Feature Learning in Feature–Sample Networks using Multi-objective Optimization

Filipe Alves Neto Verri
Institute of Mathematical and Computer Sciences
University of São Paulo – São Carlos, SP, Brazil
E-mail: filipeneto@usp.br

Renato Tinós, Liang Zhao
Department of Computing and Mathematics, FFCLRP
University of São Paulo – Ribeirão Preto, SP, Brazil
E-mail: rtinos@ffclrp.usp.br, zhao@usp.br

***Abstract*—Data and knowledge representation are fundamental concepts in machine learning. The quality of the representation impacts the performance of a learning model directly. Feature learning transforms or enhances raw data to structures that are effectively exploited by those methods. In recent years, several works have been using complex networks for data representation and analysis. However, no feature learning method has been proposed to enhance such category of representation. Here, we present an unsupervised feature learning mechanism that works on datasets with binary features. First, the dataset is mapped into a feature–sample network. Then, a multi-objective optimization process selects a set of new vertices to produce an enhanced version of the network. The new features depend on a nonlinear function of a combination of preexisting features. Effectively, the process projects the input data into a higher-dimensional space. To solve the optimization problem, we design two metaheuristics based on the lexicographic genetic algorithm and the improved strength Pareto evolutionary algorithm (SPEA2). We show that the enhanced network contains more useful information and can be exploited to improve the performance of machine learning methods. The advantages and disadvantages of each optimization strategy are discussed.**

***Index Terms*—Feature learning, complex networks, multi-objective optimization, genetic algorithm**

I. Introduction

A good representation of the encoded knowledge in a machine learning model is fundamental to its success. Several data structures have been used for this purpose, for instance, matrices of weights, trees, and graphs [1], [2].

In recent years, several works have been using *complex networks* for data representation and analysis [3]–[5]. Complex networks are graphs with a nontrivial topology and can be used to represent the interactions of a dynamical system [6]. Advances in the science of complex systems bring several tools to understand such systems.

In [7], we describe how to map a dataset with binary features into a bipartite complex-network. Such network is called *feature–sample network*. We used this representation in a semi-supervised learning task called positive-unlabeled (PU) learning. Such task consists of a binary classification problem in which few positive samples and many unlabeled samples are given [8].

When dealing with machine learning problems, we often need to pre-process the input data. *Feature learning* transforms or enhances raw data to structures that are effectively exploited by learning models. Autoencoders and manifold learning are examples of feature learning methods [9]. Traditional approaches, however, can not be directly applied in feature–sample networks.

In this paper, we propose a feature learning process to enhance feature–sample networks. In summary, we optimize the network by adding a limited number of new vertices based on a non-linear function of the preexisting ones. The set of new vertices is determined by a multi-objective algorithm, in which the goal is to maximize the number of features while maintaining some properties of the original data. Two multi-objective approaches are proposed: a lexicographic genetic algorithm (LGA) and an algorithm based on the improved strength Pareto evolutionary algorithm (SPEA2). The choice of the optimization methods aims to compare opposite results: one that considers diversity – SPEA2 – and another that prioritizes fast convergence – LGA.

We show that enhanced feature–sample networks improve the performance of learning methods in the two main machine learning paradigms: supervised and unsupervised learning. We also expose the pros and cons of the proposed optimization approaches.

The rest of this paper is organized as follows. Section II describes how to enhance feature–sample networks as an optimization problem. Section III presents the two proposed algorithms. In Section IV, computer simulations illustrate the optimization process and assess the performance improvements in machine learning tasks. Finally, we conclude this paper in Section V.

II. Enhanced feature–sample networks

In this section, we describe how we enhance a feature–sample network by adding to it a constrained number of new features. Connections between the samples and each new feature depend on a nonlinear function of a combination of preexisting features. The chosen set of new features is the result of an multi-objective optimization process: the algorithm should maximize the number of new features and uniformly distribute them along the samples. The second objective is adopted because the unbalanced distribution of the new features leads to artifacts that hinder the performance of learning methods.

In the following subsections, we first review the feature–sample networks, then describe the creation of new features.

978-1-5090-6017-7/18/$31.00 ©2018 IEEE

Moreover, we elaborate an optimization problem to enhance feature–sample networks.

A. Feature–sample network

Assume a machine learning task where the inputs are represented by a dataset $\mathcal{B} = \{\vec{x}_1, \ldots, \vec{x}_N\}$. The dataset $\mathcal{B}$ consists of N binary samples, each one with D features, i.e., $\vec{x}_i = [x_{i1}, \ldots, x_{iD}] \in \{0,1\}^D$. The feature vectors are sparse, that is, the number of elements with value 1 is much lower than the dimension D.

