Editor: **Erik Cambria,** MIT Media Laboratory, cambria@media.mit.edu

Extreme Learning Machines

Erik Cambria, *MIT Media Laboratory*
Guang-Bin Huang, *Nanyang Technological University, Singapore*

Machine learning and artificial intelligence have seemingly never been as critical and important to real-life applications as they are in today's autonomous, big data era. The success of machine learning and artificial intelligence relies on the coexistence of three necessary conditions: powerful computing environments, rich and/or large data, and efficient learning techniques (algorithms). The extreme learning machine (ELM) as an emerging learning technique provides efficient unified solutions to generalized feed-forward networks including but not limited to (both single- and multi-hidden-layer) neural networks, radial basis function (RBF) networks, and kernel learning.

ELM theories[1–4] show that hidden neurons are important but can be randomly generated and independent from applications, and that ELMs have both universal approximation and classification capabilities; they also build a direct link between multiple theories (specifically, ridge regression, optimization, neural network generalization performance, linear system stability, and matrix theory). Consequently, ELMs, which can be biologically inspired, offer significant advantages such as fast learning speed, ease of implementation, and minimal human intervention. They thus have strong potential as a viable alternative technique for large-scale computing and machine learning.

This special edition of Trends & Controversies includes eight original works that detail the further developments of ELMs in theories, applications, and hardware implementation. In "Representational Learning with ELMs for Big Data," the authors propose using the ELM as an auto-encoder for learning feature representations using singular values. In "A Secure and Practical Mechanism for Outsourcing ELMs in Cloud Computing," the authors propose a method for handling large data applications by outsourcing to the cloud that would dramatically reduce ELM training time. In "ELM-Guided Memetic Computation for Vehicle Routing," the authors consider the ELM as an engine for automating the encapsulation of knowledge memes from past problem-solving experiences. In "ELMVIS: A Nonlinear Visualization Technique Using Random Permutations and ELMs," the authors propose an ELM method for data visualization based on random permutations to map original data and their corresponding visualization points. In "Combining ELMs with Random Projections," the authors analyze the relationships between ELM feature-mapping schemas and the paradigm of random projections. In "Reduced ELMs for Causal Relation Extraction from Unstructured Text," the authors propose combining ELMs with neuron selection to optimize the neural network architecture and improve the ELM ensemble's computational efficiency. In "A System for Signature Verification Based on Horizontal and Vertical Components in Hand Gestures," the authors propose a novel paradigm for hand signature biometry for touchless applications without the need for handheld devices. Finally, in "An Adaptive and Iterative Online Sequential ELM-Based Multi-Degree-of-Freedom Gesture Recognition System," the authors propose an online sequential ELM-based efficient gesture recognition algorithm for touchless human machine interaction.

We thank all the authors for their contributions to this special issue. We also thank *IEEE Intelligent Systems* and its editor in chief, Daniel Zeng, for the opportunity of publishing these works.

References

1. G.-B. Huang, L. Chen, and C.-K. Siew, "Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes," *IEEE*

1541-1672/13/$31.00 © 2013 IEEE
Published by the IEEE Computer Society

Trans. Neural Networks, vol. 17, no. 4, 2006, pp. 879–892.
2. G.-B. Huang, X. Ding, and H. Zhou, "Optimization Method Based Extreme Learning Machine for Classification," *Neurocomputing*, vol. 74, 2010, pp. 155–163.
3. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, 2006, pp. 489–501.
4. G.-B. Huang et al., "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 42, no. 2, 2011, pp. 513–529.

Erik Cambria is an associate researcher at MIT Media Laboratory. Contact him at cambria@media.mit.edu.

Guang-Bin Huang is in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Contact him at egbhuang@ntu.edu.sg.

Representational Learning with ELMs for Big Data

Liyanaarachchi Lekamalage Chamara Kasun, Hongming Zhou, and Guang-Bin Huang, *School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore*
Chi Man Vong, *Faculty of Science and Technology, University of Macau*

A machine learning algorithm's generalization capability depends on the dataset, which is why engineering a dataset's features to represent the data's salient structure is important. However, feature engineering requires domain knowledge and human ingenuity to generate appropriate features.

Geoffrey Hinton[1] and Pascal Vincent[2] showed that a restricted Boltzmann machine (RBM) and auto-encoders could be used for feature engineering. These engineered features then could be used to train multiple-layer neural networks, or *deep networks*. Two types of deep networks based on RBM exist: the deep belief network (DBN)[1] and the deep Boltzmann machine (DBM).[3] The two types of auto-encoder-based deep networks are the stacked auto-encoder (SAE)[2] and the stacked denoising auto-encoder (SDAE).[3] DBNs and DBMs are created by stacking RBMs, whereas SAEs and SDAEs are created by stacking auto-encoders. Deep networks outperform traditional multilayer neural networks, single-layer feed-forward neural networks (SLFNs), and support vector machines (SVMs) for big data, but are tainted by slow learning speeds.

Guang-Bin Huang and colleagues[4] introduced the extreme learning machine (ELM) as an SLFN with a fast learning speed and good generalization capability. Similar to deep networks, our proposed multilayer ELM (ML-ELM) performs layer-by-layer unsupervised learning. This article also introduces the ELM auto-encoder (ELM-AE), which represents features based on singular values. Resembling deep networks, ML-ELM stacks on top of ELM-AE to create a multilayer neural network. It learns significantly faster than existing deep networks, outperforming DBNs, SAEs, and SDAEs and performing on par with DBMs on the MNIST[5] dataset.

Representation Learning

The ELM for SLFNs shows that hidden nodes can be randomly generated. The input data is mapped to L-dimensional ELM random feature space, and the network output is

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\beta, \quad (1)$$

where $\beta = [\beta_1, \ldots, \beta_L]^T$ is the output weight matrix between the hidden nodes and the output nodes, $\mathbf{h}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_L(\mathbf{x})]$ are the hidden node outputs (random hidden features) for input $\mathbf{x}$, and $g_i(\mathbf{x})$ is the output of the ith hidden node. Given N training samples $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N}$, the ELM can resolve the following learning problem:

$$\mathbf{H}\beta = \mathbf{T}, \quad (2)$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^T$ are target labels, and $\mathbf{H} = [\mathbf{h}^T(\mathbf{x}_1), \ldots, \mathbf{h}^T(\mathbf{x}_N)]^T$. We can calculate the output weights β from

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}, \quad (3)$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$.

To improve generalization performance and make the solution more robust, we can add a regularization term as shown elsewhere:[6]

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}. \quad (4)$$

ELM-AE's main objective to represent the input features meaningfully in three different representations:

- *Compressed*. Represent features from a higher dimensional input data space to a lower dimensional feature space.
- *Sparse*. Represent features from a lower dimensional input data space to a higher dimensional feature space.
- *Equal*. Represent features from an input data space dimension equal to feature space dimension.

The ELM is modified as follows to perform unsupervised learning: input data is used as output data $\mathbf{t} = \mathbf{x}$, and random weights and biases of the hidden nodes are chosen to be orthogonal. Bernard Widrow and colleagues[7] introduced a least mean square (LMS) implementation for the ELM and a corresponding ELM-based auto-encoder that uses nonorthogonal random hidden parameters (weights and biases). Orthogonalization of these

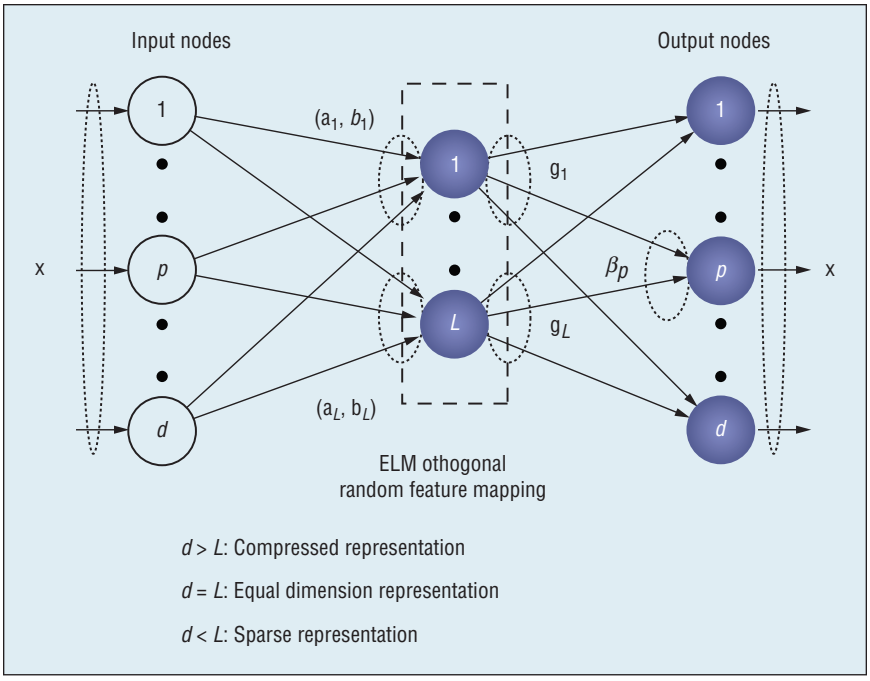

Figure 1. ELM-AE has the same solution as the original extreme learning machine except that its target output is the same as input *x*, and the hidden node parameters (a_i, b_i) are made orthogonal after being randomly generated. Here, $g_i(x) = g(a_i, b_i, x)$ is the *i*th hidden node for input *x*.

randomly generated hidden parameters tends to improve ELM-AE's generalization performance.

According to ELM theory, ELMs are universal approximators,[8] hence ELM-AE is as well. Figure 1 shows ELM-AE's network structure for compressed, sparse, and equal dimension representation. In ELM-AE, the orthogonal random weights and biases of the hidden nodes project the input data to a different or equal dimension space, as shown by the Johnson-Lindenstrauss lemma[9] and calculated as

$$\mathbf{h} = g(\mathbf{a} \cdot \mathbf{x} + \mathbf{b})$$
$$\mathbf{a}^T\mathbf{a} = \mathbf{I},\ \mathbf{b}^T\mathbf{b} = 1, \quad (5)$$

where $\mathbf{a} = [\mathbf{a}_1, \ldots, \mathbf{a}_L]$ are the orthogonal random weights, and $\mathbf{b} = [b_1, \ldots, b_L]$ are the orthogonal random biases between the input and hidden nodes.

ELM-AE's output weight β is responsible for learning the transformation from the feature space to input data. For sparse and compressed ELM-AE representations, we calculate output weights β as follows:

$$\beta = \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{X}, \quad (6)$$

where $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_N]$ are ELM-AE's hidden layer outputs, and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ are its input and output data.

For equal dimension ELM-AE representations, we calculate output weights β as follows:

$$\beta = \mathbf{H}^{-1}\mathbf{T}$$
$$\beta^T\beta = \mathbf{I}. \quad (7)$$

Singular value decomposition (SVD) is a commonly used method for feature representation. Hence we believe that ELM-AE performs feature representation similar to SVD. Equation 6's singular value decomposition (SVD) is

$$\mathbf{H}\beta = \sum_{i=1}^{N}\mathbf{u}_i\frac{\mathbf{d}_i^2}{\mathbf{d}_i^2 + C}\mathbf{u}_i^T\mathbf{X}, \quad (8)$$

where $\mathbf{u}$ are eigenvectors of $\mathbf{H}\mathbf{H}^T$, and $\mathbf{d}$ are singular values of $\mathbf{H}$, related to the SVD of input data $\mathbf{X}$. Because $\mathbf{H}$ is the projected feature space of $\mathbf{X}$ squashed via a sigmoid function, we hypothesize that ELM-AE's output weight β will learn to represent the features of the input data via singular values. To test if our hypothesis is correct, we created 10 mini datasets containing digits 0 to 9 from the MNIST dataset. Then we sent each mini dataset through an ELM-AE (network structure: 784-20-784) and compared the contents of the output weights β (Figure 2a) with the manually calculated rank 20 SVD (Figure 2b) for each mini dataset. As Figure 2 shows, ELM-AE output weight β and the manually calculated SVD basis.

Multilayer neural networks perform poorly when trained with back propagation (BP) only, so we initialize hidden layer weights in a deep network by using layer-wise unsupervised training and fine-tune the whole neural network with BP. Similar to deep networks, ML-ELM hidden layer weights are initialized with ELM-AE, which performs layer-wise unsupervised training. However, in contrast to deep networks, ML-ELM doesn't require fine tuning.

ML-ELM hidden layer activation functions can be either linear or nonlinear piecewise. If the number of nodes L^k in the kth hidden layer is equal to the number of nodes L^{k-1} in the $(k-1)$th hidden layer, g is chosen as linear; otherwise, g is chosen as nonlinear piecewise, such as a sigmoidal function:

$$\mathbf{H}^k = g((\beta^k)^T\mathbf{H}^{k-1}), \quad (9)$$

where $\mathbf{H}^k$ is the kth hidden layer output matrix. The input layer $\mathbf{x}$ can be considered as the 0th hidden layer, where $k = 0$. The output of the connections between the last hidden layer and the output node $\mathbf{t}$ is analytically calculated using regularized least squares.

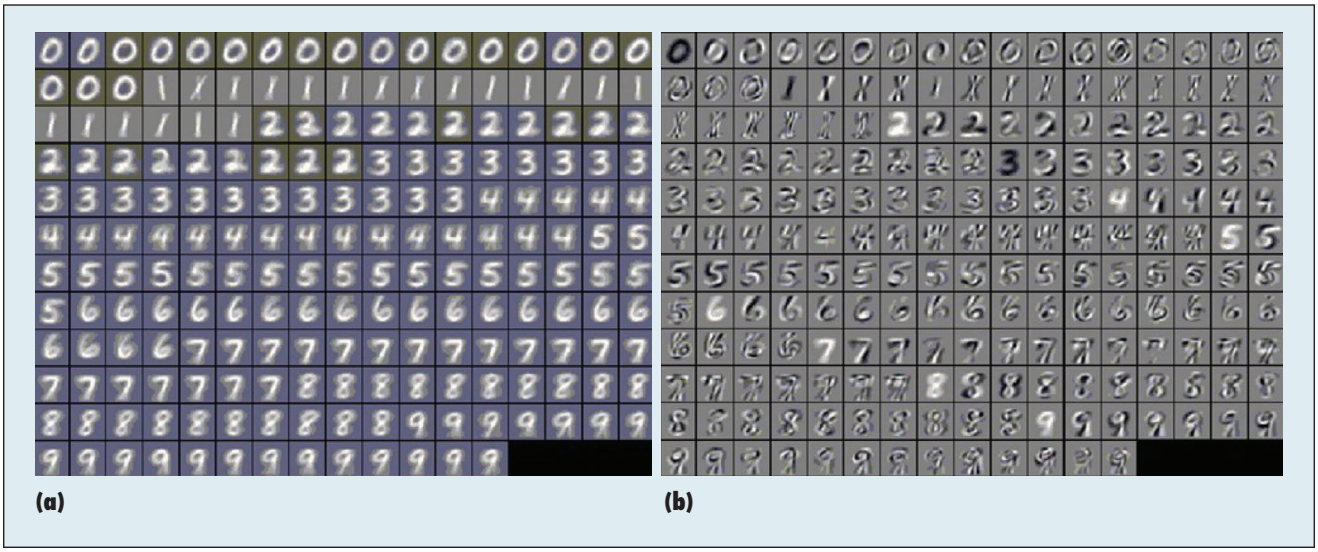

Figure 2. ELM-AE vs. singular value decomposition. (a) The output weights β of ELM-AE and (b) rank 20 SVD basis shows the feature representation of each number (0–9) in the MNIST dataset.

Performance Evaluation

The MNIST is commonly used for testing deep network performance; the dataset contains images of handwritten digits with 60,000 training samples and 10,000 testing samples. Table 1 shows the results of using the original MNIST dataset without any distortions to test the performance of ML-ELM with respect to DBNs, DBMs, SAEs, SDAEs, random feature ELMs, and Gaussian kernel ELMs.

We conducted the experiments on a laptop with a core i7 3740QM 2.7-GHz processor and 32 Gbytes of RAM running Matlab 2013a. Gaussian-kernel ELMs require a larger memory than 32 Gbytes, so we executed on a high-performance computer with dual Xeon E5-2650 2-GHz processors and 256 Gbytes of RAM running Matlab 2013a. ML-ELM (network structure: 784-700-700-15000-10 with ridge parameters 10^{-1} for layer 784-700, 10^{3} for layer 700-15000 and 10^{8} for layer 15000-10) with sigmoidal hidden layer activation function generated an accuracy of 99.03. We used DBNs and DBM network structures 748-500-500-2000-10 and 784-500-1000-10, respectively, to generate the results shown in Table 1. As a two-layer DBM network produces better results than a three-layer one,[3] we tested the two-layer network.

Table 1. Performance comparison of ML-ELM with state-of-the-art deep networks.

Algorithms	Testing accuracy % (standard deviation %)	Training time
Multi-layer extreme learning machine (ML-ELM)	99.03 (±0.04)	444.655 s
Extreme learning machine (ELM random features)	97.39 (±0.1)	545.95 s
ELM (ELM Gaussian kernel); run on a faster machine	98.75	790.96 s
Deep belief network (DBN)	98.87	20,580 s
Deep Boltzmann machine (DBM)	99.05	68,246 s
Stacked auto-encoder (SAE)	98.6	–
Stacked eenoising auto-encoder (SDAE)	98.72	–

As Table 1 shows, ML-ELM performs on par with DBMs and outperforms SAEs, SDAEs, DBNs, ELMs with random feature, and Gaussian kernel ELMs. Furthermore, ML-ELM has the least amount of required training time with respect to deep networks:

- In contrast to deep networks, ML-ELM doesn't require fine-tuning.
- ELM-AE output weights can be determined analytically, unlike RBMs and traditional auto-encoders, which require iterative algorithms.
- ELM-AE learns to represent features via singular values, unlike RBMs and traditional auto-encoders, where the actual representation of data is learned.

