance vocabulary weights in behavior and motivation modules. Vocabulary weights are affected by many factors, such as frequency, TF-IDF values, and occurrence order of words. The weight matrix is set for special vocabularies.

C. GTend

In order to accurately predict judgments, establishing correlation between case modeling and judgment prediction is crucial. This article presents the guidance decomposition method GTend, which provides technical support for the establishment of connection between case modeling and judgment prediction. As shown in Fig. 4, GTend decomposes normalized tensors into core tensors under the action of the guidance tensor. Core tensors approach both the guidance tensor and normalized tensors in elemental and structural terms.

RnEla intervenes case modeling in GTend through optimizing the guidance tensor. Core tensors are regarded as the structural and elemental information that is most beneficial to improve the judgment prediction accuracy of RnEla. GTend is consists of two parts: 1) approximate the normalized tensor with the guidance tensor and 2) approximate the core tensor with the normalized tensor. The former maps normalized tensors into high-dimensional space represented by the guidance tensor, and the latter combines resulting mapping information and normalized tensors in order to figure out core tensors. The guidance tensor is the bridge between case modeling and decision prediction. The construction and iterative optimization of the guidance tensor are crucial to GTend.

In practical applications, the guidance tensor is regarded as hyperparameters and randomly initialized. Several basic principles are provided for the initialization of the guidance tensor: 1) element values are avoided to be the same, in order to prevent model degradation caused by the same partial derivatives of elements during backward propagation of neural networks; 2) elements are initialized to small random numbers close to 0, which break the symmetry of neural networks, such as random numbers sampled from the multidimensional Gaussian or uniform distributions; 3) variance normalization, Xavier, and He initialization are used to help the initialization of the guidance tensor; and 4) multiple linear dependencies in the guidance tensor need to be removed. Multicollinearity can be eliminated through the principal component analysis.

1) Approximate the Normalized Tensor With the Guidance Tensor: The normalized tensor is approached to the guidance tensor through factor matrices. As is shown in Problem 4, factor matrices are equivalent to the mapping information of the normalized tensor in high-dimensional space composed of the guidance tensor. Factor matrices build the connection between the normalized tensor and the core tensor.

Problem 4: Given the normalized tensor χ derived by OTenr and the guidance tensor ϖ, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $\varpi \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$. χ represents alegal case. Figure out matrices $\{P_n\}$, $P_n \in \mathbb{R}^{I_n \times J_n}, n \in [1, N]$, where $\{P_n\}$ minimizes

$$F_{\text{Pro1}} = \left\| \varpi - \chi \prod_{n=1}^{N} \times_{\text{n}} P_n \right\|_F^2. \tag{1}$$

Each matrix in $\{P_n\}$ is an orthogonal matrix, $P_n^T P_n = I$ and $P_n P_n^T = I$.

$\{P_n\}$ in (1) keeps χ close to ϖ and is considered as structure factors in χ, which is most beneficial to improve the performance of RnEla. Alternating least squares (ALSs) are used to figure out the values of factor matrices $\{P_n\}$. ALS is an extension of least square method while dealing with multiple parameters. ALS updates the values of parameters by alternately calculating gradients on each parameter. Implementation of ALS is as follows.

1) Randomly initialize values of factor matrices in $\{P_n\}$.
2) Fix the value of $\{P_n\}$, where $n \neq n_0$, $n, n_0 \in [1, N]$, and calculate the partial derivative of loss function F_{Pro1} with respect to P_{n_0}, that is $(\partial F_{\text{Pro1}} / \partial P_{n_0})$.
3) Let $(\partial F_{\text{Pro1}} / \partial P_{n_0}) = 0$, and figure out the value of P_{n_0}.
4) Update n_0 and P_{n_0}.
5) Repeat above steps until the algorithm converges, or the maximum number of iterations is achieved.

Therefore, calculating the partial derivative of loss function with respect to factor matrix $(\partial F_{\text{Pro1}} / \partial P_{n_0})$ $(n_0 \in [1, N])$ is crucial for solving Problem 4. A series of lemmas are proposed to solve Problem 4 in this article. Proofs A.1–A.6 in appendix give proofs of Lemmas 5–10, respectively.

Lemma 5: Given a matrix A, $A \in \mathbb{R}^{I \times J}$, then $\|A\|_F^2 = \text{Trace}(A^T A) = \sum_{i=1}^{I} \sum_{j=1}^{J} A_{ij}^2$. The same rule applies to tensors.

Lemma 6: Given a tensor χ and a matrix A, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $A \in \mathbb{R}^{I_n \times J_n}$, $n \in [1, N]$; then, $(\chi \times_n A)_{(n)} = \chi_{(n)} A$.

Lemma 7: Given tensors χ, ϖ and matrices $\{P_n\}$, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $\varpi \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, $P_n \in \mathbb{R}^{I_n \times J_n}, n, n_0 \in [1, N]$. Let $F_1 = \text{Trace}((\chi \prod_{n=1}^{N} \times_n P_n)^T \varpi)$; then, $(\partial F_1 / \partial P_{n_0}) = (\chi \prod_{n \neq n_0}^{N} \times_n P_n)_{(n_0)}^T \varpi_{(n_0)}$.

Lemma 8: Given a tensor χ and a set of factor matrices $\{P_n\}$, $\chi \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $P_n \in \mathbb{R}^{I_n \times J_n}$, $n, n_0 \in [1, N]$; then, $((\partial \text{Trace}((\chi \prod_{n=1}^{N} \times_n P_n)^T (\chi \prod_{n=1}^{N} \times_n P_n))) / \partial P_{n_0}) = \eta (\chi \prod_{n \neq n_0}^{N} \times_n P_n)_{(n_0)}^T (\chi \prod_{n \neq n_0}^{N} \times_n P_n)_{(n_0)} P_{n_0}$, where η is a constant.

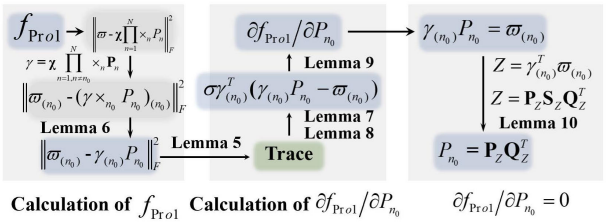

Fig. 5. Derivation process of $\{P_n\}$.

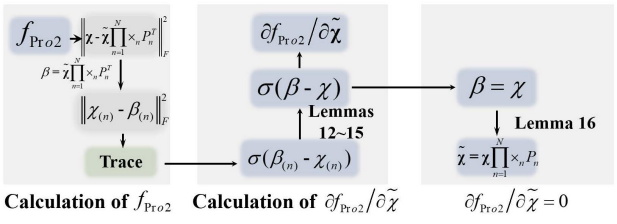

Fig. 6. Derivation process of $\widetilde{\chi}$.

Lemma 9: Given tensors χ, ϖ and a set of factor matrices $\{P_n\}$, $\chi \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $\varpi \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$, $P_n \in \mathbb{R}^{I_n\times J_n}, n,n_0 \in [1,N]$; then, $((\partial\|\varpi-\chi\prod_{n=1}^{N}\times_n P_n\|_F^2)/\partial P_{n_0}) = \sigma(\chi\prod_{n\neq n_0}^{N}\times_n P_n)_{(n_0)}^T((\chi\prod_{n\neq n_0}^{N}\times_n P_n)_{(n_0)}P_{n_0}-\varpi_{(n_0)})$, where σ is a constant.

Lemma 10: Given matrices $\gamma_{(n_0)}$ and $\varpi_{(n_0)}$, $\gamma_{(n_0)}\in\mathbb{R}^{J\times I_{n_0}}$, $\varpi_{(n_0)} \in \mathbb{R}^{J\times J_{n_0}}$, $J = J_1J_2,\ldots,J_{n_0-1}J_{n_0+1},\ldots,J_N$. Let $Z=\gamma_{(n_0)}^T\varpi_{(n_0)}$, $Z\in\mathbb{R}^{I_{n_0}\times J_{n_0}}$. According to singular value decomposition (SVD), $Z=P_Z S_Z Q_Z^T$, where S_Z is a diagonal matrix, and P_Z and Q_Z are orthogonal matrices. Then, there exists a matrix $P_{n_0}=P_Z Q_Z^T$, which satisfies $\gamma_{(n_0)}P_{n_0}-\varpi_{(n_0)}=0$. P_{n_0} is an orthogonal matrix, $P_{n_0}\in\mathbb{R}^{I_{n_0}\times J_{n_0}}$.

As shown in Fig. 5, according to Lemma 9, $(\partial F_{Pro1}/\partial P_{n_0}) = \sigma\gamma_{(n_0)}^T(\gamma_{(n_0)}P_{n_0}-\varpi_{(n_0)})$, where $\gamma = \chi\prod_{n\neq n_0}^{N}\times_n P_n$, where σ is a constant. Based on step c) in ALS and Lemma 10, let $(\partial F_{Pro1}/\partial P_{n_0})=0$; then, $P_{n_0}=P_Z Q_Z^T$, where P_Z and Q_Z are, respectively, the left and right singular matrices obtained by performing SVD on Z. That is, $Z=P_Z S_Z Q_Z^T$. At the same time, $Z=\gamma_{(n_0)}^T\varpi_{(n_0)}$.

2) Approximate the Core Tensor With the Normalized Tensor: GTend establishes the connection between the guidance tensor ϖ and the normalized tensor χ through factor matrices $\{P_n\}$. $\{P_n\}$ approaches χ and ϖ in structural and elemental terms. Problem 11 is proposed to make the core tensor approach χ and ϖ. A series of lemmas are proposed in order to solve Problem 11. Proofs A.7–A.11 in appendix give proofs of Lemmas 12–16, respectively.

Problem 11: Given the normalized tensor χ and the factor matrix set $\{P_n\}$, $\chi\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $P_n\in\mathbb{R}^{I_n\times J_n}, n\in[1,N]$, P_n is an orthogonal matrix. Figure out the tensor $\widetilde{\chi}$, $\widetilde{\chi}\in\mathbb{R}^{J_1\times J_2\times\cdots\times_N}$, and $\widetilde{\chi}$ minimizes

$$F_{Pro2}=\left\|\chi-\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T\right\|_F^2 \tag{2}$$

where $\widetilde{\chi}$ is the core tensor corresponding to χ.

Lemma 12: Given tensors χ, $\widetilde{\chi}$ and a set of factor matrices $\{P_n\}$, $\chi\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $\widetilde{\chi}\in\mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$, $P_n\in\mathbb{R}^{I_n\times J_n}, n\in[1,N]$. Then, $(\partial\text{Trace}(\chi^T\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T))/(\partial(\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T))=\chi$.

Lemma 13: Given a tensor $\widetilde{\chi}$ and a set of factor matrices $\{P_n\}$, $\widetilde{\chi}\in\mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$, $P_n\in\mathbb{R}^{I_n\times J_n}, n\in[1,N]$. Then, $(\partial\text{Trace}((\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T)^T(\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T)))/(\partial(\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T))=\varsigma\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T$, where ς is a constant.

