Semi-Supervised Clustering with Multiresolution Autoencoders

Dino Ienco
TETIS, IRSTEA, Univ. Montpellier
LIRMM
Montpellier, France
dino.ienco@irstea.fr

Ruggero G. Pensa
Department of Computer Science
University of Turin
Turin, Italy
ruggero.pensa@unito.it

***Abstract*—In most real world clustering scenarios, experts generally dispose of limited background information, but such knowledge is valuable and may guide the analysis process. Semi-supervised clustering can be used to drive the algorithmic process with prior knowledge and to enable the discovery of clusters that meet the analyst's expectations. Usually, in the semi-supervised clustering setting, the background knowledge is converted to some kind of constraint and, successively, metric learning or constrained clustering are adopted to obtain the final data partition. Conversely, we propose a new semi-supervised clustering algorithm that directly exploits prior knowledge, under the form of labeled examples, avoiding the necessity to derive constraints. Our algorithm employs a multiresolution strategy to generate an ensemble of semi-supervised autoencoders that fit the data together with the background knowledge. Successively, the network models are employed to supply a new embedding representation on which clustering is performed. The proposed strategy is evaluated on a set of real-world benchmarks also in comparison with well-known state-of-the-art semi-supervised clustering methods. The experimental results highlight the benefit of directly leveraging the prior knowledge and show the quality of the representation learnt by the multiresolution schema.**

***Index Terms*—semi-supervised clustering, background knowledge, autoencoders, ensemble**

I. Introduction

Clustering is by far one of the most popular machine learning techniques due to the wide range of unsupervised application settings [14]. Although unsupervised problems are very common, analysts/data scientists are often unsatisfied of clustering algorithms, since their expectation is frequently violated by the results. Indeed, some (limited) knowledge about the data is likely to be owned by the expert who may know the expected cluster structure of few samples of interest. Semi-supervised clustering [13] addresses exactly this problem: by driving the algorithmic process with prior knowledge, it enables the discovery of clusters that meet the analyist's expectations.

Prior knowledge may come in form of known cluster labels [27] or pairwise constraints [25], i.e., a set of must-link and cannot-link constraints that state whether two data examples should be in the same cluster or not. In the former setting, usually adopted in semi-supervised classification scenarios, side information is kept in the form of labels. The known labels are propagated to the unlabeled data samples and the prediction is usually evaluated directly on the labels [28], [30]. In the latter — more popular — scenario, semi-supervised clustering is often referred to as constraint-based or constrained clustering [4]. Most research works address the problem of semi-supervised clustering inducing pairwise constraints from the background knowledge. Such constraints are successively exploited by either learning a distance metric [9], [15], [16] or forcing constraints during the clustering process [25], [26], although the most effective methods usually combine both strategies [3], [5], [18]. However, it has been shown that using labels is equivalent to converting them into constraints [27]. In addition, labels are more expressive than pairwise constraints (a set of pairwise constraints may not correspond to a unique labeling). Thus, in this paper, we propose a semi-supervised clustering method that directly processes labels by skipping the unnecessary conversion step.

In particular, our contribution consists in a semi-supervised clustering technique based on semi-supervised autoencoders. Autoencoders are usually adopted to learn a low-dimensional representation of the data via an encoding-decoding schema. In addition, in a semi-supervised autoencoder, the bottleneck layer is also trained to deal with the prediction task. Furthermore, inspired by image processing and remote sensing, we leverages a multiresolution strategy to perform clustering by training multiple autoencoders of different sizes. The intuition behind this choice is that autoencoders at multiple resolutions capture better the multifaceted diversity relationships among attributes during the clustering process. We assess the effectiveness of our framework by comparing its performances to several competitors' ones. In our experimental study, we show that our algorithm outperforms state-of-the-art semi-supervised clustering techniques, whether they are based on pairwise constraints, on metric-learning or on both approaches.

The remainder of the paper is organized as follows: Section II presents some closely related works; Section III introduces the theoretical foundations of our framework; we describe our semi-supervised clustering method in Section IV and report the experimental results in Section V; finally, Section VI concludes and provides some ideas for future research directions.

