Using metalearning for parameter tuning in neural networks

Catarina Félix, Carlos Soares, Alípio Jorge and Hugo Ferreira

Abstract Neural networks have been applied as a machine learning tool in many different areas. Recently, they have gained increased attention with what is now called *deep learning*. Neural networks algorithms have several parameters that need to be tuned in order to maximize performance. The definition of these parameters can be a difficult, extensive and time consuming task, even for expert users. One approach that has been successfully used for algorithm and parameter selection is metalearning. Metalearning consists in using machine learning algorithm on (meta)data from machine learning experiments to map the characteristics of the data with the performance of the algorithms. In this paper we study how a metalearning approach can be used to obtain a good set of parameters to learn a neural network for a given new dataset. Our results indicate that with metalearning we can successfully learn classifiers from past learning tasks that are able to define appropriate parameters.

Catarina Félix
INESC TEC, Faculdade de Ciências, Universidade do Porto, Portugal
R. Dr. Roberto Frias, 4200 Porto
e-mail: cfo@inesctec.pt

Carlos Soares
INESC TEC, Faculdade de Engenharia, Universidade do Porto
R. Dr. Roberto Frias, 4200-465 Porto, Portugal
e-mail: csoares@fe.up.pt

Alípio Jorge
INESC TEC, Faculdade de Ciências, Universidade do Porto, Portugal
R. Dr. Roberto Frias, 4200 Porto
e-mail: amjorge@fc.up.pt

Hugo Ferreira
INESC TEC, R. Dr. Roberto Frias, 4200 Porto,
e-mail: hmf@inesctec.pt

1 Introduction

Machine learning (ML) processes consist in collecting data, applying an algorithm to that data to obtain a model and using the model to understand the problem (descriptive ML) and/or make predictions concerning new observations from that problem. To obtain a model we need to choose an algorithm and tune its parameters. The parameter tuning task is usually done manually or semi-automatically (e.g. by searching on a grid of parameter values for the configuration that obtain the best model), based on the user's expertise. Even for experienced user, it is a complex task that consumes much time and resources. Neural networks (NN) have many parameters with significant impact on model performance, which are well-know to be particularly hard to tune [1].

Metalearning has been used to address the problem of parameter tuning [11]. It essentially consists in using a ML approach to that problem, namely to map the characteristics of the data to the performance of the algorithms (and different parameter configurations) [13].

This means collecting (meta)data about the characteristics of problems modeled in the past and the performance of different algorithms and parameter configurations on those problems; applying a ML algorithm to that (meta)data to obtain a (meta)model, mapping problem characteristics (i.e. metafeatures) to algorithm performance; and using that metamodel to predict the best algorithm and parameter configuration for a new dataset. One of the main challenges in metalearning is the development of metafeatures that provide useful information about the performance of different algorithms and their configurations [2].

In this paper, we use metalearning methods for parameter tuning in NN. We propose a new set of metafeatures that are specifically designed for NN. We test the approach with a set of benchmark datasets from the UCI repository [9]. The results indicate that metalearning is able to predict good configurations for NN and is, thus, a suitable approach to the problem of parameter tuning of NN.

The main contributions of this work are:

- developing of specific metafeatures for the problem of parameter tuning of NN, and
- showing that metalearning is a viable approach to that problem.

In the remainder of the paper we introduce NNS (Section 2) and metalearning (Section 3). Then we describe the empirical study carried out (Section 5) and discuss the corresponding results (Section 6). In the last section, we present some conclusions and directions for future work.

2 Neural Networks

In ML, a neural network consists of a network with (possibly) several layers of nodes that are used to model data. In classification and regression problems, the first layer

represents the input variables and the final layer is the target variable. Each layer, except the input layer, receives the values of the previous layer as input, transforms them and outputs the result to the following layer. The connection between two nodes is associated with a weight, that regulates the effect of the output of one node on the other. The algorithm to learn a neural network uses training data to determine the value of the weights [1].

Figure 1 shows an example of the neural networks considered in this work. It is the traditional three-layered architecture namely, the input, hidden and output layers.

