An algorithm for multi-task learning based on combinatorial optimization of U-curves

Giovana Kerche Bonás   *Marcelo da Silva Reis*
Marcos Medeiros Raimundo

Relatório Técnico - IC-PFG-22-43
Projeto Final de Graduação
2022 - Dezembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

An algorithm for multi-task learning based on combinatorial optimization of U-curves

Giovana Kerche Bonás* Marcelo S. Reis* Marcos M. Raimundo*

Abstract

Here we report the development of an algorithm that explores properties of U-shaped curves ("U-curves") in cost functions of multi-task transfer learning (MTL) models. The proposed algorithm works even with the insertion of different tasks that may or may not be related to the original task of learning. To find a global minimum of the described curve, a Boolean lattice is organized based on the enumeration of the search space with the weights of the group of different tasks used in training. This traversal of that lattice is carried out through a branch-and-bound procedure, in which the pruning criterion is the increase of the cost in a chain of that lattice. To benchmark the proposed algorithm against established MTL methods, we carried out computational experiments with both synthetic and real datasets. We expect that this proposed algorithm will represent a relevant alternative for multi-task learning models.

1 Introduction

The data collection process is an arduous assignment that can take years to complete but is essential for building machine learning models [1]. The procedure to create a machine learning model for a dataset we can name as a learning task, and if we have only one dataset it is called single-task learning [2]. However, there are situations that one cannot gather enough data to produce a high-quality machine learning model. One way to surpass such challenge is to use different sets of data (called tasks) that have some relationship with each other to improve the performance of these models. This is called multi-task learning and it is performed by exploring the relationship between these data it is possible to improve the models of individual tasks, as it provides a better generalization of them [3].

In the literature, finding this relationship between tasks (a shared representation) is explored in several ways, of which three have been gaining prominence, sharing based on data samples [4, 5], based on features [6, 7, 8], and based on parameters [9, 10]. However, in the case of sharing based on features or parameters, when we insert prior knowledge about the task domain in this shared representation, we impose the desired characteristics on the parameters or on the model's regularization terms, this practice runs the risk of forcing relationships between unrelated tasks or presumed relationships that are not present in the data, in addition to requiring prior knowledge of the base [3].

*Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, 13083-852.

In practice, different levels of regularization yield different models, and the fine-tuning of regularization hyper-parameters is a model selection task. The more a model is regularized, the more a generalization error is introduced due to the increased bias. On the other hand, too few regularization regularization might increase the generalization error due to the increase variance (i.e., the model starts to become too complex for the available data). In the context of multi-task learning, the same phenomenon is observed when one regulates the importance of samples from a source task: too few importance increases the bias error, whereas too much importance might increase the variance error.

In both cases, models can be organized in a partially order set (poset). For instance, all possibilities of considering or not the samples of a given source task, in a set S tasks, can be organized into a Boolean lattice with $2^{|S|}$ subsets, in which the a subset $A \subseteq S$ contains a subset $B \subseteq S$ iff consider the samples of all source tasks of B plus one or more another tasks. Such lattice has a remarkably property: for each chain, the generalization error of the models describe a U-shaped curve. Minimization of U-shaped curve cost functions is an optimization procedure that was explored previously in other applications, such as feature selection [11, 12].

In this work, we propose a branch-and-bound algorithm to find the source tasks that can improve generalization to a target task, based on U-curve optimization. This search occurs on a Boolean lattice that organizes all possibilities of considering or not the samples of a given source task. The algorithm uses the backtracking technique to seek the minimum global cost of the U-curve, promoting pruning whenever there is an increase in the cost of the loss function in validation. The algorithm yields the ideal group of tasks that help in the training of the target task and its importance in the training of the task.

The remainder of this work is organized as follows: In Section 2, we present some related works in multi-task learning. In Section 3, we provide some theoretical background necessary to follow this work. In Section 4, we introduce the proposed algorithm. In Section 5, we explain the experimental setup of this work, including the used datasets (both real and synthetic ones). In Section 6, we show and discuss results of computational experiments of the proposed method, benchmarking it against well-established methods in MTL. Finally, in Section 7, we recall the main contributions of this work and point out possible future directions in this research.

2 Related works

The sharing of information between tasks, essential for training multi-task learning, has been explored in different ways in the literature. The most used is performed as a form of regularization because through it, an inductive bias is introduced into the equation, penalizing models that do not have good generalizations and, consequently, helping in selecting models that would best predict the new input data [13].

This information sharing can occur in three main ways. First, feature sharing has been exploited by constraining all models to share the same common set of features as in l2,1-norm [6, 7, 8]. Second, parameter sharing allows tasks to share some model parameters, as in the low-rank representation [9, 10]. And finally, sharing data samples can be done by algorithms that try to find the best configuration of samples that can be used to improve the training of the individual task.

Multi-task learning has grown in Deep Learning, mainly through sharing hidden layer parameters. One of the reasons for this use is that multi-task learning increases the size of your dataset, as it generates the combination of datasets from each task. By adding these samples to the training set of the target learning task, the model will better learn what may be task-specific noise or biases.

In this study, we employed sample sharing, where initially, the algorithm calculates the probabilities of each sample/label for the task itself. Then the tasks are retrained by inserting training samples from other tasks. Other research from the literature has already explored this kind of knowledge transfer technique, such as in [4] that treats the problem of predicting the therapeutic success of a given combination of drugs for Human Immunodeficiency Virus-1 (HIV-1), or yet, in [5] where data about different patients with Epileptic Seizures are used to improve the training of new cases that the model does not know about.

The strategy used here has the advantage of being model agnostic. It does not make assumptions about the generation of tasks or the relationship between them so that it can deal with arbitrarily different data distributions for different tasks. In addition, it explores features of the model error that can be analyzed both in classifieds and regression models.

3 Theoretical background

In this chapter, will be presented the definition of the main concepts used in the development of the project.

3.1 Logistic regression

In machine learning, there are two main categories: supervised learning and unsupervised learning. In supervised learning, labeled data is used to train an algorithm to provide classification or regression of the data, where it is possible to build a model generalist enough that, given an input that this model does not know, it is possible to provide a mapping of this new data to the corresponding output. In unsupervised learning, unlike the previous one, the data does not have labels that the model can use to learn, so the model must seek the possible relationships between them by exploring the data. "This is a much less well-defined problem since we are not told what patterns to look for, and there is no obvious error metric to use" [14].