II. Related work

Semi-supervised clustering is a fifteen years old albeit still very active research field (see, e.g., [4] for a state-of-the-art survey). It supports classification tasks when labeled data are limited and/or expensive to collect. As such, it has been mainly studied in the context of semi-supervised learning where two alternative classes of methods have been studied: metric-based methods, consisting in learning a metric considering labeled data before applying standard clustering, and constraint-based methods, which force pairwise (e.g., must-link and cannot-link) constraints satisfaction during the clustering process. A solution is to use the knowledge provided by the few available labeled instances within a clustering algorithm. In [26], a simple adaptation of k-means which enforces must-link and cannot-link constraints during the clustering process is described. [2] proposes a constrained clustering approach that leverages labeled data during the initialization and clustering steps. An example of metric-based approach is given in [16]. Instead, [5] integrates both constraint-based and metric-based approaches in a k-means-like algorithm. In [3], the authors propose a probabilistic model for semi-supervised clustering, which also combines constraints and metric learning. [27] proposes to exploit labeled examples to generate pairwise constraints via a label propagation process. The derived constraints are successively integrated in a constrained spectral clustering algorithm. Davis *et al.*, instead, propose an information-theoretic approach to learning a Mahalanobis distance function [9]. They leverage a Bregman optimization algorithm [1] to minimize the differential relative entropy between two multivariate Gaussians under constraints on the distance function. This approach has been recently extended by Nogueira *et al.*, who combine distance metric learning and cluster-level constraints [18], while in [15] the authors propose an integration of kernelization technique with Mahalanobis-based distance metric learning.

In more recent works, Zhu *et al.* present a pairwise similarity measure framework to perform a more effective constraint diffusion and handle noisy constraints [29], whereas Ganji *et al.* introduce a Lagrangian constrained clustering algorithm that gives high priority to satisfying constraints [10]. Recent research has also addressed scalability issues. For instance, in [7], the authors present a fast constrained spectral clustering approach based on a generalized eigenvalue problem in which both matrices are graph Laplacians.

In our work, we address the semi-supervised clustering problem using semi-supervised autoencoders. Autoencoders [20] are frequently used in unsupervised learning and dimensionality reduction but very few research works use them in semi-supervised settings. In [19], the authors propose a model trained to simultaneously minimize the sum of supervised and unsupervised cost functions by backpropagation. Gogna and Majumdar [11], instead, propose a stacked architecture that acts as a standard unsupervised autoencoder for unlabeled data, while learning a linear classifier for labeled data. Contrary to these works, which are intended as a way to improve classification when few labels are available, we specifically focus on semi-supervised clustering by leveraging an adaptive learning strategy to train an ensemble of stacked semi-supervised autoencoder architectures.

III. Semi-supervised clustering with autoencoders

In this section, we provide the theoretical foundations of our semi-supervised clustering approach. Before presenting the core part of our method, we introduce some notations and preliminaries.

A. Semi-supervised clustering

In a typical semi-supervised learning scenario there are two different sets of examples: a set $X^u = \{x_i\}_{i=1}^N$ of N instances with no available class information, and a set $X^l = \{(x_j, y_j)\}_{j=1}^M$ of M instances for which the class information is available. In particular, each data instance $x_j \in X^l$ is associated to a class variable $y_j \in C$, where C the set of possible labels. The general assumption is that $|X^u| >> |X^l|$ with the extreme (and more realistic) case where only very few examples (e.g., one) per class exist.

The goal of semi-supervised clustering is to group together data examples belonging to $X = X^u \cup X^l$ in k clusters exploiting as much as possible the knowledge provided by the labeled set X^l.

B. Autoencoders

Figure 1(a) visually summarizes the structure of a generic autoencoder. An autoencoder is a particular kind of feed-forward multi-layer neural networks that performs successive linear (or non linear) transformations with the aim of reconstructing the original data. An autoencoder is composed of two parts: i) an encoder that compresses the original data into a low-dimensional representation, and ii) a decoder that reconstructs the original data from the low-dimensional representation. The outputs of the smallest layer constitute the low-dimensional representation of the original data. This layer is usually named bottleneck layer (the central layer in Figure 1(a)).