ELM-AE can be seen as a special case of ELM, where the input is equal to output, and the randomly generated

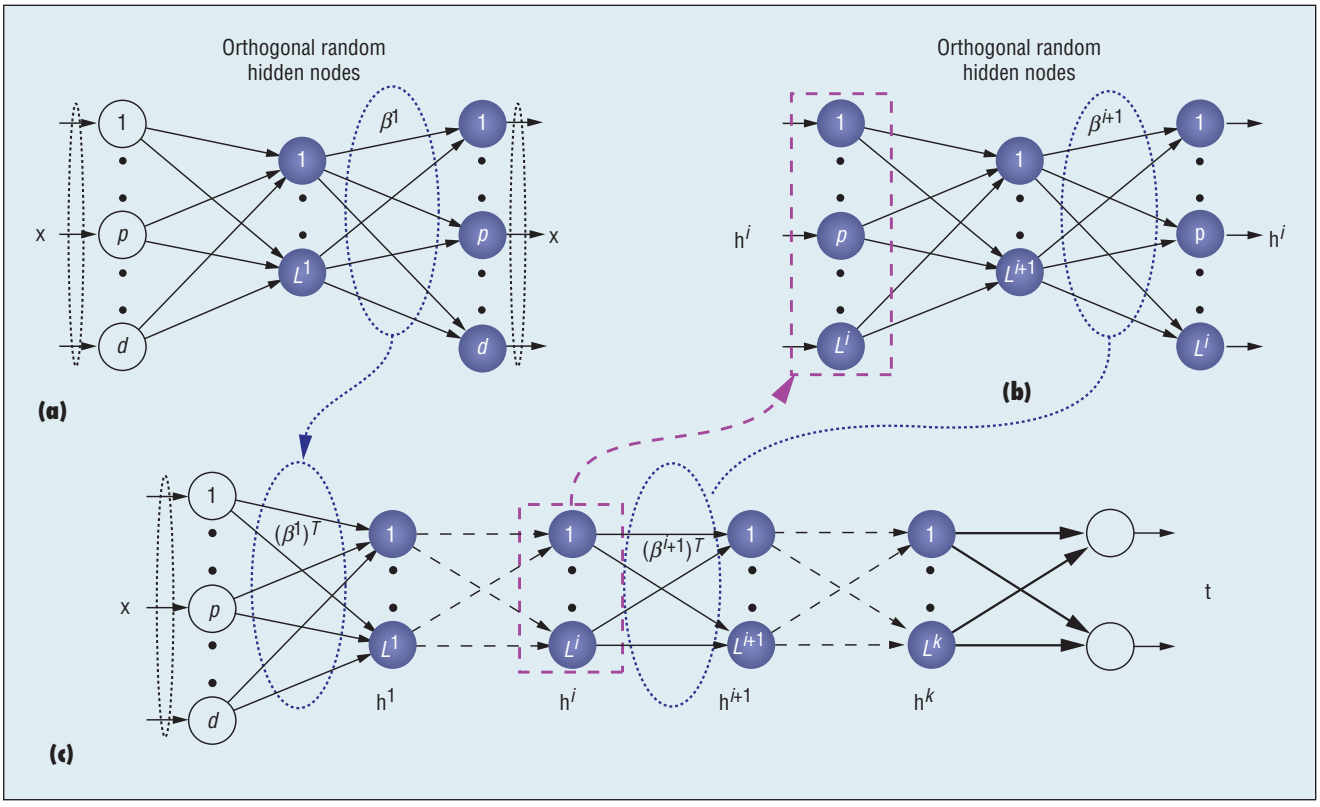

Figure 3. Adding layers in ML-ELM. (a) ELM-AE output weights β^1 with respect to input data *x* are the first-layer weights of ML-ELM. (b) The output weights β^{i+1} of ELM-AE, with respect to *i*th hidden layer output h^i of ML-ELM are the (*i* + 1)th layer weights of ML-ELM. (c) The ML-ELM output layer weights are calculated using regularized least squares.

weights are chosen to be orthogonal (see Figure 3). ELM-AE's representation capability might provide a good solution to multilayer feed-forward neural networks. ELM-based multilayer networks seem to provide better performance than state-of-the-art deep networks.

References

1. G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, 2006, pp. 504–507.
2. P. Vincent et al., "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion," *J. Machine Learning Research*, vol. 11, 2010, pp. 3371–3408.
3. R. Salakhutdinov and H. Larochelle "Efficient Learning of Deep Boltzmann Machines," *J. Machine Learning Research*, vol. 9, 2010, pp. 693–700.
4. G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, 2006, pp. 489–501.
5. Y. LeCun et al., "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, vol. 86, no. 11, 1998, pp. 2278–2324.
6. G.-B. Huang et al.,"Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 42, no. 2, 2012, pp. 513–529.
7. B. Widrow et al., "The No-Prop Algorithm: A New Learning Algorithm for Multilayer Neural Networks," *Neural Networks*, vol. 37, 2013, pp. 182–188.
8. G.-B. Huang, L. Chen, and C.-K. Siew, "Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Node," *IEEE Trans. Neural Networks*, vol. 17, no. 4, 2006, pp. 879–892.
9. W. Johnson and J. Lindenstrauss, "Extensions of Lipschitz Mappings into a Hilbert Space," *Proc. Conf. Modern Analysis and Probability*, vol. 26, 1984, pp. 189–206.

Liyanaarachchi Lekamalage Chamara Kasun is at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Contact him at chamarak001@e.ntu.edu.sg.

Hongming Zhou is at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Contact him at hmzhou@ntu.edu.sg.

Guang-Bin Huang is at the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Contact him at egbhuang@ntu.edu.sg.

Chi Man Vong is in the Faculty of Science and Technology, University of Macau. Contact him at cmvong@umac.mo.

A Secure and Practical Mechanism for Outsourcing ELMs in Cloud Computing

Jiarun Lin, Jianping Yin, Zhiping Cai, Qiang Liu, and Kuan Li, *National University of Defense Technology, China*
Victor C.M. Leung, *University of British Columbia, Vancouver, Canada*

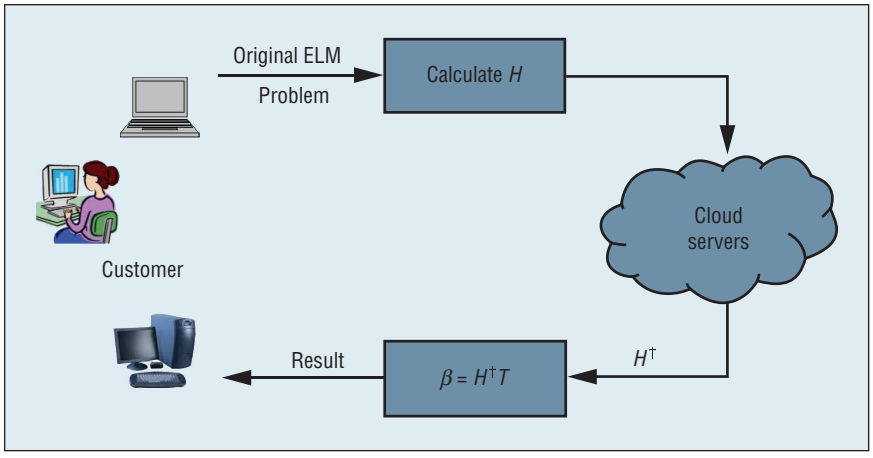

Figure 4. Architecture for outsourcing the extreme learning machine (ELM) to the cloud.

The extreme learning machine (ELM)[1–3] is a newly proposed algorithm for generalized single-hidden layer feedforward neural networks (SLFNs) that not only tends to reach the smallest training error but also the smallest norm of weights at an extremely fast learning speed, which provides good generalization performance. However, the growing volume and increasingly complex structure of the data involved in today's applications make using the ELM over large-scale data a challenging task. To address this challenge, researchers have proposed enhanced ELM variants,[4,5] but not all users have abundant computing resources or distributed computing frameworks at hand. Instead, they need to be able to outsource the expensive computation associated with ELM to the cloud to utilize its literally unlimited resources on a pay-per-use basis at relatively low prices.

To the best of our knowledge, we're the first to outsource ELM in cloud computing while assuring the I/O's confidentiality. ELM problems, in which the parameters of hidden nodes are assigned randomly and the desired output weights can be determined analytically, are suitable for being outsourced to the cloud.

This article proposes a secure and practical outsourcing mechanism called Partitioned ELM to address the challenge of performing the ELM over large-scale data. The Partitioned ELM algorithm can significantly improve the training time of the original ELM algorithm by outsourcing the heaviest computation.

We've conducted extensive experiments to evaluate our proposed mechanism's performance. The experimental and analytical results show that our proposal can save considerable ELM training time. When the size of the ELM problem increases, the speedups achieved by the proposed mechanism also grow.

Outsourcing ELM in Cloud Computing

We modeled N arbitrary distinct samples with matrices (X, T). Other work has proved that adjusting the input weights w and the biases b when training SLFNs iteratively isn't necessary.[1–3] Instead, they can be randomly assigned if the activation functions in the hidden layer are infinitely differentiable. We use M to denote the number of the hidden nodes and H to denote the output matrix of the hidden layer whose size is $N \times M$. The smallest norm least-squares solution of the output weights can be theoretically determined by $\beta = H^{\dagger}T$, where $H^{\dagger}$ is the Moore-Penrose generalized inverse.[6]

To reduce the time used for training or executing the ELM on large-scale data, it's natural to want to outsource any bottle-neck computations to the cloud. However, doing so also relinquishes the user's direct control over his or her data, and could expose sensitive information.[7]

Cloud computing can follow an "honest but curious" model, also called a semi-honest model in previous research,[8] in which the cloud server is persistently interested in analyzing data to mine more information for various purposes, either intentionally or because it's compromised. Here, we assume that cloud servers can behave unfaithfully—that is, cheat the customer to save power or reduce executing time while hoping not to be caught. To enable secure and practical outsourcing, our proposed mechanism must be ingeniously designed so as to ensure the confidentiality of ELM problems while guaranteeing correctness and soundness. We first assume that the cloud server performs the computation honestly and discuss the verification of correctness and soundness later.

Partitioned ELM Architecture

Two different entities are involved in our architecture: cloud customers and cloud servers. The former has several computationally expensive large-scale ELM problems to outsource; the latter has unlimited resources and provides utility computing services. Figure 4 shows our

architecture for outsourcing the ELM in cloud computing.

To focus on outsourcing, we omitted the authentication processes in this article, assuming that the communication channels are reliably authenticated and encrypted, which can be achieved in practice with little overhead.[9]

As the name Partitioned ELM indicates, our mechanism explicitly decomposes the ELM algorithm into a public and a private part. The private part consists of generations of random parameters and some light matrix operations. The customer calculates the output matrix of the hidden layer locally and sends it to the cloud server, which is mainly responsible for calculating the Moore-Penrose generalized inverse, the most time-consuming calculations in the ELM. Finally, the customer multiplies the inverse with the target matrix to calculate β.

Encryption of Training Samples

The ELM is instinctively suitable for outsourcing in cloud computing, while still assuring the confidentiality of the training samples and the desired parameters of neural networks, because of encryption. In the private part, the parameters ($\mathbf{w}$, $\mathbf{b}$) are assigned randomly and are part of the desired parameters of the training SLFNs. These parameters must be assigned by the cloud customer, not the server. Without any knowledge of the activation function or the parameters, the cloud server can't obtain knowledge about the exact X or ($\mathbf{w}$, $\mathbf{b}$) from H. Random parameter generation is also associated with input data confidentiality: with random parameters and randomly chosen activation functions, the customer calculates the hidden layer's output matrix, which the cloud server can't mine. In short, the encryption of X is embedded in the ELM. The confidentiality of the input and the training SLFN's parameters ($\mathbf{w}$, $\mathbf{b}$) is achieved by the randomly generated parameters and randomly chosen activation functions.

For convenience, we denote this as $H = \mathbf{g}(H_0)$, where $\mathbf{g}$ is the activation functions, and H_0 is the temporary matrix for H. Even with knowledge of the infinitely differentiable activation functions associated with the hidden nodes, the cloud server can't exactly determine X, $\mathbf{w}$, or $\mathbf{b}$ from the mediate matrix H_0. Therefore, we can also outsource the computation of the activation functions to the cloud.

The communication overhead between the customer and the cloud server can be further reduced by using pipeline parallelization, where the cloud server calculates the activation functions and receives H_0 in a pipeline manner.

Calculation of Output Weights

The cloud server receives the mediate matrix H_0 and then calculates the hidden layer's output matrix. Thereafter, it calculates the Moore-Penrose generalized inverse, whose execution time dominates the training time of the original ELM problem and sends the Moore-Penrose generalized inverse back to the customer. Finally, the customer calculates the output weights β by multiplying the inverse $H^{\dagger}$ and the target output T of the training samples locally.

During the whole process, the parameters ($\mathbf{w}$, $\mathbf{b}$, β) of the training SLFNs are kept away from the cloud server: it can't mine special information about the original ELM problems or the trained SLFNs, such as the input training samples (X, T) or the desired parameters.

Result Verifications

Up until now, we've assumed that the cloud server is honestly performing the computation, yet still interested in learning information. However, the server might behave unfaithfully, so the customer must be able to verify result correctness and soundness.

In our mechanism, the returned inverse itself can serve as the verification proof. From the definition of Moore-Penrose generalized equations, we can verify whether the returned matrix is the desired inverse.[6] Therefore, the correctness and soundness of the results can be verified while incurring low computational overhead or extra communication.

In this article, we only focus on outsourcing the basic ELM algorithm, but it's worth noting that the proposed mechanism isn't limited to a specific type of ELM and can be employed for a large variety of ELM algorithms. Applying our outsourcing mechanism to various ELM variants, especially those with regularization factor or kernels,[3] is one of our future works.

Performance Evaluation

We use t_{original} to denote the training time of the original ELM and $t_{\text{outsource}}$ to denote that of the proposed mechanism. In Partitioned ELM, the time costs at the customer and cloud server sides are denoted as t_{customer} and t_{cloud}, respectively. Then, we define the asymmetric speedup of the proposed mechanism as $\lambda = t_{\text{original}} / t_{\text{customer}}$, which physically means the savings of the customer's computing resources and is independent of how resourceful the cloud server is; rather, it's directly related to ELM problem size.

In our series of experiments, we conducted the customer computations on a workstation with an Intel Xeon Quad Processor running at 3.60 GHz with 2-Gbytes RAM and 1-Gbyte Linux swap space; we did

the cloud server computations on a workstation with an Intel Core Duo Processor running at 2.50 GHz with 4-Gbytes RAM and Windows Virtual Memory. By outsourcing the bottle-neck ELM computation from a workstation with lower resources to one with more computing power, we could evaluate the training speed of our proposed mechanism without a real cloud environment.

We tested Partitioned ELM on a large-scale dataset called CIFAR-10,[10] which consists of 50,000 32 × 32 training color images and 10,000 testing images in 10 classes; we had 5,000 training images and 1,000 testing images per class. To reduce the number of attributes, we transformed the color images into gray. We conducted five trials for each M, and randomly chose two classes from the 10 classes as the training and testing samples for each trial. Table 2 shows the results. With the increase of M, memory becomes the dominant computing resource when solving the ELM problem. The asymmetric speedup also increases, which means that the larger the problems' overall size, the larger speedups the proposed mechanism can achieve.

The training accuracy inclines steadily from 83 to 95 percent with the number of hidden nodes while the testing accuracy changes between 80 and 84 percent. We also tested the proposed mechanism over the whole CIFAR-10 dataset with feature extraction in advance. SVM and Fastfood[11] built on ELM can achieve 42.3 and 63.1 percent testing accuracy, respectively, while our method can achieve 64.5 percent testing accuracy. To find specific M for the ELM problem with the best testing accuracy, customers might want to test multiple experiments under different values of M. Then, they can realize the computing power of the cloud in a way that tests multiple ELM problems with different M simultaneously to reduce the overall training time.

Table 2. Performance over parts of the CIFAR-10 dataset.

M	$T_{original}$ (s)	$T_{outsource}$ (s)	$T_{customer}$ (s)	T_{cloud} (s)	λ
500	12.65	6.19	2.70	3.48	4.69
1,000	53.94	17.07	5.07	12.00	10.64
1,500	114.29	33.62	7.46	26.16	15.32
2,000	347.02	57.84	10.10	47.74	34.36
2,500	485.30	89.78	12.58	77.20	38.58
3,000	1,055.95	135.74	14.79	120.95	71.40
3,500	1,513.80	191.40	17.29	174.11	87.55

Given that the activation functions are infinitely differentiable, the input weights and biases involved in Partitioned ELM weren't tuned iteratively but assigned randomly, which helped us determine the output weights theoretically. Compared with traditional learning algorithms for SLFNs and deep learning algorithms, Partitioned ELM requires much less human intervention and potentially smaller training time.

By outsourcing the calculation of the Moore-Penrose generalized inverse, which is the computationally heaviest operation in the ELM, Partitioned ELM can release the customer from the heavy burden of expensive computations. The high physical savings of computing resources and the literally unlimited resources in cloud computing enable our proposed mechanism to be applied to multiple big data applications.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (project no. 61379145, 61170287, 61232016, 61070198). This research has been enabled by the use of computing resources provided by WestGrid and Compute/Calcul Canada. We thank Guang-Bin Huang and the reviewers for their constructive and insightful comments of this article.

