Meta-learning based Optimization of Metabolic Pathway Data-Mining Inference System

Tomás Arredondo V.1, Wladimir Ormažábal O.1, Diego Candel C.2, and Werner Creixell13

1 Departamento de Electrónica
e-mail: tomas.arredondo@usm.cl
2 Departamento de Informática,
Universidad Técnica Federico Santa María
Av. España 1680, Valparaíso, Chile
3 CSIS The University of Tokyo, Tokyo, Japan

Abstract. This paper describes a novel meta-learning (MLL) based methodology used to optimize a neural network based inference system. The inference system being optimized is part of a bioinformatic application built to implement a systematic search scheme for the identification of genes which encode enzymes of metabolic pathways. Different MLL implementations are contrasted with manually optimized inference systems. The MLL based approach was found to be flexible and able to produce better results than manual optimization.

1 Introduction

A common problem in bioinformatic research consists of determining the genes that encode enzymes of a particular metabolic process of interest. Obtaining this information is essential to produce a functional map of the metabolic capabilities of the organism and can be a benefit in determining the advantages of using one specific microorganism in a particular application. In our research, we are interested in determining the genes which encode enzymes involved in the bacterial degradation of aromatic compounds that may be harmful to human health. Given that not all bacteria are capable of performing such degradation it is important to determine the genes that are responsible for said functions because it provides a fingerprint which can be used to assess and understand such capability in various bacteria.

It is toward this goal, that we search for the genes that encode the enzymes (i.e. proteins) which participate in the process of aromatic compound degradation. With these genes, a search for other such organisms can take place using well known tools and databases such as BLAST [1] and Vector NTI [2]. These tools perform DNA sequence alignment and search in order to find a sequence segment (known as an Open Reading Frame or ORF) which codifies a protein (known as a coding sequence or CDS). The data sources used for said comparisons have DNA, protein sequences and metabolic pathway maps stored within them (MetaCyc [3], UM-BBD [4], KEGG [5], NCBI [6] amongst others).
The integration of Bioinformatic data sources is a continuing problem given the complexity and skills required to utilize them in an efficient manner. With this in mind, we have produced two such applications: GeXpert ([7],[8]) and Phylogist [10]. Both of these systems attempt to improve and optimize the process of integrating data and applications from different public sources. Part of the workflow of these applications involves the selection of candidate genes which encode an enzyme of interest. In both of these applications, an inference system is used to classify the candidates from genes previously analyzed for an organism under study.

Previous approaches toward metabolic network reconstruction have used various algorithmic methods such as name-matching and using EC-codes to link metabolic information to genes (e.g. IdentiCS [9], metaSHARK [14]). The AUTOGRAPH (AUtomatic Transfer by Orthology of Gene Reaction Associations for Pathway Heuristics) method [15] uses manually curated metabolic networks, orthologs and their associated reactions to compare predicted gene-reaction associations. Our current approach in optimizing the inference system is applicable to any such data-mining application.

In Section 2, we describe the bioinformatics application and the inference system used. In Section 3, we expose design details and the proposed implementation solution. We describe our experiments and results obtained in Section 4. Finally in Section 5, we present conclusions and outline ideas for future work.

2 Bioinformatic Inference Systems

2.1 Integration Framework Process

Toward determining if certain genes participate in a metabolic pathway of interest (e.g. biodegrading of an aromatic compound), many current bioinformatic data-mining systems ([7]-[12]) utilize an iterative workflow. As in [12], an Artificial Neural Network (ANN) is used to estimate the quality of an alignment match of a DNA candidate sequence in terms of corresponding to the gene encoding a specific protein for a metabolic pathway of interest. This inference system, uses a selection of BLAST results as inputs and seeks to improve the selection of genes based on meta-optimization. In addition, the information provided by experts on previously researched genes must be scored in the organism under study and the inference system (i.e. the Learner) must learn from this information in order to generate future scorings for other genes.

2.2 Inference System Workflow

The idea behind our proposed workflow as used in [13] is that once an alignment is performed, the results returned by the BLAST algorithm are passed to an ANN classifier which provides the user with an indicator of its goodness and also permits a constant updating of the training data with the objective of continually improving the neural network.
The input parameters correspond to the BLAST sequence alignment scores, which are defined as follows:

- **Identity**: Coincidence percentage.
- **Positives**: Symbol similarity percentage.
- **E-value**: Random match result probability.
- **Gaps**: Missing sequence segment percentage.
- **BitScore**: Statistical significance measure.