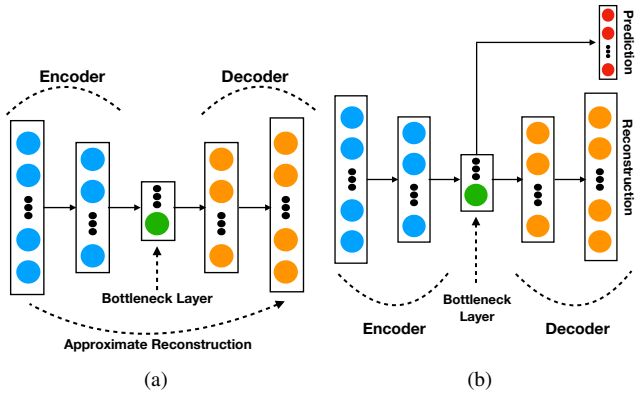
Formally, the autoencoder optimizes the following Loss funciton:

$$L_{ae}(\Theta_1, \Theta_2) = \frac{1}{|X|} \sum_{i=1}^{|X|} ||X_i - AE(X_i, \Theta_1, \Theta_2)||^2 \quad (1)$$

where $||\cdot||$ is the L^2 norm, Θ_1 and Θ_2 are the parameters of the encoder (resp. decoder) part of the autoencoder, and *AE* is the function implemented by the autoencoder. The goal is to train the model ($AE(X, \Theta_1, \Theta_2)$) with the aim of reconstructing the input X as closely as possible.

In a typical autoencoder structure, the number of nodes in the output layer is the same as in the input layer and the network structure is layered and symmetric. In the encoder part, the number of neurons in the internal layers decrease gradually. Symmetrically, it increases in the decoder part. Therefore, the only way to reconstruct the original data accurately is to learn

Θ_1, Θ_2 so that the encoding-decoding process achieves good data compression and reconstruction abilities.



(a) (b)

Fig. 1. Layered architecture of an autoencoder (a) and of a semi-supervised autoencoder (b).

C. Semi-supervised autoencoders

A semi-supervised autoencoder (SSAE) [11], [19] is a particular kind of neural network architecture that solves two different tasks at the same time: i) a data reconstruction task via a classic encoding-decoding schema and ii) a classification task through the encoding part of the network. In this work, we employ a semi-supervised autoencoder in which the bottleneck layer of the autoencoder also deals with the prediction task. Figure 1(b) visually presents our semi-supervised autoencoder. In addition to the reconstruction task (Equation 1), a part of the parameters is also optimized to address the classification task. In particular, the loss function for the classification task is the categorical cross-entropy between the labels and the prediction of the SSAE:

$$L_{cl}(\Theta_1, \Theta_3) = -\frac{1}{|X|} \sum_{j=1}^{|X|} \sum_{c=1}^{|C|} y_{jc} \cdot log(\hat{y}_{jc}) \tag{2}$$

where $y_{j\cdot}$ is the one hot encoding of the class label for the example j, $\hat{y}_{j\cdot} = SSAE(X_j, \Theta_1, \Theta_3)$ is the probability distribution of the SSAE prediction over the set of possible labels, Θ_1 are the parameters of the encoder (the same as in Equation 1), and Θ_3 are the parameters used to perform classification starting from the bottleneck layer of the autoencoder.

The overall loss function optimized by the semi-supervised autoencoder is then:

$$L_{SSAE}(\Theta_1, \Theta_2, \Theta_3) = L_{ae} + \lambda L_{cl} \tag{3}$$

where

$$L_{ae} = \frac{1}{|X|} \sum_{i=1}^{|X|} ||X_i - SSAE(X_i, \Theta_1, \Theta_2)||^2 \tag{4}$$

$$L_{cl} = -\frac{1}{|X^l|} \sum_{j=1}^{|X^l|} \sum_{c=1}^{|C|} y_{jc} \cdot log(\hat{y}_{jc}) \tag{5}$$

In our architecture, the two loss functions involve two sets of data: L_{ae} is trained on the whole dataset X, while L_{cl} is learnt by exploiting only the labeled subset X_l. This supports the learning of low-dimensional representations that well summarize the information carried by the original data while taking into account the background knowledge supplied by the few available labeled examples. In the following, we provide the details of the procedure we employ for the optimization of Equation 3.

