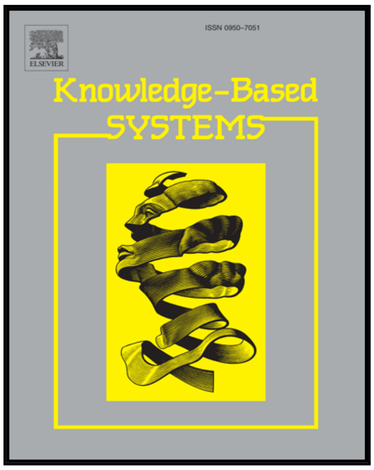


Optimizing area under the roc curve via extreme learning machines

Zhiyong Yang, Taohong Zhang, Jingcheng Lu, Dezheng Zhang,
Dorothy Kalui

PII: S0950-7051(17)30224-1
DOI: 10.1016/j.knosys.2017.05.013
Reference: KNOSYS 3911

To appear in: *Knowledge-Based Systems*

Received date: 23 November 2016
Revised date: 15 May 2017
Accepted date: 17 May 2017

Please cite this article as: Zhiyong Yang, Taohong Zhang, Jingcheng Lu, Dezheng Zhang, Dorothy Kalui, Optimizing area under the roc curve via extreme learning machines, *Knowledge-Based Systems* (2017), doi: 10.1016/j.knosys.2017.05.013

Highlights

- We propose a novel off-line binary AUC optimization algorithm called ELM^{AUC} by bridging the least square AUC optimization method with ELM framework.
- Two potential multi-class extensions of AUC are compared theoretically.
- A unified objective function for multi-class AUC optimization is proposed. Subsequently, two novel off-line algorithms named ELM_{M}^{AUC} and ELM_{macro}^{AUC} respectively are proposed for multi-class AUC optimization.
- The generalization analysis of ELM_{M}^{AUC} is established.
- The experimental results on 11 binary-class datasets and 15 multi-class datasets suffering from class imbalance problem show the effectiveness of our work.

Optimizing area under the roc curve via extreme learning machines

Zhiyong Yang[a,b], Taohong Zhang[a,b], Jingcheng Lu[a], Dezheng Zhang[a,b,*], Dorothy Kalui[a,b]

[a]*Department of Computer, School of Computer and Communication Engineering, University of Science and Technology Beijing (USTB), Beijing, 100083, China.*
[b]*Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, 100083, China.*

Abstract

Recently, Extreme learning machine (ELM), an efficient training algorithm for single-hidden-layer feedforward neural networks (SLFN), has gained increasing popularity in machine learning communities. In this paper the ELM based Area Under the ROC Curve (AUC) optimization algorithms are studied so as to further improve the performance of ELM for imbalanced datasets. For binary class problems, a novel ELM algorithm is proposed based on an efficient least square method. For multi-class problems, the following works are done in this paper: First of all, theoretical comparison analysis is proposed for the potential multi-class extensions of AUC; Secondly, a unified objective function for multi-class AUC optimization is proposed following the theoretical analysis; Subsequently, two ELM based multi-class AUC optimization algorithms called ELM_M^{AUC} and ELM_{macro}^{AUC} respectively are proposed followed with complexity analyses; Finally, the generalization analysis is established for ELM_M^{AUC} in search of theoretical supports. Empirical study on a variety of real-world datasets show the effectiveness of our proposed algorithms.

Keywords:
Extreme learning machine (ELM), Area Under the ROC Curve (AUC), Imbalanced Datasets, Multi-class AUC Optimization

1. Introduction

For classification problems, the overall accuracy has long been adopted as a standard evaluation metric. However, for class imbalance datasets, the overall accuracy is by no means a proper metric as the minor class is prone to be ignored. A well-known remedy for this metric is Area Under the ROC Curve (AUC) [1], which measures the performance based on the frequency of mis-ranking instance pairs. An early study [2] pointed out that, though, on average, AUC is a monotonic function of accuracy, the large variances of AUC for imbalanced datasets imply that two classifier sharing similar accuracy may exhibit significantly different AUC value. Such a result posed an encouraging sign for a comprehensive studies on straight-forward AUC optimization.

Unfortunately, direct AUC optimization is a $\mathcal{NP}$ hard problem as the ranking loss is discrete. In practice, ranking loss is often replaced with a convex and differentiable surrogate loss while AUC optimization is approximated by the corresponding convex optimization problem. During the past decades, a lot of related algorithms have been proposed to optimize AUC following such an idea [3–7].

Among all the proposed losses, the squared loss is the only one that could generate a closed-form solution while remain iteration-free at the same time. Moreover, aiming at further boosting the efficient of this algorithm, in Ref. [7], an efficient least square algorithm was proposed to speed up least square AUC optimizations by avoiding explicit calculation and storage of the Laplacian matrix. As for theoretical supports, recent studies [8, 9] have lent theoretical supports to square loss via showing the corresponding consistency with true AUC loss. Nonetheless, in the light of the fact that least square AUC optimization is only aimed at obtaining linear classifiers in nature, the model complexities are limited. Correspondingly, when dealing with complicated datasets, such classifiers may suffer from poor generalization performance due to high biases.

As for multi-class extensions of AUC, a simple generalization of AUC toward multi-class cases was proposed in Ref. [10], which we referred to as AUC_M through out this paper. AUC_M has been widely studied and applied thereafter. There exists a lot of related works towards AUC_M [11–16]. However, for the majority of the relevant works, AUC_M serves merely as an implicit metric but not an objective function. To the best of our knowledge, there are only two existed works that try to find classifiers with optimized AUC_M. One is called Evolutionary AUC Maximization (EAM) [17], where the AUC_M is optimized by evolutionary neural networks. While the other is named as Decomposition based Classification method (MDC) [18] where AUC_M is optimized by a modified RankBoost [19] algorithm with decision stumps being the weak learners. The experimental results of Ref. [18] show that MDC could significantly outperform EAM with higher efficiency. However, for MDC, in order to optimize AUC_M with k different classes, RankBoost has to be executed k times to get the final classifier. When k is large, a lot of weak learners have to be trained for a complete classifier. Consequently, for MDC, it is impractical to use more complicated weak learners than decision stump. Moreover, since MDC only serves as an ensemble method to optimize AUC_M, how to use a single learner to optimize AUC_M and its corresponding theoretical analysis still remains to be studied. Thus it is valuable to push the relevant studies a step further towards multi-class extensions of AUC optimization.

Among all the popular machine learning algorithms, a novel training algorithm for Single Hidden Layer Feedforward Network(SLFN) called Extreme Learning Machine (ELM) [20] is well-known for its efficiency and solid theoretic supports. Extreme learning machine(ELM) was first proposed to overcome the problems faced by traditional feedforward neural networks training algorithms such as BackProp. It is believed that the hidden layer parameters need not be tuned provided that the number of hidden layer nodes are sufficiently large. Due to the tuning-free nature of the hidden layer parameters, Extreme learning machine could be regarded as a least square method combined

*Corresponding author.
Email addresses: JZhiyongYoung@yahoo.com (Zhiyong Yang), dezzhangchina@yahoo.com (Dezheng Zhang)

with a random projection and a nonlinear transformation. These revisions leads to significant improvements of the corresponding training efficiency. Besides the practical improvements, the universal approximation capability for ELM was also proved in Ref. [21], which lends theoretical supports to the applications of ELM. Furthermore, though the randomization of hidden layer parameters may seemingly reduce the precision of BP algorithm, according to the most recent theoretical analyses [22, 23], the generalization performance of ELM is at least no worse than its counterparts with fine-tuned hidden layer weights, which further solidifies the theoretical backgrounds of ELM. The aforementioned achievements of ELM have attracted a lot of researchers from a wide range of domains. As a result, ELM algorithms have been successfully extended to a great deal of complicated machine learning tasks including online sequential learning [24, 25], learning with increasemental neurons [26, 27], architecture learning [28], sparse learning[29, 30], dynamic ensemble learning [31],multi-label learning[32, 33] multi-instance learning [34], manifold learning [35], learning with missing data [36], dimension reduction[37], deep learning [38], big data analysis[39–42], and learning to rank [43, 44]. Specific ELM applications for image processing [45–47], natural language processing [48–50], bioinformatics [51], indoor positioning [52–54] and so on have also been widely reported. Considering the limited space of this paper, the readers are referred to Ref. [55] for a more thorough review of the state-of-art ELM algorithms and their applications.

Many efforts have been made to adapt ELM algorithms to class imbalance datasets. Some of the relevant works focus on off-line algorithms. To name a few, two Weighted Extreme Learning Machines (WELM) [56] were first proposed to fit ELM better with class imbalance datasets; Then a Regularized Weighted Circular Complex Valued Extreme Learning Machine (RWCC_ELM) [57] is proposed to provide WELM with fully complex neurons; An improved boosting method aimed at ensembling WELM [58] was also proposed to learn adaptive weights for different instances. Some of the studies mainly deal with on-line sequential learning, for instance , a Weighted Online Sequential Extreme Learning Machine(WOS-ELM) was proposed in Ref. [59]; Based on WOS-ELM A framework to ensemble on-line sequential ELMs [60] was then proposed class imbalance learning from concept-drifting data streams; Most recently, a meta-cognitive online sequential extreme learning machine (MOS-ELM) [61] was also proposed for class imbalance and concept drift learning. While the other works deal with big data extensions [62, 63] and specific application for real-world problems [64, 65]. However, most of the existed works, especially those concerns off-line algorithms, were based on cost-sensitive learning with fixed cost matrices. For most cases, we don't have sufficient prior information of the cost distribution. So fixing the cost matrix may completely ignore the prior knowledge of cost matrix. Moreover, as addressed in Ref. [10], class imbalance learning problems and the cost-sensitive learning problems may happen simultaneously and sometimes may even lead to conflict results. It is thus not reasonable to fix the cost matrix merely by the virtue of class distribution. According to the analyses in Ref. [10], AUC could be regarded as an average precision measure of different cost choices rather than a specific choice. Moreover, AUC has been shown to be insensitive to both class skews and changes of class distributions [66]. It is just these two attractive properties that make AUC a good metric in point for imbalance datasets.

According to Ref. [67], least square AUC optimization problems could be regarded as a special case of the least square learning to rank problem. Meanwhile, there do exists ELM algorithms for learning to rank problems: Zong *etal*. [43] introduced Extreme Learning Machines to solve least square learning to rank problems; Chen *etal*. [44] carried out a thorough analysis of the generalization ability of Ranking based ELMs. However, none of these works focus on optimizing AUC, let alone the specific improvements.

As a result, it is necessary to bridge the advantages of extreme learning machines and least square AUC optimization algorithms.

After a total review of related works, our distinct contributions are highlighted as follows.

- We propose a novel off-line binary AUC optimization algorithm called ELM^{AUC} by bridging the work in Ref. [7] with ELM framework. Though ELM^{AUC} mainly comes from Ref. [7], certain revisions are done with proof of correctness.
- To justify AUC_M, it is compared theoretically with the macro average of binary AUCs.
- A unified objective function for multi-class AUC optimization is proposed. Subsequently, two novel off-line algorithms named ELM^{AUC}_M and ELM^{AUC}_{macro} respectively are proposed for multi-class AUC optimization.
- The generalization analysis of ELM^{AUC}_M is established Based on the mathematical technologies employed in Ref. [22, 44, 68, 69] .
- We implement systematic experiments based on a wide range of real world datasets suffering from class imbalance. The experimental results are analyzed with statistical hypothesis tests.

The rest of the paper is organized as follows. Section 2 clarifies the basic notations of this paper and the necessary preliminaries are reviewed. Section 3 discusses our proposed algorithms. Section 4 records the generalization analysis of ELM^{AUC}_M. Section 5 exhibits our experiment setting and corresponding results. Finally, in Section 6, we summarizes all valuable conclusions of this paper.

2. Preliminaries

2.1. Notations

In this paper, scalars, vectors and matrices are denoted like x (lowercase letters), $\boldsymbol{x}$ (bold lowercase letters) and $\boldsymbol{X}$ (bold uppercase letters) respectively. $\boldsymbol{X}_i$ denotes the ith row of X, while $\boldsymbol{X}^{(i)}$ denotes the i th column. Given an event A, the indicator function is denoted by $I[A]$, where $I[A] = 1$ if A is true and $I[A] = 0$ otherwise. $\mathbf{1}_{a\times b}$ stands for a matrix with a rows and b columns, where all of its elements are exactly 1. diag(a) represents a diagonal matrix with $diag(a)_{ii} = a_i$. $\mathbb{P}(A)$ denotes the probability of event A. $\mathbb{E}_x(\cdot)$ denotes the expectation with respect to a random variable x. All the empirical risk functions are denoted as $\mathcal{R}_A$ while all the expected risk functions are denoted as $\overline{\mathcal{R}}_A$, where the A describes the meaning of different risks. Moreover, all frequently used variables are listed in table 1.

2.2. AUC optimization based on square ranking loss

Given an instance space $\mathcal{X}$, a binary label space $\mathcal{Y} = \{-1, 1\}$, and $\mathcal{D}$ as the distribution of $(\boldsymbol{x}, \boldsymbol{y})$ i.e. the joint distribution of an instance and its corresponding label, the target of a classification algorithm is to learn a scoring function s as a map $\mathcal{X} \to \mathbb{R}$, where $s(\boldsymbol{x}_i)$ is in proportion to $\mathbb{P}(y = 1|x)$, namely the conditional probability of the label to be positive given its input feature.

The Area Under roc Curve(AUC) refers to the possibility that a randomly sampled positive instance has a higher score than a negative one, which could be formally conveyed as Eq.(1)

$$AUC = \mathbb{P}(s(x_1) > s(x_2)|y_1 = 1, y_2 = -1) \tag{1}$$

Table 1: mathematical nomenclature

notation	interpretation
m	number of instances within $\mathcal{S}$
$\mathcal{S}$	$S = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^m$ denotes a training dataset.
d	number of input features within $\mathcal{S}$
$\mathbf{X}$	$\mathbf{X} \in \mathbb{R}^{m\times d}$ is the input matrix of a training dataset
n_c	number of distinct class labels for a given dataset $\mathcal{S}$
$\mathbf{Y}$	$\mathbf{X} \in \mathbb{R}^{m\times n_c}$ is the label matrix of a training dataset
n_h	number of hidden layer neurons
$\mathbf{H}$	$\mathbf{H} \in \mathbb{R}^{m\times n_h}$ denotes the hidden layer features of all training instances
$h(\mathbf{x})$	the hidden layer features for instance $\boldsymbol{x}$
n_+	number of positive instances for a given training data
n_-	number of negative instances for a given training data
n_i	number of instances with the class label of which being i
$\mathcal{N}_i$	the index set of all the instances that belongs to the ith class.
$s(\centerdot)$	the scoring function
$s_i(\centerdot)$	the scoring function for the ith subtask (see Section 3.2 for explanations)
$l(\centerdot)$	A predefined surrogate loss function.

Unfortunately, for most of the practical problems, the distribution $\mathcal{D}$ is unknown.Thus AUC couldn't be calculated directly from Eq.(1). Let $\mathcal{S}$ be a dataset with m instances sampled from $\mathcal{D}$, i.e. $\mathcal{S} = (\boldsymbol{x}_i, y_i)_{i=1}^m \subset (\mathcal{X}\times\mathcal{Y})^m$. According to the large number laws,To address such dilemma, we could estimate AUC through replacing the possibility with its corresponding frequency on $\mathcal{S}$. The loss form of this estimation is denoted by $\overline{AUC}$, which is expressed as Eq.(2).

$$\overline{AUC} = \sum_{i\in\mathcal{N}_+}\sum_{j\in\mathcal{N}_-} \frac{I\left[s(\boldsymbol{x}_i) < s(\boldsymbol{x}_j)\right] + \frac{1}{2}I\left[s(\boldsymbol{x}_i) = s(\boldsymbol{x}_j)\right]}{n_+n_-} \tag{2}$$

where $I[A]$ is the indicator function : $I[A] = 1$ if and only if A is true, $I[A] = 0$ otherwise; $\mathcal{N}_+$ is the set of all the indexes of positive instances and $\mathcal{N}_-$ is the set of that of the negative instances; $n_+ = |\mathcal{N}_+|$; $n_- = |\mathcal{N}_-|$.