Lemma 14: Given a tensor χ and two matrices A, B, $\chi\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $A\in\mathbb{R}^{I_k\times J_k}$, $B\in\mathbb{R}^{J_k\times L_k}$. Then, $\chi\times_k A\times_k B=\chi\times_k(AB)$.

Lemma 15: Given a tensor χ and two matrices A, B, $\chi\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $A\in\mathbb{R}^{I_p\times J_p}$, $B\in\mathbb{R}^{I_q\times J_q}$. Then, $\chi\times_p A\times_q B=\chi\times_q B\times_p A$.

Lemma 16: Given tensors χ, $\widetilde{\chi}$ and a set of factor matrices $\{P_n\}$, $\chi\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, $\widetilde{\chi}\in\mathbb{R}^{J_1\times J_2\times\cdots\times J_N}$, $P_n\in\mathbb{R}^{I_n\times J_n}, n\in[1,N]$. P_n is an orthogonal matrix. $\widetilde{\chi}$ satisfies condition $\chi=\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T$. Then, $\widetilde{\chi}=\chi\prod_{n=1}^{N}\times_n P_n$.

As shown in Fig. 6, the least square method is used to solve Problem 11. Figuring out expression of $(\partial F_{Pro2}/\partial\widetilde{\chi})$ is crucial. According to Lemma 5, $F_{Pro2}=\text{Trace}((\chi-\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T)^T(\chi-\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T))$. Let $\beta=\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T$, $\beta\in\mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$. Then, there exist $F_{Pro2}=\text{Trace}((\chi-\beta)^T(\chi-\beta))$, $F_{Pro2}=\text{Trace}(\chi^T\chi-\beta^T\chi-\chi^T\beta+\beta^T\beta)$. Since $\text{Trace}(\beta^T\chi)=\text{Trace}(\chi^T\beta)$, $F_{Pro2}=\text{Trace}(\chi^T\chi)-2\text{Trace}(\chi^T\beta)+\text{Trace}(\beta^T\beta)$. $\text{Trace}(\chi^T\chi)$ is not a function of $\widetilde{\chi}$; then, $((\partial\text{Trace}(\chi^T\chi))/\partial\widetilde{\chi})=0$.

According to the function derivation rule and Lemma 12, $((\partial\text{Trace}(\chi^T\beta))/\partial\widetilde{\chi}) = ((\partial\text{Trace}(\chi^T\beta))/\partial\beta)(\partial\beta/\partial\widetilde{\chi})$, $((\partial\text{Trace}(\chi^T\beta))/\partial\beta)=\chi$. Then, $((\partial\text{Trace}(\chi^T\beta))/\partial\widetilde{\chi}) = \chi(\partial\beta/\partial\widetilde{\chi})$. In the same manner, $((\partial\text{Trace}(\beta^T\beta))/\partial\widetilde{\chi}) = ((\partial\text{Trace}(\beta^T\beta))/\partial\beta)(\partial\beta/\partial\widetilde{\chi})$. Based on Lemma 13, $((\partial\text{Trace}(\beta^T\beta))/\partial\beta)=\sigma\beta$, where σ is a constant. Then, $((\partial\text{Trace}(\beta^T\beta))/\partial\widetilde{\chi})=\sigma\beta(\partial\beta/\partial\widetilde{\chi})$. $(\partial F_{Pro2}/\partial\widetilde{\chi})=((\partial\text{Trace}(\chi^T\chi))/\partial\widetilde{\chi})-\sigma((\partial\text{Trace}(\chi^T\beta))/\partial\widetilde{\chi})+((\partial\text{Trace}(\beta^T\beta))/\partial\widetilde{\chi})$. That is, $(\partial F_{Pro2}/\partial\widetilde{\chi}) = \sigma(\beta-\chi)$. According to the least square method and Lemma 16, Let $(\partial F_{Pro2}/\partial\widetilde{\chi}) = 0$, $\widetilde{\chi}\prod_{n=1}^{N}\times_n P_n^T=\chi$; then,

$$\widetilde{\chi}=\chi\prod_{n=1}^{N}\times_n P_n. \tag{3}$$

D. RnEla

RnEla is proposed to capture similarities between cases in order to generate fine-grained judgment prediction. RnEla consists of the similarity correlation Bi-LSTM and optimized Elastic-Net regression. The former takes similarity between cases as an important factor in determining judgments. The latter figures out fine-grained judgment prediction. RnEla combines results obtained by classification and regression models through the judgment weights and fuzzy boundaries. RnEla optimizes the guidance tensor in order to intervene in the case modeling process. Core tensors carry information that is most beneficial to improve the accuracy of RnEla.

1) Similarity Correlation Bi-LSTM: Similar cases have similar judgments. The similarity correlation Bi-LSTM takes similarities between cases as an important factor for judgment prediction using the similarity gate. The similarity correlation

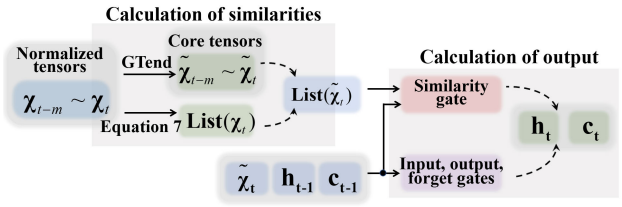

Fig. 7. Framework of the similarity correlation Bi-LSTM.

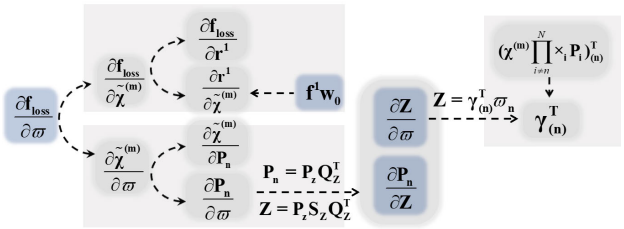

Fig. 8. Derivation process of $(\partial f_{\text{loss}}/\partial \varpi)$.

Bi-LSTM consists of forward and backward LSTMs, which have similar structures except for different directions. This section introduces the forward LSTM. Fig. 7 shows the framework of the similarity correlation Bi-LSTM.

a) Forward propagation: The input of the similarity correlation Bi-LSTM is the vectorization of core tensors obtained by GTend. The formalization of the similarity gate is defined as follows:

$$s_t = f_{\text{ELU}}\left([w_{\text{sh}}, w_{s\widetilde{\chi}}]\begin{bmatrix} h_{t-1} \\ \widetilde{\chi}_t \end{bmatrix} + b_s\right) \tag{4}$$

where s_t is the similarity gate. Let $w_s = [\,w_{\text{sh}}\ w_{s\widetilde{\chi}}\,]$, $b_s = [b_{\text{sh}}\ \text{b}_{s\widetilde{\chi}}\,]$. w_s and b_s are the weight matrix and bias term of s_t, respectively. h_{t-1} is the output of neural networks at time $t-1$. x_t is the input of neural networks at time t. f_{ELU} is the activation function of similarity gate. $w_{s\widetilde{\chi}}$ and $b_{s\widetilde{\chi}}$ enable the similarity correlation Bi-LSTM to automatically capture deep similarity information between current case $\widetilde{\chi}_t$ and its similar cases. $s_t = [s_{\text{th}}\ s_{t\widetilde{\chi}}]$. s_{th} and $\text{s}_{t\widetilde{\chi}}$ correspond to the outputs of $h_{t-1}, \widetilde{\chi}_t$ in the similarity gate s_t, respectively.

Cases with different crimes and complexity have different sensitivities to Euclidean and Cosine distance. A new method for calculating the similarity between legal cases is proposed, as shown in Definition 17. The current unit state of forward LSTM in the similarity correlation Bi-LSTM is calculated by f_t, i_t, o_t, and $\widetilde{c}_t$

$$c_t = f_t * c_{t-1} + i_t * \widetilde{c}_t + \eta \tag{5}$$

where $\eta = (1/|\text{List}(\chi_t)|)\sum_{\chi_0 \in \text{List}(\chi_t)} \text{Simi}(\chi_0, \chi_t)\vartheta_{t\widetilde{\chi}_0}$. $\vartheta_{t\widetilde{\chi}_0} = [s_{\text{th}}, (\text{s}_{t\widetilde{\chi}} + s_{t\widetilde{\chi}_0})]$. $*$ stands for multiplying by elements. f_t, i_t, and o_t are traditional gates in LSTM. Similarity(χ_0, χ_t) is used as an important factor for judgment prediction. List(χ_t) is the list of core tensors. List(χ_t) represents cases with normalized tensors whose similarity to χ_t greater than threshold η. η is preset based on the dataset and computing power of devices. Therefore, the output at time t of forward LSTM in the similarity correlation Bi-LSTM is

$$h_t = o_t * \tanh(c_t). \tag{6}$$

Definition 17 (Similarity Between Legal Cases): Given two normalized tensors χ and γ representing legal cases, $\chi, \gamma \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the similarity between χ and γ is

$$\text{Simi}(\chi, \gamma) = \text{Euc}(\chi, \gamma) + w_{\cos}\text{Cos}(\chi, \gamma) \tag{7}$$

where Euc() and Cos() are the functions for calculating the Euclidean and Cosine distances, respectively. $w_{\cos}$ is the weight of Cosine distance, which is preset according to crimes and complexity of legal cases.

b) Backward propagation: This section introduces backpropagation of the forward LSTM in the similarity correlation Bi-LSTM. The key to backward propagation lies in the optimization of the guidance tensor. The essence is to solve the partial derivative of the loss function with respect to the guidance tensor. Problem 18 gives the formal definition. As shown in Fig. 8, according to the function derivation rule, $(\partial f_{\text{loss}}/\partial \varpi) = \sum_{m=1}^{M}(\partial f_{\text{loss}}/\partial \widetilde{\chi}^{(m)})(\partial \widetilde{\chi}^{(m)}/\partial \varpi)$, $(\partial f_{\text{loss}}/\partial \widetilde{\chi}^{(m)}) = (\partial f_{\text{loss}}/\partial r^1)(\partial r^1/\partial \widetilde{\chi}^{(m)}) = f^1 w_0$. f^1 is the active function of the first layer. w_0 is the corresponding weight. $(\partial \widetilde{\chi}^{(m)}/\partial \varpi) = ((\partial(\chi^{(m)}\prod_{n=1}^{N}\times_n P_n))/\partial \varpi) = \chi^{(m)}\prod_{n=1}^{N}\times_n(\partial P_n/\partial \varpi)$. Then, $(\partial P_n/\partial \varpi) = (\partial P_n/\partial Z)(\partial Z/\partial \varpi) = (\partial P_n/\partial Z)\gamma_{(n)}^T$. $\gamma_{(n)}^T = (\chi^{(m)}\prod_{i\neq n}^{N}\times_i P_i)_{(n)}^T$.

Problem 18: Given the guidance tensor ϖ, $\varpi \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, and the loss function of forward LSTM in the similarity correlation Bi-LSTM f_{loss}, figure out the partial derivative of f_{loss} with respect to ϖ, that is, $(\partial f_{\text{loss}}/\partial \varpi)$.