Algorithm 1 Semi-supervised autoencoder optimization

Require: X^u, X^l, N_EPOCHS, fl_size, b_size
Ensure: $\Theta_1, \Theta_2, \Theta_3$.
1: i = 0
2: $X = X^u \cup \{x|(x,y) \in X^l\}$
3: $initSSAE(fl\_size, b\_size)$
4: **while** i < N_EPOCHS **do**
5: Update Θ_1 and Θ_2 by descending the gradient:
6: $\nabla_{\Theta_1,\Theta_2} \frac{1}{|X|} \sum_{i=1}^{|X|} ||X_i - SSAE(X_i, \Theta_1, \Theta_2)||^2$
7: Update Θ_1, Θ_2 and Θ_3 by descending the gradient:
8: $\nabla_{\Theta_1,\Theta_2,\Theta_3} \frac{1}{|X^l|} \sum_{j=1}^{|X^l|} ||X_j - SSAE(X_j, \Theta_1, \Theta_2)||^2 - \lambda\Big(\frac{1}{|X^l|} \sum_{j=1}^{|X^l|} y_j \cdot log(SSAE(X_j, \Theta_1, \Theta_3))\Big)$
9: i = i + 1
10: **end while**
11: **return** $\Theta_1, \Theta_2, \Theta_3$

D. Semi-Supervised autoencoder structure and optimization

The internal structure of our network uses Rectifier Linear Units (*ReLU* [17]) as activation functions for all the encoder and decoder layers with the exception of the last layer. In the latter, in fact, we employ a sigmoid activation function. To this purpose, all data attributes are normalized in the range $[0, 1]$ before feeding the network. As regards the classification task, we link the bottleneck layer of the autoencoder to the classification output by a linear activation function followed by a softmax.

The optimization strategy we adopt is reported in Algorithm 1. During each epoch, the algorithm performs the minimization of the reconstruction loss involving Θ_1 and Θ_2 parameters on the whole dataset (line 5-6) and the minimization of the reconstruction and classificaton loss involving Θ_1, Θ_2 and Θ_3 parameters for the subset of labeled instances (line 7-8). The optimization is realized via the use of mini-batches.

IV. The *MSAEClust* model

In this section, we present the details of our Multiresolution Semi-supervised AutoEncoder-based Clustering, namely *MSAEClust*. It follows a multiresolution schema sketched in Algorithm 2. The intuition behind *MSAEClust* is related to the low-dimensional representation supplied by a generic semi-supervised autoencoder. As we discussed before, a semi-supervised autoencoder produces a low-dimensional embedding of the original data addressing a reconstruction task (on the whole dataset) and a classification task (on the set of labeled examples) at the same time. Following the general idea of ensemble learning [12] in which a committee is preferred to one single model, we leverage an ensemble of semi-supervised autoencoders with the aim of computing different low-dimensional embeddings. However, directly combining a

Algorithm 2 MultiResolution Strategy

Require: X^u, X^l, ENS_SIZE
Ensure: X_{newR}.
1: $X_{newR} = \emptyset$
2: $n\_attrib = getNumAttributes(X^u)$
3: $i = 0$
4: **while** i < ENS_SIZE **do**
5: $fl\_size = random(n\_attrib/2, n\_attrib)$
6: $b\_size = random(n\_attrib/4, n\_attrib/2)$
7: $SSAE = buildSSAE(X^c, X^l, fl\_size, b\_size)$
8: $current\_emb = extractEmbedding(SSAE, X^u, X^l)$
9: $X_{newR} = juxtapose(X_{newR}, current\_emb)$
10: $i++$
11: **end while**
12: **return** X_{newR}

set of semi-supervised autoencoders with exactly the same network structure will generate very similar low-dimensional representations that would not be helpful to the semi-supervised clustering task. This lack of diversity is unhelpful from the point of view of variance reduction. To address this issue, in our strategy we use several models at different resolutions to enforce diversity among the different embeddings learnt by our approach. Diversity is deemed to be a key property in the conception of ensemble learning schema and it is crucial to ameliorate the performances of ensemble methods [6]. Changing the size of the different hidden and bottleneck layers results in different granularities of the new learnt representations. The embeddings have different sizes highlighting different coexisting properties of the original data: low-dimensional embeddings capture coarse relationships among the data, while higher-dimensional ones capture finer-grained interactions.