The inference system output corresponds to an indicator of the goodness of the candidate gene alignment, which take continuous values in the interval [0, 1]. Values higher than 0.5 indicate a good candidate, otherwise it is a bad candidate.

3 Meta-learning based Solution

It is possible to find diverse approaches that have been categorized as MTL [16,17], all these approaches eventually aim to obtain the best result from the learning process. Some have centered in the study of the nature of the data as in the number of characteristics to analyze, the classes being classified or the amount of training data amongst others [16,17] (e.g. Landmarking [18]); others have applied combinations of base learners (e.g. Stacked Generalization [19]).

The majority of previous works [20–22], consider certain aspects of interest (e.g. learning parameters or learning model structural elements). To the best of our knowledge, only [23] approaches both parametric aspects and model configuration but in a manner specific to his context. Our current proposal is a general conceptual framework that generalizes the concept of MTL independent of the inference system (i.e. learner or weak learner) and the algorithm being used to optimize it (e.g. backpropagation, genetic algorithms, etc).

3.1 Training and Validation Data Definition

There are diverse possible data sets, given a determined context or problem of interest. Considering one of these possible sets, $\mathbf{D}$, as a set of $N$ elements:

$$\mathbf{D} = \{d_1, d_2, \ldots, d_N\}, |\mathbf{D}| = N$$

(1)

Where:

- $d_i = (I_i, O_i)$, with $1 \leq i \leq N$.
- $I_i$: Input values.
- $O_i$: Output values.
- $|X|$: Cardinality of the set $X$.

Following [24], it is possible to divide the set $\mathbf{D}$ in $r$ disjoint subsets $T$:

$$\mathbf{D} = T_1 \cup T_2 \ldots \cup T_r, \text{ with } T_i \cap T_j = \emptyset, \forall i, j \in \{1, 2, \ldots, r\}, \ i \neq j$$

(2)
Starting from these \( r \) subsets, we can generate a training set \((S_i)\) and a test set \((T_i)\). \( T_i \) corresponds to one of the \( r \) subsets, and \( S_i \) is the union of the \( r - 1 \) remaining subsets. We thus define a training set \( S_i \) as:

\[
S_i = D - T_i, \quad \text{with } D = \{T_1, T_2, \ldots, T_{i-1}, T_{i+1}, \ldots, T_r\}
\]

Considering expression (2) we establish that:

\[
D = T_i \cup S_i, \quad \forall i \in \{1, 2, \ldots, r\}
\]

### 3.2 Training Methods and Inference System Model

Extending the idea of learner (i.e. inference system) training used in [24], we define \( L_k \) as a learner formed by the pair (learning model, learning method) as:

\[
L_k = (\text{mod}_k, \text{tr}_k), \quad \text{with } \text{mod}_k \in \text{MOD}, \ \text{tr}_k \in \text{TR}
\]

Where \( \text{MOD} \) is the set of possible learners (e.g. neural network, fuzzy inference system, etc.) and \( \text{TR} \) is the set of possible learning methods that can be used to train those models (e.g. backpropagation, genetic algorithm, etc.).

Consider \( L \) as the set of possible different learners, that is the valid combinations \((\text{mod}_k, \text{tr}_k)\) of learning models and training methods:

\[
L = \{L_1, L_2, \ldots, L_k, \ldots\}, \quad \forall k \in \mathbb{N}
\]

Considering expressions (5) and (6), as well as the aforementioned restriction it is possible to assume that:

\[
|\text{MOD} \times \text{TR}| \geq |L|
\]

For every \( L_k \), a vector \( p \) can be defined that contains all parameters of the training method being used by \( L_k \). We let \( P_k \) be the set of all \( \gamma \) possible \( p \) vectors:

\[
\forall L_k \exists a P_k = \{p_{k,1}, p_{k,2}, \ldots, p_{k,\gamma}\}
\]

In a similar vein, we define the set \( M_k \) of all \( \theta \) possible inference system model configurations, each one of these represented by a vector \( m \):

\[
\forall L_k \exists a M_k = \{m_{k,1}, m_{k,2}, \ldots, m_{k,\theta}\}
\]

### 3.3 Hypothesis

We understand a hypothesis \( h \), as an instance of an inference system type (e.g. a neural network learner) after undergoing through the learning process described in [24]. There is an infinite set of hypotheses \( h \) associated with each one of the learners in \( L \). For example, \( H_A \) is the set of all possible hypotheses generated from learner \( L_A \):

\[
L_A \rightarrow H_A
\]
Finally, considering all factors previously mentioned it is possible to establish that a hypothesis \( h \) depends on the learner \( L_A \) being used, on the training data \( (S_i) \), on the model being used \( (M_A) \), and on the parameter values of the training method being used \( (P_A) \).

