Can metalearning be applied to transfer on heterogeneous datasets?

Catarina Félix[1,2], Carlos Soares[1,3], and Alípio Jorge[2,4]

[1] INESC TEC, Portugal
[2] Faculdade de Ciências da Universidade do Porto, Portugal
[3] Faculdade de Engenharia da Universidade do Porto, Portugal
[4] LIAAD-INESC TEC, Portugal
`cfo@inescporto.pt, csoares@fe.up.pt, amjorge@fc.up.pt`

Abstract. Machine learning processes consist in collecting data, obtaining a model and applying it to a given task. Given a new task, the standard approach is to restart the learning process and obtain a new model. However, previous learning experience can be exploited to assist the new learning process. The two most studied approaches for this are metalearning and transfer learning. Metalearning can be used for selecting the predictive model to use on a new dataset. Transfer learning allows the reuse of knowledge from previous tasks. However, when multiple heterogeneous tasks are available as potential sources for transfer, the question is which one to use. One approach to address this problem is metalearning. In this paper we investigate the feasibility of this approach. We propose a method to transfer weights from a source trained neural network to initialize a network that models a potentially very different target dataset. Our experiments with 14 datasets indicate that this method enables faster convergence without significant difference in accuracy provided that the source task is adequately chosen. This means that there is potential for applying metalearning to support transfer between heterogeneous datasets.

1 Introduction

Machine learning processes consist of 1) collecting training data for the new task; 2) obtaining a model; 3) applying the model to new data. This is done even when the new task is related to previously solved tasks, for example, when there are relationships between variables or between the processes used to obtain the models.

There are two approaches to using previous learning experience in new tasks: metalearning and transfer learning. Both use information about a domain to learn efficiently and effectively in a new one. Metalearning typically focuses on the choice of a learning algorithm while transfer learning tries to reuse the models obtained from previous tasks. This suggests that transfer learning and metalearning may be used together.

Our ultimate aim is to investigate if metalearning can be used to support transfer learning in tasks consisting of heterogeneous data, reducing computational cost without loss in predictive performance and, eventually, cutting down the time data scientists need to invest in the process.

The method for using metalearning to support transfer learning is illustrated id Figure 1 and has three steps:

1. **Source selection step:** finding the best source problem for approaching our new target problem;
2. **Transfer step:** adapting the source model and use some of its components (or characteristics) in the target model;
3. **Learning step:** training the target model (adapted from the source model) on the target data.

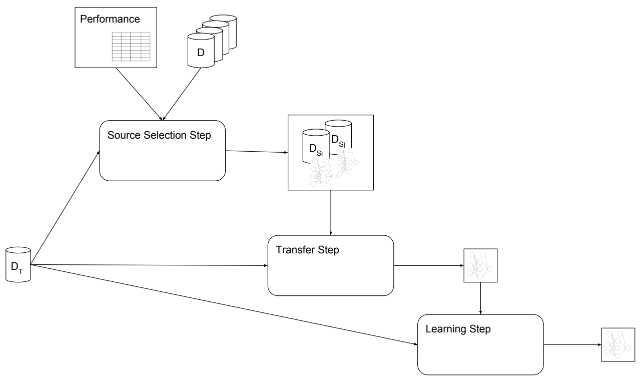

Fig. 1. Method for using metalearning

In this paper we propose a method for the transfer step. We show that if we find a good solution for the meta step, the transfer step can improve the learning speed, even with very simple methods. Future work is to develop a process for the meta step.

2 Metalearning and transfer learning

This section presents the basic concepts related with our work. First we describe metalearning, some of its methods and examples of use. After that, we present transfer learning, its motivation, operation mode and some techniques used. Finally, we describe some examples of the combination of metalearning and transfer learning.

2.1 Metalearning

Metalearning is typically used for algorithm recommendation: helping in the process of selecting a predictive algorithm to use on a new dataset. It also aims at taking advantage of the repetitive use of a given method over a set of similar tasks.

