Low-Interference Output Partitioning for Neural Network Training

Shang Yang, Sheng-Uei Guan, Wei Fan Li, and Lin Fan Zhao

Abstract—This paper presents a new output partitioning approach with the advantages of constructive learning and output parallelism. Classification error is used to guide the partitioning process so that several smaller sub-dimensional data sets are divided from the original data set. When training each sub-dimensional data set in parallel, the smaller constructively trained sub-network uses the whole input vector and produces a portion of the final output vector where each class is represented by one unit. Three classification data sets are used to test the validity of this algorithm, while the results show that this method is feasible.

Index Terms—Constructive learning algorithm, output partitioning, parallel growing, output interference

I. INTRODUCTION

Neural networks [1]-[3], evolutionary algorithms [4], fuzzy logic [5], [6] and other methods have been proposed to tackle classification problems. Among them, neural networks based solutions have attracted much attention and become one of the most popular techniques for classification.

However, when neural-network is applied to real-world classification problems, it still suffers from some drawbacks, especially when used in large-scale problems. Internal interference exists during the training process [7], whenever updating the weights of hidden units the influence from two or more output units due to clash in their weight-updating directions [1].

The strategy “divide-and-conquer” is applied. The interference among outputs can be reduced by dividing the original problem into several sub-problems. However, several important issues are raised: how to divide the original problem into several smaller and simpler problems, how to construct a network module to learn each of the sub-problem, how to combine the individual modules into the solution to the original task. Up to now, there are several approaches to tackle these issues: functional modularity [8], domain decomposition, class decomposition [9], [10] and state decomposition [11].

At the same time, parallel training has also been used to gain faster training. By training several sub networks at the same time, the time spent on training can be greatly reduced [1].

For our work, we applied the output partitioning approach. A data set to be classified can be partitioned into several smaller sub-dimensional data sets with distinct classes. Each sub-dimensional data set is then handled by a smaller sub-network using the whole input vector as input and producing a portion of the final output vector. This method reduces computational time and improves performance.

In Section II, we briefly recall the constructive learning algorithm. The concept of output partitioning is described in Section III. The proposed partitioning algorithm is then described in Section IV. In Section V, experiments based on partitioning are implemented with results analyzed. Finally, the conclusions are presented in Section VI.

II. CONSTRUCTIVE BACKPROPAGATION (CBP) NEURAL NETWORK ALGORITHM

Constructive Learning Algorithm consists of Dynamic Node Creation method [12], Cascade-Correlation [13] as well as its variations [14]-[16], Constructive Single-Hidden-Layer Network [17] and Constructive Backpropagation [16] (CBP) and etc. For our work, CBP is used.

III. SUB-GROUP MODEL OF INPUT ATTRIBUTES

A. Output Attribute Group Model

All of the output attributes are partitioned into r sub-group containing at least one output:

\[ E = \sum_{p=1}^{P} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2. \]

\[ = \sum_{p=1}^{P} \sum_{k_1=1}^{S_1} (o_{pk_1} - t_{pk_1})^2 + \sum_{k_2=1}^{S_2} (o_{pk_2} - t_{pk_2})^2 + \ldots + \sum_{k_r=1}^{S_r} (o_{pk_r} - t_{pk_r})^2 \]

\[ = \sum_{p=1}^{P} \sum_{k=r+1}^{S_r} (o_{pk_r} - t_{pk_r})^2 + \sum_{p=1}^{P} \sum_{k_2=s_{r+1}}^{S_2} (o_{pk_2} - t_{pk_2})^2 + \ldots + \sum_{p=1}^{P} \sum_{k_2=s_{r+1}}^{S_r} (o_{pk_r} - t_{pk_r})^2 \]

\[ \sum_{p=1}^{P} \sum_{k=r+1}^{S_r} (o_{pk_r} - t_{pk_r})^2 + \sum_{p=1}^{P} \sum_{k_2=s_{r+1}}^{S_2} (o_{pk_2} - t_{pk_2})^2 + \ldots + \sum_{p=1}^{P} \sum_{k_2=s_{r+1}}^{S_r} (o_{pk_r} - t_{pk_r})^2 \]

\[ = (S_1 + S_2 + \ldots + S_r = K) \] (1)

Especially, \( E_1, E_2, \ldots, E_r \) are independent from each other. And the sum of them must be less than \( E_{th} \).

