Proceedings of Machine Learning Research 140:127–136, 2021 AAAI Workshop on Meta-Learning and MetaDL Challenge

Learning Abstract Task Representations

Mikhail M.Meskhi MMESKHI@UH.EDU
Department of Computer Science
University of Houston, Houston, TX 77204
Adriano Rivolli RIVOLLI@UTFPR.EDU.BR
Universidade Tecnológica Federal do Paraná
Cornélio Procópio, Brazil
Rafael G. Mantovani RAFAELMANTOVANI@UTFPR.EDU.BR
Universidade Tecnológica Federal do Paraná
Apucarana, Brazil
Ricardo Vilalta RVILALTA@UH.EDU
Department of Computer Science
University of Houston, Houston, TX 77204

Editors: Isabelle Guyon, Jan N. van Rijn, Sébastien Treguer, Joaquin Vanschoren

Abstract

A proper form of data characterization can guide the process of learning-algorithm selection and model-performance estimation. The field of meta-learning has provided a rich body of work describing effective forms of data characterization using different families of meta-features (statistical, model-based, information-theoretic, topological, etc.). In this paper, we start with the abundant set of existing meta-features and propose a method to induce new abstract meta-features as latent variables in a deep neural network. We discuss the pitfalls of using traditional meta-features directly and argue for the importance of learning high-level task properties. We demonstrate our methodology using a deep neural network as a feature extractor. We demonstrate that 1) induced meta-models mapping abstract meta-features to generalization metrics outperform other methods by $\sim 18\%$ on average, and 2) abstract meta-features attain high feature-relevance scores.

1. Introduction

Meta-learning (MtL) allows rational agents to improve on their learning abilities through a process known as *learning to learn* (Hospedales et al., 2020; Schmidhuber, 1987; Vanschoren, 2018; Vilalta and Drissi, 2002). One major goal is to build self-adaptive systems that adjust their learning mechanism automatically with new tasks. Automatic adaptation can be described in a plethora of ways; it can be as simple as tuning hyper-parameters, selecting a different family of learning algorithms, or simply warm-starting a model. Meta-learning relies on past experience stored in the form of *meta-knowledge*. One type of meta-knowledge encompasses families of meta-features used as a form of data (or task) characterization. Meta-features capture various types of data properties such as statistical, e.g., number of numerical attributes; degree of class separation, e.g., Fisher's Linear Discriminant (Ho and Basu, 2002); or level of concept complexity, e.g., concept variation (Vilalta, 1999; Pérez and Rendell, 1996).

The proper identification of data properties is essential to map tasks to learning mechanisms.

Several approaches in meta-learning use families of meta-features as input to quantify task similarity. It is common to compute task similarity as the Euclidean distance between two meta-feature vectors. While this approach has shown to be effective in simple scenarios (Vanschoren, 2018), it exhibits clear limitations. First, selecting a subset of relevant meta-features is a non-trivial task. What criteria should we invoke to select or discard a family of meta-features? For example, statistical meta-features are not always intuitive and lack expressiveness. Previous work has shown how different datasets may share identical statistical properties but markedly different data distributions (Matejka and Fitzmaurice, 2017). Second, computing certain types of meta-features on large datasets is computationally costly. For example, topological meta-features perform multiple passes over the training dataset to compute a single figure of merit (Ho and Basu, 2002; Lorena et al., 2018). Ultimately, the selection of meta-features is an ad hoc process based on domain knowledge.

In this paper, we propose an approach that learns abstract meta-features from families of traditional meta-features using a deep neural network. We argue that traditional meta-features are not always capable of capturing crucial task characteristics. This can be attributed to inherent limitations such as being hand-crafted, and not being tuned to specific tasks (lacking in general applicability). Extracting meta-features on large datasets can quickly scale up computational costs and execution times. For example, complexity meta-features involve geometrical computations resulting in long computing times.

The main contribution of this paper is a novel method of inducing abstract meta-features as latent variables learned by a deep neural network; experimental results demonstrate the efficacy of the learned abstract meta-features in improving generalization performance estimation.