A well-known model in classification-supervised machine learning is logistic regression. "Logistic regression is one of the most important statistical and data mining techniques employed by statisticians and researchers for the analysis and classification of binary and proportional response datasets" [15]. In binary logistic regression, the objective is to map from the features of the labeled dataset to the binary targets to predict the probability that a new sample belongs to one of the label classes. The function responsible for doing this mapping is the sigmoid function.

3.2 Multi-Task Learning (MTL)

Multi-task learning is a paradigm in machine learning where one seeks to train tasks simultaneously to improve performance. This is achieved through inductive transfer, improving the generalization of models using data from related tasks, contained in the training domain [16]. In other words given m learning tasks in a dataset $\{T_1, \dots, T_m\}$ that are related to each other, or that a subset of them are related, multi-task learning helps in learning a model for T_i-th using the knowledge contained in all, or some of other tasks [9].

Contrary to what was shown in the single task, where it would be necessary to learn each task individually, with the model in question, it is possible to use sets with different data, which maintain some relationship with each other, aiming at a more effective generalization. In the same domain, the concept of transfer learning has been gaining prominence since it allows agglutinating a set of related tasks to improve the model of a target task. This approach is a machine learning paradigm in which its objective is to improve the learning of a target task through the transfer of knowledge from source tasks, where t is the target task, and T is a set of tasks, there is a subset $\tau \subset \{1, \dots, T\}$ such that all tasks in τ have some similarities with task t. The tasks of τ can have different levels of relationship with task t and help in the training of the task.

In transfer learning, we start with one or more models already pre-trained for source tasks, and we also have new tasks called destination. The main objective is to benefit from previously trained tasks, using them as a basis for target tasks. In addition to speeding up model training, it improves generalization performance with related tasks.

3.3 VC dimension & U-curve

An important step in classification models is model selection, as different classifications can describe the same dataset. In general, candidate models are specified by the user to select the best one [17]. Choosing the best one can be answered by exploring some properties of classifiers such as the VC dimension.

"The theory of learning based on the VC-dimension predicts that the behavior of the difference between training error and test error as a function of the training set size is characterized by a single quantity-the VC-dimension-which characterizes the machine's capacity" [18]. For a fixed-size training sample, as the VC dimension of the model grows, the in-sample error decreases, but the generalization error increases, and this event is known as a U-curve. A U-curve happens when we make an inference in the training process and when we increase that inference, the generalization power is improved by the learning machine up to a tipping point beyond which the generalization power degrades, so the error increases again, forming a U-shaped curve, we can use this feature to explore the minimum cost point for regularization [19, 12].

3.4 Branch & bound

"Branch and bound algorithms are methods for global optimization in nonconvex problems" [20]. This algorithm systematically enumerates possible solutions to the problem in a search space, such as a tree. The algorithm explores these branches, which are made up of solution subsets. It builds new branches if the verified branch respects the restrictions imposed by the limits estimated in the optimal solution. If it does not, it is rejected as a possible solution, and the algorithm can explore another branch, reaching the end in the optimal solution. If the algorithm fails to eliminate the options with the bounds, the algorithm will resume an exhaustive search.

In the literature, some algorithms have already been proposed that use the selection of attributes based on Branch-and-Bound (BB) [5, 11], where in each iteration of the algorithm, the tree generated by the combination of possibilities is traversed (Branch). The cost obtained by the function is registered (Bound). A combinatorial optimization uses some characteristic properties of the cost function, such as monotonicity, to enumerate the sets of characteristics in tree form.

4 Proposed algorithm

The branch and bound combinatorial optimization algorithm was developed to find the best group of source tasks for a target task, assuming that the tasks have some kind of relationship in the dataset. Thus, this algorithm consists of a depth-first search with backtracking in the tree built with the possibilities of source task groupings. The algorithm can be divided into three steps: construction of the tree by combinatorial enumeration of groups of tasks; traversal of the tree with the training of the new configuration; search for the smallest loss.

Combinatorial construction of the tree. In the first step, we started by processing the target task, dividing the data randomly into training and testing (50 % of the data for each), and normalizing both sets with the help of the sklearn library[1]. Soon after, the best configuration of hyperparameters for the task is obtained with the help of the optuna[2] library. Finally, the root node of the tree is obtained by computing the test loss of the data using a regularized logistic regression binary classification method. After obtaining the root node, an enumeration of the available task combinations is performed in groups of two, the first being the target task and the second a candidate to belong to the group of origin tasks. These groups pass (similar to what was done to obtain the root node) through the processes already described for processing the data and obtaining the loss for the test data set. The pseudocode of the tree construction algorithm can be seen in Algorithm 1.

Traversal of the tree. It is then that when the second step begins, the loss obtained by grouping two tasks will only be stored if it is smaller than the loss obtained at the root. If it is larger, the node will not be explored anymore. However, if it is smaller, the algorithm will repeat the same process for the combinations of three in three tasks with the difference that now the lowest loss reference will be the previous configuration (in this case, groups with two tasks). This process will repeat until groups with the depth - 1 tasks (passed as an execution argument).

[1]Available at `scikit-learn.org/stable`.

[2]Available at optuna.org.

