Linear Matrix Factorization Embeddings for Single-objective Optimization Landscapes

Tome Eftimov, Gorjan Popovski, Quentin Renau, Peter Korošec, Carola Doerr

Linear Matrix Factorization Embeddings for Single-objective Optimization Landscapes

1st Tome Eftimov
Computer Systems Department
Jožef Stefan Institute
Ljubljana, Slovenia
tome.eftimov@ijs.si

2nd Gorjan Popovski
Computer Systems Department
Jožef Stefan Institute
Ljubljana, Slovenia
gorjan.popovski@ijs.si

3rd Quentin Renau
Thales and École Polytechnique
Institut Polytechnique de Paris
Palaiseau, France
quentin.renau@thalesgroup.com

4th Peter Korošec
Computer Systems Department
Jožef Stefan Institute
Ljubljana, Slovenia
peter.korosec@ijs.si

5th Carola Doerr
LIP6
Sorbonne Universit, CNRS
Paris, France
carola.doerr@lip6.fr

Abstract—Automated per-instance algorithm selection and configuration have shown promising performances for a number of classic optimization problems, including satisfiability, AI planning, and TSP. The techniques often rely on a set of features that measure some characteristics of the problem instance at hand. In the context of black-box optimization, these features have to be derived from a set of $(x, f(x))$ samples. A number of different features have been proposed in the literature, measuring, for example, the modality, the separability, or the ruggedness of the instance at hand. Several of the commonly used features, however, are highly correlated. While state-of-the-art machine learning techniques can routinely filter such correlations, they hinder explainability of the derived algorithm design techniques.

We therefore propose in this work to pre-process the measured (raw) landscape features through representation learning. More precisely, we show that a linear dimensionality reduction via matrix factorization significantly contributes towards a better detection of correlation between different problem instances – a key prerequisite for successful automated algorithm design.

Index Terms—exploratory landscape analysis, automated algorithm design, representation learning, benchmarking

I. INTRODUCTION

Supervised machine learning techniques can support the user in selecting a best suitable optimization heuristics for a given problem instance by providing automated data-driven recommendations [1], [2]. Such machine-trained approaches require meaningful features that quantify different aspects of the problem instance at hand.

We are concerned in this work with *black-box optimization,* where the problem instances are not accessible other than by evaluating the quality of solution candidates (e.g., through experiments or computer simulations). In this context, the features (apart from generic ones that only describe the

decision space itself but not its relationship with the objective values) have to be derived from sampling this decision space. Extracting meaningful features from these samples is studied under the notion of *exploratory landscape analysis* [3]. Especially for the case of numerical black-box optimization, a large number of features has been suggested over the last decades [4], [5]. Each of these features measures a different characteristic of the problem instance. Building upon these features, several works have demonstrated the feasibility of automated algorithm selection [2], [6] and configuration [7] in the black-box scenario. However, it is well known that several of the existing feature sets contain highly correlated features. In terms of performance, this poses problems only to naïve learning techniques, since state-of-the-art ML can handle such correlations in an automated way. However, when the focus is less on achieving peak performance but rather on *understanding* the underlying recommendations, then these correlations pose a problem, as they hinder the explainability of the automated recommendations. It is therefore common practice to invest substantial computing resources in a proper feature selection [8].

We propose in this work an alternative way to deal with redundant information in the feature sets. We demonstrate that representation learning via matrix factorization (singular value decomposition (SVD) [9]) can effectively detect correlated features and can properly normalize their importance via so-called fingerprint embeddings. With this decomposition, new feature vectors can easily be mapped into comparable *fingerprints* of much higher expressiveness than the original feature value.

We demonstrate the effectiveness of the representation learning approach by applying it to detect correlations between different instances obtained from the 24 functions of the BBOB benchmark set (black-box optimization benchmark [10]), a standard benchmark set of numerical optimization problems. While it is difficult to detect which instances

are obtained from the same problem when using the original feature vectors (either normalized or non-normalized features values), we obtain high accuracy when computing instance correlation based on the fingerprint representations. We also perform a stratified 5-fold cross classification of the instances, and observe that classifiers trained with the embeddings are more robust than those trained with the original features values.

Availability of Code and Data: The code and data of this project are available at [11], feature values at [12].

II. BACKGROUND

We provide some basic background on Exploratory Landscape Analysis and Representation Learning.

A. Exploratory Landscape Analysis

Exploratory landscape analysis (ELA) was introduced in [3] to support the user of black-box optimization heuristics through machine-trained recommendations for the algorithm design that best suits the problem at hand. ELA builds on fitness landscape analysis [13], [14] and translates it into a setting in which the underlying problem is only available as a query-able black-box and not – as required in classical fitness landscape analysis – as an explicit model $f : S \to \mathbb{R}$. The key idea of ELA is to describe the characteristics of an optimization problem through a set of features, where each feature measures a different aspect of the problem. As the problems are black-box, the feature values have to be estimated from an (ideally small) set of samples, where each sample is an $(x, f(x))$ pair, which requires evaluation of the solution candidate x. A number of ELA features can be conveniently computed by the R-package *flacco* [5], which comprises up to 343 feature values, which are grouped into 17 feature sets [2].