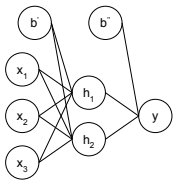

Fig. 1: Example of a neural network

As it is described in more detail in [3], here we only introduce the notation: x_i are the values fed to the input layer; h_j are hidden neutrons; (b') and (b'') are the bias neurons; y is the neuron on the output layer, representing the target; the set of weights W contains $(i+1)\times j+(j+1)\times o$ elements, where i is the number of input units, j is the number of hidden neurons, and o is the number of outputs.

Neural networks are a very popular learning algorithm [12]. Many different variants of NN have been proposed and they have been used for many different tasks, such as: pattern recognition, prediction, optimization, associative memory, and control [4]. More recently, *deep learning*, which is essentially a NN with many hidden layers, have been very successful [7].

Several papers contain good introductions to NN (e.g. [6, 10]).

3 Metalearning

Metalearning is essentially the use of machine learning (ML) techniques to model the behavior of ML algorithms based on data collected from their usage [2]. It has been mostly used for the algorithm selection problem [14]. This is a very general problem, that can be described as: given a new problem and a set of algorithms that may be used to solve it, choose the best algorithm to solve it, according to some performance measure. In the case case of ML, the problems are typically datasets and the algorithms are learning algorithms but we can also include the choice or tuning of algorithm parameters.

Metalearning can naturally be applied to the problem of algorithm (and parameter) selection in ML [2]. Figure 2 shows this process. The approach con-

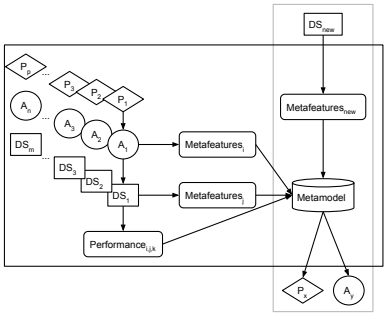

Fig. 2: Metalearning for algorithm selection

sists of collecting *metadata* about datasets as they are processed, characterizing the dataset ($Metafeatures_j$), the algorithm ($Metafeatures_i$) and the corresponding $Performance_{i,j,k}$ of different algorithms and parameter configurations with each dataset. Then, a ML algorithm is applied to that metadata to obtain a *metamodel*. The metamodel maps metafeatures to algorithm performance. The metamodel can then be used to predict the best algorithm (A_y) and parameter configuration (P_x) for a new dataset, given the values of its metafeatures.

4 Metafeatures

An important issue in the development of metalearning approaches to ML is the design of useful metafeatures, i.e. characteristics from the data that provide information about the performance of the algorithms [2]. These can be grouped into tree main categories: 1) simple, statistical and information-theoretic metafeatures – features based on the dataset characteristics, (e.g: number of classes, number of features); 2) model-based – features based on models obtained with the data (e.g. building a decision tree from a dataset and collect properties of the tree, like nodes per feature, maximum tree depth, etc.); and 3) landmarkers – quick estimates of the performance of the learners obtained by either running simplified versions of the algorithms, or running the algorithm on a sample of the dataset.

In this paper we consider 75 metafeatures, that are presented in Table 1. Instead of the three groups presented earlier, we group them according to the type of information they represent:

G1 Global dataset: number of examples and attributes and their ratio;
G2 Individual attributes: number and ratio of attributes consisting of only two or three distinct values, or attributes with outliers;
G3 Relationship between attributes: correlations among the attributes;
G4 Attribute distributions: distribution of the values of the attributes;
G5 Target attributes: existence of outliers;

G6 Target distribution: distribution of the values of the target variable;
G7 Relationship between attributes and target: correlations between the attributes and the target variables;
G8 General landmarkers: result of applying simple models to the data. Here, clustering.3,5,10,20 means we performed clustering with the referred numbers of clusters, measuring the standard deviation of the number of elements placed in each cluster;
G9 NN specific landmarkers: result of applying simpler models to the data. Here, we chose to use smaller neural networks and measuring the initial *MSE* (*mse0*), distribution of the differences between initial and final weights (*w.mean.dif* and *w.sd.dif*), and also the difference between the initial and final *MSEs* (*mse.dif*). We used neural networks with only one neuron in the hidden layer (*1h*), but also with three (*3h*). In this last case, we limited the maximum number of iterations between 1 and 10.

