Dataset Generation for Meta-Learning

Matthias Reif, Faisal Shafait, and Andreas Dengel

German Research Center for Artificial Intelligence,
Trippstadter Str. 122, 67663 Kaiserslautern, Germany
{matthias.reif,faisal.shafait,andreas.dengel}@dfki.de

Abstract. Meta-learning tries to improve the learning process by using knowledge about already completed learning tasks. Therefore, features of dataset, so-called meta-features, are used to represent datasets. These meta-features are used to create a model of the learning process. In order to make this model more predictive, sufficient training samples and, thereby, sufficient datasets are required.
In this paper, we present a novel data-generator that is able to create datasets with specified meta-features, e.g., it is possible to create datasets with specific mean kurtosis and skewness. The publicly available data-generator[1] uses a genetic approach and is able to incorporate arbitrary meta-features.

1 Introduction

Meta-learning – or "learning to learn" – uses previously gathered knowledge about a learning task in order to provide an automatic selection, recommendation, or support for a future task. One intensively investigated meta-learning field is algorithm or model selection: for a new dataset, one or more suitable algorithms are selected or recommended based on the knowledge about the suitability of algorithms on other datasets. Common approaches for making recommendations use classification [1], regression [10], or ranking [2]. A different meta-learning task supports parameter optimization of learning algorithms [9].

All these approaches use characteristics or properties of datasets as foundation for the actual meta-learning. These properties of datasets are typically called meta-features. Different groups of meta-features have been proposed in the literature: simple meta-features [5] are directly extractable from the dataset such as the number of features or the number of samples. Statistical meta-features [4] use statistical measures of the probability distributions such as the kurtosis or the skewness. Information-theoretic meta-features [3] are based on the entropy such as the joint entropy between feature and class label or the mutual-information. Model-based and landmarking meta-features build a model from the dataset. While landmarking [7] uses the performance achieved by this model as meta-feature of the dataset, model-based meta-features [6] are diverse properties of this model. Typical, model-based meta-features are properties of a decision tree such as its width or depth.

[1] `http://www.dfki.uni-kl.de/~reif/generator/`

The meta-features construct the feature space for the meta-learning. As for any pattern recognition method, it is problematic if this space is high-dimensional and only sparsely populated. Hence, a sufficient number of datasets is required. Since real-world datasets are rare and hard to obtain, artificially created datasets might solve the issue. In this paper, we present a novel data-generator that is specially designed to support the investigation and development of meta-learning approaches. It is able to generate datasets with user-defined values of the meta-features, e.g. with a certain mean kurtosis and a certain Naive Bayes accuracy. Being able to generate datasets with specific meta-features, meta-learning can be supported in two different ways:

Sparse Feature Space: Typically, many meta-features are extracted and the high-dimensional feature space is only sparely populated. Randomly generated datasets can not ensure that they are sufficiently distributed over the meta-feature space. Using the presented generator, the meta-feature space can be filled in a more controlled way and discovered "empty areas" can be populated.

Investigation of Meta-features: Most of the presented meta-features have not been thoroughly investigated according to their descriptive power for a certain meta-learning task. Generating datasets with specific values of meta-features allow more controlled experiments that might lead to conclusions about the usefulness of particular meta-features.

2 Design

We treat the data generation as an optimization problem. A candidate solution is a specific dataset defined by its data points. Since we want the dataset to fulfill multiple meta-feature requirements, this is a multi-objective optimization problem. Therefore, we constructed a single aggregate objective function using a weighted sum over the meta-feature vector x of size n, the vector of desired values y, and the vector of the according weights w:

$$f(x) = \sum_{i=1}^{n} w_i \cdot |x_i - y_i|. \qquad (1)$$

This objective function measures the difference of the measured meta-features of a dataset and the desired values. For solving this minimization problem, we use a genetic algorithm, that mutates datasets by shifting data points and recombines two datasets by swapping fractions of them. Since we know that the best possible value of the objective function is zero, we can stop the genetic algorithm if it achieves a certain interval.