Though we could now estimate AUC on any dataset sampled from $\mathcal{D}$, seeing that the ranking loss :

$$I\left[s(\boldsymbol{x}_i) < s(\boldsymbol{x}_j)\right] + \frac{1}{2}I\left[s(\boldsymbol{x}_i) = s(\boldsymbol{x}_j)\right], x_i \in \mathcal{N}_+, x_j \in \mathcal{N}_- \tag{3}$$

is a discrete and non-differentiable function, straight-forward optimization of $\overline{AUC}$ is a $\mathcal{NP}$ hard problem. Practically, Eq.(3) is often approximated by a differentiable surrogate loss function $l(t)$. Substituting $l(t)$ into Eq.(2), we then obtain a surrogate empirical risk $\mathcal{R}_{AUC}$ function based on the training data $\mathcal{S}$.

$$\mathcal{R}_{AUC} = \sum_{i\in\mathcal{N}_+}\sum_{j\in\mathcal{N}_-} \frac{l(s(\boldsymbol{x}_i) - s(\boldsymbol{x}_j))}{n_+n_-} \tag{4}$$

In order to evaluate theoretical justifications of different loss functions Ref.[8] defines the AUC consistency for pair-wise surrogate loss, while Ref.[9] justifies the square loss by the following lemma and proposed an efficient on-line learning algorithm for the corresponding optimization problem as well.

Lemma 1. *The surrogate loss $l(t) = (1-t)^2$ is consistent with AUC.*

Substituting $l(t)$ defined in Lemma1 and $s(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{x}$ into Eq.(4), we then attain $\mathcal{R}_{AUCLS}$:the empirical risk based on square loss.

$$\mathcal{R}_{AUCLS} = \sum_{i\in\mathcal{N}_+}\sum_{j\in\mathcal{N}_-} \frac{\left(1 - (\boldsymbol{\beta}^\top(\boldsymbol{x}_i - \boldsymbol{x}_j))\right)^2}{n_+n_-} \tag{5}$$

Following the well-known structural risk minimization principle, least square AUC optimization could be expressed as the following problem.

$$\boldsymbol{P_{AUCLS}} : \min_{\boldsymbol{\beta}} \mathcal{R}_{AUCLS} + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2 \tag{6}$$

For Off-line learning, the square loss based AUC optimization is a special case of least-square learning to rank framework [70]. It is thus easy to obtain a closed-form solution for P_{AUCLS}.

2.3. Brief review of Extreme Learning Machine

ELM could be obtained directly from traditional training method of Artificial Neural Networks if hidden layer parameters are chosen randomly .The original ELM algorithm could be summarized as follows.

1. **Input**:
   - $\mathcal{D} = \{\boldsymbol{X}_i, \boldsymbol{Y}_i\}_{i=1}^m$: training data
   - $g(\cdot)$: piecewise continuous activation function
   - n_h : the numbers of hidden layer neurons
   - C : learning coefficient
2. Randomly generate input layer weight matrix $W \in \mathbb{R}^{d\times n_h}$ and the hidden layer bias $b \in \mathbb{R}^{n_h}$, where n_h is the number of hidden layer neurons.
3. Calculate the hidden layer features

$$\boldsymbol{H} = \begin{pmatrix} h_1(\boldsymbol{X}_1) & \cdots & h_{n_h}(\boldsymbol{X}_1) \\ \vdots & \ddots & \vdots \\ h_1(\boldsymbol{X}_m) & \cdots & h_{n_h}(\boldsymbol{X}_m) \end{pmatrix} = \begin{pmatrix} \boldsymbol{h}(\boldsymbol{X}_1) \\ \cdots \\ \boldsymbol{h}(\boldsymbol{X}_m) \end{pmatrix}$$

   where $\boldsymbol{h}_i(\boldsymbol{X}_j) = g(W^{(i)\top}\boldsymbol{X}_j + b_i)$ and $W^{(i)}$ is the ith column of W
4. Estimate the output layer weight $\boldsymbol{\beta}$ by following equation

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^\dagger \boldsymbol{Y} = \left(\boldsymbol{H}^\top \boldsymbol{H} + \frac{\boldsymbol{I}}{C}\right)^{-1} \boldsymbol{H}^\top \boldsymbol{Y} \tag{7}$$

   where $\boldsymbol{Y}$ is the target vector $(\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_m)^\top$.

From Eq.(7), $\boldsymbol{\beta}$ could also be regraded as the solution of the following least square problem :

$$(P) : minimize : \frac{1}{2}tr\left(\boldsymbol{\beta}^{\top}\boldsymbol{\beta}\right) + \frac{1}{2}Ctr\Big((\boldsymbol{H\beta} - \boldsymbol{Y})^{\top}(\boldsymbol{H\beta} - \boldsymbol{Y})\Big)$$

where $tr(\boldsymbol{A})$ is the trace of a matrix $\boldsymbol{A}$.

3. Proposed Algorithms

3.1. $\boldsymbol{ELM^{AUC}}$: an efficient AUC optimization algorithm for binary classification

Before deploying ELM, we need to transfer the raw input to the hidden layer feature of the ELM algorithm. Consequently, $\boldsymbol{h}(\boldsymbol{x}_i)$ is denoted as the hidden layer feature of $\boldsymbol{x}_i$. The empirical risk for ELM denoted by $\mathcal{R}_{AUCELM}$ could then be obtained from the following equation:

$$\mathcal{R}_{AUCELM} = \frac{1}{n_+n_-}\sum_{i\in N_+}\sum_{j\in N_-}(1 - \boldsymbol{\beta}^{\top}(\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_j)))^2$$

Following the structural risk minimization principle, our goal is to solve the following problem

$$\begin{aligned}
&\boldsymbol{P_{AUCELM}} : \\
&\hat{\boldsymbol{\beta}}^*_{AUC} &&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\left(C\mathcal{R}_{AUCELM} + \frac{1}{2}\|\boldsymbol{\beta}\|_2^2\right)\\
& &&= \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\left(n_+n_-\mathcal{R}_{AUCELM} + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2\right)
\end{aligned}$$

where $\lambda = \dfrac{1}{C}$. The corresponding scoring function for the optimal solution would be :

$$s(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^{\top}\hat{\boldsymbol{\beta}}^*_{AUC}$$

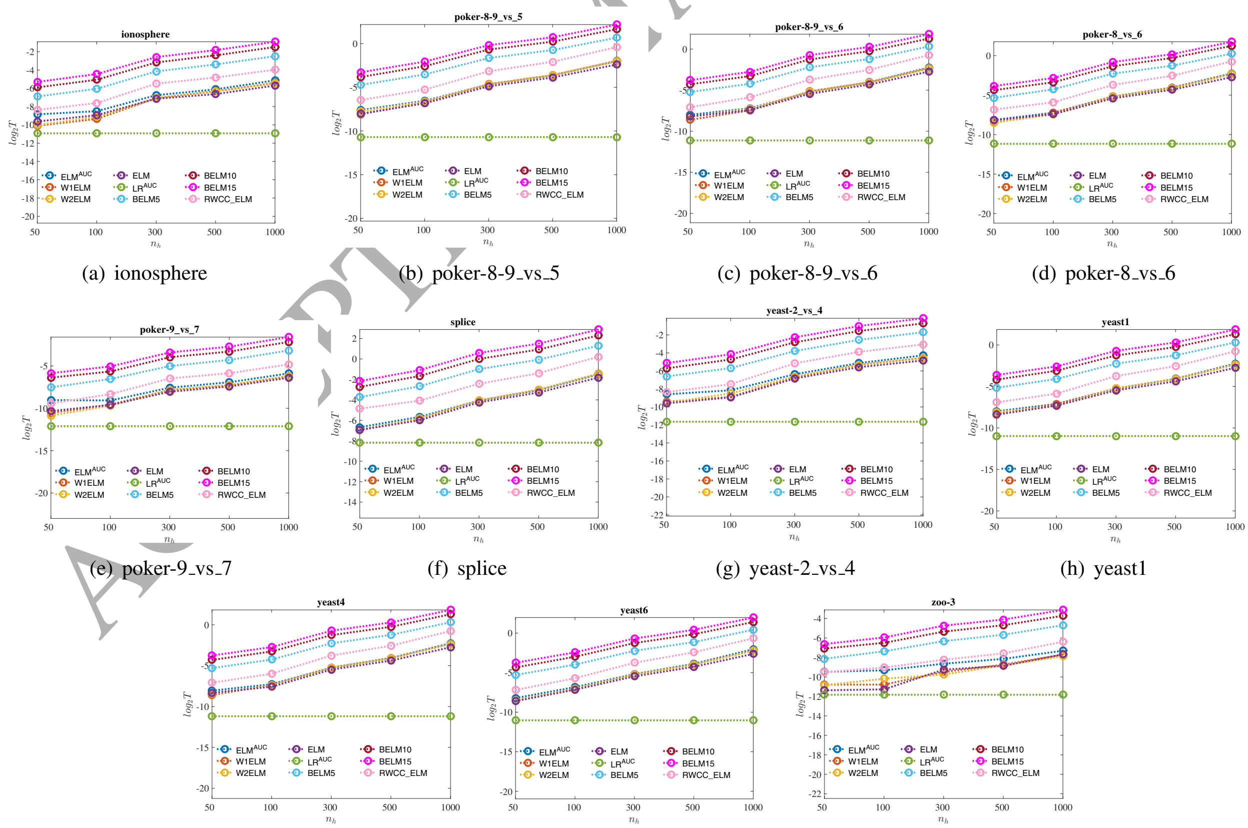
Similar as previous studies [7, 71], such a problem could be solved by a graph-based approach. Considering a weighted graph $\mathcal{G} =< \mathcal{V}, \mathcal{E}, \boldsymbol{W} >$, where $\mathcal{V}$ is the vertex set with the elements of which being all the training instances; $\mathcal{E}$ is edge set. There is an edge between two nodes v_i and v_j if and only if $y_i \neq y_j$. Denoted by $\boldsymbol{W}$, the affinity matrix recording these weights is defined as

$$W_{ij} = \begin{cases} 1, & y_i \neq y_j \\ 0, & otherwise \end{cases} \tag{8}$$

Then, with simple mathematics [70], $\boldsymbol{P_{AUCELM}}$ could be rearranged to a weighted least square problem with weighting matrix being $\boldsymbol{L}$.

$$\boldsymbol{L} = D - \boldsymbol{W} \tag{9}$$

where $D = diag(W\mathbf{1}_{m\times 1})$. In graph theory, L is often referred to as the Laplacian matrix [72] of $\mathcal{G}$. Consequently, a naive closed-form solution could be obtained by the virtue of the sophisticated weighted least square algorithm. However, following such a straight-forward method, we have to calculate and store the Laplacian matrix $\boldsymbol{L}$ with complexity $O(m^2)$ and $O(m^2n_h)$ respectively, which makes it not scalable with respect to the sample size m. For AUC optimization, since $\boldsymbol{L}$ contains nothing more than the label information, it is completely not necessary to store and compute $\boldsymbol{L}$ explicitly. Pahikkala *etal.* Ref. [7] proposed an efficient algorithm to decompose Laplacian matrix based



(a) ionosphere (b) poker-8-9_vs_5 (c) poker-8-9_vs_6 (d) poker-8_vs_6

(e) poker-9_vs_7 (f) splice (g) yeast-2_vs_4 (h) yeast1

(i) yeast4 (j) yeast6 (k) zoo-3

Figure 1: Average Running time for binary classification algorithms

on training labels, provided that the training set is sorted according to the labels.

Now, based on the previous works and $\boldsymbol{P_{AUCELM}}$, we are now ready to propose a novel algorithm entitled ELM^{AUC} as shown in Algorithm 1, with the following revision with respect to Ref. [7].

- Some simplifications are done in ELM^{AUC} to avoid redundant computations.
- We also consider the case when $m < n_h$.

Algorithm 1: ELM^{AUC}

Input: $\boldsymbol{X}$, $\boldsymbol{Y}$, C, n_h, N, n_-, n_+
Output: $\hat{\boldsymbol{\beta}}^*_{AUC}$,$W$,$b$

for i=1:m **do**
    $\tilde{\boldsymbol{y}}_i := I\,[\boldsymbol{Y}_i = 1]$;
end
randomly generate input layer weight matrix $W \in \mathbb{R}^{d\times n_h}$ and $b \in \mathbb{R}^{n_h}$;
calculate hidden layer output as follows: $\boldsymbol{H} = g(W^\top \boldsymbol{X} + b)$;
$\boldsymbol{D} := diag\Big(n_-\tilde{\boldsymbol{y}} + n_+(\mathbf{1}_{m\times 1} - \tilde{\boldsymbol{y}})\Big)$;
$C := \dfrac{C}{n_+ n_-}$;
$\boldsymbol{h}^+ := \boldsymbol{H}^\top \tilde{\boldsymbol{y}}$;
$\boldsymbol{h}^- := \boldsymbol{H}^\top(\mathbf{1}_{m\times 1} - \tilde{\boldsymbol{y}})$;
if $n_h > m$ **then**
    $\tilde{\boldsymbol{H}} := \boldsymbol{D}\boldsymbol{H} - \tilde{\boldsymbol{y}}\boldsymbol{h}^{-\top} - (\mathbf{1}_{m\times 1} - \tilde{\boldsymbol{y}})\boldsymbol{h}^{+\top}$;
    $\hat{\boldsymbol{\beta}}^*_{AUC} := \boldsymbol{H}^\top\Big(\tilde{\boldsymbol{H}}\boldsymbol{H}^\top + \dfrac{\boldsymbol{I}}{C}\Big)^{-1}\Big(n_-\tilde{\boldsymbol{y}} - n_+(\mathbf{1}_{m\times 1} - \tilde{\boldsymbol{y}})\Big)$;
else
    $\boldsymbol{hh} := \boldsymbol{h}^-\boldsymbol{h}^{+T}$;
    $\hat{\boldsymbol{\beta}}^*_{AUC} = \Big(\dfrac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{D}\boldsymbol{H} - \boldsymbol{hh} - \boldsymbol{hh}^\top\Big)^{-1}\Big(n_-\boldsymbol{h}^+ - n_+\boldsymbol{h}^-\Big)$;
end

Theorem 1. *$\hat{\boldsymbol{\beta}}^*_{AUC}$ in Algorithm 1 is a global optimal solution of $\boldsymbol{P_{AUCELM}}$*

Proof. According to the least square AUC optimization [7], $\boldsymbol{P_{AUCELM}}$ could be rearranged to the following problem.

$$\hat{\boldsymbol{\beta}}^*_{AUC} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\left(\frac{1}{2}(\boldsymbol{H\beta} - \tilde{\boldsymbol{y}})^\top \boldsymbol{L}(\boldsymbol{H\beta} - \tilde{\boldsymbol{y}}) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2\right) \tag{10}$$

where $\boldsymbol{L}$, as mentioned above, is the Laplacian matrix for $\boldsymbol{W}$, and $\lambda = \frac{n_+n_-}{C}$. Then it is easy to obtain a naive solution as follows:

- if $m > n_h$, then

$$\hat{\boldsymbol{\beta}}^*_{AUC} = \Big(\boldsymbol{H}^\top\boldsymbol{L}\boldsymbol{H} + \lambda I\Big)^{-1}\boldsymbol{H}^\top\boldsymbol{L}\tilde{\boldsymbol{y}}$$