In order to enforce diversity in *MSAEClust*, we randomly sample the size of the different layers in each of the semi-supervised autoencoders composing the ensemble. For each model, we have four layers. The last one has the same dimensionality of the input layer and the penultimate layer has the same dimensionality of the first (hidden) layer. This means that, for each model we need to sample two values, one for the size of the first/penultimate layer and one for the size of the bottleneck layer. The random layer sampling proceeds as follows. We assume that our input data has a dimensionality equals to d and we force the size of the first (penultimate) layer to be sampled uniformly in the interval $[d/2, d)$. Similarly, the size of the bottleneck layer is drawn uniformly in the interval $[d/4, d/2)$. In addition, if the value $d/4$ is lower than 3, the size of the layer is set to 3. This is done to avoid an excessive level of compression in the bottleneck layer so that the data cannot be properly reconstructed. The intervals $[d/2, d)$ and $[d/4, d/2)$ are fixed in this way because their aim is to define the two biggest (and not overlapping) ranges from which the hidden layer sizes can be sampled.

Once the set of semi-supervised autoencoders (at different resolutions) is built, the algorithm proceeds by training such models on the original data and, successively, by stacking together all the different low-dimensional representations in order to obtain the new data embedding (line 4-9 of Algorithm 2). The final partition is obtained by applying *k-means* on the stacked representation.

It is worth noting that our strategy is fundamentally different from *dropout* [21] and other similar methods. Our ensemble strategy is built on a completely independent (different) set of neural networks and the results are successively combined. The main objective of *dropout*, instead, is to mask the different connections among neurons with the aim of avoiding co-adaptation of the weights within the same network. The idea behind our framework is to use an ensemble to increase the diversity through several network structures that represent the different granularities of the relationships existing in the original data.

V. Experiments

In this section we report the results of several experiments we perform to assess the behavior and the performances of our algorithm. First, we compare *MSAEClust* to other semi-supervised clustering methods. Then, we perform an in-depth analysis of *MSAEClust*'s results to assess its sensitivity w.r.t. to the ensemble size. Finally, we investigate qualitatively the new embedding generated by our algorithm through visual inspection.

A. Competitors

For the comparative study, we consider the following competitors: Information Theoretic metric learning (*ITML*), pairwise constrained k-means (*PCKmeans*), metric-pairwise constrained k-means (*MPCKmeans*), and seeded k-means (*SeededKmeans*). *ITML* exploits information theory to learn a suitable metric space that meets the supervision supplied as side-information on the distance function [9]. Since it only supplies a new metric space, we use *k-means* to successively obtain the data partition. *PCKmeans* leverages must-link and cannot-link constraints to modify the internal objective function [3]. *MPCKmeans* combines metric-learning and constrained clustering to exploit as much as possible the supplied supervision [5]. *SeededKmeans* uses labeled instances to initialize a standard *k-means* clustering algorithm [2]. Finally, we also include *k-means* in the comparative study, so as to have a simple fully unsupervised baseline.