Algorithm 1 Algorithm based on combinatorial U-curve optimization

```
Require: lossBase, appendData, tunnedHyperparams, groupTasks
Ensure: bestGroup
 1: function OptimizationUcurve($depth$)
 2:     global variables: $bestGroup$
 3:     local variables: $x\_target, y\_target$
 4:     $loss\_target \leftarrow lossBase(x\_target, y\_target)$
 5:     $dictionary\_options \leftarrow appendData(lossBase(x\_target, y\_target), dictionary\_options)$
 6:     $bestGroup \leftarrow appendData(dictionary\_options, bestGroup)$
 7:     $hyperparams \leftarrow tunnedHyperparams(x\_target, y\_target)$
 8:     $train\_target, test\_target \leftarrow splitData(x\_target, y\_target)$
 9:     while depth > 0 and len(dictionary_options) > 0 do
10:         $dictionary\_options \leftarrow groupTasks(dictionary\_options, train\_target, test\_target,$
    $hyperparams)$
11:         $depth \leftarrow depth - 1$
12:         for $i\_loss, group\_task$ in $dictionary\_options$ do
13:             if $i\_loss < previous\_group\_loss$ then
14:                 loss_target = i_loss
15:                 $bestGroup \leftarrow (loss\_target, group\_origin)$
16:             end if
17:         end for
18:     end while
19:     return bestGroup
20: end function
```

Algorithm 2 Algorithm for group tasks (used by Algorithm 1)

```
Require: trainModel, predictModel, appendData
Ensure: bestGroup
 1: function groupTasks($dictionary\_options, train\_target, test\_target, hyperparams$)
 2:     global variables: $bestGroup$
 3:     local variables: $x\_target, y\_target$
 4:     for $i\_loss, group\_task$ in $dictionary\_options$ do
 5:         for $tasks = 1, 2, \ldots, N$ do
 6:             if $tasks$ not in $group\_task$ then
 7:                 $train\_target \leftarrow appendData(tasks\_train, train\_target)$
 8:                 $model \leftarrow trainModel(train\_target, hyperparams)$
 9:                 $dictionary\_options \leftarrow predictModel(model, test\_target)$
10:             end if
11:         end for
12:     end for
13:     return dictionary_options
14: end function
```

Search for the smallest loss. After having all the best losses of the analyzed groups, a comparison is made, and the configuration chosen will be the one that obtained the lowest loss.

For demonstration purposes, we will analyze a fictitious example of using the algorithm with depth 3 for a base with four tasks where the task target is t_0. Suppose we are interested in building an automated system to determine if a patient may or may not have the disease COVID-19. In our dataset, we have information about the symptoms of 4 patients (t_0, t_1, t_2, and t_3), of whom three patients have the disease (t_0, t_1, and t_2), and the other one has a common cold. In the process of building this base for training the model, we initially need to label the patients as with the disease (1) and without the disease (0). In addition, for the training procedure, the symptoms would be reduced in a numerical feature space, and patient features could be described by an X in a $\mathbb{R}_{m \times n}$ domain. In the execution of the algorithm proposed in this project, a tree for task target of the enumeration of tasks present in the dataset would be used as a base. This tree can be seen in Figure 1.

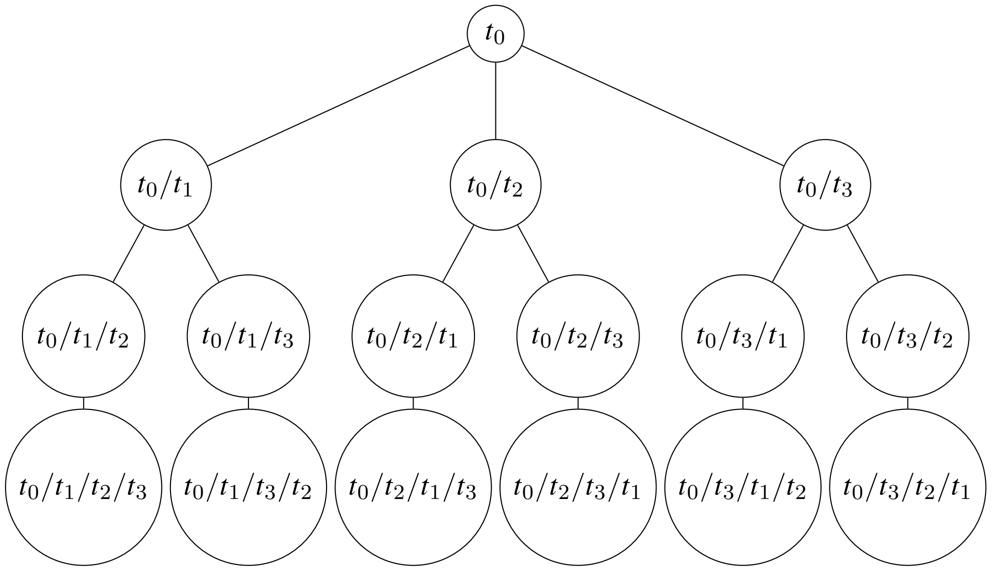


Figure 1: Tree for a simulation of the proposed algorithm with t_0 as the target task.

In addition to the built tree, we can elaborate a Boolean lattice that indicates the presence or not of the task in the target task's origin group with three positions (t_1, t_2, and t_3). If a task is present, its value in the node is 1. Otherwise, its value is zero. At node 000, only target task t_0 is present in the training set. When the value is 1, the task can be weighted by a measure i that can receive the values [0.5, 0.6, 0.7, 0.8, and 0.9]. In the algorithm, measure i is used to weigh the importance of the task added in training the target task. This value is multiplied by the task feature present in the origin task group, and only the weight corresponding to the smallest loss is recorded. The Boolean lattice is presented in Figure 2.

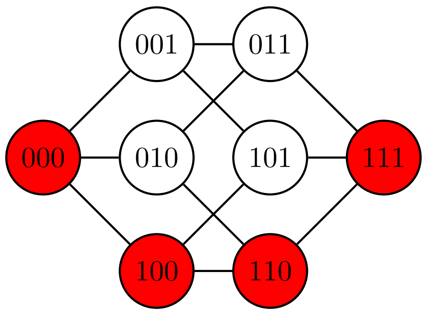


Figure 2: Boolean lattice for presence the tasks in group tasks origin. A chain of the lattice is highlighted in red.

To build the t_0 model, we initially divided the dataset into 50% for training and 50% for testing, followed by feature normalization. In addition, we chose to build the model using the classification method known as binary logistic regression, implemented with the `sklearn` library. For the best performance of the model, we then performed a hyperparameter optimization with the help of the `optuna` library. Finally we compute the loss of the test dataset of the task compared to the predicted values using the test features (in our case the loss would result in 0.5). The loss of the model t_0 decreases in the inclusion of related tasks (in this case t_1 and t_2) and with each addition of a non-related task the resulting loss increases, thus yielding graphs with U-shaped curves (Figure 3).