The *feature–sample network* $\mathcal{G}$ is a bipartite complex-network whose edges connect samples and features of the dataset $\mathcal{B}$. A simple, unweighted, undirected graph $(\mathcal{V}, \mathcal{E})$ represents the feature–sample network. A vertex of G represents either a feature or a sample, and edges only exist between a *sample* v_i and *feature* v_{N+j} if $x_{ij} = 1$, where $i = 1, \ldots, N$ and $j = 1, \ldots, D$.

B. And-features definition

According to the *Cover's Theorem* [10], given a not-densely populated space of a classification problem, a training dataset that is not linearly separable can be transformed into a linearly separable dataset with high probability if it is projected to a high dimensional space using a nonlinear transformation.

Since we assume the input feature–sample network is sparse, we can create new features using nonlinear transformation to exploit the results of the Cover's Theorem and, as a consequence, obtain better performance for machine learning algorithms. One way to produce new features is using the *and* operator, which is a nonlinear Boolean function.

We call *and-feature* the new feature v that links to all samples connected to a given subset of two or more preexisting features.

Given a feature–sample network $\mathcal{G}$ with N samples and D features, we can produce an and-feature v for each combination $\mathcal{W}$ of q features such that $|\mathcal{W}| \geq 2$ and $\mathcal{W} \subseteq \{v_{N+1}, \ldots, v_{N+D}\}$. We call q the order of the and-feature v. Thus, the number of possible and-features is

$$\sum_{q=2}^{D} \binom{D}{q} = \sum_{q=2}^{D} \frac{D!}{q!\,(D-q)!} = 2^D - D - 1.$$

In the rest of this paper, we index every possible and-feature using the parameter $\vec{a} = [a_1, \ldots, a_D] \in \{0,1\}^D$ such that $\sum_j a_j \geq 2$. Each element a_j indicates the presence or absence of one of the original features in the composition of a new feature. The feature v_{N+j} is part of the combination if, and only if, $a_j = 1$. Thus, the set $\mathcal{W}$ is $\{v_{N+j} \mid a_j = 1\}$.

Using this notation, we say that the and-feature $v(\vec{a})$ connects to each sample v_i if, and only if, $(\neg a_1 \vee x_{i1}) \wedge \cdots \wedge (\neg a_D \vee x_{iD}) = 1$ holds.

From this discussion, it is easy to observe that enumerating every combination has exponential cost. Moreover, once the network is sparse, we expect that many of the and-features have no connections at all. Also, arbitrarily including and-features is not a valid approach, since it would impair the network.

C. Optimization problem definition

The problem of enhancing the network can be viewed as a optimization problem. Given an input feature–sample network $\mathcal{G}$ with N samples and D features, we denote $\mathcal{G}(\mathcal{Y})$ the enhanced network from the original $\mathcal{G}$ by adding every and-feature $v \in \mathcal{Y}$. The number of features of the enhanced network, excluding the and-features that have no connections, is given by $D(\mathcal{Y})$. Thus, $\mathcal{G} = \mathcal{G}(\varnothing)$ and $D = D(\varnothing)$.

Let $M_{\max}$ be the maximum allowed number of generated features, we want to

$$\begin{aligned} \underset{\mathcal{Y}}{\text{maximize}} \quad & D(\mathcal{Y}) \\ \text{subject to} \quad & D(\mathcal{Y}) - D(\varnothing) \leq M_{\max}. \end{aligned}$$

The disadvantage of this approach is that the and-features might not be well distributed. Thus, while some samples may have few new features, others may have many. To overcome this limitation, we introduce the *disproportion* $\Delta(\mathcal{G}, \mathcal{G}') \in [0, \infty)$ between the network $\mathcal{G}$ and its enhanced version $\mathcal{G}'$.

The disproportion is zero if the number of new connections in each sample is proportional to its initial sparsity. In this way, while the sparsity of each sample might change, we keep the same shape of the degree distribution of the samples.

Let k_i be the degree of the vertex v_i, the disproportion between two networks is

$$\Delta(\mathcal{G}, \mathcal{G}') = \operatorname{sd}\left(\frac{k_1(\mathcal{G}') - k_1(\mathcal{G})}{k_1(\mathcal{G})}, \ldots, \frac{k_N(\mathcal{G}') - k_N(\mathcal{G})}{k_N(\mathcal{G})} \right),$$

where sd is the standard deviation of the arguments.

Using the disproportion in our optimization problem, the goal becomes

$$\begin{aligned} \underset{\mathcal{Y}}{\text{maximize}} \quad & D(\mathcal{Y}), \\ \underset{\mathcal{Y}}{\text{minimize}} \quad & \Delta\big(\mathcal{G}(\varnothing), \mathcal{G}(\mathcal{Y})\big), \\ \text{subject to} \quad & D(\mathcal{Y}) - D(\varnothing) \leq M_{\max}. \end{aligned}$$

III. Methods

In this section, we study the multi-objective problem stated in the previous section and describe the two optimization approaches proposed in this work.

