Metalearning for multiple-domain Transfer Learning

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 over a determined dataset. Transfer learning allows the reuse of knowledge from previous tasks. Our aim is to use metalearning to support transfer learning and reduce the computational cost without lost in terms of performance, as well as the user effort needed for the algorithm selection. In this paper we propose some methods for mapping the transfer of weights between neural networks to improve the performance of the target network, and describe some experiments performed in order to test our hypothesis.

1 Introduction

Machine learning processes consist in 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 one or more tasks previously solved, for example, when there are relationships between variables or between the processes used to obtain the models.

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

Our aim is to investigate if metalearning can be used to support transfer learning in tasks consisting of very diverse subtasks, reducing computational cost without loss in predictive performance and cutting down the time data scientists need to perform their tasks.

In this paper we describe some aspects of the state of the art in metalearning and transfer learning. We propose some methods for mapping the transfer of weights between neural networks and describe experiments performed to test the hypothesis that the transfer of weights improves the results of the target network.

2 Metalearning and Transfer Learning

This chapter 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 aims at helping in the process of selecting a predictive algorithm to use on a determined dataset. It also aims at taking advantage of the repetitive use of a determined method over a set of similar tasks.

There are several applications for metalearning. It can be used in combining base learners: using several learners together to create a composite model that better predicts the result. Another application of metalearning is bias management, mostly used for data streams (continuous flows of data, for example from large and continuously growing databases) that require context adaptation due to the fact that the domain is not static. Metalearning can also be used to transfer metaknowledge across tasks. It is mostly used for the Algorithm Selection Problem, described next.

Algorithm Recommendation Choosing the best algorithm for processing a given dataset is a difficult process. Besides, the 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 [1], originally formulated by Rice [2].

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, gives the system the ability to recommend the best suited algorithm.

Earlier applications of metalearning addressed the most common tasks - classification [3], regression [4] and time series [5]. These approaches were then extended to selecting parameter settings for a single algorithm [6], the whole data mining process [7] and also to problems from domains other than machine learning, e.g.: different optimization problems [8, 9]. More recently, they were also used to deal with new problems in data mining: data streams [10].

2.2 Transfer Learning

A definition of transfer learning can be found in [11]: 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 help improve the 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 the tasks and distributions used in training and testing to be different. Here, the knowledge is transferred from one task, the source task, to another, the target task. It is inspired in the logic used by the human brain: the methods that allow, for example, someone to recognize pears based on previous knowledge on recognizing apples.

Transfer learning allows algorithms to adapt to new tasks based on the knowledge obtained in previous ones, and 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 improve the performance of the target model, as in TrAdaBoost [12] algorithm; feature-representation-transfer, where a set of feature representations is extracted from the source domain and transferred, obtaining a feature representation of the target domain as in [13]; parameter-transfer that is done by calculating the source model, extracting its parameters and, assuming that the models for related tasks share some parameters, transferring them to build the target model as in [14]; and relational-knowledge-transfer, that consists in trying to transfer the knowledge about data between the domains, as is the case of the TAMAR [15] algorithm.

How to transfer? After knowing the information that should be transferred, the focus is on *how to transfer?*, that is, on the development of learning algorithms to perform the transfer. For example, the DBT (Discriminability-based transfer) algorithm [16] consists in modifying the neural network weights obtained in the source classification task in order to use them on a target network. In [17], a "transfer-aware" naive Bayes classification algorithm is proposed. In [18], first order decision trees are used for reinforcement learning, and some tree statistics are transferred from the source to the target problem. In [19], graph-based transferability is determined: it automatically determines the parameters to transfer between biased logistic regression tasks. The Kolmogorov complexity between tasks is used in [20] to transfer knowledge between bayesian decision trees. [21] introduces a context-sensitive multi task learning that helps improving performance in neural networks for classification. In [22] the authors use clustering to perform a feature selection to be transferred, improving the performance of a Bayesian algorithm.

