Iterative Extreme Learning Machine for Single Class Classifier using General Mapping Convergence framework

NGUYEN HA VO, MINH-TUAN T. HOANG, HIEU T. HUYNH, JUNG-JA KIM, YONGGWAN WON.

1 Department of Computer Engineering, Chonnam National University, 300 Yongbong-Dong, Buk-Gu, Kwangju 500-757, REPUBLIC OF KOREA
2 Division of Bionics and Bioinformatics, Chonbuk National University, 664-14 St. #1 Dukjin-Dong, Dukjin-Gu, Chonbuk 561-756, REPUBLIC OF KOREA

Abstract: Single Class Classification (SCC) is the problem to distinguish one class of data (called positive class) from the rest data of multiple classes (negative class). SCC problems are common in real world where positive and unlabeled data are available but negative data is expensive or very hard to acquire. In this paper, extreme leaning machine (ELM), a recently developed machine learning algorithm, is fused with mapping convergence algorithm that is based on the support vector machine (SVM). The proposed method achieves both high accuracy in classification, very fast learning and high speed in operation.

Key-Words: Single Class Classification, Extreme Learning Machine, Mapping Convergence

1 Introduction

Single Class Classification (SCC) or One Class Classification is the problem to distinguish one class of data (called positive class) from the rest data of multiple classes (negative class). SCC problems are common in real world where positive and unlabeled data are available but negative data is expensive or very hard to acquire. For examples, data for normal peoples are widely available, but that for patients, acquired after many tests and procedures, are expensive. In converse, data for patients are easily collectable at the hospital. In this case, data include only positive cases.

Conventional learning methods, which generally perform competitive learning, are not suitable for SCC because of the serious unbalance of positive and negative class, or lack of negative class. They could ignored the class having relatively ignorable number of data and declare all the data samples as the major class, but still have high accuracy. Thus, we will use only positive and unlabeled samples to build a classifier. Of course, the absence of negative examples has some consequences, and one should not suppose results as good as two-class problem.

A common approach to SCC is based on probability density function (pdf) [1]-[4]. Typically, a pdf is estimated from the training examples using an appropriate density estimation technique, and then a probability threshold is selected. Input samples which produce a value larger than the threshold are classified as positive class, and others are negative. Probability density function is not easy to estimate, especially in high-dimensional cases.

Another common approach is to find a boundary (close or open) or hyper-sphere which surrounds the region containing positive data [6]-[10]. The key for these methods is how to determine the boundary close to positive data without negative data. Tax and Duin [10] suggested creating outliers uniformly in and around positive class. The fraction of accepted outlier by the classifier is an estimate of the volume of the feature space covered by the classifier and an optimization of the parameters can be performed. The number of such artificial outliers increase vastly in very high dimensional data, thus this method becomes infeasible.

In [11] – [14], Yu has a different way to find the boundary. The mapping convergence (MC), general mapping convergence (GMC) and support vector mapping convergence (SVMC) proposed by Yu, use the set U of unlabeled samples, besides positive samples. Then, the natural “gap” between positive and negative data in the feature space can be found by incrementally labeling negative data from U using a margin...
maximization algorithm \( \text{SVM} \). The margin maximization algorithm ensures xMC (GMC, MC, and SVMC) algorithms’ loops converge fast and efficiently. xMC’s outstanding classification performance had been proved on various domains of real data sets such as text classification, letter recognition, diagnosis of breast cancer. However, SVM is pretty slow, especially for a large volume of high dimensional data set.

Recently, a new learning algorithm, extreme learning machine (ELM), proposed by G. B. Huang [15]-[18], is available for training single hidden layer feed-forward neural networks (SLFN). This algorithm tends to provide good generalization performance with extremely fast learning speed. Some comparison between SVM and ELM conducted in [15]-[17] and [19]. Based on margin maximization idea, SVM is still comparable with xMC algorithms in order to have a fast, accurate and stable Single Class Classifier.

The rest of this paper is organized as follow. Some previous work, briefly introduction to ELM and xMC, are presented in section 3, and experiment details and the results are given in section 4. Finally, conclusions and further works are mentioned in section 5.

2 Related Works

2.1 Extreme Learning Machine (ELM)

Unlike popular implementations such as Back-Propagation (BP) for Single hidden Layer Feed-forward Neural networks (SLFNs), in ELM, one can arbitrarily choose the values for the weights from the input layer to the hidden layer and the biases for the hidden units without further training. After that, the hidden layer and the output layer of the SLFNs can be simply considered as a linear system. Therefore, the output weights of SLFNs can be analytically determined through simple generalized inverse operation on matrix of the outputs from the hidden layer for all input data. ELM can avoid common difficulties in tuning/adjustment methods such as stopping criteria, learning rate, learning epochs, and local minima.