References

1. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks," *Proc. Int'l Joint Conf. Neural Networks* (IJCNN2004), vol. 2, IEEE, 2004, pp. 985–990.
2. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, 2006, pp. 489–501.
3. G.-B. Huang et al., "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, 2012, pp. 513–529.
4. Q. He et al., "Parallel Extreme Learning Machine for Regression Based on MapReduce," *Neurocomputing*, vol. 102, 2013, pp. 52–58.
5. M. van Heeswijk et al., "GPU-Accelerated and Parallelized ELM Ensembles for Large-Scale Regression," *Neurocomputing*, vol. 74, no. 16, 2011, pp. 2430–2437.
6. D. Serre, *Matrices: Theory and Applications*, Springer, 2010.
7. Y. Cheng et al., "Efficient Revocation in Ciphertext-Policy Attribute-Based Encryption Based Cryptographic Cloud Storage," *J. Zhejiang University-Science C (Computers & Electronics)*, vol. 14, Feb. 2013, pp. 85–97.
8. C. Wang, K. Ren, and J. Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing," *Proc. INFOCOM*, IEEE, 2011, pp. 820–828.
9. P. Shi et al., "Dependable Deployment Method for Multiple Applications in Cloud Services Delivery Network," *China Communications*, vol. 8, July 2011, pp. 65–75.
10. A. Krizhevsky and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," master's thesis, Dept. of Computer Science, University of Toronto, 2009.

11. Q. Le, T. Sarlos, and A. Smola, "Fastfood-Approximating Kernel Expansions in Loglinear Time," to appear in *Proc. ICML*, 2013.

Jiarun Lin is a PhD candidate at the National University of Defense Technology, Changsha, China. Contact him at nudtjrlin@gmail.com.

Jianping Yin is a professor at the National University of Defense Technology, Changsha, China. Contact him at jpyin@nudt.edu.cn.

Zhiping Cai is an associate professor at the National University of Defense Technology, Changsha, China. Contact him at zpcai@nudt.edu.cn.

Qiang Liu is a PhD candidate at the National University of Defense Technology, Changsha, China. Contact him at libra6032009@gmail.com.

Kuan Li is an assistant professor at the National University of Defense Technology, Changsha, China. Contact him at li.kuan@163.com.

Victor C.M. Leung is a professor at the University of British Columbia, Vancouver, Canada. Contact him at vleung@ece.ubc.ca.

ELM-Guided Memetic Computation for Vehicle Routing

Liang Feng and Yew-Soon Ong, *School of Computer Engineering, Nanyang Technological University, Singapore*
Meng-Hiot Lim, *School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore*

The significance of solving the vehicle routing problem (VRP) is increasingly apparent in the fields of transportation and logistics, mainly due to the escalation of costs related to soaring fuel prices and inflation. It also poses significant national and international implications because of the traffic congestion and increased air pollution experienced in many urban cities worldwide. The VRP is a particularly challenging problem due to its complex combinatorial nature, which seeks to service a set of customers with a fleet of capacity-constrained vehicles.[1] The VRP is NP-hard, with only explicit enumeration approaches known to solve such problems optimally. However, enumeration methods don't cope well computationally with large-scale problems.

Evolutionary algorithms (EAs), on the other hand, have demonstrated notable performances and scale well. However, because of their inherent nature, which involves the iterative process of reproduction—selection, crossover, and mutation—EAs are deemed to be slow and unable to meet the pressure of delivering fast, high-quality solutions.

It's notable that learning serves as a core mechanism in human functioning and adaptation to a quickly evolving society. Past research studies sped up conventional EAs by incorporating problem-specific knowledge[2] or memes (the basic unit of cultural transmission stored in brains[3]).[4] Knowledge and memes usually exist as data structures, procedures, or rule representations, and when assimilated into search can result in faster convergence to desirable solutions.

Recently, the extreme learning machine (ELM) has been a hot topic in neural network research. Here, we consider the ELM as a meme encapsulation engine for speeding up evolutionary search on vehicle routing problems. The ELM enhances the conventional EA by automating the learning of knowledge memes from previous vehicle routing experiences. In particular, we model the knowledge meme here as an ELM-encapsulated instruction that recommends high-quality task assignments of vehicles on fresh routing problems, thus speeding up the evolutionary search towards the global optima.

Vehicle Routing with Stochastic Demand

We showcase here the VRP with stochastic demand (VRPSD), whereby consignments are delivered and collected from delivery centers to customers' doors, or vice versa, and each customer's demand is uncertain before the customer is serviced. This part of the logistics often involves routing a fleet of vehicles for physical consignment distribution; it plays a crucial role in ensuring that consignments are distributed in correct quantities. In most supply chains, this accounts for the majority of shipment costs and is the main cause of air pollution and traffic congestion in urban areas. For instance, Figure 5 depicts an example VRPSD involving 10 customers served by three capacity-constrained vehicles located at the delivery center. In VRPSD, each customer $\mathbf{v}_i$ is modeled with a stochastic *demand*(i), which is only revealed at each stop of customer $\mathbf{v}_i$. In the delivery/collection process, the assigned route τ_k might fail to fulfill the capacity constraint of its ith customer, where $C < \Sigma_{i=1,\mathbf{v}_i \in \tau_k}^{m}$ *demand*(i), at which point vehicle k will have to take a recourse action from $\mathbf{v}_{i-1}$ to the delivery center to replenish before returning to service $\mathbf{v}_i$.

The objective is thus defined as finding a route $\mathbf{s} = \{\tau_1, \tau_2, \ldots, \tau_k, \ldots, \tau_K\}$ (here, K denotes the total number of vehicles, $\tau_k = \{\mathbf{v}_0, \mathbf{v}_i, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_m, \mathbf{v}_0\}$, where $\mathbf{v}_0$ denotes the depot) that satisfies all customer demands as well as vehicle capacity constraint C, while at the same time minimizing the overall expected distance

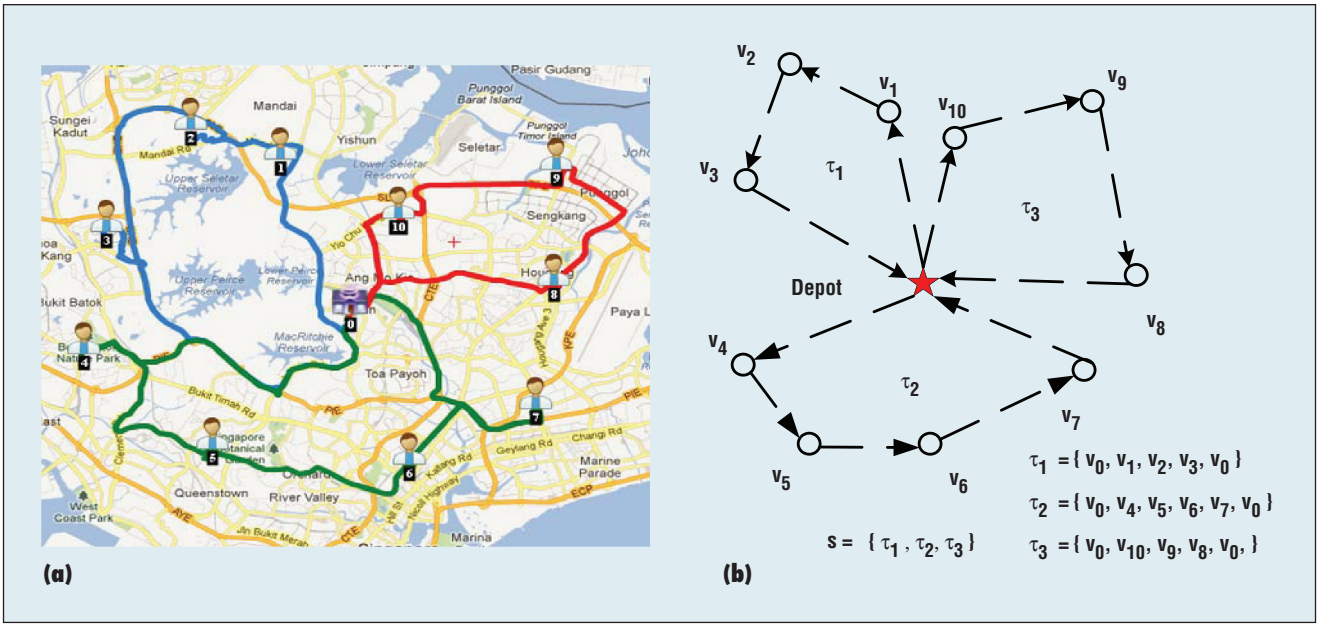

Figure 5. Realistic logistic vehicle routing. (a) The logistical vehicle routing in a typical courier service, and (b) a graph representation of that same routing plan.

traveled by all vehicles, $Cost_{VRPSD}(s)$, as given by

$$Cost_{VRPSD}(\mathbf{s}) = \sum_{k=1}^{K} L_{VRPSD}(\tau_k), \qquad (1)$$

where $L_{VRPSD}(\tau_k)$ is the expected distance traveled by vehicle k.

ELM-Guided Memetic Computation

The ELM was proposed by Guang-Bin Huang and colleagues[5] for single-layer feed-forward neural networks (SLFNs). It reported notable generalization performance with high learning efficiency and little human intervention. The training process is equivalent to finding a least-squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$, where $\mathbf{H}$ is the hidden-layer output matrix, and $\mathbf{T}$ is the target output.

Learning of Task Assignments from Previous Routing Experiences

The objective of the learning task assignment via the ELM is to create association lists of customers to vehicles from optimized routes. Suppose $\mathbf{V} = \{\mathbf{v}_i | i = \{1, \ldots, n\}\}$, where n is the number of customers, and $\mathbf{s} = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_0, \ldots, \mathbf{v}_i, \mathbf{v}_0\}$ denotes customer data and optimized routes, respectively. The location (the Cartesian coordinates) of each customer $\mathbf{v}_i$ defines the features of the learning task, $\mathbf{v}_i = \{x_1, \ldots, x_i, \ldots, x_d\}$, where d denotes the dimension. An SLFN-ELM structure is then designed to learn the task pair vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ that are served by a common vehicle in $\mathbf{s}$. To achieve this goal, we define the task pair feature vector representation as

$$\left\{f(\mathbf{v}_i), f(\mathbf{v}_j)\right\} = \left\{\left|x_1^i - x_1^j\right|, \ldots, \left|x_d^i - x_d^j\right|\right\}, \qquad (2)$$

where $|\cdot|$ denotes the absolute value operation. If $\mathbf{v}_i$ and $\mathbf{v}_j$ are served by a common vehicle in $\mathbf{s}$, the respective $\{f(\mathbf{v}_i), f(\mathbf{v}_j)\}$ will be classified with output 1; otherwise, they will be classified with output 0. The training data of class 1 task pairs and class 0 task pairs are extracted from the obtained optimized routes $\mathbf{s}$. In this manner, the recommendations for effective task assignments on unseen VRPSDs are realized via the ELM trained from previous routing experiences.

Prediction of Task Assignments in Unseen VRPSDs

The recommendations of effective task assignments involve a prediction of the vehicle to be assigned to serve each customer of the unseen VRPSD of interest. Given routing customers $\mathbf{V}' = \left\{\mathbf{v}_i' \middle| i = \{1, \ldots, m\}\right\}$, where m is the number of customers, the task pairs $\left\{f\left(\mathbf{v}_i'\right), f\left(\mathbf{v}_j'\right)\right\}$ are constructed via Equation 2. The $\mathbf{H}\beta$ output of the trained ELM classifier describes how probable the task pairs will be served by a common vehicle. With the Sigmoid function $S(t) = 1/1 + e^{-t}$, $S(\mathbf{H}\beta)$ then gives the distances between constructed task pairs in the unseen VRPSD. In this manner, for m customers, an $m \times m$ symmetric distance matrix $\mathbf{DM}$ is attained and simple clustering (such as K-Medoids) on $\mathbf{DM}$ leads to the prediction of the task assignments.

The predicted task assignments are then encoded to form the population of unseen VRPSD solution individuals in an EA so as to positively bias the search toward high-quality solutions

Learning of Task Assignment:

```
For each solved routing problem instance
  Construct task pair feature representation based on Equation 2
  Assign binary label to the constructed task pairs based on the optimized routing
  solution s.
  Train the SLFN ELM with the labeled task pairs.
End For
/*End of learning task assignment with ELM */
```

Prediction of Task Assignment for Evolutionary Search:

```
Population Initialization
  Construct task pair feature representation of the newly encountered or unseen routing
  problem instance.
  Derive distance matrix DM based on the Hβ output of the trained ELM classifier of the
  constructed task pairs.
  Apply K-Medoids on DM to obtain the task assignment.
  Encode the obtained task assignments as solutions. See Fig. 1.
  Insert the encoded solutions into the initial population of the evolutionary search.
End Initialization
While (the termination criteria are not met)
  Reproduction operator (i.e., crossover, mutation, etc.)
  Selection operator (i.e., elitism, etc.)
End While
```

Algorithm 1. Outline of proposed ELM-Guided memetic computation for vehicle routing.

Table 3. Statistical results of ELM-MEA and MEA on VRPSDs with a stochastic demand of variance 0.25.*

#	VRPSD	*Ave.Cost*	*Ave.R*	*Ave.CS*
1	A-n33-k5	≈	≈	1.23%
2	A-n45-k7	≈	≈	29.23%
3	A-n61-k9	≈	≈	19.30%
4	A-n65-k9	≈	≈	24.61%
5	B-n31-k5	≈	≈	24.79%
6	B-n45-k5	≈	≈	22.78%
7	B-n50-k7	≈	≈	14.97%
8	B-n52-k7	≈	≈	31.99%
9	B-n56-k7	≈	+	45.91%
10	B-n67-k10	≈	≈	32.30%
11	B-n78-k10	≈	≈	22.01%
12	E-n22-k4	≈	≈	30.60%
13	E-n30-k3	≈	≈	54.22%
14	E-n76-k14	≈	≈	20.03%
15	E-n76-k7	≈	≈	31.09%
16	F-n45-k4	≈	≈	25.72%
17	F-n72-k4	≈	≈	32.15%
18	M-n121-k7	≈	≈	43.99%
19	P-n101-k4	≈	≈	49.12%
20	P-n22-k8	≈	≈	51.66%

+, ≈, and – denote ELM-MEA is statistically better than, competitive with, or significantly poorer than MEA, respectively.

rapidly. Algorithm 1 details our proposed ELM-guided memetic computational framework for vehicle routing.

Realistic Logistical Vehicle Routing

We tested the numerical performance of our proposed approach on realistic logistical vehicle routing by comparing it to the recently published Monte Carlo evolutionary algorithm (MEA) for reliable VRPSD route design.[6] Our approach, which incorporates ELM-encapsulated knowledge memes from previously solved problems to provide the recommendations for high-quality solutions in the baseline MEA search on unseen VRPSDs, is thus notated as ELM-MEA. For a fair comparison, we keep all parameters and operator settings of MEA and ELM-MEA consistent with that in the original work.[6] All results reported are for 30 independent runs on 20 VRPSD instances.[6]

Table 3 tabulates the statistical results of ELM-MEA and MEA based on the Wilcoxon rank sum test under

a 95 percent confidence level. *Ave. Cost* denotes the averaged cost solution, *Ave.R* refers to the averaged route reliabilities, and *Ave.CS* denotes the mean percentage computational cost savings (in terms of the number of fitness evaluation) observed on ELM-MEA to arrive at the converged optimized solution of MEA.

From the results obtained, ELM-MEA achieved competitive solution qualities and route reliabilities to MEA on all the VRPSDs considered. But on search efficiency, ELM-MEA demonstrated superiority over MEA. When solving VRPSD "A-n33-k5," where no previous routing experience is available, MEA and ELM-MEA performed alike. On subsequent VRPSDs, ELM-MEA had increased computational cost savings of up to 54.22 percent over MEA to arrive at competitive routing solutions. It's worth highlighting that because ELM-MEA and MEA share a common baseline VRPSD solver, ELM-MEA's superior performance in search efficiency is clearly attributed to the effectiveness of the ELM-guided memetic computation approach.

Our proposed approach for efficient vehicle routing comprises two core ingredients: the automated learning of task assignments as knowledge memes from previous vehicle routing experiences and ELM prediction, which defines the task assignments of customers to vehicles based on encapsulated knowledge memes. Our demonstrations with realistic logistical vehicle routing showcase our approach's effectiveness.

Acknowledgments

This work is partially supported under the A*Star-TSRP funding, the Singapore Institute of Manufacturing Technology, and the Center for Computational Intelligence (C2I) at Nanyang Technological University.

References

1. G. Dantzig and J.H. Ramser, "The Truck Dispatching Problem," *Management Science*, vol. 6, 1959, pp. 80–91.
2. Y.C. Jin, *Knowledge Incorporation in Evolutionary Computation*, Springer, 2010.
3. X.S. Chen et al., "A Multi-Facet Survey on Memetic Computation," *IEEE Trans. Evolutionary Computation*, no. 5, 2011, pp. 591–607.
4. Y.S. Ong, M.H. Lim, and X.S. Chen, "Research Frontier: Memetic Computation–Past, Present & Future," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, 2010, pp. 24–36.
5. G.B. Huang, Q.Y. Zhu, and C.K. Siew, "Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks," *Proc. IEEE Int'l Joint Conf. Neural Networks*, IEEE, 2004, pp. 985–990.
6. X. Chen, L. Feng, and Y. S. Ong, "A self-adaptive memeplexes robust search scheme for solving stochastic demands vehicle routing problem," *Interational Journal Systems Science*, vol. 43, no. 7, pp. 1347–1366, 2012.

Liang Feng is a PhD student at the Center for Computational Intelligence in the School of Computer Engineering at Nanyang Technological University, Singapore. Contact him at feng0039@e.ntu.edu.sg.

Yew-Soon Ong is an associate professor and director of the Center for Computational Intelligence in the School of Computer Engineering at Nanyang Technological University, Singapore. Contact him at asysong@ntu.edu.sg.

Meng-Hiot Lim is an associate professor with the School of Electrical and Electronic Engineering at Nanyang Technological University, Singapore. Contact him at emhlim@ntu.edu.sg.

ELMVIS: A Nonlinear Visualization Technique Using Random Permutations and ELMs

Anton Akusok, Amaury Lendasse, and Francesco Corona, *Aalto University, Finland*
Rui Nian, *Ocean University, China*
Yoan Miche, *Aalto University, Finland*

Data visualization is an old problem in machine learning.[1] High-dimensional data is ill suited for human analysis, and only two or three dimensions can be perceived successfully. One of the simplest methods for dimensionality reduction is variable selection, in which the data can be explained by a smaller set of transformed variables.