2. Related Work

Data characterization consists of extracting meaningful task properties. Simple, statistical, and information-theoretic meta-features can be straightforwardly extracted from datasets by capturing information concerning data dimensionality, distribution, and the amount of information present in the data. Model-based and landmarking meta-features characterize datasets indirectly by using the induced learning models; these meta-features comprise model properties and model performance (Rivolli et al., 2018).

Another family of meta-features is based on capturing the complexity of a learning task (Ho and Basu, 2002), and has been successfully used in different scenarios (Lorena et al., 2018). Most complexity measures are computationally expensive; one approach to reduce the associate computational cost is to train a metaregressor using traditional meta-features as input (Garcia et al., 2020). The metaregressor estimates complexity values for any dataset at low cost; the predicted values are called simulated meta-features.

Meta-features can be transformed to summarize the data, e.g., by reducing data dimensionality. Principal Component Analysis PCA (Hotelling, 1933) is the most straightforward and generic approach, even though it ignores the metatarget. For example, after running PCA, Bilalli et al. (2017) select the most relevant components (according to the cumulative total variance); next, a filter capturing the correlation with the metatarget identifies the most discriminating variables. Muñoz et al. (2018) propose a new dimensionality reduction technique suited for

data visualization; the most statistically significant traditional meta-features representing the hardness of the datasets are first identified, after which the metadataset is transformed into a 2-dimensional space to search for an optimal projection through an iterative optimization process; the new 2D space can project different model-performance footprints to investigate their strengths and weaknesses.

A different approach that has achieved popularity in recent years invokes Deep Neural Networks (DNNs). A strength behind DNNs is the capacity to learn data characteristics from a diverse and large amount of data. DNNs have had a strong impact in application areas such as speech recognition and image understanding (Deng and Yu, 2014). However, its use in meta-learning is still incipient and requires further investigation. A few studies explore deep learning for feature generation, representing different tasks and datasets in terms of embeddings generated by pre-trained DNNs (Achille et al., 2019). Gosztolya et al. (2017) solve different automatic speech recognition tasks through a two-step learning process: performing classification with DNNs, followed by the extraction of intrinsic features from the DNN output. In the second phase, features were used to improve model predictions. The same strategy is explored by Notley and Magdon-Ismail (2018), using DNNs to extract features from both images and numeric data.

Our hypothesis is that DNNs provide the means to extract *intrinsic* features from data. Our approach lies between transformation and deep learning methods since a DNN is induced from traditional meta-features and the knowledge captured by the hidden layers are used to generate abstract meta-features. In the process, traditional meta-features are transformed into latent variables used by the deep learning model to make predictions. Once extracted, abstract meta-features can be used by any meta-learning algorithm.

Our approach differs from traditional meta-learning settings. While model-based and landmarking induce a learning model for each dataset (metainstances) and extract features from this model, in the current approach, a model is induced using the metabase (all datasets). This is akin to the simulated meta-feature approach; however, instead of using the model's predicted values, we use the abstract representation of the deep neural network model to extract new meta-features.

3. Problem Statement

Given a classification task on a dataset $\mathcal{D}$ with n instances, our goal is to compute a meta-feature f on $\mathcal{D}$. A meta-feature is usually a hand-crafted characterization function capturing a specific property of interest on a given task. Meta-features are regarded as a form of meta-knowledge collected over a distribution of tasks to learn *how to learn*. Not all meta-features are informative, and some of them are very task specific. Learning relevant meta-features can prove useful in identifying hidden relationships across tasks, and is necessary to build accurate meta-learners.

Knowledge extracted across tasks, a.k.a. meta-knowledge, is key to the success of meta-learning by obviating learning from scratch on new tasks. By exploiting meta-knowledge, the metalearner can effectively construct an optimal solution based on past experience (Hospedales et al., 2020; Vanschoren, 2018). For example, a meta-learner can identify that a new task is similar to previous tasks and warm-start a similar model with near optimal hyperparameters. This avoids the –sometimes painstakingly– slow processes of error and trial in building a new model. Meta-knowledge can be understood as meta-features, model hyperparameters, performance mea-

sures, etc. In our work, meta-knowledge consists of meta-features and performance measures gathered from previous tasks.