2) Optimized Elastic-Net Regression: RnEla uses regression models to produce fine-grained judgment prediction. Core tensors obtained by GTend are easily multicollinearity. Results obtained by traditional linear regression models have large variance. The optimized Elastic-Net regression intervenes in GTend using the guidance tensor. As shown in Problem 19, the optimized Elastic-Net applies the guidance tensor to the objective function. Values of the guidance tensor and other model parameters are figured out using mini-batch gradient descent (MBGD). A series of lemmas are proposed in order to solve Problem 19. Proofs A.12–A.15 in appendix give proofs of Lemmas 20–23, respectively.

Problem 19: Given a set of core tensors $\{\widetilde{\chi}^{(m)}\}$ and the guidance tensor ϖ, and $\widetilde{\chi}^{(m)}, \varpi \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$, $m \in [1, M]$. $\widetilde{\chi}^{(m)}$ represents a legal case. The loss function of the optimized Elastic-Net is defined as $f_{\text{OEnet}} = (1/2m)\sum_{m=1}^{M}\|\psi(\widetilde{\chi}^{(m)}) - \varphi(\widetilde{\chi}^{(m)}, \xi)\|_F^2 + \phi(\alpha_1, \lambda_1, \xi) + \phi(\alpha_2, \lambda_2, \varpi)$. $\psi(\widetilde{\chi}^{(m)})$ is judgment of case represented by $\widetilde{\chi}^{(m)}$, including sentences and fines. $\varphi(\widetilde{\chi}^{(m)}, \xi) = \widetilde{\chi}^{(m)T}\xi$. $\phi(\alpha, \lambda, \mu) = \alpha(\lambda\|\mu\|_1 + (1-\lambda)\|\mu\|_F^2)$. $\|\mu\|_1$ and $\|\mu\|_F^2$ are $L1$ and $L2$ regularization terms, respectively.

Lemma 20: Given a tensor $\widetilde{\chi}^{(m)}$ and parameters of the optimized Elastic-Net ξ, $\widetilde{\chi}^{(m)} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}, m \in [1, M]$. $\widetilde{\chi}^{(m)}$ represents a legal case, $\xi \in \mathbb{R}^{J_1 J_2, \ldots, J_N}$. Let $f_\psi = \text{Trace}(\xi^T \widetilde{\chi}^{(m)}\psi(\widetilde{\chi}^{(m)}))$, where $\psi(\widetilde{\chi}^{(m)})$ is judgment of $\widetilde{\chi}^{(m)}$. Then, the partial derivative of f_ψ to $\widetilde{\chi}^{(m)}$ is $(\partial f_\psi/\partial \widetilde{\chi}^{(m)}) = \xi\psi(\widetilde{\chi}^{(m)})$.

Lemma 21: Given matrices A, B, and C, $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, and $C \in \mathbb{R}^{p \times m}$; then, $\text{Trace}(ABC) = \text{Trace}(CAB)$.

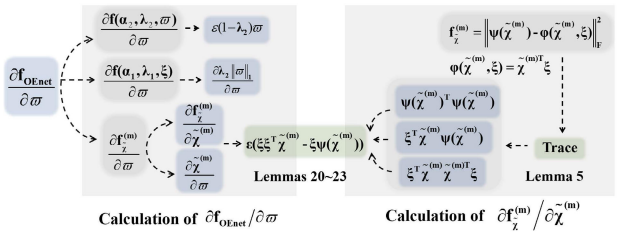

Fig. 9. Derivation process of $(\partial f_{\text{OEnet}}/\partial \varpi)$.

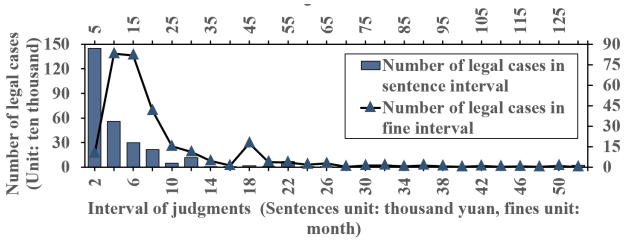

Fig. 10. Distribution of the dataset.

Lemma 22: Given a tensor $\tilde{\chi}^{(m)}$ and parameters of the optimized Elastic-Net ξ, $\tilde{\chi}^{(m)}$ represents a legal case, $\tilde{\chi}^{(m)} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}, m \in [1, M]$, $\xi \in \mathbb{R}^{J_1 J_2, \ldots, J_N}$. Let $f_\xi = \text{Trace}(\tilde{\chi}^{(m)T} \xi \xi^T \tilde{\chi}^{(m)})$; then, the partial derivative of f_ξ with respect to $\tilde{\chi}^{(m)}$ is $(\partial f_\xi / \partial \tilde{\chi}^{(m)}) = \varepsilon \xi \xi^T \tilde{\chi}^{(m)}$, where ε is a constant.

Lemma 23: Given the guidance tensor $\varpi, \varpi \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$. Let $f_\varpi = \|\varpi\|_F^2$; then, the partial derivative of f_ϖ with respect to ϖ is $(\partial f_\varpi / \partial \varpi) = \varepsilon \varpi$, where ε is a constant.

MBGD is used to figure out parameters ξ and ϖ. The key is to calculate the partial derivative of f_{OEnet} with respect to ϖ. As shown in Fig. 9, let $f_{\tilde{\chi}}^{(m)} = \|\psi(\tilde{\chi}^{(m)}) - \varphi(\tilde{\chi}^{(m)}, \xi)\|_F^2$; then, $f_{\text{OEnet}} = (1/2m) \sum_{m=1}^{M} f_{\tilde{\chi}}^{(m)} + \phi(\alpha_1, \lambda_1, \xi) + \phi(\alpha_2, \lambda_2, \varpi)$. $(\partial f_{\text{OEnet}}/\partial \varpi) = (1/2m) \sum_{m=1}^{M} (\partial f_{\tilde{\chi}}^{(m)}/\partial \varpi) + ((\partial \phi(\alpha_1, \lambda_1, \xi))/\partial \varpi) + ((\partial \phi(\alpha_2, \lambda_2, \varpi))/\partial \varpi)$. Based on Lemma 5, $f_{\tilde{\chi}}^{(m)} = \text{Trace}((\psi(\tilde{\chi}^{(m)}) - \tilde{\chi}^{(m)T}\xi)^T(\psi(\tilde{\chi}^{(m)}) - \tilde{\chi}^{(m)T}\xi))$. Then, $f_{\tilde{\chi}}^{(m)} = \text{Trace}(\psi(\tilde{\chi}^{(m)})^T \psi(\tilde{\chi}^{(m)}) - \psi(\tilde{\chi}^{(m)})^T \tilde{\chi}^{(m)T} \xi - \xi^T \tilde{\chi}^{(m)} \psi(\tilde{\chi}^{(m)}) + \xi^T \tilde{\chi}^{(m)} \tilde{\chi}^{(m)T} \xi)$.

According to Lemma 21, $\text{Trace}(\psi(\tilde{\chi}^{(m)})^T \tilde{\chi}^{(m)T} \xi) = \text{Trace}(\xi^T \tilde{\chi}^{(m)} \psi(\tilde{\chi}^{(m)}))$, $f_{\tilde{\chi}}^{(m)} = \text{Trace}(\psi(\tilde{\chi}^{(m)})^T \psi(\tilde{\chi}^{(m)}) - 2\xi^T \tilde{\chi}^{(m)} \psi(\tilde{\chi}^{(m)}) + \xi^T \tilde{\chi}^{(m)} \tilde{\chi}^{(m)T} \xi)$. Then, $(\partial f_{\tilde{\chi}}^{(m)}/\partial \varpi) = (\partial f_{\tilde{\chi}}^{(m)}/\partial \tilde{\chi}^{(m)})(\partial \tilde{\chi}^{(m)}/\partial \varpi)$. Since $((\partial \text{Trace}(\psi(\tilde{\chi}^{(m)})^T \psi(\tilde{\chi}^{(m)})))/\partial \tilde{\chi}^{(m)}) = 0$, according to Lemma 20, $((\partial \text{Trace}(\xi^T \tilde{\chi}^{(m)} \psi(\tilde{\chi}^{(m)})))/\partial \tilde{\chi}^{(m)}) = \xi \psi(\tilde{\chi}^{(m)})$. Based on Lemmas 21 and 22, $\text{Trace}(\xi^T \tilde{\chi}^{(m)} \tilde{\chi}^{(m)T} \xi) = \text{Trace}(\tilde{\chi}^{(m)T} \xi \xi^T \tilde{\chi}^{(m)})$, $((\partial \text{Trace}(\tilde{\chi}^{(m)T} \xi \xi^T \tilde{\chi}^{(m)}))/\partial \tilde{\chi}^{(m)}) = 2\xi \xi^T \tilde{\chi}^{(m)}$

$$\frac{\partial f_{\tilde{\chi}}^{(m)}}{\partial \tilde{\chi}^{(m)}} = 2(\xi \xi^T \tilde{\chi}^{(m)} - \xi \psi(\tilde{\chi}^{(m)})). \tag{8}$$

Based on the loss function of the optimized Elastic-Net, $(\partial f_{\text{OEnet}}/\partial \varpi) = (1/2m) \sum_{m=1}^{M} (\partial f_{\tilde{\chi}}^{(m)}/\partial \varpi) + ((\partial \phi(\alpha_1, \lambda_1, \xi))/\partial \varpi) + ((\partial \phi(\alpha_2, \lambda_2, \varpi))/\partial \varpi)$. The value of $(\partial \tilde{\chi}^{(m)}/\partial \varpi)$ can be figured out. Then, the value of $(\partial f_{\tilde{\chi}}^{(m)}/\partial \varpi)$ can be calculated using the function derivative rules, $(\partial f_{\tilde{\chi}}^{(m)}/\partial \varpi) = (\partial f_{\tilde{\chi}}^{(m)}/\partial \tilde{\chi}^{(m)})(\partial \tilde{\chi}^{(m)}/\partial \varpi)$. In addition, $((\partial \phi(\alpha_1, \lambda_1, \xi))/\partial \varpi) = 0$. $((\partial \phi(\alpha_2, \lambda_2, \varpi))/\partial \varpi) = ((\partial(\alpha_2(\lambda_2 \|\varpi\|_1 + (1 - \lambda_2)\|\varpi\|_F^2)))/\partial \varpi)$. Based on Lemma 23, $(\partial \|\varpi\|_F^2/\partial \varpi) = \varepsilon \varpi$, where ε is a constant. Since $(\partial \|\varpi\|_1/\partial \varpi) = \pm 1$, $((\partial \phi(\alpha_2, \lambda_2, \varpi))/\partial \varpi) = \alpha_2(\pm 1 + \varepsilon(1 - \lambda_2)\varpi)$. Then, $(\partial f_{\text{OEnet}}/\partial \xi) = \alpha_1(\pm 1 + \varepsilon(1 - \lambda_1)\xi)$, $(\partial f_{\text{OEnet}}/\partial \varpi)$ and $(\partial f_{\text{OEnet}}/\partial \xi)$.

