Bioinspired Architecture Selection for Multitask Learning

Andrés Bueno-Crespo *, Rosa-Maria Menchón-Lara †, Raquel Martinez-España 1 and José-Luis Sancho-Gómez 2

1 Department of Computer Science, Universidad Católica de Murcia, Murcia, Spain, 2 Department of Information and Communications Technologies, Universidad Politécnica de Cartagena, Cartagena, Spain

OPEN ACCESS

Edited by:
Jose Manuel Ferrandez, Universidad Politécnica de Cartagena, Spain

Reviewed by:
Emilio Soria-Olivas, Universitat de València, Spain
D. Andina, Universidad Politécnica de Madrid (UPM), Spain
Francisco J. González-Serrano, Universidad Carlos III de Madrid, Spain

*Correspondence:
Andrés Bueno-Crespo
abueno@ucam.edu

† Present Address:
Rosa-Maria Menchón-Lara, Image Processing Laboratory, ETSI Telecomunicación, Universidad de Valladolid, Valladolid, Spain

Received: 08 March 2017
Accepted: 06 June 2017
Published: 22 June 2017

Citation:
Bueno-Crespo A, Menchón-Lara R-M, Martinez-España R and Sancho-Gómez J-L (2017) Bioinspired Architecture Selection for Multitask Learning. Front. Neuroinform. 11:39. doi: 10.3389/fninf.2017.00039

1. INTRODUCTION

The Hebbian learning in neural networks consists in establishing new synapses according to new lived experiences. Thus, this learning is directly related to the so-called structural plasticity which is the brain's ability to alter their physical structure in response to the learning of new information, skills, or habits. This means that when a human being modifies the knowledge about a particular field with new information (both from the same field and from other related fields), new neural connections are established and others are inhibited. This is how human beings can improve knowledge on a specific topic: by incorporating new experiences or related knowledge.

In this context, Multitask Learning (MTL) is a type of machine learning that tries to mimic the structural plasticity of human beings (Baxter, 1993; Caruana, 1995, 1998; Silver and Mercer, 2001). By using a shared representation, the MTL method learns simultaneously a problem (called the main task) along with other related problems (called secondary tasks). Thus, the artificial neural connections obtained by MTL are different from those obtained when the main task is learned by means of a single task learning (STL) scheme. This often leads to a better model for the main task, because there exists a transfer of information from the secondaries to the main task, i.e., the learning...
of the main task is modified by the information of the secondary tasks. However, in real world applications, it is not always easy to find tasks related with the main one, or to evaluate whether the relationship between them can produce a positive information transfer. Moreover, for machine learning, it is extremely difficult to determine whether the simultaneously training of several tasks can produce a better performance for one of them (considered the main task), in comparison with the result obtained when it is individually trained. This is because a task can contain information that can be helpful or harmful.

In Bueno-Crespo et al. (2015), a method to select related tasks with the main one is presented. Now, this method is used as a part of a new procedure to completely design MTL architectures. A particular pruning connections procedure leads to a positive transfer of information from the secondary tasks to the main task because only the most relevant connections are preserved. In this sense, the proposed method performs a complete design of the MTL networks. To achieve this, the method takes advantage of the benefits of Extreme Learning Machine algorithm (ELM) (Huang et al., 2006), specifically the Optimally Pruned ELM (OP-ELM) (Miche et al., 2010), and the Architecture Selection based on ELM (ASELM) procedures (Bueno-Crespo et al., 2013).

The rest of the paper is organized as follows: Section 2. describes the ASELM algorithm to design Multilayer Perceptrons (MLP). A summarized description of MTL is presented in Section 3. The proposed method is described in Section 4. Section 5 shows the results and finally, conclusions and prospective works close the paper.