- otherwise

$$\hat{\boldsymbol{\beta}}^*_{AUC} = \boldsymbol{H}^\top\Big(\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^\top + \lambda I\Big)^{-1}\boldsymbol{L}\tilde{\boldsymbol{y}}$$

Now all we need to do is to simplify $\boldsymbol{W}$ and thus $\boldsymbol{L}$.
Considering Eq.(8), it is easy to see that

$$W_{ij} = I\left[\tilde{y}_i \neq \tilde{y}_j\right] = \tilde{y}_i(1 - \tilde{y}_j) + \tilde{y}_j(1 - \tilde{y}_i)$$

According to the equation above, it is easy to observe that $\boldsymbol{W}$ is in nature the sum of two outer products, namely:

$$\boldsymbol{W} = \tilde{\mathbf{y}}(\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})^\top + (\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})\,\tilde{\mathbf{y}}^\top$$

According to Eq.(9), $\boldsymbol{L}$ could then be expressed as follows.

$$\begin{aligned}\boldsymbol{L} \quad &= diag\left(\tilde{\mathbf{y}}(\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})^\top\mathbf{1}_{m\times 1} + (\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})\tilde{\mathbf{y}}^\top\mathbf{1}_{m\times 1}\right)\\ &\qquad - \tilde{\mathbf{y}}(\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})^\top - (\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})\tilde{\mathbf{y}}^\top\end{aligned}$$

Note that $(\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})^\top\mathbf{1}_{m\times 1}$=$n_-$ and $\tilde{\mathbf{y}}^\top\mathbf{1}_{m\times 1}$=$n_+$, it is easy to rearrange D in Eq.(9) as follows:

$$\mathbf{D} = diag(n_-\tilde{\mathbf{y}} + n_+(\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}}))$$

Finally, it is easy to obtain:

$$L = \mathbf{D} - \tilde{\mathbf{y}}(\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})^\top - (\mathbf{1}_{m\times 1} - \tilde{\mathbf{y}})\tilde{\mathbf{y}}^\top \tag{11}$$

Following Eq.(11), it is easy to further simplify $\boldsymbol{H}^\top\boldsymbol{L}\boldsymbol{H}$, $\boldsymbol{H}^\top\boldsymbol{L}\tilde{\boldsymbol{y}}$, $\boldsymbol{L}\boldsymbol{H}\boldsymbol{H}^\top$, and $\boldsymbol{L}\tilde{\boldsymbol{y}}$. Finally Algorithm 1 could be obtained by substituting the simplified terms into the naive solution. Since all of these simplifications are straightforward, they're all omitted.

According to Gershgorin circle theorem [73], $\boldsymbol{L}$ must be positive semi-definite. As a result, the global optimality of $\hat{\boldsymbol{\beta}}^*_{AUC}$ obviously holds.

□

In Algorithm 1, the computational complexity of the matrix inversion is either $O(m^3)$ or $O\left(n_h^3\right)$. The matrix multiplications $\boldsymbol{H}^\top\boldsymbol{D}\boldsymbol{H}$ and $\tilde{\boldsymbol{H}}\boldsymbol{H}^\top$ could be performed within $O\left(mn_h^2\right)$ time and $O\left(m^2n_h\right)$ time respectively, while all the other operations have lower computational complexity. As a result, the overall computational complexity of ELM^{AUC} is $O\left(\min\{m^3{+}n_hm^2, {n_h}^3{+}m{n_h}^2\}\right)$, which is exactly the same as that of the original ELM algorithm. For space complexity, the only extra space used in Algorithm 1 is $O(m)$ which is much smaller than the space needed for storing H and the space for storing the matrix inversion. Accordingly, it is concluded that the space complexity of ELM^{AUC} also remains comparative to the original one.

3.2. A unified objective function for multi-class AUC optimization

In advance of discussing multi-class AUC optimization, it is essential to clarify the related notations. Similar as binary cases, for multi-class classifications, the instance space is denoted as $\mathcal{X}$, and the label set now turns out to be $\mathcal{Y} = \{1, 2, \cdots n_c\}$, where nc is the number of all possible classes. Following the traditional One vs. All scheme, it is easy to decompose the original multi-class classification task into nc binary classification tasks. For the ith subtask, the label set is defined as $\mathcal{Y}_i = \{-1, 1\}$. If the underlying instance belongs to the ith class then we have $\boldsymbol{Y}_i = 1$ otherwise this instance will be labeled by -1. $s_i(\cdot)$, the scoring function for the ith subtask, is defined as a mapping $\mathcal{X} \to \mathbb{R}$, which is in proportion to $\mathbb{P}(y = i|x)$.

With the notations above, it is easy to obtain a naive multi-class AUC extension named as AUC_{macro} in this paper:

$$AUC_{macro} = \frac{1}{nc}\sum_{i=1}^{nc} AUC_i$$

where AUC_i refers to the AUC measure for the ith subtask according to the One vs. All scheme i.e. $\mathbb{P}(s_i(x_1) > s_i(x_2)|y_1 = i, y_2 \neq i)$. As discussed in introduction section, an alternative generalization of binary AUC would be AUC_M:

$$AUC_M = \frac{1}{nc(nc-1)} \sum_{i \neq j} AUC(i|j); \tag{12}$$

According to Ref. [10], $AUC(i|j)$ is the possibility that an instance sampled from the ith class has a higher score of $s_i(\cdot)$ than an instance sampled from the jth class. Obviously, following the One vs. All scheme, $AUC(i|j)$ could be represented by

$$AUC(i|j) = \mathbb{P}(s_i(x_1) > s_i(x_2)|y_1 = i, y_2 = j)$$

Note that it is not necessary to have $AUC(i|j) = AUC(j|i)$. $AUC(i|j)$ is a partial metric for the ith binary classification subtask, $AUC(j|i)$ is that of the jth subtask.

Given a dataset $\mathcal{S} = (x_i, y_i)_{i=1}^m \subset (\mathcal{X} \times \mathcal{Y})^m$, both AUC_{macro} and AUC_M could be estimated by replacing the corresponding possibilities with their frequencies on $\mathcal{S}$. As a result the unbiased estimation for AUC_M could be expressed as the following equation.

$$\widehat{AUC}_M = \sum_{i=1}^{n_c} \sum_{j \neq i} \frac{1}{n_i n_j} \sum_{n \in \mathcal{N}_i} \sum_{k \in \mathcal{N}_j} \left(I\left[s_i(\mathbf{x}_n) > s_i(\mathbf{x}_k)\right] + \frac{1}{2} I\left[s_i(\mathbf{x}_n) = s_i(\mathbf{x}_k)\right] \right)$$

where $\mathcal{N}_i = \{l : \{\mathbf{x}_l, y_l\} \in \mathcal{S} \wedge y_l = i, l = 1, 2, \cdots, m\}$. By turning the ranking reward into the ranking loss, we could then obtain the empirical risk function $\mathcal{R}_{AUC_M}$ based on AUC_M:

$$\mathcal{R}_{AUC_M} = \sum_{i=1}^{n_c} \sum_{j \neq i} \frac{1}{n_i n_j} \sum_{n \in \mathcal{N}_i} \sum_{k \in \mathcal{N}_j} \left(I\left[s_i(\mathbf{x}_n) < s_i(\mathbf{x}_k)\right] + \frac{1}{2} I\left[s_i(\mathbf{x}_n) = s_i(\mathbf{x}_k)\right] \right) \tag{13}$$

Likewise, the empirical risk function for AUC_{macro} is the same as $\mathcal{R}_{AUC_M}$ except that n_j should be replaced with $n_{\setminus i}$, the total amount of instances that doesn't belongs to the ith class. We denote this risk as $\mathcal{R}_{AUC_{macro}}$.

The following theorem decomposes AUC_{macro} into a weighted average of $AUC(i|j)$s. Further, it could, to some degree, reveal the disadvantages of the naive extension from a micro perspective based on pairwise AUC.

Theorem 2. *The following two facts hold:*

- $$\begin{aligned} AUC_{macro} &= \frac{1}{nc} \sum_{i=1}^{n_c} \sum_{i \neq j} AUC(i|j)\mathbb{P}(y = j|y \neq i) \\ &= \frac{1}{nc} \sum_{i=1}^{nc} \mathbb{E}_{j \neq i}(AUC(i|j)) \end{aligned}$$
- *$AUC_{macro} = AUC_M$ when the labels are subject to discrete uniform distribution i. e.* $\forall i, \mathbb{P}(y = i) = \frac{1}{nc}$

Proof.

$$\begin{aligned} AUC_i &= \mathbb{P}(s_i(x_1) > s_i(x_2)|y_1 = i, y_2 \neq i) \\ &= \frac{\mathbb{P}(s_i(x_1) > s_i(x_2), y_1 = i, y_2 \neq i)}{\mathbb{P}(y_1 = i)\mathbb{P}(y2 \neq i)} \\ &= \frac{\sum_{j \neq i} \mathbb{P}(s_i(x_1) > s_i(x_2), y_1 = i, y_2 = j)}{\mathbb{P}(y_1 = i)\mathbb{P}(y2 \neq i)} \\ &= \frac{\sum_{j \neq i} AUC(i|j)\mathbb{P}(y_2 = j)}{\mathbb{P}(y_2 \neq i)} \\ &= \sum_{j \neq i} AUC(i|j)\mathbb{P}(y_2 = j|y_2 \neq i) \end{aligned}$$

Now we could rearrange AUC_{macro} with the aforementioned equation :

$$\begin{aligned} AUC_{macro} &= \frac{1}{nc} \sum_i AUC_i \\ &= \frac{1}{nc} \sum_{i=1}^{n_c} \sum_{i \neq j} AUC(i|j)\mathbb{P}(y_2 = j|y \neq i) \\ &= \frac{1}{nc} \sum_{i=1}^{nc} \mathbb{E}_{j \neq i}(AUC(i|j)) \end{aligned}$$

Note that the probability $\mathbb{P}(y_2 = i), i = 1, 2, \cdots n_c$ is replaced with a more general form $\mathbb{P}(y = i), i = 1, 2, \cdots n_c$, due to the i.i.d. sampling property. If the labels are subjected to discrete uniform distribution, by replacing both $\mathbb{P}(y = i)$ and $\mathbb{P}(y = j)$ with $\frac{1}{nc}$, we have:

$$\begin{aligned} AUC_{macro} &= \frac{1}{nc} \sum_{i=1}^{n_c} \sum_{i \neq j} \frac{AUC(i|j)\frac{1}{nc}}{1 - \frac{1}{nc}} \\ &= \frac{1}{nc(nc-1)} \sum_{i=1}^{n_c} \sum_{i \neq j} AUC(i|j) \\ &= AUC_M \end{aligned}$$

□

From Theorem 2, we could reach that AUC_{macro} weighs different $AUC(i|j)$ based on $\mathbb{P}(y = j|y \neq i)$, the label distribution conditioned on $y \neq i$. Moreover, for imbalanced datasets , the label distribution is dominated by the major classes. Hence the $AUC(i|j)$ for minor classes may prone to be ignored if we adopt the AUC_{macro} as the objective. On the contrary, AUC_M better reflects the overall performance based on pair-wise AUC, as an identical weight is adopted for all the $AUC(i|j)$s.

Meanwhile, it is easy to observe that both AUC_{macro} and AUC_M are essentially weighted averages of $AUC(i|j)$s. Inspired by this conclusion, we then adopt the weighted sum of the risk form of all $AUC(i|j)$s as the unified form of objective functions for multi-class AUC optimization.

Following the construction of binary AUC optimization framework, the ranking loss function in Eq.(13) is replaced with a general surrogate loss $l(\cdot)$. $\frac{1}{n_i n_j}$ in Eq.(13) is replaced with more general weights w_{ij}. In consequence, a general surrogate empirical risk entitled R_M could be expressed as follows.

$$\mathcal{R}_M = \sum_{i=1}^{n_c} \sum_{j \neq i} w_{ij} \sum_{n \in \mathcal{N}_i} \sum_{k \in \mathcal{N}_j} l\left(s_i(\mathbf{x_n}|\theta) - s_i(\mathbf{x_k}|\theta)\right) \tag{14}$$

Following the structural risk minimization, the unified objective function could be expressed as $\mathcal{L}_M$:

$$\mathcal{L}_M = \mathcal{R}_M + \frac{\lambda}{2}\Omega(s_1, \cdots, s_{nc}) \tag{15}$$

where $\Omega(s_1, \cdots, s_{nc})$ is a predefined regularizer to punish high model complexity. According to Theorem 2, if we set $w_{ij} = \frac{1}{n_i n_j}$ then the underlying objective function could be employed to optimize AUC_M. On the other hand, if we set $w_{ij} = \frac{1}{n_i n_{\backslash i}}$, then the underlying objective function could be employed to optimize AUC_{macro}.
Furthermore, if we set $w_{12} = \frac{1}{n_+ n_-}$, and let all other w_{ij}s be zero, then Eq.(15) degenerates to a binary AUC optimization objective. As a result, the proposed objective function could explain the optimization for AUC_M, AUC_{macro}, as well as binary AUC optimization approaches.

3.3. Extreme learning machines for multi-class AUC optimization

According to Eq.(15), there are four building-blocks necessary for a multi-class AUC optimization algorithm : the weights w_{ij}, the scoring function for all subtasks $s_i(\cdot)$, the surrogate loss for ranking loss $l(\cdot)$, and the regularizer $\Omega(\cdot)$. In order to employ ELM to solve multi-class AUC optimization, these four building blocks are set as follows.

- w_{ij} : As discussed in the previous subsection, we choose $w_{ij} = \frac{1}{n_i n_j}$ for optimizing AUC_M and the corresponding algorithm is denoted by ELM_M^{AUC}, while $w_{ij} = \frac{1}{n_i n_{\backslash i}}$ is selected for AUC_{macro} and the corresponding algorithm is named as ELM_{macro}^{AUC}
- $s_i(\cdot)$: Since ELM is adopted, the scoring function thus becomes $s_i(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})^\top \boldsymbol{\beta}^{(i)}$, which is identical with the original ELM algorithm.
- $l(\cdot)$: In the light of the fact that ELM adopts least square method to solve its output layer parameters, we select $l(t) = (1-t)^2$ as the surrogate loss.
- $\Omega(\cdot)$: This paper adopts the standard ridge regression method, the regularizer is defined as $\Omega(\cdot) = tr\left(\boldsymbol{\beta}^\top \boldsymbol{\beta}\right) = \sum_{i=1}^{n_c} \|\boldsymbol{\beta}^{(i)}\|_2^2$

According to the aforementioned discuss, the objective function for optimizing AUC_M, denoted by $\mathcal{L}_{M_1}$, could be expressed as follows:

$$\mathcal{L}_{M_1} = \sum_{i=1}^{n_c} \sum_{j \neq i} \frac{1}{n_i n_{\backslash i}} \sum_{n \in N_i} \sum_{k \in N_j} \left(1 - \boldsymbol{\beta}^{(i)\top} (\mathbf{h}(\mathbf{x}_n) - \mathbf{h}(\mathbf{x}_k))\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_c} \|\boldsymbol{\beta}^{(i)}\|_2^2$$

where $\lambda = \frac{1}{C}$.

Correspondingly, a novel algorithm called ELM_{macro}^{AUC} is proposed to solve the problem: $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \mathcal{L}_{M_1}$. This problem could be easily decomposed into n_c binary AUC optimization problems which could be solved by Algorithm 1. Bearing this in mind, we could summarize the main process of ELM_{macro}^{AUC} as Algorithm 2.