We formally define the process of meta-feature extraction as a function f that receives as input a dataset $\mathcal{D}$, and returns as output a set of k values characterizing the dataset (Rivolli et al., 2018):

$$f(\mathcal{D}) = \sigma(m(\mathcal{D})), \tag{1}$$

where m is a characterization metric mapping $\mathcal{D} \rightarrow \mathbb{R}^{k'}$, k' is the original number of meta-features, and σ is a summarization function mapping $\mathbb{R}^{k'} \rightarrow \mathbb{R}^{k}$. The purpose of the summarization function is to reduce the size of the output to a fixed size. A dataset $\mathcal{D}$ is characterized by a meta-feature space $\mathcal{F}$. Our goal is to find a subset of meta-features, $F \subset \mathcal{F}$, capturing relevant task information.

4. Abstract Meta-features

We propose a novel approach to learning new abstract meta-features by constructing new representations from traditional meta-features using a deep neural network. A neural network –parameterized by $\mathbf{w}$– is a universal function estimator. The goal is to learn a function g to predict the true target y through an approximation $\hat{y}$, where $g(\mathbf{x}) = \hat{y}$. Training is achieved by computing gradients of the loss function with respect to the weights $\nabla_{\mathbf{w}} J(\mathbf{w})$ and then back-propagating the errors through the network to update the weights.

The architecture of a neural network is described by the number of neurons per layer and by the number of hidden layers ℓ_h^n, where h is the index for the hidden layer and n is the total number of neurons in that layer. Forward propagating input $\mathbf{x}$ through the network leads to a sequence of non-linear transformations; non-linearity is achieved via an activation function ϕ. Increasing the number of hidden layers and neurons allows the neural network to approximate highly non-linear functions. Each layer contains a learnt latent representation of the input data. The last hidden layer comprises the final learnt latent variables, $\{z_i\}$, where each latent variable z_i is a representation of the original input in an abstract space. The number of latent variables is user-defined by controlling the number of neurons in that layer. By training a deep neural network on the traditional meta-feature space $\mathcal{F}_t$, we can learn a new latent representation $\mathcal{F}_a$ (abstract meta-features). The resulting deep neural network serves as a feature extractor where the learnt latent space Z is extracted from the last hidden layer. This process is highlighted in Figure 1.

Hyper-parameter	**Value**
Learning rate	0.005
Hidden layers	5
Latent variables	16
Criterion	Smooth_{L_1}
Optimizer	Adam
Activation	ReLU

Table 1: AbstractNet hyper-parameters.

4.1. AbstractNet

Our methodology constructs a deep neural network (DNN) to act as a feature extractor on each pair of input and output meta-instances. After providing traditional meta-features as input and a performance measure of three different algorithms as target, we train our `AbstractNet` to learn a meaningful abstract representation of the meta-dataset. `AbstractNet` consists of 5 fully connected layers with sixty four neurons in each of the first four layers ($\ell_h^{64}, 1 \leq h \leq 4$), and a final latent layer with sixteen neurons (ℓ_5^{16}). Our target is a three dimensional output consisting of a performance estimation for each of three learning algorithms applied to a given task (represented using meta-features). Non-

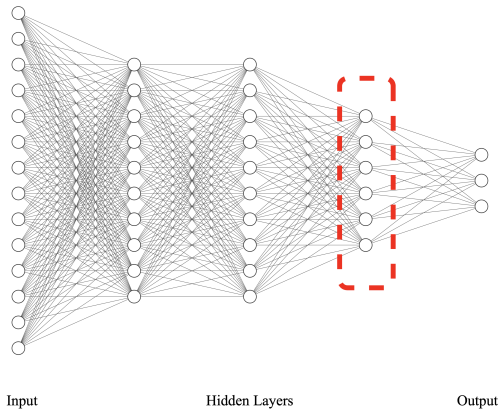

Figure 1: A deep neural network for learning abstract meta-features. The last hidden layer is used to extract the learnt latent variables $\{z_i\}$ to produce abstract meta-features.

