U-FLEX: Unsupervised Feature Learning with Evolutionary eXploration

Nicolo' Bellarmino[1][0000−0001−5887−2598], Riccardo Cantoro[1][0000−0002−1745−5293], and Giovanni Squillero[1][0000−0001−5784−6435]

CAD Research Group
Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, IT
email{nicolo.bellarmino,riccardo.cantoro,giovanni.squillero}@polito.it
http://www.cad.polito.it/

Abstract. Feature selection is an essential task in machine learning and data mining that involves identifying a subset of relevant features from a larger set. This paper proposes a novel technique for unsupervised feature selection based on a Neural Network in conjunction with an evolutionary algorithm. The proposed method aims to extract subsets of the most discriminative and relevant features from high-dimensional data, which can be eventually used for efficient and accurate machine learning. An evolutionary algorithm is employed to generate the feature subsets, and the goodness of a feature subset is evaluated through the ability of a neural network to reconstruct the whole original input space by mean squared error minimization (in an auto-encoder fashion). Experimental results demonstrate the effectiveness of the proposed approach in finding relevant feature subsets for successive learning tasks, achieving better classification and regression accuracy compared to state-of-the-art feature selection methods.

Keywords: Unsupervised Learning · Feature Selection · Genetic Algorithm · Deep AutoEncoder · Supervised Learning.

1 Introduction

Feature selection is a crucial step in machine learning and data analysis, as it aims to identify the most informative subset of features from a large set of potential predictors [11]. This process can greatly improve the accuracy and efficiency of predictive machine learning models by reducing the dimensionality of the input space for successive supervised or unsupervised tasks, removing irrelevant or redundant features, and avoiding overfitting. It may also help to uncover underlying patterns and relationships in the data. However, traditional supervised feature-selection methods rely on labeled data, which can be a major limitation in real-world applications, where labels may be expensive or difficult to obtain. In recent years, unsupervised feature selection has emerged as an alternative active research area that aims to identify relevant feature subsets that can capture the underlying structure of the data without relying on labels [26].

Evolutionary Algorithms (EAs) represent a powerful, general-purpose, and well-known paradigm among evolutionary computation techniques that has been applied to a wide range of optimization problems, including feature selection [18]. EAs are particularly well-suited for feature selection tasks because they can explore a large search space of potential feature subsets and automatically generate candidate solutions using a combination of mutation, crossover, and selection operations [4]. In recent years, they have been used as effective techniques for unsupervised feature selection [7, 1], allowing for the automatic discovery of non-redundant feature subsets.

This paper proposes a novel approach to unsupervised feature selection based on genetic programming. The proposed approach aims to identify a subset of relevant features that maximize the ability to reconstruct the whole input data features space from subsets of features, eliminating redundant or irrelevant features. Subsets of features are generated by a Genetic Algorithm, and the mean squared error (MSE) between the features space reconstructed by deep neural networks and the original input space is used to evaluate the goodness of feature subsets. We use the MSE in conjunction with a regularization term as the fitness of the Genetic Algorithm to guide the search towards simpler and more generalizable feature subsets, preferring lower-dimensional solutions. We used a Deep (asymmetric) Auto Encoder to reconstruct the input space, which takes as input a subset of the original features.

The experimental results demonstrate that our proposed method can effectively identify informative feature subsets in various real-world datasets. We evaluate the proposed approach on several benchmark datasets and compare its performance with two classical feature selection methods, both supervised and unsupervised. The experimental results show that the proposed approach outperforms the compared methods in terms of classification/regression accuracy and feature subset size. Our approach offers a promising direction for unsupervised feature selection, enabling more efficient and accurate predictive modeling in a wide range of applications.

The rest of the paper is organized as follows. Section 2 provides a review of related work on unsupervised feature selection and genetic programming. Section 3 presents background information that describes theory and concept useful for understanding successive experiments. Section 4 describes the proposed approach in detail, including the genetic programming framework, the fitness function, and the diversity preservation mechanism. Section 5 presents the experimental setup, the dataset used, the comparison method, and the results of the evaluation. Finally, section 6 concludes the paper with the implications and limitations of the proposed approach, a summary of the main results, and future directions.