RnEla combines the similarity correlation Bi-LSTM and optimized Elastic-Net regression using the judgment weights and fuzzy boundaries. $\text{Pre}_{\text{final}} = w_\theta \text{Dist}(\text{Pre}_{\text{simi}}, \text{Pre}_{\text{OEnet}}) + \beta_\theta \text{Inte}(\text{Pre}_{\text{simi}}, \beta_\theta)$, where Dist(), Inte(), w_θ, and β_θ are difference and integer quotient calculation functions, judgment weights, and fuzzy boundaries, respectively.

TABLE I
METHOD STRATEGIES

Strategies	Interpretation
STG_{b1}	Feature models and prediction algorithms
STG_{b2}	Matrix decomposition and prediction algorithms
STG_{b3}	OTenr and prediction algorithms
STG_{b4}	Unsupervised tensor decomposition and prediction algorithms
STG_{b5}	OTenr, GTend and prediction algorithms
STG_{b6}	OTenr, GTend, conventional similarity calculation method and prediction algorithms
STG_{b7}	OTenr, GTend, the optimized Elastic-Net regression and prediction algorithms

IV. Empirical Results

A. Dataset Description

This article uses data collected from multiple judicial devices as the experimental dataset, which is scraped from China judgment network and other online processing platforms and involves more than 20 provinces and cities across China. The dataset contains 3 000 000 cases from 2005 to 2018 and involves 128 crimes. Judgment of each case is divided into fines and sentences. Fig. 10 gives distribution of the dataset. The upper and lower abscissas are the prison term and fine intervals, respectively. The vertical axis is the number of legal cases within the sentence or fine intervals.

B. Baseline Approaches

In order to reflect roles of OTenr, GTend, and RnEla in the method proposed in this article, based on the existing judgment prediction methods, Table I shows method strategies, which are set from the perspective of case modeling and judgment prediction process. Prediction algorithms consist of nearest neighbor, naive Bayes, ANN, SVM, Fasttext, TextCNN, TextCNN attention, TextRNN, TextRNN attention, LSTM, Bi-LSTM, GRU, Bi-GRU, Bi-LSTM attention, and Bi-GRU attention methods, which are denoted by JPM_{c1} to JPM_{c15}, respectively.

Initialization values of the weight matrix in OTenr and the guidance tensor seriously affect judgment prediction accuracy and convergence speed of our method. According to the element distribution and multilinear correlation of the initialization tensor, a series of initialization values are set for the

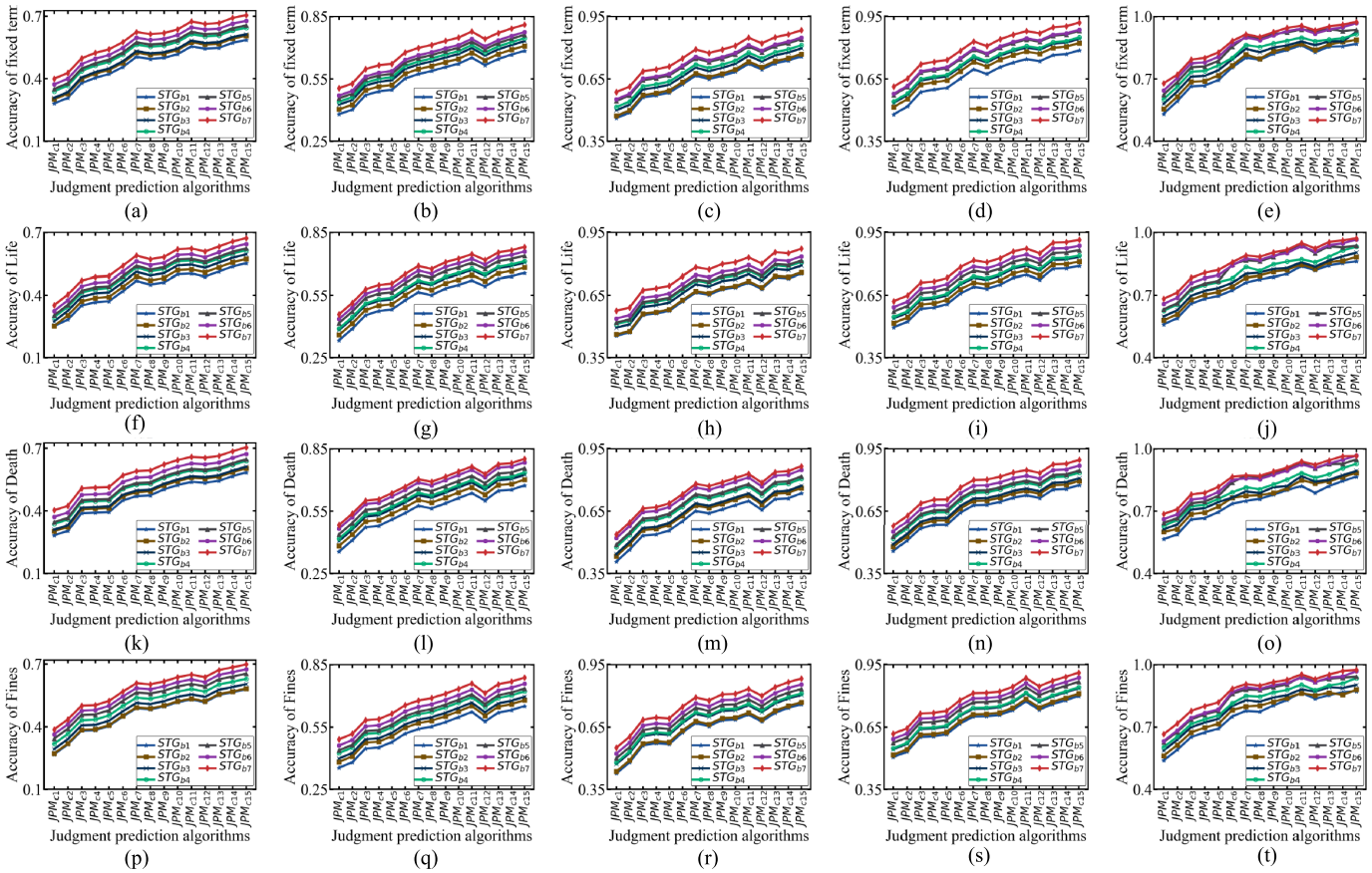

Fig. 11. Judgment prediction accuracy of algorithms on $\mathrm{ENV}_{\chi_1} \approx \mathrm{ENV}_{\chi_5}$. (a) ENV_{χ_1} (fixed term). (b) ENV_{χ_2} (fixed term). (c) ENV_{χ_3} (fixed term). (d) ENV_{χ_4} (fixed term). (e) ENV_{χ_5} (fixed term). (f) ENV_{χ_1} (life). (g) ENV_{χ_2} (life). (h) ENV_{χ_3} (life). (i) ENV_{χ_4} (life). (j) ENV_{χ_5} (life). (k) ENV_{χ_1} (death). (l) ENV_{χ_2} (death). (m) ENV_{χ_3} (death). (n) ENV_{χ_4} (death). (o) ENV_{χ_5} (death). (p) ENV_{χ_1} (fines). (q) ENV_{χ_2} (fines). (r) ENV_{χ_3} (fines). (s) ENV_{χ_4} (fines). (t) ENV_{χ_5} (fines).

normalized tensor and the guidance tensor, which are represented by $\mathrm{ENV}_{\chi_1} \approx \mathrm{ENV}_{\chi_5}$ and $\mathrm{ENV}_{\varpi_1} \approx \mathrm{ENV}_{\varpi_5}$, respectively. The larger the subscript value, the better the initialization value is to break symmetry of neural networks and eliminate multicollinearity.

C. Experimental Settings

Our method for judgment prediction consists of OTenr, GTend, and RnEla. Initialization of the weight matrix in OTenr and the guidance tensor is crucial for the performance of our method. The weight matrix and the guidance tensor are initialized with the Xavier method and follows the following principles: 1) element values are not the same in order to avoid repeated feature extraction and model degradation; 2) elements are random numbers sampled from the Gaussian or uniform distribution and close to 0; and 3) multiple linear dependencies are eliminated.

For common neural networks, such as CNNs, RNNs, LSTMs, and GRU, six iterations are performed, each batch size is 32, the size of the hidden layer is 512, the number of the hidden layer is 3, and the learning rate is 0.001. TensorFlow is used to implement our methods.

D. Experimental Results and Analysis

1) Accuracy of Judgment Prediction: Figs. 11 and 12 show the accuracy of judgment prediction algorithms under the initialization of $\mathrm{ENV}_{\chi_1} \approx \mathrm{ENV}_{\chi_5}$ and $\mathrm{ENV}_{\varpi_1} \approx \mathrm{ENV}_{\varpi_5}$, which indicate the influence of method strategies, initialization of the weight matrix in OTenr and guidance tensor, and prediction algorithms on accuracy of judgment prediction algorithms. "Fixed," "life," "death," and "fines" correspond to accuracy of fixed-term imprisonment, life imprisonment, death sentence, and fines, respectively. For sentences and fines, the confidence interval is $\min[20\% Y_{\text{target}}, C]$, Y_{target} is the real sentence or fine, C is two months for sentences, and 1000 yuan for fines. If the error is within the confidence interval, the prediction is considered correct.

In terms of method strategies, as shown in Figs. 11 and 12, the accuracy of prediction methods based on tensor models ($\mathrm{STG}_{b3} \approx \mathrm{STG}_{b7}$) is higher than that based on feature models (STG_{b1} and STG_{b2}). The main reason is that compared with feature models, tensor models can automatically extract case features, describe cases from multiple levels, and capture correlation information between different case modules. On the basis of traditional tensor models, OTenr can obtain normalized tensors that take into account the background information of the parties and weights of special vocabularies, which is more conducive to the improvement of accuracy of the judgment prediction.