B. Representation Learning

Representation learning aims at mapping data into new embeddings which store the essential information of the original data set in a more appropriate and often more dense form, and this via automated mappings [15]. The representations can catch, for example, redundancies in the data, or effects that stem from a misfortunate scaling of the data. They are also successfully used to reduce the dimension of the data, via automatically detecting correlations. Representation learning has its most important applications in machine learning, where bias and redundancies in data can have severe effects on performance.

Among the most popular representation learning approaches are unsupervised techniques, which require only limited human domain knowledge and interaction. Matrix factorization is one of these unsupervised techniques. It has proven to be effective in a broad range of applications, see [16]–[18] for examples in feature selection.

III. METHODOLOGY

To describe the matrix factorization approach in our context of ELA applications, let us assume that we have n benchmark problems, for each of which we consider m different instances. Let us further assume that for each instance we calculate k features via the ELA sampling approach described above. We organize the obtained data in an $nm \times k$ matrix X with entries that we refer to as $X_{i,j;t}$. That is, for $i \in [1..n]$, $j \in [1..m]$, and $t \in [1..k]$, the value $X_{i,j;t}$ corresponds to the t-th feature value obtained for the j-th instance of the i-th benchmark problem, and the row $\mathbf{p_{i,j}}^T = [X_{i,j;1}, X_{i,j;2}, \cdots, X_{i,j;k}]$ is the vector of feature values for this instance.

As mentioned above, several standard ELA features are highly correlated, which may bias the interpretation of the feature vectors, by giving stronger weight to highly redundant information. To address this issue, we propose to learn a vector representation (i.e., an *embedding*) for the feature vectors by applying *singular value decomposition* (SVD) [9]. That is, we decompose the matrix X into $X = U\Sigma V^T$, where U and V are are orthogonal matrices, Σ is a diagonal matrix with singular values on the diagonal, and the column vectors of U and V correspond to the left and right singular vectors, respectively. The benchmark problem instance embedding can be conveniently calculated as: $\hat{\mathbf{p}}_{i,j} = \Sigma^{-1}V^T\mathbf{p_{i,j}}$.

By applying SVD, we map ("embed") our instance vectors $\mathbf{p_{i,j}}^T$ to another vector space in which the new vectors are presented in different uncorrelated dimensions of the data. The space of the embedded vectors is defined by the singular value components, where each instance embedding is a combination of the contribution of each singular value. Note also that SVD performs a *linear dimensionality reduction*. When the matrix X is a square matrix, the learned embeddings are unit-norm vectors, whereas the norm of the learned embeddings are less than one otherwise. This follows from the fact that the column vectors of U are always unit-norm vectors.

In most cases, the matrix X is a full-rank matrix so that, necessarily, the embedded vector space has the same dimension as the matrix. However, a low-rank approximation X_r for the matrix X can also be calculated by this approach. To this end, the r largest singular values and only the first r columns of U and V are used to express $X_r = U_r\Sigma_rV_r^T$. The choice of r is basically a trade-off between time and accuracy, as smaller r can save computing time at the risk of an increased approximation accuracy error. We will analyse the impact of the size of r on our ELA data in Section IV-A. Approaches to estimate the number of useful components are the *non-graphical Cattells Scree test* [19] and so-called *parallel analysis* [20].

IV. EXPERIMENTS

Our test-bed are the 24 benchmark problems from the BBOB benchmark set [21] of the COCO (Comparing Continuous Optimizers) platform [10]. For each problem, we consider the first five instances of the functions in 5D. These instances all stem from the same base problem, and are obtained by translations in search and objective space. The BBOB benchmark set is a well established environment to benchmark derivative-free numerical optimization techniques

and forms the basis for discussions in the BBOB workshop series at the annual ACM GECCO conference.

From the *flacco* software mentioned in Section II-A we use 38 features, which stem from the following five sets: dispersion (disp), information content (ic), nearest better clustering (nbc), meta model (ela_meta), and y-distribution (ela_distr). These 38 features have been chosen because they do not require adaptive sampling, nor have they been dismissed as inappropriate in previous works. For example, it has been shown that some of the *flacco* features exhibit low robustness to the random sampling [22], [23] and to instance transformations such as translation and scaling [24].

To address a potential lack of feature value robustness, we use median values that are computed from 100 independent approximations. Each approximation is based on 1250 samples, which are generated using the Python package *sobol_seq* (version 0.1.2), with randomly chosen initial seeds. This package computes so-called Sobol' sequences [25], a construction that is known to provide samples of small star discrepancy [26]. We note that different sampling strategies are known to provide different feature values [27]. We have therefore repeated some of the analyses presented in this work with uniformly chosen samples and obtained results that are comparable to those reported in the subsequent sections. We omit a detailed discussion of these results for clarity of presentation.