2 Related Work

Several surveys about feature selection have been published [20, 11]: they provide basic information on feature selection approaches to better understand the

underlying problem. Evolutionary Algorithms (EAs) have been successfully applied to feature selection problems in the past [18]. In [2], the problem of the high time needed to evaluate the fitness of any individual in the EA process has been faced: the computational time of the EAs may be quite high. Therefore, the combination of EA and a classification method maybe not be efficient. They faced the problem by fitness function approximation, reducing the time needed to evaluate each candidate solution. In [1], an unsupervised feature selection approach based on EA was developed for text-clustering, relying on the mean absolute difference in text embedding. A similar approach was developed in [7], where the fitness function relies on KMeans clustering metrics. In [5], an unsupervised, model-agnostic, wrapper method for feature selection has been proposed. The authors assumed that if a feature can be predicted using the others, it adds little information to the problem. Therefore, it could be removed without impairing the performance of whatever model will be eventually built. There, the proposed method iteratively identifies and removes predictable, or nearly-predictable, redundant features, allowing trade-off complexity with expected quality. The theory and the philosophy behind that approach are similar to the one proposed here, but our novel work is trying to reconstruct the whole feature set all at once by starting from feature subsets.

3 Background

3.1 Features selection

Feature selection is the process of identifying and selecting the most relevant feature subset in a dataset from a large pool of potential input variables for building a predictive model, to identify the most informative features that capture the essential characteristics of the underlying data while discarding redundant or noisy features that may lead to overfitting or poor generalization. It is a critical pre-processing step in machine learning and a fundamental research topic.

The need for feature selection arises due to the increasing complexity and dimensionality of real-world datasets, which often contain numerous variables, many of which are irrelevant or redundant, and data analysts may have no or limited domain knowledge to pre-prune the data input space. These techniques may help reduce the data's dimensionality, improve the interpretability of the results, and enhance the performance of machine learning models.

Input space reduction can also be achieved with Feature Extraction [15]: it involves reducing the dimensionality of a dataset by extracting the most important features. Principal Component Analysis (PCA) is one of the most popular techniques for Feature Extraction [17]. The main difference between Feature Extraction and Feature Selection is that Feature Extraction creates new features by combining the original ones, while Feature Selection selects a subset of the original features.

Various methods of feature selection have been developed, and they can be grouped into three main groups: filter methods, wrapper methods, and embedded methods [11].

Filter methods rely on statistical measures or machine learning algorithms to rank the features based on their relevance and importance. Correlation-based metrics can be used for the selection of features that are less-correlated with each other, thereby discarding redundant information from the dataset in an unsupervised fashion. Alternatively, we can select only the features that are (highly) correlated with the target label we aim to predict in a supervised fashion [28]. A correlation threshold is chosen and the features that score below this threshold are discarded. Once a subset of features is selected, it can be used as an input to the chosen classifier algorithm. Unlike the other feature selection methods (wrapper and embedded), filter methods are independent/separated from the successive ML algorithm [25], and thus they can be considered an independent pre-processing step.

Wrapper methods use a machine learning model to evaluate the performance of different subsets of features and select the best one, by computing error measures on a selected test set, eventually relying on cross-validation. As an example, evolutionary algorithms can be used as wrapper methods for feature selection [18], by relying on an underlying ML model trained on possible candidate feature subsets. The ML model is then tested on a proper test set, usually in a cross-validation fashion. A widely-used wrapper method is the Recursive Feature Elimination (RFE) [12]

Finally, embedded methods incorporate feature selection into the process of model training, by optimizing the feature subset during the learning process, as it happens in tree-based models like Decision Trees or Random Forests.

The choice of feature selection method depends on the characteristics of the dataset, the type of machine learning model used, and the specific application. Each method has its strengths and weaknesses, and the selection of the best approach requires careful evaluation and experimentation. Furthermore, recent advances in deep learning and neural networks (NN) have opened up new opportunities for feature selection, by leveraging the representation learning capabilities of these models [10]: feature selection has traditionally been an important technique for shallow learning models, but for deep learning the need for explicitly selecting the features has been partially alleviated by the ability of the NNs to learn meaningful representations directly from raw data. In these types of models, the concept of feature selection is often replaced by the concept of representation learning, where the goal is to learn a set of features that capture the most relevant information from the input data. NNs are capable of automatically discovering and extracting high-level features from the raw data, without the need for manual feature engineering [10]. This is achieved by stacking multiple layers of nonlinear transformations, each layer learning increasingly abstract representations of the input data. In some cases, however, it may still be beneficial to apply feature selection techniques in conjunction with deep learning. For example, to reduce computational complexity and memory requirements of deep neural networks when dealing with high-dimensional data, while also improving their interpretability. Additionally, feature selection can be useful in situations

where the dataset contains noisy or irrelevant features, which can negatively impact the performance of deep learning models.