The presented generator is able to incorporate a variable set of arbitrary meta-features. The user is able to build a custom set of meta-features by simply providing the functions computing the meta-features. However, the number of samples of each class and the number of features are not optimized but fixed in advance and used for creating the random start population of the genetic algorithm. For each feature and class, values are sampled either from a normal or a uniform distribution using random parameters (mean and variance or minimum and maximum, respectively). Since features are typically normalized anyway, we fixed the range for the parameters of the probability distributions.

3 Implementation

We implemented the data generator in Python because it is easy to use and becomes more and more popular in scientific computing. As implementation for the genetic algorithm, we used DEAP [8]. This framework already provides different variants for the components of a genetic algorithm, such as different selection and cross-over schemes. Additionally, parallel and even distributed computation is easily possible as well. The data generator uses Gaussian mutation and two-point cross-over.

Adding new meta-feature is done by calling `add_measure` with at least two parameters: a tupel of functions and the desired value. The functions are successively applied, e.g., the tuple `(kurtosis, mean)` will compute the kurtosis of all features first, and, then, calculate the mean. This avoids the definition of additional aggregation functions and enables a caching mechanism to save computation time. If, e.g., the minimum and the maximum of the kurtosis are added, the kurtosis itself is computed only once. The remaining parameters of `add_measure` are optional: The third parameter defines if the measure also requires the label as input. The last parameter defines the weight of the measure for the objective function. By default, a value of 1.0 is used.

4 Examples

For illustration of the presented approach, we applied it for the generation of datasets with the following properties:

#classes:	2	mean skew:	0.0	naive bayes:	0.8
#features:	2	min skew:	-0.8	nearest neighbor:	0.9
#samples:	400	max skew:	0.8		

We used a weight of 2.0 for the naive bayes and the nearest neighbor accuracies and 1.0 for the remaining properties. The genetic algorithm uses a population size of 100 and was stopped if the error was below 0.01.

Figure 1 shows the results of four different runs of the data generator. For each run, the final dataset as well as the error of the five optimized properties (#classes, #features, and #samples are fixed) over the generations are plotted. It is notable that different runs generate quite different datasets although all datasets have the same specified characteristics.