A. Homogeneity of the BBOB Instances

To explore the homogeneity of the BBOB instances, we first study the Pearson correlation of the vector representations of the 38 (non-normalized) features mentioned in Section II-A. The Pearson correlation was chosen since SVD performs linear dimensionality reduction and the Pearson correlation is a measure for the linear correlation between two vectors. We plot these pairwise correlations of the $5 \cdot 24$ problem instances in a heatmap, which can be found in Figure 1a. This heatmap lists all BBOB instances in the sequence of the problems, i.e., we start with instance 1 of function f_1 on the lower left corner, which is followed by instance 2 of function 1 to the right and to the top, and we continue this way up to instance 5 of problem f_{24} in the rightmost column and in the uppermost row. We clearly observe that almost all instances show high correlation, either positive or negative. Only three exceptions to this picture exist, and those are the second instance of f_{14}, the first instance of f_{17}, and the fourth instance of f_{19}. Their feature vectors show no linear correlation with the feature vectors of the other problem instances, and this is not the case even for the other instances of the same problem. If asked to interpret this data, one would likely be tempted to conclude that there is no reason to select and use most of the instances, as they hardly show sufficient complementarity. Even when replacing the original feature values by normalized values, the result does not change much in term of outcomes from the correlation analysis, because the normalization alone does only reduce direction not the intensity of the linear correlation. Using the normalized original features, for most of the instances, we are still not able to distinguish between instances from the same problem.

A simple analysis helps to explain the effects that lead to the high pairwise correlation of almost all 120 tested BBOB functions. We simply transpose the feature matrix and compute the Pearson correlation between the landscape features – as opposed to computing correlation between the instances. Figure 1b shows the result of this experiment, and we clearly observe a very high correlation between the features belonging to the same feature set. For example, the dispersion features on the lower left corner show high positive correlation between all feature pairs. Since the sizes of the feature sets are not identical, more weight is given to the larger ones when computing the correlation between the instances. This explains, to a large extent, the high correlation observed in Figure 1a, as we shall see in the next paragraphs.
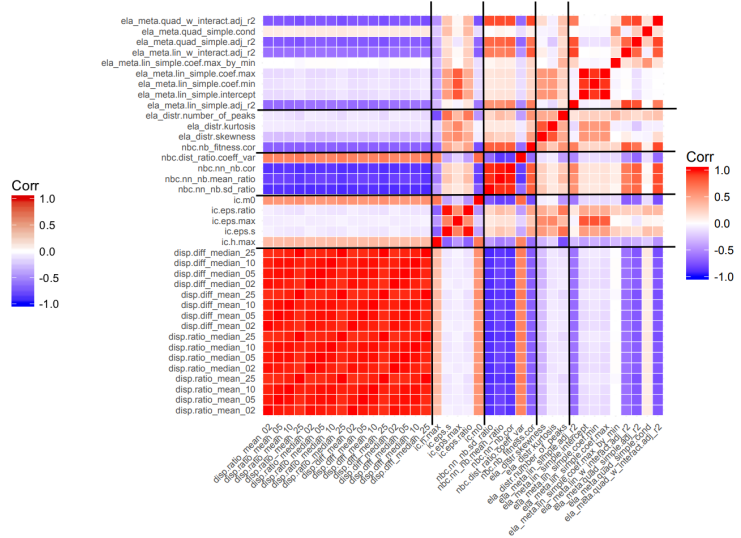
The high correlation between the feature values observed in Figure 1b indicates a strong need for representation learning. We apply the suggested matrix factorization approach described in Section III. That is, we first compute the embeddings ("fingerprints") for each benchmark problem instance. Since, we are interested in high accuracy (i.e., low approximation error from the decomposition) and since computational efficiency is not a problem when dealing with such (comparatively) small matrix sizes (120×38), we have used all 38 singular values to calculate the embeddings.

The pairwise Pearson correlation between the embedded feature vectors of the BBOB instances are plotted in Figure 2. The diagonal pattern in this figure corresponds to a positive pairwise correlation of instances that belong to the same problem, in almost all cases. High positive correlations are detected within instances of the following problems: f_7, f_{16}, f_{21}, f_{22}, f_{23}, and f_{24}. In addition, all instances from f_{23} are positively correlated with all instances from f_{24} and f_{16}. The third and fourth instances from the f_{11} are more correlated than their correlation with the other three instances from the same problem. There is no obvious linear correlation between instances of a basic problem and its rotated variant (i.e., f_3 and f_{15}, and f_8 and f_9, respectively). Though many cases with missing strong linear correlation between different instances of the same problem might look strange at first, it is in fact what one should expect when considering that, by the way instances are generated in BBOB, the different instances are essentially forcing the sampling to focus on different parts of the same base problem – a topic that we will return to in Figure 3b.

Comparing the correlations between the BBOB instances in the original and in the embedded space (Figures 1a and 2), it is obvious that using the embeddings we see the positive correlations between the instances that belong to the same problem. We also detect groups of instances that are correlated with instances from other problems, which was not possible in the original feature space. These results can be further used in other applications. For example, working on automated algorithm design, it will be enough to take only one instance from the f_{23} problem, since all of them are highly positive correlated, while for the f_{11} problem all five instance should be
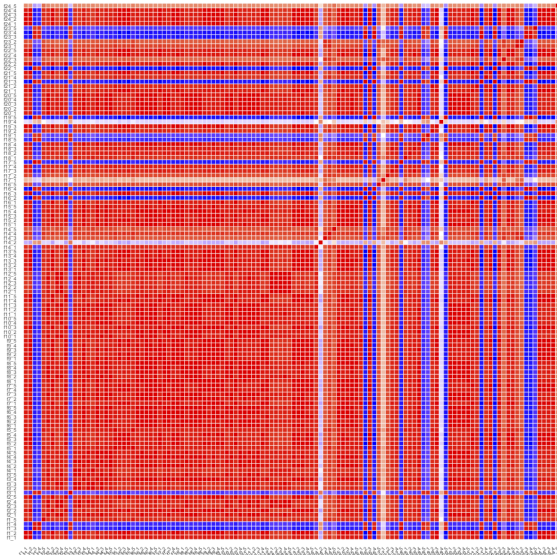
(a) Correlations between BBOB instances using their representations with 38 landscape features in the original space