Table 1: Metafeatures used in this work

G1	**G4**	**G7**
n.examples	avg.skewness	prop.target.cor.gt.50
n.attrs	avg.abs.skewness	avg.abs.target.correlation
r.n.attrs.n.examples	avg.kurtosis	
r.n.examples.n.attrs	avg.means	**G8**
	avg.sds	r.squared
G2		clustering.{3, 5, 10, 20}
n.bin.fea	**G5**	d.tree.leaves
n.h.outlier	target.h.outlier	d.tree.mse
n.tri.fea	target.has.outliers	mean.mse
r.num.bin.fea.n.examples		
r.n.h.outlier.n.attrs	**G6**	**G9**
r.n.h.outlier.n.examples	range.target.rel.avg	l.nnet.1h.mse
r.num.tri.fea.n.attrs	target.coefficient.variation	l.nnet.1h.w.mean.dif
r.num.tri.fea.n.examples	abs.target.coefficient.variation	l.nnet.1h.w.sd.dif
r.num.bin.fea.n.attrs	target.cv.sparsity	l.nnet.1h.mse0
	target.abscv.sparsity	l.nnet.3h.mse0
G3	target.stationarity	l.nnet.3h.it.{1..10}.w.mean.dif
avg.abs.attr.correlation	target.hist.sparsity	l.nnet.3h.it.{1..10}.w.sd.dif
prop.cor.gt.50	avg.mean.res.dist.adjacent.target	l.nnet.3h.it.{1..10}.mse.dif

Given the large number of features, we performed three types of feature selection (FS): filter, wrapper and knowledge-based.

Filter FS is based on the correlation matrix of the metafeatures. In groups of metafeatures with a correlation higher than a given threshold, we may consider all but one as redundant. Thus, the redundant metafeatures are discarded. For this we used R package `caret` [5], specifically the method `findCorrelation` which, using a cutoff factor of 0.75, recommended removing 59 metafeatures.

Wrapper FS was applied to the remaining 16 metafeatures. Here, we used `caret`'s function `rfe` to perform recursive feature elimination, resulting in sub-

sets of 2, 14, and 11 features that can be used to produce a more accurate model for the neurons, decay, and abstol problems, respectively.

In knowledge-based feature selection, for each of the three problems considered, we chose a subset of metafeatures that could better describe th behavior of each NN parameter.

To extract some knowledge from the metamodel we analyzed a decision tree and also measured the importance of the variables. For this we used `caret`'s method `varImp`, that estimates the importance of the variables for a model.

5 Experimental Setup

We conducted a set of experiments to test the use of metalearning to predict the best configuration of a neural network. The experiments consisted in the following two levels: 1) at the base level we performed a grid search over three neural network parameters, measuring the resulting *MSE* (Section 5.2) 2) at the meta level, we learn and test metamodels from the data collected with the base level experiments. Before presenting the experimental setup at each level, we start by presenting the datasets used.

5.1 Datasets

Table 2 shows the datasets used in this paper. Some of the datasets considered have more than one target variable. In the table, the $\star$ symbol means that several datasets were created by splitting those by target variable.

Table 2: UCI Datasets used and number of datasets generated from each one of them

id	name	nr. datasets
11_*	Parkinsons Telemonitoring	2
12_*	Concrete Slump Test	3
15_1, 16_1	Wine Quality	1
17_1	Yacht Hydrodynamics	1
1_1	Airfoil Self-Noise	1
2_*	Condition Based Maintenance of Naval Propulsion Plants	2
3_1	Combined Cycle Power Plant	1
4_1	Communities and Crime	1
5_*	Communities and Crime Unnormalized	4
6_1	Concrete Compressive Strength	1
7_1	Computer Hardware	1
8_1, 9_1	Challenger USA Space Shuttle O-Ring	2

5.2 Base level Experiments

The structure of the NN used is similar to the one presented in [3]: the networks have three layers and the number of neurons in the input and output layers are, respectively, the number of independent and dependent variables of the dataset, which is one in our case. To perform the experiment we used the R package `nnet` [16]. The neural network implementation considered in this package uses traditional backpropagation, and does not allow specifying the learning rate.