2. ARCHITECTURE SELECTION USING EXTREME LEARNING MACHINE

The Extreme Learning Machine (ELM) is based on the concept that if the MLP input weights are fixed to random values, the MLP can be considered as a linear system and the output weights can be easily obtained by using the pseudo-inverse of the hidden neurons outputs matrix \( \mathbf{H} \) for a given training set. Although related ideas were previously analyzed in other works (Pao et al., 1994; Igelnik and Pao, 1997), Huang was the author who formalized it (Huang and Chen, 2007; Huang et al., 2011). He demonstrated that the ELM is an universal approximator for a wide range of random computational nodes, and all the hidden node parameters can randomly be generated according to any continuous probability distribution without any prior knowledge. Thus, given a set of \( N \) input vectors, a MLP can approximate \( N \) cases with zero error, \( \sum_{i=1}^{N} \| y_i - \mathbf{t}_i \| = 0 \), being \( y_i \in \mathbb{R}^m \) the output network for the input vector \( \mathbf{x}_i \in \mathbb{R}^n \) with target vector \( \mathbf{t}_i \in \mathbb{R}^m \). Thus, there exist \( \beta_j \in \mathbb{R}^m \), \( \mathbf{w}_j \in \mathbb{R}^n \) and \( b_j \in \mathbb{R} \) such that,

\[
y_i = \sum_{j=1}^{M} \beta_j f(\mathbf{w}_j \cdot \mathbf{x}_i + b_j) = \mathbf{t}_i, \quad i = 1, ..., N.
\]

where \( \beta_j = [\beta_{j1}, \beta_{j2}, ..., \beta_{jm}]^T \) is the weight vector connecting the \( j \)th hidden node with the output nodes, \( \mathbf{w}_j = [w_{j1}, w_{j2}, ..., w_{jm}]^T \) is the weight vector connecting the \( j \)th hidden node with the input nodes, and \( b_j \) is the bias of the \( j \)th hidden node.

For a network with \( M \) hidden nodes, the previous \( N \) equations can be expressed by

\[
\mathbf{H} \mathbf{B} = \mathbf{T},
\]

where

\[
\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_M, b_1, \ldots, b_M, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \\
= \begin{bmatrix} f(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \ldots & f(\mathbf{w}_M \cdot \mathbf{x}_1 + b_M) \\ \vdots & \ddots & \vdots \\ f(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \ldots & f(\mathbf{w}_M \cdot \mathbf{x}_N + b_M) \end{bmatrix}_{N \times M}
\]

\[
\mathbf{B} = \begin{bmatrix} \mathbf{\beta}_1^T \\ \vdots \\ \mathbf{\beta}_M^T \end{bmatrix}_{M \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}
\]

where \( \mathbf{H} \in \mathbb{R}^{N \times M} \) is the hidden layer output matrix of the MLP, \( \mathbf{B} \in \mathbb{R}^{M \times m} \) is the output weight matrix, and \( \mathbf{T} \in \mathbb{R}^{N \times m} \) is the target matrix of the \( N \) training cases. Thus, as \( \mathbf{w}_j \) and \( b_j \) with \( j = 1, \ldots, N \) are randomly selected, the MLP training is given by the solution of the least square problem of Equation (2), i.e., the optimal output weight layer is \( \hat{\mathbf{B}} = \mathbf{H}^+ \mathbf{T} \), where \( \mathbf{H}^+ \) is the Moore-Penrose pseudo-inverse (Serre, 2002).

ELM for training MLPs can be therefore summarized as shown in Algorithm 1.

**Algorithm 1 Extreme Learning Machine (ELM)**

Given a training set \( \mathcal{D} = \{ (\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbb{R}^n, \mathbf{t}_i \in \mathbb{R}^m, i = 1, \ldots, N \} \), an activation function \( f \) and an hidden neuron number \( M \),

1. Assign arbitrary input weights \( \mathbf{w}_j \) and biases \( b_j, j = 1, \ldots, M \).
2. Compute the hidden layer output matrix \( \mathbf{H} \) using Equation (3).
3. Calculate the output weight matrix \( \mathbf{B} = \mathbf{H}^+ \mathbf{T} \), where \( \mathbf{B} \) and \( \mathbf{T} \) are both defined in Equation (4).