We define \( H^j_A(p_{A,1}, m_{A,j}, D) \) as the set of all hypotheses generated using vectors \( p_{A,1} \) and \( m_{A,j} \) while trained with \( S_i \) (1 ≤ i ≤ r) obtained from \( D \):

\[
H^j_A(p_{A,1}, m_{A,j}, D) = \{ h^1_A(p_{A,1}, m_{A,j}, S_1), \ldots, h^r_A(p_{A,1}, m_{A,j}, S_r) \} \quad (11)
\]

We define \( h_{A,i,j,i} \) as a hypothesis generated using vectors \( p_{A,1} \) and \( m_{A,j} \), and the training set \( S_i \). Also, the output for hypothesis \( h_{A,i,j,i} \) given an input \( I_x \) is defined as: \( h_{A,i,j,i}(I_x) \).

As seen in Fig. 1, the MTL seeks to obtain the best possible values for the parameters vector \( (p_{A,1}) \), this process is denoted as tuning. It is also tasked with finding the best possible values for the model configuration \( (m_{A,j}) \), this process is known as fitting. For a particular template, the generated hypotheses allow us to obtain a measure of fitness for the template than instantiated them. The measure of fitness used to evaluate the diverse hypotheses generated is the mean error over \( r \) of the test set \( T_i \) applied to all hypotheses (i.e. \( E_{MTL} \)). This fitness measure allows the MTL to evaluate and select the best combination of \( p_{A,1} \) and \( m_{A,j} \) found for a specific problem.

\[
E_{MTL} = \frac{1}{r} \sum_{i=1}^{r} \left( 1 - \frac{\text{Correctly classified}}{\text{Total cases}} \right) \quad (12)
\]

![Fig. 1. Inference System Meta-optimizer](image-url)
4 Experiments and Results

4.1 Neural Networks

We use a multilayer feed forward artificial neural network with one hidden layer and backpropagation for weight updating:

\[
\Delta w_{ij}(n + 1) = \eta \cdot \delta_{pj} o_{pj} + \alpha \cdot \Delta w_{ij}(n)
\] (13)

We applied two implementations of backpropagation, the first one (NN1) using with a fixed learning rate (\(\eta\)) and the second one (NN2) with a variable learning rate. According to [25], during each learning cycle the initial value of \(\eta\) decreases exponentially with the \(d\) factor until reaching the value \(\eta_{low}\). Upon reaching this value \(\eta_{low}\), \(\eta\) takes the value of \(\eta_{high}\) and this process repeats:

\[
\eta(t) = \eta(t - 1) \cdot \exp\left(\frac{\log\left(\frac{\eta_{low}}{\eta_{high}}\right)}{d}\right)
\] (14)

In Table 1, we show the values that are used for each variable. The chosen parameters are those that were considered to have the greatest possible impact upon hypothesis generation.

<table>
<thead>
<tr>
<th>Meta-learner</th>
<th>(N_i)</th>
<th>(N_h)</th>
<th>(N_o)</th>
<th>(\eta)</th>
<th>(\eta_{low})</th>
<th>(\eta_{high})</th>
<th>(\alpha)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NN1</td>
<td>4</td>
<td>1 - 30</td>
<td>1</td>
<td>0 - 0.1</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0 - 0.9</td>
<td>N.A.</td>
</tr>
<tr>
<td>NN2</td>
<td>4</td>
<td>1 - 30</td>
<td>1</td>
<td>0.3</td>
<td>0.01</td>
<td>0.01 - 0.3</td>
<td>0 - 0.9</td>
<td>20 - 84</td>
</tr>
</tbody>
</table>

To avoid overtraining, we utilized \(T_i\) to monitor the change in error according to (15). After each training round, in case of an increase in error the process is stopped. We restricted the possible maximum number of training iterations to 20000 and the minimum number to 1000. Toward the selection of training and test sets, we applied cross validation random sub-sampling [26], in a 70% - 30% proportion respectively.

\[
Error_i = \frac{1}{|T_i|} \sum_{x=0}^{|T_i|-1} \frac{1}{2} (O_j - h_{A_{i,j,i}}(I_x))^2
\] (15)