linearity between layers is achieved via the ReLU activation function,

$$\phi(q) = \max(0, q), \tag{2}$$

where $q = \mathbf{wx} + \mathbf{b}$ is the linear transformation of the input. Variance is controlled via dropout; regularization is applied between layers two and four with probabilities $p = (0.1, 0.05)$ respectively. We selected the smooth L_1 loss function (Girshick, 2015) as our criterion function:

$$J(\hat{y}, y) = \frac{1}{n} \sum_i \nu(\hat{y}_i - y_i), \tag{3}$$

where the summation goes over all training examples, $\hat{y}$ and y are the estimated and true response values respectively, and ν is defined as

$$\nu(u) = \begin{cases} (0.5 * u^2)/\lambda & \text{if} \quad |u| < \lambda \\ |u| - 0.5 * \lambda & \text{otherwise.} \end{cases} \tag{4}$$

Parameter λ specifies the threshold that defines the step function. The smooth L_1 loss is less sensitive to outliers, has a smoother landscape, and prevents exploding gradients. The list of full neural network hyper-parameters is available in Table 1. Once the deep neural network is trained, we forward propagate the meta-dataset validation partition to extract the latent variables $\{z_i\}$ from the last hidden layer to induce our meta-models.

5. Experiments

In this section we describe the experimental design to induce abstract meta-features. Figure 2 provides an overview of the entire process, with three different phases corresponding to data characterization, meta-database construction, and induced meta-model evaluation. We explain these steps in detail next.

5.1. Datasets and Performance Evaluation

Phase 1 of our experiments includes dataset selection, performance evaluation, and meta-feature extraction. We collected a total of 517 classification datasets from OpenML (Vanschoren et al., 2014), a free scientific platform for standardization of experiments and sharing of empirical results. These datasets were selected following some criteria: the number of features does not exceed 500; there are no missing values; there are at least 2 classes; and the minority class must have at least 10 examples.

Next, we evaluated three common learning algorithms: Support Vector Machine (SVM), Random Forest (RF), and Multilayer Perceptron (MLP) on each dataset, recording their generalization performance in terms of the Area Under the ROC Curve (AUC) (Hand and Till, 2001).

In parallel, we extracted the traditional family of meta-features using the `PyMFE` library (Alcobaça et al., 2020) for these categories: general, statistical, info-theoretical, concept, model-based and landmarking. The

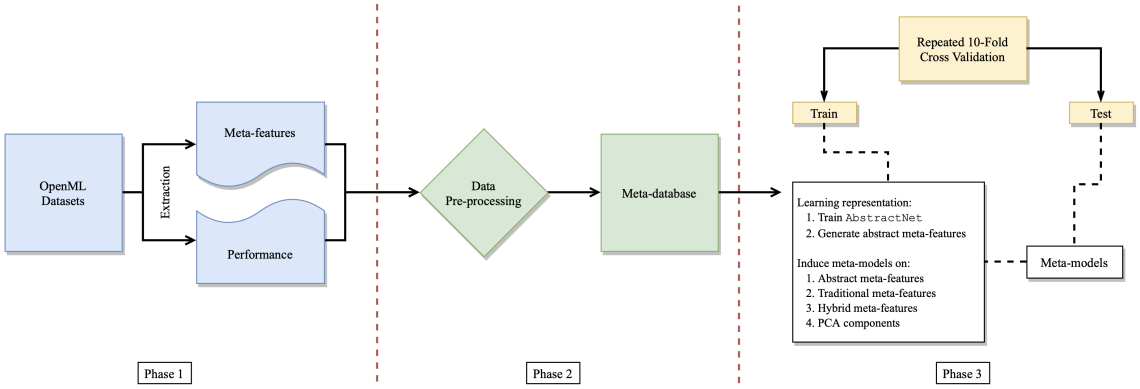

Figure 2: Sequence of steps to generate abstract meta-features and to assess meta-model performance.

functions *max, min* and *mean* were used to summarize multi-valued results.

5.2. Meta-database

In phase 2, we combined meta-features and performance values per dataset to construct a meta-database, $\mathcal{D}_{meta}$. A pre-processing step was also performed at the meta-database by removing meta-instances with more than 100 missing values, meta-features with more than 70% of missing values, constant meta-features, and highly correlated meta-features. The remaining missing values were imputed using a k-NN with $k = 10$. Lastly, we generated a meta-database with three different targets, one for each performance prediction problem (SVM, RF, MLP). The final meta-database has a total of 517 meta-examples (instances) and 265 meta-features (characteristics); it was used to train the deep neural network to induce the latent abstract meta-features.