Prediction methods based on tensor decomposition ($\mathrm{STG}_{b4} \approx \mathrm{STG}_{b7}$) have higher accuracy than that based on tensor models (STG_{b3}). Tensor decomposition algorithms can

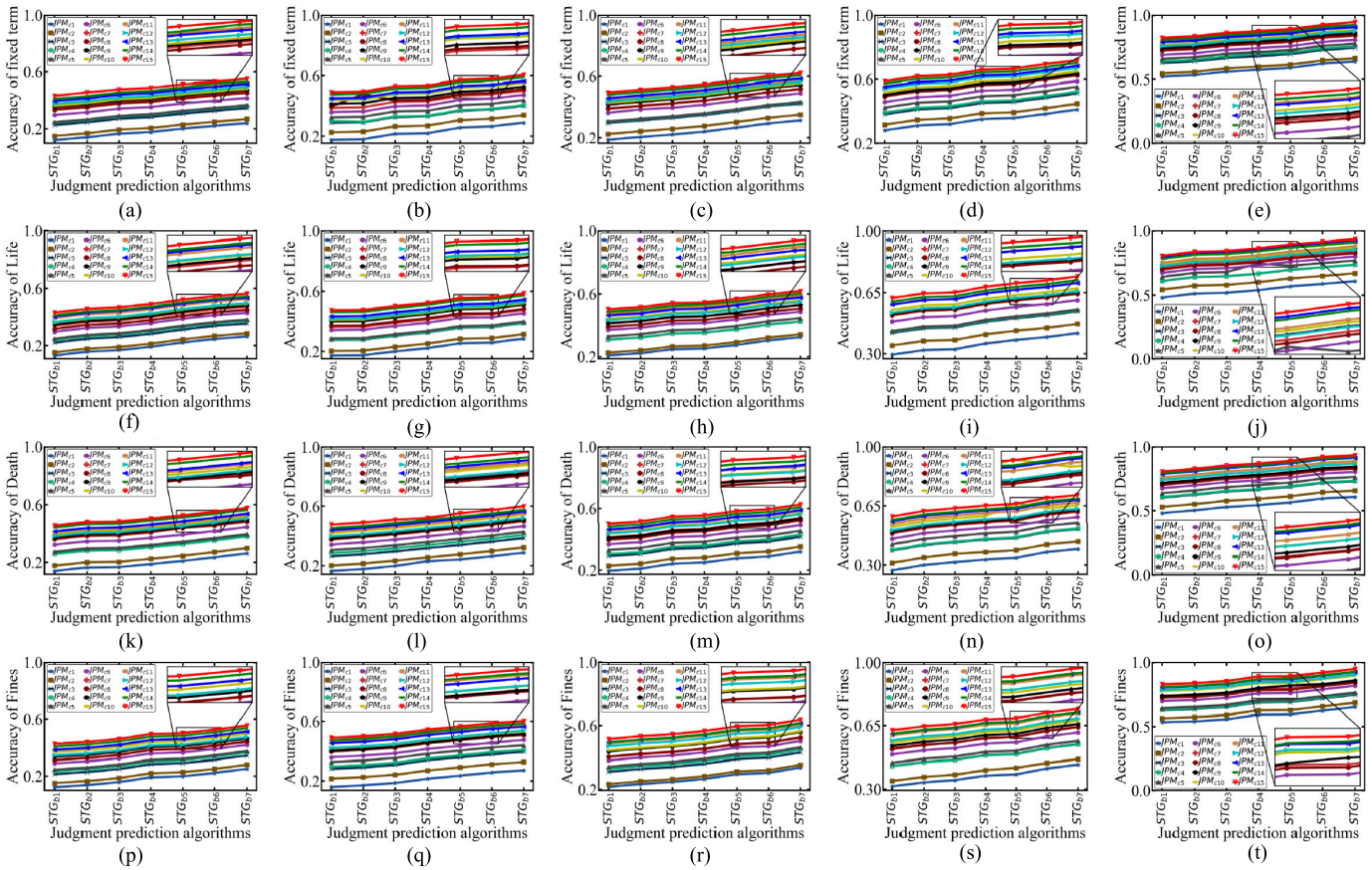

Fig. 12. Judgment prediction accuracy of algorithms on $\text{ENV}_{\varpi_1} \approx \text{ENV}_{\varpi_5}$. (a) ENV_{ϖ_1} (fixed term). (b) ENV_{ϖ_2} (fixed term). (c) ENV_{ϖ_3} (Fixed term). (d) ENV_{ϖ_4} (fixed term). (e) ENV_{ϖ_5} (fixed term). (f) ENV_{ϖ_1} (life). (g) ENV_{ϖ_2} (life). (h) ENV_{ϖ_3} (life). (i) ENV_{ϖ_4} (life). (j) ENV_{ϖ_5} (life). (k) ENV_{ϖ_1} (death). (l) ENV_{ϖ_2} (death). (m) ENV_{ϖ_3} (death). (n) ENV_{ϖ_4} (death). (o) ENV_{ϖ_5} (death). (p) ENV_{ϖ_1} (fines). (q) ENV_{ϖ_2} (fines). (r) ENV_{ϖ_3} (fines). (s) ENV_{ϖ_4} (fines). (t) ENV_{ϖ_5} (fines).

remove redundant and meaningless information in normalized tensors and reduce the impact of noise on accuracy of judgment prediction. Compared with unsupervised tensor decomposition (STG_{b4}), prediction methods based on GTend ($\text{STG}_{b5} \approx \text{STG}_{b7}$) have higher accuracy. This is because GTend establishes the connection between case modeling and judgment prediction processes through the guidance tensor, which ensures the extraction of valid information. Core tensors obtained by GTend represent tensor structural and elemental information that is most beneficial to improve the accuracy of RnEla.

Prediction methods that apply similarity between cases to judgment prediction (STG_{b6} and STG_{b7}) have higher accuracy than other methods. Similarity between cases can help to correct judgment results and improve the accuracy of judgment prediction. Compared with the direct application of the Euclidean or cosine distance (STG_{b6}), prediction methods based on RnEla (STG_{b7}) have higher accuracy. RnEla utilizes the similarity gate and new regularization terms to obtain deep similarity information between cases. In order to figure out fine-grained judgments, RnEla fuses prediction results of the similarity correlation Bi-LSTM and optimized Elastic-Net regression using the judgment weights and fuzzy boundaries.

In terms of prediction algorithms, compared with traditional methods ($\text{JPM}_{c1} \approx \text{JPM}_{c5}$), prediction methods based on deep neural networks ($\text{JPM}_{c6} \approx \text{JPM}_{c15}$) have higher accuracy of judgment prediction. Deep neural networks can extract deep case features. Compared with CNNs (JPM_{c6} and JPM_{c7}), prediction methods based on RNNs ($\text{JPM}_{c8} \approx \text{JPM}_{c15}$) have higher accuracy. Case documents are time series data. RNNs establish connections between inputs in different time periods, in order to capture contextual information, which helps to improve the accuracy of judgment prediction.

As shown in Figs. 11 and 12, LSTMs solve the gradient disappearance in RNNs through gating settings, in order to obtain long-term dependency information and improve the accuracy of judgment prediction. GRUs reduce the number of parameters in LSTMs, and the performance of the two is comparable. For prediction methods with GTend, LSTMs outperform GRUs slightly. The main reason is that the number of adjustable parameters is proportional to the intervention strength of the guidance tensor. Prediction methods with attention mechanism (JPM_{c14} and JPM_{c15}) are more accurate. The attention mechanism can weaken the weight of irrelevant information and select features, which are most relevant to judgment prediction.

In terms of initialization of the weight matrix in OTenr and guidance tensor, the larger the subscript values of ENV_{χ} and ENV_{ϖ}, the more conducive the initialization value is to breaking the symmetry of neural networks and eliminating

TABLE II
CONVERGENCE TIME OF JUDGMENT PREDICTION ALGORITHMS ON $\mathrm{ENV}_{\chi_1}\approx\mathrm{ENV}_{\chi_5}$