For \( N \) distinct samples \( \{\mathbf{x}_i, \mathbf{t}_i\}_{i\in[1..N]} \) where \( \mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \in \mathbb{R}^n \) and \( \mathbf{t}_i = [t_{i1}, t_{i2}, \ldots, t_{im}]^T \in \mathbb{R}^m \), a standard SLFNs with \( N \) hidden nodes, activation function \( g(\mathbf{x}) \) can be modeled as

\[
\sum_{j=1}^{n} \mathbf{b}^*_j g(\mathbf{w}_j^* \cdot \mathbf{x}_i - b_j^*) \cdot \mathbf{t}_j^* \quad \text{where} \quad \mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^T \quad \text{is the weight vector connecting the } j \text{-th hidden neuron and the input neurons,} \\
\mathbf{b}_j = [b_{j1}, b_{j2}, \ldots, b_{jn}]^T \quad \text{is the weight vector connecting the } j \text{-th hidden node and the output node, and} \\
b_j \text{ is the threshold (bias) of the } j \text{-th hidden node.}
\]

That standard SLFNs can approximate \( N \) samples with zero error means that \( \sum_{j=1}^{n} \| \mathbf{p}_j / \mathbf{t}_j \| > 0 \), i.e. there exist \( \mathbf{b}, \mathbf{w}, \mathbf{b}_j \) such that

\[
\sum_{i=1}^{n} \mathbf{b}^*_j g(\mathbf{w}_j^* \cdot \mathbf{x}_i - b_j^*) \cdot \mathbf{t}_j^* \quad \text{for all } \mathbf{t}_j. 
\]

The above equation can be written compactly as

\[
\mathbf{H} \mathbf{\hat{b}} = \mathbf{T}
\]

where

\[
\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_N, \mathbf{b}_1, \ldots, \mathbf{b}_N, \mathbf{x}_1, \ldots, \mathbf{x}_N) \equiv \left[ \begin{array}{c|c|c|c|c|c} g(\mathbf{w}_1 \cdot \mathbf{x}_1 - b_1) & \ldots & g(\mathbf{w}_1 \cdot \mathbf{x}_N - b_1) \\ \vdots & \ddots & \vdots \\ g(\mathbf{w}_n \cdot \mathbf{x}_1 - b_n) & \ldots & g(\mathbf{w}_n \cdot \mathbf{x}_N - b_n) \end{array} \right]_{n \times n}
\]

The solution is

\[
\mathbf{\hat{b}} = \mathbf{H}^+ \mathbf{T} = \mathbf{H}^\dagger \mathbf{T}
\]

where \( \mathbf{H}^+ \) is Moore-Penrose generalized inverse of matrix \( \mathbf{H} \).

As mentioned in [15], we have some important properties of the solution

1. Minimum training error.
2. Smallest norm of weights
3. The minimum norm least-squares solution of

\[
\mathbf{H} \mathbf{\hat{b}} = \mathbf{T}
\]

The solution is unique, which is \( \mathbf{\hat{b}} = \mathbf{H}^+ \mathbf{T} \).

In summary, we have the ELM algorithm as follows:

Step 1: Randomly assign input weights \( \mathbf{w}_i \) and bias \( b_i, i=1..N \).

Step 2: Calculate the hidden layer output matrix \( \mathbf{H} \) for all data samples

Step 3: Calculate the output weight \( \mathbf{\hat{b}} = \mathbf{H}^+ \mathbf{T} \) where

\[
\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N]^T
\]

2.2 Mapping Convergence algorithm
The key idea of mapping convergence (MC) algorithm is to exploit the natural “gap” between the positive and the negative classes in the feature space by incrementally labeling negative data from U using a margin maximization algorithm \( \text{svm} \) (SVM). The MC algorithm can be divided into two parts:

1. First, MC uses the given positive samples \( P \) with or without the unlabeled sample set \( U \) to define a draft classifier (loose classifier) using algorithm \( \{1\} \). Then, the draft classifier is applied to \( U \) to obtain the truly “strong negative” sample set \( N_0 \). The “strong negative” samples are the ones very far from positive region defined by \( P \). All remaining unlabeled samples \( \hat{U} \), which is classified as positive by \( \{1\} \), will be used in the second step. Note that \( U = \hat{U} + N_0 \).

2. Use a margin maximization algorithm \( \{2\} \) on the positive set \( P \) and the negative set \( N(=N_0) \) for initial point) to construct a new classifier. Apply this classifier on the remaining unlabeled samples \( \hat{U} \). Samples that are classified as negative \( (N_{i+1} \text{ set}) \) will be merged into existing negative set \( N \) set. Note that \( N = \sum_{i=0}^{t} N_i \), and \( \hat{U}(t) \equiv \hat{U}(t/1) / N_i \). This step will be looped until there are no more unlabeled samples classified as negative, which is equivalent to empty \( N \).