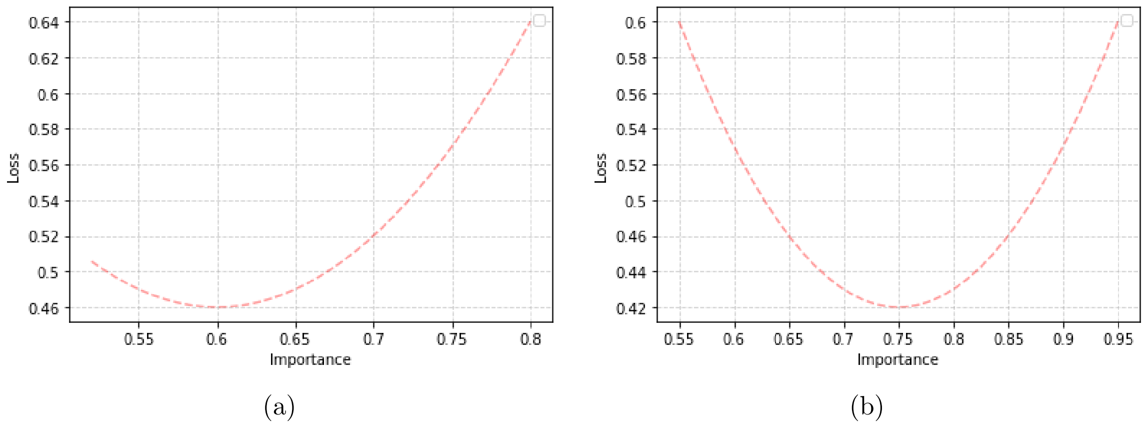


Figure 3: Fictitious examples of the behavior of loss functions. (a) Loss curve by the importance of a chain that was **not** highlighted in Figure 2; (b) Loss curve by the importance of a chain that was highlighted in Figure 2.

However, it is possible to notice an inflection point in the curve of the importance of the added tasks that are related (origin tasks) by the loss. As this importance increases, or as the dataset of the origin tasks grows, the loss increases again, this behavior can be described by a U-curve. In the case of adding unrelated tasks, initially (at low importance), there may be a decrease in loss compared to the single task model (only with t_0 training), but the loss returns to rise quickly. In Figure 3, the loss curve is represented by the importance of the chains in the lattice described in Figure 2. We can notice that when adding related tasks, due to the behavior at U, we extracted a global minimum, considering that the curve's minimum with unrelated tasks is greater than the minimum only with additions of related tasks. During the execution of the algorithm, the search space is reduced due to the bounds. If the current node has a loss greater than the loss of the previous node, new additions to the current node will not be explored. In the case of the tree described in Figure 1, the search space could be reduced to the tree described in Figure 4. We can see that although most of the tasks are related, we managed to reduce the search space by about 40%.

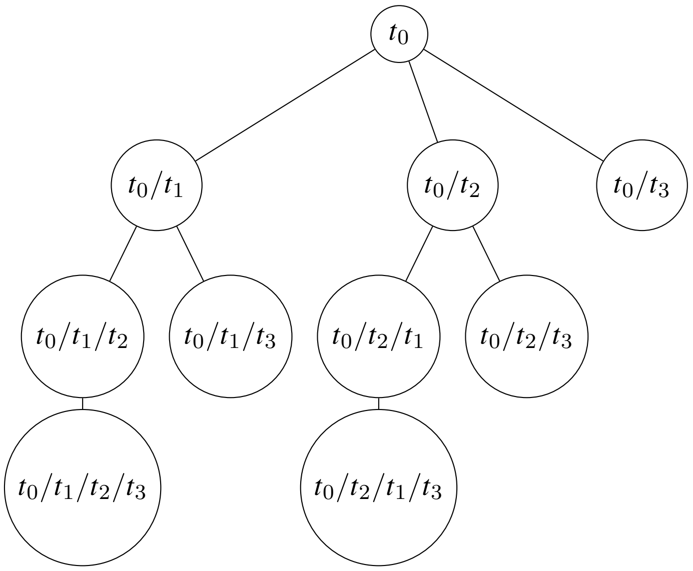


Figure 4: Pruned tree for t_0 as the target task. Compare it with the complete tree depicted in Figure 1.

The losses generated by the prediction of each model contained in each node of the tree showed in Figure 4 be registered by the algorithm and the result will return the smallest loss and the group that generated it.

5 Experimental methodology

This section describes the material and methods used in the computational experiments carried out to evaluate the novel algorithm proposed in the previous section. This includes the used datasets, computational setup, parameters of the experiments and the used benchmarking algorithm.

5.1 Datasets

An important step towards validating the proposal was the choice of multitasking learning bases. Aiming at greater control of the bases and task groupings, we built datasets that we call synthetic datasets. In addition, we chose three datasets with real multitasking data. Both cases will be described in the following.

5.1.1 Synthetic datasets

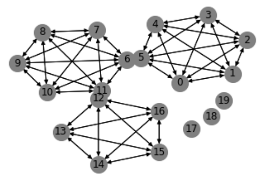
To perform the algorithm tests, a piece of prior knowledge about the sharing of structures in the data provides a better understanding of the possible groupings between tasks. Therefore, we chose to create artificial datasets of different dimensions following a similar procedure in the literature [21, 22, 23]. We designed three sets of synthetic data, all of them with 399 points:

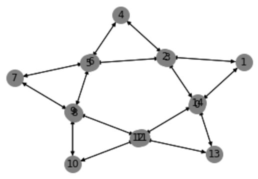
a) 20 tasks, 21 features, 3 outlier points;

b) 15 tasks, 16 features, no outlier points;

c) 12 tasks, 16 features, no outlier points.