A. Problem study

The number of possible and-features scales exponentially with the number of features D (Section II-B). As a consequence, storing every possible and-feature is not feasible.

Furthermore, the number of candidate solutions is also exponential with the number of possible new features. Precisely, there are at the most

$$\sum_{m=1}^{2^D - D - 1} \binom{2^D - D - 1}{m} = \sum_{m=1}^{2^D - D - 1} \frac{\big(2^D - D - 1\big)!}{m!\,(2^D - D - m - 1)!}$$

solutions $\mathcal{Y}$ to explore. The size of the set $\mathcal{Y}$ cannot be restricted by $M_{\max}$ since many and-features may have no connection.

The three common approaches for solving multi-objective optimization problems are weighted-formula, lexicographic,

and Pareto [11]. The first strategy transforms the problem into a single-objective one, usually by weighting each objective and adding them up. The lexicographic approach assigns a priority to each objective and then optimizes the objectives in that order. When comparing two candidate solutions, the highest-priority objective is compared and, if the evaluations are similar, the second objective is compared. If the second objective is also similar for both solutions, the third one is used, and so on. Both the weighted-formula and the lexicographic strategies return only one solution for the problem. This is an advantage when compared to Pareto methods. A disadvantage for the lexicographic approach is that the priority of the objectives must be provided. Pareto methods use different mathematical tools to evaluate candidate solutions, finding a set of non-dominated solutions. A solution is said to be non-dominated if it is not worse than any other solution concerning all the criteria [11].

We use two objectives for the optimization problem stated in this paper (Section II-C). The first objective clearly is more important than the second one. So, the lexicographic approach is attractive for this problem because it outputs only one solution for each run of the algorithm. On the other hand, the Pareto approach does not require a definition a priori of the relative importance of the objectives. In this way, it is important to compare the lexicographic and Pareto approaches for the optimization of the feature-sample networks.

We design two population-based optimization algorithms. Specifically, we consider the use of two metaheuristics: a) a lexicographic genetic algorithm (LGA); and b) the improved strength Pareto evolutionary algorithm (SPEA2) [12].

Although the methods are different, both approaches share many properties – individual representation, population initialization, operators of mutation, recombination and selection – which are explained in Section III-D. The main difference between them is in the evaluation of the individuals.

In the LGA, the individuals are ordered lexicographically, that is, they are ordered according to the first objective function (number of new features) and, in case of tie, to the second objective (disproportion). SPEA2, however, consider not only the Pareto front but also the density of the solutions. SPEA2 is a popular multi-objective evolutionary algorithm that is attractive for higher dimensional objective spaces [12].

B. Lexicographic genetic algorithm

The pseudo-code of the standard GA is presented in Algorithm 1. Initially, a random population of candidate solutions is generated. Then, while the stop condition is not met, the next population is composed of the best individuals of the previous population and the individuals originated by recombining and mutating parents selected from the previous population [13].

The standard GA and the lexicographic GA are different because during the evaluation of the candidate solution the best individuals are decided lexicographically in the second [14].

In our specific problem, a solution $\mathcal{Y}_1$ is better than solution $\mathcal{Y}_2$ if

- $D(\mathcal{Y}_1) > D(\mathcal{Y}_2)$; or
- $D(\mathcal{Y}_1) = D(\mathcal{Y}_2)$ and $\Delta\big(\mathcal{G}, \mathcal{G}(\mathcal{Y}_1)\big) < \Delta\big(\mathcal{G}, \mathcal{G}(\mathcal{Y}_2)\big)$.

Originally, the lexicographic ordering takes into account a threshold to compare the objective function values. However, given that our first objective function is a discrete value, we opted out of the threshold.

Algorithm 1 Pseudo-code of a standard GA [13].

```
X ← INITIALPOPULATION()
while STOPCONDITION() = false do
    EVALUATE(X)
    X_next ← ELITISM(X)
    while |X_next| < |X| do
        parents ← SELECT(X)
        children ← RECOMBINE(parents)
        MUTATE(children)
        X_next ← X_next ∪ children
    end while
    X ← X_next
end while.
```

C. Improved Strength Pareto Evolutionary Algorithm

SPEA2 works similarly to a traditional GA. The major difference is that it keeps an archive with the candidate solutions for the Pareto set. If the number of non-dominated solutions is greater than the limit of the archive, some solutions are discarded. Such operation is called truncation. The truncation operator tries to maintain the candidate solutions uniformly distributed along the Pareto front [12].

Here, we select individuals by employing binary tournament. Also, let A be the archive size, we fix the parameter $k = \sqrt{A}$ in the truncation operator [12].