When to transfer? The last question means to know in which situations the transfer should be performed. Ultimately, the objective is to avoid *negative*

transfer: when the transfer can harm the learning process in the target task. This issue is referred in [17], where the authors wish to identify when transfer learning will hurt the performance of the algorithm instead of improving it.

2.3 Metalearning and Transfer Learning

Some work has been performed in using metalearning together with transfer learning. We analyzed some literature related to classification tasks that is described next.

Metafeatures are used in [23] for calculating similarities between the datasets. The algorithms used for this task is the *k-nearest neighbors*. In [24, 25] there is no use of metafeatures, since the transfers are made without choosing the best source dataset to use with a certain target dataset. In [24], metalearning is used to find matrix transformations capable of producing good kernel matrices for the source tasks. The matrices will then be transferred for the target tasks.

The results are evaluated by performance measures as accuracy [25] and more precisely by the area under the ROC curve in [23, 24].

The transferred objects found on the studied papers are SVM parameter settings in [23], the kernel matrices in [24] and the *parameter function* (responsible for mapping statistics to parameters in "bag-of-words" text classification problems) in [25].

3 Mapping of variables for transfer

We now propose some methods for mapping variables for transfer and show the results of applying the methods in some experiments, using neural networks with three neurons on the hidden layer. The transfer is made from one variable on the source dataset to another one on the target dataset. In a neural network each neuron corresponds to a variable on the dataset, and has a connection to all the neurons on the hidden layer.

The methods proposed are described next:

1. **Random:** the weights are randomly ordered. We repeat this 100 times and generate 100 sets of randomly ordered weights.
2. **Direct:** the weights are transferred directly between corresponding variables, when the datasets have the same structure.
3. **Mapped:** the weights are ordered according to some criteria:
   (a) **Kullback-Leibler divergence:** we obtain the KL divergence between all the attributes of the source dataset and and all the attributes of the target dataset. The transfer is made between the attributes with smaller divergence.
   (b) **Pearson, Spearman and Kendall correlations:** we obtain the correlation between every attribute in each dataset and its target. The transfer is made between the attributes with the most similar correlation to the respective target.

4 Experiments performed

Some experiments have been performed to study if the transfer of knowledge improves the performance of an algorithm. The aim is to measure the success of transferring the weights of a neural network learned on a source dataset to a new neural network that will be trained on a target dataset.

In the experiments, the source and target datasets may be unrelated or related (e.g. generated by the same process in different times or generated by processes with the same structure). Weight transfer is performed from one variable in the source model to one variable in the target model. The datasets used were retrieved from UCI [26] and different experiments have been made.

4.1 Experiment 1

The objective of this experiment is to study the behavior of the transfer of knowledge between tasks. We compared random transfer (made between randomly chosen variables) with direct transfer (performed between correspondent variables, in related datasets).

Experiment Description In this experiment, source and target datasets may be unrelated or related (e.g. generated by the same process in different times or generated by processes with the same structure). Weight transfer is performed from one variable in the source model to one variable in the target model. The mapping of variables for the transfer can be random or between corresponding variables.

One of the datasets used, Communities and Crime Unnormalized, has 18 target variables. It was used to generate new datasets using the same original independent variables. These datasets are, then, related to each other. The other datasets are, in principle, independent among themselves. All the datasets were normalized in a preprocessing phase.

For this experiment we ran each dataset through a neural network with three neurons in the hidden layer, using ten-fold crossed validation. First the networks are trained with a random initial set of weights, and we measure the *Mean Squared Error*,$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$. Then each network is trained with the best set of weights found for the other networks and we also measure the MSE for each network. For each network we compare the error obtained with random initial weights (MSE_R) with the ones obtained with the weights transferred from other networks (MSE_T). We consider that the transfer has improved the result when MSE_R is bigger than MSE_T.

For the unrelated datasets, the transfer was performed randomly. For the related datasets the transfer was performed in two different ways: randomly and also directly between corresponding variables.

Results Figures 1 to 3 show the distribution of the improvements for the experiments. In the x and y axis we have the datasets. We calculated the number of

times when transfer improves the MSE. In these charts the color of the squares represents the number of times the transfer between those datasets improved the performance on the target task: darker squares represent a higher probability of reducing the error when using transfer of weights.