Many nonlinear dimensionality reduction methods aim to find and unfold a manifold in the data using various cost functions and training algorithms. A common cost function is a preservation of neighborhood in original and reduced spaces. Without evident manifold structure, or if the dimensionality of manifold is still higher than the one of a reduced space, topology-preserving methods lose their point. These cases require a nonlinear dimensionality reduction method with a general cost function without other assumptions. The extreme learning machine (ELM)-based visualization method we propose here uses natural reconstruction error, while the ELM's nonlinearity provides the desired nonlinear projection.

Our proposed ELM visualisation method, denoted ELMVIS for convenience, maps the data points to some fixed points—or prototypes—in the visualization space. Their exact position is weakly relevant to data and can be chosen arbitrarily, for example, as a grid or Gaussian distributed points. The prototypes are then randomly assigned to data points, and

an ELM is used to estimate the reconstruction error. To train the visualizer, several points are chosen, their assignment permuted, and the error re-estimated. Any better solution found is kept; otherwise, the permutation is abandoned. Although the exact solution requires a factorial number of trials (all possible permutations of N points), experiments show acceptable convergence rates with up to several hundred points due to the ELM's extremely fast reconstruction error estimation. Benefits of the method are its generality and the presence of only one parameter—the number of neurons in the ELM, which doesn't require exact tuning. The method also works with very high data dimensionality.

The competitive visualization methods used here for comparison are Principal Component Analysis (PCA), self-organizing maps (SOMs),[2] and the neighborhood retrieval visualizer (NeRV).[3] PCA is a simple linear regressor with an exact solution, which maximizes the variance of a projection under orthogonality constraint. SOMs are initialized with a low-dimensional lattice embedded in the data space, which is then iteratively fit to the given data points using the quantization error. When a vertice is moved in the data space, its neighbors on a lattice perform a smaller move in the same direction, which preserves the whole lattice's integrity. NeRV approaches visualization as an information retrieval task—given a data sample as a query, the probability distributions over all the other samples to be its neighbors in both the original space and in the visualization space should be as close as possible. NeRV derives its optimization function from the Kullback-Leibler divergence between these two distributions, and thus it's the most general visualization method of the aforementioned. More details about the last two methods appear elsewhere.[2,3]

Methodology

The ELM algorithm was originally proposed by Guang-Bin Huang and colleagues[4] to use the structure of a single-layer feed-forward (SLFN) network. The main concept behind the ELM is the replacement of a computationally costly procedure of training a hidden layer by its random initialization. An output weights the matrix between the hidden representation of inputs; the true outputs remains to be found, which is a linear task. The method is proven to be universal approximator given enough hidden neurons.[5]

Consider a set of N distinct samples $(\mathbf{x}_i, \mathbf{y}_i)$ with $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \mathbb{R}^d$. An SLFN with K hidden neurons is modeled as

$$\sum_{k=1}^{K} \beta_k \phi(\mathbf{w}_k \mathbf{x}_i + b_k), i \in [1, N], \text{with } \phi$$

being the matrix activation function, $\mathbf{w}$ the input weights, b the biases, and β the output weights.

If the SLFN perfectly approximates the data, the errors between the estimated outputs $\hat{y}_i$ and the actual outputs y_i are zero, and the relation among inputs, weights, and outputs is

$$\text{then } \sum_{k=1}^{K} \beta_k \phi(\mathbf{w}_k \mathbf{x}_i + b_k) = y_i, i \in [1, N],$$

which can be written compactly as $\mathbf{H}\beta = \mathbf{Y}$, with $\beta = \left(\beta_1^T \ldots \beta_K^T\right)^T$, $\mathbf{Y} = \left(\mathbf{y}_1^T \ldots \mathbf{y}_N^T\right)^T$.

Solving the output weights β from the hidden layer representation of inputs $\mathbf{H}$ and true outputs $\mathbf{Y}$ is achieved using the Moore-Penrose generalized inverse of the matrix $\mathbf{H}$, denoted as $\mathbf{H}^{\dagger}$.[6] The ELM's training requires no iterations; the most computationally costly part is the calculation of a pseudo-inverse of the matrix $\mathbf{H}_{(D \times K)}$, which makes the ELM an extremely fast artificial neural network method.

Data Visualization with the ELM

The goal of our ELMVIS method is to maximize recall by minimizing a mean square error (MSE) of a nonlinear reconstruction provided by an ELM. Given the N data points $\mathbf{x}_i \in \mathbb{R}^D$, compactly written as a matrix $\mathbf{X} = \left(\mathbf{x}_1^T \ldots \mathbf{x}_N^T\right)^T$, the goal is to find such points $\mathbf{v}_i \in^d$ (schematically shown in Figure 6), denoted as $\mathbf{V} = \left(\mathbf{v}_1^T \ldots \mathbf{v}_N^T\right)^T$, which minimizes the recall using the ELM's reconstruction error as a nonlinear metric. Typically, d equals 2 or 3, while D could be large. Note that an ELM in the methodology performs an inverse projection $\mathbb{R}^D \leftarrow \mathbb{R}^d$ from low-dimensional visualization space to a high-dimensional original data space to estimate a reconstruction error; other dimensionality reduction methods mostly use a direct projection $\mathbb{R}^D \rightarrow \mathbb{R}^d$.

The ELM needs both input and output samples to be able to train. Data points $\mathbf{X}$ are already known, so we must set the visualization points $\mathbf{V}$. Because the manifold structure of high-dimensional data $\mathbf{X}$, if any, is unlikely to project well onto a 2D or 3D plane (except in artificially created datasets), the exact positioning of points $\mathbf{V}$ isn't of great importance. This allows fixing the positions of $\mathbf{V}$ at the beginning. Knowing $\mathbf{V}$ and $\mathbf{X}$, the only thing left to find is which point $\mathbf{v}_i$ corresponds to which point $\mathbf{x}_i$. This correspondence (or *pairings* in Figure 6) might be expressed as an ordering matrix $\mathbf{O}$. At initialization, $\mathbf{O}^0$ is an identity matrix of size $N \times N$. Some of its ones exchange indexes, such as $(1_{a,a}\ 1_{b,b}) \rightarrow (1_{a,b},\ 1_{b,a})$, which swaps samples $\mathbf{v}_a$ and $\mathbf{v}_b$ after application.

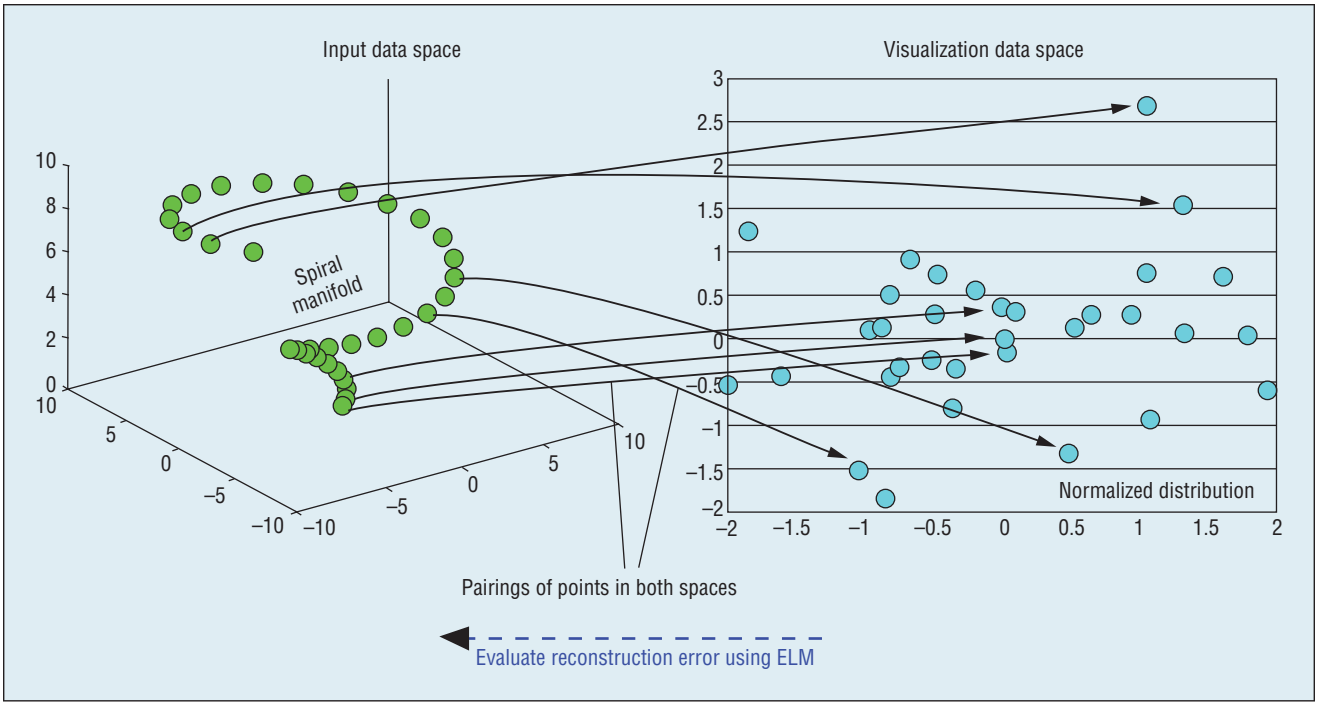

Figure 6. Projecting a high-dimensional spiral manifold data x_i to a lower-dimensional visualization space points v_i. Visualization points are fixed, and only the pairings (stored in an ordering matrix *O*) of the original and visualization data samples are changed.

Several such swaps constitute for an update:

$$\mathbf{V}^{\text{iter}} \leftarrow \mathbf{V}^{\text{iter}-1}\mathbf{O}^{\text{iter}}. \quad (1)$$

ELMVIS starts by initializing N visualization space points $\mathbf{v}_i$, taken either from a Gaussian distribution or from a regular grid. Then an ELM is initialized, and the ordering matrix $\mathbf{O}$ is set to an identity matrix. An initial reconstruction MSE is calculated, after which an iteration starts by choosing a random number of samples out of N and permuting the corresponding rows of $\mathbf{O}$. The ordering matrix $\mathbf{O}$ is applied to visualization points by multiplication, which permutes the prototypes $\mathbf{V}$ in the same way. The reconstruction error is recalculated: if it increases, the permutation of rows of $\mathbf{O}$ is rolled back; new iteration begins by again choosing a number of samples and permuting the corresponding rows in $\mathbf{O}$. Convergence is achieved once the error attains a desired threshold or the iteration limit is reached.

Adapting the ELM for Data Visualization

The direct data visualization algorithm requires recalculation of the whole ELM. The most computationally costly part is a recalculation of matrix $\mathbf{H}$ and its pseudo-inverse $\mathbf{H}^{\dagger}$. For changes in $\mathbf{V}$, the whole ELM needs recalculating, but for changes in $\mathbf{X}$, the points $\mathbf{V}$ and a hidden layer representation $\mathbf{H}$ can remain constant, so only the output weight matrix needs to be updated.

The reconstruction mean squared error

$$MSE_{\text{rec}} = \frac{1}{ND}\sum_{i=1}^{N}\sum_{j=1}^{D}\left(\hat{x}_{ij} - x_{ij}\right)^2$$

depends on the $\hat{x}_i$, which is an output of an ELM, trained using data pairs ($\mathbf{v}_i$, $\mathbf{x}_i$). But the solution of the ELM is a linear system of equations, and the nonlinear part of the ELM is applied to each transformed input vector separately of the others. So the nonlinear mapping of an ELM is independent of the order of training pairs ($\mathbf{v}_i$, $\mathbf{x}_i$), as is the MSE_{rec}.

This fact lets us adapt the ELM in ELMVIS to cut the computational load. Multiplying an ordering matrix $\mathbf{O}$ with either $\mathbf{V}$ or $\mathbf{X}$ yields exactly the same new pairs ($\mathbf{v}'_i$, $\mathbf{x}'_i$), although their order will differ. But because the reconstruction error doesn't depend on a particular ordering of the pairs, these operations are interchangeable. Our proposed adaptation of the ELM thus consists of replacing changes in $\mathbf{V}$ by changes in $\mathbf{X}$, as in Equation 2:

$$(\mathbf{X}^{\text{iter}} \leftarrow \mathbf{X}^{\text{iter}-1}\,\mathbf{O}^{\text{iter}}) \Leftarrow (\mathbf{V}^{\text{iter}} \leftarrow \mathbf{V}^{\text{iter}-1}\,\mathbf{O}^{\text{iter}}). \quad (2)$$

In the ELM structure, replacing changes in $\mathbf{V}$ with changes in $\mathbf{X}$ will keep the matrices $\mathbf{H}$ and $\mathbf{H}^{\dagger}$ constant. They need to be calculated only once on initialization; during iterations, the reconstruction of $\mathbf{X}$ is obtained using the following rule:

$$\hat{\mathbf{X}} = \mathbf{H}\beta = \mathbf{H}\left(\mathbf{H}^{\dagger}\mathbf{X}\right) = \left(\mathbf{H}\mathbf{H}^{\dagger}\right)\mathbf{X}. \quad (3)$$

Denoting a new matrix $\mathbf{H}^2 = \mathbf{H}\mathbf{H}^{\dagger}$ and calculating it at the initialization, the training of the ELM on each iteration is reduced to a single matrix

Table 4. MSE of reconstruction on all datasets. The best error of 100 restarts is shown for all methods except PCA, due to a random initialization procedure.

Dataset	PCA	SOM	NeRV	ELMVIS (Gaussian)	ELMVIS (PCA)
Spiral	0.482	0.054	0.011	0.049	0.060
Sculptural faces	0.980	0.916	0.769	0.718	0.724
Real faces	0.724	0.511	0.501	0.462	0.449

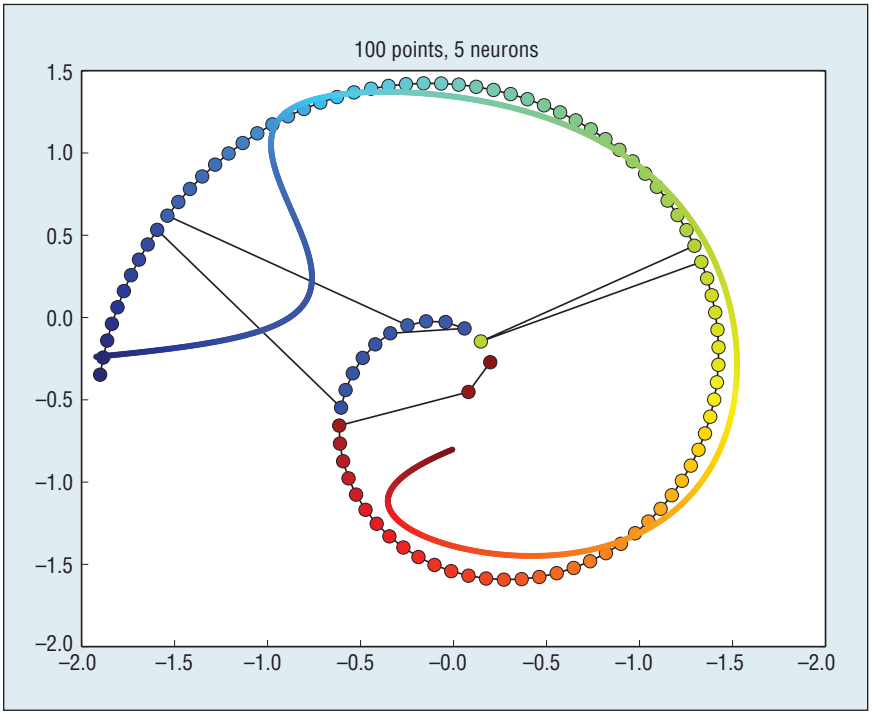

Figure 7. An example of ELMVIS fitting the spiral data. The thinner color line is a back projection of the ELM; black lines and color gradient denote the ordering of points. Some points are mapped incorrectly because the solution isn't exact.

multiplication. This gives the necessary speed to run hundreds of thousands or even millions of iterations within a few minutes.

Experimental Results

The ELMVIS visualization methodology was tested on three datasets. The selected reference methods are PCA as the baseline, SOM[2] as another method that uses fixed visualization points, and NeRV[3] as a state-of-the-art nonlinear visualization method.

The primary comparison uses reconstruction error that's an MSE of a reconstruction of the original data. A visualization method is assumed to have good performance if its visualization has a low MSE_{rec}. Reverse projection of visualized data to the original space is required to obtain the error; for NeRV, the only method that doesn't provide such projection, the reverse projection is learned by using a separate ELM. Table 4 lists the errors for all methods.

The first dataset for testing is a spiral toy dataset, a common and relatively hard benchmark. The spiral is drawn in a 2D space, and the goal is to project it into one dimension. It consists of $N = 100$ points, distributed evenly along its line by including a squared root term into the input data $\mathbf{X}$ equation:

$$\mathbf{X} = \begin{pmatrix} 2\sqrt{\alpha}\cos(\pi L\sqrt{\alpha}) \\ 2\sqrt{\alpha}\sin(\pi L\sqrt{\alpha}) \end{pmatrix}, \quad (4)$$

where α is distributed evenly between 0 and 1; L determines the amount of swings the spiral makes and is set to 3 in the experiment. The visualization points $\mathbf{V}$ are evenly distributed on a line, and both $\mathbf{X}$ and $\mathbf{V}$ are normalized to have zero mean and unit variance. In this experiment, the amount of neurons of the ELM and SOM is set to 5. Figure 7 shows the ELMVIS model and data mapping; Figure 8 shows a reconstruction learned from NeRV results.

The PCA projection squashes the second dimension of a spiral along the direction of the largest variance. NeRV succeeded in finding a manifold, showing great results even after estimating its mapping by a separate ELM. SOM showed good results as well. ELMVIS partially unfolds the spiral, but some parts remain torn and misplaced. Also, eventual outliers appear because the random permutation algorithm hasn't found the best solution in a given range of iterations. Still, the results of ELMVIS on a spiral dataset are acceptable, far better than the naive PCA.