D. Metaheuristic design

The common implementation characteristics of our metaheuristic is exposed as follows.

1) Individual representation: In our problem, each solution is a set $\mathcal{Y}$ of zero or more and-features. If we enumerate every possible and-feature, the solution can also be viewed as a binary vector with entries 1 for the present and-features.

2) Population initialization: Given $\mu, \sigma > 0$, we sample, without replacement, $\lfloor M \rfloor$ random and-features to compose each candidate solution $\mathcal{Y}$ such that $M \sim \mathcal{N}(\mu, \sigma)$. And-features are sampled so that the probability of having order $q \geq 2$ is

$$\frac{q-1}{q!}.$$

3) Recombination operator: We use the uniform crossover operator with two parents generating two children. In Algorithm 2, we show how to implement it efficiently using our set representation.

4) Selection operator: The binary tournament method is chosen to select the parents for the recombination step.

5) Mutation operator: Based on other mutation operators for similar problems, we propose in this paper a specific mutation operator to exploit the characteristics of the problem of optimization of the feature-sample network.

Given a candidate solution $\mathcal{Y}$, we apply $\eta \in \{1, 2, \dots\}$ random changes in the individual. For each change, there is a equal probability of either

- trying to add a new and-feature; or
- trying to remove an and-feature $v \in \mathcal{Y}$; or
- trying to modify an and-feature $v \in \mathcal{Y}$.

For the first case, one and-feature v is sampled and the solution is updated to $\mathcal{Y} \cup \{v\}$. Note that $\mathcal{Y} \cup \{v\} = \mathcal{Y}$ if the and-feature v was previously present, that is $v \in \mathcal{Y}$.

For the second case, with probability $(|\mathcal{Y}| + 1)^{-1}$, the individual $\mathcal{Y}$ remains unchanged. With probability $1-(|\mathcal{Y}| + 1)^{-1}$, one and-feature $v \in \mathcal{Y}$ is sampled uniformly, and the candidate solution is updated to $\mathcal{Y} \setminus \{v\}$.

Finally, in the last case, an and-feature $v(\vec{a}) \in \mathcal{Y}$ with order $q = \sum_j a_j$ is selected uniformly to be modified. Once the and-feature is selected, a modified and-feature $v'(\vec{a}')$ will be produced. Two cases may happen: a) with probability $\frac{1}{q}$, an index $j' \in [1, D]$ is selected uniformly, and $\vec{a}'$ is

$$\begin{cases} a'_{j'} = 1 \\ a'_j = a_j \quad \forall j \neq j'; \end{cases}$$

b) with probability $\frac{q-1}{q}$, two indexes $j' \in \{j \mid a_j = 1\}$ and $j'' \in [1, D]$ are chosen uniformly. The modified and-feature is

$$\begin{cases} a'_{j''} = a_{j'} \\ a'_{j'} = a_{j''} \\ a'_j = a_j \qquad \forall j \notin \{j', j''\} \end{cases}$$

The first case will include one more term into the and-feature if $a_{j'} = 0$. The second one swaps two elements of $\vec{a}$ and, effectively, takes effect when $a_{j''} = 0$. The candidate solution is then updated to $(\mathcal{Y} \setminus \{v\}) \cup \{v'\}$. The size $|\mathcal{Y}|$ is never increased, and the candidate solution will be preserved if $v = v'$.

6) Performance considerations: We can view both solutions and and-features as binary vectors. However, the set representation is more practical because of the high space-complexity of the problem. Moreover, there is no need to store entries for and-features that lack connections. Instead of just ignoring them, we exploit the evaluation step to determine which and-features are useless and remove them from the set.

Also, using the set representation, the crossover of the candidate solutions $\mathcal{Y}_1^{\text{parent}}$ and $\mathcal{Y}_2^{\text{parent}}$ can be implemented efficiently as shown in Algorithm 2. First, both children, $\mathcal{Y}_1^{\text{child}}$ and $\mathcal{Y}_2^{\text{child}}$, will have the and-features that the parents share. Then, for each and-feature v that is in either of the parents and not in both, a child is chosen uniformly to have the new feature v. The $\triangle$ in the algorithm stands for the symmetric difference operator.

Finally, in the population initialization, a sampled and-feature is composed by q with probability $\frac{q-1}{q!}$. Using the set representation $\mathcal{W}$ of and-features, one can sample them efficiently using the Algorithm 3. Original features v_{N+j}, $j \in \{1, \dots, D\}$, are sampled with same probability and without replacement. The sampled features are included iteratively in the and-feature $\mathcal{W}$. At each iteration, with probability, $1 - |\mathcal{W}|^{-1}$ the procedure stops.