3.2 Evolutionary Algorithm

EAs are adaptive search techniques, for which improvement over random and local search methods has been demonstrated [9, 18]. An EA maintains a population of individuals representing potential solutions across the generations. Each individual is evaluated based on its fitness with respect to the objective function being minimized or maximized. The fittest individuals are selected, and through crossover and mutation, new individuals are generated, leading the population toward better solutions in the given domain. EAs have been proposed as a tool for identifying and selecting the optimal subset of features for subsequent machine learning algorithms, such as classification systems [9, 18].

In a population of individuals, represented as chromosomes $(C_1, C_2 \cdots C_p)$, each chromosome is a potential solution of the underlying task (and in the feature selection domain, a set of relevant features). EAs are advantageous for classification processes due to their insensitivity to noise and independence from domain knowledge. EAs rely on selection, crossover, and mutation. The selection phase involves choosing the best feature subsets based on their fitness values. The crossover operator combines these selected subsets to generate offspring solutions, while the mutation operator introduces random changes to explore new areas of the search space. Various evolutionary operators have been developed in the past to drive the optimization process of EAs [21, 16]. When utilizing EAs, several aspects need to be determined, including chromosome codification, chromosome length, population size, genetic operators, and fitness function. Typically, the population size is set to 50 or 100 [18]. Standard genetic operators, such as one-point crossover and standard mutation, are commonly employed [21, 16]. The probabilities of crossover and mutation are hyper-parameters that should be optimized [22]. The choice of fitness function depends on the specific problem being optimized. In ML tasks, both supervised (based on labels, such as prediction performance in classification algorithms) and unsupervised (e.g., clustering capacity) fitness functions can be used, sometimes combined with penalties or additional factors.

4 Methods

The core idea behind the proposed approach is to have a model able to predict the whole feature space from a subset. This approach is similar to the one commonly used by autoencoders (AE), which project the input data into a latent space, and from this, they aim to reconstruct the original input space by minimizing the MSE between the inputs and their reconstruction. But while in traditional AE the inputs and the outputs of the networks are the same, in our models the inputs are feature subsets of the original one, thus making the model asymmetric. The chosen ML model is a fully-connected NN, that is a multi-layer perceptron,

closely similar to an auto-encoder. It consists of an encoder and a decoder, where the encoder maps the input features to a lower-dimensional representation and the decoder reconstructs the original features from the encoded representation by MSE minimization. If the model can reconstruct the whole input space by a subset of the original feature set, the selected subset is representative of the whole feature space, and thus, redundant information from the dataset should have been removed. By starting from this idea, we propose to evaluate feature subsets based on how good is the model to reconstruct the original input space, generating feature subsets using an EA. The EA work on binary bit strings chromosomes, of length equal to the total number of features in the dataset. For each chromosome, if the gene in position i is zero, the i^{th} feature was not selected. Each gene codes the presence or absence of a feature in the subset. The GA approach permits to evolve through feature subsets that both minimize the reconstruction MSE and try to keep the size of the feature subsets small. We added a regularization term to the MSE to prevent preferring high-dimensional feature subsets. This encourages simpler and better generalizing solutions while avoiding keeping the whole feature set as the best set. In formula, given X as the original input, $\tilde{X}$ the output of the autoencoder, Fs the chosen subset of features, our fitness function is defined as:

$$(\alpha \cdot MSE(X, \tilde{X}) + \beta \cdot length(Fs))^{-1} \tag{1}$$

Since we are solving a minimization problem (minimizing both the MSE and the length of the chosen feature subset) but we want to maximize the fitness function, we raised to the power of -1 to have that higher fitness function mean better solutions, to be compliant with classical EA framework. The same results can be obtained without raising to the power of -1, but directly minimizing the fitness function. We terminate the EA algorithm when a maximum number of generations g is reached (Alg. 1). Then, the feature subsets in the final population are evaluated through supervised error metrics (R2, MSE, Accuracy, ...) on a proper validation set. The chromosome with the lowest error on the validation set is selected as the final solution. Then, we evaluate the selected feature subset by comparing its performance to the original model using all features, on a proper test set.