Similar as AUC_{macro}, for AUC_M, the objective function could be expressed as follows:

$$\mathcal{L}_{M_2} = \sum_{i=1}^{n_c} \sum_{j \neq i} \frac{1}{n_i n_j} \sum_{n \in N_i} \sum_{k \in N_j} \left(1 - \boldsymbol{\beta}^{(i)\top} (\mathbf{h}(\mathbf{x}_n) - \mathbf{h}(\mathbf{x}_k))\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_c} \|\boldsymbol{\beta}^{(i)}\|_2^2$$

Algorithm 2: ELM_{Macro}^{AUC}

Input: $\boldsymbol{X}, \boldsymbol{y} = \{y_i\}_{i=1}^m, y_i \in \{1, 2, \cdots, n_c\}, C, n_h$
Output: $\hat{\boldsymbol{\beta}}_{AUC}^*$,$W$,$b$

for $k = 1{:}nc$ **do**

$$\mathbf{Y}^{(i)} = \begin{pmatrix} I[\mathbf{y}_1 = k] \\ \vdots \\ I[\mathbf{y}_m = k] \end{pmatrix};$$

end

randomly generate input layer weight matrix $W \in \mathbb{R}^{d \times n_h}$ and $b \in \mathbb{R}^{n_h}$;

calculate hidden layer output as follows: $\boldsymbol{H} = g(\boldsymbol{X}W + b)$;

for $k = 1{:}nc$ **do**

$C' := \frac{C}{n_i n_{\backslash i}}$;

calculate $\hat{\boldsymbol{\beta}}_{AUC}^{(k)*}$ based on $\boldsymbol{H}$ and $\boldsymbol{Y}^{(i)}$ with line 8-16 of Algorithm 1 with the $\tilde{\mathbf{y}}$ replaced with $\boldsymbol{Y}^{(i)}$ and C replaced with C' ;

end

$\hat{\boldsymbol{\beta}}_{AUC}^* = \left[\hat{\boldsymbol{\beta}}_{AUC}^{(1)*}, \cdots, \hat{\boldsymbol{\beta}}_{AUC}^{(n_c)*}\right]$;

where $\lambda = \frac{1}{C}$.
We propose a novel multi-class classification algorithm named ELM_M^{AUC} to solve the corresponding optimization problem called P_{ELMAUC_M}

$$\boldsymbol{P_{ELMAUC_M}} : \boldsymbol{\beta}_{AUC}^* = \operatorname*{argmin}_{\boldsymbol{\beta}} \mathcal{L}_{M_2}$$

The whole procedure of ELM_M^{AUC} could be summarized as Algorithm 3. Furthermore, Theorem 3 is proposed to justify this algorithm.

Theorem 3. *$\hat{\boldsymbol{\beta}}_{AUC}^*$ in Algorithm 3 is a global optimal solution of $\boldsymbol{P_{ELMAUC_M}}$*

Proof. Similar as Ref. [18], we could solve P_{ELMAUC_M} by decomposing the original optimization problem into independent subproblems.

Let $\mathcal{L}_i = \sum_{j \neq i} \frac{1}{n_i n_j} \sum_{n \in N_i} \sum_{k \in N_j} \left(1 - \boldsymbol{\beta}^{(i)\top} (\mathbf{h}(\mathbf{x}_n) - \mathbf{h}(\mathbf{x}_k))\right)^2 + \frac{\lambda}{2} tr\left(\boldsymbol{\beta}^{(i)\top} \boldsymbol{\beta}^{(i)}\right)$. It is easy to show that $\mathcal{L}_{M_2} = \sum_{i=1}^{n_c} \mathcal{L}_i$. Furthermore, since $\boldsymbol{\beta}^{(i)}$ is only relevant with $\mathcal{L}_i$. We could find out a solution of $\boldsymbol{P_{ELMAUC_M}}$ by combining the results of $\boldsymbol{P}_i : \hat{\boldsymbol{\beta}}^{(i)*} = \operatorname*{argmin}_{\boldsymbol{\beta}^{(i)}} \mathcal{L}_i$ (see Line 24 of Algorithm 3).

Given an affinity matrix $\boldsymbol{W}_i$ defined as :

$$\boldsymbol{W}_i = D(\mathbf{1}_{m \times 1} - \mathbf{Y}^{(i)})\mathbf{Y}^{(i)\top} + \mathbf{Y}^{(i)}(\mathbf{1}_{m \times 1} - \mathbf{Y}^{(i)})^\top D$$

where D is defined according to Line 4-6 in Algorithm 3. Then it is easy to show that P_i is equivalent with the following problem.

$$\min_{\boldsymbol{\beta}^{(i)}} \frac{1}{2} (\boldsymbol{H}\boldsymbol{\beta}^{(i)} - \boldsymbol{Y}^{(i)})^\top \boldsymbol{L}_i (\boldsymbol{H}\boldsymbol{\beta}^{(i)} - \boldsymbol{Y}^{(i)}) + \frac{\lambda}{2} (\boldsymbol{\beta}^{(i)\top} \boldsymbol{\beta}^{(i)}) \tag{16}$$

where $\lambda = \frac{n_i}{C} = \frac{1}{C'}$ and C' is defined as Line 10 in Algorithm3, $\boldsymbol{L}_i$ is the Laplacian matrix, $\boldsymbol{Y}^{(i)}$ is obtained according to Line 1-3 of Algorithm 3. Based on Eq.(16), $\hat{\boldsymbol{\beta}}_{AUC}^{(i)*}$ could be proved to be a solution of $\boldsymbol{P}_i$ following similar procedure as the proof of Theorem 1.

Algorithm 3: ELM_M^{AUC}

Input: $X, y = \{y_i\}_{i=1}^m, y_i \in \{1, 2, \cdots, n_c\}, C, n_h$

Output: $\hat{\beta}^*_{AUC}$,W,b

```
for k = 1:nc do
    Y^(i) = ( I[y_1 = k] ; ... ; I[y_m = k] );
end
for n = 1:m do
    D(n,n) = 1 / n_{y_n};
end
randomly generate input layer weight matrix W ∈ R^{d×n_h} and b ∈ R^{n_h} ;
calculate hidden layer output : H = g(XW + b) ;
for i = 1:nc do
    C' = C / n_i;
    Ybar^(i) = D(1_{m×1} − Y^(i));
    H_p := H^T Y^(i);
    H_n := H^T Ybar^(i);
    D_i := diag((nc − 1) Y^(i) + n_i Ybar^(i));
    if n_h < m then
        hh := H_p H_n^T;
        H~_d := D_i H;
        β^^(i)*_AUC := (H^T H~_d + I/C' − hh − hh^T)^{-1} ((nc − 1) H_p − n_i H_n);
    else
        H~ := H^T D_i − H_p Ybar^(i)T − H_n Y^(i)T;
        β^^(i)*_AUC := H^T (H~^T H + I/C')^{-1} ((nc − 1) Y^(i) − n_i Ybar^(i));
    end
end
β^*_AUC = [β^^(1)*_AUC, ··· , β^^(n_c)*_AUC];
```

For global optimality, according to Gershgorin circle theorem [73], all the $\boldsymbol{L}_i$s are positive semi-definite. As a result, $\forall i = 1, 2, \cdots n_c$ $\hat{\boldsymbol{\beta}}^{(i)*}_{AUC}$ is guaranteed be a global optimal solution of $\boldsymbol{P}_i$ and thus $\hat{\boldsymbol{\beta}}^*_{AUC}$ is a global optimal solution of P_{ELMAUC_M} $\square$

Since ELM^{AUC}_{macro} only solves n_c binary AUC optimization subproblems, it could thus be performed in $O\left(n_c \min\{m^3{+}n_h m^2, {n_h}^3{+}m{n_h}^2\}\right)$. Comparing with ELM^{AUC}_{macro}, the only extra computation for ELM_{AUC_M} comes from the operation $\overline{\boldsymbol{Y}}^{(i)} = D(\mathbf{1}_{m\times 1} - \boldsymbol{Y}^{(i)})$. As a result, the computational complexity for ELM_{AUC_M} is also $O\left(n_c \min\{m^3{+}n_h m^2, {n_h}^3{+}m{n_h}^2\}\right)$. For most of datasets with a small or medium size, the proposed algorithms could finish its computations within several seconds. Hence, if n_c is not too large, both ELM^{AUC}_{macro} and ELM_{AUC_M} will still be efficient. Empirical validation of this conclusion could be found in Section 5.

Similar as ELM_{AUC}, the space complexity of both ELM^{AUC}_{macro} and ELM^{AUC}_M remains the same as the original ELM algorithm.

4. Generalization analysis for ELM^{AUC}_M

seeing that ELM^{AUC}_M could optimize AUC_M, according to the discussion in section 3.2, it is thus a more reasonable algorithm than ELM^{AUC}_{macro}. As a result, this section mainly focus on the generalization analysis of ELM^{AUC}_M.

4.1. Preliminaries and notations

Define $s_i(\boldsymbol{x}) = \boldsymbol{\beta}^{(i)\top}\boldsymbol{h}(\boldsymbol{x}) = \sum_{l=1}^{n_h} \beta_l^{(i)}\phi(\boldsymbol{w}^{(l)}, \boldsymbol{x})$ as the scoring function of the ith class. $\phi(\boldsymbol{w}^{(l)}, \boldsymbol{x}) = g(\boldsymbol{w}^{(l)\top}\boldsymbol{x})$ is the hidden feature of the lth neuron for instance $\boldsymbol{x}$. Recall what discussed in section 3, our objective function is based on square surrogate loss.

$$l\Big(s_i(\mathbf{x}_1) - s_i(\mathbf{x}_2)\Big) = \Big(1 - \left(s_i(\mathbf{x}_1) - s_i(\mathbf{x}_2)\right)\Big)^2 \tag{17}$$

For the case of simplicity, we will denote $l\Big(s_i(\mathbf{x}_1) - s_i(\mathbf{x}_2)\Big)$ as $l\left(s_i, \mathbf{x}_1, \mathbf{x}_2\right)$ through out this section.

Before discussing the generalization ability of ELM^{AUC}_M, we first define the surrogate expected risk and empirical risk for $l\left(s_i, \mathbf{x}_1, \mathbf{x}_2\right)$ and the hypothesis space for ELM^{AUC}_M.

According to the relationship between probability and expectation of indicator function, $AUC(i|j)$ could be rearranged to form the following equation.

$$AUC(i|j) = \mathbb{E}_{\mathbf{x}_1,\mathbf{x}_2}\left\{I\left[s_i(\mathbf{x}_1) > s_i(\mathbf{x}_2)\right] + \frac{1}{2}I\left[s_i(\boldsymbol{x}_1) = s_i(\boldsymbol{x}_2)\right]|y_1 = i, y_2 = j\right\}$$

where $(\boldsymbol{x}_1, y_2)$, $(\boldsymbol{x}_2, y_2)$ are two randomly sampled data pairs. By turning the ranking reward to the ranking loss, we could define the expected risk of $AUC(i|j)$ as $\overline{R}_{AUC(i|j)}$ with

$$\overline{\mathcal{R}}_{AUC(i|j)} = \mathbb{E}_{\mathbf{x}_1,\mathbf{x}_2}\left\{l_{rank}\left(s_i, x_1, x_2\right)|y_1 = i, y_2 = j\right\} \tag{18}$$

where $l_{rank}\left(s_i, x_1, x_2\right) = I\left[s_i(\boldsymbol{x_1}) < s_i(x_2)\right] + \dfrac{1}{2}I\left[s_i(\boldsymbol{x_1}) = s_i(x_2)\right]$. Then the expected risk denoted by $\overline{\mathcal{R}}_{AUC_M,i}\left(s_i\right)$ for the ith problem becomes:

$$\overline{\mathcal{R}}_{AUC_M,i}\left(s_i\right) = \sum_{j\neq i}\overline{\mathcal{R}}_{AUC(i|j)}$$

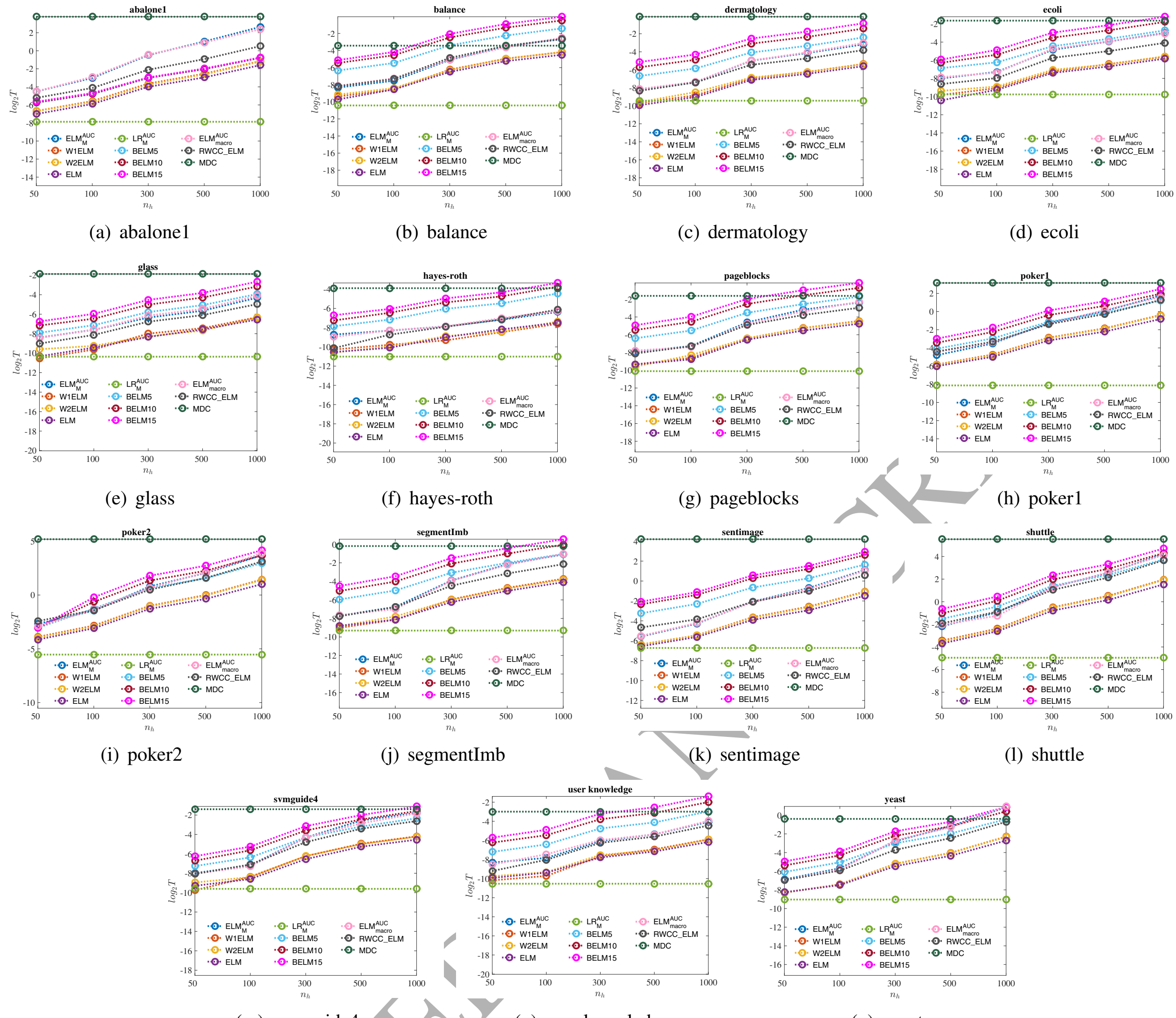
Ignoring the constant term $\dfrac{1}{n_c(n_c-1)}$ in Eq.(12), the expected risk of AUC_M could be expressed as $\overline{\mathcal{R}}_{AUC_M}\left(s_1, \cdots, s_{n_c}\right)$.