Algorithm 2 Pseudo-code of the efficient uniform crossover operator using the set representation of the individuals.

```
1: Y1^child, Y2^child ← Y1^parent ∩ Y2^parent
2: for v ∈ Y1^parent △ Y2^parent do
3:     if SAMPLEUNIFORM(0, 1) < 0.5 then
4:         Y1^child ← Y1^child ∪ {v}
5:     else
6:         Y2^child ← Y2^child ∪ {v}
7:     end if
8: end for
```

Algorithm 3 Pseudo-code of the efficient sampling of and-features for the initial population.

```
1: j ← SAMPLE(1, ..., D)
2: W ← {v_{N+j}}
3: loop
4:     j ← SAMPLE(1, ..., D)
5:     W ← W ∪ {v_{N+j}}
6:     if SAMPLEUNIFORM(0, 1) < 1 − |W|^{−1} then
7:         break
8:     end if
9: end loop
```

IV. Experimental results

In this section, we present experiments of our feature learning technique in problems of supervised and unsupervised learning.

We illustrate the optimization process in the famous Iris dataset. In this example, we compare the results of data clustering using the original dataset against using the enhanced one. To solve the clustering problem, we use community detection in both networks.

In a similar way, we analyse the performance gain of the proposed feature learning method in supervised learning tasks. First, feature–sample networks are constructed from other four UCI datasets. Then, we use our optimization approach to enhance each network. The k-nearest neighbors [1] classifier is applied in each scenario. Accuracy results are compared before and after the optimization process.

A. Enhanced community detection and clustering

The UCI Iris dataset [15] contains 150 samples and 4 features. The classes are *Iris setosa*, *Iris virginica*, and *Iris versicolor*. In [7], we built a feature–sample network from this dataset by discretizing the features.

Figure 1 shows the generated network. In the experiments, we use the same network as input for both algorithms, SPEA2 and LGA.

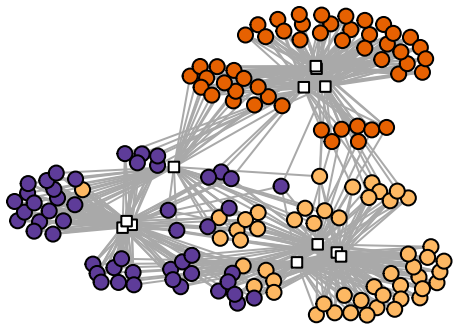

Figure 1. Feature–sample network for Iris dataset. Circles are vertices associated with samples and squares with features. Colors represent the classes. After discretization of the original dataset, each numeric feature becomes 3 binary ones.

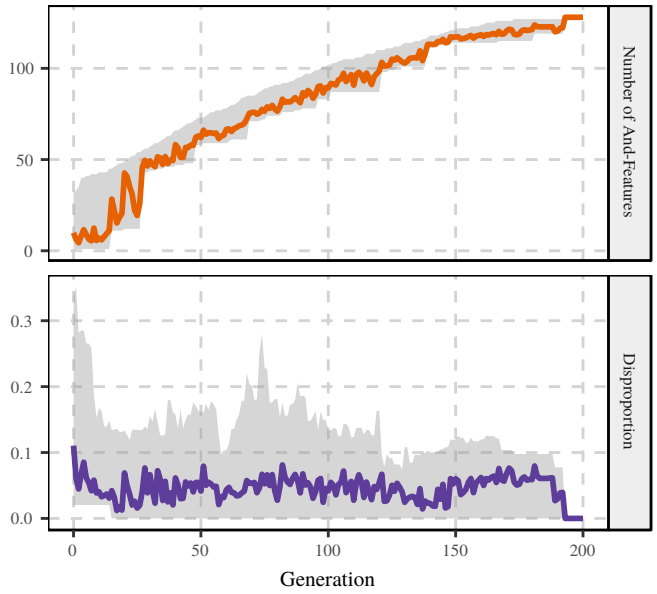

Figure 2. Evolution of the number of and-features and disproportion over time of one execution of the SPEA2 algorithm with Iris dataset as the input network. Solid lines are the average disproportion and number of and-features of the non-dominated solution at a given generation. Shadows cover the range of the measurements.

1) Evolution of the candidate solutions: As an illustration, we execute the optimization process once for each strategy. Such example provides a big picture of the strategies when dealing with our specific optimization problem. Given the characteristics of the machine learning problems involved, it is interesting for us to observe how the candidate solutions evolve over time. With this information, one can choose a strategy given one's time or memory restrictions.

We fix the population in 1000 individuals. For SPEA2, the archive has size 100 and, for the LGA, we keep the 100 best solutions over time. In the initial population, we use $\mu = 10$ and $\sigma = 5$. The recombination rate is 0.6 and the $\eta = 1$ random changes are applied in each generated individual.