(b) Correlations between landscape features using their representations, which are the values obtained for each BBOB instance.
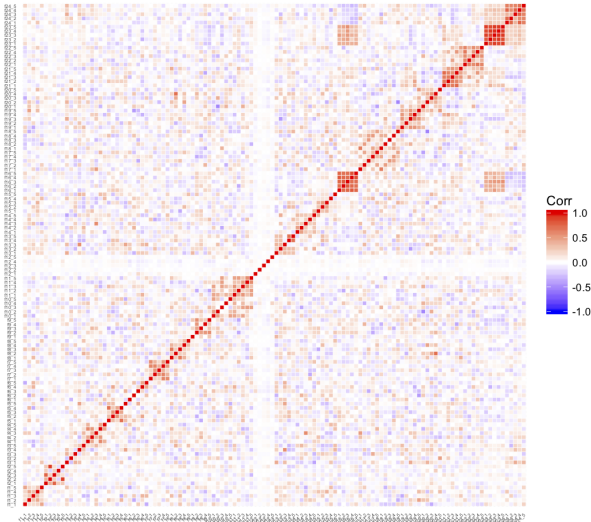
Fig. 1: Pearson correlation analysis.

Fig. 2: Pearson correlation between BBOB instances using the embeddings calculated with 38 singular values.

involved, since there is only weak positive correlation between them. The correlation threshold for selecting the instances should be done by a subject-matter expert and depends on the application. Another striking pattern in Figure 2 is the white cross at the instances of problem f_{12}. These white tiles indicate that there is no linear correlation between the five instances of this problem, nor with any of the other 115 considered BBOB instances.

In Figure 3b we show the Pearson correlation between the embedded feature vectors that comprise only the 18 features that were classified in [24] as invariant to translations and

scaling. Comparing them to the correlations presented in Figure 2, we observe the same patterns, but with different correlation intensities (we recall that the selection of a correlation threshold can be adapted with regard to the application that is being solved). Likewise, the correlation results in the original space described by the 18 selected features are presented in Figure 3a. The only difference appears in the second instance of the f_{14}, which has high negative correlation with most of the BBOB instances. This does not show in the plots using all 38 features, where this instance shows weak negative correlation with most of the other BBOB instances. This result shows that features that are not robust with respect to translation and scaling can have big impact on the comparison between problems in the original feature space. However, comparing the embedding spaces obtained with 38 features and 18 features, similar patterns appear.

B. Complementarity of HappyCat and HGbat Functions

To illustrate how to test instance similarity of problems that are not part of the training set, we study the correlation of the BBOB functions with the HappyCat and the HGbat problems introduced in [28].

The instances belonging to the HappyCat problem have a unique global minimum. According to [28], they are difficult to be optimized using state-of-the-art black-box optimizers such as the CMA-ES [29], particle swarm optimization [30], and differential evolution [31]. HGbat is similar to HappyCat, but differs in one term where a degree 8 polynomial instead of a degree 4 polynomial appears. For our illustration, we have select one random instance per each problem. To find the embeddings of these instances, we project their original feature value representation into the subspace that was learnt and described by the 120 BBOB instances with all 38 singular
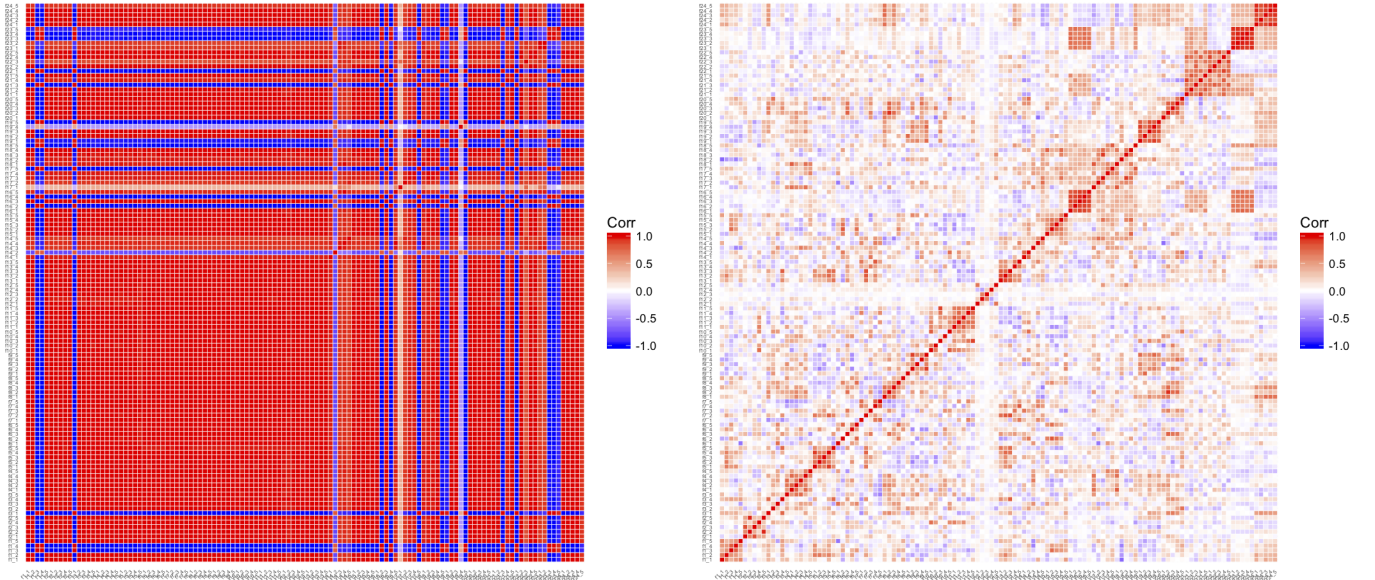
(a) Representation with 18 invariant features on translation and scaling in the original space.