ELM provides a fast and efficient MLP training (Huang et al., 2006), but it needs to fix the number of hidden neurons to obtain a good generalization capability. In order to avoid the exhaustive search for the optimal value of \( M \), several pruned methods have been proposed (Mateo and Lendasse, 2008; Miche et al., 2008a,b; Rong et al., 2008; Miche and Lendasse, 2009; Miche et al., 2010). Among them, the most commonly used is the ELM Optimally Pruned (OP-ELM) (Miche et al., 2010). The OP-ELM sets a very high initial number of hidden neurons \( M \gg N \) and, by using Least Angle Regression algorithm (LARS) (Similä and Tikka, 2005), sorts the neurons according to their importance to solve the problem (Equation 2). The pruning of neurons is done by utilizing Leave-One-Out Cross-Validation...
(LOO-CV) and choosing the combination of neurons (which have been previously sorted by the LARS algorithm) that provides lower LOO error. The LOO-CV error is efficiently computed using the Allen’s formula (Miche et al., 2010). For more detail, a summary of the OP-ELM algorithm is shown in Algorithm 2 (García-Laencina et al., 2011).

**Algorithm 2** Optimally Pruned-ELM (OP-ELM)

Given a training set \( D = \{(x_i, t_i) | x_i \in \mathbb{R}^n, t_i \in \mathbb{R}^m, j = 1, \ldots, N\} \), a mix of activation functions (sigmoid, gaussian, and linear), and a large number of neurons \( M \),
1. Randomly assign input weights \( \{w_i, b_i\}_{i=1}^M \).
2. Calculate the hidden layer output matrix \( H \) using \( X \) and input weights.
3. Ranking the hidden outputs using the MRSR algorithm, i.e., \( H \) is ranked, and set \( H^0 \) as an empty matrix.
4. for \( k = 1 \) to \( N \) do
5. Add the \( k \)-th node to the model \( \rightarrow H^k = [H^{k-1}, h_k] \), being \( h_k \) the \( k \)-th column of \( H \).
6. Computes LOO error (\( \epsilon_k^{PRESS} \)) with \( H^k \).
7. end for
8. Select the network size (\( M^* \)) according to \( \epsilon_k^{M^*} < \epsilon_k^{PRESS}, \forall k \in \{1, 2, \ldots, M\} \).
9. Calculate the output weights matrix: \( B^* = (H^*)^{-1}T \).

Recently, a new method to design MLP architectures has been presented in Bueno-Crespo et al. (2013). It is called ASELM (“Architecture Selection Using Extreme Learning Machine”) and is based on the OP-ELM. Thus, once the initial MLP architecture is defined, the OP-ELM optimally discards those hidden neurons whose combination of input variables is not relevant to the target task. Because of the binary value of the input weights, the selection of hidden nodes implies also the selection of those relevant connections between the input and hidden layers. Thus, only input connections corresponding to selected hidden neurons and with input weights values equal to 1 will be part of the final architecture. A summary of the ASELM algorithm is shown below (Algorithm 3).

### 3. MULTITASK LEARNING ARCHITECTURE

The MTL architecture for a neural network is similar to the classical scheme STL (Single Task Learning). They differ in that MTL scheme has an output for each task to be learned, whereas STL scheme has a separate network for each one (Figure 1). Thus, when we speak about MTL, we are referring to a type of learning where a main task and other tasks (considered as secondary tasks) are learned all at once in order to help learning of the main one.

In a MTL scheme, there is a common part shared by all tasks and a specific one for each task. The common part is formed by the weights connections from the input features to the hidden layer, allowing common internal representation for all tasks (Caruana, 1993). Thanks to this internal representation, learning can be transferred from one task to another (Caruana, 1998). The specific part, formed by the weights that connect the hidden layer to the output layer, specifically allows modeling each task from the common representation. The main problem with this type of learning is to find tasks related to the main one. Even in case of finding them, it may be difficult to know the kind of relationship they have, because it can be a positive or negative influence to learn the main task.

### 4. PROPOSED METHOD

The method proposed in this paper is called MTLASELM since it is based on the ASELM to design MTL architectures. To do this, it is necessary to introduce a couple of modifications to the original method so as to adapt it to MTL. Firstly, the targets of secondary tasks will be used as new input features (removing them from the outputs of the classic MTL scheme) so that a similar architecture to that shown in Figure 1A is obtained. There is only a single output corresponding to the main task and an input vector composed now by the original input features and the targets of secondary tasks. This network is designed and trained using ASELM which, as it was commented before, realizes a selection of hidden nodes that implies also the selection of those relevant connections between the input and hidden layer. The selection of relevant secondary tasks is now performed since they are part of the input vector.