5 Experiments

The proposed method is implemented in Python using `scikit-learn` [24], `pandas` [19, 27], and `numpy` [14] libraries for dataset preprocessing. The model was built and run on GPU using `pytorch` [23]. We conduct experiments on four benchmark datasets, to evaluate the effectiveness of the proposed method and compare it to state-of-the-art unsupervised feature selection methods. All experiments were conducted on a server equipped with an Intel ® Core™ i9-9900K CPU @ 3.60GHz × 16, 32 GiB of RAM, and an Nvidia ® 2080 TI GPU with 12 GiB of reserved RAM.

Algorithm 1 U-FLEX Evolutionary Algorithm

$max_g \leftarrow generation$
$g \leftarrow 0$
$tournamentSize \leftarrow t$
$populationSize \leftarrow p$
$mutationRate \leftarrow m$
$population \leftarrow random(popSize)$
while $g \leq max_g$ **do**
  $fitnessValues = evaluate(population)$
  $parents = tournamentSelection(population, fitnessValues, tournamentSize)$
  $population \leftarrow []$
  for $i \in range(0, populationSize, 2)$ **do** ▷ iterate over couples of parents
    $offspring1, offspring2 = crossover(parents[i], parents[i+1])$
    $population.append(mutate(offspring1, mutationRate))$
    $population.append(mutate(offspring2, mutationRate))$
  end for
  g = g + 1
end while

5.1 Datasets

Here, we describe the dataset used for validating our feature selection framework. The number of samples, features, and the task performed are described in Tab. 1.

MSD The Million Song Dataset (MSD) [6] is a collection of audio features and metadata for over a million songs, created by The Echo Nest and released to the public in 2011. The year prediction task is one of several tasks that can be performed on the Million Song Dataset, and it has been used as a benchmark for evaluating machine learning models for music analysis. The goal of the task is to train a machine learning model able to predict the year of release for a set of test songs. The audio features used in the task include features like tempo, timbre, and spectral contrast, which are extracted from the audio signal using signal processing techniques. The metadata associated with each song, such as the artist name and album title, can also be used as input to the machine learning model. The year prediction task is typically framed as a regression problem, where the output is a continuous value representing the predicted year of release.

Artificial Dataset A random artificial regression problem (ART). The input X was generated with a low-rank-fat tail singular profile. The dataset was created by means of `scikit-learn` package. Most of the variance can be explained by a bell-shaped curve of width 30, while we have 500 components. The low-rank part of the profile can be considered the structured signal part of the data while the tail can be considered the noisy part of the data that cannot be summarized by a low number of linear components. The output is generated by applying a

random linear regression model with bias equal to 100 and with 30 informative and nonzero regressors to the previously generated input and Gaussian-centered noise with a standard deviation equal to 0.05. 30, 000 samples were created.

Madelon MADELON (MAD) is an artificial dataset for balanced binary classification, containing data points grouped in 32 clusters placed on the vertices of a five-dimensional hypercube and randomly labeled. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. An additional number of 480 distractor features called 'probes' having no predictive power were added, for a total of 500 features. It is a classical dataset used for feature selection challenges [13].

Table 1. Datasets information

Name	Pseudo	# Samples	# Features	Task	Reference
Million Song Dataset	MSD	515,345	90	Regression	[6]
Artificial	ART	30,000	100	Regression	[24]
Madelon	MAD	2600	500	Classification	[13]

5.2 Comparison

We compared our framework with two classical feature selection approaches: a Pearson Correlation-based univariate filter method, based on a threshold, and a wrapper method based on an Evolutionary Algorithm.

Univariate Feature Selection Pearson correlation can be used as a filter method for unsupervised univariate feature selection by identifying the linear correlation between independent features in the dataset, without relying on the target label. The most informative features in a dataset are the less-correlated ones: if two pairs of features present high linear correlation, they may provide redundant or overlapping information and can be eliminated from the dataset without losing much information. It is important to note that Pearson correlation is only effective for identifying linear relationships between variables. If the relationship between variables is non-linear, other measures of correlation such as Spearman's rank correlation or Kendall's tau correlation may be more appropriate. By eliminating redundant or overlapping information, the resulting subset of features can provide a more efficient and informative representation of the data. An unsupervised correlation-based approach for feature selection can be performed by calculating the Pearson correlation between each pair of features in the dataset. Then, it is possible to identify pairs of highly correlated features by setting a threshold value for the correlation coefficient and removing

any features that have a correlation value above the threshold. Perfectly correlated variables are truly redundant in the sense that no additional information is gained by adding them, but very high variable correlation (or anti-correlation) does not mean the absence of variable complementarity [11]. We used 80% as the correlation threshold.