(b) Embeddings with 18 singular values for the features invariant on translations and scaling.

Fig. 3: Pearson correlation between BBOB instances.

values, which describes our learnt embedded space. The Pearson correlation between the instances is presented in Figure 4, where we refer to the HappyCat instance as f_{25} and to the HGbat instance as f_{26}. As expected, the HappyCat instance is highly positively correlated with the HGbat instance. Both of them are also highly positively correlated with all instances of f_{23} (the so-called "Katsurra function"). Additionally, both instances are positively correlate with all instance of f_{16} (the "Weierstrass function"), with f_{21} (Gallagher's Gaussian 101-me Peaks function), and f_{24} ("Lunacek bi-Rastrigin function"). In [28], the authors discuss that the HappyCat function shares local similarities with the sharp ridge function, f_{13}. Looking in our results, however, the feature values do not seem to confirm (or be able to detect) such a correlation, raising the question if some essential properties are not captured by the selected features. These examples show that measuring instance correlation can be interesting also from a feature extraction point of view.

C. Sensitivity Analysis wrt Low-rank Approximations

Our previous analyses were done using all 38 singular values, since we were interested in high approximation accuracy obtained from the SVD. To explore the sensitivity of the results, different low-rank matrix approximations were generated by selecting a smaller number of singular values (starting from 6 and going up to the full set of 38 singular values in steps of 1) from which the instance embeddings are calculated. The non-graphical solution to the Cattell subjective scree test estimator [19] indicates that 16 singular values are a good starting point. To estimate the accuracy using the low-rank approximation X_r, the Frobenius norm $||X - X_r||_F$ of

the difference between the original matrix X and its low-rank approximation X_r is calculated (Figure 5). The result shows that when the number of singular values increases, the approximation is much better, which is expected.

Figure 6 presents the Pearson correlations between BBOB instances using the embeddings calculated with 19 and with 30 singular values, respectively. Comparing them, we observe once again similar patterns. By starting with the smaller number of singular values (Figure 6a) and going to a larger number of singular values (Figures 6b and 2, respectively), we just obtain higher approximation accuracy. This indicates that in cases where computational efficiency is also important, a smaller number of singular values can be used. To do this, the Frobenius norm of the difference between the original and the low-rank approximation can be a good indicator for the trade-off between time and accuracy.

D. Classification of BBOB instances

To further probe the utility of the learned embeddings, we also performed a multi-class classification, where for each instance we ask to predict the ID of the problem that it belongs to. That is, in our case we have 24 classes (one for each BBOB function), and for each class we have 5 instances. To evaluate the classification results, we used stratified 5-cross fold validation, where each fold consists of the first, second, third, fourth, and fifth instances for each problem, respectively. Then we repeated the process five times, where one fold was used for testing, and four of them for training the classifier. We should mention here that SVD is applied only on the training instances in order to learn their linear embedding, which are further used to train the classification model. In
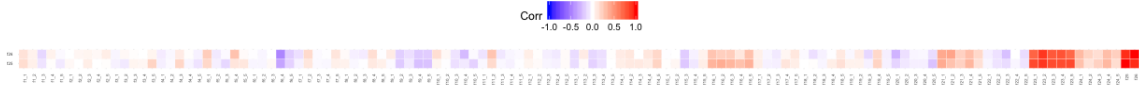
Fig. 4: Pearson correlation between BBOB instances, HappyCat and HGbat problem using the embeddings generated using 38 singular values.
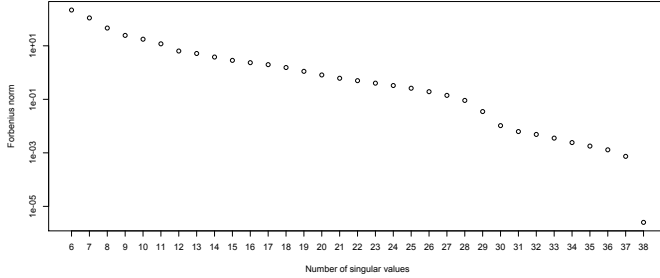

Fig. 5: Frobenius norm of the difference between the original matrix X and its low rank approximation X_r, for different numbers r of singular values.

TABLE I: Tested hyperparameters for the selected classifiers.

Algorithm	Hyperparameters
SGD	$\alpha \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^0\}$
SVM	$c \in \{0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.8, 2\}$
RF	$ntrees \in \{5, 10, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200\}$
ET	$ntrees \in \{5, 10, 20, 40, 60, 80, 100, 120, 140, 160, 180, 200\}$
AdaBoost	50 estimators, base classifier is ExtraTrees with 10 trees
Bagging	50 estimators, base classifier is ExtraTrees with 10 trees

case of testing, the test instances were only projected into the subspace that was learnt and described by the training instances, which describes our learnt embedded space.