There are several applications of metalearning: combining base learners, namely using several learners together to create a composite model that better predicts the result; bias management, mostly used for data streams that require context adaptation due to the fact that the domain is not static; and transferring metaknowledge across tasks. But metalearning is mostly used for the algorithm recommendation, as described next.

Algorithm Recommendation Choosing the best algorithm for processing a given dataset is a difficult process. Besides, algorithms normally have parameters that affect its efficiency and tuning them can be a difficult and slow task. This constitutes the motivation for the Algorithm Selection Problem [?], originally formulated by Rice [?].

This problem consists in determining the best algorithm to use for a certain dataset. The metalearning approach takes advantage of information previously obtained on several datasets and also on several algorithms. This knowledge is used to build a metamodel that, given a new dataset, predicts which is(are) the most suitable algorithm(s).

Earlier applications of metalearning addressed the most common tasks - classification [?], regression [?] and time series [?]. These approaches were then extended to selecting parameter settings for a single algorithm [?], the whole data mining process [?] and also to problems from domains other than machine learning, e.g.: different optimization problems [?,?]. More recently, they were also used to deal with new problems in data mining: data streams [?].

2.2 Transfer Learning

A definition of transfer learning is: given a source domain D_S and a learning task T_S, a target domain D_T and a learning task T_T, transfer learning aims to improve the process of learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$ [?].

Transfer learning allows algorithms to adapt to new tasks based on the knowledge obtained in previous ones. The three main research issues in this topic are related to *what*, *how* and *when* to transfer.

What to transfer? This question concerns the type of information transferred between problems: instance-transfer, where instances from the source domain are used together with the ones on the target domain, to use more training data for the target model, as in the TrAdaBoost [?] algorithm; feature-representation-transfer, where a set of feature representations is extracted from the source domain and is transferred to become a new feature representation of the target domain [?]; parameter-transfer that consists of transferring some of the parameters of the source to the target model, assuming that the models for related tasks share some parameters [?]; and relational-knowledge-transfer, that consists in trying to transfer the knowledge about data between the domains, as is the case of TAMAR [?], that maps a source Markov Logic Network to the target domain and revises only its incorrect portions.

How to transfer? After knowing what to transfer, the focus is on *how to transfer?*, that is, on the development of learning algorithms to perform the transfer. Approaches to this question can be characterized in two dimensions: the type of algorithm and the type of data used. As for the algorithm the approaches were based, for example, in neural networks [?, ?], naive bayes [?] or decision trees [?, ?]. In what concerns the data some consider propositional data [?, ?, ?, ?], while others use more complex types, such as graphs [?].

When to transfer? The last question is concerned with the situations in which the transfer should be performed. Ultimately, the objective is to avoid *negative transfer*, i.e. when, instead of improving the performance, the transfer can even harm the learning process in the target task. Some approaches have been proposed to identify when transfer learning will hurt the performance of the algorithm instead of improving it (e.g. [?]).

2.3 Metalearning and Transfer Learning

Some work has been performed in using metalearning together with transfer learning.

Metafeatures are used in [?] for calculating similarities between the datasets. The algorithms used for this task is the *k-nearest neighbors*. The best configurations for the selected source domain are then transferred for the target domain.

In [?], metalearning is used to find matrix transformations capable of producing good kernel matrices for the source tasks. These transformations can be used to compute the kernel matrix for new related tasks. Another example is [?], where metalearning is used to construct a new classification algorithm that will be applied to new problems.

The results are evaluated by performance measures as accuracy [?] and more precisely by the area under the ROC curve in [?, ?].

The transferred objects found on the studied papers are SVM parameter settings in [?], the kernel matrices in [?] and the *parameter function* (responsible for mapping statistics to parameters in “bag-of-words” text classification problems) in [?].

3 Weight transfer in neural networks

In this section we propose a method, with four variants, for transferring weights from a source neural network, specifically a Multilayer Perceptron, to a target one. As explained above, the aim is to reduce the computational time needed for the target network to converge.

We start by revisiting the concept of neural network and then describe the weight transfer method and also the mapping method used for the weight transfer.

3.1 Neural Networks

Neural networks have been used for problems in diverse areas, such as: pattern recognition, prediction, optimization, associative memory, and control [**?**].