The parameters we want to tune are:

neurons (n): number of neurons in the hidden layer: $n \in \{3,5,10,20\}$
decay (d): parameter for weight decay: $d = 1 \times 10^{-i}, i \in [0,4]$
abstol (a): stopping criterion: $a = 1 \times 10^{-j}, j \in [3,5]$

We tested every possible combination of these parameters, leading to 60 different parameter configurations.

In each of those configurations, the neural networks were initialized with sets of weights generated randomly from two distributions: uniform: $\mathscr{U}[0,1]$; and based on Bishop's recommendation [1]: $\mathscr{N}\left(0, \sqrt{1/x}\right)$, where x is the number of independent variables of the dataset. We repeated each experiment 20 times, with different initial weights. We have, thus, $20 \times 2 \times 60 = 2,400$ experiments for each one of the benchmark datasets used.

The performance of each neural network is estimated with 10-fold cross-validation. For each fold, we create a model by running the neural network on the train set until it converges. Then we apply the resulting model to the test set and evaluate it by computing its *MSE*.

Based on these results, we can determine the best set of parameters: the one that originated the smallest *MSE*.

5.3 Meta level Experiments

Taking into account the grouping and feature selection referred in section 4, in this experiment we used 13 different groups of metafeatures:

- G1 ... G9: each of the groups referred in Table 1M
- NR: the 16 non redundant metafeatures suggested by filter feature selection;
- IMP: metafeatures with variable importance over 50;
- RFE: metafeatures recommended by recursive feature elimination for each problem;
- KB: metafeatures selected from knowledge-based feature selection.

Although the target variables of the metadatasets are numeric, we addressed the metalearning problems as classification tasks: the prediction of the number of `neurons`, value of `decay` and value of `abstol`.

The reason for this is because we are not treating these parameters as continuous. In other words, as we only have results for the parameter values that define the grid, we can only handle directly predictions that are also in that grid.

The quality of the recommendations of the metamodel was measured in two different ways: 1) measuring the accuracy of the predictions in the three metatargets separately; and 2) measuring base-level performance, i.e. MSE, obtained by a NN using the configuration recommended by the three metamodels. The meta-level performance was estimated using leave-one-out cross-validation.

The base-level performance can be compared to the performances of the best and worst networks (respectively, the NNs with smallest and greatest *MSE* from within the several NNs tested). It can also be compared to the baseline prediction: always predicting the class with more observations.

The algorithms used at the meta level were decision trees and random forests. The baseline recommendation method consists of the most common class for each problem, commonly known as the default class.

6 Results

Table 3 shows the meta level accuracies obtained on each of the three problems, when using decision trees (rpart [15]) and a random forests (randomForest [8]) with each group of metafeatures described earlier.

Table 3: Meta level accuracies

	neurons		decay		abstol	
	rpart	randomForest	rpart	randomForest	rpart	randomForest
G1	0.476	0.429	0.381	0.429	0.286	0.429
G2	0.429	0.429	0.381	0.476	0.524	0.619
G3	0.476	0.429	0.333	0.429	0.333	0.333
G4	0.286	0.381	0.429	0.429	0.476	0.476
G5	0.333	0.476	0.095	0.048	0.333	0.476
G6	0.381	0.429	0.429	**0.524**	0.476	0.476
G7	0.429	0.476	0.333	0.333	0.286	0.333
G8	0.333	0.429	0.333	0.476	0.333	0.381
G9	0.381	0.286	0.333	0.286	0.286	0.333
NR	0.524	0.476	0.286	0.429	0.381	0.524
IMP	0.524	0.571	0.190	0.381	0.524	0.619
RFE	**0.619**	**0.619**	0.286	0.429	0.524	**0.667**
KB	0.571	0.524	0.381	0.381	0.381	0.524
baseline	0.476		0.238		0.476	

As the results in the table show, we achieved good meta level accuracies when compared to the baseline method. Looking at the values we can see that random forests (RF) generally achieves the best results. Concerning the groups of metafeatures, RFE leads to higher accuracy for predicting the number of neurons (0.619)

and the value to use as abstol (0.667), while G6 is the best to predict the decay parameter (0.524).

Table 4 shows the average performances (in terms of MSE) of the neural networks executed following the recommended configuration. We can compare these with the average performance of the ones executed with the configuration recommended by the baseline, and also with the best and the worst average performances obtained for each dataset.