B. Sub-Network Model

Sub-NN, sub-NN, sub-NN, ... sub-NN, replace the original network after grouping. These sub-networks are trained by CBP network. Each sub-NN produces only a portion of the result while the final result is generated by integrating the sub-networks by ensemble learning method.
IV. PARTITIONING ALGORITHM BASED ON OUTPUT ATTRIBUTES

A. Definition

CBP neural networks are very sensitive to the change of training time. If training time is too short, the neural network won’t be able to produce good result. However, long training will result in overfitting and poor bad generalization. In this article, the validation set is applied to determine the training time [18], [19].

A dataset is divided into three sub dataset: a training set is used to train the network; a validation set is used to evaluate the quality of the network to avoid overfitting during the training; finally, a test set is used to evaluate the resultant network. In this paper, the size of training, validation and test size is 50%, 25% and 25% of the dataset’s total available patterns.

Training error $E$ is mean square error percentage [18]. It is used to reduce the number of the coefficients in formula (1) and dependence on the range of output values.

$$E = 100 \frac{o_{\text{max}}-o_{\text{min}}}{\kappa \rho} \sum_{p=1}^{\rho} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2$$ (2)

In the above formula, $o_{\text{max}}$ and $o_{\text{min}}$ are the maximum and minimum output values in formula (1-1).

$E_{tr}(t)$ is per pattern’s average error of training network upon epoch $t$. $E_{eq}(t)$ is the corresponding error on validation set and is used to determine the time to stop training. $E_{eq}(t)$ is the test error, used to describe the quality of the network. $E_{opt}(t)$ stands for the minimum validation error from the start to epoch $t$.

$$E_{opt}(t) = \min_{tr\leq t} E_{eq}(t')$$ (3)

The relative increase of the validation error over the minimum so far is defined as the generalization loss at epoch $t$.

$$GL(t) = 100 \frac{E_{eq}(t)}{E_{opt}(t)} - 1$$ (4)

Training will stop if the generalization loss is too high. Otherwise, it will result in overfitting. A training strip of length $m$ [18] is defined as the sequence of $m$ times repeating from $n+1$ to $n+m$. Especially, $n$ can be divided exactly by $m$. During the training strip, training progress is measured by $P_m(t)$ which means how much larger the average error is than the minimum.

$$P_m(t) = 1000 (\frac{\sum_{t'_{m-1}}^{t_{m}} E_{eq}(t')}{m \min_{t'_{m-1}}^{t_{m}} E_{eq}(t')} - 1)$$ (5)

B. Process for Sub-Network Growing and Training

Fig. 2 shows the procedure for sub-network growing and training.

$$\begin{align*}
\text{Construct and initialize sub-NN} \\
\text{sub_epoch} = 1 \\
\text{While (total_epoch < Z)} \\
\quad \text{Train the current configuration of sub-NN for one epoch} \\
\quad \text{if (sub_epoch = 1)} \\
\quad \quad \text{E_{eq} = E_{eq} (Record the weights accordingly as the optimal weights)} \\
\quad \quad \text{if (sub_epoch} < 1) \\
\quad \quad \quad \text{E_{eq} = E_{eq} (Record the weights accordingly as the optimal weights)} \\
\quad \quad \quad \text{if (E_{eq} < \text{little improvement from last new hidden unit})} \\
\quad \quad \quad \quad \text{Break} \\
\quad \quad \text{Calculate E_{eq}} \\
\quad \text{end sub_epoch} \\
\text{end sub_epoch} \\
\text{end While} \\
\text{Calculate E_{eq} and Exit} \\
\end{align*}$$

We used $X = 80$ epochs while $Y = 500$ epochs.