For the classification, different state-of-the-art classifiers were trained exploring their hyperparameters by iterative grid search (see Table I). For the plots only the best-performing hyperparameter configuration for each classifier was selected. The classifiers that were used are: Stochastic Gradient Descent (SGD) (learning rate α=0.01) [32], k-Nearest Neighbours (k=4) [33], Support Vector Machine (SVM) with linear and radial basis kernel (regularization parameter $c = 1.8$) [34], Random Forest (RF) with Gini index (number of trees 80) and entropy selection of features (number of trees 160) [35], Extra Trees (ET) with Gini index (number of trees 80) and entropy selection (number of trees 200) [36], AdaBoost [37] and Bagging [38] with number of estimators 50 and Extra Trees as a base classifier with number of trees 10. The classification was performed using normalized (i.e., min-max scaling) and non-normalized features in both original and embedded space.

The average accuracy obtained for the original feature space and for each of the embeddings calculated using 19, 27, 30, and 38 singular values, respectively, are summarized in Figure 7. We also include in this figure the average classification accuracy when restricting the feature space to the 18 invariant features, once again in the original features space and in the embedded space (using all 18 singular values).

Looking at the results plotted in Figure 7a for the non-normalized features, the embeddings perform much better when they are used in a combination with classifiers that are not tree-based (i.e., SGD, KNN, and SVM), while the tree-based classifiers (i.e., RF, ET, AdaBoost, and Bagging) perform better with the original feature vectors. The absolute classification accuracy of the tree classifiers, however, is quite comparable (around 4-8% percentile difference in accuracy). Comparing the results of the sensitivity analysis with respect to the number of singular values, we see that the embeddings obtained for 27, 30, and 38 singular values (blue, green, and purple lines, respectively) have higher accuracy than the embeddings obtained with 19 singular values. However, these differences are again quite small (around 3% percentile).
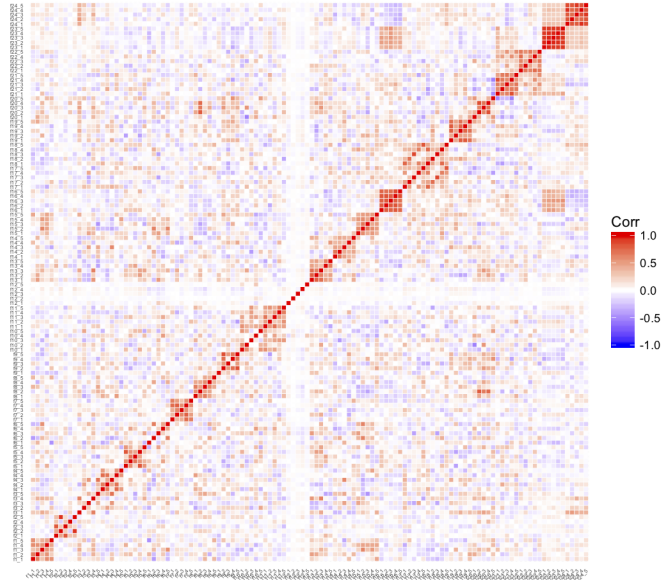
We note that all embeddings that were generated using the data for all 38 features have better accuracy than both the original and the embedded feature vectors using the invariant features only. This shows that some additional features contained in the 38 contribute to the classification accuracy. These results open new directions for future research, where the proposed methodology can be explored and can help the process of feature selection. In the case of non-normalized features, the non tree-based classifiers never converge with their combination with the original features. In the case of normalized features, the original features space performs also well with the non tree-based classifiers, however, the absolute performance differences are again small. The classification results obtained with the embeddings are similar and are not affected by the normalization. The learned embeddings have already performed some kind of normalization on the data.