Table 4: Base level performances

	rpart	randomForest
G1	0.344	0.339
G2	0.347	0.319
G3	0.285	0.298
G4	0.262	**0.243**
G5	0.318	0.467
G6	0.459	0.342
G7	0.258	**0.243**
G8	0.345	0.273
G9	0.411	0.658
IMP	0.302	0.266
NR	0.269	0.333
RFE	0.247	0.262
KB	0.264	0.252
best	0.205	
worst	0.856	
baseline	0.475	

As the results on the table show, the recommended configurations lead to NN with MSE very close to the best performances. They are clearly better than the worst and only in a single case, metalearning does not lead to better results than the baseline (0.658 when using the group G9 with random forest). The best performance is achieved when using the group of metafeatures G4 or G7 with random forests.

7 Conclusions

We use metalearning to recommend a good set of parameters for a neural network on new datasets. For the characterization of the datasets we have identified groups of metafeatures that are used in this approach as attributes in the metalearning process. The experimental results obtained are very encouraging. The metalearning approach can predict a good configuration for a neural network. This indicates that metalearning can be used for this purpose, thus reducing the effort required in the tuning process done by the user manually or semi-automatically (e.g. using a grid). As future work we aim to improve results by proposing new metafeatures and other metalearning algorithms.

Acknowledgements 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 Unions Horizon 2020 research and innovation programme and Spain, Finland, Denmark, Belgium, Netherlands, Portugal, Italy, Austria, United Kingdom, Hungary, Slovenia, Germany

References

1. Bishop, C.M.: Neural networks for pattern recognition. Oxford university press (1995)
2. Brazdil, P., Giraud-Carrier, C.G., Soares, C., Vilalta, R.: Metalearning - Applications to Data Mining. Cognitive Technologies. Springer (2009). DOI 10.1007/978-3-540-73263-1. URL http://dx.doi.org/10.1007/978-3-540-73263-1
3. Félix, C., Soares, C., Jorge, A.: Can metalearning be applied to transfer on heterogeneous datasets? In: International Conference on Hybrid Artificial Intelligence Systems, pp. 332–343. Springer (2016)
4. Jain, A.K., Mao, J., Mohiuddin, K.: Artificial neural networks: A tutorial. Computer (3), 31–44 (1996)
5. from Jed Wing, M.K.C., Weston, S., Williams, A., Keefer, C., Engelhardt, A., Cooper, T., Mayer, Z., Kenkel, B., the R Core Team, Benesty, M., Lescarbeau, R., Ziem, A., Scrucca, L., Tang, Y., Candan, C., Hunt., T.: caret: Classification and Regression Training (2017). URL https://CRAN.R-project.org/package=caret. R package version 6.0-76
6. Kröse, B., Krose, B., van der Smagt, P., Smagt, P.: An introduction to neural networks (1993)
7. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
8. Liaw, A., Wiener, M.: Classification and regression by randomforest. R News **2**(3), 18–22 (2002). URL http://CRAN.R-project.org/doc/Rnews/
9. Lichman, M.: UCI machine learning repository (2013). URL http://archive.ics.uci.edu/ml
10. Lippmann, R.: An introduction to computing with neural nets. IEEE Assp magazine **4**(2), 4–22 (1987)
11. Molina, M., Luna, J., Romero, C., Ventura, S.: Meta-learning approach for automatic parameter tuning: A case study with educational datasets. International Educational Data Mining Society (2012)
12. Paliwal, M., Kumar, U.A.: Neural networks and statistical techniques: A review of applications. Expert systems with applications **36**(1), 2–17 (2009)
13. Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. Machine learning **87**(3), 357–380 (2012)
14. Rice, J.R.: The algorithm selection problem. Advances in Computers **15**, 65–118 (1976). DOI 10.1016/S0065-2458(08)60520-3. URL http://dx.doi.org/10.1016/S0065-2458(08)60520-3
15. Therneau, T., Atkinson, B., Ripley, B.: rpart: Recursive Partitioning and Regression Trees (2017). URL https://CRAN.R-project.org/package=rpart. R package version 4.1-11
16. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S, fourth edn. Springer, New York (2002). URL http://www.stats.ox.ac.uk/pub/MASS4. ISBN 0-387-95457-0