4.2 Meta-learning Algorithms

A genetic algorithm (GA) and Stochastic Hill Climbing (SHC) were used in order to implement the process of MTL. These modified the number of hidden layer neurons (\(N_h\)) as well as backpropagation parameters according to the test case (fixed or dynamic learning rate).
**Genetic Algorithm** The performance of the GA is associated with a series of parameters and considerations which determine its behavior. Previously, researchers have determined certain reasonable ranges for GA parameters to be used in a variety of optimization problems [27, 28]. From these, we have chosen the following values for GA settings and parameters: tournament selection, one Elite individual, 100 iterations, a 20 individual population, two point crossover, probability of crossover of 0.95, and probability of mutation of 0.01. In GA-NN1, a granularity of 5 bits was used for \( N_h \), 14 bits for \( \eta \), and 14 bits for the momentum (\( \alpha \)). For GA-NN2 we used 5 bits for \( N_h \), 6 bits for \( \eta_{high} \), 12 bits for \( \alpha \) and 6 bits for \( d \).

**Stochastic Hill Climbing** Another algorithm that was utilized for MTL was SHC [29]. This algorithm modified the same NN parameters as the GA. We used an equivalent number of objective function evaluations in order to have a comparable effort with respect to the GA. The parameters used were 200 iterations and 10 restarts. In SHC-NN1, we used a granularity of 5 bits for \( N_h \), 14 bits for \( \eta \) and \( \alpha \) was also 14 bits. For SHC-NN2 we used 5 bits for \( N_h \), 6 bits for \( \eta_{high} \), 12 bits for \( \alpha \) and 6 bits for \( d \).

### 4.3 Results

In Table 2, the total average correct % classification results reached by all classifiers is compared. GA-NNX corresponds to the GA based MTL training NNX and SHC-NNX to the SHC based MTL training NNX. Naive-NNX and Expert-NNX correspond to a superficial (3 hours) and a more detailed (6 hours) manual parameter search respectively.

<table>
<thead>
<tr>
<th></th>
<th>KT</th>
<th>JMP</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive-NN1</td>
<td>0.9472 ± 0.0236</td>
<td>0.9324 ± 0.0169</td>
<td>0.8878 ± 0.0239</td>
</tr>
<tr>
<td>Expert-NN1</td>
<td>0.9608 ± 0.0148</td>
<td>0.9513 ± 0.0300</td>
<td>0.9175 ± 0.0224</td>
</tr>
<tr>
<td>Naive-NN2</td>
<td>0.9540 ± 0.0145</td>
<td>0.9391 ± 0.0264</td>
<td>0.9013 ± 0.0235</td>
</tr>
<tr>
<td>Expert-NN2</td>
<td>0.9581 ± 0.0134</td>
<td>0.9513 ± 0.0264</td>
<td>0.9202 ± 0.0148</td>
</tr>
<tr>
<td>SHC-NN1</td>
<td>0.9635 ± 0.0143</td>
<td>0.9581 ± 0.0195</td>
<td>0.9256 ± 0.0182</td>
</tr>
<tr>
<td>SHC-NN2</td>
<td>0.9581 ± 0.0085</td>
<td>0.9567 ± 0.0182</td>
<td>0.9189 ± 0.0222</td>
</tr>
<tr>
<td>GA-NN1</td>
<td>0.9729 ± 0.0124</td>
<td>0.9702 ± 0.0166</td>
<td>0.9229 ± 0.0231</td>
</tr>
<tr>
<td>GA-NN2</td>
<td>0.9729 ± 0.0099</td>
<td>0.9608 ± 0.0227</td>
<td>0.9229 ± 0.0231</td>
</tr>
</tbody>
</table>

Table 3, presents the best parameter and model values obtained by the different methods considered for the KT, JMP and LB data respectively. In Table 4, we show cross validation results obtained when testing the learners obtained by GA-NN1 y GA-NN2 with all the data. The rows indicate the training data and the columns indicate the test data used.
Table 3. Best Parameter and Model Values Obtained