Wrapper Supervised EA Feature Selection We compared the obtained results with a supervised wrapper method for feature selection, based on EA (Supervised EA). The fitness function used is the error metric of a supervised model on a validation set, with the objective of maximizing the prediction performance of the underlying model in a 5-folds CV fashion. The chromosomes with the best fitness score are selected for reproduction, and the process continues until a stopping criterion is met (maximum number of generations). We used `scikit- learn genetic-opt` package [3]. It is a popular open-source Python package that provides a genetic algorithm-based approach for hyperparameter tuning and feature selection in `scikit-learn`. The genetic feature selection algorithm starts by randomly generating a set of chromosomes, each representing a different subset of features. It then applies a genetic algorithm to iteratively select the most relevant subset of features, optimizing the prediction performance (R2 score for regression, Accuracy for classification) of a specified estimator using k-fold cross-validation. The package implements the $\mu + \lambda$ evolutionary strategy, thus keeping both parent and children at each generation. μ represents the number of parent individuals in a population, while λ is the number of offspring individuals.

5.3 Parameter Setting

As written in Section 4, the fitness function we used is a combination of two terms that we aim to optimize: the ability of the network to reconstruct the original inputs and the dimensionality of the feature subsets found. The parameters α and β in equation 1 control the stability of the learning process, and they are dataset-dependent. In our settings, for a dataset with n total features, $\alpha = 10$ and $\beta = 1.5 \cdot \frac{1}{n}$ worked well in general, making the two terms of the objective functions do not prevail one over the other.

The EA was implemented by using very simple criteria for the creation of a new population at each generation. The selection was based on tournament selection with a tournament size of 5. First, the candidate with the best fitness function is added to the new population (*elitism*). The participants in each tournament are selected at random among the population, and the winner of each tournament is added to the parents, that will be subjected to crossover in the next phase (see Alg. 1)

The crossover is a uniform crossover between each parent. The mutation was a single uniform bit-flip based on a mutation rate of 0.01, meaning that each gene has a probability of 0.01 of being flipped. The mutation was applied to each of the offspring. The population size was set to 30.

The deep autoencoder was trained for just 1 epoch: even if this number seems to be low or not enough to say something about the reconstruction ability of the autoencoder, we didn't find great improvement in the genetic learning process by increasing the number of epochs (we tried from units to tens of epochs): what we aim is having a rough idea of the ability of the network to reconstruct the original feature set from feature subset. Increasing the number of epochs of training leads for sure to more reliable measurements of the input reconstruction ability of the feature subset but at the cost of an increasing time in the evaluation of each individual of the population: the main bottleneck of our approach is precisely the evaluation step since evaluating each candidate requires the full training of a NN. Considering just one epoch of training is an approximation of the fitness function computation, but it is needed to reduce the fitness evaluation time [2]. More sophisticated ways of tackling the problem can be found. The architecture is a simple fully connected autoencoder. The encoder has 2 hidden layers of size 128 and 64, while the decoder is symmetric to the autoencoder, but the last layer has as output dimensionality the number of original features. The activation function is the Leaky-ReLU. We used ADAM optimizer with learning rate $lr = 1e - 3$ For fitting the successive supervised model, we used Linear Regression for regression tasks and SVM for the classification, with rbf kernel and parameter $C = 1$ (no hyperparameters tuning for the shallow model, we just used the default parameters provided by `scikit-learn`). We randomly split the dataset into train, validation, and test sets. We trained all the models (for the supervised and unsupervised evolutionary feature selection) on the same training set, and we tested them on the test set. We used 80% of the samples as the training set, and the remaining 20% as a blind test set, for each dataset. Additionally, the 15% of the training set was used as the validation set for the NN evaluation, for computing the fitness function, at each generation, and for choosing the best feature set.

The performances of the supervised model were tested on the test set. We used as error metrics the *accuracy*, in percentage, for the classification task and the *normalized Root Mean Squared Error* (nRMSE) for the regression task. The nRMSE is nothing more than the root mean squared error RMSE but normalized by the mean of continuous label $y_t rue$ in the test set, i.e. $nRMSE = RMSE(y_{true}, y_{pred})/mean(y_{true})$. The lower this value, the better the performance of the model. These metrics were used to have error percentages rather than absolute values.