Figures 2 and 3 describe the obtained results. Both disproportion and number of discovered and-feature are shown over time. Solid lines are the average result in the population and

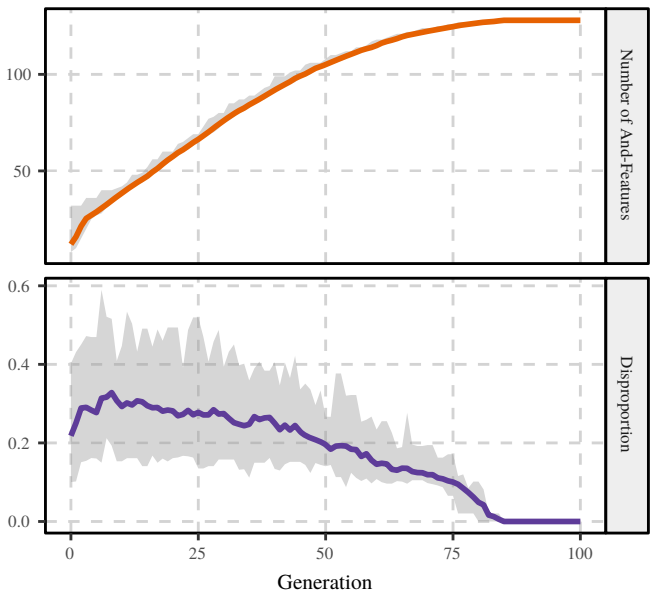

Figure 3. Evolution of the number of and-features and disproportion over time of one execution of the LGA with Iris dataset as the input network. Solid lines are the average disproportion and number of and-features of the best 100 solutions (elitism) at a given generation. Shadows cover the range of the measurements.

shadows cover the range – from minimum to maximum – of each measurement. The results include only the non-dominated solutions in SPEA2 and the 100 best solutions in the LGA.

Using both strategies, we could reach the optimal solution: 128 and-features with at least one connection and 0 disproportion. However, the optimization strategies differ as to how to achieve this.

SPEA2 tries to find as many new and-feature while keeping the ones with lowest values of disproportion. When a larger set of and-features with disproportion 0 is discovered, such solution dominates all solutions found so far. Thus, we observe "steps" in the evolution of the number of and-features.

In LGA, the disproportion is only considered when the number of discovered and-features is the same. As a result, the algorithm greedily produce and-features disregarding the disproportion until it cannot find more new features to add. It enables a faster convergence, but it may find only solutions with high disproportion when it is unfeasible to reach the maximum number of and-feature – which is very common in practice. To solve this issue in larger problems, one can set the limit in the number of new features, $M_{\max}$.

Finally, Figure 4 summarizes the average best results of 30 independent runs with the same parameters. The results are consistent with the discussion above.

2) Clustering and community detection: The optimal enhanced feature–sample network for this dataset is in Figure 5. One can notice that there are many new features and the samples are more clustered.

Applying a greedy community detection method [16] in both networks, the enhanced network has modularity 12.4% ($Q = 0.561$) higher than the input network ($Q = 0.499$.)

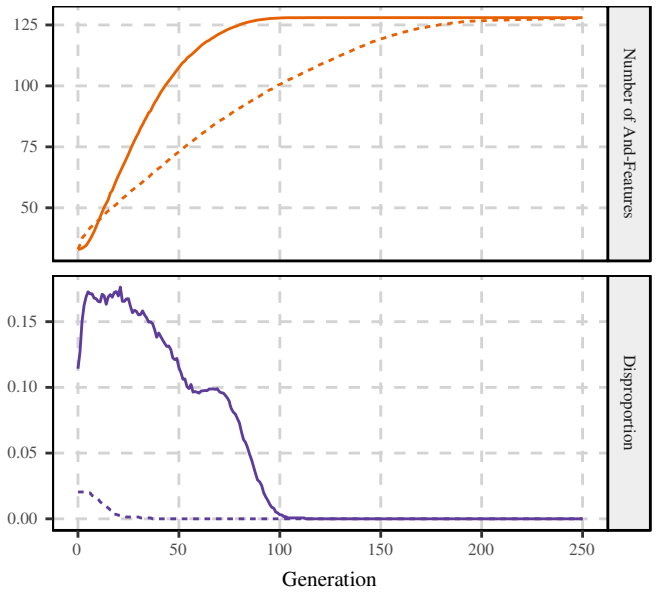

Figure 4. Average number of and-features and disproportion in the Iris dataset. Graphs show the average values of 30 runs for each algorithm over time. In each run, only the best value of each objective is considered. Solid lines correspond to LGA and dashed lines to SPEA2.