$$\overline{\mathcal{R}}_{AUC_M}\left(s_1, \cdots, s_{n_c}\right) = \sum_{i=1}^{nc}\overline{\mathcal{R}}_{AUC_M,i}(s_i)$$

$$= \sum_{i=1}^{n_c}\sum_{j\neq i}\mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{x}_2}\left\{l_{rank}(s_i, \boldsymbol{x}_1, \boldsymbol{x}_2)|y_1 = i, y_2 = j\right\}$$

As mentioned in section2 and section 3, $l_{rank}(s_i, \boldsymbol{x}_1, \boldsymbol{x}_2)$ is discrete and non-differentiable, we replace $l_{rank}(s_i, \boldsymbol{x}_1, \boldsymbol{x}_2)$ in $\overline{\mathcal{R}}_{AUC_M,i}\left(s_i\right)$ and $\overline{\mathcal{R}}_{AUC_M}\left(s_1, \cdots, s_{n_c}\right)$ with surrogate loss defined in Eq.(17) so as to form a feasible optimization problem. Correspondingly the surrogate expected risk for the ith subtask and the overall problem denoted by $\overline{\mathcal{R}}_i\left(s_i\right)$ and $\overline{\mathcal{R}}\left(s_1, \cdots, s_{n_c}\right)$ respectively could be expressed as Eq.(19) and Eq.(20).

$$\overline{\mathcal{R}}_i(s_i) = \sum_{j\neq i}\mathbb{E}_{\mathbf{x}_1,\mathbf{x}_2}\left\{l\left(s_i, x_1, x_2\right)|y_1 = i, y_2 = j\right\} \tag{19}$$



(a) abalone1 (b) balance (c) dermatology (d) ecoli

(e) glass (f) hayes-roth (g) pageblocks (h) poker1

(i) poker2 (j) segmentImb (k) sentimage (l) shuttle

(m) svmguide4 (n) user knowledge (o) yeast

Figure 2: Average Running time for multi-class classification algorithms

$$\overline{\mathcal{R}}(s_1,\cdots,s_{n_c}) = \sum_{i=1}^{nc}\overline{\mathcal{R}}_i(s_i) \tag{20}$$

$$= \sum_{i=1}^{n_c}\sum_{j\neq i}\mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{x}_2}\{l(s_i,\boldsymbol{x}_1,\boldsymbol{x}_2)|y_1=i,y_2=j\} \tag{21}$$

Given a training dataset $\mathcal{S}$, through replacing the expectations with the mean value on $\mathcal{S}$, it is easy to find that the empirical risk for the ith subproblem of ELM_M^{AUC}, denoted by $\mathcal{R}_{S,i}(s_i)$, is equivalent with the following equation:

$$\mathcal{R}_{S,i}(s_i) = \sum_{j\neq i}\sum_{n\in N_i,k\in N_j}\frac{1}{n_i n_j}l(s_i,\boldsymbol{x}_n,\boldsymbol{x}_k)$$

Correspondingly, the empirical risk for the overall problem denoted by $\mathcal{R}_S(s_1,\cdots,s_{n_c})$ is simply the sum of all $\mathcal{R}_{S,i}(s_i)$:

$$\mathcal{R}_S(s_1,\cdots,s_{n_c}) = \sum_{i=1}^{n_c}\mathcal{R}_{S,i}(s_i)$$

It is easy to show that $\mathcal{R}_S(s_1,\cdots,s_{n_c})$ and $\mathcal{R}_{S,i}(s_i)$ are the unbiased estimations of $\overline{\mathcal{R}}(s_1,\cdots,s_{n_c})$ and $\overline{\mathcal{R}}_i(s_i)$ respectively, which is equivalent with :

$$\mathbb{E}(\mathcal{R}_{S,i}(s_i)) = \overline{\mathcal{R}}_i(s_i) \tag{22}$$

and

$$\mathbb{E}(\mathcal{R}_S(s_1,\cdots,s_{n_c})) = \overline{\mathcal{R}}(s_1,\cdots,s_{n_c}) \tag{23}$$

Similar as Ref. [22, 44] , given the hidden layer parameters W and b,we will consider the generalization ability of ELM_M^{AUC} on a ELM-based hypothesis space $\mathcal{H}_{n_h}$.

$$\mathcal{H}_{n_h} = \left\{s(\cdot) = \sum_{l=1}^{n_h}\beta_l\phi(W^{(l)},\cdot):\ \boldsymbol{\beta}\in\mathbb{R}^{n_h}\right\}$$

where $\boldsymbol{\beta} = \left(\beta_1,\cdots,\beta_{n_h}\right)$. For a given training data $\mathcal{S}$, and the coefficient C, we could denote the minimizer of ELM_M^{AUC} as $s_{S,C} = (s_{1,S,C},\cdots,s_{n_c,S,C})$,where

$$s_{i,S,C} = \underset{f\in\mathcal{H}_{n_h}}{\operatorname{argmin}}\left\{\mathcal{R}_{S,i}(s_i) + \frac{1}{C}\|s_i\|_{l_2}^2\right\}$$

and

$$\|s_i\|_{l_2} = inf\left\{\|\boldsymbol{\beta}^{(i)}\|_2 : f(\cdot) = \boldsymbol{\beta}^{(i)\top}\boldsymbol{h}(W, b, \cdot)\right\}$$

with $s_i(\cdot) = \boldsymbol{\beta}^{(i)\top}\boldsymbol{h}(W, b, \cdot)$

For the ith subproblem, we define the classifier with best generalization ability as $s_i^*(\cdot)$.

$$s_i^*(\cdot) = \underset{s \in \mathcal{M}}{\operatorname{argmin}} \overline{\mathcal{R}}_i(s_i)$$

where $\mathcal{M}$ is the space for all measurable mapping : $\mathbb{R}^d \to \mathbb{R}$.

For the case of simplicity, we adopt the following assumptions.

Assumption 1. *Given multi-class classifier with the best generalization performance denoted as* $\left(s_1^*, \cdots, s_{n_c}^*\right)$, $\forall i \neq j$ $s_i^*(\cdot)$ *is not a function of* $s_j^*(\cdot)$. *Furthermore we have :*

$$\inf_{s_1 \in \mathcal{M}, \cdots, s_{n_c} \in \mathcal{M}} \overline{\mathcal{R}}(s_1, \cdots, s_{n_c}) = \sum_{i=1}^{n_c} \inf_{s_i \in \mathcal{M}} \overline{\mathcal{R}}_i(s_i)$$

where $\mathcal{M}$ *is the space for all measurable mapping :* $\mathbb{R}^d \to \mathbb{R}$.

Assumption 2. *The hidden layer feature mapping* $h(\cdot)$ *is a continuous function. Furthermore,* $\forall \boldsymbol{x} \in \mathbb{R}^d$, $\|h(\boldsymbol{x})\|_2 \leq \kappa$ *with* κ *being a positive real number.*

Assumption 3. $\forall i = 1, \cdots, n_c$, $\|s_i^*\|_\infty \leq K$ *where* $K \geq 1$.

$\forall R > 0$, define a set of functions with bounded norm as

$$\mathcal{B}_R = \left\{f \in \mathcal{H}_{n_h} : \|f\|_{l_2} \leq R\right\}$$

4.2. Generalization analysis of ELM_M^{AUC}

Without loss of Generality, we first discuss the generalization ability for the ith subproblem. The corresponding results are then generalized to reach the conclusion for the overall optimization problem for ELM_M^{AUC}.

As for the ith subproblem, following the traditional routines of generalization analysis [22, 23, 44], we first decompose the excess error$\mathbb{E}_{\mathcal{S}}\left(\overline{\mathcal{R}}_i(s_{i,\mathcal{S},C}) - \overline{\mathcal{R}}_i\left(s_i^*\right)\right)$ as the following Equation

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}}\left(\overline{\mathcal{R}}_i(s_{i,\mathcal{S},C}) - \overline{\mathcal{R}}_i(s_i^*)\right) \leq \\
&\mathbb{E}_{\mathcal{S}}\Big(\underbrace{\left(\overline{\mathcal{R}}_i(s_{i,\mathcal{S},C}) - \overline{\mathcal{R}}_i(s_i^*)\right) - \left(\mathcal{R}_{S,i}(s_{i,\mathcal{S},C}) - \mathcal{R}_i(s_i^*)\right)}_{l_1}\Big) \\
&+ \mathbb{E}_{\mathcal{S}}\Big(\underbrace{\mathcal{R}_{S,i}(s_{i,\mathcal{S},C}) - \mathcal{R}_i(s_i^*) + \frac{1}{C}\|s_{i,\mathcal{S},C}\|_{l2}^2}_{l_2}\Big) \\
&:= \mathbb{E}_{\mathcal{S}}(l_1) + \mathbb{E}_{\mathcal{S}}(l_2)
\end{aligned} \tag{24}$$

Here we adopt the same definitions as Ref. [44]. We refer to $\mathbb{E}_{\mathcal{S}}(l_1)$ as sample error and $\mathbb{E}_{\mathcal{S}}(l_2)$ approximation error. According to Eq.(20), it is easy to reach a similar decomposition for the overall optimization problem. In this subsection , we construct upper bounds for the excess error based on such decomposition.

Lemma 2. *For* $i = 1, 2, \cdots, nc$,*and if* $\forall x$ $|f(x)| \leq B$, *we have for any* $\boldsymbol{x}_1, \boldsymbol{x}_1', \boldsymbol{x}_2, \boldsymbol{x}_2'$, *the following two inequalities hold*

$$|l(f, \boldsymbol{x}_1, \boldsymbol{x}_2) - l(f, \boldsymbol{x}_1, \boldsymbol{x}_2')| \leq 2(2 + 2B)B \tag{25}$$

$$|l(f, \boldsymbol{x}_1, \boldsymbol{x}_2) - l(f, \boldsymbol{x}_1', \boldsymbol{x}_2)| \leq 2(2 + 2B)B \tag{26}$$

Proof.

$$\begin{aligned}
&|l(f, \boldsymbol{x}_1, \boldsymbol{x}_2) - l(f, \boldsymbol{x}_1, \boldsymbol{x}_2')| \leq \\
&|(1 - (f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)))^2 - (1 - (f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2')))^2| \leq \\
&|2 - (f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2) + f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2'))||f(\boldsymbol{x}_2') - f(\boldsymbol{x}_2)| \leq \\
&(2 + 2B)2B = 2(2 + 2B)B
\end{aligned}$$

Eq.(26) could be proofed directly following the same routine. □

Define $\psi_i(\mathcal{S}, f)$ as

$$\psi_i(\mathcal{S}, f) = \mathcal{R}_{S,i}(f) - \mathcal{R}_{S,i}(s_i^*)$$

Following the mathematical technologies employed in Ref. [68] we have the following Lemma.

Lemma 3. $\forall f \in \mathcal{B}_R$, *and* $i = 1, \cdots, n_c$ *we have :*

$$\mathbb{P}_{X|Y}\left\{\mathbb{E}(\psi_i(\mathcal{S}, f)) - \psi_i(\mathcal{S}, f) \geq \epsilon\right\} \leq exp\left\{\frac{-\epsilon^2\rho_i m}{32(1 + \kappa\overline{R})^4}\right\}$$

where $\overline{R} = max\left\{R, \dfrac{K}{\kappa}\right\}$. *Define* $r_k = \dfrac{n_k}{m}$, $k = 1, 2, \cdots, nc$, *then*

$$\rho_i = \frac{1}{\sum_{j\neq i}\frac{1}{r_j} + \frac{(nc-1)^2}{r_i}}$$

Proof. Suppose $\mathcal{S}_{1k}$ is a training dataset with a fixed label set $\boldsymbol{Y}$ and an instance set $\mathcal{X}_{2k} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k, \cdots, \boldsymbol{x}_m\}$, while $\mathcal{S}_{2k}$ is training dataset with a fixed label set $\boldsymbol{Y}$ and instance set $\mathcal{X}_{2k} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k', \cdots, \boldsymbol{x}_m\}$. Then $\mathcal{S}_{1k}$ is different from $\mathcal{S}_{2k}$ only by its kth instance.Now we bound the difference between $\psi(\mathcal{S}_{1k})$ and $\psi(\mathcal{S}_{2k})$.

Given that $f \in \mathcal{B}_R$, according to the Cauchy-Schwarz inequality and Assumption 2, we have :

$$|f(\boldsymbol{x})| \leq \|f\|_{l2}\|h(\boldsymbol{x})\|_2 \leq \kappa R.$$

As a result, if the label of $\boldsymbol{x}_k$ and $\boldsymbol{x}_k'$ is $j \neq i$, following assumption 3 and Lemma 2, we have.

$$\begin{aligned}
&|\psi_i(\mathcal{S}_{1k}, f) - \psi_i(\mathcal{S}_{2k}, f)| \\
&\leq \frac{1}{n_i n_j}\sum_{n\in N_i}|l(f, \boldsymbol{x}_n, \boldsymbol{x}_k) - l(f, \boldsymbol{x}_n, \boldsymbol{x}_k')| + |l(s_i^*, \boldsymbol{x}_n, \boldsymbol{x}_k) - l(s_i^*, \boldsymbol{x}_n, \boldsymbol{x}_k')| \\
&\leq \frac{8}{n_j}(1 + \kappa\overline{R})^2
\end{aligned}$$

Similarly, if the label of $\boldsymbol{x}_k$ and $\boldsymbol{x}_k'$ is exactly i,we have

$$\begin{aligned}
&|\psi_i(\mathcal{S}_{1k}, f) - \psi_i(\mathcal{S}_{2k}, f)| \\
&\leq \frac{1}{n_i}\sum_{j\neq i}\frac{1}{n_j}\sum_{n\in N_j}|l(f, \boldsymbol{x}_k, \boldsymbol{x}_n) - l(f, \boldsymbol{x}_k', \boldsymbol{x}_n)| + |l(s_i^*, \boldsymbol{x}_k, \boldsymbol{x}_n) - l(s_i^*, \boldsymbol{x}_k', \boldsymbol{x}_n)| \\
&\leq \frac{8(nc-1)}{n_i}(1 + \kappa\overline{R})^2
\end{aligned}$$

According to the famous McDiarmid inequality [22, 68] We have

$$\begin{aligned}
&\mathbb{P}_{X|Y}\left\{\mathbb{E}(\psi_i(\mathcal{S}, f), i) - \psi_i(\mathcal{S}, f) \geq \epsilon\right\} \\
&\leq exp\left\{\frac{-2\epsilon^2}{64(1 + \kappa\overline{R})^4\left((\sum_{j\neq i}\frac{n_j}{n_j^2}) + \frac{(nc-1)^2 n_i}{n_i^2}\right)}\right\} \\
&\leq exp\left\{\frac{-\epsilon^2\rho m}{32(1 + \kappa\overline{R})^4}\right\}
\end{aligned}$$

□

The following Lemma directly follows Lemma 3 of this paper and the proof of Lemma 4 in Ref. [44]

Lemma 4. $\forall \epsilon > 0$*, given the hidden parameters W and b, we have*

$$\mathbb{P}_{X|Y}\left\{ \sup_{f\in(\mathcal{B}_R)} \left(\left(\overline{\mathcal{R}}_i(f) - \overline{\mathcal{R}}_i(s_i^*)\right) - \left(\mathcal{R}_{S,i}(f) - \mathcal{R}_{S,i}(s_i^*)\right) \right) \geq \epsilon \right\} \leq$$

$$N\Big(\mathcal{B}_R, \frac{\epsilon}{16\,(n_c-1)\,(1+2\kappa R)}\Big) exp\left\{ -\frac{\epsilon^2 \rho_i m}{128(1+\kappa \overline{R})^4} \right\}$$

where $N(\mathcal{B},\eta)$ is covering number of the set $\mathcal{B}$ with radius η [22, 44],

Note that since the expectation with respect to the training data $\mathcal{S}$ is equivalent with the expectation under the joint distribution of the inputs $\boldsymbol{X}$ and outputs $\boldsymbol{Y}$, hence we have:

$$\mathbb{E}_{\mathcal{S}}(\cdot) = \mathbb{E}_{(X,Y)}(\cdot) = \mathbb{E}_{Y}\Big(\mathbb{E}_{X|Y}(\cdot)\Big) \tag{27}$$

Bearing this in mind, we first propose the following theorem conditioned on $\boldsymbol{Y}$.