The synthetic task grouping of for each of the three synthetic datasets is depicted in Figure 5. In every case, all sample (i) for all task (t) is generated by a multivariate normal distribution $x(t)$, where the multiple random variables can be correlated with each other or not since each basis is built by generating destination parameters such that $\Theta \equiv \{\theta_1, \theta_2, ..., \theta_t\}$. For the created datasets $\theta(t)_i \sim \mathcal{N}(0_d, I_{d\times d})$. The outliers in the base with 20 tasks are created with other struct inspired in [21, 22] that can be described for: $\theta(t)_i \sim \mathcal{N}(0_d, 25 * I_{d\times d})$. The binary output is calculated on the probability yielded by the logistic regression, and we considered the output $y(t)_i = 1$ if $p(t)_i > 0.5$ and $y(t)_i = 0$ if $p(t)_i \leq 0.5$.


(a) Dataset of 20 tasks with 3 outliers.


(b) Dataset of 15 tasks.

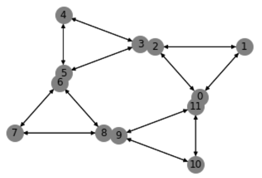

(c) Dataset of 12 tasks.

Figure 5: Clusters created on the synthetic datasets (a)–(c).

5.1.2 Real datasets

In addition to the synthetic data explored, to reach a greater number of cases, we analyzed the behavior of the algorithm with some sets of real data. The actual datasets explored were: landmine detection, ECML/PKDD3 spam detection with 3 and 15 users.

The dataset about landmine detection has 29 tasks that contain between 444 to 689 data samples (depending on the task) with 10 features. With three users, the test set for spam has 3 tasks containing between 2499 data samples (with 203 features). Finally, the test set for spam, with 15 users, has 15 tasks that contain between 399 data samples (with 477 features) each task.

5.2 Computational setup

With the aid of the infrastructure provided by the Laboratory of Artificial Intelligence and Inference in Complex Data (Recod.ai) at University of Campinas, the experiments were performed on an Intel(R) Xeon(R) E5-2620 0 @ 2.00GHz CPU with 24GB of RAM. The defined tree depth limit was four. Thus, tests with groups larger than the depth initially defined as a parameter of the algorithm were unable to complete their execution and the returned solution might not be a global optimum.

5.3 Parameters of the experiment

Initially, through the command prompt, the algorithm receives both the path where the dataset is located and also the desired depth. Furthermore, it is important to note that since the technique chosen for creating the models was logistic regression, we also needed to perform hyperparameter optimization during development. For this purpose, we used the `optuna` library as support.

5.4 Benchmarking algorithm

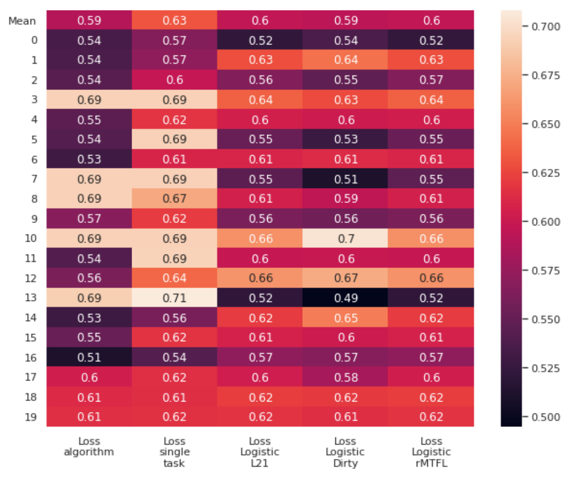
To make a comparison of the algorithm proposal presented in this article with the different methodologies found in the literature, we designed a comparison with three multi-tasking learning methods: Joint Feature Selection (Logistic L21) [24], Dirty Model (Logistic Dirty) [25] and Robust Learning of Multitasking Features (Logistic rMTFL) [23]. In our comparison, hyper-parameters was tuned using hyperopt with the same parameters of our algorithm.

6 Results and discussion

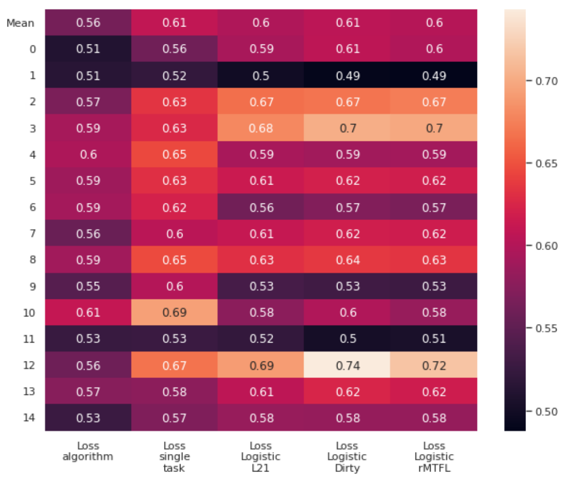
The results were obtained by comparing the 5 bases (3 real and 2 synthetic datasets) with the 3 multi-task methods already tried in the literature. In the synthetic datasets, we have full control over the generative parameters, providing knowledge of the relationship between the tasks (which can help in training the target task). Still, in real datasets, the similarity between tasks is not known. The comparison is made through the prediction loss of the group selected by the algorithm with the prediction loss in the literature algorithms. We will divide the results into two parts, the first being the application of the algorithm on the synthetic bases and the second the application of the algorithm on the real bases.

6.1 Results for synthetic datasets

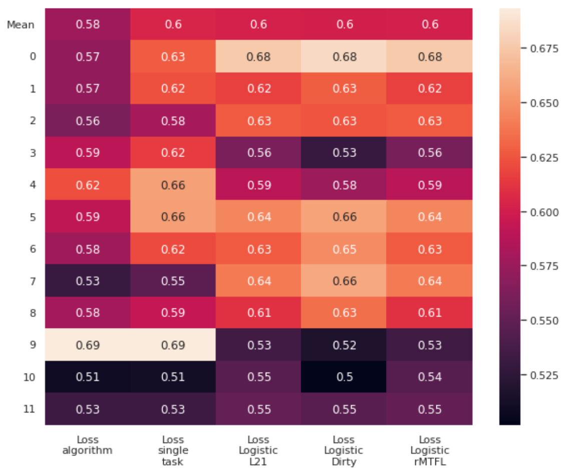
The summary of the results for the synthetic dataset are portraited in Figure 6, the proposed algorithm is the first column for all of the three figures for three different datasets. The result with 20 tasks and three outliers are present in Figure 6a, for the dataset with 15 tasks is in Figure 6b, for the dataset with 12 tasks are in Figure 6c. We can see in all datasets the proposed method presents a good mean performance (shown in the first row of each Figure) with some tasks with poor performance: tasks 3, 7, 8, 10 and 13 in Figure 6a, and task 9 in Figure 6c. These performances are going to be investigated in further detail below.