<table>
<thead>
<tr>
<th>MTL KT</th>
<th>N_l</th>
<th>N_h</th>
<th>N_o</th>
<th>η</th>
<th>η_low</th>
<th>η_high</th>
<th>α</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA-NN1</td>
<td>4</td>
<td>25</td>
<td>1</td>
<td>0.06256</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.3357</td>
<td>N.A.</td>
</tr>
<tr>
<td>GA-NN2</td>
<td>4</td>
<td>27</td>
<td>1</td>
<td>0.3</td>
<td>0.01</td>
<td>0.0471</td>
<td>0.4446</td>
<td>51</td>
</tr>
<tr>
<td>SHC-NN1</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>0.0318</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.6692</td>
<td>N.A.</td>
</tr>
<tr>
<td>SHC-NN2</td>
<td>4</td>
<td>19</td>
<td>1</td>
<td>0.3</td>
<td>0.01</td>
<td>0.0085</td>
<td>0.4775</td>
<td>53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MTL JMP</th>
<th>N_l</th>
<th>N_h</th>
<th>N_o</th>
<th>η</th>
<th>η_low</th>
<th>η_high</th>
<th>α</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA-NN1</td>
<td>4</td>
<td>27</td>
<td>1</td>
<td>0.07543</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.6649</td>
<td>N.A.</td>
</tr>
<tr>
<td>GA-NN2</td>
<td>4</td>
<td>30</td>
<td>1</td>
<td>0.3</td>
<td>0.01</td>
<td>0.0614</td>
<td>0.0718</td>
<td>25</td>
</tr>
<tr>
<td>SHC-NN1</td>
<td>4</td>
<td>25</td>
<td>1</td>
<td>0.6683</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.5249</td>
<td>N.A.</td>
</tr>
<tr>
<td>SHC-NN2</td>
<td>4</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.01</td>
<td>0.0342</td>
<td>0.7712</td>
<td>74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MTL LB</th>
<th>N_l</th>
<th>N_h</th>
<th>N_o</th>
<th>η</th>
<th>η_low</th>
<th>η_high</th>
<th>α</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>GA-NN1</td>
<td>4</td>
<td>15</td>
<td>1</td>
<td>0.09980</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.8825</td>
<td>N.A.</td>
</tr>
<tr>
<td>GA-NN2</td>
<td>4</td>
<td>27</td>
<td>1</td>
<td>0.3</td>
<td>0.01</td>
<td>0.0971</td>
<td>0.4428</td>
<td>81</td>
</tr>
<tr>
<td>SHC-NN1</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>0.0318</td>
<td>N.A.</td>
<td>N.A.</td>
<td>0.8692</td>
<td>N.A.</td>
</tr>
<tr>
<td>SHC-NN2</td>
<td>4</td>
<td>13</td>
<td>1</td>
<td>0.3</td>
<td>0.01</td>
<td>0.04</td>
<td>0.6793</td>
<td>68</td>
</tr>
</tbody>
</table>

Table 4. Cross Validation Results

<table>
<thead>
<tr>
<th></th>
<th>GA-NN1</th>
<th>GA-NN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>KT</td>
<td>0.9729 ± 0.0124</td>
<td>0.9621 ± 0.0188</td>
</tr>
<tr>
<td>JMP</td>
<td>0.9594 ± 0.0142</td>
<td>0.9702 ± 0.0166</td>
</tr>
<tr>
<td>LB</td>
<td>0.9364 ± 0.0247</td>
<td>0.9094 ± 0.0278</td>
</tr>
<tr>
<td></td>
<td>0.9270 ± 0.0222</td>
<td>0.9283 ± 0.0211</td>
</tr>
</tbody>
</table>

5 Conclusions and Future Work

As seen by the test results, a clear benefit of using MTL can be observed. Better results are obtained than in the manual optimization results and the complete automation of the search by using well known search algorithms (GA and SHC) alleviates the user from a tedious and non trivial task. Also, the proposed conceptual model leaves open a wide variety of potential enhancements and experiments without requiring model modifications and considers aspects previously left out by other learner models. The MTL software developed for this optimization is open source and is freely available [30].

The range of obtained performance results demonstrates that there is an impact in varying selected parameter and model values in fixed and dynamic learning rate cases. The utilization of the backpropagation method with a dy-
Dynamic learning rate does not improve obtained results, on the contrary it seems to worsen obtained results [25].

When comparing classification performance, we can observe that using a GA rather than SHC as the MTL produces better results. Even so, both GA and SHC obtain better average results than the manually optimized classifiers.

Future work includes the utilization of other algorithms in the MTL, the verification of their performance on a wider variety of problems as well as testing different parameters and model values of different granularity.

Acknowledgment

This research was partially financed by Universidad Técnica Federico Santa María DGIP Grants 231138 and 231021.

References