The neural networks considered in this work are composed by three layers of neurons: input, hidden and output. Figure 2 shows an example of a neural network with this structure designed for a dataset with three input and one output variables.

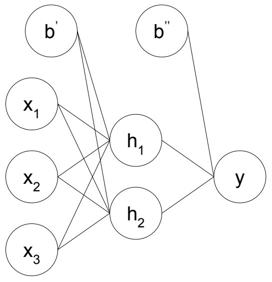

Fig. 2. Example of a neural network

The neurons in the input layer correspond to the independent variables of the dataset (x_1, x_2 and x_3) and there is an extra neuron for the bias (b'). The number of neurons in the hidden layer can vary and it is one of the parameters of the algorithm. In the example the neural network has two hidden neurons: h_1 and h_2. This layer also has an additional bias neuron (b''). The neuron on the output layer (y) corresponds to the dependent variable of the dataset.

There are connections between input and hidden layers and, also, between the latter and the output layer. The connections have associated weights, which are adjusted in each iteration of the network to better predict the output variable. A neural network is initialized with a set of weights and, after converging or reaching the (parametrized) maximum number of iterations, outputs the adjusted set of weights.

3.2 Weight transfer method

In our method, we transfer the weights from a (source) network to another (target) network according to Algorithm 1.

```
   Input: $D_S, D_T, N_S$
   Output: $N_{T \leftarrow S}$
 1 Map $vars(D_S)$ into $vars(D_T)$
 2 foreach $x_j \in vars(D_T)$ do
 3     $w_{x_j,h_p} = w_{M^F_{x_j},h_p}$
 4 end
 5 foreach hidden node $h$ of $N_T$ do
 6     $w_{h_p,y_T} = w_{h_p,y_S}$
 7 end
 8 foreach bias node $b$ of $N_T$ do
 9     $w_{b'_T,h_p} = w_{b'_S,h_p}$
10     $w_{b''_T,y_T} = w_{b''_S,y_S}$
11 end
```
Algorithm 1: Transfer algorithm

The input D_S corresponds to the source dataset that is composed by the independent variables $x_i : i \in \{1, \dots, n\}$ and the dependent variable y_S. D_T is the target dataset and consists of $x_j : j \in \{1, \dots, n\}$ as independent and y_T as dependent variables. N_S is the neural network learned from the source dataset with a randomly generated initial set of weights.

The first step of the algorithm consists in mapping the variables from the source to the ones on the target datasets. The mapping process is explained in 3.3.

After this, for each mapping $M^F_{x_j} = x_i$, we transfer every weight originating in the neuron corresponding to x_i in N_S to the connections with origin in x_j in N_T (line 3 in the algorithm).

The weights of the connections with origin in hidden (h_p) and bias (b' and b'') neurons are directly transferred, since N_S and N_T have the same hidden layer configuration (lines 6, 9 and 10 in the algorithm, respectively).

3.3 Mapping method

The mapping process is performed independently of the transfer step and uses the entire dataset. The mapping method is represented by the generic mapping

$$M^F_{x_j} = x_i$$

that assigns to each target variable the most adequate source variable.

This generic mapping can be instantiated with different specific mapping functions (F):

– Kullback-Leibler (KL) divergence

$$M^{KL}_{x_j} = argmin_i\{KL(x_j, x_i)\}, \forall i \in \{1...n\}, j \in \{1...m\}$$

– Pearson, Spearman and Kendall correlations

$$M^C_{x_j} = argmin_i\{|corr(x_j, y_T) - corr(x_i, y_S)|\}, \forall i \in \{1...n\}, j \in \{1...m\}$$

We also use random mapping and compare the improvement of the resulting neural networks with the ones obtained when using the mapping functions mentioned above.

4 Experiment

Now that we have described the weight transfer method, we will perform an exhaustive experiment using 14 datasets. Our hypothesis is that a neural network algorithm can converge faster on a target dataset if the initial weights are transferred from a source network trained on a well chosen source dataset.

For that purpose, we test the methods proposed here on all source/target combinations (the source is always different from the target) and pick, for each target, the best source.

