# NDSE: Instance Generation for Classification by Given Meta-Feature Description

Alexey Zabashta and Andrey Filchenkov

ITMO University, Computer Technology Lab,
Kronverksky pr. 49, 197101 St.Petersburg, Russia
{azabashta,afilchenkov}@corp.ifmo.ru

**Introduction.** Meta-features [1] are variables reflecting various properties of datasets. In machine learning, they are typically used for two purposes. The first one is predicting algorithm performance on new datasets never seen before, that is meta-learning [2]. The second one is analyzing algorithms and finding their weaknesses [3]. Thus, meta-features allow obtaining vector representations of datasets and considering them as points in the corresponding space. The key problem is that such a space will be extremely sparse if we use only real-world datasets. This problem was recognized by researchers and several methods exists for generating new datasets, e.g. [4]. However, most of them generate datasets in an uncontrolled manner, without any guesses on how biased the resulting set of datasets may be in terms of meta-feature space, which does not allow using them for algorithm analysis. This paper is devoted to solve the dataset generation problem in a guided way. The method we propose generates a dataset such that its meta-feature vector is as close as possible to a target meta-feature vector.

**Related works.** We found only two papers devoted to the problem we solve. In both of them, the authors solve this problem by reducing it to a minimization problem, in which the target function is the distance between target and resulting datasets in a meta-feature space. To solve this problem, they use different evolutionary techniques. In [5], the authors explicitly represent datasets for the genetic algorithm as vectors in $\#attributes \times \#objects$ dimensional space. We will refer to the search in this space as **Vect**. In [6], the authors use the CMA-ES algorithm to optimize parameters of the Gaussian mixture distribution, which they used for sampling objects the of resulting dataset. We will refer to it as **GMM**.

**NDSE Method.** The NDSE method (**N**atural **D**ata**S**ets **E**volution), that we present earlier [7], shares the advantages of both related approaches. This method also uses evolutionary algorithms and its operators have direct access to datasets, not their intermediate representations.

NDSE is based on using operators of adding and removing attributes or objects from a dataset. These operators are equivalent to feature extraction, feature selection, oversampling and undersampling, respectively. We use the following implementation of these operators: random removing of attributes and objects from a dataset, copying of random subsets of objects from the same class for adding new object to dataset, applying random functions, which depends on attributes, class label and random noise, for adding a new attribute to a dataset.

In order to apply evolutionary algorithms, mutation and crossover operators are required. Mutation operator uses the operators of adding and removing to change the number of attributes or objects in a dataset. Crossover operator merges two datasets into a single big dataset and then splits it by attributes into two resulting datasets.

NDSE works as follows. It receives a list of meta-features (functions of dataset) and a target meta-feature vector as input. Vanilla version of this method (which we will refer to as NDSE) starts with an empty dataset, while an improved version (which we will refer to as NDSE') uses real-world datasets as an initial population. During its runtime, the method follows a specific evolutionary algorithm, which is the hyperparameter of this method, and which exploits suggested mutation and crossover operators. Fitness function is the distance between the target meta-feature vector and meta-feature vector of a generated dataset.

**Experiments.** We conduct two experiments. In both of them, the target vector is meta-features of a real dataset for binary classification from OpenML[1]. We excluded a target dataset from the initial population for the methods that use them in that way. We use different evolutionary strategies implemented in jMetal[2] library. We compare **NDSE** and **NDSE**' with **Vect**, **Vect**' (uses real-world datasets as an initial population) and **GMM**. The error is the resulting fitness-function value, that is the normalized Euclidean distance between the target and resulting datasets: $E(\theta) = \sum_i((f_i - \theta_i)/\sigma_i)^2$, where $\theta_i$ and $f_i$ are $i$-th meta-feature values of the target and resulting datasets correspondingly, and $\sigma_i$ is its standard deviation for real-world datasets. We use 29 meta-features, such as general information, informational and statistical metrics, and characteristic of decision tree. Each run in the Short-Run experiment is limited by the 2000 generated datasets, while this limitation is $10^5$ for the Long-Run experiment. The results of the experiments are shown in Table 1.

**Table 1.** Experimental results. The average error over the 298 target datasets by different evolutionary strategies for the Short-Run experiment and minimal average error at the three target datasets for the Long-Run experiment.

| | Short-Run (evolutionary algorithms) | | | | | | Long-Run (datasets) | | |
|---|---|---|---|---|---|---|---|---|---|
| | CMAES | MOCell | NSGAII | RAND | SMSOA | SPEA2 | fertility | vote | Australian |
| GMM | 4.51 | 6.48 | 6.08 | 6.76 | 6.02 | 6.08 | $2.39\cdot10^{-5}$ | 0.91 | 6.54 |
| Vect | — | 13.29 | 13.66 | 21.57 | 12.55 | 13.06 | $1.67\cdot10^{-9}$ | 5.50 | 12.29 |
| NDSE | — | 2.61 | 2.83 | 3.89 | 2.90 | 2.80 | $3.13\cdot10^{-5}$ | 0.07 | 0.51 |
| Vect' | — | 2.04 | 1.64 | 2.63 | 1.88 | 1.94 | $2.38\cdot10^{-9}$ | 0.69 | 2.21 |
| NDSE' | — | 1.84 | 1.75 | 6.95 | 1.74 | **1.60** | $1.18\cdot10^{-4}$ | 0.11 | 0.54 |

As it can be seen, the proposed methods are superior to the baselines. Usage of real-world datasets as an initial population is always better than usage of anything else. The best result is shown by NDSE' with SPEA2.

[1] OpenML.org

[2] jmetal.github.io/jMetal

**Conclusion.** In this paper, we suggested a method for generating datasets given their meta-feature description. The method outperformed the baselines.

**Acknowledgments.** This work was financially supported by The Russian Science Foundation, Agreement 17-71-30029 and the Russian Foundation for Basic Research, Grant 16-37-60115 mol_a_dk.

## References

1. Filchenkov, A., Pendryak, A.: Datasets meta-feature description for recommending feature selection algorithm. In: AINL-ISMW FRUCT. pp. 11–18 (2015)
2. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: Applications to data mining. Springer Science & Business Media (2008)
3. Smith-Miles, K., Baatar, D., Wreford, B., Lewis, R.: Towards objective measures of algorithm performance across instance space. Computers & Operations Research 45, 12–24 (2014)
4. Soares, C.: Uci++: Improved support for algorithm selection using datasetoids. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 499–506. Springer (2009)
5. Reif, M., Shafait, F., Dengel, A.: Dataset generation for meta-learning. Poster and Demo Track of the 35th German Conference on Artificial Intelligence (KI-2012) pp. 69–73 (2012)
6. Muñoz, M.A., Smith-Miles, K.: Generating custom classification datasets by targeting the instance space. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 1582–1588. GECCO '17, ACM, New York (2017)
7. Zabashta, A., Filchenkov, A.: NDSE: Method for classification instance generation given meta-feature description. In: AutoML Workshop, International Conference on Machine Learning (2017)