V. OUTPUT PARTITIONING ALGORITHM

When comparing with conventional large neural networks, employing several smaller sub-networks for learning tends to have lower classification errors as it can reduce the internal interference. The classification error is an important measurement of NN’s performance. To obtain a lower classification error, an output partitioning algorithm, which employ several sub-network, was designed. We propose to get a near-optimal result via this algorithm. The details are presented as follows:

Step 1: Find the classification error $C_i$ of each class and order them in ascending order as \{\(C_a, C_b, \ldots, C_e\}\}, where \(C_a < C_b < \ldots < C_e\). To obtain the individual $C_i$, \(1 < i < K\) ($K$ is the number of class), all patterns not belong to class $i$ are labeled as patterns of class $\bar{i}$. A single NN is then used for the resulting two-class classification problem.

Step 2: Find the classification error of the every partition \{\(i, j\)\} \(1 < i < k, \ i \neq j\). And record the classification error as $C_{(i,j)}$.

Step 3: Judge the interference among every two classes according to the equation

$$F = C_{(i,j)} - (C_i + C_j)$$ (6)

Which means that if $F$ is negative the combination of $(i, j)$ lower the classification error, vice versa.

Step 4: Pick up a class $A$ in the sequence obtained in step1 and form a partition. If there is no group, create a new group and include $A$ in that group. Else, iteratively find a group, in which all the contained class has no interference with $A$, and include $A$ in that group. If failed to find such a group, build a new group and include that class. Delete $A$ from the sequence.

Step 5: Repeat step 4 until all until sequence obtained in
step1 is empty.

VI. EXPERIMENTAL RESULTS AND ANALYSIS

UCI machine learning datasets were used.

A. Glass

<table>
<thead>
<tr>
<th>TABLE I: GLASS INTERFERENCE MATRIX (UNIT %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output No.</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

Each diagonal element represents the classification errors for each individual class while the rest represents the classification error of partition \((i, j)\). Numbers in red represent results from interference-less pairs.

<table>
<thead>
<tr>
<th>TABLE II: GLASS INTERFERENCE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class No.</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

B. Vowel

<table>
<thead>
<tr>
<th>TABLE IV: VOWEL INTERFERENCE TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Number</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>

According to the classification errors for individual classes, we can obtain two kinds of ordering: ascending and descending order. The ascending order is 2-1-6-3-4-5. The descending order is 5-4-3-6-1-2. So, we can get two different group by using the two orders: \{2,1,6,3\} \{4\} \{5\} and \{5,6\} \{4,1\} \{3,2\}.

C. Thyroid

<table>
<thead>
<tr>
<th>TABLE VII: INTERFERENCE MATRIX (UNIT %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class No.</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Apparentley, all the three outputs don’t interference with one another. This means non-partitioning i.e. \{1,2,3\}.

Random partitioning means no specific strategy and/or ordering is applied.

<table>
<thead>
<tr>
<th>TABLE VIII: EXPERIMENTAL RESULTS OF THYROID (UNIT %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class Number</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

Random partitioning means no strategy and/or ordering is applied.

VII. CONCLUSION

This paper presented a new approach for growing and
training of neural network. By partitioning the output space, the performance of neural network is improved due to reduced interference. According to the experimental results of Glass and Vowel, this algorithm is better than full-partitioning, non-partitioning and the result from Yinan Qi et al [8]. We didn’t get a good result for Thyroid and the possible reason is the small output number. However, the strategy would likely work on datasets with higher dimensions.

REFERENCES


[23] M. H. Li, Based on the low interference the integration study method of neural network, Shanxi: Xi’an Jiaotong University, 2012.