SVMC and GMC are variants of MC algorithms. In SVMC, after each iteration new training set is redefined by adding the support vector of the current classifier with the new negative samples. Using this scheme, number of training samples at each iteration can be kept minimum, which obviously requires less computation for a training cycle.

In GMC [14], beside the criterion to stop the loop in MC where \( N_i \) is empty, another new criterion is abrupt decrement in the number of negative samples \( N_i \) detected at the looping time \( t \). When we have enough size of the given positive data \( P \) or there is a large “gap” between positive class and negative class, the stopping criterion of empty \( N_i \) reaches before the criterion of abrupt decrement. In this case, GMC behaves exactly the same as MC.

xMC algorithms, as implemented in [12]-[14], used SVM as \( \{2\} \). However, xMC algorithms are slow in a medium or large data set with high dimension. In the following section, we propose a novel algorithm to overcome the drawback of the xMC caused by using SVM as \( \{ 2 \} \). Our proposed algorithm aims at replacing the SVM used for the margin maximize

![Figure 1](image-url)

**Extended General Mapping Convergence Framework**

| Input: | - Positive data set \( P \), unlabeled data set \( U \) |
| Output: | - A classifier of positive and negative class. |

\[
\begin{align*}
\{1\} & \quad \text{A weak/loose classifier designed to classify only “strong negatives”} \ U. \\
\{2\} & \quad \text{A supervised learning algorithm}.
\end{align*}
\]

Algorithm:

1. Use \( \{1\} \) with \( P \) and \( U \) to get “strong negatives” set \( N_0 \).
2. \( i = 0 \).
3. Do loop
   3.1 \( U = U - N_i \); \( N = N \cup N_i \).
   3.2 Use \( \{2\} \) with \( P \) and \( N \) to get the new classifier.
   3.3 Apply the new classifier on \( U \), data that labeled “negative” are put into \( N_{i+1} \).
   3.4 \( K = \frac{|N_{i+1}||N_i|}{|N|} \)
   3.5 Exit the loop if \( (i > 0) \) and \( |N_i| = 0 \) or \( (K > k_1) \) or \( (K < k_1) \).

3.6 \( i = i + 1 \).

algorithm with a faster method. ELM for SLFNs can be a good candidate for replacement.

3. **Iterative ELM Classifier using GMC Framework**

As mentioned in [12]-[14], xMC algorithms used margin maximization algorithm to find the boundary between the negative and the positive classes iteratively. However, we think that we can extend to use some supervised classifier here.

Our proposed method is an extended version of GMC framework as shown in Fig. 1. The difference between our method and GMC algorithm proposed by Yu [14] is on the algorithm \( \{2\} \) and stopping criterion.

\( \{1\} \) is a weak/loose classifier designed to classify only “strong negatives” \( N_0 \) from \( U \) as negative ones. “Strong negatives” are the samples located very far from positive region. Weak classifier does not have to produce high accuracy in classification, and it is said to be “loose” meaning that the boundary does not tightly wrap the positive set \( P \). The most important thing is that it should not reject potential positive samples in \( U \) as negative. Many algorithms can be used for \( \{1\} \), i.e. OSVM, Rocchio, etc. In practice, even 2-class classifier with noisy positive and noisy negative could be used. In that case, reasonable percentage of \( U \) are merged into \( P \) (noisy positive), the remaining of \( U \) are considered as noisy negative. A conventional classifier with these noise positive and negative data could be used as a loose classifier.

\( \{2\} \) is not necessary to be a margin maximize algorithm. We argue that any type of supervised learning
method can be \( \{2\} \). In this study, we used ELM, a least squares errors method, for \( \{2\} \).

In step 3.4 in the Fig. 1, the first stopping criterion is satisfied when the gap between positive class and negative class is found. There are no more unlabeled samples classified as negative, which is equivalent to empty \( N \). The “nearly optimal” boundary is found, then the loop is exit.

The second exit criterion is the difference between MC and GMC algorithm. Without this criterion, GMC becomes MC. Supposed \( U \) is uniformly distributed in the feature space. We have \( |N| = 2^m \times |N_i| \) where \( m \) is the number of dimension of the feature space. Then, normally, \( |N|/|N_i| \gg 1 \) and \( K = (|N|/|N_i|) / |N| \equiv 1 \).

In original version of GMC [14], only upper boundary of \( K \) is used - the \( k_2 \) in Fig. 1, and this stopping criterion will become unstable when data set has a highly skewed distribution in feature space. However, with specified data set, a suitable range of \( k_2 \) can be found. In some data set in [14], \( k_2 \) is in the range \([2.5, 4]\). We should note that this criterion can provide its strength when the given positive data is under-sampled. Otherwise, this condition is never reached, and GMC and MC will be the same.