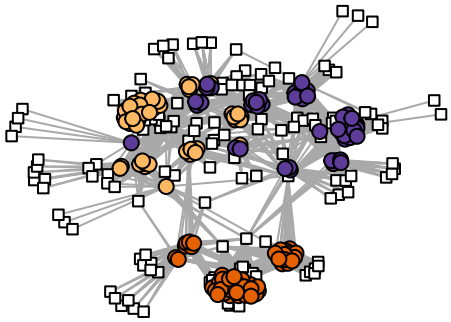

Figure 5. Feature–sample network for Iris dataset with all possible 128 and-features. Circles are vertices associated with samples and squares with features. Colors represent the classes.

The enhanced network can also improve clustering tasks. If comparing the expected class and the obtained communities, the enhanced version achieves higher Jaccard index, 0.731 against 0.719.

B. Performance enhancement in supervised learning

We also apply our proposal in 4 classification tasks from UCI [15]. Table I presents the datasets along with the number of samples N, features D, and possible and-features M. We highlight that it is unfeasible to list every possible combination among the features even for small datasets. The input networks are generated as specified in [7] with 3 bins.

Table I
UCI DATASETS ALONG WITH THE NUMBER OF SAMPLES N, FEATURES D, AND POSSIBLE AND-FEATURE M.

Dataset	N	D	M
Breast 2010	106	27	134,217,700
Ecoli	336	19	524,268
Glass	214	25	33,554,406
Wine	178	39	549,755,813,848

Table II
NUMBER OF AND-FEATURES AND DISPROPORTION OBTAINED BY THE OPTIMIZATION PROCESS FOR BOTH STRATEGIES. AVERAGE AND STANDARD DEVIATION ARE SHOWN FOR EACH MEASUREMENT.

Dataset	LGA	SPEA2
Breast 2010	2700 ± 0, 0.2 ± 0.03	513.3 ± 272.4, 0.02 ± 0.03
Ecoli	1900 ± 0, 0.13 ± 0.01	372.6 ± 268.3, 0.04 ± 0.04
Glass	2500 ± 0, 0.14 ± 0.01	481 ± 399, 0.05 ± 0.04
Wine	3900 ± 0, 0.28 ± 0.03	561 ± 452.8, 0.03 ± 0.02

Table III
NUMBER OF AND-FEATURES, DISPROPORTION, AND STRATEGY OF THE BEST RESULTS.

Dataset	Lowest disproportion	Highest number of and-features
Breast 2010	513, 0.000 (SPEA2)	2700, 0.129 (LGA)
Ecoli	166, 0.011 (SPEA2)	1900, 0.114 (LGA)
Glass	416, 0.019 (SPEA2)	2500, 0.130 (LGA)
Wine	413, 0.015 (SPEA2)	3900, 0.226 (LGA)

The optimization process is executed 15 times for each strategy – SPEA2 and LGA. We fix the population size in 1000, the archive size in 100, and the elitism 100 solutions. For the initial population, we use $\mu = 50$ and $\sigma = 10$. The recombination rate is 0.6 and $\eta = 1$ mutation is performed for each candidate solution. We limit the number of and-feature by $M_{max} = 100D$. The execution is stopped at the 1000th generation.

Table II summarizes the number of and-features and disproportion obtained by the optimization process. For the LGA, we show the average and the standard deviation of the measurements among the 100 best individuals. For SPEA2, only the non-dominated solutions are considered.

As expected by considering the previous study (Section IV-A,) the LGA achieved better count of and-features – the maximum allowed –, but worse values of disproportion. The candidate solutions of SPEA2 present wide variation, but consistent lower disproportion.

For each one of the datasets, we take the candidate solution with highest count of and-features and with the lowest disproportion among every solution produced. (Solutions with less than 100 and-features are ignored.) Such candidate solutions, and the strategy that has found them, are indicated in Table III.

To estimate the improvements given by the new features, we solve the classification problems for each one of the datasets using the k-nearest neighbors method. As inputs we use the interaction matrices of the original network and the enhance ones from the selected candidate solutions. We performed 20 independent holdout validations with 60%, 70% and 80% of labeled samples for each case. We varied $k \in \{1, 2, \ldots, 20\}$.

The best results for each configuration are shown in Table IV. Improvements are highlighted in bold. We see higher

Table IV
ACCURACY OF THE k-NN METHOD ON THE UCI DATASETS USING THREE DIFFERENT INPUT NETWORKS – ORIGINAL, LOWEST DISPROPORTION, AND HIGHEST NUMBER OF AND-FEATURES. FOR EACH SETTING, THE BEST VALUE OF k IS SHOWN.