Strategies	Judgment prediction algorithms														
	JPM_{c1}	JPM_{c2}	JPM_{c3}	JPM_{c4}	JPM_{c5}	JPM_{c6}	JPM_{c7}	JPM_{c8}	JPM_{c9}	JPM_{c10}	JPM_{c11}	JPM_{c12}	JPM_{c13}	JPM_{c14}	JPM_{c15}
$STG_{b1}^{ENV_{\chi_1}}$	2.47e+3	5.17e+0	1.74e+2	3.03e+4	8.70e+1	6.86e+2	8.45e+2	2.49e+3	2.88e+3	1.59e+3	5.41e+3	6.47e+2	3.57e+3	8.45e+3	5.47e+3
$STG_{b1}^{ENV_{\chi_1}}$	1.69e+3	3.54e+0	1.19e+2	2.07e+4	5.96e+1	4.70e+2	5.79e+2	1.70e+3	1.97e+3	1.09e+3	3.71e+3	4.43e+2	2.45e+3	5.79e+3	3.75e+3
$STG_{b2}^{ENV_{\chi_1}}$	3.28e+3	6.86e+0	2.30e+2	4.01e+4	1.15e+2	9.10e+2	1.12e+3	3.30e+3	3.83e+3	2.11e+3	7.18e+3	8.59e+2	4.74e+3	1.12e+4	7.26e+3
$STG_{b3}^{ENV_{\chi_1}}$	1.23e+3	2.57e+0	8.63e+1	1.50e+4	4.32e+1	3.41e+2	4.20e+2	1.24e+3	1.43e+3	7.92e+2	2.69e+3	3.22e+2	1.77e+3	4.20e+3	2.72e+3
$STG_{b4}^{ENV_{\chi_1}}$	**1.67e+3**	**3.49e+0**	**1.17e+2**	**2.04e+4**	**5.87e+1**	**4.63e+2**	**5.71e+2**	**1.68e+3**	**1.95e+3**	**1.08e+3**	**3.66e+3**	**4.37e+2**	**2.41e+3**	**5.71e+3**	**3.70e+3**
$STG_{b5}^{ENV_{\chi_1}}$	1.84e+3	3.86e+0	1.30e+2	2.26e+4	6.49e+1	5.12e+2	6.31e+2	1.86e+3	2.15e+3	1.19e+3	4.04e+3	4.83e+2	2.66e+3	6.31e+3	4.09e+3
$STG_{b6}^{ENV_{\chi_1}}$	**2.00e+3**	**4.17e+0**	**1.40e+2**	**2.44e+4**	**7.02e+1**	**5.54e+2**	**6.82e+2**	**2.01e+3**	**2.33e+3**	**1.29e+3**	**4.37e+3**	**5.22e+2**	**2.88e+3**	**6.83e+3**	**4.42e+3**
$STG_{b7}^{ENV_{\chi_2}}$	2.27e+3	4.75e+0	1.60e+2	2.78e+4	8.00e+1	6.31e+2	7.77e+2	2.29e+3	2.65e+3	1.47e+3	4.98e+3	5.95e+2	3.28e+3	7.78e+3	5.04e+3
$STG_{b1}^{ENV_{\chi_2}}$	1.56e+3	3.26e+0	1.09e+2	1.91e+4	5.48e+1	4.32e+2	5.32e+2	1.57e+3	1.82e+3	1.00e+3	3.41e+3	4.08e+2	2.25e+3	5.33e+3	3.45e+3
$STG_{b2}^{ENV_{\chi_2}}$	3.02e+3	6.31e+0	2.12e+2	3.69e+4	1.06e+2	8.37e+2	1.03e+3	3.04e+3	3.52e+3	1.94e+3	6.61e+3	7.90e+2	4.36e+3	1.03e+4	6.68e+3
$STG_{b3}^{ENV_{\chi_2}}$	1.13e+3	2.36e+0	7.94e+1	1.38e+4	3.98e+1	3.14e+2	3.86e+2	1.14e+3	1.32e+3	7.28e+2	2.48e+3	2.96e+2	1.63e+3	3.87e+3	2.50e+3
$STG_{b4}^{ENV_{\chi_2}}$	**1.54e+3**	**3.21e+0**	**1.08e+2**	**1.88e+4**	**5.40e+1**	**4.26e+2**	**5.25e+2**	**1.55e+3**	**1.79e+3**	**9.90e+2**	**3.36e+3**	**4.02e+2**	**2.22e+3**	**5.25e+3**	**3.40e+3**
$STG_{b5}^{ENV_{\chi_2}}$	1.70e+3	3.55e+0	1.19e+2	2.08e+4	5.97e+1	4.71e+2	5.80e+2	1.71e+3	1.98e+3	1.09e+3	3.72e+3	4.44e+2	2.45e+3	5.81e+3	3.76e+3
$STG_{b6}^{ENV_{\chi_2}}$	**1.84e+3**	**3.84e+0**	**1.29e+2**	**2.25e+4**	**6.46e+1**	**5.09e+2**	**6.28e+2**	**1.85e+3**	**2.14e+3**	**1.18e+3**	**4.02e+3**	**4.81e+2**	**2.65e+3**	**6.28e+3**	**4.07e+3**
$STG_{b7}^{ENV_{\chi_3}}$	2.19e+3	4.57e+0	1.54e+2	2.68e+4	7.69e+1	6.07e+2	7.48e+2	2.20e+3	2.55e+3	1.41e+3	4.79e+3	5.73e+2	3.16e+3	7.48e+3	4.84e+3
$STG_{b1}^{ENV_{\chi_3}}$	1.50e+3	3.13e+0	1.05e+2	1.83e+4	5.27e+1	4.16e+2	5.12e+2	1.51e+3	1.75e+3	9.65e+2	3.28e+3	3.92e+2	2.16e+3	5.12e+3	3.32e+3
$STG_{b2}^{ENV_{\chi_3}}$	2.90e+3	6.07e+0	2.04e+2	3.55e+4	1.02e+2	8.05e+2	9.92e+2	2.92e+3	3.38e+3	1.87e+3	6.36e+3	7.60e+2	4.19e+3	9.93e+3	6.43e+3
$STG_{b3}^{ENV_{\chi_3}}$	1.09e+3	2.27e+0	7.63e+1	1.33e+4	3.82e+1	3.02e+2	3.72e+2	1.09e+3	1.27e+3	7.00e+2	2.38e+3	2.85e+2	1.57e+3	3.72e+3	2.41e+3
$STG_{b4}^{ENV_{\chi_3}}$	**1.48e+3**	**3.09e+0**	**1.04e+2**	**1.81e+4**	**5.20e+1**	**4.10e+2**	**5.05e+2**	**1.49e+3**	**1.72e+3**	**9.52e+2**	**3.24e+3**	**3.87e+2**	**2.13e+3**	**5.05e+3**	**3.27e+3**
$STG_{b5}^{ENV_{\chi_3}}$	1.63e+3	3.41e+0	1.15e+2	2.00e+4	5.74e+1	4.53e+2	5.58e+2	1.64e+3	1.90e+3	1.05e+3	3.58e+3	4.27e+2	2.36e+3	5.58e+3	3.62e+3
$STG_{b6}^{ENV_{\chi_3}}$	**1.77e+3**	**3.69e+0**	**1.24e+2**	**2.16e+4**	**6.21e+1**	**4.90e+2**	**6.04e+2**	**1.78e+3**	**2.06e+3**	**1.14e+3**	**3.87e+3**	**4.62e+2**	**2.55e+3**	**6.04e+3**	**3.91e+3**
$STG_{b7}^{ENV_{\chi_4}}$	2.06e+3	4.31e+0	1.45e+2	2.52e+4	7.25e+1	5.71e+2	7.04e+2	2.07e+3	2.40e+3	1.33e+3	4.51e+3	5.39e+2	2.98e+3	7.05e+3	4.56e+3
$STG_{b1}^{ENV_{\chi_4}}$	1.41e+3	2.95e+0	9.91e+1	1.73e+4	4.96e+1	3.91e+2	4.82e+2	1.42e+3	1.65e+3	9.09e+2	3.09e+3	3.69e+2	2.04e+3	4.83e+3	3.12e+3
$STG_{b2}^{ENV_{\chi_4}}$	2.73e+3	5.71e+0	1.92e+2	3.35e+4	9.62e+1	7.58e+2	9.34e+2	2.75e+3	3.19e+3	1.76e+3	5.99e+3	7.16e+2	3.95e+3	9.35e+3	6.05e+3
$STG_{b3}^{ENV_{\chi_4}}$	1.02e+3	2.14e+0	7.19e+1	1.25e+4	3.60e+1	2.84e+2	3.50e+2	1.03e+3	1.19e+3	6.60e+2	2.24e+3	2.68e+2	1.48e+3	3.50e+3	2.27e+3
$STG_{b4}^{ENV_{\chi_4}}$	**1.39e+3**	**2.91e+0**	**9.77e+1**	**1.70e+4**	**4.90e+1**	**3.86e+2**	**4.76e+2**	**1.40e+3**	**1.62e+3**	**8.97e+2**	**3.05e+3**	**3.64e+2**	**2.01e+3**	**4.76e+3**	**3.08e+3**
$STG_{b5}^{ENV_{\chi_4}}$	1.54e+3	3.22e+0	1.08e+2	1.88e+4	5.41e+1	4.27e+2	5.26e+2	1.55e+3	1.79e+3	9.91e+2	3.37e+3	4.03e+2	2.22e+3	5.26e+3	3.41e+3
$STG_{b6}^{ENV_{\chi_4}}$	**1.66e+3**	**3.48e+0**	**1.17e+2**	**2.04e+4**	**5.85e+1**	**4.61e+2**	**5.69e+2**	**1.67e+3**	**1.94e+3**	**1.07e+3**	**3.64e+3**	**4.35e+2**	**2.40e+3**	**5.69e+3**	**3.68e+3**
$STG_{b7}^{ENV_{\chi_5}}$	1.99e+3	4.16e+0	1.40e+2	2.44e+4	7.00e+1	5.52e+2	6.81e+2	2.00e+3	2.32e+3	1.28e+3	4.36e+3	5.21e+2	2.88e+3	6.81e+3	4.41e+3
$STG_{b1}^{ENV_{\chi_5}}$	1.36e+3	2.85e+0	9.58e+1	1.67e+4	4.80e+1	3.78e+2	4.66e+2	1.37e+3	1.59e+3	8.79e+2	2.99e+3	3.57e+2	1.97e+3	4.66e+3	3.02e+3
$STG_{b2}^{ENV_{\chi_5}}$	2.64e+3	5.52e+0	1.85e+2	3.23e+4	9.29e+1	7.33e+2	9.03e+2	2.66e+3	3.08e+3	1.70e+3	5.79e+3	6.92e+2	3.82e+3	9.04e+3	5.85e+3
$STG_{b3}^{ENV_{\chi_5}}$	9.89e+2	2.07e+0	6.95e+1	1.21e+4	3.48e+1	2.75e+2	3.38e+2	9.96e+2	1.15e+3	6.38e+2	2.17e+3	2.59e+2	1.43e+3	3.38e+3	2.19e+3
$STG_{b4}^{ENV_{\chi_5}}$	**1.34e+3**	**2.81e+0**	**9.44e+1**	**1.65e+4**	**4.73e+1**	**3.73e+2**	**4.60e+2**	**1.35e+3**	**1.57e+3**	**8.67e+2**	**2.95e+3**	**3.52e+2**	**1.94e+3**	**4.60e+3**	**2.98e+3**
$STG_{b5}^{ENV_{\chi_5}}$	1.49e+3	3.11e+0	1.04e+2	1.82e+4	5.23e+1	4.12e+2	5.08e+2	1.50e+3	1.73e+3	9.58e+2	3.26e+3	3.89e+2	2.15e+3	5.08e+3	3.29e+3
$STG_{b6}^{ENV_{\chi_5}}$	**1.61e+3**	**3.36e+0**	**1.13e+2**	**1.97e+4**	**5.66e+1**	**4.46e+2**	**5.49e+2**	**1.62e+3**	**1.87e+3**	**1.04e+3**	**3.52e+3**	**4.21e+2**	**2.32e+3**	**5.50e+3**	**3.56e+3**
STG_{b7}															

multicollinearity. As shown in Figs. 11 and 12, weak multicollinearity and strong elimination of neural network symmetry help improve the accuracy of judgment prediction. The main reason is that multicollinearity reflects the correlation between different features. Strong multicollinearity provides repeated information for prediction models. The interaction between features makes prediction models unstable and uninterpretable, which is not conducive to the improvement of accuracy of judgment prediction. Symmetry of neural networks impairs the ability of parameter adjustment and limits the improvement of method performance.

2) Convergence Analysis: Tables II and III show the convergence time of judgment prediction algorithms under the initialization of $\mathrm{ENV}_{\chi_1}\approx\mathrm{ENV}_{\chi_5}$ and $\mathrm{ENV}_{\varpi_1}\approx\mathrm{ENV}_{\varpi_5}$, which indicate the influence of method strategies, initialization of the weight matrix in OTenr and guidance tensor, and prediction algorithms on convergence time of judgment prediction algorithms. Computing device used in this article is Titan V (12G). Unit of convergence time is second. In terms of method strategies, the prediction method based on the normalized tensor (STG3) has the longest convergence time. This is because OTenr automatically extracts the features of cases and maps them into high-dimensional space, which increases the computational complexity of prediction algorithms.

Prediction methods based on decomposition (STG_{b2}, $\mathrm{STG}_{b4}\approx\mathrm{STG}_{b7}$) converge faster than original methods (STG_{b1} and STG_{b3}). Decomposition algorithms reduce the dimension of features and removes meaningless information, which reduces the computational complexity of judgment prediction algorithms. Convergence time of tensor decomposition ($\mathrm{STG}_{b4}\approx\mathrm{STG}_{b7}$) is longer than that of matrix decomposition (STG_{b2}). Tensor decomposition is equivalent to multilevel matrix decomposition, which is more time-consuming. Compared with prediction algorithm based on unsupervised tensor decomposition (STG_{b4}), prediction algorithms based on GTend ($\mathrm{STG}_{b5}\approx\mathrm{STG}_{b7}$) have slightly longer convergence time. The time consumption is brought by iterative optimization of the guidance tensor. Convergence time of methods based on RnEla (STG_{b7}) is slightly higher than that of similarity calculation methods (STG_{b6}). This is because of the use of the similarity gate and new regularization terms.

In terms of prediction algorithms, convergence time of prediction algorithms based on SVM (JPM_{c4}) and RNNs ($\mathrm{JPM}_{c8}\approx\mathrm{JPM}_{c15}$) is longer, followed by the nearest neighbor method (JPM_{c1}), which has no training time, and the test time for finding the nearest points is longer. The attention layer (JPM_{c14} and JPM_{c15}) increases convergence time of prediction algorithms. In terms of initialization of the weight matrix in OTenr and guidance tensor, the larger the values of subscripts of ENV_{χ} and ENV_{ϖ}, the shorter the convergence time. This is because of the normalization tensor and guidance tensor. Breaking symmetry of neural networks and eliminating multicollinearity contribute to the reduction of convergence time.