We assess the results of our methods against the usual random weight initialization method. If the best source/target pairing shortens the convergence time when compared to the baseline, then we have evidence to support our hypothesis.

The experiment was performed using datasets retrieved from UCI [**?**] (Table 1). Most of the datasets are heterogeneous, i.e. generated from very different processes (i.e. they are represented by very different sets of variables). However, some of the problems used have more than one target variable, and, thus, originate more than one dataset.

This means that these datasets are related, as they share the same independent variables, with the same values.

4.1 Experiment Description

We performed the experiment using neural networks with ten neurons in the hidden layer. Our objective is to assess the potential improvement that the right transfer of weights brings to the learning time of a target neural network.

This improvement is calculated by comparing the time needed for the network to converge when using:

– randomly generated initial weights
– weights transferred from another neural network

The experiment is performed for every possible combination of source/target datasets.

It is organized into two steps as illustrated by the boxes in Figure 3: the source learning step and the target learning step.

The source learning step (top of the figure) consists in learning a neural network with standard randomly generated initial weights (W_S) for each dataset (D_S) considered. After the learning process we store the error (MSE_S), duration of the process ($duration_S$) and the best set of weights found (W_S^L).

This is repeated twenty times to outwit the effect of using random values to initialize the network. Thus, we get twenty sets of weights for transfer. Although

Table 1. Datasets used for the experiment

Problem	**Derived Dataset**
Concrete Compressive Strength	concreteCompressiveStrength
Wine Quality - Red	wineQuality-red
Wine Quality - White	wineQuality-white
Challenger USA Space Shuttle O-Ring - Erosion only	o-ring-erosion-only
Challenger USA Space Shuttle O-Ring - Erosion or blowby	o-ring-erosion-or-blowby
Concrete Slump Test	slumpTestSLUMP
	slumpTestFLOW
	slumpTestCompressiveStrength
Parkinsons Telemonitoring	parkinsonsMotorUPDRS
	parkinsonsTotalUPDRS
Airfoil Self-Noise	airfoilSelfNoise
Energy efficiency	ENB2012-Y1
	ENB2012-Y2
Yacht Hydrodynamics	yachtHydrodynamics

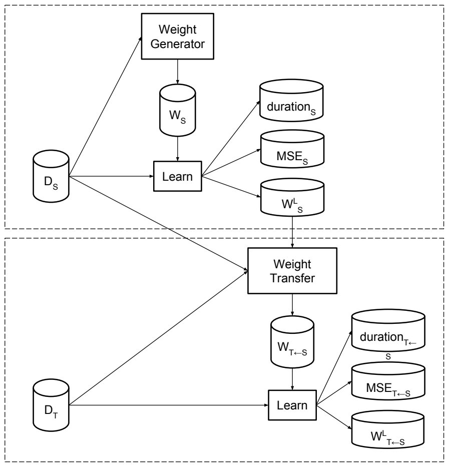

Fig. 3. Experiment workflow

this is performed for all the datasets considered, to simplify, we only represent one dataset.

We call it source dataset (D_S) because the weights used for the transfer in the second part will be the ones learned here.

The target learning step (bottom of the figure) consists of learning a neural network for the dataset considered (target dataset - D_T), by initializing it with weights transferred from one of the other datasets instead of randomly generated ones.

4.2 Results

In our experiment we measure the improvement achieved by transferring weights from another neural network – the source network –, considering the time needed for a neural network to converge (duration). We consider that an improvement has occurred when $duration_r > duration_t$, where $duration_r$ refers to the network with randomly generated initial weights and $duration_t$ refers to the network with initial weights transferred the source network.

The heat map on the left of Figure 4 shows the percentage of improvements of the best transfer for each target dataset (y axis) when using each of the mapping methods (x axis).

A red (dark) square corresponds to having almost 100% probability of improvement in the target dataset Y when using the mapping method X. The lighter the square, the closest this probability is to 0%. Grey squares correspond to the cases where the transfer worsened the result instead of improving it. The histogram on the right shows the frequency with which the percentage of improvements occur, using the same color code. The mapping methods correspond to the ones shown in Table 2.