Theorem 4. *The training label is fixed as $\boldsymbol{Y}$.For random generated W and b, $\exists \overline{C}_i > 0$ which is independent of m, n_h and n_c such that*

$$E_{X|Y}\left\{ \overline{\mathcal{R}}_i(s_{i,\mathcal{S},C}) - \overline{\mathcal{R}}_i(s_i^*) \right\} \leq$$

$$32\overline{C}_i\left(1+\kappa\overline{R}\right)^2 \sqrt{\frac{n_h logm - log(n_h-1)}{\rho_i m}} + \inf_{f\in\mathcal{H}_n h}\left\{ \overline{\mathcal{R}}_i(f) - \overline{\mathcal{R}}_i(s_i^*) + \frac{\|f\|_{l_2}^2}{C} \right\}$$

where $\overline{R}$ is defined as Lemma3, and $i = 1,\cdots,n_c$.

Proof. According to the definition of $s_{i,\mathcal{S},C}$, we have:

$$\mathcal{R}_{S,i}(s_{i,\mathcal{S},C}) + \frac{\left\|s_{i,\mathcal{S},C}\right\|_{l_2}^2}{C} \leq \mathcal{R}_{S,i}(0) + \frac{\|0\|_{l_2}^2}{C}$$

Note that $\mathcal{R}_{S,i}(s_{i,\mathcal{S},C})$ is always non-negative, we then have $s_{i,\mathcal{S},C} \in \mathcal{B}_{\sqrt{C}}$.
Since l_1 in Eq.(24) is non-negative, we have

$$\mathbb{E}_{X|\mathbf{Y}}(l_1) = \int_0^t P_{X|\mathbf{Y}}(l_1 \geq \epsilon)d\epsilon + \int_t^{+\infty} P_{X|\mathbf{Y}}(l_1 \geq \epsilon)d\epsilon$$

Following the same technology employed in proofing theorem 2 of Ref. [44], based on Lemma 4 of this paper, we could reach that

$$\begin{aligned}&\mathbb{E}_{X|\mathbf{Y}}(l_1)\\ &\leq t + exp\left\{ -\frac{t^2\rho_i m}{128(1+\kappa\overline{R})^4} \right\} \left(\frac{64\sqrt{C}\,(n_c-1)\,(1+2\kappa R)}{mt} \right)^{n_h} \frac{tm^{n_h}}{n_h - 1}.\end{aligned}$$

Choose t such that:

$$t = 16\bar{C}_i(1+\kappa\overline{R})^2 \sqrt{\frac{n_h logm - log(n_h-1)}{\rho_i m}}$$

Meanwhile, select sufficient large $\overline{C}_i$ so that :

$$t \geq \frac{64\sqrt{C}\,(n_c-1)\,(1+2\kappa R)}{m}$$

We then have

$$\mathbb{E}_{X|\mathbf{Y}}(l_1) \leq \left[1 + \exp\left(-2\bar{C}_i\right)\right] t \leq 2t \leq 32\bar{C}_i(1+\kappa\overline{R})^2 \sqrt{\frac{n_h logm - log(n_h-1)}{\rho_i m}}$$

(28)

For $l2$ in Eq.(24), following the same technology used in theorem 5 of Ref.[22] and Theorem 2 of Ref. [44], we have

$$\mathbb{E}_{X|\mathbf{Y}}(l_2) = \inf_{f\in\mathcal{H}_{n_h}} \left\{ \overline{\mathcal{R}}_i(f) - \overline{\mathcal{R}}_i({s^*}_i) + \frac{\|f\|_{l_2}}{C} \right\} \tag{29}$$

The correctness of this Theorem directly follows Eq.(29), Eq.(28) and Eq.(24). □

Based on Assumption 1 and Eq.(27), we could now generalize the result of Theorem 4 to the overall multi-class problem with the following theorem.

Theorem 5. *Given training label $\mathbf{Y}$, and the hidden layer parameters W and b, we have*

$$\begin{aligned}&\mathbb{E}_{X|Y}\left\{ \overline{\mathcal{R}}(s_{1,\mathcal{S},C},\cdots,s_{n_c,\mathcal{S},C}) - \overline{\mathcal{R}}\left({s^*}_1,\cdots,{s^*}_{n_c}\right) \right\}\\ &\leq 32\overline{C}\left(1+\kappa\overline{R}\right)^2 \sqrt{n_c\left(\frac{n_h logm - log(n_h-1)}{\rho m}\right)}\\ &+ \inf_{\substack{f_i\in\mathcal{H}_{n_h}\\ i=1,2,\cdots,n_c}} \left\{ \overline{\mathcal{R}}(f_1,\cdots,f_{n_c}) - \overline{\mathcal{R}}\left({s^*}_1,\cdots,{s^*}_{n_c}\right) + \sum_{i=1}^{n_c}\frac{\|f_i\|_{l_2}}{C} \right\}\end{aligned}$$

where $\overline{C} = max\left\{\overline{C}_1,\cdots,\overline{C}_{n_c}\right\}$, and $\frac{1}{\rho} = \left(\sum_{i=1}^{n_c} {}^1/_{\rho_i}\right) = n_c\,(n_c-1)\sum_{i=1}^{n_c}\left({}^1/_{r_i}\right)$

Proof. Since $\sqrt{\cdot}$ is concave, it is easy to show that

$$\frac{1}{n_c}\sum_{i=1}^{n_c}\sqrt{{}^1/_{\rho_i}} \leq \sqrt{\frac{1}{n_c}\sum_{i=1}^{n_c}{}^1/_{\rho_i}} \tag{30}$$

Then correctiveness of this theorem directly follows Eq.(20), Eq.(30), Assumption 1 and Theorem 4. □

Finally, the condition of $\boldsymbol{Y}$ is removed according to the following theorem.

Theorem 6. *If the label are sampled independently with possibility $\mathbb{P}(Y_l = k) = p_k$, we have:*

$$\begin{aligned}&\mathbb{E}_{\mathcal{S}}\left\{ \overline{\mathcal{R}}(s_{1,\mathcal{S},C},\cdots,s_{n_c,\mathcal{S},C}) - \overline{\mathcal{R}}({s^*}_1,\cdots,{s^*}_{n_c}) \right\}\\ &\leq 32\overline{C}'(n_c)^{3/2}\left(1+\kappa\overline{R}\right)^2 \sqrt{\frac{n_h logm - log(n_h-1)}{m}}\\ &+ \inf_{\substack{f_i\in\mathcal{H}_{n_h}\\ i=1,2,\cdots,n_c}} \left\{ \overline{\mathcal{R}}(f_1,\cdots,f_{n_c}) - \overline{\mathcal{R}}({s^*}_1,\cdots,{s^*}_{n_c}) + \sum_{i=1}^{n_c}\frac{\|f_i\|_{l_2}}{C} \right\}\end{aligned}$$

where $\overline{C}' \approx \overline{C}\sum_{i=1}^{n_c} {}^1/_{p_i}$

Proof. According to theorem 5 we have

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}}\left\{\overline{\mathcal{R}}(s_{1,\mathcal{S},C},\cdots,s_{n_c,\mathcal{S},C})-\overline{\mathcal{R}}(s^*{}_1,\cdots,s^*{}_{n_c})\right\}\\
&=\mathbb{E}_{\mathbf{Y}}\left\{\mathbb{E}_{\mathcal{X}|\mathbf{Y}}\left\{\overline{\mathcal{R}}(s_{1,\mathcal{S},C},\cdots,s_{n_c,\mathcal{S},C})-\overline{\mathcal{R}}(s^*{}_1,\cdots,s^*{}_{n_c})\right\}\right\}\\
&\le 32\mathbb{E}_{\mathbf{Y}}\left(\sqrt{\frac{1}{\rho}}\right)\overline{C}\left(1+\kappa\overline{R}\right)^2\sqrt{n_c\frac{n_h logm-log(n_h-1)}{m}}\\
&+\inf_{\substack{f_i\in\mathcal{H}_{n_h}\\ i=1,2,\cdots,n_c}}\left\{\overline{\mathcal{R}}(f_1,\cdots,f_{n_c})-\overline{\mathcal{R}}(s^*{}_1,\cdots,s^*{}_{n_c})+\sum_{i=1}^{n_x}\frac{\|f_i\|_{l_2}}{C}\right\}
\end{aligned}$$

According to Jasen Inequality we have $\mathbb{E}_{\mathbf{Y}}\left(\sqrt{\frac{1}{\rho}}\right)\le\sqrt{\mathbb{E}_{\mathbf{Y}}\left(\frac{1}{\rho}\right)}$. According to the definition of ρ we have

$$\mathbb{E}_{\mathbf{Y}}\left(\frac{1}{\rho}\right)=n_c(n_c-1)m\sum_{i=1}^{n_c}\mathbb{E}_{\mathbf{Y}}\left({}^1/_{n_i}\right)$$

Then according to the delta method (see Theorem 5.5.24 of Ref. [74]), asymptotically we have

$$E_Y\left(\frac{1}{n_j}\right)\approx\frac{1}{mp_k},\forall j=1,2,\cdots,nc$$

That is to say $E_Y\left(\frac{1}{\rho}\right)\approx n_c\,(n_c-1)\sum_{i=1}^{n_c}\frac{1}{p_i}$. Then the proof is complete with the definition of $\overline{C}'$ and the continuity of square root function. □

In this section, we provide two kinds of convergence rates for the generalization performance of ELM_M^{AUC} under Assumption 1-3. For a specific training label set $\boldsymbol{Y}$, $\overline{\mathcal{R}}(s_{1,\mathcal{S},C},\cdots,s_{n_c,\mathcal{S},C})-\overline{\mathcal{R}}(s^*{}_1,\cdots,s^*{}_{n_c})$ will, on average, converge to the minimum approximation error on $\mathcal{H}_{n_h}$ with rate $O\left(n_c^{3/2}\sqrt{\frac{n_h logm}{\rho m}}\right)$, where the actual sample size is enlarged by a factor $\frac{1}{\rho}$. Since ρ is partially determined by m , we couldn't ignore it when considering the asymptotic complexity, which is similar as the generalization bound of binary AUC [68].Meanwhile, for extremely imbalance datasets, ELM_M^{AUC} may suffer from extra complexity and slower convergence rate depending on the ratio of the minor class. However after taking expectation over all possible $\boldsymbol{Y}$, if m is sufficient large, ELM_M^{AUC} could reach the convergence rate of $O\left(n_c^{3/2}\sqrt{\frac{n_h logm}{m}}\right)$. Moreover, for most of the practical applications, we often reach the conclusion that $n_c << m$ and $n_c << n_h$. Consequently, the rate of ELM^{AUC} converging to the best possible approximation error on $\mathcal{H}_{n_h}$ given W and b would be $O\left(\sqrt{\frac{n_h logm}{m}}\right)$, which could catch up with the result of learning to rank ELM algorithms [44].

5. Experiments

5.1. Comparing methods

In order to show the effectiveness of the proposed algorithms in this paper, we compare ELM^{AUC} with the following algorithms for binary classification problems.

- $W1ELM$: The W1ELM algorithm [56] for imbalanced datasets
- $W2ELM$: The W2ELM algorithm [56] for imbalanced datasets

Table 2: Basic Information for Binary-class Classification Benchmark Datasets

dataset	#Attribute	#instance	#class	IR
ionosphere	34	351	2	1.79
poker-8-9_vs_5	10	2075	2	82
poker-8-9_vs_6	10	1485	2	58.4
poker-8_vs_6	10	1477	2	85.89
poker-9_vs_7	10	244	2	29.50
splice	60	3175	2	1.08
yeast-2_vs_4	8	514	2	9.08
yeast1	8	1484	2	2.46
yeast4	8	1484	2	28.10
yeast6	8	1484	2	41.40
zoo-3	16	101	2	19.20

Table 3: Basic Information for Multi-class Classification Benchmark Datasets

dataset	#Attribute	#instance	#class	IR
abalone1	8	4177	18	40.53
balance	4	625	3	5.88
dermatology	34	357	6	5.55
ecoli	7	336	8	71.5
glass	9	214	6	8.44
hayes-roth	4	132	3	1.7
pageblocks	10	548	5	164
poker1	10	7749	6	497.25
poker2	10	29383	7	2704.25
segmentImb	19	749	7	20.30
sentimage	36	4435	6	2.58
shuttle	9	43500	7	5684.67
svmguide4	10	612	6	1.45
user knowledge	5	258	4	3.67
yeast	8	1484	10	92.6

- ELM : the original ELM algorithm
- LS^{AUC} : LS^{AUC} refers to ELM^{AUC} algorithm without hidden layer feature transformation
- $RELM$: Regularized Weighted Circular Complex Valued Extreme Learning Machine [57]
- $BELM5-15$: boosting ELM ensembling [58] method with 5,10,15 ELM weak learners respectively

For multi-class classification problems, we compare our proposed algorithms: ELM_M^{AUC} and ELM_{macro}^{AUC} with the multi-class extensions of $W1ELM$, $W2ELM$, $BELM5-15$,, $RELM$ and the following two extra algorithms.

- MDC: the Decomposition based Classification method [18] as discussed in Section 1.
- LS_M^{AUC}: LS_M^{AUC} refers to ELM_M^{AUC} algorithm without hidden layer feature transformation

Note that, for MDC algorithm, since the modified RankBoost algorithm is employed to optimize AUC_M, it is straightforward to obtain its objective function as :

$$\mathcal{L}_{MDC}=\sum_{i=1}^{n_c}\sum_{j\neq i}\frac{1}{n_i n_j}\sum_{n\in N_i}\sum_{k\in N_j}\exp\left(-\left(s_i(\mathbf{x_n}|\theta)-s_i(\mathbf{x_k}|\theta)\right)\right)\tag{31}$$

Obviously, from Eq.(31), the objective function for MDC could also be interpreted by the unified objective function proposed in section 3.2.