V. CONCLUSIONS AND FUTURE WORK

In this work we have investigated how representation learning can help avoid bias and redundancies in the feature sets commonly studied in the context of black-box optimization. More precisely, we have applied a matrix factorization approach to exploratory landscape analysis (ELA). Experimental results obtained for the BBOB benchmark set showed that it is difficult to distinguish between instances from the same benchmark problem when using the original feature vectors, whereas the new fingerprint representations achieved a much better fit. We have also trained classifiers to predict which problem a given instance belongs to. Results for the original feature sets and those trained with the embedded feature vectors provided similar results, demonstrating that the learned embeddings contain a lot of information from the original space. We have also analyzed the dimensionality reduction
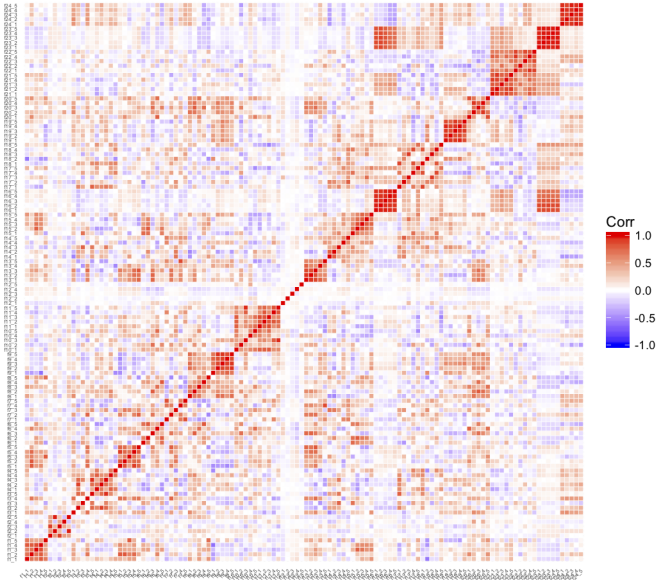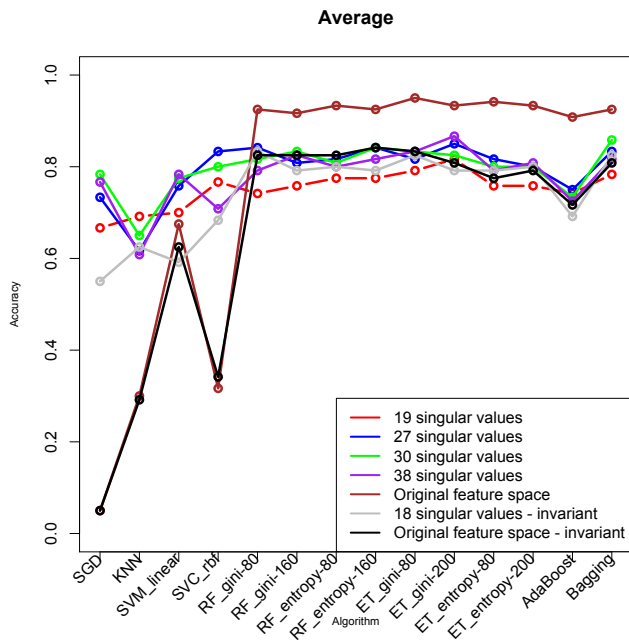
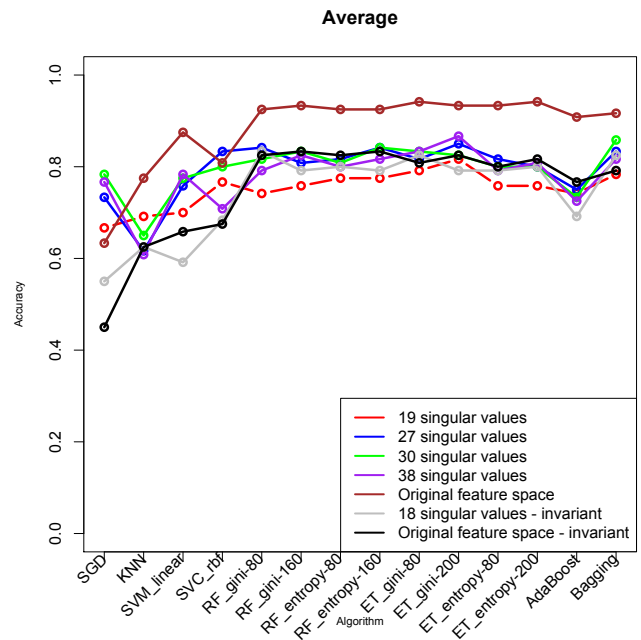(a) Embeddings calculated with 19 singular values.   (b) Embeddings calculated with 30 singular values.

Fig. 6: Pearson correlation between BBOB instances using their embeddings.

(a) Non-normalized features.   (b) Normalized features.

Fig. 7: 5-cross fold validation accuracy for predicting the problem class.

trade-off, showing that decent performance can be obtained also when reducing the dimension of the embedded vectors.

The main motivation for our study is in deriving explainable ELA-based algorithm design techniques. The study indicates that representation learning can be a meaningful complement to computing-intensive feature construction and selection

methods. Our study also opens a plethora of promising research directions, from which we summarize a few below. First and foremost, we will investigate the impact of our proposed approach on classical autoML tasks such as performance regression, algorithm selection, and algorithm configuration, both in the static [2], [7] as well as in the dynamic [39], [40]

case. We furthermore believe that the learned embeddings will prove useful for the selection as well as for the generation [41] of suitable benchmark instances.

We have focused in this work on computing vector representation of benchmark problem instances through linear dimensionality reduction on the five-dimensional BBOB problems. An extension to higher dimensions is a direction that we plan to follow, to generate insight into how the BBOB problems in different dimensions relate to each other. Additionally, we plan to investigate the non-linear properties of the problem by applying non-linear dimensionality reduction techniques.

In the longer term, we also plan on extending the analyses presented above to discrete and to mixed-integer optimization, where automated algorithm selection and configuration for general (i.e., not problem-specific) black-box optimization problems is still in its infancy [2].

Acknowledgment

References

[1] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning - Methods, Systems, Challenges.* Springer, 2019. [Online]. Available: https://doi.org/10.1007/978-3-030-05318-5

[2] P. Kerschke, H. Hoos, F. Neumann, and H. Trautmann, "Automated Algorithm Selection: Survey and Perspectives," *Evolutionary Computation*, vol. 27, no. 1, pp. 3–45, Mar. 2019.

[3] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory Landscape Analysis," in *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 2011, pp. 829–836.