Table 2. Average percentage of improvements by mapping method

	Mapping Measure	Average
1	Random	40.25%
2	KL-divergence	56.43%
3	Pearson Correlation	57.86%
4	Spearman Correlation	65.00%
5	Kendall Correlation	58.57%

As discussed earlier, our goal is to assess whether the right choice of source dataset could lead to improvements in the learning process of the target datasets. To investigate this, we analyze the best possible transfer observed for each dataset, instead of all the transferences, as in the previous paragraph. This approach simulates the situation where the source problem is optimally chosen. Table 2 shows the average percentage of improvement of the best transfer for each of the methods considered.

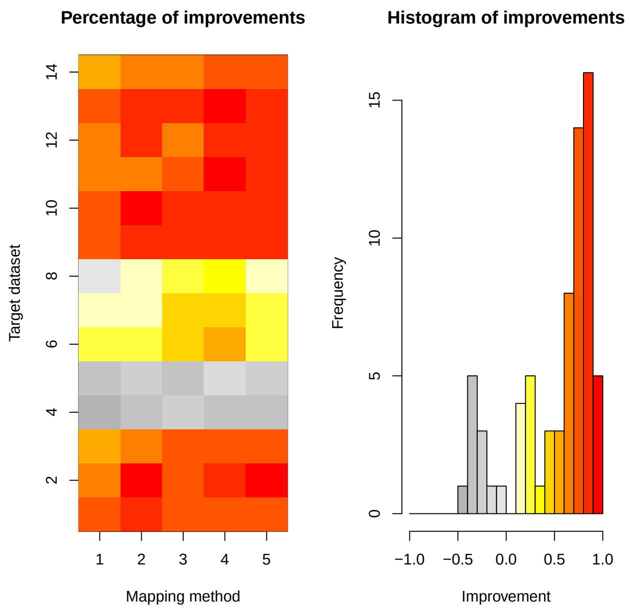

Fig. 4. Percentage of improvements for the best results

Here we can see that the proportion of improvements for the proposed mapping methods are clearly higher than for the random mapping. While random mapping of variables can improve the time of convergence of the network in around 40% of the times, a similarity-based mapping can improve it in up to 65% of the times, with the mapping functions considered. This suggests that we can benefit from the transfer of weights from a network previously trained on a different problem to a new network.

However, this benefit only occurs if the source problem is well chosen. One approach that can be used for that purpose is metalearning. Given a new target problem for which we want to train a network, a metalearning approach aims to select the most adequate source dataset based on characteristics of the datasets, usually referred to as metafeatures. The next step in our work is to investigate a metalearning approach for that purpose.

5 Conclusions and Future Work

In this paper we have proposed one specific method for transferring weights from a source network to another. This method has four variants based on different similarity measures between dataset variables: KL divergence and the Pearson, Spearman and Kendall correlation coefficients. In our experiments we have shown that transferring weights can accelerate the learning of the target network as long as the source network is well chosen. This suggests that we can benefit from a metalearning approach that is able to select a good source problem given a new one. In other words, it indicates that metalearning can be used to support transfer learning.

As future work, we wish to identify the appropriate problem characteristics (metafeatures) to be used for selecting the best variable (and dataset) to use as source in the transfer to a target network. The similarity between datasets will be used to determine the dataset whose network will be the best one to transfer the weights from. The similarity between variables will be used to determine the mapping to be used when transferring the weights between neural networks.

Acknowledgments

This project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 662189. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Spain, Finland, Denmark, Belgium, Netherlands, Portugal, Italy, Austria, United Kingdom, Hungary, Slovenia, Germany.