We also tested the experimental convergence speed of ELMVIS; the spiral test is the fastest of the three due to a smaller number of neurons and lower original data dimensionality, while convergence speed is independent of these values and only relies on the amount of test points. Note that the graphs here represent averages over many runs; other results of ELM runs show the best outcome, corresponding to the best random initialization of a hidden layer of that ELM.

As stated earlier, complexity of the exact solution of ELMVIS is factorial in the number of points. The real

speed of convergence was estimated on different-sized subsets of the spiral data, ranging from 20 to 100 points. For each separate amount of points, 100,000 training steps were performed, and experiments restarted 100 times with different initial pairings. Figure 9 shows the obtained convergence plot with average values and some standard deviations.

Variance in ELMVIS convergence is explained by the convergence speed: while all the individual runs tend to the same lower bound, best cases converge very quickly, and worst cases spend much time on MSE plateaus seeking a better solution. For 50 points, convergence is reached on average at iteration 60,000, which is far less than the factorial of 50. The results show that the real convergence speed remains feasible for applications with a low to medium amount of data samples.

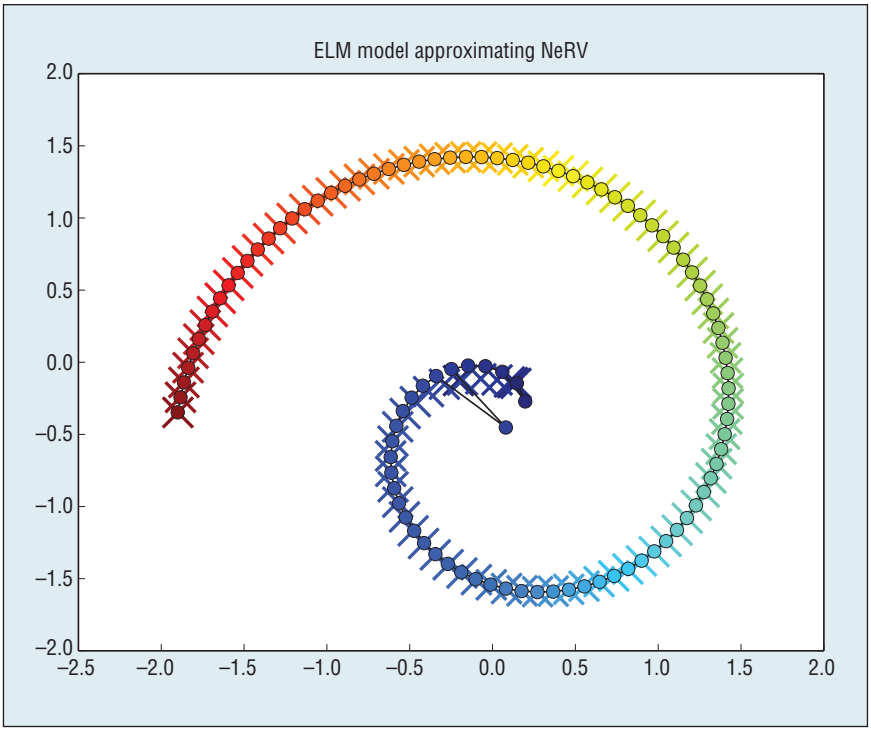

Figure 8. ELM reconstruction, learned from NeRV results. Only one point deviates from the perfect approximation. The ELM model printed with crosses is for visibility, as it mostly coincides with the data.

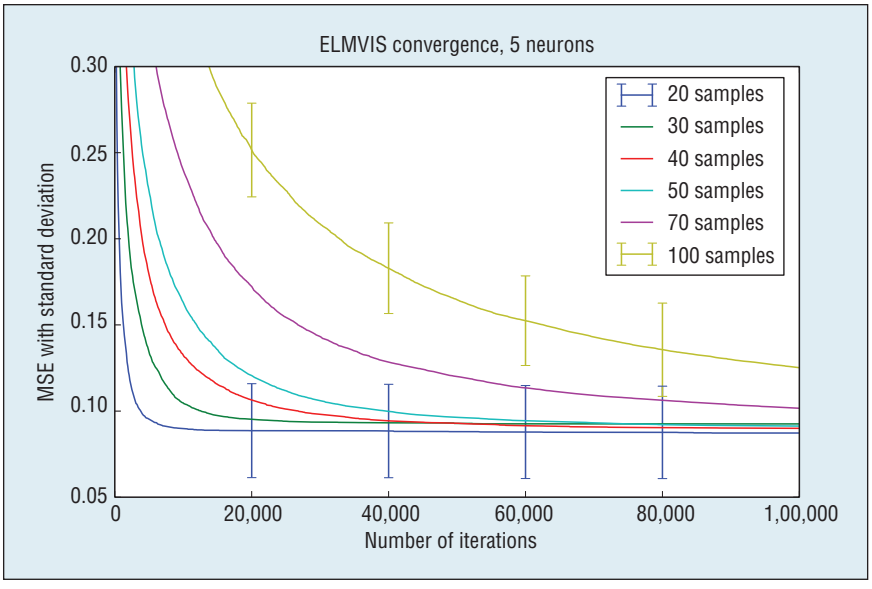

Figure 9. Convergence of the ELM visualization algorithm on the spiral dataset, with 100,000 training steps and 100 restarts. Plots are ordered from 20 samples (lowest) to 100 (highest). Only some standard deviations are shown.

The ELMVIS method is most suitable for the purposes of visualization of complex data or data without a simple manifold. Another benefit of the ELM is the presence of the reverse projection, which can be used to check how visualization space areas correspond to the data space ones. Using PCA for initialization didn't prove useful—points from a simple Gaussian distribution proved to be a better alternative.

References

1. J.A. Lee and M. Verleysen, *Nonlinear Dimensionality Reduction*, Springer, 2007.
2. T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, no. 1, 1982, pp. 59–69.
3. J. Venna et al., "Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization," *J. Machine Learning Research*, vol. 11, 2010, pp. 451–490.
4. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, no. 1, 2006, pp. 489–501.
5. G.-B. Huang, L. Chen, and C.-K. Siew, "Universal Approximation Using

Incremental Constructive Feedforward Networks with Random Hidden Nodes," *IEEE Trans. Neural Networks*, vol. 17, no. 4, 2006, pp. 879–892.
6. C.R. Rao and S.K. Mitra, *Generalized Inverse of a Matrix and Its Applications*, J. Wiley, 1971.

Anton Akusok is a PhD student in the Department of Information and Computer Science at Aalto University, Finland. Contact him at anton.akusok@aalto.fi.

Amaury Lendasse is a docent in the Department of Information and Computer Science at Aalto University, Finland, and also affiliated with IKERBASQUE, Basque Foundation for Science, Computational Intelligence Group, Computer Science Faculty, University of the Basque Country, and Arcada University of Applied Sciences. Contact him at amaury.lendasse@aalto.fi.

Francesco Corona is a docent in the Department of Information and Computer Science at Aalto University, Finland. Contact him at francesco.corona@aalto.fi.

Rui Nian is an associate professor in the College of Information and Engineering at Ocean University, China. Contact her at nianrui_80@163.com.

Yoan Miche is a postdoctoral researcher in the Department of Information and Computer Science at Aalto University, Finland. Contact him at yoan.miche@aalto.fi.

Combining ELMs with Random Projections

Paolo Gastaldo and Rodolfo Zunino, *University of Genoa, Italy*
Erik Cambria, *MIT Media Laboratory*
Sergio Decherchi, *Italian Institute of Technology, Italy*

In the extreme learning machine (ELM) model,[1] a single-layer feed-forward network (SLFN) implements inductive supervised learning by combining two distinct components. A hidden layer performs an explicit mapping of the input space to a feature space; the mapping isn't subject to any optimization, since all the parameters in the hidden nodes are set randomly. The output layer includes the only degrees of freedom—that is, the weights of the links that connect hidden neurons to output neurons. Thus, training requires solving a linear system by a convex optimization problem. The literature has proven that the ELM approach can attain a notable representation ability.[1]

According to the ELM scheme, the configuration of the hidden nodes ultimately defines the feature mapping to be adopted. Actually, the ELM model can support a wide class of activation functions. Indeed, an extension of the ELM approach to kernel functions has been discussed in the literature.[1]

Here, we address the specific role played by feature mapping in the ELM. The goal is to analyze the relationships between such feature mapping schema and the paradigm of *random projection* (RP).[2] RP is a prominent technique for dimensionality reduction that exploits random subspaces. This research shows that RP can support the design of a novel ELM approach, which combines generalization performance with computational efficiency. The latter aspect is attained by the RP-based model, which always performs a dimensionality reduction in the feature mapping stage, and therefore shrinks the number of nodes in the hidden layer.

ELM Feature Mapping

Let $\mathbf{x} \in \mathcal{R}^d$ denote an input vector. The function $f(\mathbf{x})$ of an output neuron in an ELM that adopts L hidden units is written as

$$f(\mathbf{x}) = \sum_{j=1}^{L} w_j \cdot a(\mathbf{r}_j \cdot \mathbf{x} + b_j) . \quad (1)$$

Thus, a set of random weights $\{\mathbf{r}_j \in \mathcal{R}^d; j = 1, \ldots, L\}$ connects the input to the hidden layer; the jth hidden neuron embeds a random bias term b_j and a nonlinear activation function $a(\cdot)$. A vector of weighted links, $\mathbf{w} \in \mathcal{R}^L$, connects the hidden layer to the output neuron.

The vector quantity $\mathbf{w} = [w_1, \ldots, w_L]$ embeds the degrees of freedom in the ELM learning process, which can be formalized after introducing the following notations:

- $\mathbf{X}$ is the $N \times (d + 1)$ matrix that originates from the training set. $\mathbf{X}$ stems from a set of N labeled pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is the ith input vector and $y_i \in \mathcal{R}$ is the associate expected target value.
- $\mathbf{R}$ is the $(d + 1) \times L$ matrix with the random weights.

Here, by using a common trick, both the input vector $\mathbf{x}$ and the random weights $\mathbf{r}_j$ are extended to $\mathbf{x} := [x_1, \ldots, x_d, 1]$ and $\mathbf{r}_j \in \mathcal{R}^{d+1}$ to include the bias term.

Accordingly, the ELM learning process requires solving the following linear system:

$$\mathbf{y} = \mathbf{Hw}, \quad (2)$$

where $\mathbf{H}$ is the hidden layer output matrix obtained by applying the activation function, $a()$, to every element of the matrix:

$$\mathbf{XR}. \quad (3)$$

Equation 3 clarifies that in the ELM scheme in Equation 1, the hidden layer performs a mapping of the original d-dimensional space into an L-dimensional space through the random matrix $\mathbf{R}$, which is set

independently from the distribution of the training data. In principle, the feature mapping phase can either involve a reduction in dimensionality ($L < d$) or, conversely, remap the input space into in an expanded space ($L > d$).

Both theoretical and practical criteria have been proposed in the literature to set the parameter L.[1,3] This quantity is crucial because it determines the ELM's generalization ability. At the same time, it affects the eventual computational complexity of both the learning machine and the trained model. These aspects become critical in hardware implementations of the ELM model, where resource occupation is of paramount importance.

A few pruning strategies for the ELM model have been proposed in the literature to balance generalization performance and computational complexity.[3] The present work tackles this problem from a different perspective and proposes to exploit the fruitful properties of random projections. The approach discussed here applies RP to reduce the dimensionality of data; the study, however, opens interesting vistas on using RP to tune the basic quantity L as well.

Dimensionality Reduction by Using RP

RP is a simple and powerful dimension reduction technique that uses a suitably scaled random matrix with independent, normally distributed entries to project data into low-dimensional spaces. The procedure to get an RP is straightforward and arises from the Johnson-Lindenstrauss (JL) lemma.[2] The lemma states that any N point set lying in d-dimensional Euclidean space can be embedded into a r-dimensional space, with $r \geq O(\varepsilon^{-2}\ln(N))$, without distorting the distances between any pair of points by more than a factor $1 \pm \varepsilon$, where $\varepsilon \in (0, 1)$.

Over the years, the use of probabilistic methods greatly simplified the original JL proof, and at the same time led to straightforward randomized algorithms for implementing the transformation. In matrix notation, the embedding operation is expressed as

$$\mathbf{K} = \mathbf{XP}, \tag{4}$$

where $\mathbf{X}$ is the original set of N, d-dimensional observations, $\mathbf{K}$ is the projection of the data into a lower, r-dimensional subspace, and $\mathbf{P}$ is the random matrix providing an embedding that satisfies the JL lemma.

In principle, Equation 4 is a projection only if $\mathbf{P}$ is orthogonal; this ensures that similar vectors in the original space remain close to each other in the low-dimensional space. In very high-dimensional spaces, however, bypassing orthogonalization saves computation time without affecting the quality of the projection matrix significantly. In this regard, the literature provides a few practical criteria to build $\mathbf{P}$.[2]

RP-ELM

The ability of RP to preserve, approximately, the distances between the N data vectors in the r-dimensional subspace is a valuable property for machine learning applications in general.[4] Indeed, this property is the conceptual basis of the novel approach that connects the ELM feature mapping scheme in Equation 3 to the RP paradigm.

A new ELM model can be derived from Equation 1 if we set as hypotheses that L should be smaller than d and the mapping implemented by the weights $\mathbf{r}_j$ satisfies the JL lemma. Under these assumptions, the mapping scheme in Equation 3 always implements the dimensionality reduction process (as in Equation 4). In practice, we can take advantage of the properties of RP to obtain an ELM model that shrinks the size L of the hidden layer and reduces the computational overhead accordingly. The eventual model will be denoted as RP-ELM. The crucial point is that the JL lemma guarantees that the original geometry of the data is only slightly perturbed by the dimensionality reduction process;[2] indeed, the degradation grows gradually as L decreases (given d and N).[2]

In principle, the literature provides several criteria for the construction of a random matrix that satisfies the JL lemma. The present work focuses on matrices in which the entries are independent realizations of ± 1 Bernoulli random variables;[2] hence, matrix $\mathbf{R}$ in Equation 3 is generated as follows:

$$\mathbf{R}_{i,j} = \begin{cases} 1/\sqrt{L} & \text{with probability } 1/2 \\ -1/\sqrt{L} & \text{with probability } 1/2. \end{cases} \tag{5}$$

Richard Baraniuk and colleagues[2] showed that this kind of random matrix actually satisfies both the JL lemma and the *restricted isometry property*, thus bringing out a connection between RP and compressed sensing.

Experimental Results

The performance of the proposed RP-ELM model was tested on two binary classification problems (www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html): *colon cancer* and *leukemia*. The former dataset contains expression levels of 2,000 genes taken in 62 different samples; 40 samples refer to tumors. The latter dataset provides the expression levels of 7,129 genes taken over 72 samples; 25 samples refer to "acute lymphoblast leukemia" and 47 samples refer to "acute myeloid leukemia." The datasets share two interesting features: the number of patterns is very low, and the dimensionality of data is very high as compared with the number of patterns. In both cases, data are quite noisy because gene expression profiles are involved.

Table 5. Error rates scored by RP-ELM and standard ELM on the two binary classification problems.

Colon cancer			Leukemia		
	Error rate (%)			Error rate (%)	
L	RP-ELM	ELM	*L*	RP-ELM	ELM
10	38.7	38.7	35	25.0	40.3
20	40.3	35.5	70	27.8	31.9
30	43.5	45.2	105	47.2	27.8
40	32.3	45.2	140	30.6	33.3
50	29.0	50.0	175	37.5	37.5
60	37.1	48.4	210	25.0	37.5
70	37.1	40.3	245	27.8	40.3
80	29.0	37.1	280	31.9	36.1
90	29.0	43.5	315	31.9	30.6
100	25.8	40.3	350	38.9	33.3

The experimental session aimed to evaluate the ability of the RP-ELM model to suitably trade off generalization performance and computational complexity (that is, the number of nodes in the hidden layer). It's worth noting that the experiments didn't address gene selection. Table 5 reports on the results of the two experiments, and gives the error rates attained for 10 different settings of L. In both cases, the highest values of L corresponded to a compression ratio of 1:20 in the feature-mapping stage. The performances were assessed by adopting a leave-one-out (LOO) scheme, which yielded the most reliable estimates in the presence of limited-size dataset. Error rates were worked out as the percentage of misclassified patterns over the test set.

The table compares the results of the RP-ELM model with those attained by the standard ELM model. Results showed that, in both experiments, RP-ELM attained lower error rates than the standard ELM. Moreover, the RP- ELM performed comparably with approaches reported in the literature, in which ELM models included 1,000+ neurons and didn't adopt a LOO validation procedure.

Our theory showed that, by a direct implementation of the JL lemma, we can sharply reduce the number of neurons in the hidden node without affecting the generalization performance in prediction accuracy. As a result, the eventual learning machine always benefits from a considerable simplification in the feature-mapping stage. This allows the RP-ELM model to properly balance classification accuracy and resource occupation.

The experiments also showed that the proposed model can attain satisfactory performance. Further investigations will aim to confirm the effectiveness of the RP-ELM scheme by additional theoretical insights and a massive campaign of experiments.

References

1. G.-B. Huang et al., "Extreme Learning Machine for Regression and Multiclass Classification," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 42, no. 2, 2012, pp. 513–529.
2. R. Baraniuk et al., "A Simple Proof of the Restricted Isometry Property for Random Matrices," *Constructive Approximation*, vol. 28, no. 3, 2008, pp. 253–263.
3. G.-B. Huang, D.H. Wang, and Y. Lan, "Extreme Learning Machines: A Survey," *Int'l J. Machine Leaning and Cybernetics*, vol. 2, no. 2, 2011, pp. 107–122.
4. Y. Miche, B. Schrauwen, and A. Lendasse, "Machine Learning Techniques based on Random Projections," *Proc. of European Symp. Artificial Neural Networks – Computational Intelligence and Machine Learning*, 2010, pp. 295–302.

Paolo Gastaldo is an assistant professor at the University of Genoa, Italy. Contact him at paolo.gastaldo@unige.it.

Rodolfo Zunino is an associate professor at the University of Genoa, Italy. Contact him at rodolfo.zunino@unige.it.