In a second stage, a MTL architecture is created. The secondary tasks selected in the previous stage as the most relevant to learn the main task are included as output components in the MTL neural network. A scheme of the proposed method can be seen in Figure 2.

This idea of exchanging outputs for inputs is not new. Caruana proposed that some inputs may work better as outputs, i.e., as new secondary tasks (Caruana, 1998). This idea is very interesting in machine learning and it has been used, for example, for developing efficient procedures to classify patterns with missing data (García-Laencina et al., 2010, 2013).

MTLASELM method allows pruning to take place both at the hidden layer and the output layer, at the same time that provides a unique solution. This uniqueness comes from the
FIGURE 1 | Different learning schemes. In (A), a STL architecture is shown. It is used to solve a single task alone. In (B), a set of tasks are learned simultaneously by means of a MTL architecture. In this case, there is a common part (from the input to the hidden layer) and a specific part (from the hidden layer to the output) for each task.

binary initialization of the hidden weights, which eliminates the random component thereof.

To further clarify the MTLASELM method, the following section includes an example of how the method is applied step by step to solve a particular problem (Logic Domain problem).

5. EXPERIMENTS

In order to show the goodness of the MTLASELM method for designing an MTL architecture, results of classification test obtained with the single-task learning (STL), classic multitask learning (MTL), and MTLASELM have been compared. While the MTLASELM architecture is directly obtained by the proposed method, the best architecture for STL and MTL has been selected by cross-validation. For experiments, the three architectures are trained using the stochastic back-propagation with a cross-validation with 10-fold × 30 initializations. “Logic Domain,” “Monk’s Problems,” “Telugu,” “Iris Data,” and “User Knowledge Modeling” datasets, will be used to show the performance of the method. These data sets are available at the UCI ML Repository (Asuncion and Newman, 2007), excepting “Logic Domain” problem (McCracken, 2003). Specific details about results for each dataset are described below.

“Logic Domain” dataset is used to see how MTL architecture is created by the MTLASELM method. This dataset is a toy problem specially designed for multitask learning. In this problem, targets are represented by the combination of four real variables (from seven inputs: x₁,...,x₇), considering the first task as the main task, and the others as secondary ones.

Table 1 shows the logical expression for each task. Note, that the main task (T_p) is only determined by the first four features of the problem. The secondary tasks share one or more variables with the main one. Nevertheless, only the second secondary task (TSec2) shares a common logic subexpression (x₃ > 0.5 ∨ x₄ > 0.5) with the main task.

Initially, the neural network architecture is composed by M = 1023 (2ⁿ − 1, with n = 10; seven input features + three extra features corresponding to three secondary tasks) hidden units (see Figure 3). This suppose a large enough hidden layer number according to the ELM theory. Once this model is trained with ASELM method, the result is quite significant. The ASELM selects only two hidden neurons as the most relevant to learn the main task (Bueno-Crespo et al., 2013). By relevance order, these hidden weights are w₁₉₄ = [0 0 1 1 0 0 0 0 1 0] and w₇₆₈ = [1 1 0 0 0 0 0 0 0 0] corresponding to hidden neuron number 194 and 768, respectively. For simplicity, we will be referred to them as first neuron or w₁ and second neuron or w₂. From w₁, it can be observed that the first selected hidden neuron is only connected to input features x₃ and x₄, as well as the second secondary task (TSec2). From w₂, it follows that only x₁ and x₂ contribute to learning through their connection to the second hidden neuron (see Figure 3).

This means that only TSec2 is influencing in the learning of the Tp through the first neuron that learns the input features x₃ and x₄, which is an expected result according to the previous comment indicating the relationship between T_p and TSec2 (see Table 1). The second selected hidden neuron is only composed by the input features x₁ and x₂, without any input connection from the secondary tasks. Figure 4 shows the final architecture given by ASELM method.