Dataset	Original	(k)	Lowest disproportion	(k)	Highest number of and-features	(k)
			60% labeled			
Breast 2010	0.63 ± 0.06	(1)	0.62 ± 0.06	(1)	$\mathbf{0.65 \pm 0.06}$	(1)
Ecoli	0.76 ± 0.03	(4)	0.76 ± 0.03	(4)	0.76 ± 0.03	(11)
Glass	0.66 ± 0.05	(5)	0.68 ± 0.05	(7)	$\mathbf{0.69 \pm 0.06}$	(7)
Wine	0.92 ± 0.02	(15)	$\mathbf{0.93 \pm 0.03}$	(9)	$\mathbf{0.93 \pm 0.02}$	(3)
			70% labeled			
Breast 2010	0.62 ± 0.07	(1)	0.63 ± 0.07	(1)	$\mathbf{0.64 \pm 0.06}$	(1)
Ecoli	0.76 ± 0.03	(4)	$\mathbf{0.77 \pm 0.03}$	(4)	0.76 ± 0.04	(12)
Glass	0.66 ± 0.06	(7)	0.69 ± 0.05	(6)	$\mathbf{0.71 \pm 0.05}$	(8)
Wine	0.92 ± 0.03	(17)	$\mathbf{0.93 \pm 0.03}$	(3)	$\mathbf{0.93 \pm 0.02}$	(5)
			80% labeled			
Breast 2010	0.65 ± 0.11	(5)	0.62 ± 0.10	(4)	$\mathbf{0.66 \pm 0.10}$	(3)
Ecoli	0.77 ± 0.03	(6)	$\mathbf{0.78 \pm 0.04}$	(4)	$\mathbf{0.78 \pm 0.04}$	(13)
Glass	0.66 ± 0.07	(7)	0.69 ± 0.06	(6)	$\mathbf{0.72 \pm 0.07}$	(9)
Wine	0.93 ± 0.05	(19)	$\mathbf{0.94 \pm 0.05}$	(3)	$\mathbf{0.94 \pm 0.03}$	(3)

improvements using the solutions with the highest number of and-features than using those with lower disproportion.

V. CONCLUSION

In this paper, we presented an unsupervised feature learning mechanism that works on datasets with binary features. First, the dataset is mapped into a feature–sample network. Then, a multi-objective optimization process selects a set of new vertices that correspond to new features to produce an enhanced version of the network.

We show that the enhanced network contains more information and can be exploited to improve the performance of machine learning methods.

To solve the optimization problem, we designed population-based metaheuristics. We used both a LGA and the SPEA2 algorithm to find the candidate solutions.

From the experiments, we conclude that the LGA produces more new features in fewer generations. However, candidate solutions in SPEA2, besides having less new features, also improved the performance of machine learning methods.

We realize that the disproportion is a good measurement of the quality of the selected set of and-features. In future works, we will correlate improvement and disproportion of the solutions with the same number of features. Regarding the optimization phase, we will use other well-known metaheuristics, such as NSGA-II, and different mutation operators to compare with the presented approaches.

Furthermore, the learning techniques used – fast-greedy community detection and k-nearest neighbors – do not take full advantage of the new features. In subsequent studies, we will elaborate learning models to exploit the enhanced feature–sample network explicitly. The explicit knowledge of the new features may increase the performance gain.

ACKNOWLEDGMENTS

This research was supported by the São Paulo State Research Foundation (FAPESP) and the Brazilian National Research Council (CNPq).

REFERENCES

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.
[2] X. Zhu and A. B. Goldberg, "Introduction to Semi-Supervised Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.
[3] T. C. Silva, L. Zhao, and T. H. Cupertino, "Handwritten data clustering using agents competition in networks," *Journal of Mathematical Imaging and Vision*, vol. 45, no. 3, pp. 264–276, 2013.
[4] T. C. Silva and L. Zhao, *Machine Learning in Complex Networks*, 1st ed. Springer, 2016.
[5] F. A. N. Verri, P. R. Urio, and L. Zhao, "Network unfolding map by vertex-edge dynamics modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–14, 2016.
[6] M. E. J. Newman, *Networks: An Introduction*, 1st ed. New York, NY: Oxford University Press, 2010.
[7] F. A. N. Verri and L. Zhao, "Random walk in feature-sample networks for semi-supervised classification," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, 2016, pp. 235–240.
[8] J. Mũnoz Marí, F. Bovolo, L. Gómez-Chova, L. Bruzzone, and G. Camp-Valls, "Semisupervised One-Class Support Vector Machines for Classification of Remote Sensing Data," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 8, pp. 3188–3197, 2010.
[9] 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.
[10] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Transactions on Electronic Computers*, vol. EC-14, no. 3, pp. 326–334, 1965.
[11] A. A. Freitas, "A critical review of multi-objective optimization in data mining: A position paper," *SIGKDD Explor. Newsl.*, vol. 6, no. 2, pp. 77–86, 2004.
[12] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*. Athens, Greece: International Center for Numerical Methods in Engineering, 2001, pp. 95–100.
[13] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.
[14] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.
[15] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml
[16] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Physical Review E*, vol. 70, no. 6, pp. 1–6, 2004.