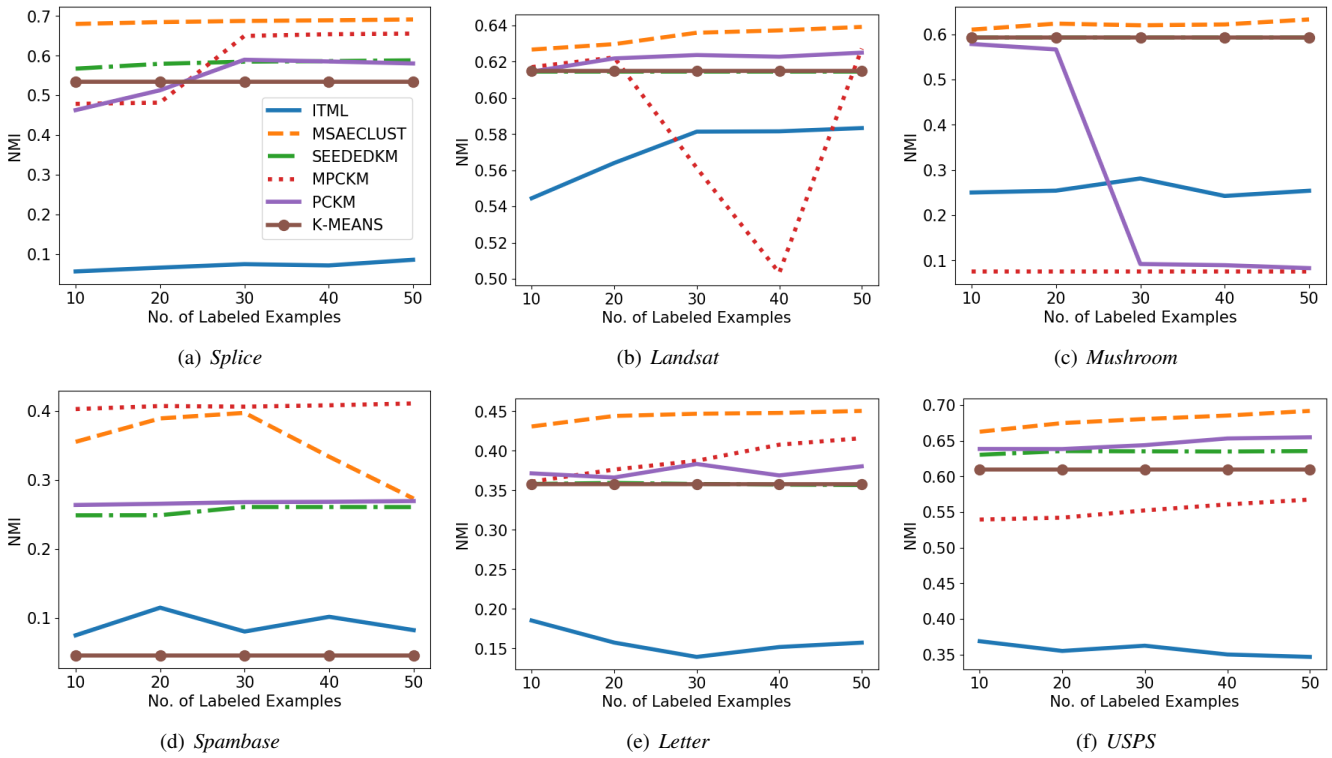
B. Datasets

To evaluate the comparative and in-depth analysis results, the experiments are performed over six publicly available datasets published in the UCI machine learning repository[1].

TABLE I
Main characteristics of the datasets

Dataset	# Instances	# Features	# Classes
Splice	3190	60	3
Landsat	4435	36	6
Mushroom	8124	22	2
Spambase	4601	57	2
Letter	20000	16	26
USPS	9298	256	10

[1] http://archive.ics.uci.edu/ml/index.php



(a) *Splice* (b) *Landsat* (c) *Mushroom*

(d) *Spambase* (e) *Letter* (f) *USPS*

Fig. 2. NMI values of the comparative study *(best viewed in color version, available in electronic format).*

A summary of the information about the datasets is shown in Table I, where, for each dataset, we report the number of instances, the number of features, and the number of classes. These datasets have been chosen because they exhibit a variety of properties in terms of type of tasks/data, number of objects and number of features.

C. Experimental settings

To measure the clustering performances of all the competitors, including *MSAEClust*, we use the Normalized Mutual Information (NMI) [22]. This measure takes its maximum value when the clustering partition completely matches the original one, i.e., the partition induced by the available class labels. NMI may be considered as an indicator of the purity of the clustering result.