5.3. Evaluation

In phase 3, we evaluated our approach under four different settings:

- **Abstract**: this strategy explores our approach alone, by constructing abstract meta-features through the `AbstractNet` and extracting a latent representation from the last hidden layer.
- **Traditional**: we induce meta-models on traditional meta-features only.
- **Hybrid**: we induce meta-models using a combination of traditional and abstract meta-features.
- **PCA**: as a baseline for comparison, PCA (Hotelling, 1933) is invoked to transform traditional meta-features through linear combinations. PCA also generates latent features keeping the components with 95% of the cumulative variance.

The hybrid approach is instrumental to assess feature relevance; it can show the value of abstract meta-features in predicting model performance over traditional meta-features.

We performed 10×10-fold cross-validation; in each iteration, nine folds are used to obtain the abstract meta-features (`AbstractNet`) and induce the different meta-models; and the remaining fold is used to validate the

Element	Feature	Value
Base level	Datasets	517
	Target Algos.	SVM, RF, MLP
	Performance	AUC
Meta-features	Abstract	16
	Traditional	154
	Hybrid	170
	PCA	60 (95% of var)
Meta-level	Tasks	3 (base level)
	Regressor	RF, DT, SVM
	Resampling	10 x 10-CV
	Performance	RMSE, R^2
Statistical Val.	Test	Hier. Bayesian

Table 2: Experimental and statistical validation settings across base and meta-levels.

meta-models. We employed three learning algorithms: Decision Trees, Random Forest, and Support Vector Machines as meta-inducers. Finally, we evaluated the performance of our models on the validation sets and report the Root Mean Squared Error (RMSE) and the coefficient of determination (R^2). We also repeated the experiments 10 times with different seeds to perform statistical validations using the Hierarchical Bayesian correlated t-test (Benavoli et al., 2017). Here, we compared the performance of the meta-inducers using distinct subsets of meta-features. The test evaluates in pairs, resulting in probabilities concerning which approach is better (left and right) for a particular evaluation measure. It also defines a region of equivalence (rope) that indicates the probability that the difference in performance is insignificant. The complete experimental methodology is shown in Table 2.

6. Results and Discussion

Experimental results are shown in Table 3. Results in bold stand for best results. As we can see, abstract meta-features achieve best RMSE and R^2 scores across all four settings. For instance, abstract meta-features perform vastly better than traditional meta-features with Decision Trees; the dimensionality of abstract meta-features is sixteen, while that of traditional meta-features is one hundred and fifty four. The reduced size of the space of abstract meta-features leads to high-generalization meta-models. Similar results are seen when different learning algorithms are used to induce meta-models. PCA transformation ranked lowest in terms of performance across the four settings; we hypothesize that PCA's inherent linear components are not expressive enough. `AbstractNet` is capable of learning highly abstract representations to capture complex relationships between meta-features and the target variable.

We can see that abstract meta-features closely follow the true AUC values across datasets, while traditional and PCA meta-features exhibit instability and poor performance. Variance along our performance metrics per learning algorithm is shown in Figures 3, 4. The hybrid approach allows us to combine traditional and abstract meta-features. By ranking the top fifteen most important features using Gini index from the Random Forest meta-model, seven out of sixteen learnt abstract meta-features ranked in the top fifteen features.

Table 4 shows probabilities obtained with the Bayesian Hierarchical t-test over different meta-databases and performance values. Abstract meta-features improved the use of traditional and PCA meta-features significantly, confirming the generalization ability of our approach. Unlike PCA, our deep neural network learnt a non-linear abstract transformation of traditional meta-features while increasing their predictive power.