Next, a MTL architecture is created considering as outputs those corresponding to the main task and secondary ones selected in the previous stage. The latter are incorporated into the output layer preserving the connections established by the
ASELM. In our case, only $T_{Sec 2}$ has been selected. Figure 5 shows the final $MTL_{ASELM}$ architecture. $MTL_{ASELM}$ has removed the input features $x_5$, $x_6$, and $x_7$, and has selected only 2 neurons in the hidden layer from the 1023 neurons initially considered.

Figure 6 shows the $MTL_{ASELM}$ schemes for other studied datasets. “Monk’s Problems” dataset is a collection of three toy problems that present the same domain (six input features). In this problem, the targets associated to each task are described by the logical relations. Thus, Monk 1 ($T_P$) is described by $x_1 = x_2$ or $x_3 = 1$; in Monk 2 ($T_{Sec 1}$) exactly two identities from $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$, $x_5 = 1$, $x_6 = 1$ must be satisfied; and in Monk 3 ($T_{Sec 2}$), $x_5 \neq 4$ and $x_2 \neq 3$ have to be fulfilled. $MTL_{ASELM}$ selects 14 neurons in the hidden layer from a total of 255. Figure 6A presents the first five neurons and the last one for the selected architecture. For example, if we

---

**TABLE 1** Description of the “Logic Domain” tasks.

| Task   | Logical expression                     |
|--------|----------------------------------------|
| $T_P$  | $(x_1 > 0.5 \lor x_2 > 0.5) \land (x_3 > 0.5 \lor x_4 > 0.5)$ |
| $T_{Sec 1}$ | $(x_2 > 0.5 \lor x_3 > 0.5) \land (x_4 > 0.5 \lor x_5 > 0.5)$ |
| $T_{Sec 2}$ | $(x_3 > 0.5 \lor x_4 > 0.5) \land (x_5 > 0.5 \lor x_6 > 0.5)$ |
| $T_{Sec 3}$ | $(x_4 > 0.5 \lor x_5 > 0.5) \land (x_6 > 0.5 \lor x_7 > 0.5)$ |

Each task is described by a logical combination of four input features.
observe the first neuron, it connects the input feature $x_3$ with the outputs of $T_P$ and $T_{Sec_1}$, but not with $T_{Sec_2}$. It can be observed that target associated to $T_P$ and $T_{Sec_1}$ match the value of $x_5$, what does not happen for $T_{Sec_2}$.

“Telugu” language dataset represents one of six languages designated a classical language of India. This dataset consists of three input features that represent language formants. For “Telugu,” $MTL_{ASELM}$ selects 4 neurons in the hidden layer from a total of 255 initial neurons. Figure 6B shows the final architecture obtained. As can be seen, this architecture uses only two of the three input features, what is quite interesting because in dialects with fewer than six vowels, two formants are only required to classify (Pal and Majumder, 1977).

“Iris Data” (Figure 6C) represent a dataset of three types of flowers represented by four input features. For this dataset, 5 neurons are selected in the hidden layer from a total of 63 neurons. It can be observed that the input feature $x_1$ has been removed. The results show that the proposed method has a much more simplified architecture than classical multitask learning, although the classification
test is similar due to the simplicity of the problem (see Table 2).

“User Knowledge Modeling” (Figure 6D) is the real dataset about the students’ knowledge status about the subject of Electrical DC Machines. The target is represented by four levels (very low, low, middle, and high). To give a multitasking approach a pairwise combination has been made ($T_P = \text{(very low } \lor \text{ low)}$, $T_{Sec_1} = \text{(low } \lor \text{ middle)}$, and $T_{Sec_2} = \text{(middle } \lor \text{ high)}$). It can be observed that $T_{Sec_1}$ is removed. It is because $T_{Sec_2}$ is more important to $T_P$, since $T_{Sec_2}$ represents its opposite. Finally, 5 neurons are selected in the hidden layer from a total of 127 initial neurons.