(a) Dataset with 20 tasks (being 3 outliers).



(b) Dataset with 15 tasks (without outliers).



(c) Dataset with 12 tasks (without outliers).

Figure 6: Results obtained with the synthetic datasets described in the previous section. In each graph, rows represent tasks and columns different algorithms.

Since the sharing of information in these datasets was known and controlled, we also present tables showing the relationships among the tasks and how well the algorithm performed when reconstructing the task relations. Table 1 represents the dataset with 20 tasks and three outliers; Table 2 represents the dataset with 15 tasks; and Table 3 represents the dataset with 12 tasks.

Table 1: Comparison of real clusters and the clusters found by the proposed algorithm in the base of 20 tasks and three outliers. The first column shows the reference task, the second and third columns show the found relations and the reference relations respectively; and the fourth and fifth columns show the rate of correct and incorrect relations found by the algorithm respectively.

Task	Group by the algorithm	Real relationship	Correct tasks	Incorrect tasks
t_0	$t_0/t_4/t_1$	$t_0/t_1/t_2/t_3/t_4/t_5$	3/5	0
t_1	$t_1/t_4/t_2/t_5$	$t_0/t_1/t_2/t_3/t_4/t_5$	4/5	0
t_2	$t_2/t_4/t_5$	$t_0/t_1/t_2/t_3/t_4/t_5$	3/5	0
t_3	$t_3/t_0/t_4/t_5$	$t_0/t_1/t_2/t_3/t_4/t_5$	4/5	0
t_4	$t_4/t_0/t_1$	$t_0/t_1/t_2/t_3/t_4/t_5$	3/5	0
t_5	$t_5/t_4/t_0$	$t_0/t_1/t_2/t_3/t_4/t_5$	3/5	0
t_6	$t_6/t_9/t_{11}$	$t_6/t_7/t_8/t_9/t_{10}/t_{11}$	3/5	0
t_7	$t_7/t_9/t_6/t_{10}$	$t_6/t_7/t_8/t_9/t_{10}/t_{11}$	4/5	0
t_8	$t_8/t_6/t_9/t_7$	$t_6/t_7/t_8/t_9/t_{10}/t_{11}$	4/5	0
t_9	$t_9/t_7/t_6/t_{10}$	$t_6/t_7/t_8/t_9/t_{10}/t_{11}$	4/5	0
t_{10}	$t_{10}/t_9/t_6/t_{11}$	$t_6/t_7/t_8/t_9/t_{10}/t_{11}$	4/5	0
t_{11}	$t_{11}/t_7/t_9/t_6$	$t_6/t_7/t_8/t_9/t_{10}/t_{11}$	4/5	0
t_{12}	$t_{12}/t_{13}/t_{16}$	$t_{12}/t_{13}/t_{14}/t_{15}/t_{16}$	3/5	0
t_{13}	$t_{13}/t_{12}/t_9/t_{18}$	$t_{12}/t_{13}/t_{14}/t_{15}/t_{16}$	4/5	2/15
t_{14}	t_{14}/t_{12}	$t_{12}/t_{13}/t_{14}/t_{15}/t_{16}$	2/5	0
t_{15}	$t_{15}/t_{16}/t_{13}/t_{12}$	$t_{12}/t_{13}/t_{14}/t_{15}/t_{16}$	4/5	0
t_{16}	t_{16}/t_{12}	$t_{12}/t_{13}/t_{14}/t_{15}/t_{16}$	2/5	0
t_{17}	t_{17}/t_7	t_{17}	1	1/19
t_{18}	t_{18}/t_{17}	t_{18}	1	1/19
t_{19}	t_{19}	t_{19}	1	0

In Tables 1–3, it is possible to see that in most groups the tasks added in the target task were created with information sharing as described in the section on the synthetic bases. Therefore, the algorithm was able to recognize the similarities between the tasks in an efficient manner. However, in Table 1, we see that the presence of outliers makes the pattern recognition process difficult, as they were added to several unrelated tasks.

In Table 1, the algorithm managed to make 41 correct additions and four incorrect additions. Considering that as the depth was capped at 4, the maximum additions he could make would be three tasks for each task target. As on this base, we had groups of 5 tasks, the algorithm was limited to producing a maximum of 80% of the group (4 tasks out of 5) correctly, considering that it does not manage to test the 4th addition. At the output, a total of 17 groups of 20 were found correctly, that is, only with tasks where we had prior knowledge that, due to the formation of the base (already described earlier), they were related.

In Table 2, the algorithm managed to make 24 correct additions and 8 incorrect additions. Considering that as the depth was capped at 4, the maximum additions he could make would be three tasks for each task target. As in this case, we had groups of 3 tasks, the algorithm could produce the groups in a complete way. At the output, a total of 8 groups of 15 were found correctly, that is, only with tasks where we had prior knowledge that, due to the formation of the base (already described earlier), they were related. This result may show that even unrelated tasks can help in training. Still, the maximum number of unrelated tasks that the algorithm added to the base was 2 in the group with task target t_7. That is, of the 12 tasks that did not belong to a group o, a maximum number of additions shows that, despite helping in training, it is still less effective than adding related tasks, considering that in larger quantities, the performance was worse. In summary, unrelated tasks can help in training if they are a minority in the group. When increasing the number of unrelated tasks within the same group, their performance worsens.

In Table 3, the algorithm managed to make 15 correct additions and four incorrect additions. Similar to Table 2, as the depth was limited to 4, the maximum additions it could make would be 3 tasks for each task target, and since, in this case, we had groups of 3 tasks, the algorithm could produce the groups completely. At the output, a total of 8 groups of 12 were correctly found, that is, only with tasks where we had prior knowledge that, due to the base formation (already described earlier), they were related.