[4] B. Derbel, A. Liefooghe, S. Vérel, H. Aguirre, and K. Tanaka, "New features for continuous exploratory landscape analysis based on the SOO tree," in *FOGA.* ACM, 2019, pp. 72–86.

[5] P. Kerschke and H. Trautmann, "Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco," in *Applications in Statistical Computing: From Music Data Analysis to Industrial Quality Improvement*, N. Bauer, K. Ickstadt, K. Lübke, G. Szepannek, H. Trautmann, and M. Vichi, Eds. Springer, 2019, pp. 93–123.

[6] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge, "Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges," *Inf. Sci.*, vol. 317, pp. 224–245, 2015.

[7] N. Belkhir, J. Dréo, P. Savéant, and M. Schoenauer, "Per instance algorithm configuration of CMA-ES with limited budget," in *Genetic and Evolutionary Computation Conference (GECCO).* ACM, 2017, pp. 681–688.

[8] P. Kerschke and H. Trautmann, "Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning," *Evolutionary Computation*, vol. 27, no. 1, pp. 99–127, 2019.

[9] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," in *Linear Algebra.* Springer, 1971, pp. 134–151.

[10] N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," 2016. [Online]. Available: http://arxiv.org/abs/1603.08785

[11] T. Eftimov, G. Popovski, Q. Renau, P. Korošec, and C. Doerr, "Gitlab repository with project data," https://repo.ijs.si/teftimov/mr-bec/-/tree/master/Exploratory%20Landscape%20Analysis/, 2020.

[12] Q. Renau, C. Doerr, J. Dreo, and B. Doerr, "Experimental Data Set for the study "Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy"," Jun. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3886816

[13] K. Malan and A. P. Engelbrecht, "A survey of techniques for characterising fitness landscapes and some possible ways forward," *Inf. Sci.*, vol. 241, pp. 148–163, 2013.

[14] P. F. Stadler, *Fitness landscapes.* Springer, 2002, pp. 183–204.

[15] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[16] S. Du, Y. Ma, S. Li, and Y. Ma, "Robust unsupervised feature selection via matrix factorization," *Neurocomputing*, vol. 241, pp. 115 – 127, 2017.

[17] S. Wang, J. Chen, W. Guo, and G. Liu, "Structured learning for unsupervised feature selection with high-order matrix factorization," *Expert Systems with Applications*, vol. 140, p. 112878, 2020.

[18] S. Wang, W. Pedrycz, Q. Zhu, and W. Zhu, "Subspace learning for unsupervised feature selection via matrix factorization," *Pattern Recognition*, vol. 48, no. 1, pp. 10–19, 2015.

[19] G. Raîche, T. A. Walls, D. Magis, M. Riopel, and J.-G. Blais, "Non-graphical solutions for cattells scree test," *Methodology*, 2013.

[20] J. C. Hayton, D. G. Allen, and V. Scarpello, "Factor retention decisions in exploratory factor analysis: A tutorial on parallel analysis," *Organizational research methods*, vol. 7, no. 2, pp. 191–205, 2004.

[21] S. Finck, N. Hansen, R. Ros, and A. Auger, "Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions," http://coco.gforge.inria.fr/downloads/download16.00/bbobdocfunctions.pdf, 2010.

[22] Q. Renau, J. Dreo, C. Doerr, and B. Doerr, "Expressiveness and robustness of landscape features," in *GECCO (Companion).* ACM, 2019, pp. 2048–2051.

[23] S. Saleem, M. Gallagher, and I. Wood, "Direct feature evaluation in black-box optimization using problem transformations," *Evol. Comput.*, vol. 27, no. 1, pp. 75–98, 2019.

[24] U. Škvorc, T. Eftimov, and P. Korošec, "Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis," *Applied Soft Computing*, vol. 90, p. 106138, 2020.

[25] I. Sobol', "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 86–112, Jan. 1967.

[26] J. Matoušek, *Geometric Discrepancy*, 2nd ed. Springer, 2009.

[27] Q. Renau, J. Dreo, C. Doerr, and B. Doerr, "Exploratory landscape analysis is strongly sensitive to the sampling strategy," in *Parallel Problem Solving from Nature (PPSN).* Springer, 2020, pp. 139–153.

[28] H.-G. Beyer and S. Finck, "Happycat–a simple function class where well-known direct search algorithms do fail," in *Parallel Problem Solving from Nature.* Springer, 2012, pp. 367–376.

[29] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[30] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[31] K. V. Price, "Differential evolution," in *Handbook of Optimization.* Springer, 2013, pp. 187–214.

[32] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *COMPSTAT'2010.* Springer, 2010, pp. 177–186.

[33] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & security*, vol. 21, pp. 439–448, 2002.

[34] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[35] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[36] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in neural information processing systems*, 2013, pp. 431–439.

[37] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[38] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[39] A. Biedenkapp, H. F. Bozkurt, F. Hutter, and M. Lindauer, "Towards white-box benchmarks for algorithm control," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1906.07644

[40] D. Vermetten, S. van Rijn, T. Bäck, and C. Doerr, "Online selection of CMA-ES variants," in *the Genetic and Evolutionary Computation Conference (GECCO).* ACM, 2019, pp. 951–959.

[41] K. Smith-Miles and S. Bowly, "Generating new test instances by evolving in instance space," *Computers & OR*, vol. 63, pp. 102–113, 2015.