Table 2 shows the classification accuracy results for all the data sets. Because the Logic Domain is an easy problem to solve for an MLP in an STL scheme, the number of samples has been reduced to 50 so that multitask learning can be appreciated. With all training samples, the result between STL and MTL is practically invaluable. Taking into account this reduction of samples for the Logic Domain problem, MTLASELM presents better classification accuracy than STL and MTL. However, STL is better than the classic MTL, since MTL presents a completely interconnected scheme that is positively influenced by the related task ($T_{Sec_2}$) and negatively by unrelated tasks ($T_{Sec_1}$ and $T_{Sec_3}$). This is not a general rule but it is an empirical result that shows
that there are tasks that help the main task and others that are harmful.

For the rest of the data sets, the $MTL_{ASELM}$ always provides the best results on average with a low standard deviation. This robustness in the solution is due to the particular initialization of the hidden weights that $MTL_{ASELM}$ realizes.

To validate this assertion, a non-parametric statistical test has been performed. Specifically, the Wilcoxon Signed Ranks Test is used (Kruskal, 1957). A peer review has been performed. Comparing $MTL_{ASELM}$ to STL, the $p$-value obtained is 0.078, which indicates that there are significant differences to 92%, being the best $MTL_{ASELM}$. Likewise, when applying the test with $MTL_{ASELM}$ against classic MTL, the $p = 0.080$ indicates that there are significant differences to 92%, being $MTL_{ASELM}$ better than MTL. However, there are no significant differences between STL and MTL, since the $p = 0.683$.

6. DISCUSSION AND FUTURE WORK

This paper presents a method to select tasks to be used in a MTL scheme providing information about weight connection, hidden nodes, input features, and most helpful secondary tasks for the learning of the main task. This method has been named $MTL_{ASELM}$ because it is based on the ASEL algorithm (Bueno-Crespo et al., 2013), which proves to be an efficient method and single solution for the complete design of a MLP (input features, weights connections, and hidden nodes). By using secondary tasks as input features, $MTL_{ASELM}$ applies the ASEL on the initial network which only has a single output corresponding to the main task. Thus, irrelevant nodes and connections are eliminated, what implies a selection of features (among which are the secondary tasks). After this stage, a final network is built with a dimension in the output layer equal to the number of secondary tasks selected as relevant plus one. Thus, the main drawback of multitask learning is eliminated, i.e., the negative influence of unrelated tasks. In addition, the modification of ASEL method to adapt it for a multitask scheme is achieved not only to eliminate connections from inputs features to hidden layer, but also from hidden to output layer. It is worth highlighting that, as well as ASEL, the $MTL_{ASELM}$ method provides a single solution. This is due the fact that a binary initial selection of the hidden weights substitutes any random initialization process. Another important advantage is that it requires no parameter to be configured by the user. In the experiments section, it has been observed over real problems that the method $MTL_{ASELM}$ gets a simplified solution with good generalization capabilities, in comparison to those obtained by a fully connected solution given by the classic MTL scheme.

Authors are working on extending the method to other learning models, such as Radial Basis Functions (RBF). Applying $MTL_{ASELM}$ to regression problems is another research field since the ASEL is optimized for classification according to Huang et al. (2010). This limitation of the present method is due to the nature of ELM method, which is based on the pseudoinverse calculation. In this regard, we are working to use the sequential calculation pseudoinverse of Moore-Penrose (Van Heeswijk et al., 2011; Tapson and Van Schaik, 2013). Another line of research is to extend this method in the field of Deep Learning since new works on MultiTask Learning have lately appeared, most of them within the scope of Deep Learning (Liu et al., 2015; Thanda and Venkatesan, 2017).

AUTHOR CONTRIBUTIONS

AB: Proposed method, software programming, experiments, reviews, results, and conclusions. RMML: Introduction, software programming, contributions of ideas and reviews. RME: Method description, contributions of ideas and reviews. JS: Work direction, architecture selection, contributions of ideas, reviews, conclusions, and future work.

ACKNOWLEDGMENTS

This work has been supported by Spanish MINECO under grant TIN2016-78799-P (AEI/FEDER, UE). This paper is an extension of the one presented at the IWINAC 2015 conference (6th INTERNATIONAL WORK-CONFERENCE on the INTERPLAY between NATURAL and ARTIFICIAL COMPUTATION). Dedicated to the memory of Ph.D. Pedro José García-Laencina.