5.4 Results

The main results are presented in table 2. Our framework successfully found relevant feature subsets, and the obtained results in the supervised task are comparable with the other methods considered, both for the number of features found, and prediction error reached (comparable with other research experiments on the same datasets, [6, 8]) We let the algorithm run for a different number of generations, depending on the dataset. Since the MAD dataset contains a low number of samples, the evaluation of each candidate lasts a few seconds. In this

case, to reach better performance (for both size and prediction error, thus more valuable feature subsets), we can let the algorithm run for more generations, and train the NN for more epochs (with a minor increase in time). With 50 steps, we were able to reach the 72.34% of accuracy on the test set with 136 features. With 150 generations, about 119 features are selected, with 77.50% of accuracy. With the Supervised EA, after 150 generations, for the MAD dataset 220 features were selected, with an accuracy of 74.23% (see Tab 2). Fig 1 presents the trends of the mean fitness function and the max accuracy in the population for the MAD dataset. For the Supervised EA, we are directly optimizing the accuracy over the validation set (that increases until a plateau), while for U-FLEX not. But the fitness function correctly increases over the generation, meaning that we are pruning the original feature set. In MSD, after 50 generations, we found final feature subsets of similar size (just 9 more, 10% of the total) with respect to the supervised EA method, reaching the same prediction error (Tab 2). For all the datasets explored, the Pearson Correlation method was not able to prune a relevant number of features, and in some cases (for example, the artificial dataset), no features were removed. Datasets with a high amount of correlation among features (MSD) are easily managed by our framework. Other types of datasets (such as MAD and ART), in which many features are practically noise, are not so easily tractable with reconstruction approaches but would require more knowledge on the type of data we must manage. These types of problems are more suitable to be solved with supervised approaches rather than unsupervised ones since the features are mutually independent, and thus we would need more features to reconstruct the original input space, even if they would be noise for a supervised task. But even with this type of problem, the difference between the number of features found by the supervised and unsupervised genetic algorithm in the ART dataset is not so pronounced (50 features, 10% of the total after 50 generations). A comparison between the fitness function of the supervised and unsupervised approach is shown in Fig 2. [1].

Table 2. Error metrics (nRMSE for MSD and ART, Accuracy for MAD) on the benchmark datasets. 150 generations for the MAD dataset, 50 generations for MSD and ART datasets. The higher the accuracy (MAD) the better the result. The lower the nRMSE (ART, MSD), the better the result.

	Features Used/Error metric%			
Dataset	All	Pearson	Supervised EA	U-FLEX (Our)
MSD	90/0.48%	87/0.49%	62/0.49%	71/0.51%
MAD	500/(68.85%)	490/(65.96%)	220/(74.23%)	119/(77.50%)
ART	500/(0.05%)	500/(0.05%)	232/(0.05%)	282/(0.13%)

[1] Promising results were also found on an industrial classified dataset for regression: the computed nRMSE on all the features was 1.83%. With Pearson filter methods, 2.5% of the features were selected, with a nRMSE of 1.86%. The wrapper-supervised approach selected the 33% of features, with an error of 1.37%. Our approach converged to 19% of the features, with 1.38% of nRMSE

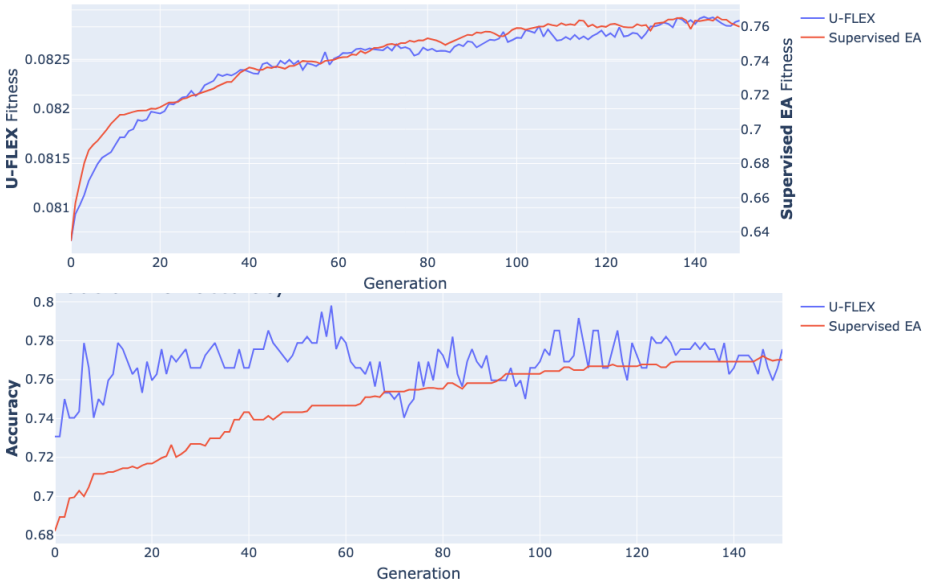

Fig. 1. Comparison of U-FLEX and Supervised EA mean fitness function (upper) and max accuracy (lower) over the population. MAD dataset, 150 generations. For the Supervised EA, we are directly optimizing the accuracy over the validation set (that increases until a plateau), while for U-FLEX not.