7. Conclusions

Data-driven meta-learning (MtL) requires new forms of data characterization. Given

Inducer	Meta-features	R^2 Score			RMSE		
		SVM AUC	RF AUC	MLP AUC	SVM AUC	RF AUC	MLP AUC
DT	Abstract	**0.906 (0.072)**	**0.839 (0.17)**	**0.864 (0.129)**	**0.043 (0.014)**	**0.057 (0.032)**	**0.048 (0.022)**
	Traditional	0.554 (0.132)	0.510 (0.155)	0.561 (0.14)	0.101 (0.019)	0.114 (0.024)	0.097 (0.018)
	Hybrid	0.877 (0.078)	0.803 (0.163)	0.841 (0.128)	0.050 (0.015)	0.067 (0.031)	0.054 (0.022)
	PCA	0.424 (0.136)	0.384 (0.132)	0.418 (0.12)	0.121 (0.018)	0.131 (0.022)	0.117 (0.016)
RF	Abstract	0.925 (0.059)	**0.856 (0.17)**	**0.882 (0.125)**	0.038 (0.012)	**0.052 (0.033)**	**0.044 (0.023)**
	Traditional	0.761 (0.076)	0.715 (0.094)	0.752 (0.079)	0.071 (0.012)	0.081 (0.018)	0.070 (0.011)
	Hybrid	**0.928 (0.055)**	0.851 (0.167)	0.880 (0.123)	**0.037 (0.012)**	0.054 (0.032)	0.044 (0.023)
	PCA	0.688 (0.083)	0.641 (0.085)	0.685 (0.075)	0.081 (0.012)	0.092 (0.016)	0.080 (0.011)
SVM	Abstract	**0.889 (0.050)**	**0.862 (0.109)**	**0.873 (0.083)**	**0.062 (0.008)**	**0.076 (0.017)**	**0.068 (0.011)**
	Hybrid	0.829 (0.049)	0.774 (0.100)	0.805 (0.073)	0.070 (0.007)	0.083 (0.016)	0.074 (0.010)
	Traditional	0.685 (0.082)	0.603 (0.108)	0.644 (0.081)	0.088 (0.010)	0.102 (0.016)	0.092 (0.009)
	PCA	0.715 (0.067)	0.640 (0.105)	0.704 (0.075)	0.082 (0.009)	0.096 (0.016)	0.085 (0.009)

Table 3: Generalization performance of meta-models using different meta-databases containing all families of meta-features.

the difficulty inherent to the size of the meta-feature space, this paper explores a promising direction to solve the meta-feature selection problem by learning abstract meta-features via a deep neural network. We introduce and discuss data characterization as meta-knowledge. In order to optimally meta-learn over a distribution of tasks, the right form of meta-knowledge is required *a priori*. By defining our meta-objective as generalization performance, we construct a deep neural network to learn an abstract representation of traditional meta-features, i.e., we generate abstract meta-features as meta-knowledge to be used by our meta-models. We contend that abstract meta-features are more expressive and effectively capture hidden variable relationships.

Our experimental results demonstrate the efficacy of abstract meta-features as strong predictors of generalization performance, while reducing the size of the meta-feature space. Feature importance values computed on hybrid meta-features show that abstract meta-features frequently achieve top results. Our results show that PCA linear transformations are not as expressive as the non-linear transformations learnt by our deep neural network.

7.1. Limitations & Future work

There is no clear subset of traditional meta-features capable of capturing task properties well over highly diversified tasks. Just as deep learning usually outperforms hand-crafted features, learning meta-features with DNN can offer great insight into improving the data characterization process. Future research directions involve exploring abstract meta-features in a cost-effective fashion, and decomposing complex task characteristics as functions of simpler building properties. The goal is to identify fundamental characteristics that cover a broad spectrum of complex tasks.

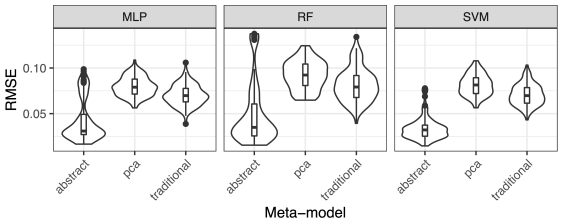

Figure 3: Violin plot of RMSE scores of meta-models on three approaches: abstract, traditional, and PCA.

Meta-databases	Measure	left	rope	right
Traditional × Abstract	RMSE	0.001	0.000	**0.999**
	R^2 Score	0.000	0.000	**1.000**
PCA × Abstract	RMSE	0.000	0.000	**1.000**
	R^2 Score	0.001	0.000	**0.999**
Traditional × PCA	RMSE	0.292	**0.696**	0.012
	R^2 Score	**0.911**	0.000	0.089

Table 4: Hierarchical Bayesian statistical probabilities by comparing pairs of meta-databases. The columns *left* and *right* indicate each meta-database's probability outperforming the other. The *rope* column indicates the probability of them being similar.