TABLE III
CONVERGENCE TIME OF JUDGMENT PREDICTION ALGORITHMS ON $\text{ENV}_{\varpi_1} \approx \text{ENV}_{\varpi_5}$

Strategies	Judgment prediction algorithms														
	JPM_{c1}	JPM_{c2}	JPM_{c3}	JPM_{c4}	JPM_{c5}	JPM_{c6}	JPM_{c7}	JPM_{c8}	JPM_{c9}	JPM_{c10}	JPM_{c11}	JPM_{c12}	JPM_{c13}	JPM_{c14}	JPM_{c15}
$STG_{b1}^{ENV_{\varpi_1}}$	1.91e+3	9.33e+0	1.73e+2	3.03e+4	1.00e+2	6.89e+2	8.06e+2	2.11e+3	2.79e+3	1.54e+3	5.61e+3	7.24e+2	3.67e+3	7.99e+3	5.24e+3
$STG_{b2}^{ENV_{\varpi_1}}$	1.31e+3	6.41e+0	1.19e+2	2.08e+4	6.88e+1	4.73e+2	5.54e+2	1.45e+3	1.92e+3	1.06e+3	3.85e+3	4.98e+2	2.52e+3	5.49e+3	3.60e+3
$STG_{b3}^{ENV_{\varpi_1}}$	2.46e+3	1.20e+1	2.23e+2	3.91e+4	1.29e+2	8.88e+2	1.04e+3	2.72e+3	3.60e+3	1.99e+3	7.22e+3	9.34e+2	4.73e+3	1.03e+4	6.75e+3
$STG_{b4}^{ENV_{\varpi_1}}$	9.65e+2	4.72e+0	8.73e+1	1.53e+4	5.06e+1	3.48e+2	4.08e+2	1.07e+3	1.41e+3	7.79e+2	2.83e+3	3.66e+2	1.85e+3	4.04e+3	2.65e+3
$STG_{b5}^{ENV_{\varpi_1}}$	**1.29e+3**	**6.30e+0**	**1.17e+2**	**2.05e+4**	**6.76e+1**	**4.65e+2**	**5.44e+2**	**1.42e+3**	**1.89e+3**	**1.04e+3**	**3.79e+3**	**4.89e+2**	**2.48e+3**	**5.40e+3**	**3.54e+3**
$STG_{b6}^{ENV_{\varpi_1}}$	1.44e+3	7.05e+0	1.31e+2	2.29e+4	7.57e+1	5.21e+2	6.09e+2	1.59e+3	2.11e+3	1.17e+3	4.24e+3	5.47e+2	2.77e+3	6.04e+3	3.96e+3
$STG_{b7}^{ENV_{\varpi_1}}$	**1.54e+3**	**7.50e+0**	**1.39e+2**	**2.44e+4**	**8.05e+1**	**5.54e+2**	**6.48e+2**	**1.69e+3**	**2.25e+3**	**1.24e+3**	**4.51e+3**	**5.83e+2**	**2.95e+3**	**6.43e+3**	**4.21e+3**
$STG_{b1}^{ENV_{\varpi_2}}$	1.77e+3	8.65e+0	1.60e+2	2.81e+4	9.28e+1	6.39e+2	7.47e+2	1.95e+3	2.59e+3	1.43e+3	5.20e+3	6.71e+2	3.40e+3	7.41e+3	4.85e+3
$STG_{b2}^{ENV_{\varpi_2}}$	1.22e+3	5.94e+0	1.10e+2	1.93e+4	6.37e+1	4.39e+2	5.13e+2	1.34e+3	1.78e+3	9.81e+2	3.57e+3	4.61e+2	2.33e+3	5.09e+3	3.33e+3
$STG_{b3}^{ENV_{\varpi_2}}$	2.28e+3	1.11e+1	2.06e+2	3.62e+4	1.20e+2	8.23e+2	9.63e+2	2.52e+3	3.34e+3	1.84e+3	6.70e+3	8.65e+2	4.38e+3	9.55e+3	6.25e+3
$STG_{b4}^{ENV_{\varpi_2}}$	8.95e+2	4.37e+0	8.10e+1	1.42e+4	4.69e+1	3.23e+2	3.78e+2	9.88e+2	1.31e+3	7.22e+2	2.63e+3	3.39e+2	1.72e+3	3.74e+3	2.45e+3
$STG_{b5}^{ENV_{\varpi_2}}$	**1.19e+3**	**5.84e+0**	**1.08e+2**	**1.90e+4**	**6.27e+1**	**4.31e+2**	**5.04e+2**	**1.32e+3**	**1.75e+3**	**9.65e+2**	**3.51e+3**	**4.53e+2**	**2.29e+3**	**5.00e+3**	**3.28e+3**
$STG_{b6}^{ENV_{\varpi_2}}$	1.34e+3	6.54e+0	1.21e+2	2.13e+4	7.01e+1	4.83e+2	5.65e+2	1.48e+3	1.96e+3	1.08e+3	3.93e+3	5.07e+2	2.57e+3	5.60e+3	3.67e+3
$STG_{b7}^{ENV_{\varpi_2}}$	**1.42e+3**	**6.96e+0**	**1.29e+2**	**2.26e+4**	**7.46e+1**	**5.14e+2**	**6.01e+2**	**1.57e+3**	**2.08e+3**	**1.15e+3**	**4.18e+3**	**5.40e+2**	**2.73e+3**	**5.96e+3**	**3.90e+3**
$STG_{b1}^{ENV_{\varpi_3}}$	1.70e+3	8.32e+0	1.54e+2	2.70e+4	8.92e+1	6.14e+2	7.18e+2	1.88e+3	2.49e+3	1.37e+3	5.00e+3	6.46e+2	3.27e+3	7.12e+3	4.67e+3
$STG_{b2}^{ENV_{\varpi_3}}$	1.17e+3	5.71e+0	1.06e+2	1.86e+4	6.13e+1	4.22e+2	4.93e+2	1.29e+3	1.71e+3	9.43e+2	3.43e+3	4.43e+2	2.24e+3	4.89e+3	3.20e+3
$STG_{b3}^{ENV_{\varpi_3}}$	2.19e+3	1.07e+1	1.98e+2	3.48e+4	1.15e+2	7.91e+2	9.26e+2	2.42e+3	3.21e+3	1.77e+3	6.44e+3	8.32e+2	4.21e+3	9.18e+3	6.01e+3
$STG_{b4}^{ENV_{\varpi_3}}$	8.60e+2	4.20e+0	7.78e+1	1.37e+4	4.51e+1	3.10e+2	3.63e+2	9.50e+2	1.26e+3	6.94e+2	2.53e+3	3.26e+2	1.65e+3	3.60e+3	2.36e+3
$STG_{b5}^{ENV_{\varpi_3}}$	**1.15e+3**	**5.62e+0**	**1.04e+2**	**1.83e+4**	**6.02e+1**	**4.15e+2**	**4.85e+2**	**1.27e+3**	**1.68e+3**	**9.28e+2**	**3.37e+3**	**4.36e+2**	**2.21e+3**	**4.81e+3**	**3.15e+3**
$STG_{b6}^{ENV_{\varpi_3}}$	1.29e+3	6.29e+0	1.16e+2	2.04e+4	6.74e+1	4.64e+2	5.43e+2	1.42e+3	1.88e+3	1.04e+3	3.78e+3	4.88e+2	2.47e+3	5.38e+3	3.53e+3
$STG_{b7}^{ENV_{\varpi_3}}$	**1.37e+3**	**6.69e+0**	**1.24e+2**	**2.17e+4**	**7.17e+1**	**4.94e+2**	**5.78e+2**	**1.51e+3**	**2.00e+3**	**1.10e+3**	**4.02e+3**	**5.19e+2**	**2.63e+3**	**5.73e+3**	**3.75e+3**
$STG_{b1}^{ENV_{\varpi_4}}$	1.52e+3	7.42e+0	1.37e+2	2.41e+4	7.96e+1	5.48e+2	6.41e+2	1.68e+3	2.22e+3	1.23e+3	4.46e+3	5.76e+2	2.92e+3	6.35e+3	4.16e+3
$STG_{b2}^{ENV_{\varpi_4}}$	1.04e+3	5.10e+0	9.43e+1	1.66e+4	5.47e+1	3.76e+2	4.40e+2	1.15e+3	1.53e+3	8.42e+2	3.06e+3	3.96e+2	2.00e+3	4.36e+3	2.86e+3
$STG_{b3}^{ENV_{\varpi_4}}$	1.96e+3	9.56e+0	1.77e+2	3.11e+4	1.03e+2	7.06e+2	8.26e+2	2.16e+3	2.86e+3	1.58e+3	5.74e+3	7.42e+2	3.76e+3	8.19e+3	5.36e+3
$STG_{b4}^{ENV_{\varpi_4}}$	7.67e+2	3.75e+0	6.94e+1	1.22e+4	4.02e+1	2.77e+2	3.24e+2	8.47e+2	1.12e+3	6.20e+2	2.25e+3	2.91e+2	1.47e+3	3.21e+3	2.10e+3
$STG_{b5}^{ENV_{\varpi_4}}$	**1.02e+3**	**5.01e+0**	**9.27e+1**	**1.63e+4**	**5.37e+1**	**3.70e+2**	**4.33e+2**	**1.13e+3**	**1.50e+3**	**8.27e+2**	**3.01e+3**	**3.89e+2**	**1.97e+3**	**4.29e+3**	**2.81e+3**
$STG_{b6}^{ENV_{\varpi_4}}$	1.15e+3	5.61e+0	1.04e+2	1.82e+4	6.02e+1	4.14e+2	4.84e+2	1.27e+3	1.68e+3	9.26e+2	3.37e+3	4.35e+2	2.20e+3	4.80e+3	3.15e+3
$STG_{b7}^{ENV_{\varpi_4}}$	**1.22e+3**	**5.97e+0**	**1.10e+2**	**1.94e+4**	**6.40e+1**	**4.41e+2**	**5.15e+2**	**1.35e+3**	**1.79e+3**	**9.85e+2**	**3.58e+3**	**4.63e+2**	**2.34e+3**	**5.11e+3**	**3.35e+3**
$STG_{b1}^{ENV_{\varpi_5}}$	1.48e+3	7.25e+0	1.34e+2	2.36e+4	7.77e+1	5.35e+2	6.26e+2	1.64e+3	2.17e+3	1.20e+3	4.35e+3	5.63e+2	2.85e+3	6.21e+3	4.07e+3
$STG_{b2}^{ENV_{\varpi_5}}$	1.02e+3	4.98e+0	9.21e+1	1.62e+4	5.34e+1	3.67e+2	4.30e+2	1.12e+3	1.49e+3	8.22e+2	2.99e+3	3.86e+2	1.96e+3	4.26e+3	2.79e+3
$STG_{b3}^{ENV_{\varpi_5}}$	1.91e+3	9.34e+0	1.73e+2	3.04e+4	1.00e+2	6.90e+2	8.07e+2	2.11e+3	2.80e+3	1.54e+3	5.61e+3	7.25e+2	3.67e+3	8.00e+3	5.24e+3
$STG_{b4}^{ENV_{\varpi_5}}$	7.49e+2	3.66e+0	6.78e+1	1.19e+4	3.93e+1	2.71e+2	3.16e+2	8.27e+2	1.10e+3	6.05e+2	2.20e+3	2.84e+2	1.44e+3	3.14e+3	2.06e+3
$STG_{b5}^{ENV_{\varpi_5}}$	**1.00e+3**	**4.89e+0**	**9.06e+1**	**1.59e+4**	**5.25e+1**	**3.61e+2**	**4.23e+2**	**1.10e+3**	**1.47e+3**	**8.08e+2**	**2.94e+3**	**3.80e+2**	**1.92e+3**	**4.19e+3**	**2.75e+3**
$STG_{b6}^{ENV_{\varpi_5}}$	1.12e+3	5.48e+0	1.01e+2	1.78e+4	5.87e+1	4.04e+2	4.73e+2	1.24e+3	1.64e+3	9.05e+2	3.29e+3	4.25e+2	2.15e+3	4.69e+3	3.07e+3
$STG_{b7}^{ENV_{\varpi_5}}$	**1.19e+3**	**5.83e+0**	**1.08e+2**	**1.89e+4**	**6.25e+1**	**4.30e+2**	**5.03e+2**	**1.32e+3**	**1.74e+3**	**9.62e+2**	**3.50e+3**	**4.52e+2**	**2.29e+3**	**4.99e+3**	**3.27e+3**