References

1. Brazdil, P., Giraud-Carrier, C.G., Soares, C., Vilalta, R.: Metalearning - Applications to Data Mining. Cognitive Technologies. Springer (2009)
2. Rice, J.R.: The algorithm selection problem. Advances in Computers **15** (1976) 65–118
3. Brazdil, P., Soares, C., da Costa, J.P.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning **50**(3) (2003) 251–277
4. Gama, J., Brazdil, P.: Characterization of classification algorithms. In: Progress in Artificial Intelligence, 7th Portuguese Conference on Artificial Intelligence, EPIA '95, Funchal, Madeira Island, Portugal, October 3-6, 1995, Proceedings. (1995) 189–200
5. Prudêncio, R.B.C., Ludermir, T.B.: Meta-learning approaches to selecting time series models. Neurocomputing **61** (2004) 121–137
6. Gomes, T.A.F., Prudêncio, R.B.C., Soares, C., Rossi, A.L.D., Carvalho, A.C.P.L.F.: Combining meta-learning and search techniques to select parameters for support vector machines. Neurocomputing **75**(1) (2012) 3–13
7. Serban, F., Vanschoren, J., Kietz, J.U., Bernstein, A.: A survey of intelligent assistants for data analysis. ACM Comput. Surv. **45**(3) (July 2013) 31:1–31:35
8. Abreu, P., Soares, C., Valente, J.M.S.: Selection of heuristics for the job-shop scheduling problem based on the prediction of gaps in machines. In: Learning and Intelligent Optimization, Third International Conference, LION 3, Trento, Italy, January 14-18, 2009. Selected Papers. (2009) 134–147
9. Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. **41**(1) (2008)

10. Gama, J., Kosina, P.: Learning about the learning process. In: Advances in Intelligent Data Analysis X - 10th International Symposium, IDA 2011, Porto, Portugal, October 29-31, 2011. Proceedings. (2011) 162–172
11. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10) (2010) 1345–1359
12. Dai, W., Yang, Q., Xue, G., Yu, Y.: Boosting for transfer learning. In: Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007. (2007) 193–200
13. Blitzer, J., McDonald, R., Pereira, F.: Domain adaptation with structural correspondence learning. In: Proceedings of the 2006 conference on empirical methods in natural language processing, Association for Computational Linguistics (2006) 120–128
14. Gao, J., Fan, W., Jiang, J., Han, J.: Knowledge transfer via multiple model local structure mapping. In: In International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV. (2008)
15. Mihalkova, L., Huynh, T., Mooney, R.J.: Mapping and revising markov logic networks for transfer learning. In: In Proceedings of the 22 nd National Conference on Artificial Intelligence (AAAI. (2007) 608–614
16. Pratt, L.Y., Pratt, L.Y., Hanson, S.J., Giles, C.L., Cowan, J.D.: Discriminability-based transfer between neural networks. In: Advances in Neural Information Processing Systems 5, Morgan Kaufmann (1993) 204–211
17. Silver, D.L., Poirier, R., Currie, D.: Inductive transfer with context-sensitive neural networks. Machine Learning **73**(3) (2008) 313–336
18. Rosenstein, M.T., Marx, Z., Kaelbling, L.P., Dietterich, T.G.: To transfer or not to transfer. In: In NIPS'05 Workshop, Inductive Transfer: 10 Years Later. (2005)
19. Ramon, J., Driessens, K., Croonenborghs, T.: Transfer learning in reinforcement learning problems through partial policy recycling. In: Machine Learning: ECML 2007. Springer (2007) 699–707
20. Mahmud, M., Ray, S.: Transfer learning using kolmogorov complexity: basic theory and empirical evaluations. In: Advances in neural information processing systems. (2007) 985–992
21. Eaton, E., Desjardins, M., Lane, T.: Modeling transfer relationships between learning tasks for improved inductive transfer
22. Biondi, G., Prati, R.: Setting parameters for support vector machines using transfer learning. Journal of Intelligent & Robotic Systems (2015) 1–17
23. Aiolli, F.: Transfer learning by kernel meta-learning. In: ICML Unsupervised and Transfer Learning. (2012) 81–95
24. Do, C., Ng, A.Y.: Transfer learning for text classification. In: NIPS. (2005)
25. Jain, A.K., Mao, J., Mohiuddin, K.: Artificial neural networks: A tutorial. Computer (3) (1996) 31–44
26. Bache, K., Lichman, M.: UCI machine learning repository (2013)