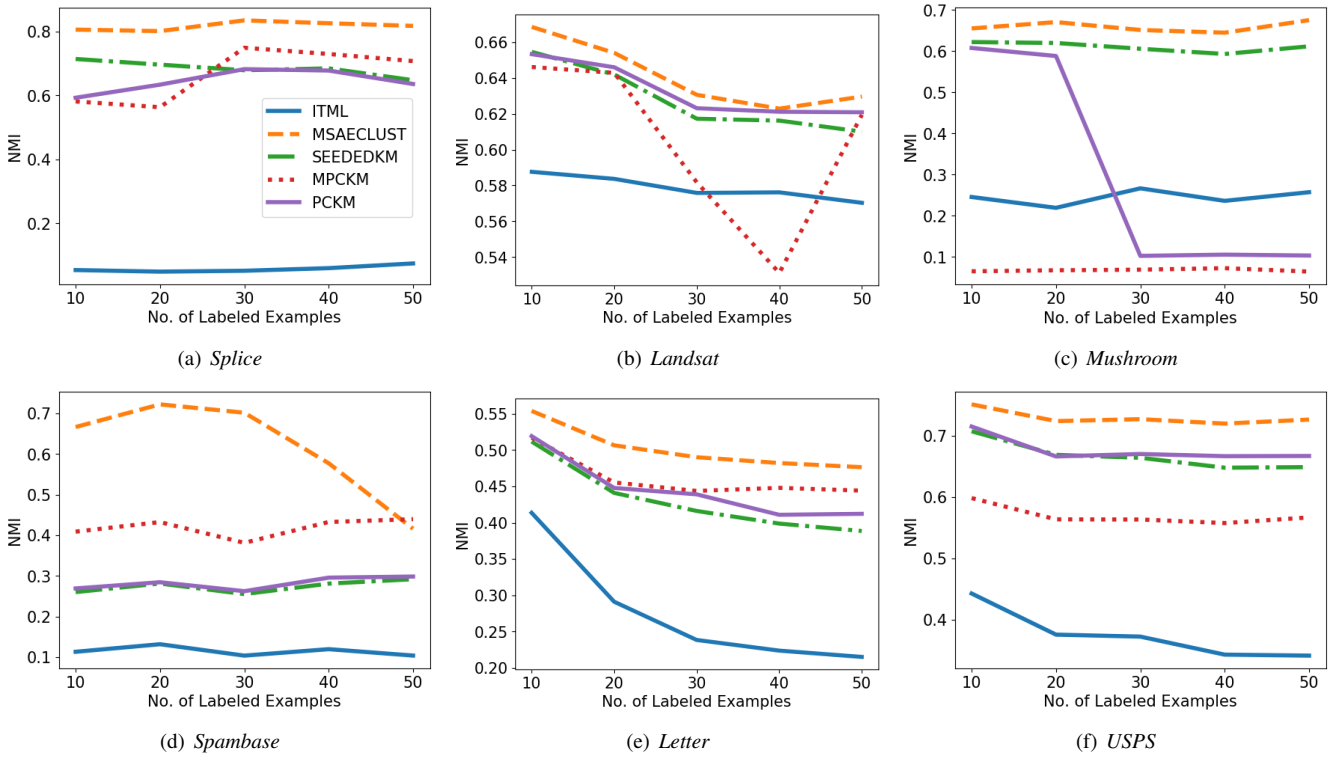
We analyze the behavior of the different methods according to increasing levels of supervision. More in detail, we vary the number of labeled examples per class from 10 to 50, with a step of 10. For *ITML*, *MPCKmeans* and *PCKmeans*, the labeled examples are used to generate the necessary constraints. Due to the randomness of the sample selection process and the non deterministic nature of the clustering algorithms, we repeat the sample selection step 30 times for each number of per-class labels and, successively, we repeat the clustering process 30 times. In practice, for each combination of dataset, method and number of labeled samples, the statistics are computed on 900 runs: 30 random sample selections *times* 30 repetitions of the clustering process. Finally, we report the average values of NMI.

MSAEClust is implemented via the *Keras* python library[2] with *Theano*[3] as back end. For the comparison, we set the ensemble size equal to 30. To train the model, we set the learning rate to 5×10^{-4} with a decay factor of 5×10^{-5}. We use the *RMSProp*[4] optimizer [23] to learn the parameters of the models. The basic idea of this strategy is to divide the gradient by a running average of its recent magnitude. For all the datasets, the models are trained for 200 epochs with a batch size equals to 16 (resp. 8) for the reconstruction (resp. classification) task. Finally, we fix hyperparameter λ equal to 1. Experiments are carried out on a workstation equipped with an Intel(R) Xeon(R) E5-2667 v4@3.20Ghz CPU, with 256Gb of RAM and one TITAN X GPU. The implementation of our approach is publicly available online [5].