Erik Cambria is an associate researcher at MIT Media Laboratory. Contact him at cambria@media.mit.edu.

Sergio Decherchi is a postdoc researcher at Italian Institute of Technology, Italy. Contact him at sergio.decherchi@iit.it.

Reduced ELMs for Causal Relation Extraction from Unstructured Text

Xuefeng Yang and Kezhi **Mao,** *School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore*

Natural language is the major intermediary tool for human communication. However, it's unstructured and therefore hard for computers to understand. In recent decades, knowledge extraction, which transfers unstructured language text into machine-understandable knowledge, has received considerable attention.[1,2] Knowledge can be categorized into descriptive and logic information, both of which are indispensable in knowledge expression. Think of the following example: *Jim is happy today because his favourite basketball team won the final.*

The descriptive information *Be*(Jim, Happy) and *Win*(Team, Final) don't make much sense without the causal relation *Because*. In the literature, most research focuses on descriptive information extraction, and research in logic information extraction is relatively rare. We focus here on extracting the logic level relation, namely, the causal relation from unstructured text.

In recent years, machine learning and semantic resources for causal relation extraction has been explored. Some researchers, for example,[3] extracted <NP1 verb NP2> syntactic patterns and then employed semantic constraints to classify candidates as causal or non-causal. Other work[4] modified this and used the C4.5 decision tree instead of simple constraints to perform classification for a question-and-answer application. One team[5] proposed a novel boundary feature extracted from WordNet to help semantic relation classification between nominals that contained causal relation. Another team[6] employed predefined syntactic patterns to extract candidates containing any of the four relators "because," "after," "as," and "since," and then classified the patterns using the bagging ensemble method.

Our study expands both the syntactic and semantic perspectives to cover purpose, explanation, condition, and intra-sentential explicitly marked causal relations. The larger coverage generates more candidate relations to classify, which requires a computationally efficient pattern classifier for both training and testing. In addition, among the generated candidate relations, only a small portion is causal, hence imbalance problem exists in both training and testing data. To address the computational efficiency problem and the imbalance data problem, we propose an ensemble with the extreme learning machine (ELM). This ensemble alleviates the imbalance problem,[7] and lets the ELM[8] address the computational efficiency requirement. The ELM is a newly developed learning paradigm for single-layer feed-forward neural networks, in which the weights from the input layer to the hidden layer are randomly assigned, while the weights from the hidden layer to the output layer are obtained using linear least square estimation. Because of its non-iterative nature, the ELM is computational efficient. Please note, our proposed algorithm isn't a simple combination of an ensemble technique with the ELM. We propose restricted boosting sampling to further enhance the ensemble's capability to handle the imbalance problem, while neuron selection/reduction helps reduce the ELM architecture and hence the computational cost for testing data. In the literature, several algorithms have been proposed to reduce hidden layer neurons,[9–12] but they use a set-based selection method and are computationally expensive due to their attempts at finding optimal or suboptimal neurons. Here, we use Fisher's ratio to measure and select hidden layer neurons.

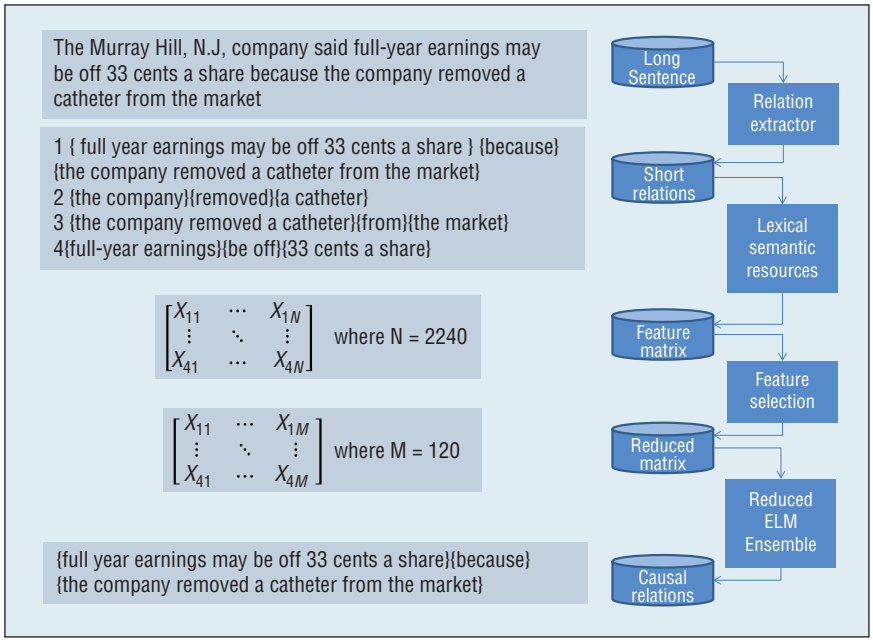

Figure 10. System architecture. The relation extractor is built on the Stanford Parser, which provides both a dependency relation format and a constituent tree format.

Figure 10 gives a full picture of our system. The relation extractor, built on the Stanford Parser, provides both a dependency relation format and a constituent tree format.

The extracted relations are categorized as either a verb or preposition type based on their *cue's* part of speech. Feature generation and the selection module combine various resources including named entity recognition tool, English syntactic knowledge, linguistic expert knowledge, and lexical semantic resources to generate candidate features and then select the informative ones. After this, every candidate relation is classified into causal or non-causal using our proposed ensemble of a reduced ELM classifier.

Ensemble of the Reduced ELM

Compared with non-causal relations, the causal relation is relatively rare. The data of causal class and non-causal class are often imbalanced, a problem that usually results in biased classifiers neglecting the minor class. In recent years, the ensemble technique has been used to alleviate this imbalance problem because the technique trains individual classifiers with balanced or less skewed data.

Table 6. Classification results of eight algorithms. Cs, RBO, BA, and MS denote cost sensitive, restricted boosting, bagging, and model selection, respectively.

Algorithm	F-score	G-mean	Accuracy
CsELM+RBO	**0.6637**	**0.8237**	**0.8869**
CsELM	0.6206	0.7472	**0.8915**
CsSvm+RBO	**0.6524**	**0.8316**	0.8764
CsSvm+BA	0.6311	**0.8327**	0.8606
Svm+RBO	0.6356	0.7784	**0.8879**
Svm+BA	0.6056	0.8031	0.8565
AdaBoost	0.5784	0.8237	0.8180
CsSvm+MS	**0.6420**	0.7840	**0.8894**

Table 7. Neuron selection reduces the neuron number without hurting performance. NS is neuron selection, CI is confidence interval, and Time is the time needed for one repeat of five-fold cross validation.

Data type	NS	Number	Time (s)	F	CI
prep	No	1000	16.35	0.6287	0.5907,0.6678
prep	No	2000	29.38	0.6325	0.5942,0.6708
prep	No	5000	69.16	0.6407	0.6032,0.6781
prep	Yes	1000	34.74	0.6363	0.5996,0.6729
verb	No	500	5.66	0.6578	0.6043,0.7113
verb	No	1000	10.15	0.6697	0.6272,0.7122
verb	No	2000	17.84	0.6812	0.6458,0.7169
verb	No	4000	37.84	0.6889	0.6505,0.7273
verb	Yes	500	34.98	0.6721	0.6219,0.7224
verb	Yes	1000	19.25	0.6765	0.6354,0.7177

AdaBoost[13] is the most widely used ensemble technique. Assuming that $p_i(k)$ denotes the predicted class label of the ith data by kth weak classifier at the kth iteration, and l_i denotes its true class label, then the total error is calculated as follows:

$$\dot{o}(\mathrm{k}) = \sum_{i=1}^{n} w_i(k) I(l_i \neq p_i(k)), \quad (1)$$

where function I is the indicator function whose output is 1 if inputs are equal and 0 otherwise, n is the number of data points, and $w_i(k)$ is the weight of ith data at the kth iteration. The weight is updated at each iteration as follows:

$$\alpha(\mathrm{k}) = \frac{1}{2} \ln \frac{1 - \dot{o}(\mathrm{k})}{\dot{o}(\mathrm{k})} \quad (2)$$

$$w_i(k+1) = w_i(k) \exp(-\alpha(k) l_i\, p_i(k)). \quad (3)$$

The weights are then normalized to make $\sum_i w_i\,(k+1) = 1$.

To further enhance the imbalance-handling capability of the ensemble technique, we propose restricted boosting in this study, with the goal of restricting the data's weight adjustment in the majority class. In restricted boosting, the error for minority and majority classes are calculated separately:

$$\dot{o}_1(\mathrm{k}) = \sum_{i=1}^{n} w_i(k) I(l_i \neq p_i(k)) I(l_i = 1) \quad (4)$$

$$\dot{o}_{-1}(\mathrm{k}) = \sum_{i=1}^{n} w_i(k) I(l_i \neq p_i(k)) I(l_i = -1). \quad (5)$$

The weights for data of the minority class and the majority class are then adjusted as in Equations 2 and 3 based on their respective error. By this restricted boosting, the base classifier won't give the majority class more attention than the minority class.

An ensemble of classifiers can consist of a large number of base classifiers. The training of those base classifiers must be computationally efficient, so we use the ELM for that purpose. Because of the random assignment and the linear least estimation of weights, the training of the ELM is extremely fast, but due to the random assignment of weights, the ELM usually demands a relatively large number of hidden layer neurons, which harms its computational efficiency for testing data. To deal with this problem, researchers have proposed using the ELM with neuron selection,[9–12] which aims to pick the best subset of neurons in a randomly projected large neuron set. However, these set-based selection methods are computationally intensive, which is why we use the individual-based neuron selection method, to improve neuron selection efficiency.

The role of a hidden layer neuron is to map data from the original feature space into a new dimension in which data of different classes are separable. Thus, the importance of a hidden layer neuron can be evaluated based on its capability to provide large class separation in the new dimension. Assuming that the hidden layer neuron j maps data to a new dimension z_j, on which the means of data of two classes are μ_1^j and μ_2^j respectively, and the standard deviations are σ_1^j and σ_2^j respectively, the class separation provided by the hidden layer neuron j can be measured by Fisher's ratio, which is defined as follows:

$$F_j = \frac{\left\| \mu_1^j - \mu_2^j \right\|^2}{(\sigma_1^j)^2 + (\sigma_2^j)^2}. \quad (6)$$

Neurons providing large class separation are retained, while those providing

little or no class separation are removed. Due to its nature of individual neuron selection, the neurons selected by Fisher's ratio are non-optimal, but this non-optimality is wanted because the ensemble requires weak classifiers.

Experiment

For high-level semantic relation extraction, data is very expensive. In this study, we labeled 300 sentences from Propbank[14] based on Matthew Hausknecht's annotation. The relation extractor extracted 1,683 relations, of which 280 are causal.

We conducted two experiments to test the proposed algorithm. The first tested the method's capability to deal with imbalanced data compared with other sampling methods, and the second evaluated the capability of the proposed neuron selection algorithm to reduce the ELM architecture. The performance of the ensemble of the original ELM and the ensemble of the reduced ELM were compared for an equal number of neurons. The results are based on 100 repeats of five-fold cross validation.

Table 6 lists the results of the first experiment. Apparently, the best *F*-score and *G*-mean are obtained by combining the ELM and restricted boosting. Compared with the original AdaBoost and other sampling methods, the proposed restricted boosting improves both the accuracy and *F* score. It is also observed that the ELM outperforms SVM in this application.

Table 7 gives the results of the second experiment. The *F* score shows that the ensemble of the reduced ELM with 1,000 neurons outperforms the ensemble of the ELM with 2,000 random neurons, while the Time column shows that the time needed for one repeat of five-fold cross validation is similar. The results in Table 7 also verify that simple individual-based neuron selection can significantly cut down the number of neurons and hence the computational cost for testing data, with little performance loss. In addition, the smaller confidence interval indicates that the reduced ELM is more robust than the original ELM.

The restricted boosting and neuron selection algorithm effectively addresses the concerns of imbalanced data and computational efficiency in causal relation extraction. Our proposed method has been tested using a real problem of knowledge extraction.

References

1. D. Wimalasuriya and D. Dou, "Ontology-Based Information Extraction: An Introduction and a Survey of Current Approaches," *J. Information Science*, vol. 36, no. 3, 2010, pp. 306–323.
2. E. Cambria, T. Mazzocco, and A. Hussain, "Application of Multi-Dimensional Scaling and Artificial Neural Networks for Biologically Inspired Opinion Mining," *Biologically Inspired Cognitive Architectures*, vol. 4, no. 0, 2013, pp. 41–53.
3. R. Girju et al., "Text Mining for Causal Relations," *Proc. FLAIRS Conf.*, AAAI Press, 2002, pp. 360–364.
4. R. Girju, "Automatic Detection of Causal Relations for Question Answering," *Proc. ACL 2003 Workshop on Multilingual Summarization and Question Answering*, Assoc. Computational Linguistics, 2003, pp. 76–83.
5. B. Beamer, A. Rozovskaya, and R. Girju, "Automatic Semantic Relation Extraction with Multiple Boundary Generation," *Proc. 23rd Nat'l Conf. Artificial Intelligence*, AAAI Press, 2008, pp. 824–829.
6. E. Blanco, N. Castell, and D. Moldovan, "Causal Relation Extraction," *Proc. 6th Int'l Language Resources and Evaluation*, European Language Resources Assoc., 2008, pp. 310–313.
7. M. Galar et al., "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 42, no. 4, 2012, pp. 463–484.
8. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, no. 1, 2006, pp. 489–501.
9. H.-J. Rong et al., "A Fast Pruned-Extreme Learning Machine for Classification Problem," *Neurocomputing*, vol. 72, no. 1, 2008, pp. 359–366.
10. Y. Lan, Y.C. Soh, and G.-B. Huang, "Random Search Enhancement of Error Minimized Extreme Learning Machine," *European Symp. Artificial Neural Networks*, European Neural Network Soc., 2010, pp. 327–332.
11. Y. Lan, Y.C. Soh, and G.-B. Huang, "Constructive Hidden Nodes Selection of Extreme Learning Machine for Regression," *Neurocomputing*, vol. 73, no. 16, 2010, pp. 3191–3199.
12. Y. Miche et al., "Op-elm: Optimally Pruned Extreme Learning Machine," *IEEE Trans. Neural Networks*, vol. 21, no. 1, 2010, pp. 158–162.
13. Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting," *Computational Learning Theory*, Springer, 1995, pp. 23–37.
14. P. Kingsbury and M. Palmer, "From Treebank to Propbank," *Proc. 3rd Int'l Conf. Language Resources and Evaluation*, Citeseer, 2002, pp. 1989–1993.

Xuefeng Yang is a PhD candidate in the School of Electrical and Electronic Engineering at Nanyang Technological University, Singapore. Contact him at yang0302@e.ntu.edu.sg.

Kezhi Mao is an associate professor in the School of Electrical and Electronic

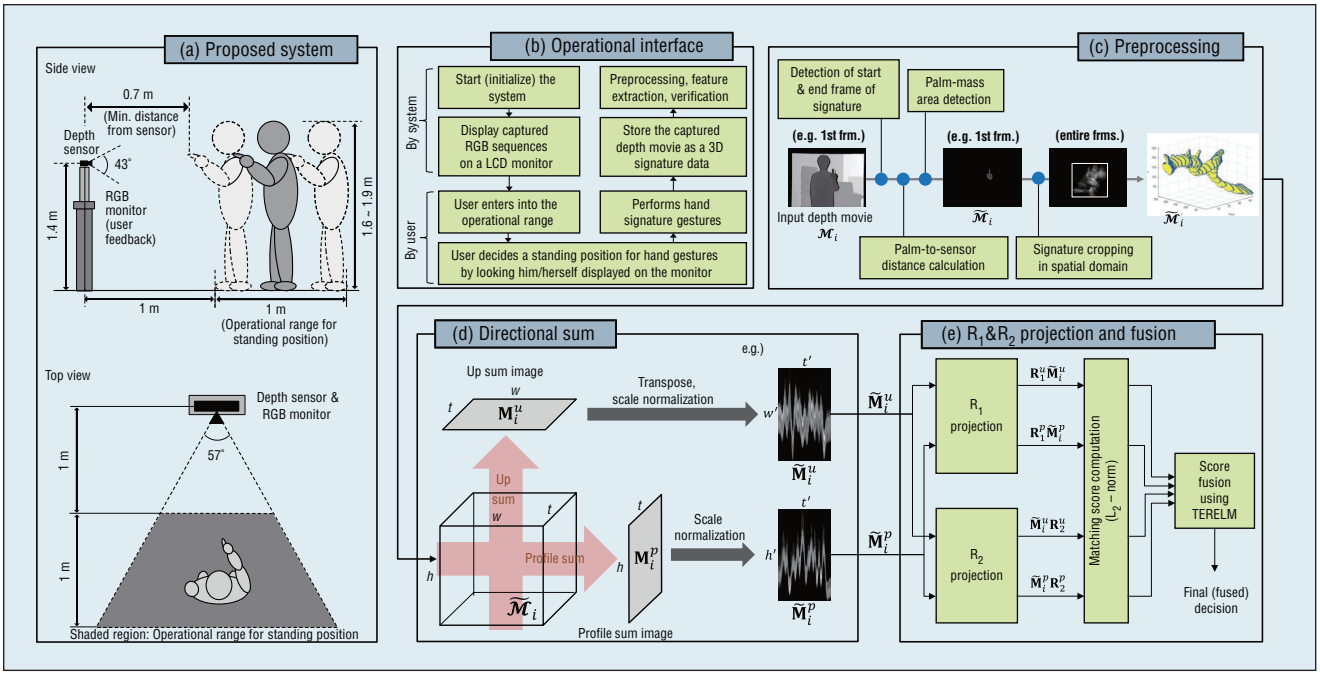

Figure 11. A flow diagram of the proposed hand gesture signature verification system. (a) and (b) The user's hand signature is captured using a depth sensor and stored as a video sequence, (c) Each sample is preprocessed and (d) represented by a set of directional features. (e) Finally, the obtained match scores are fused using TERELM.