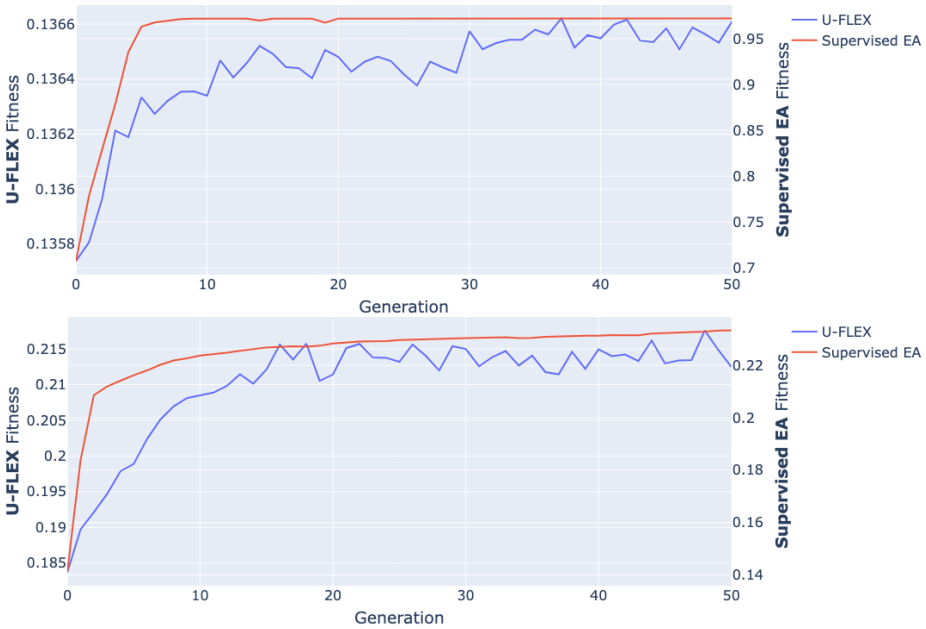

Fig. 2. Mean fitness function comparison between Supervised EA and U-FLEX in the population for each generation, computed on ART (upper) and MSD (lower) datasets.

6 Conclusion

We presented a novel method for unsupervised feature selection, namely U-FLEX, based on an evolutionary approach. We aim to select feature subsets that maximize the ability of an underlying NN to reconstruct the input space by MSE loss minimization, in conjunction with a regularization term to prefer low-size feature subsets. Experiments showed that our framework outperformed a classical unsupervised Pearson correlation-based filter, and with comparable performances over a supervised wrapper method based on an EA. Our framework effectively selected lower-size feature subsets, eliminating redundant information from the datasets. Experiments were based on very simple crossover, selection, and mutation operations and no hyperparameter tuning was done for the baseline shallow model and NN architecture. Despite that, good feature subsets were selected, making the underlying idea promising. With more sophisticated methods for evolving the solutions through the generations (such as increasing the epochs of training of the underlying NN, dynamically change the coefficient α and β in the equation 1, experimenting with more underlying methods to evolve through the generation and more efficient way to train the NN), convergence time could be improved and plateaus in the fitness function could be avoided, but these were not implemented for this work. Also, using more elaborated deep underlying NN could help in adapting the proposed techniques to other types of datasets (natural images, audio wavelengths, or spectra). More rigorous work should be done over a higher number of datasets and hyper-parameters, but the evaluation and the results make, in the authors' minds, the proposed approach promising in the unsupervised feature selection panorama.