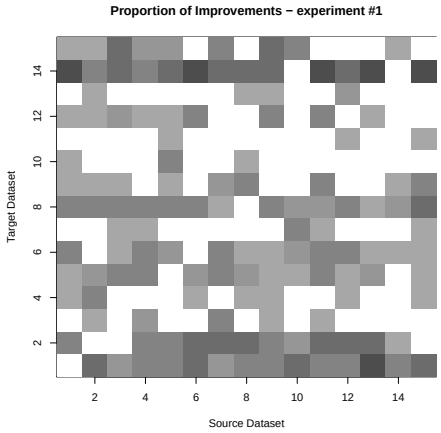

Fig. 1. Distribution of the number of improvements for the first variant of the experiment

In Figures 2 and 3, the values inside the squares represent the Pearson Correlation of the target variables for each pair of datasets.

Figure 4 shows boxplots with the distributions of improvements for the 20 runs of each variant of the experiment. This chart shows that in the first variant the improvement is lowest. Note that datasets used in the second and third variants of the experiment are related, unlike the ones used in the first.

A plausible cause for the last variant of the experiment being the one with more improvements is that not only the datasets are related, but also the transfer of weights is made directly between corresponding variables from one dataset to another, because the structure of the neural network is the same.

The improvement obtained was near 50% for the random transfer between unrelated datasets. This means that random transfer has the same probability of improving the result as it has of deteriorating it. The random and direct transfers between related datasets (with the same attributes but different target variables) show, respectively, around 60% and 70% of improvements. This means that the transfer between related datasets increases the probability of improving the result of a neural network. This probability increases even more when the transfer is made directly between corresponding variables, showing that the transfer between similar (in this case, the same) variables is advantageous.

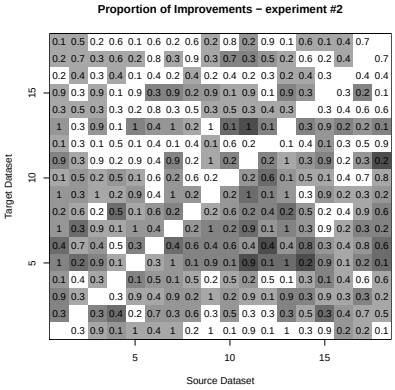

Fig. 2. Distribution of the number of improvements for the second variant of the experiment

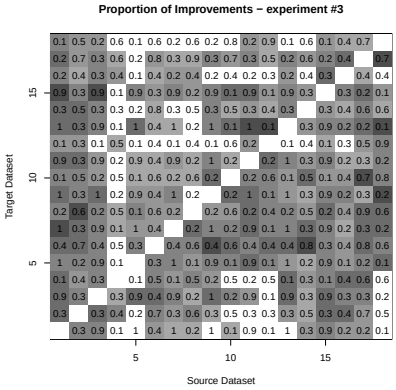

Fig. 3. Distribution of the number of improvements for the third variant of the experiment

4.2 Experiment 2

The objective of this experiment was to study the behavior of the transfer of knowledge between similar variables, comparing it to the random transfer of knowledge.

Experiment description In this experiment, source and target datasets are in principle unrelated. The datasets used were the ones considered as unrelated on Experiment 1.

Before running the experiment, the datasets were subject to a preprocessing phase, which included the normalization. For this experiment we ran the datasets through a neural network with three neurons in the hidden layer, using ten-fold crossed validation.

First, the initial set of weights fed to each neural network is composed by values generated randomly between 0 and 1. In order to outwit the randomness of the weight generation, the whole processed is repeated 100 times for each dataset. The dataset and weights are fed to the neural network and, using ten-fold crossed validation, we obtain the Mean Squared Error and the Aggregated Weights (mean of the ten sets of weights obtained from the network).

These aggregated weights are transfered to other neural networks to try to improve their performance. The transfer is performed in two ways: random and mapped and the weights are fed to the neural network, together with the target dataset. The learning process occurs and the resulting mean squared error is saved.