Engineering at Nanyang Technological University, Singapore. Contact him at EKZMao@ntu.edu.sg.

A System for Signature Verification Based on Horizontal and Vertical Components in Hand Gestures

Beom-Seok Oh, Jehyoung Jeon, Kar-Ann Toh, Andrew Beng Jin Teoh, and Jaihie Kim, *School of Electrical and Electronics Engineering, Yonsei University, Korea*

Due to its ease of use and behavioral uniqueness, the signature has played an important role in personal identification since the dawn of civilization. The most frequently and widely used form of signature is either a written version or a stamp that uses a seal, both of which have drawbacks. First, once a signature is written or stamped on a document, it's revealed to anyone who can access that document. This opens a vulnerability to forgery. Second, both have limitations in terms of remote authentication. To authenticate a handwritten signature on a document, the signers have to be physically present during signature acquisition.

Recent research[1,2] proposes a new paradigm for signature biometry: a user holding a positional sensor or wearing a glove with markers attached performs his signature in the air instead of on a surface. Because of this interface's contactless nature, no trace of signature is left for forgery, and the signers don't need to be physically present. However, existing in-air systems are rather limited. Holding a positioning sensor such as a smartphone for in-air signature isn't natural; the range of wrist usage is rather narrow, which limits hand gestures.

Here, we propose an in-air hand gesture signature verification system that doesn't require a handheld device. A depth image sensor captures signature gestures and records each signature as a 3D volume. A structured projection[3] is then applied to the directionally accumulated images for feature extraction. Subsequently, these features are fused for possible performance enhancement. The total error rate minimization of extreme learning machine (TERELM)[4] was adopted for fusion due to its classification-goal-driven learning without the need of an iterative search.

Proposed System

Figure 11 shows the configuration of our prototype system for hand gesture signature verification. As illustrated in the figure, a depth sensor (Microsoft Kinect) was placed at 1.4 m above an LCD monitor that displays an RGB movie taken by the sensor for real-time user feedback. The sensor height is determined to cover the upper-body motion of a user whose height falls between 1.6 and 1.9 m standing approximately 1 to 2 m away from the sensor. The user spreads his arm out toward the sensor to perform the intended hand signature gestures.

The signature data acquired using the prototype system contains not only the region of the body but also

noise such as imaging distortion and background clutter. We're particularly interested in the movement of the palm-mass region ("palm" is the targeted hand region that includes the palm, fingers, and back of palm) that forms the desired signature information. To segment the region of interest, four steps of preprocessing were performed on the acquired raw signature data as follows:

- Start and end of signature detection. Because there's no clear indication of when a user starts and ends a hand gesture signature, we manually detect them. The output of this process is signature movies $\widetilde{\boldsymbol{\mathcal{M}}}_i \in \mathrm{R}^{h\times w\times t}$, where $i = 1, ..., m$ denotes the number of samples, h and w respectively indicate the height and width of a depth frame, and t denotes time indexing, which equals the number of frames.
- Palm-to-sensor distance estimation. Because the user's hand is the closest object from the sensor, pixels that correspond to fingertips might have the smallest depth value. Moreover, our pre-analysis on signature data showed that the acquired hand gesture signatures are relatively consistent in terms of depth. With these in mind, we recorded the lowest depth value per frame in which their average $\bar{z}_i$ is used as an estimated distance between the palm-mass and the sensor.
- Palm-mass area detection. The next task is to segment the palm-mass area from each frame of $\widetilde{\boldsymbol{\mathcal{M}}}_i$. The size (number of pixels) of palm-mass area is estimated by a first-order exponential function defined as $n_i = \lfloor p_1 \times \exp(p_2 \times \bar{z}_i) \rfloor + \gamma$, where $\lfloor \cdot \rfloor$ is a floor function, p_1 and p_2 are variables of the first-order exponential function, $\bar{z}_i$ is the calculated palm-to-sensor distance of ith sample, and γ is an offset. The n_i number of pixels that correspond to the n_i lowest depth values are selected and utilized as a region of palm-mass. The output of this step is $\widetilde{\boldsymbol{\mathcal{M}}}_i$, which contains only palm-mass area.
- Signature cropping in spatial domain. Finally, a rectangular mask that covers the region of hand movement is applied on $\widetilde{\boldsymbol{\mathcal{M}}}_i$ to crop only the signature region out.

As shown in Figures 11c and 11d, the preprocessed signature data $\widetilde{\boldsymbol{\mathcal{M}}}_i$ is in the form of 3D volume. To efficiently extract necessary features, we adopt a summation of the volume data along the up (y-axis) and profile (x-axis) directions, respectively.

The upward summing of $\widetilde{\boldsymbol{\mathcal{M}}}_i$ generates a 2D signature image that's called an up-summed image $\mathbf{M}_i^u \in \mathbb{R}^{w\times t}$. This $\mathbf{M}_i^u$ exhibits the way the signature moves horizontally (see Figure 11d). In a similar manner, a profile summing of the volume yields another signature image called $\mathbf{M}_i^p \in \mathbb{R}^{h\times t}$. Through this accumulation, we can observe how the signature varies vertically.

Different signatures have different spatial size and time duration. To standardize the spatial image size and time duration, a simple image resizing technique that uses the bicubic interpolation is adopted. As a result of this step, $\mathbf{M}_i^u \in \mathbb{R}^{w\times t}$ and $\mathbf{M}_i^p \in \mathbb{R}^{h\times t}$ are normalized as $\widetilde{\mathbf{M}}_i^u \in \mathbb{R}^{w'\times t'}$ and $\widetilde{\mathbf{M}}_i^p \in \mathbb{R}^{h'\times t'}$, where w', h', and t' are the normalized width, height, and time sizes.

From both the up-summed image $\widetilde{\mathbf{M}}_i^u$ and the profile summed image $\widetilde{\mathbf{M}}_i^p$, we can observe how the user's hand moves horizontally and vertically. To extract directional information for verification, both sum images were projected onto two structured projection bases, such as horizontal projection basis matrix $\mathbf{R}_1$ and vertical projection basis matrix $\mathbf{R}_2$.[3]

Considering the conformation of matrix inner-product, the size of $\mathbf{R}_1$ projection matrix should be $k \times w'$ for pre-multiplication to $\widetilde{\mathbf{M}}_i^u$ (which we call $\mathbf{R}_1^u$), and $k \times h'$ for pre-multiplication to $\widetilde{\mathbf{M}}_i^p$ (denoted as $\mathbf{R}_1^p$), respectively. Here, the k indicates an arbitrary number of projection vectors. Similar to the $\mathbf{R}_1$ projection, the $\mathbf{R}_2^u \in \mathbb{R}^{t'\times k}$ projection matrix is post-multiplied to $\widetilde{\mathbf{M}}_i^u$, and the $\mathbf{R}_2^p \in \mathbb{R}^{t'\times k}$ projection matrix is post-multiplied to $\widetilde{\mathbf{M}}_i^p$ (see Figure 11e).

Here, $\mathbf{R}_1^u \widetilde{\mathbf{M}}_i^u \in \mathbb{R}^{k\times t'}$ and $\mathbf{R}_1^p \widetilde{\mathbf{M}}_i^p \in \mathbb{R}^{k\times t'}$ extract vertically compressed features of the hand position in horizontal and vertical direction. $\widetilde{\mathbf{M}}_i^u \mathbf{R}_2^u \in \mathbb{R}^{w'\times k}$ captures hand movement in the horizontal direction, and the feature matrix that results from $\widetilde{\mathbf{M}}_i^p \mathbf{R}_2^p \in \mathbb{R}^{h'\times k}$ contains information on how the hand moves along the vertical direction.

Experiments

To enhance the verification accuracy of individual features, the four projected features discussed above are fused at score level using TERELM.[4]

Database

We acquired a database of hand gesture signatures from 100 subjects. Each subject was briefly instructed about the proposed signature system and asked to perform his or her own 2D signature using a hand in the air. Participants performed the in-air signature 10 times, with each trial recorded as a movie sequence. The first five trial sequences per subject were used for system training, and the remaining five were used for performance evaluation.

Evaluation scenario

The goal of our experimental study is to observe our proposed signature system's feasibility for identity verification under three scenarios: individual

Table 8. Average EER (%) accuracy and CPU time (s) performance (elapsed for learning) benchmarking along the evaluation scenarios.

Scenarios	Feature / fusion type	Individual/ fusion algorithm	EER (%)	CPU time for learning (s)
Scenario 1, individual features	Projection features	R_1 on $\widetilde{\mathbf{M}}^u$ images	10.17	N/A
		R_2 on $\widetilde{\mathbf{M}}^u$ images	9.32	(no learning
		R_1 on $\widetilde{\mathbf{M}}^p$ images	7.72	required)
		R_2 on $\widetilde{\mathbf{M}}^p$ images	7.15	
	Trajectory features[5]	Fingertip position	7.27	
		Fingertip velocity	2.92	N/A
		Fingertip acceleration	10.48	(no learning
		Palm-mass center position	7.78	required)
		Palm-mass center velocity	3.04	
		Palm-mass center acceleration	5.92	
Scenario 2, unimodal fusion	Case 1: fusion of all projection features at score level	SVM (linear)	4.07	110.63
		SVM (Poly, order = 3)	3.37	102.59
		VM (RBF, $\sigma = 1$)	3.52	153.47
		ELM ($\widetilde{N} = 100$)	3.39	1.55
		TERELM ($\widetilde{N} = 50$)	3.43	0.16
	Case 2: fusion of all trajectories at feature level		2.10	N/A
Scenario 3, bimodal fusion	Fusion of all projected and trajectory features at score level (10 features)	SVM (linear)	0.72	13.17
		SVM (Poly, order = 3)	0.66	13.17
		SVM (RBF, $\sigma = 1$)	0.62	26.46
		ELM ($\widetilde{N} = 90$)	0.75	1.32
		TERELM ($\widetilde{N} = 100$)	0.63	0.29

features, unimodal fusion, and bimodal fusion. Under the first scenario, the proposed four projection features are evaluated in terms of accuracy. Beside the projection features, we also evaluate the discriminative power of six trajectory features[5] under the same experimental setup. The six trajectory features were extracted from both fingertip and palm-mass center trajectories.

Under the second scenario, we fused all four projection features at score level and all the six trajectory features at feature level, respectively. Under the third scenario, all four projection features and six trajectory features are fused at score level.

Evaluation protocols

To stabilize the palm-mass area detection, the exponential parameters $p_1 = 13,\ 910$, $p_2 = -0.001929$, and $\gamma = 495$ are found manually using the training set. The normalization ranges $w' = 97$, $h' = 69$, and $t' = 30$ were determined based on the minimum sizes of the entire training palm-mass area samples.

The $\mathbf{R}_1$ and $\mathbf{R}_2$ projections have two parameters, namely, projection size k and group size l.[3] In this work, we set $k = 100$, $l = 10$ for $\mathbf{R}_1^u$ and $\mathbf{R}_1^p$, and $l = 5$ for $\mathbf{R}_2^u$ and $\mathbf{R}_2^p$. These parameters were obtained based on 10 runs of two-fold cross-validation using only the training set.

For trajectory features, the fingertip and palm-mass trajectories are extracted from a signature data sample $\boldsymbol{\mathcal{M}}$.[5] From the trajectories, we also extracted velocity and acceleration features,[5] giving us six trajectory features in total. Dynamic time warping (DTW) is adopted for trajectory matching.

In scores fusion, verification accuracy and CPU time (elapsved for learning) performances of TERELM will be compared with that of the extreme learning machine (ELM)[6] and support vector machine (SVM)[7] using linear, polynomial (at different orders within the range {2, … , 6}), and radial basis function (RBF) (at different σ values selected within {0.1, 0.5, 1, 1.5, …, 5}) kernels. For the ELM and TERELM, different numbers of hidden nodes $\widetilde{N} \in \{10, 20, \ldots, 100\}$ are experimented. In this fusion performance benchmarking, only the best test performances among the evaluated parameter settings are reported. Following related work,[4] we set the threshold $\tau = 0$ and offset $\eta = 1$ for TERELM and normalized all the attributes into the range [0,1].

Results

Table 8 shows the average equal error rate (EER) over 30 runs using 30 different $\mathbf{R}_1$ and $\mathbf{R}_2$ projection bases along with the investigated experimental scenarios. As shown in the table, $\mathbf{R}_1$ and $\mathbf{R}_2$ projections on profile-summed images $\widetilde{\mathbf{M}}^p$ show about 2.5 to 3 percent lower EER performance than that of up-summed images $\widetilde{\mathbf{M}}^u$. Among the four projections, $\mathbf{R}_1$ on $\widetilde{\mathbf{M}}^p$ shows the best EER performance, while $\mathbf{R}_1$ on $\widetilde{\mathbf{M}}^u$ gives the worst.

The best performance of trajectory features was observed in "Fingertip velocity," with "Fingertip acceleration" giving the worst performance. Generally, the palm-mass center features show better EER performances than that of fingertip features. This could be due to the extracted palm-mass center point being more stable than the extracted fingertip point.

Under Scenario 2, we observed verification performance enhancements as a result of information fusion. Particularly in Case 1, all the fusion results show about 3 to 4 percent lower EER performance than that of the best projection feature, the $\mathbf{R}_1$ on $\widetilde{\mathbf{M}}^p$. The three investigated fusion schemes appear to have similar accuracy performance. However, TERELM outperformed SVM and ELM in terms of learning speed. The main reason for the fast learning speed of ELM and TERELM is due to their noniterative solution; TERELM is seen to be slightly faster than ELM due to its split covariance with smaller sizes. In Case 2, the feature level fusion of trajectory features yields about 0.8 percent lower EER performance than that of the best trajectory feature, the palm-mass center velocity.

The last five rows of the table show the EER accuracy and CPU learning speed (in seconds) under Scenario 3. The three investigated fusion algorithms yield a similar range of 0.6 to 0.7 percent EER performances, about 1.4 to 1.5 percent lower EER values than that of the best unimodal fusion. Similar to Scenario 2, TERELM shows the faster learning among the three compared algorithms due to its split covariance computation.

Observations and Discussion

The up-summed images $\widetilde{\mathbf{M}}^u$ contain horizontal movements of users' hands while the hand movements in vertical direction are captured by the profile-summed images $\widetilde{\mathbf{M}}^p$. From Table 8, we observe that the $\mathbf{R}_1$ and $\mathbf{R}_2$ projections on $\widetilde{\mathbf{M}}^u$ produced better EER performances than that of $\widetilde{\mathbf{M}}^p$. From these clues, we conclude that summing signature volumes upward would be more beneficial than taking profile summation in terms of verification accuracy.

The table shows that the usage of palm-mass center features for identity verification yields better accuracies than that of using the fingertip features. This could be due to stability of the extracted features as mentioned previously. The table also reveals that the velocity feature contains the most discriminative information among the investigated trajectory features.

Under Scenarios 2 and 3, we observed performance enhancement resulting from information fusion. Particularly, the lowest learning cost was observed for TERELM over ELM and SVMs with similar performance enhancement over that of single modality.

Our experiments showed that the proposed signature system, with adequate features and parameters settings, can be used for identity verification.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (Grant number: 2012-0001306).

References

1. M. Katagiri and T. Sugimura, "Personal Authentication by Free Space Signing with Video Capture," *Proc. 5th Asian Conf. Computer Vision*, Japan: Asian Federation of Computer Vision Societies, 2002, pp. 350–355.
2. G. Bailador et al., "Analysis of Pattern Recognition Techniques for In-Air Signature Biometrics," *Pattern Recognition*, vol. 44, no. 10, 2011, pp. 2468–2478.
3. B.-S. Oh et al., "Combining Local Face Image Features for Identity Verification," *Neurocomputing*, vol. 74, no. 16, 2011, pp. 2452–2463.
4. K.-A. Toh, "Deterministic Neural Classification," *Neural Computation*, vol. 20, no. 6, 2008, pp. 1565–1595.
5. J.-H. Jeon et al., "A System for Hand Gesture based Signature Recognition," *Proc. 12th Int'l Conf. Control Automation Robotics & Vision*, IEEE, 2012, pp. 171–175.
6. G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, no. 1–3, 2006, pp. 489–501.
7. C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, 1998, pp. 121–167.

Beom-Seok Oh is a PhD candidate in the School of Electrical and Electronics Engineering at Yonsei University, Korea. Contact him at a-bullet@yonsei.ac.kr.

Jehyoung Jeon is an MS candidate in the School of Electrical and Electronics Engineering at Yonsei University, Korea. Contact him at jh.jeon@yonsei.ac.kr.

Kar-Ann Toh is a professor in the School of Electrical and Electronics Engineering at Yonsei University, Korea. Contact him at katoh@yonsei.ac.kr.

Andrew Beng Jin Teoh is an associate professor in the School of Electrical and Electronics Engineering at Yonsei University, Korea. Contact him at bjteoh@yonsei.ac.kr.

Jaihie Kim is a professor in the School of Electrical and Electronics Engineering at Yonsei University, Korea. Contact him at jhkim@yonsei.ac.kr.