References

1. Abualigah, L., Khader, A.T., Al-Betar, M.: Unsupervised feature selection technique based on genetic algorithm for improving the text clustering. pp. 1–6 (07 2016). https://doi.org/10.1109/CSIT.2016.7549453
2. Altarabichi, M.G., Nowaczyk, S., Pashami, S., Mashhadi, P.S.: Fast genetic algorithm for feature selection — a qualitative approximation approach. Expert Systems with Applications p. 118528 (2023). https://doi.org/https://doi.org/10.1016/j.eswa.2022.118528, https://www.sciencedirect.com/science/article/pii/S0957417422016049
3. Arenas, R.: sklearn-genetic-opt. https://github.com/rodrigo-arenas/Sklearn-genetic-opt (2023)
4. Barbiero, P., Lutton, E., Squillero, G., Tonda, A.: A novel outlook on feature selection as a multi-objective problem. In: Idoumghar, L., Legrand, P., Liefooghe, A., Lutton, E., Monmarché, N., Schoenauer, M. (eds.) Artificial Evolution. pp. 68–81. Springer International Publishing, Cham (2020)
5. Barbiero, P., Squillero, G., Tonda, A.: Predictable Features Elimination: An Unsupervised Approach to Feature Selection, pp. 399–412 (02 2022). https://doi.org/10.1007/978-3-030-95467-329
6. Bertin-Mahieux, T., Ellis, D.P., Whitman, B., Lamere, P.: The million song dataset. In: Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011) (2011)
7. Boutegrabet, W., Piot, O., Guenot, D., Gobinet, C.: Unsupervised feature selection by a genetic algorithm for mid-infrared spectral data. Analytical Chemistry **94**(46), 16050–16059 (2022). https://doi.org/10.1021/acs.analchem.2c03118, https://doi.org/10.1021/acs.analchem.2c03118, pMID: 36346912
8. De Stefano, C., Fontanella, F., Scotto di Freca, A.: Feature selection in high dimensional data by a filter-based genetic algorithm. In: Squillero, G., Sim, K. (eds.) Applications of Evolutionary Computation. pp. 506–521. Springer International Publishing, Cham (2017)
9. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer Publishing Company, Incorporated, 2nd edn. (2015)
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
11. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (mar 2003)
12. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning **46**, 389–422 (01 2002). https://doi.org/10.1023/A:1012487302797
13. Guyon, I.M.: Design of experiments for the nips 2003 variable selection benchmark (2003)
14. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
15. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference, and prediction. Springer Science Business Media (2009)

16. Heiss-Czedik, D.: An introduction to genetic algorithms. Artificial Life **3**, 63–65 (1997)
17. Jolliffe, I.T.: Principal component analysis. Springer (2011)
18. Martin-Bautista, M., Vila, M.A.: A survey of genetic feature selection in mining issues. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 2, pp. 1314–1321 Vol. 2 (1999). https://doi.org/10.1109/CEC.1999.782599
19. Wes McKinney: Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman (eds.) Proceedings of the 9th Python in Science Conference. pp. 56 – 61 (2010). https://doi.org/10.25080/Majora-92bf1922-00a
20. Miao, J., Niu, L.: A survey on feature selection. Procedia Computer Science **91**, 919–926 (2016). https://doi.org/https://doi.org/10.1016/j.procs.2016.07.111, https://www.sciencedirect.com/science/article/pii/S1877050916313047, promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management (ITQM 2016)
21. Mitchell, M.: An introduction to genetic algorithms (1996)
22. Mitchell, M.: An introduction to genetic algorithms. Complex adaptive systems, Cambridge, Mass., 7. print edn. (2001)
23. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
24. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
25. Pudjihartono, N., Fadason, T., Kempa-Liehr, A.W., O'Sullivan, J.M.: A review of feature selection methods for machine learning-based disease risk prediction. Frontiers in Bioinformatics (2022). https://doi.org/10.3389/fbinf.2022.927312, https://www.frontiersin.org/articles/10.3389/fbinf.2022.927312
26. Solorio-Fernández, S., Carrasco-Ochoa, J., Martínez-Trinidad, J.F.: A review of unsupervised feature selection methods. Artificial Intelligence Review **53** (02 2020). https://doi.org/10.1007/s10462-019-09682-y
27. pandas development team, T.: pandas-dev/pandas: Pandas (Feb 2020). https://doi.org/10.5281/zenodo.3509134, https://doi.org/10.5281/zenodo.3509134
28. Xie, J., Wang, M., Xu, S., Huang, Z., Grant, P.W.: The unsupervised feature selection algorithms based on standard deviation and cosine similarity for genomic data analysis. Frontiers in Genetics **12** (2021). https://doi.org/10.3389/fgene.2021.684100, https://www.frontiersin.org/articles/10.3389/fgene.2021.684100