The errors obtained in the first learning process (with randomly generated weights - MSE_R) are compared with the ones obtained in the second learning

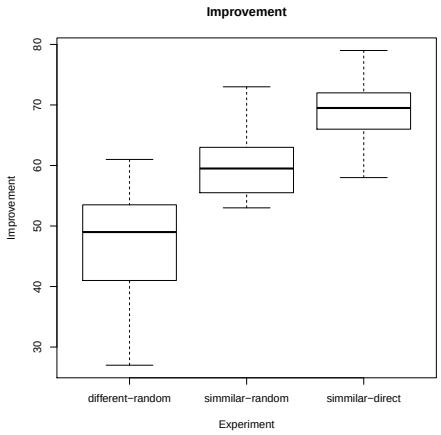

Fig. 4. Global distribution of the MSE's improvement with transfer over the three experiments

process (with the weights transferred from the other datasets - MSE_T). For this, we calculate: $\frac{MSE_O - MSE_T}{MSE_O}$.

For each pair of datasets, we repeat the transfer several times: 10000× (100×100) for random transfer and 100× for mapped transfer.

Results The chart in Figure 5 shows the probability of improving the performance of the neural network by transferring the weights using the same dataset as source and target.

We can see in the chart that the transfer of the same set of weights generates more improvements than using a new random set of weights. This is because the first is equivalent to running the neural network for twice the iterations, leading to a better fitting of the result.

The charts in Figures 6 to 10 show the results for the different types of mapping: random, using Kullback-Leibler divergence, Pearson, Spearman and Kendall correlations, respectively.

For the random mapping the figure shows, in the left, the mean number of times the transfer improves the predictions and, in the right, the histograms of the same information, where the colors match the ones on the images on the left: gray tones for when the transfer increases the error and the other colors for when there is an improvement.

The same information is shown in the charts that refer to the other types of mappings used. For these, we added a chart, in the middle, that shows the difference, in terms of improvement, between the measured mapping methods and the random mapping method. The colors also match the ones in the histogram.

In all cases the proportion of improvements is below (but near) 50%. Our aim is to find the proper features that allow this proportion to increase.

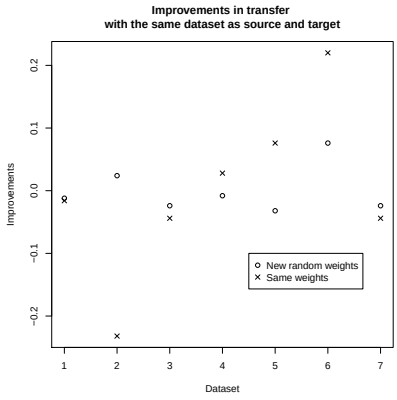

Fig. 5. Improvements in transfer with the same dataset as source and target

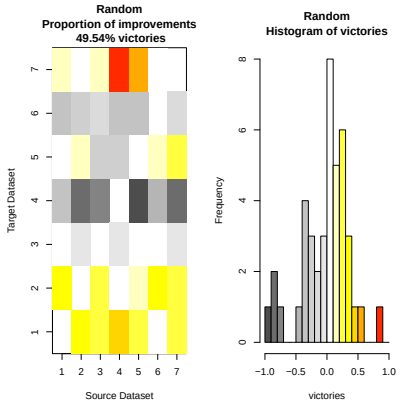

Fig. 6. Results for random mapping

5 Conclusions and Future Work

We can use related variables to identify characteristics of the model that can be transfered with the advantage of reducing the computational cost and the user effort on the process.

In this paper we described methods for mapping the transfer of weights between neural networks. We also show results of some experiments performed to test the hypothesis that the transfer of weights will improve the results of the neural network.

In the first experiment we obtained an improvement near 50% for the random transfer between unrelated datasets and around 60% and 70% of improvements random and direct transfers between related datasets, respectively. This shows that the transfer between similar datasets is advantageous, and the advantages increase even more when the transfer is performed between similar variables.