5.2. Parameter settings

For all the ELM based algorithms, C is selected from $2^{[-28:2:28]}$ and n_h is selected from $\{50,100,300,500,1000\}$. For LS_{AUC} and LS_{AUC_M} , merely C is selected since they're linear classifiers. For MDC , the parameter setting is the same as Ref. [18]. For each benchmark dataset 80% of the total samples are randomly chosen as training data with the

Table 4: Performance Comparisons based on AUC for Binary Classifications

		ELM^{AUC}	W1ELM	W2ELM	ELM	LS^{AUC}	BELM5	BELM10	BELM15	RELM
ionosphere	mean	**0.986**	0.9797**	0.9727***	0.9785***	0.9182***	0.9751***	0.9793***	0.9784***	0.9412***
	std	0.0103	0.0121	0.0156	0.0105	0.037	0.0166	0.0114	0.0146	0.0356
poker-8-9_vs_5	mean	0.7514	0.7597	0.7561	0.7248	0.5443***	0.7592	**0.7654**	0.7631	0.6803**
	std	0.0993	0.0872	0.1041	0.0967	0.1459	0.0945	0.0794	0.1108	0.1257
poker-8-9_vs_6	mean	**0.9913**	0.9753**	0.9743**	0.9547**	0.4314***	0.9735**	0.9733**	0.9724**	0.8279***
	std	0.0083	0.0257	0.0285	0.0795	0.1422	0.0282	0.0286	0.0311	0.0856
poker-8_vs_6	mean	**0.991**	0.931***	0.9324***	0.8973***	0.4329***	0.9326***	0.931***	0.9316***	0.8432***
	std	0.0092	0.0758	0.0521	0.0995	0.1448	0.0448	0.0592	0.0581	0.1103
poker-9_vs_7	mean	**0.9849**	0.9708	0.9552**	0.9547**	0.788***	0.95**	0.9516*	0.9635**	0.8484***
	std	0.0286	0.0386	0.0547	0.0588	0.1662	0.0617	0.092	0.0651	0.162
splice	mean	**0.9289**	0.9237***	0.9263	0.925**	0.9202***	0.9251***	0.9255**	0.9254***	0.8603***
	std	0.0085	0.0063	0.0079	0.0076	0.0065	0.0073	0.0059	0.0062	0.0427
yeast-2_vs_4	mean	0.9779	0.9764	0.9723	**0.9833-**	0.9296***	0.9764	0.979	0.9784	0.9581***
	std	0.0231	0.0178	0.0249	0.0133	0.0396	0.0189	0.0181	0.0195	0.0306
yeast1	mean	0.8083	0.8087	0.8074	**0.8089**	0.7993***	0.8087	0.8088	0.8087	0.7942***
	std	0.0222	0.0224	0.0218	0.022	0.0245	0.022	0.0224	0.0221	0.0259
yeast4	mean	0.8991	0.8987	**0.9019**	0.9009	0.8841**	0.9006	0.9001	0.9003	0.8815
	std	0.0271	0.0264	0.0296	0.0378	0.0354	0.0265	0.0331	0.0292	0.049
yeast6	mean	**0.9502**	0.9493	0.9497	0.9395***	0.9391***	0.9493	0.9493	0.9491	0.9409***
	std	0.0371	0.0367	0.0376	0.0482	0.0452	0.0367	0.0396	0.0377	0.0388
zoo-3	mean	**0.99**	0.9675**	0.9625*	0.965	0.9675**	0.97*	0.965*	0.9475***	0.97
	std	0.0262	0.0568	0.0686	0.0671	0.0494	0.0733	0.0745	0.0734	0.0441
win/tie/loss		0/11/0	5/6/0	5/6/0	6/4/1	11/0/0	6/5/0	6/5/0	6/5/0	9/2/0

ACCEPTED MANUSCRIPT

rest samples being test set. Such experiment is repeated 20 times. For each algorithms, the best average performance among all possible parameter combinations are reported for performance comparison.

As shown in Algorithm 1-3, our proposed methods aim at optimizing the average performance of AUC on training data. Correspondingly, line 7 in Algorithm 1, line 7 in Algorithm 2 and line 10 in Algorithm 3 show the normalization with respect to the hyper-parameter C. To be fair, all the other ELM based algorithms are normalized to optimize their corresponding mean performance. As a result, for all the other ELM based algorithms, the operation $C := \frac{C}{m}$ is done before computing $\boldsymbol{\beta}$.

All of the experiments are carried out with Matlab 2015a on a Intel(R) Core(TM) i7-4790 CPU with 20GB memory.

5.3. Datasets

The basic information of all the benchmark datasets are recorded in Table2 and Table3 for binary classification and multi-class classification respectively. #Attribute, #instances and #class represents the number of Attributes, the number of instances and the number of classes respectively. IR refers to Imbalance Ratio:

$$IR = \frac{n_{max}}{n_{min}}$$

where n_{max} is the size of the major class label set and n_{min} is that of the minor class.

For binary classification datasets recorded in table 2, ionosphere and splice could be obtained from the libsvm dataset website [1], while the rest 9 datasets could be obtained from KEEL-dataset repository [75][2]. For multi-class datasets recorded in table 3, sentimage, shuttle, svmguide 4 could be obtained from libsvm dataset website; balance,dermatology, ecoli, glass, hayes-roth page-blocks, yeast could be obtained from KEEL-dataset repository; the user knowledge dataset could be obtained from the UCImachine learning repository [3].

For libsvm dataset abalone, each age group is regarded as a class, then we merge the first three classes, and we also merge all the classes greater than 20 ,we name the obtained dataset abalone1. For poker dataset [4], we select the instances for the last 6, 7 classes of the original dataset and corresponding dataset is named as poker1 and poker2 respectively. In order to obtain segmentImb, 17, 33, 26, 13, 264, 231, 165 instances are sampled respectively from each of the 7 classes of the original dataset segment which could be downloaded from the libsvm website.

5.4. Metrics

For binary AUC optimization, AUC is employed as the comparison metric. While for multi-class AUC optimization, AUC_M is employed as the comparison metric.

5.5. Performance comparisons

For binary classification datasets, Table 4 shows the performance comparisons of all the involved algorithms. Table 5 shows the corresponding results for multi-class datasets. For Table 4 and Table 5, the best result for each dataset is in boldface. Moreover, ***, **, and * means ELM^{AUC} or ELM^{AUC}_M could significantly outperform the

[1]https://www.csie.ntu.edu.tw/ ~cjlin/libsvmtools/datasets/

[2]http://www.keel.es/

[3]http://archive.ics.uci.edu/ml/

[4]https://www.csie.ntu.edu.tw/ ~cjlin/libsvmtools/datasets/multiclass.html#poker

Table 5: Performance Comparisons based on AUC_M for Multi-class Classifications

		ELM_M^{AUC}	ELM_{macro}^{AUC}	W1ELM	W2ELM	ELM	LS_M^{AUC}	BELM5	BELM10	BELM15	RELM	MDC
abalone1	mean	**0.8029**	0.7982***	0.7975***	0.7923***	0.7917***	0.7988***	0.7977***	0.7978***	0.7979***	0.7845***	0.7868***
	std	0.0086	0.0054	0.0094	0.0103	0.0073	0.0065	0.0099	0.0096	0.0093	0.0106	0.0093
balance	mean	**0.9443**	0.9434	0.9387***	0.9424	0.9327***	0.7724***	0.9389**	0.9389***	0.939***	0.8174***	0.732***
	std	0.0129	0.0148	0.0132	0.0145	0.0135	0.0253	0.0155	0.0118	0.0109	0.068	0.0244
dermatology	mean	**0.9981**	0.998	0.9978	0.9977*	**0.9981**	**0.9981**	**0.9981**	**0.9981**	0.9979	0.9826***	0.9448***
	std	0.0016	0.0017	0.0018	0.0018	0.0017	0.0018	0.0014	0.0015	0.0016	0.0095	0.0801
ecoli	mean	**0.9091**	0.8963***	0.8828***	0.8811***	0.8765***	0.9003***	0.882***	0.8848***	0.8806***	0.8707***	0.7782***
	std	0.0155	0.0246	0.0285	0.0277	0.0281	0.0168	0.027	0.0281	0.0349	0.0396	0.0585
glass	mean	**0.9236**	0.9215	0.9216	0.9211	0.9168**	0.89***	0.9225	0.9224	0.9227	0.8629***	0.9122**
	std	0.0345	0.0337	0.0288	0.0314	0.0337	0.0386	0.0322	0.0347	0.0346	0.0471	0.0321
hayes-roth	mean	0.8862	0.8848	0.8487***	0.8512***	0.8419***	0.7783***	0.8451***	0.8452***	0.8473***	0.8612*	**0.9349—**
	std	0.0377	0.0348	0.0571	0.0548	0.0625	0.0292	0.0553	0.0547	0.051	0.0797	0.0357
pageblocks	mean	0.9589	0.9174***	0.9571	0.9577	0.9223***	0.9492**	0.9599	**0.9611**	0.9587	0.9276***	0.8944***
	std	0.0176	0.0264	0.0202	0.0151	0.0144	0.0248	0.0168	0.0154	0.0182	0.0292	0.0959
poker1	mean	0.833	**0.8423**	0.82	0.8235	0.7878***	0.5272***	0.816*	0.8214	0.8193*	0.7278***	0.6092***
	std	0.0284	0.0216	0.0281	0.033	0.0391	0.0732	0.0307	0.0282	0.0307	0.0521	0.0364
poker2	mean	**0.8336**	0.825*	0.8165***	0.8189**	0.7186***	0.5533***	0.8204***	0.8177***	0.8173***	0.731***	0.6081***
	std	0.0406	0.0244	0.0326	0.0311	0.0351	0.0574	0.0353	0.0313	0.0369	0.0523	0.0369
segmentImb	mean	**0.9872**	0.9842**	0.9841**	0.9837***	0.9722***	0.9698***	0.9844*	0.9845**	0.9838***	0.9567***	0.9694***
	std	0.0056	0.0082	0.0119	0.0086	0.0178	0.0094	0.0104	0.0099	0.0109	0.0146	0.0244
sentimage	mean	**0.9862**	0.9858**	0.9849***	0.9848***	0.9842***	0.9242***	0.9854***	0.9852***	0.9852***	0.973***	0.9727***
	std	0.0017	0.0017	0.0016	0.0017	0.0021	0.003	0.0017	0.0022	0.0019	0.0065	0.0027
shuttle	mean	0.9817	0.9394***	0.9782**	0.9769***	0.9439***	0.8554***	0.9735***	0.9732***	0.9732***	0.9605***	**0.9963—**
	std	0.0107	0.0402	0.0104	0.0099	0.0151	0.021	0.0103	0.0108	0.0105	0.0264	0.0097
svmguide4	mean	0.9254	0.9269	0.9188**	0.9182***	0.9212	0.8624***	0.9193***	0.9197***	0.9201**	0.8902***	**0.9513—**
	std	0.018	0.0177	0.0184	0.0177	0.0183	0.0162	0.0175	0.0165	0.0185	0.0239	0.0097
user knowledge	mean	0.9558	0.9507***	0.9462***	0.9434***	0.9478***	0.8516***	0.9471***	0.9479***	0.9467***	0.8998***	**0.9699–**
	std	0.0193	0.0184	0.0225	0.0209	0.0221	0.0162	0.0212	0.0195	0.0238	0.0569	0.0126
yeast	mean	**0.8866**	0.8716***	0.8799***	0.8797***	0.858***	0.8761***	0.88***	0.8798***	0.8797***	0.8543***	0.8665***
	std	0.0158	0.0169	0.0171	0.0168	0.0177	0.0165	0.0166	0.0179	0.0164	0.0234	0.0153
win/tie/loss		0/15/0	9/6/0	11/4/0	11/4/0	13/2/0	14/1/0	12/3/0	11/4/0	12/3/0	15/0/0	11/0/4

corresponding result according to pairwise wilcoxson rank sum test with $p < 0.01$, $p < 0.05$, $p < 0.1$ correspondingly. While —, –, - means ELM_M^{AUC} or ELM^{AUC} is significantly worse than the corresponding result with $p < 0.01$, $p < 0.05$, and $p < 0.10$ respectively. Finally, win/tie/loss counts the number of times ELM^{AUC} or ELM_M^{AUC} is significantly better than, not significant different with, or significantly worse than the corresponding algorithms according to pairwise wilcoxson rank sum test with $p < 0.1$.

For binary class datasets, the performance of ELM^{AUC} on ionosphere, splice, poker-8-9_vs_6, poker-8_vs_6, poker-9_vs_7, splice show significant advantage. For $zoo-3$ dataset, ELM^{AUC} is also better than all other algorithms with high mean performance difference. For multi-class datasets, ELM_M^{AUC} significantly outperforms most of the other algorithms on most of the investigated datasets.

Considering the boosting ELM ensemble models (BELM5, BELM10, BELM15), we see from both Table 4 and Table 5 that gradually adapting and ensembling different weighted ELM models couldn't improve the performance of a single weighted ELM significantly. A possible reason for this phenomenon is that BELM doesn't aim at optimizing AUC nor AUC_M directly.

Comparing MDC with our proposed algorithm ELM_M^{AUC}, since MDC is in nature a modified RankBoost algorithm, it could gradually improve the generalization performance by learning adaptive weights for different weak learners. Correspondingly, for hayes-roth, shuttle, svmguide4, and user knowledge MDC could generate promising results and significantly outperform other algorithms. However, according to Ref. [18], since MDC employs decision stump as the weak learners, it may suffer from poor results if the underlying dataset is not linear separable. As for typical instances, the performances of MDC on balance, ecoli, poker1, poker2, pageblocks, and dermatology are all unreasonable.

Since ELM_{macro}^{AUC} is designed to optimize AUC_{macro}, an alternative multi-class AUC metric that is similar to AUC_M, the performance of ELM_{macro}^{AUC} is no worse than ELM_M^{AUC} on 6 of the 15 benchmark datasets. However, according to the discussion in Section 3.2, AUC_{macro} is not a reasonable multi-class AUC metric. ELM_{macro}^{AUC} still can't optimize AUC_M directly. There's 9 benchmark datasets in which ELM_M^{AUC} exactly outperforms ELM_{macro}^{AUC}. For ecoli, pageblocks, poker2 and shuttle, this phenomenon is especially significant.

5.6. Efficiency comparisons

Aiming at comparing the running time of all the involved algorithms , we compare efficiency measured by the logarithm of average running time(s) ($log_2 T$) under different n_h. Figure 1 records the corresponding results of binary class datasets, while Figure 2 records that of the multi-class datasets.

In order to simplify the notation, the computational complexity of original ELM is denoted as $\mathcal{T}_{basic}$:

$$\mathcal{T}_{basic} = O\left(min\left\{n_h^3 + mn_h^2, m^3 + m^2 n_h\right\}\right)$$

For binary class datasets, the computational complexity of $W1ELM$, $W2ELM$, ELM, and the proposed algorithm ELM^{AUC} are all $\mathcal{T}_{basic}$. Correspondingly, the running time of these four algorithms are almost the same for all the involved datasets. For $RELM$, extra operations are needed for dealing the computations on complex domain. As a result, the running time needed for $RELM$ is always longer than the basic ELM. Finally, if employing n_w week learners, the computational complexity of $BELM$ is then $n_w\mathcal{T}_{basic}$. Furthermore, the running time taken by $BELM5$, $BELM10$, $BELM15$ are longer than all the other algorithms.

For multi-class datasets, the computational complexity of $W1ELM$, $W2ELM$ and ELM are still $\mathcal{T}_{basic}$. Correspondingly, as shown in Figure 2, ignoring the linear model LS_M^{AUC}, these three algorithms reaches the smallest log_2T for all the involved multi-class datasets. Similar as the binary case , the log_2T value for $RELM$ is always a little bit higher than the basic ELM. According the discussion in Section 3.3, both ELM_M^{AUC} and ELM_{macro}^{AUC} could be performed in $n_c\mathcal{T}_{basic}$ time. Meanwhile, given the number of weak learners n_w, $BELM$ could finish its computations within $n_w\mathcal{T}_{basic}$. As a result, in Figure 2, when $n_w > n_c$ the curve for $BELM5$, $BELM10$, $BELM15$ lies above that of ELM_{macro}^{AUC} as well as ELM_M^{AUC} and vice versa. Furthermore, for the prediction phase, the running time needed by $BELM$ is n_w times longer than that require for the basic ELM algorithms, ELM_M^{AUC} and ELM_{macro}^{AUC}. Given the number of weak learners n_w, the computational complexity for MDC should then be $O(n_w n_c \mathcal{T}_{weak})$, where T_{weak} is the computational complexity for the weak learner. If $n_w n_c$ is large, MDC will suffer from low computational efficiency. Empirically, according to Figure 2, the curve of MDC always lies above that of all the other algorithms unless n_h is sufficiently large.

According to Section 5.5 and 5.6, it is sufficient to claim the effectiveness of our proposed algorithms.