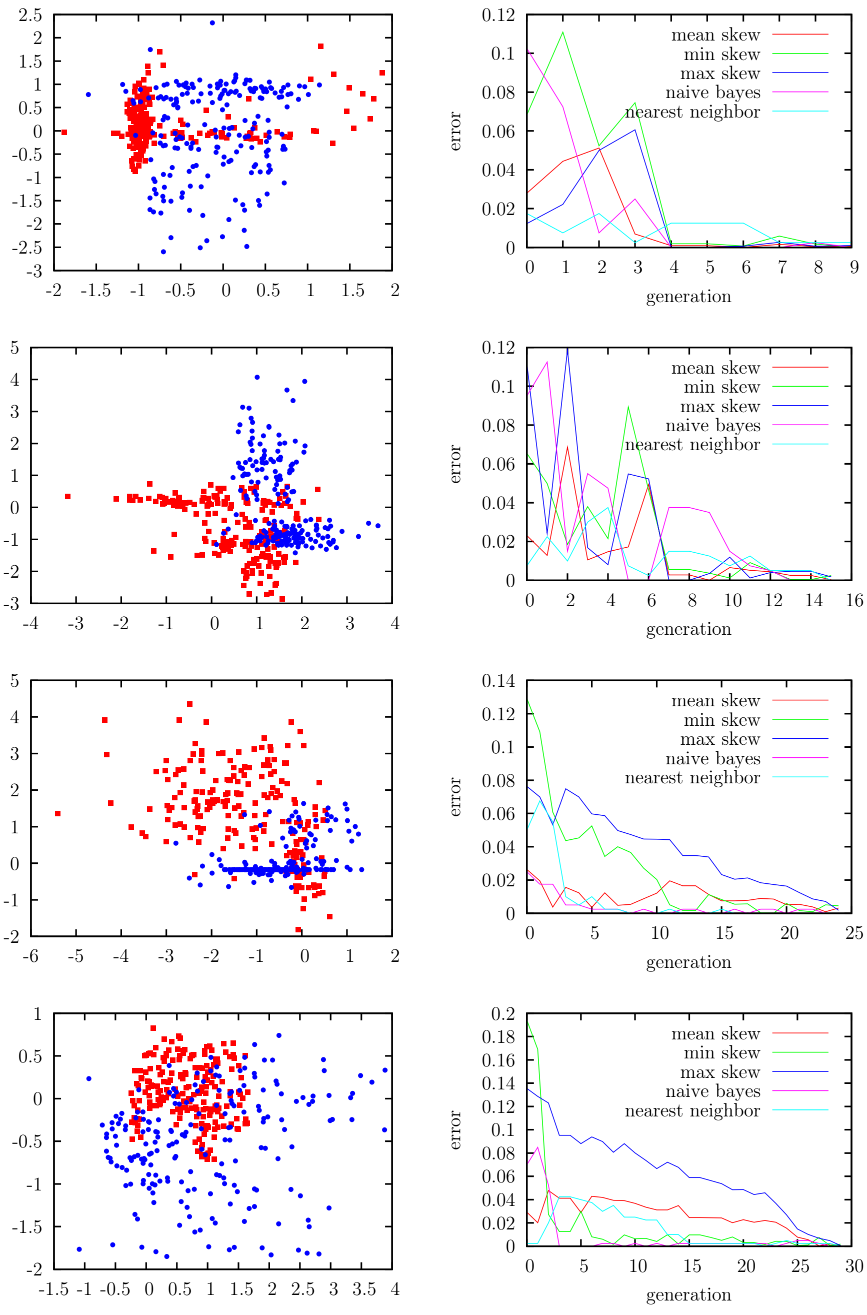

Fig. 1. Generated datasets and multi-criteria error over generations for the same datasets characteristics.

5 Conclusion

We presented a novel data generator for creating datasets with specific characteristics that can be used for the development and evaluation of meta-learning systems. Its Python implementation is open-source and publicly available. The current version is limited to numerical features and classification datasets.

References

1. Ali, S., Smith, K.A.: On learning algorithm selection for classification. Applied Soft Computing 6, 119–138 (January 2006)
2. Brazdil, P.B., Soares, C.: Zoomed ranking: Selection of classification algorithms based on relevant performance information. In: Proc. of Principles of Data Mining and Knowledge Discovery PKDD. pp. 126–135 (2000)
3. Castiello, C., Castellano, G., Fanelli, A.M.: Meta-data: Characterization of input features for meta-learning. In: Torra, V., Narukawa, Y., Miyamoto, S. (eds.) Modeling Decisions for Artificial Intelligence, Lecture Notes in Computer Science, vol. 3558, pp. 295–304 (2005)
4. Engels, R., Theusinger, C.: Using a data metric for preprocessing advice for data mining applications. In: Proc. of the European Conf. on Artificial Intelligence. pp. 430–434 (1998)
5. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood (1994)
6. Peng, Y., Flach, P., Soares, C., Brazdil, P.: Improved dataset characterisation for meta-learning. In: Lange, S., Satoh, K., Smith, C. (eds.) Discovery Science, Lecture Notes in Computer Science, vol. 2534, pp. 193–208 (2002)
7. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: Proc. of the 17th Int. Conf. on Machine Learning. pp. 743–750 (2000)
8. Rainville, F.M.D., Fortin, F.A., Gardner, M.A., Parizeau, M., Gagné, C.: Deap: A python framework for evolutionary algorithms. In: EvoSoft Workshop, Companion proc. of the Genetic and Evolutionary Computation Conference (GECCO 2012) (July 2012)
9. Reif, M., Shafait, F., Dengel, A.: Meta-learning for evolutionary parameter optimization of classifiers. Machine Learning 87(3), 357–380 (2012)
10. Reif, M., Shafait, F., Goldstein, M., Breuel, T., Dengel, A.: Automatic classifier selection for non-experts. Pattern Analysis and Applications (2012), 10.1007/s10044-012-0280-z