In the second experiment, that was performed with unrelated datasets, we obtained probabilities of improvement below (but near) 50% for all the mappings considered. We aim to find the proper features that allow increasing this value.

Acknowledgments

This work is financed by the ERDF - European Regional Development Fund through the COMPETE programme (operational programme for competitiveness) within project GNOSIS, cf. FCOMP-01-0202-FEDER-038987.

References

1. Brazdil, P., Giraud-Carrier, C.G., Soares, C., Vilalta, R.: Metalearning - Applications to Data Mining. Cognitive Technologies. Springer (2009)

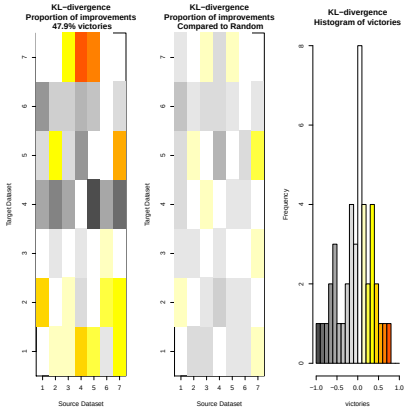

Fig. 7. Results for mapping with Kullback-Leibler divergence

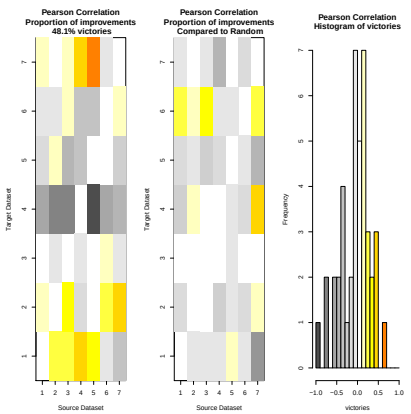

Fig. 8. Results for mapping with Pearson correlation

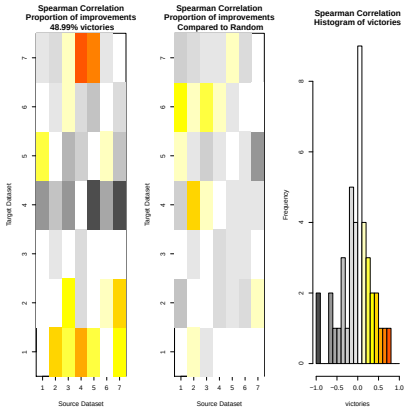

Fig. 9. Results for mapping with Spearman correlation

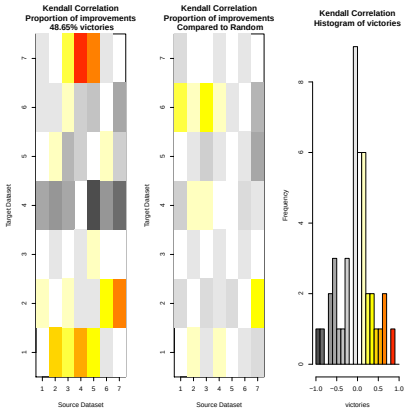

Fig. 10. Results for mapping with Kendall correlation

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. 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)
18. 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
19. Eaton, E., Desjardins, M., Lane, T.: Modeling transfer relationships between learning tasks for improved inductive transfer
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. Silver, D.L., Poirier, R., Currie, D.: Inductive transfer with context-sensitive neural networks. Machine Learning **73**(3) (2008) 313–336
22. Mishra, M., Huan, J.: Multitask learning with feature selection for groups of related tasks. In: Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE (2013) 1157–1162
23. Biondi, G., Prati, R.: Setting parameters for support vector machines using transfer learning. Journal of Intelligent & Robotic Systems (2015) 1–17
24. Aiolli, F.: Transfer learning by kernel meta-learning. In: ICML Unsupervised and Transfer Learning. (2012) 81–95
25. Do, C., Ng, A.Y.: Transfer learning for text classification. In: NIPS. (2005)
26. Bache, K., Lichman, M.: UCI machine learning repository (2013)