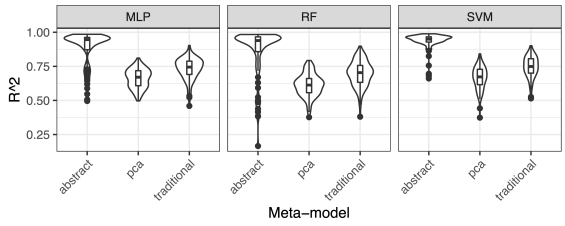

Figure 4: Violin plot of R^2 scores of meta-models on three approaches: abstract, traditional, and PCA.

References

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. *CoRR*, abs/1902.03545, 2019. URL http://arxiv.org/abs/1902.03545.

Edesio Alcobaça, Felipe Siqueira, Adriano Rivolli, Luís PF Garcia, Jefferson T Oliva, and André CPLF de Carvalho. Mfe: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111):1–5, 2020.

Alessio Benavoli, Giorgio Corani, Janez Demsar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *J. Mach. Learn. Res.*, 18:77:1–77:36, 2017.

Besim Bilalli, Alberto Abelló, and Tomàs Aluja-Banet. On the predictive power of meta-features in OpenML. *International Journal of Applied Mathematics and Computer Science*, 27(4):697 – 712, 2017.

Li Deng and Dong Yu. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3–4):197–387, 2014.

Luis Paulo Faina Garcia, Adriano Rivolli, Edesio Alcobaca, Ana Carolina Lorena, and Andre Carlos P. L. F. de Carvalho. Boosting meta-learning with simulated data complexity measures. *Intelligent Data Analysis*, in press, 2020.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

Gábor Gosztolya, Róbert Busa-Fekete, Tamás Grósz, and László Tóth. Dnn-based feature extraction and classifier combination for child-directed speech, cold and

snoring identification. pages 3522–3526, 08 2017. doi: 10.21437/Interspeech.2017-905.

David J. Hand and Robert J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Mach. Learn.*, 45(2):171–186, 2001. doi: 10.1023/A:1010920819831.

Tin K. Ho and Mitra Basu. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

Harold Hotelling. Analysis of a complex of statistical variables with principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

Ana Carolina Lorena, Luís P. F. Garcia, Jens Lehmann, Marcilio C. P. de Souto, and Tin Kam Ho. How complex is your classification problem? A survey on measuring classification complexity. *CoRR*, abs/1808.03591, 2018.

Justin Matejka and George Fitzmaurice. Same stats, different graphs: Generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 1290–1294, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346559. doi: 10.1145/3025453.3025912.

Mario A. Muñoz, Laura Villanova, Davaatseren Baatar, and Kate Smith-Miles. Instance spaces for machine learning classification. *Mach. Learn.*, 107(1):109–147, 2018. doi: 10.1007/s10994-017-5629-5.

Stephen Notley and Malik Magdon-Ismail. Examining the use of neural networks for feature extraction: A comparative analysis using deep learning, support vector machines, and k-nearest neighbor classifiers. *CoRR*, abs/1805.02294, 2018. URL http://arxiv.org/abs/1805.02294.

Eduardo Pérez and Larry A. Rendell. Learning despite concept variation by finding structure in attribute-based data. In Lorenza Saitta, editor, *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96), Bari, Italy, July 3-6, 1996*, pages 391–399. Morgan Kaufmann, 1996.

Adriano Rivolli, Luís PF Garcia, Carlos Soares, Joaquin Vanschoren, and André CPLF de Carvalho. Towards reproducible empirical research in meta-learning. *arXiv preprint arXiv:1808.10406*, pages 32–52, 2018.

Jurgen Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1(2), 1987.

Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15(2):49–60, 2014. ISSN 1931-0145.

Ricardo Vilalta. Understanding accuracy performance through concept characterization and algorithm analysis. In

Workshop on Recent Advances in Meta-Learning and Future Work, 16th International Conference on Machine Learning, pages 3–9, 1999.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.