Table 2: Clusters found in the dataset with 15 tasks.

Task	Group by the algorithm	Real relationship	Correct tasks	Incorrect tasks
t_0	$t_0/t_1/t_2$	$t_0/t_1/t_2$	3/3	0
t_1	t_1/t_0	$t_0/t_1/t_2$	2/3	0
t_2	$t_2/t_0/t_1/t_4$	$t_0/t_1/t_2$	3/3	1/12
t_3	$t_3/t_5/t_4/t_14$	$t_3/t_4/t_5$	3/3	1/12
t_4	$t_4/t_5/t_3$	$t_3/t_4/t_5$	3/3	0
t_5	$t_5/t_4/t_3$	$t_3/t_4/t_5$	3/3	0
t_6	$t_6/t_7/t_8$	$t_6/t_7/t_8$	3/3	0
t_7	$t_7/t_9/t_6/t_10$	$t_6/t_7/t_8$	2/3	2/12
t_8	$t_8/t_6/t_2$	$t_6/t_7/t_8$	2/3	1/12
t_9	$t_9/t_{10}/t_{11}$	$t_9/t_{10}/t_{11}$	3/3	0
t_{10}	$t_{10}/t_{11}/t_9/t_8$	$t_9/t_{10}/t_{11}$	3/3	1/12
t_{11}	t_{11}/t_{10}	$t_9/t_{10}/t_{11}$	2/3	0
t_{12}	$t_{12}/t_{13}/t_{14}/t_6$	$t_{12}/t_{13}/t_{14}$	3/3	1/12
t_{13}	t_{13}/t_4	$t_{12}/t_{13}/t_{14}$	1/3	1/12
t_{14}	$t_{14}/t_{13}/t_{12}$	$t_{12}/t_{13}/t_{14}$	3/3	0

Table 3: Clusters found in the datasets with 12 tasks.

Task	Group by the algorithm	Real relationship	Correct tasks	Incorrect tasks
t_0	t_0/t_7	$t_0/t_1/t_2$	1/3	1/9
t_1	$t_1/t_0/t_2$	$t_0/t_1/t_2$	3/3	0
t_2	$t_2/t_1/t_0$	$t_0/t_1/t_2$	3/3	0
t_3	t_3/t_5	$t_3/t_4/t_5$	2/3	0
t_4	$t_4/t_3/t_5$	$t_3/t_4/t_5$	2/3	0
t_5	$t_5/t_4/t_3/t_9$	$t_3/t_4/t_5$	3/3	1/9
t_6	$t_6/t_7/t_8/t_2$	$t_6/t_7/t_8$	3/3	1/9
t_7	t_7/t_8	$t_6/t_7/t_8$	2/3	0
t_8	$t_8/t_7/t_6$	$t_6/t_7/t_8$	3/3	0
t_9	$t_9/t_11/t_10/t_8$	$t_9/t_10/t_11$	3/3	1/9
t_10	t_10/t_11	$t_9/t_10/t_11$	2/3	0
t_11	t_11	$t_9/t_10/t_11$	1/3	0

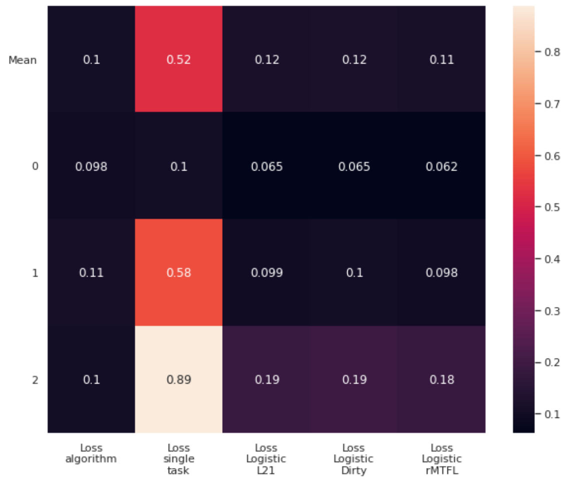
The results presented both in Figures 6 and 7 and in Tables 1–3 show that even with the limited depth, there was an improvement in the training performance of most of the tasks in the bases, both in relation to the single task and in relation to the comparison between the other methods. For instance, in the dataset with 20 tasks (being three outliers), 11 tasks had a drop or maintained the same loss in relation to the other 4 tests (55%). In the data set with 15 tasks (without outliers) 9 tasks had a drop or maintained the same loss in relation to the other 4 tests (60%). And finally, in the data set with 12 tasks (without outliers) 8 tasks dropped or maintained the same loss in relation to the other 4 tests (about 66.7%).

6.2 Results for real datasets

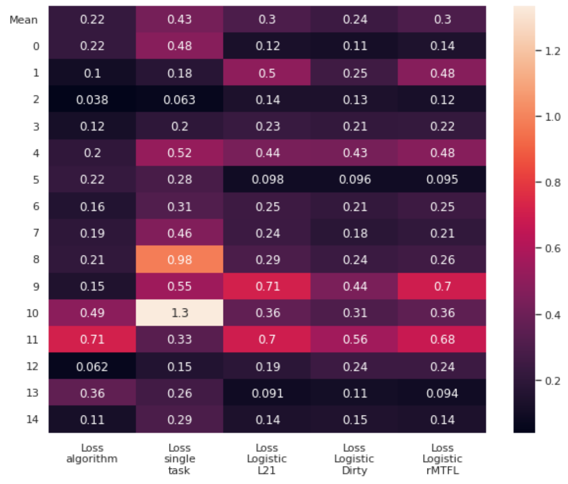
The developed algorithm returns the best cluster found of all clusters tested for each task in the dataset at the initially stipulated depth. To facilitate the comparison, a heatmap colored table was developed with the loss of the predict of target task with the loss returned from the literature methods that have already been described and also the comparison with the loss in the single task model. In addition, the average of the results per task was included. The results obtained for the Spam dataset with three tasks, the Spam dataset with 15 tasks and the Landmine dataset are presented, respectively, in Figure 7a, Figure 7b and Figure 7c.