V. CONCLUSION

This article proposes a judgment prediction method based on tensor decomposition with optimized neural networks. OTenr represents cases as normalized tensors, which describe cases from multiple levels. OTenr captures the correlation information between different case modules. GTend decomposes normalized tensors into core tensors under the action of the guidance tensor. RnEla intervenes in GTend by optimizing the guidance tensor. Core tensors represent structural and elemental information, which is most beneficial to improve the accuracy of judgment prediction. RnEla takes the similarity between cases as an important factor for judgment prediction. RnEla combines prediction results obtained by the similarity correlation Bi-LSTM and optimized Elastic-Net regression using the judgment weights and fuzzy boundaries. Compared with previous methods, our method has higher accuracy in predicting sentences and fines of cases. In the future work, we will carry out the following research: 1) simplify and optimize OTenr and GTend, in order to further reduce the computational complexity of prediction algorithms and 2) optimize the similarity gate and regularization terms in RnEla and further calculate the similarity information between cases.

REFERENCES

[1] J. Ge, Y. Huang, X. Shen, C. Li, and W. Hu, "Learning fine-grained fact-article correspondence in legal cases," *IEEE/ACM Trans. Audio, Speech, Lang., Process.*, vol. 29, pp. 3694–3706, 2021.

[2] J. Sun, S. Wang, J. Zhang, and C. Zong, "Neural encoding and decoding with distributed sentence representations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 589–603, Feb. 2021.

[3] X. Jia, J. Yang, R. Liu, X. Wang, S. Cotofana, and W. Zhao, "Efficient computation reduction in Bayesian neural networks through feature decomposition and memorization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1703–1712, May 2020.

[4] P. Saravanan, J. Selvaprabu, R. L. Arun, K. A. Azeez, and S. K. Javubar, "Survey on crime analysis and prediction using data mining and machine learning techniques," *Advances in Smart Grid Technology* (Lecture Notes in Electrical Engineering). Singapore, 2021, pp. 435–448.

[5] X. Ruan et al., "EDP: An efficient decomposition and pruning scheme for convolutional neural network compression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4499–4513, Oct. 2021.

[6] Y. Wang, J. Wang, and H. Che, "Two-timescale neurodynamic approaches to supervised feature selection based on alternative problem formulations," *Neural Netw.*, vol. 142, pp. 180–191, Oct. 2021.

[7] X. Li, Y. Wang, and R. Ruiz, "A survey on sparse learning models for feature selection," *IEEE Trans. Cybern.*, vol. 52, no. 3, pp. 1642–1660, Mar. 2022.

[8] F. M. Bianchi, S. Scardapane, S. Løkse, and R. Jenssen, "Reservoir computing approaches for representation and classification of multivariate time series," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2169–2179, May 2021.

[9] J. Šavelka and K. D. Ashley, "Legal information retrieval for understanding statutory terms," *Artif. Intell. Law*, vol. 30, no. 2, pp. 245–289, Jun. 2022.

[10] K. Bennett and M. Hannah, "Generative fusions: Integrating technical and professional communication, disability studies, and legal studies in the work of disability inclusion and access," *IEEE Trans. Prof. Commun.*, vol. 64, no. 3, pp. 235–249, Sep. 2021.

[11] S. Yang et al., "MVE-FLK: A multi-task legal judgment prediction via multi-view encoder fusing legal keywords," *Knowl.-Based Syst.*, vol. 239, Mar. 2022, Art. no. 107960.

[12] S. K. Khare and V. Bajaj, "Time–frequency representation and convolutional neural network-based emotion recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 2901–2909, Jul. 2021.
[13] I. Chalkidis, I. Androutsopoulos, and N. Aletras, "Neural legal judgment prediction in English," 2019, *arXiv:1906.02059*.
[14] N. Xu, P. Wang, L. Chen, L. Pan, X. Wang, and J. Zhao, "Distinguish confusing law articles for legal judgment prediction," 2020, *arXiv:2004.02557*.
[15] H. Zhong, Y. Wang, C. Tu, T. Zhang, Z. Liu, and M. Sun, "Iteratively questioning and answering for interpretable legal judgment prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, 2020, pp. 1250–1257.
[16] L. K. Branting et al., "Scalable and explainable legal prediction," *Artif. Intell. Law*, vol. 29, no. 2, pp. 213–238, Jun. 2021.
[17] A. Bibal, M. Lognoul, A. de Streel, and B. Frénay, "Legal requirements on explainability in machine learning," *Artif. Intell. Law*, vol. 29, no. 2, pp. 149–169, Jun. 2021.
[18] Z. Yang, P. Wang, L. Zhang, L. Shou, and W. Xu, "A recurrent attention network for judgment prediction," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, pp. 253–266, 2019.
[19] S. Long, C. Tu, Z. Liu, and M. Sun, "Automatic judgment prediction via legal reading comprehension," in *Proc. China Nat. Conf. Chin. Comput. Linguistics*. Cham, Switzerland: Springer, 2019, pp. 558–572.
[20] R. Kumar and B. Nagpal, "Analysis and prediction of crime patterns using big data," *Int. J. Inf. Technol.*, vol. 11, no. 4, pp. 799–805, Dec. 2019.
[21] P. Das and A. Das, "Application of classification techniques for prediction and analysis of crime in India," in *Computational Intelligence in Data Mining*. Singapore: Springer, 2019, pp. 191–201.
[22] C. Kadar, R. Maculan, and S. Feuerriegel, "Public decision support for low population density areas: An imbalance-aware hyper-ensemble for spatio-temporal crime prediction," *Decis. Support Syst.*, vol. 119, pp. 107–117, Apr. 2019.
[23] A. Agrawal, J. S. Gans, and A. Goldfarb, "Exploring the impact of artificial intelligence: Prediction versus judgment," *Inf. Econ. Policy*, vol. 47, pp. 1–6, Jun. 2019.
[24] W. Yang, W. Jia, X. Zhou, and Y. Luo, "Legal judgment prediction via multi-perspective bi-feedback network," 2019, *arXiv:1905.03969*.
[25] F. Yi, Z. Yu, F. Zhuang, and B. Guo, "Neural network based continuous conditional random field for fine-grained crime prediction," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4157–4163.
[26] H. Chen, D. Cai, W. Dai, Z. Dai, and Y. Ding, "Charge-based prison term prediction with deep gating network," 2019, *arXiv:1908.11521*.
[27] Y. Chen, S. Chiang, and M. Wu, "A few-shot transfer learning approach using text-label embedding with legal attributes for law article prediction," *Appl. Intell.*, vol. 52, no. 3, pp. 2884–2902, 2022.
[28] K. Hu, L. Li, J. Liu, and D. Sun, "DuroNet: A dual-robust enhanced spatial–temporal learning network for urban crime prediction," *ACM Trans. Internet Technol.*, vol. 21, no. 1, pp. 1–24, Feb. 2021.
[29] M. Han, W. Li, S. Feng, T. Qiu, and C. Chen, "Maximum information exploitation using broad learning system for large-scale chaotic time-series prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2320–2329, Jun. 2021.
[30] A. Glowacz, "Ventilation diagnosis of angle grinder using thermal imaging," *Sensors*, vol. 21, no. 8, p. 2853, Apr. 2021.
[31] D. Wang et al., "Coarse alignment of topic and sentiment: A unified model for cross-lingual sentiment classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 736–747, Feb. 2021.
[32] A. Glowacz et al., "Fault diagnosis of angle grinders and electric impact drills using acoustic signals," *Appl. Acoust.*, vol. 179, Aug. 2021, Art. no. 108070.
[33] Z. Zhang et al., "Twin-incoherent self-expressive locality-adaptive latent dictionary pair learning for classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 947–961, Mar. 2021.
[34] A. Kumar, G. Vashishtha, C. P. Gandhi, Y. Zhou, A. Glowacz, and J. Xiang, "Novel convolutional neural network (NCNN) for the diagnosis of bearing defects in rotary machinery," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–10, 2021.
[35] Z.-G. Liu, L.-Q. Huang, K. Zhou, and T. Denoeux, "Combination of transferable classification with multisource domain adaptation based on evidential reasoning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2015–2029, May 2021.
[36] Y. Wang et al., "Equality before the law: Legal judgment consistency analysis for fairness," 2021, *arXiv:2103.13868*.
[37] Y. Hou et al., "Gated value network for multilabel classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4748–4754, Oct. 2021.
[38] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Sep. 2009.

Xiaoding Guo received the B.S. degree from Xidian University, Xi'an, China, in 2015, and the Ph.D. degree in computer science and technology from the Harbin Institute of Technology, Harbin, China, in 2021.

She is currently a Lecturer with the Henan Key Laboratory of Big Data Analysis and Processing and the School of Computer Science and Engineering, Henan University, Kaifeng, China. Her research interests include artificial intelligence, computer networks, and smart justice.

Lei Zhang received the B.S. and M.S. degrees from Henan University, Kaifeng, China, in 2003 and 2006, respectively, and the Ph.D. degree in computer science and technology from the Harbin Institute of Technology, Harbin, China, in 2015.

He is currently an Associate Professor with the Institute of Data and Knowledge Engineering, School of Computer and Information engineering, Henan University. His research interests include deep learning, computer networks, and cyberspace security.

Zhihong Tian (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and technology from the Harbin Institute of Technology, Harbin, China, in 2001, 2003, and 2006, respectively.

He served in different academic and administrative positions with the Harbin Institute of Technology. He is honored as a Pearl River Scholar in Guangdong. He is currently a Professor and the Dean of the Cyberspace Institute of Advanced Technology, Guangzhou University, Guangdong, China. He is also a part-time Professor with Carlton University, Ottawa, ON, Canada. He has authored over 200 journal and conference papers. His research interests include computer networks and cyberspace security. His research has been supported in part by the National Natural Science Foundation of China, the National Key research and Development Plan of China, and the National Hightech Research and Development Program of China (863 Program).

Prof. Tian served as a member, Chair, and General Chair of a number of international conferences. He is a Distinguished Member of the China Computer Federation.