6. Conclusion and future work

For binary class problems, a novel AUC optimization algorithm called ELM^{AUC} is proposed based on ELM. According to theoretical and experimental analysis carried out in this paper, ELM^{AUC} could remain the same complexity as the basic ELM. For multi-class AUC optimization problems, AUC_M is proofed to a better multi-class AUC metric than AUC_{macro}. Subsequently, a unified objective function is proposed for multi-class AUC optimization. Then, two ELM algorithms named ELM_M^{AUC} and ELM_{macro}^{AUC} are proposed to solve multi-class AUC optimization problems. Again, these two algorithms have shown to be efficient. Subsequently, the theoretical analysis for ELM_M^{AUC} shows that the generalization performance ELM_M^{AUC} could asymptotically converge to the best possible performance for such ELM algorithms given the hidden layer parameters. Finally, the experiment results on 11 binary classification datasets and 15 multi-class classification datasets show the effectiveness of our proposed algorithms.

In the future, there are several issues that need to be further studied. First of all, as pointed out in the last section, it is valuable to further explore efficient algorithms to learn adaptive weights for ELM_M^{AUC}. Secondly, like other ELM counterparts, it is worthwhile to develop specific extensions of ELM^{AUC} and ELM_M^{AUC} to fit very large datasets and online learning scenario. Last but not least, since we only discuss the shallow models of ELM^{AUC} and ELM_M^{AUC}, it is also necessary to introduce hierarchical ELM models into this framework to learn better features.

Acknowledgments

This paper is supported by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, National Key Technology RD Program in 12th Five-year Plan of China (No. 2013BAI13B06). Meanwhile, we want to express our utmost gratitude to the editors and reviewers in charge for their valuable suggestions.

References

[1] J. A. Hanley, B. J. McNeil, The meaning and use of the area under a receiver operating characteristic (roc) curve, Radiology.

[2] C. Cortes, M. Mohri, AUC optimization vs. error rate minimization, in: Advances in Neural Information Processing Systems 16 Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada, 2003, pp. 313–320.

[3] A. H. Alan, B. Raskutti, Optimising area under the roc curve using gradient descent, in: In ICML, ACM Press, 2004.

[4] T. Calders, S. Jaroszewicz, Efficient AUC Optimization for Classification, Springer Berlin Heidelberg, 2007.

[5] T. Joachims, A support vector method for multivariate performance measures, in: Proceedings of the 22nd International Conference on Machine Learning, ACM Press, 2005, pp. 377–384.

[6] T. Joachims, Training linear svms in linear time, in: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006, 2006, pp. 217–226. doi:10.1145/1150402.1150429.

[7] T. Pahikkala, A. Airola, H. Suominen, J. Boberg, T. Salakoski, Efficient auc maximization with regularized least-squares, in: Proceedings of the 2008 conference on Tenth Scandinavian Conference on Artificial Intelligence: SCAI 2008, 2008.

[8] W. Gao, Z. H. Zhou, On the consistency of auc pairwise optimization, Computer Science.

[9] W. Gao, L. Wang, R. Jin, S. Zhu, Z. H. Zhou, One-pass auc optimization, Artificial Intelligence 236 (2013) 906–914.

[10] D. J. Hand, R. J. Till, A simple generalisation of the area under the roc curve for multiple class classification problems, Machine Learning 45 (2) (2001) 171–186.

[11] N. Lachiche, P. A. Flach, Improving accuracy and cost of two-class and multi-class probabilistic classifiers using roc curves., in: Machine Learning, Proceedings of the Twentieth International Conference, 2003, pp. 416–423.

[12] R. Wang, K. Tang, Feature selection for mauc-oriented classification systems, Neurocomputing 89 (10) (2011) 3954.

[13] P. Honzik, P. Kucera, O. Hyncica, V. Jirsik, Novel method for evaluation of multi-class area under receiver operating characteristic, in: International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, 2009, pp. 1–4.

[14] B. Yang, The extension of the area under the receiver operating characteristic curve to multi-class problems, in: Isecs International Colloquium on Computing, Communication, Control, and Management, 2009, pp. 463–466.

[15] K. Li, X. Kong, Z. Lu, W. Liu, J. Yin, Boosting weighted elm for imbalanced learning, Neurocomputing 128 (5) (2014) 15–21.

[16] P. Honzik, P. Kucera, O. Hyncica, D. Haupt, New feature selection method for multi-class data: Iteratively weighted auc (iwa)., in: IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Idaacs 2011, Prague, Czech Republic, September 15-17, 2011, pp. 336–340.

[17] X. Lu, K. Tang, X. Yao, Evolving Neural Networks with Maximum AUC for Imbalanced Data Classification, Springer Berlin Heidelberg, 2010.

[18] K. Tang, R. Wang, T. Chen, Towards maximizing the area under the roc curve for multi-class classification problems., in: AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, Usa, August, 2011.

[19] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences, Journal of Machine Learning Research 4 (6) (2003) 170–178.

[20] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (13) (2006) 489 – 501, neural NetworksSelected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN '04)7th Brazilian Symposium on Neural Networks. doi:http://dx.doi.org/10.1016/j.neucom.2005.12.126.

[21] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, Neural Networks, IEEE Transactions on 17 (4) (2006) 879–892. doi:10.1109/TNN.2006.875977.

[22] X. Liu, S. Lin, J. Fang, Z. Xu, Is extreme learning machine feasible? a theoretical assessment (part i)., IEEE Transactions on Neural Networks & Learning Systems 26 (1) (2015) 7–20.

[23] S. Lin, X. Liu, J. Fang, Z. Xu, Is extreme learning machine feasible? a theoretical assessment (part ii)., IEEE Transactions on Neural Networks & Learning Systems 26 (1) (2015) 7–20.

[24] Z. Shao, M. J. Er, An online sequential learning algorithm for regularized extreme learning machine, Neurocomputing 173, Part 3 (2016) 778 – 788. doi:http://dx.doi.org/10.1016/j.neucom.2015.08.029.

[25] L. N.-Y., H. G.-B., S. P., S. N., A fast and accurate online sequential learning algorithm for feedforward networks, IEEE Transactions on Neural Networks 17 (6) (2006) 1411–1423, å 434. doi:10.1109/TNN.2006.880583.

[26] Z. Xu, M. Yao, Z. Wu, W. Dai, Incremental regularized extreme learning machine and its enhancement, Neurocomputing 174, Part A (2016) 134 – 142. doi:http://dx.doi.org/10.1016/j.neucom.2015.01.097.

[27] F. G., H. G.-B., L. Q., G. R., Error minimized extreme learning machine with growth of hidden nodes and incremental learning, IEEE Transactions on Neural Networks 20 (8) (2009) 1352–1357, å 211. doi:10.1109/TNN.2009.2024147.

[28] J. Zhai, Q. Shao, X. Wang, Architecture selection of elm networks based on sensitivity of hidden nodes, Neural Processing Letters 44 (2) (2015) 1–19.

[29] J. Cao, J. Hao, X. Lai, C. M. Vong, M. Luo, Ensemble extreme learning machine and sparse representation classification, Journal of the Franklin Institute.

[30] J. Cao, K. Zhang, M. Luo, C. Yin, X. Lai, Extreme learning machine and adaptive sparse representation for image classification, Neural Networks the Official Journal of the International Neural Network Society 81 (C) (2016) 91–102.

[31] J. H. Zhai, H. Y. Xu, X. Z. Wang, Dynamic ensemble extreme learning machine based on sample entropy, Soft Computing 16 (9) (2012) 1493–1502.

[32] N. Zhang, S. Ding, J. Zhang, Multi layer elm-rbf for multi-label learning, Applied Soft Computing 43 (C) (2016) 535–545.

[33] Y. Kongsorot, P. Horata, Multi-label classification with extreme learning machine, in: International Conference on Knowledge and Smart Technology, 2014, pp. 81–86.

[34] Q. Liu, S. Zhou, C. Zhu, X. Liu, J. Yin, Mi-elm: Highly efficient multi-instance learning based on hierarchical extreme learning machine, Neurocomputing 173, Part 3 (2016) 1044 – 1053. doi:http://dx.doi.org/10.1016/j.neucom.2015.08.061.

[35] Q. Wang, W. Wang, R. Nian, B. He, Y. Shen, K.-M. Bjrk, A. Lendasse, Manifold learning in local tangent space via extreme learning machine, Neurocomputing 174, Part A (2016) 18 – 30. doi:http://dx.doi.org/10.1016/j.neucom.2015.03.116.

[36] D. Sovilj, E. Eirola, Y. Miche, K.-M. Bjrk, R. Nian, A. Akusok, A. Lendasse, Extreme learning machine for missing data using multiple imputations, Neurocomputing 174, Part A (2016) 220 – 231. doi:http://dx.doi.org/10.1016/j.neucom.2015.03.108.

[37] C. L. Lekamalage, Y. Yang, G. B. Huang, Z. Zhang, Dimension reduction with extreme learning machine., IEEE Transactions on Image Processing 25 (8) (2016) 3906–3918.

[38] M. D. Tissera, M. D. McDonnell, Deep extreme learning machines: supervised autoencoding architecture for classification, Neurocomputing 174, Part A (2016) 42 – 49. doi:http://dx.doi.org/10.1016/j.neucom.2015.03.110.

[39] S. Huang, B. Wang, J. Qiu, J. Yao, G. Wang, G. Yu, Parallel ensemble of online sequential extreme learning machine based on mapreduce, Neurocomputing 174 (PA) (2016) 352–367.

[40] M. V. Heeswijk, Y. Miche, E. Oja, A. Lendasse, Gpu-accelerated and parallelized elm ensembles for large-scale regression, Neurocomputing 74 (16) (2011) 2430–2437.

[41] J. Chen, G. Zheng, H. Chen, Elm-mapreduce: Mapreduce accelerated extreme learning machine for big spatial data analysis 45 (5) (2013) 400–405.

[42] J. Xin, Z. Wang, C. Chen, L. Ding, G. Wang, Y. Zhao, Elm* : distributed extreme learning machine with mapreduce, World Wide Web-internet & Web Information Systems 17 (5) (2013) 1189–1204.

[43] W. Zong, G.-B. Huang, Learning to rank with extreme learning machine, Neural Processing Letters 39 (2) (2014) 155–166. doi:10.1007/s11063-013-9295-8.

[44] H. Chen, J. Peng, Y. Zhou, L. Li, Z. Pan, Extreme learning machine for ranking: Generalization analysis and applications, Neural Networks 53 (2014) 119 – 126. doi:http://dx.doi.org/10.1016/j.neunet.2014.01.015.

[45] Z. Huang, Y. Yu, J. Gu, H. Liu, An efficient method for traffic sign recognition based on extreme learning machine., IEEE Transactions on Cybernetics.

[46] D. B. Heras, F. Argüello, P. Quesadabarriuso, Exploring elm-based spatial-spectral classification of hyperspectral images, International Journal of Remote Sensing 35 (2) (2014) 401–423.

[47] J. J. D. M. S. Junior, A. R. Backes, Elm based signature for texture classification, Pattern Recognition 51 (2016) 395–401.

[48] H. Liu, S. Li, C. Jiang, H. Liu, Sentiment analysis of chinese micro blog based on dnn and elm and vector space model, in: Proceedings of ELM-2015, Springer International Publishing, 2016.

[49] K. S. Neethu, T. S. Jyothis, J. Dev, Text classification using km-elm classifier, in: International Conference on Circuit, Power and Computing Technologies, 2016.

[50] Z. Wang, Y. Parth, Extreme learning machine for multi-class sentiment classification of tweets, in: Proceedings of ELM-2016, 2016.

[51] S. Saraswathi, S. Sundaram, N. Sundararajan, M. Zimmermann, Icga-pso-elm approach for accurate multiclass cancer classification resulting in reduced gene sets in which genes encoding secreted proteins are highly represented, IEEE/ACM Transactions on Computational Biology & Bioinformatics 8 (2) (2011) 452–63.

[52] X. Lu, H. Zou, H. Zhou, L. Xie, G. B. Huang, Robust extreme learning machine with its application to indoor positioning., IEEE Transactions on Cybernetics 46 (1) (2016) 194–205.

[53] Z. Yang, P. Zhang, L. Chen, Rfid-enabled indoor positioning method for a real-time manufacturing execution system using os-elm, Neurocomputing 174 (PA) (2016) 121–133.

[54] F. Dwiyasa, M. H. Lim, Y. S. Ong, B. Panigrahi, Equality constrained-optimization-based semi-supervised elm for modeling signal strength temporal variation in indoor location estimation, in: Proceedings of ELM-2016, Springer International Publishing, 2016.

[55] G. Huang, G. B. Huang, S. Song, K. You, Trends in extreme learning machines: A review, Neural Networks the Official Journal of the International Neural Network Society 61 (2015) 32–48.

[56] W. Zong, G. B. Huang, Y. Chen, Weighted extreme learning machine for imbalance learning, Neurocomputing 101 (3) (2013) 229–242.

[57] S. Shukla, R. N. Yadav, Regularized weighted circular complex valued extreme learning machine for imbalanced learning (2015) 1–1.

[58] K. Li, X. Kong, Z. Lu, W. Liu, J. Yin, Boosting weighted elm for imbalanced learning, Neurocomputing 128 (5) (2014) 15–21.

[59] B. Mirza, Z. Lin, K. A. Toh, Weighted online sequential extreme learning machine for class imbalance learning, Neural Processing Letters 38 (3) (2013) 465–486.

[60] B. Mirza, Z. Lin, N. Liu, Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift, Neurocomputing 149 (PA) (2015) 316–329.

[61] B. Mirza, Z. Lin, Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification, Neural Networks 80 (C) (2016) 79–94.

[62] J. Zhai, S. Zhang, C. Wang, The classification of imbalanced large data sets based on mapreduce and ensemble of elm classifiers, International Journal of Machine Learning & Cybernetics (2015) 1–9.

[63] B. Krawczyk, Gpu-accelerated extreme learning machines for imbalanced data streams with concept drift, Procedia Computer Science 80 (2016) 1692–1701.

[64] S. J. Lin, C. Chang, M. F. Hsu, Multiple extreme learning machines for a two-class imbalance corporate life cycle prediction, Knowledge-Based Systems 39 (3) (2013) 214–223.

[65] C. M. Vong, W. F. Ip, P. K. Wong, C. C. Chiu, Predicting minority class for suspended particulate matters level by extreme learning machine, Neurocomputing 128 (5) (2014) 136–144.

[66] T. Fawcett, An introduction to roc analysis, Pattern Recognition Letters 27 (8) (2006) 861–874.

[67] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Järvinen, J. Boberg, An efficient algorithm for learning to rank from preference graphs, Machine Learning 75 (1) (2009) 129–165.

[68] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, D. Roth, Generalization bounds for the area under the roc curve, Journal of Machine Learning Research 6 (2) (2005) 2005.

[69] S. Agarwal, P. Niyogi, Generalization bounds for ranking algorithms via algorithmic stability, Journal of Machine Learning Research 10 (1) (2009) 2009.

[70] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, T. Salakoski, Learning to rank with pairwise regularized least-squares, Sigir Workshop on Learning to Rank for Information Retrieval (2007) 27–33.

[71] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Jrvinen, J. Boberg, An efficient algorithm for learning to rank frompreference graphs, Machine Learning 75 (1) (2009) 129–165.

[72] M. Belkin, I. Matveeva, P. Niyogi, Regularization and Semi-supervised Learning on Large Graphs, Springer Berlin Heidelberg, 2004.

[73] G. H. Golub, C. F. Van Loan, Matrix computations (3rd ed.), 1996.

[74] G. Casella, R. L. Berger, statistical inference, Duxbury Press, 2002.

[75] J. Alcal-Fdez, A. Fernndez, J. Luengo, J. Derrac, S. Garca, Keel datamining software tool: Data set repository, integration of algorithms and experimental analysis framework., Journal of Multiple-Valued Logic & Soft Computing 17 (2011) 255–287.