We can see that all datasets in the proposed method presents a good mean performance (shown in the first row of each figure) with some tasks with poor performance: tasks 10, 11 e 13 in Figure 7b, 16, 20, 23, 25 and 28 in Figure 7c. Considering that as the maximum depth was limited to 4, the algorithm could include at most three tasks for each task target. In these cases, we had no prior knowledge about the possible groupings of the target tasks, but as in Section 6.1, if the possible groups have a size greater than 4, then the presented strategy would have a limited performance.

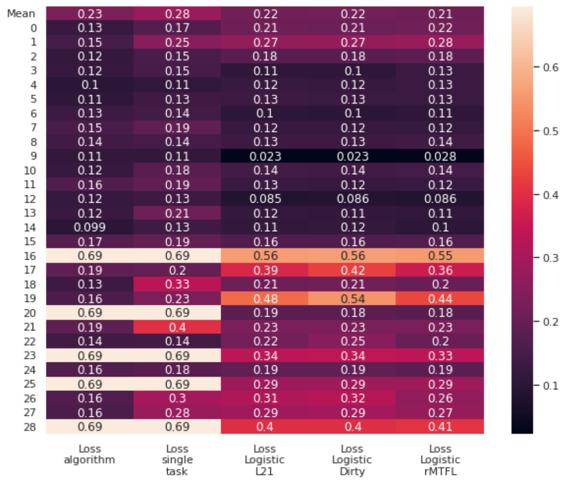
In the dataset of spam with 3 users 1 task had a drop or maintained the same loss in relation to the other 4 tests (33%). In the dataset of spam with 15 users 9 tasks had a drop or maintained the same loss in relation to the other 4 tests (60%). And finally, in the dataset of landmine 15 tasks dropped or maintained the same loss in relation to the other 4 tests (about 51.7%). In addition, we can see that the presence of few tasks makes it difficult to recognize similarities, seen in dataset of spam with 3 users, because despite having obtained a good average performance and a significant gain in relation to the single task, compared to the other methods, it obtained higher losses.



(a) Spam dataset (3 users).



(b) Spam dataset (15 users).



(c) Landmine dataset.

Figure 7: Results obtained with the real datasets described in the previous section. In each graph, rows represent tasks and columns different algorithms.

7 Conclusion

In this project, a knowledge transfer model based on combinatorial optimization in the Boolean lattice with U-curves was studied for an improvement in the training of a target task evaluating in six bases, three synthetic bases and three real bases. Each task in each studied database had its performance evaluated (through loss) not only in the developed algorithm but also using different methodologies for comparisons, such as single task, Joint Feature Selection, Dirty Model, and Robust Learning of Multi-task Features. It was observed that the algorithm had a satisfactory performance in relation to the tested methods, and provided gains (in most of the tasks present in the tested datasets) to the models that, regardless of the selected machine learning method, without prior knowledge of the data distribution, are capable of generating responses with a generalization level.

Finally, because the enumeration feature of the search space can become a sequential search with an exponential cost if the tree prunings are not effective, the depth limitation imposed in the algorithm development may not return the global minimum of the loss. However, even so, when compared to the individual models (single task) they still had similar capabilities. As directions for future work, we could explore the behavior of the method in different classifier methodologies, or even regressors. Additionally, we could explore the impact of input dimensionality on output.

References

[1] Zhiqiang Zheng and B. Padmanabhan. On active learning for data acquisition. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 562–569, 2002.

[2] Dana Maslovat, Romeo Chua, Timothy D Lee, and Ian M Franks. Contextual interference: single task versus multi-task learning. *Motor control*, 8(2):213–233, 2004.

[3] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[4] Steffen Bickel, Jasmina Bogojeska, Thomas Lengauer, and Tobias Scheffer. Multi-task learning for hiv therapy screening. In *Proceedings of the 25th international conference on Machine learning*, pages 56–63, 2008.

[5] Marcelo S. Reis, Gustavo Estrela, Carlos E. Ferreira, and Junior Barrera. featsel: A framework for benchmarking of feature selection algorithms and cost functions. *SoftwareX*, 6:193–197, 2017.

[6] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19, 2006.

[7] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine learning*, 73(3):243–272, 2008.

[8] Guillaume Obozinski, Ben Taskar, and Michael I Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2):231–252, 2010.

[9] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *ICML*, pages 521–528, 2011.

[10] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

[11] Marcelo S. Reis, Gustavo Estrela, Carlos E. Ferreira, and Junior Barrera. Optimal boolean lattice-based algorithms for the U-curve optimization problem. *Information Sciences*, 471:97–114, 2019.

[12] Gustavo Estrela, Marco D. Gubitoso, Carlos E. Ferreira, Junior Barrera, and Marcelo S. Reis. An efficient, parallelized algorithm for optimal conditional entropy-based feature selection. *Entropy*, 22(4), 2020.

[13] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Malsar: Multi-task learning via structural regularization. *Arizona State University*, 21:1–50, 2011.

[14] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[15] Maher Maalouf. Logistic regression in data analysis: an overview. *International Journal of Data Analysis Techniques and Strategies*, 3(3):281–299, 2011.

[16] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[17] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, 2012.

[18] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994.

[19] Diego Marcondes, Adilson Simonis, and Junior Barrera. Learning the hypotheses space from data through a U-curve algorithm: a statistically consistent complexity regularizer for model selection, 2021. eprint:2109.03866.

[20] Stephen Boyd and Jacob Mattingley. Branch and bound methods. *Notes for EE364b, Stanford University*, pages 2006–07, 2007.

[21] Marcos M Raimundo and Fernando J Von Zuben. Investigating multiobjective methods in multitask classification. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2018.

[22] Wenliang Zhong and James Kwok. Convex multitask learning with flexible task clusters. *arXiv preprint arXiv:1206.4601*, 2012.

[23] Pinghua Gong, Jieping Ye, and Changshui Zhang. Robust multi-task feature learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 895–903, 2012.

[24] Guillaume R Obozinski, Martin J Wainwright, and Michael Jordan. High-dimensional support union recovery in multivariate regression. *Advances in Neural Information Processing Systems*, 21, 2008.

[25] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep Ravikumar. A dirty model for multi-task learning. *Advances in neural information processing systems*, 23, 2010.