Therefore, in extended GMC framework, we proposed to use lower boundary of \( K \) as well - \( k_1 \) in Fig 1. In our experiment, \( k_1 \) is some value around 1. The new stopping criterion using \( k_1 \) help our algorithm converge faster while keeping high generalization.

4 Experiments

4.1 Experiment Methodology

The simulation studies were performed using MATLAB interface of LIBSVM\(^2\) version 2.82 (C++ complied code) for implementation of SVM based algorithms and using MATLAB code of ELM\(^3\). We compared our proposed method with GMC and 5 other methods below:

- OSVM is One Class Support Vector Machine implemented in LIBSVM.
- IELM, ISVM are Ideal ELM and Ideal SVM respectively, are trained from completely labeled training data.
- ELM NN, SVM NN are ELM with Noisy Negative and SVM with Noisy Negative respectively. They are trained using positive data, with unlabeled data as a substitute for negative data.

4.2 Data Sets

To evaluate our proposed method, we conducted performance comparison with many other algorithms for a real data sets: Diabetes\(^4\). The data set consists of 768 samples belong to either positive or negative class.

**TABLE 1**

<table>
<thead>
<tr>
<th>Data set</th>
<th>#Training samples</th>
<th>#Testing samples</th>
<th>#Positive samples</th>
<th>#Negative samples</th>
<th>#Attr.</th>
<th>#Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>576</td>
<td>192</td>
<td>263</td>
<td>505</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

As proposed in [16], 75% and 25% samples are randomly chosen for training and testing at each trial respectively. All positive samples in training sets are put into \( P \) (positive set). All remaining data are put in \( U \) (unlabeled set). \( P \) and \( U \) are used for training. We tested each method for identifying both positive and negative samples in testing data set. The information of the data set, such as number of data, attributes and classes is listed in Table 1.

4.3 Result and Discussion

The parameters \( \pi \) and \( C \) of SVM algorithms is tuned and then chose \( C = 10 \). All remain parameters are set as default. 500 trials have been done for all the algorithm and the average results are shown on Table 2 and Table 3.

We have following observations from Table 2:

- Classification rate of IELM and ISVM are slightly lower than the results of ELM and SVM methods reported in [15] (77.57% and 77.31% respectively).
- Our proposed method and GMC have the highest classification performance among the methods using positive and unlabeled data, and just a bit lower than that of IELM and ISVM. However, their results are outperform most of classifier mentioned in [15], although they use only positive and unlabeled samples.
- OSVM has the support of only positive samples, thus it has the worst performance.
- Unlike the case of GMC and our proposed method, classification rates of ELM NN and SVM NN are hurt by the positive samples in unlabeled (noisy negative)

\(^2\)http://www.csie.ntu.edu.tw/~cjlin/libsvm
\(^3\)http://www.ntu.edu.sg/home/egbhuang/
\(^4\)http://www.ntu.edu.sg/home/egbhuang/diabetes.zip
In Table 3, we compared our proposed method and GMC. They have the same classification rate but our proposed method have much more faster training speed (about 12.6 times faster) without considering that MATLAB environment may run much slower than C++ environment. Moreover, since the number of hidden nodes required by our method is much smaller than the number of support vectors of GMC, the testing time of our methods is 373 times less than GMC.

5 Conclusion and Further Work
In this paper, we presented an extended version of general mapping convergence (GMC) algorithm implemented using Extreme Learning Machine, which computes an accurate classification boundary without relying on negative data by applying classifier on unlabeled data iteratively. It is not only have the same high accuracy as GMC, comparable to the ideal case (ordinary 2 class classification), but also have much faster speed.

Currently, our method only implemented by GMC framework, it could be speed-up more by apply the idea of SVMC method. In addition, more practical real problems will be investigated in the near future.

6 Acknowledgement
This work was supported by grant No. RTI-04-03-03 from the Regional Technology Innovation Program of the Ministry of Commerce, Industry and Energy (MOCIE) of Korea.

References:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time (s)</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training</td>
<td>Testing</td>
</tr>
<tr>
<td></td>
<td>Rate</td>
<td>Dev.</td>
</tr>
<tr>
<td>GMC</td>
<td>0.00050</td>
<td>0.01866</td>
</tr>
<tr>
<td></td>
<td>0.77935</td>
<td>0.78835</td>
</tr>
<tr>
<td>Testing</td>
<td>0.00751</td>
<td>0.00509</td>
</tr>
<tr>
<td></td>
<td>0.76979</td>
<td>0.76908</td>
</tr>
<tr>
<td></td>
<td>0.02598</td>
<td>0.02793</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th># Nodes</th>
<th>SVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>330.406</td>
</tr>
<tr>
<td></td>
<td>156.3806</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>

| Dev.     | 0.01860   |
|         | 0.00050   |
|         | 0.01960   |