An Adaptive and Iterative Online Sequential ELM-Based Multi-Degree-of-Freedom Gesture Recognition System

Hanchao Yu, Yiqiang Chen, and Junfa Liu, *Institute of Computing, Chinese Academy of Sciences, China*
Guang-Bin Huang, *School of Electrical and Electronics Engineering, Nanyang Technical University, Singapore*

Gesture recognition can be divided into online recognition, where the recognition model can adapt to new users automatically to get high recognition accuracy, and offline recognition, where the model fits well to users who have contributed to training samples but might not perform as well with new users. Recently, gesture recognition technology has become a research hotspot in human-computer interaction.[1] Zhou Ren and colleagues[2] proposed a gesture recognition system based on Kinect. The system used depth and skin color information to detect hand gestures from a messy environment

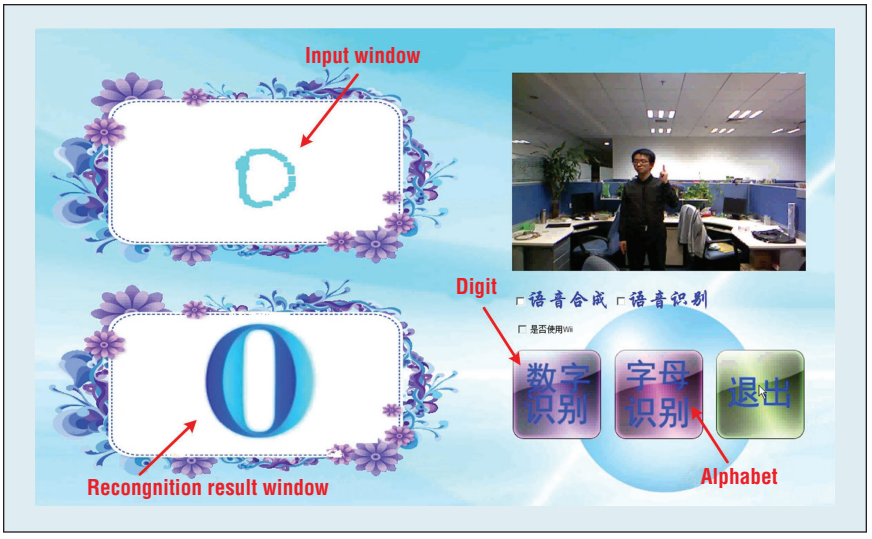

Figure 12. Online gesture recognition system interface. The upper left window displays the writing trace of users dynamically. The lower left window shows the recognition result as soon as the input finishes.

and Finger-Earth Mover's Distance for gesture recognition. To help someone communicate with a hearing- or speech-impaired person, M.K. Bhuyan and colleagues[3] presented a method for synthesizing hand gestures with the help of a computer and implemented a gesture animation framework for recognizing hand gestures. Mosiuoa Sole and colleagues[4] applied the extreme learning machine (ELM)[5] to classify static hand gestures that represent different letters of the Auslan dictionary.

These gesture recognition works are mainly for offline recognition. While in actual application, instead of working for users whose samples have been used in training, gesture recognition systems should recognize most users' gestures fast and accurately even if their samples weren't used in training beforehand. An online sequential learning framework might provide an efficient solution: they can learn from users' samples chunk by chunk and don't require all the data present at one time. Nan-Ying Liangand colleagues[6] proposed an online sequential variant of the ELM (OS-ELM). OS-ELM can process data in sequential form and update the existing model just by learning the newly arriving samples.

Here, we propose the adaptive and iterative online sequential ELM (AIOS-ELM), which executes multiple iterations to make full use of implied knowledge in each batch of incremental data. By introducing an adaptive mechanism and capitalizing on the original model's recognition ability, AIOS-ELM emphasizes the contribution of current data to the model, which can quickly improve its adaptive ability and thus improve the OS-ELM's generalization performance.

AIOS-ELM

We start by revising the parameter updating formula of OS-ELM:

$$\beta_{(k+1)} = \beta_{(k)} + \left(1 + \frac{B_{n+1}}{N_0 + \Sigma_{i=1}^{n} B_i}\right) \cdot K_{k+1}^{-1} H_{k+1}^{T} \left(T_{k+1} - H_{k+1}\beta_{(k)}\right). \quad (1)$$

In Equation 1, $\beta_{(k)}$ is the output weights linking the hidden nodes to the output nodes, and k is the index of the current model. N_0 represents the number of existing data in the system, β_i represents the amount of new user data for updating the model at the ith time, and n represents the number of batches of new user data in the system.

We adopted the Newton iterative method to update $\beta_{(k)}$. That is, every new batch of data needs to execute Equation 1 iteratively until meeting Equation 2:

$$|\beta_{(k+1)} - \beta_{(k)}| < \varepsilon, \quad (2)$$

where ε is a given minimum threshold. To keep the fast retraining speed, we limit the iterative times to 100. If the iterative execution doesn't meet Equation 2, and the iterative time reaches 100, we break the execution and use the $\beta_{(k+1)}$ as the ideal model.

Online Gesture Recognition System

Using Kinect and based on the AIOS-ELM, we developed an online gesture recognition system (OGRS) that can recognize contactless gesture inputs of 0–9 digits and a–z letters. Figure 12 shows the OGRS interface. The input window in the upper left displays the writing trace of users dynamically. The recognition result window at the lower left shows the recognition results as soon as the input finishes.

Figure 13 shows the OGRS framework, which includes gesture segmentation, data collection, fingertip tracking, feature extraction, digit/alphabet gesture recognition, and so on.

Gesture Segmentation

Skeleton and depth data are acquired through Kinect at a speed of 30 fps. Effective input gestures can be segmented out by feeling users' writing intention via Equation 3; effective input gestures are segmented out only when $\pi/6 \le \theta \le 5\pi/6$ and $0 \le h \le \left(\left|\overrightarrow{BA}\right| + \left|\overrightarrow{BC}\right|\right)/2$. OGRS only collects the depth data of segmented input gestures to do further processing:

$$\begin{cases} \theta = \arccos\left(\dfrac{\overrightarrow{BA}\cdot\overrightarrow{BC}}{\left|\overrightarrow{BA}\right|\cdot\left|\overrightarrow{BC}\right|}\right), \\ h = \left\|A_y - C_y\right\|_2 \end{cases} \quad (3)$$

where A, B, and C are the points of the wrist, elbow, and shoulder, respectively, and $|\overrightarrow{BA}|$ and $|\overrightarrow{BC}|$ are the vectors corresponding to points A, B, and C (A_y and C_y are the vertical coordinates of points A and C).

Fingertip Tracking

Based on the collected gesture data, fingertips can be detected accurately by the palm posture adaption-based robust single fingertip tracking method we described in our previous work.[7] By detecting and recording the moving trace of fingertips that's based on the effective gesture data, we can get the same dimensional data as recognition features by taking the interpolation and subsampling operations.

Gesture Recognition

OGRS uses the ELM to train the initial gesture recognition model based on the collected training data of digit/alphabet gestures; it then uses the AIOS-ELM to update models based on new users' gesture data.

It's worth mentioning that we designed a delete gesture that works by waving the other hand; it can delete incorrect input of users or incorrect recognition of the system. Inputs that aren't deleted are considered to be correctly labeled samples to be learned. Based on the labeled samples, OGRS uses the AIOS-ELM to implement the online learning by retraining the gesture recognition model whenever it receives new samples. OGRS can become more intelligent by frequently interacting with more users.

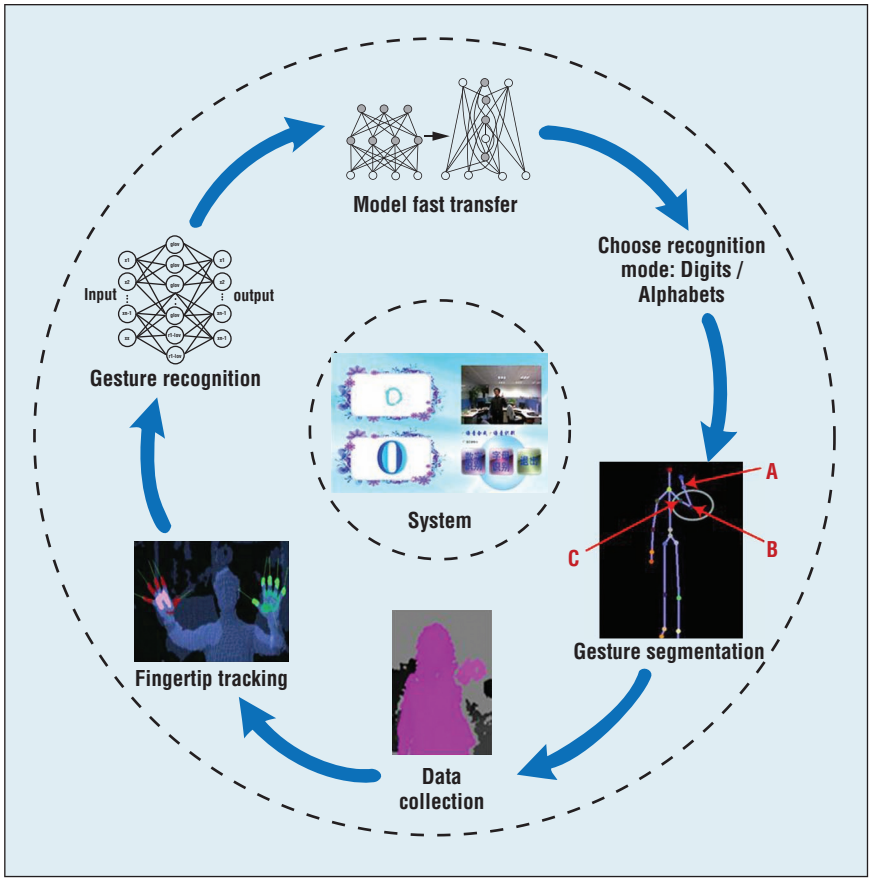

Figure 13. OGRS framework. It includes gesture segmentation, data collection, fingertip tracking, and so on.

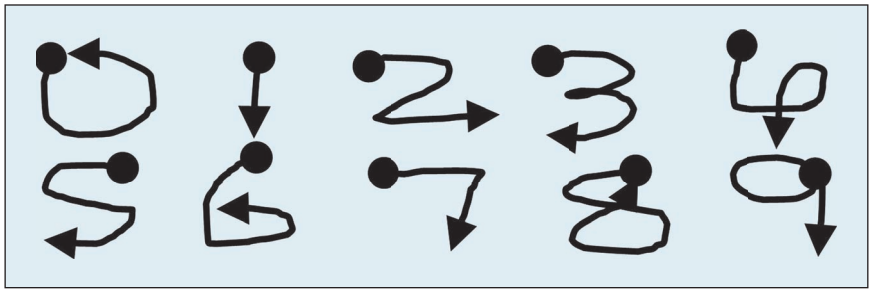

Figure 14. Samples of digit gestures.

Experiments and Results

We used the OGRS as an experiment system and samples of digit gestures 0–9 as experiment data. Figure 14 shows some samples of digit gestures from users. The experiments ran on a PC with Intel Core i5-2310 2.90-GHz processor, 4 Gbytes of RAM, and the Microsoft Windows Server 2008 operating system.

Data Source

We invited 21 users (11 males and 10 females) to use our OGRS. In practice, the system automatically collected users' writing trajectory information of digit gestures. We randomly selected 20 users' corresponding 3,000 gestures (2,700 as a training dataset, and 300 as a testing dataset) for the initial gesture recognition model, with each user accounting for 15 gestures for each digit. We treated the last person as a new user of the system, whose corresponding 500 gestures we selected as incremental training data were divided into 10 batches, and each digit accounts for five gestures in each batch of data. The other 300 gestures of the new user were selected

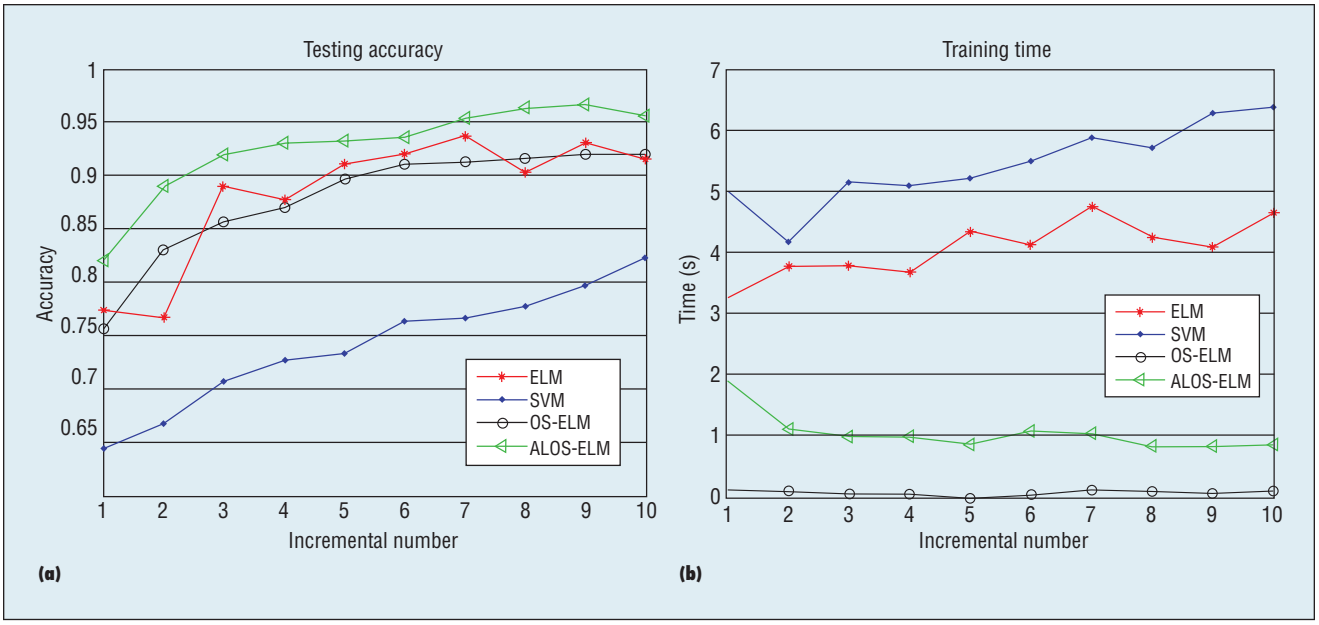

Figure 15. Incremental experiment results. Compare (a) the testing accuracy of each increment with (b) the training time of each increment.

Table 9. Initial gesture recognition models.

	Training dataset		Testing dataset		Incremental testing dataset	
Algorithm	**Accuracy**	**Time**	**Accuracy**	**Time**	**Accuracy**	**Time**
ELM	96.93%	3.69s	90.33%	0.04s	69.67%	0.04s
SVM	88.59%	4.52s	88.00%	0.58s	63.67%	0.58s

as an incremental testing dataset, and each digit accounts for 30 gestures.

Gesture Recognition Experiments

We validated the AIOS-ELM's performance by comparing it with SVM, ELM, and OS-ELM.

Initial gesture recognition models. We trained gesture recognition models by ELM and SVM with the training dataset, and then tested the initial gesture recognition model with the testing dataset and the incremental testing dataset to get testing accuracy and running time. The active function of the ELM was set as Sigmoid. We also set the amount of hidden nodes of ELM as 500, and chose the parameters c and g of SVM to be 1 and 0.06. Table 9 shows the results.

As Table 9 shows, the training and testing time for the ELM are both shorter than SVM, and the ELM's training and testing accuracy are both higher. But even though the ELM-generated gesture recognition model is faster and more accurate than SVM, the ELM can only get a testing accuracy of 69.67 percent for new users.

Incremental experiments. Based on the initial gesture recognition models, we set incremental times and used the ELM, SVM, OS-ELM, and AIOS-ELM to retrain the gesture recognition models with sequentially arriving training data.

As Figure 15 shows, the ELM is more accurate and faster than SVM in all incremental experiments, but OS-ELM and AIOS-ELM are much faster than the ELM in all incremental experiments because they used a sequential training mechanism, which means they didn't retrain with old data but updated the old model with newly arrived data. AIOS-ELM is also more accurate than the ELM and OS-ELM in all incremental experiments because it uses the adaptive weight punishment and iterative strategy, which makes it faster to adapt to new users and get higher gesture recognition accuracy by using less incremental time. Based on AIOS-ELM, the online gesture recognition system can reach a high accuracy of 96.7 percent within 10 sequential operations. AIOS-ELM needs to iterate Equation 1 in a sequential training process, which costs a little more time than OS-ELM, but it takes only about 1 second and doesn't affect the efficiency of an online gesture recognition system.

Our results show that based on AIOS-ELM, the gesture recognition system can support online lifelong learning for users and reach quick, high recognition accuracy for new user gestures.

Experiments confirm that our gesture recognition system using AIOS-ELM can quickly and accurately adapt to new users. ■

Acknowledgments

We thank Lei Zhang and Meiyu Huang for their constructive suggestions for this article.

References

1. S. Mitra, and T. Acharya, "Gesture Recognition: A Survey," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 37, no. 3, 2007, pp. 311–324.
2. Z. Ren et al., "Robust Hand Gesture Recognition with Kinect Sensor," *Proc. 19th ACM Int'l Conf. Multimedia*, ACM, 2011, pp. 759–760.
3. M.K. Bhuyan, V.V. Ramaraju, and Y. Iwahori, "Hand Gesture Recognition and Animation for Local Hand Motions," *Int'l J. Machine Learning and Cybernetics*, vol. 3, 2013, pp. 1–17.
4. M.M. Sole and M.S. Tsoeu, "Sign Language Recognition Using the Extreme Learning Machine," *Proc. 2011 IEEE Region 8 Flagship Conf. African Continent AFRICON 2011)*, IEEE, 2011, pp. 1–6.
5. G.B. Huang, Q.Y. Zhu, and C.K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, no. 1, 2006, pp. 489–501.
6. N. Y. Liang et al., "A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks," *IEEE Trans. Neural Networks*, vol. 17, no. 6, 2006, pp. 1411–1423.
7. H.C. Yu et al., "Robust Single Fingertip Tracking Method Based on Palm Posture Self-Adaption," *J. Computer-Aided Design & Computer Graphics*, vol. 25, no. 12, 2013, pp. 1793–1800.

Hanchao Yu is a PhD candidate in the Institute of Computing Technology at the Chinese Academy of Sciences, China. Contact him at yuhanchao@ict.ac.cn.

Yiqiang Chen is a professor in the Institute of Computing Technology at the Chinese Academy of Sciences, China. Contact him at yqchen@ict.ac.cn.

Junfa Liu is an associate professor in the Institute of Computing Technology at the Chinese Academy of Sciences, China. Contact him at liujunfa@ict.ac.cn.

Guang-Bin Huang is an associate professor in the School of Electrical and Electronic Engineering at Nanyang Technological University, Singapore. Contact him at egbhuang@ntu.edu.sg.