D. Comparative study

Figure 2 reports the performances of the different approaches varying the level of labeled examples over the benchmarks. At first glance, we can observe that *MSAEClust* outperforms the competitors for all the different values of labeled examples most of the time (5 datasets over 6). The only dataset for which *MSAEClust* does not obtain the best performances is *Spambase*). However, we observe that its performances are in general close to those achieved by *MPCK-means* (the leading algorithm for this dataset) and always

[2]https://github.com/fchollet/keras
[3]http://deeplearning.net/software/theano/
[4]http://climin.readthedocs.io/en/latest/
[5]https://github.com/tanodino/Semi_Supervised_Auto_Encoder



(a) *Splice* (b) *Landsat* (c) *Mushroom*

(d) *Spambase* (e) *Letter* (f) *USPS*

Fig. 3. NMI values considering only the instances involved in the background knowledge *(best viewed in color version, available in electronic format).*

better than those obtained by the rest of competitors. Note that, when the number of labeled instances is equal to 30, the average NMI value of *MSAEClust* is very close to the one obtained by *MPCKmeans*. Then, with 40 and 50 labeled samples per class, the performances of our algorithm degrade significantly (although it still outperforms the other methods). This is not that strange, indeed. Most semi-supervised clustering algorithms are affected by fluctuating response to constraints (see, for instance, *MPCKmeans* in Figure 2(b) and *PCKmeans* in Figure 2(c)).

As second general remark, we observe that our approach is the only one that always outperforms the fully unsupervised baseline (the *k-means* algorithm). This result stresses the ability of *MSAEClust* to systematically leverage the background information w.r.t. a fully unsupervised clustering process. We point out that this is not the case for any of the other methods considered in this study: *ITML* only outperforms the baseline on *Spambase* data; *PCKmeans* fails to perform better than baseline on *Mushroom*; *MPCKmeans* has issues regarding the results on *USPS*; finally, *SeededKmeans* has the same behavior of its fully unsupervised counterpart on three datasets over six (*Landsat*, *Mushroom* and *Letter*).

To better understand how much each method complies with the injected background knowledge, we compute the NMI by only considering the labeled examples in an experiment conducted similarly to the previous one. Logically, for this analysis, we discard the *k-means* algorithm. The results are reported in Figure 3. The behavior we obtain is coherent with the results we have observed in the previous comparison. Generally, the more the algorithm fits the background knowledge, the more it achieves good performances on the whole benchmark data. The only exception concerns dataset *Spambase*. In this case, *MSAEClust* is the method that best fits the background knowledge but, overall, the best performances on the whole dataset are obtained by *MPCKmeans* (Figure 2(d)). As explained before, this is not unusual in constraint-based clustering, and it is even more noticeable by observing Figure 3: higher quantities of background knowledge do not guarantee better performances on the data. This is not unexpected. In semi-supervised clustering, it is well-known that the quality of the background knowledge is more important than its quantity [8].

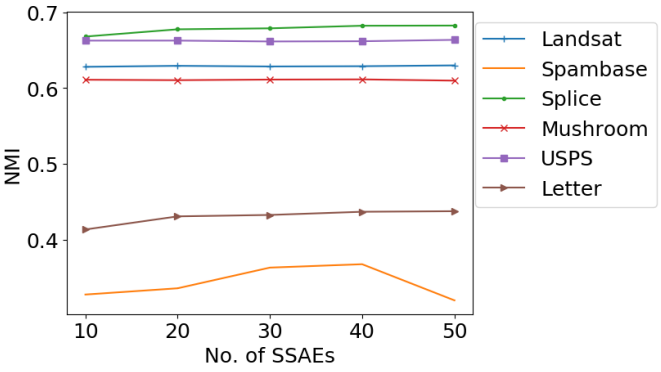
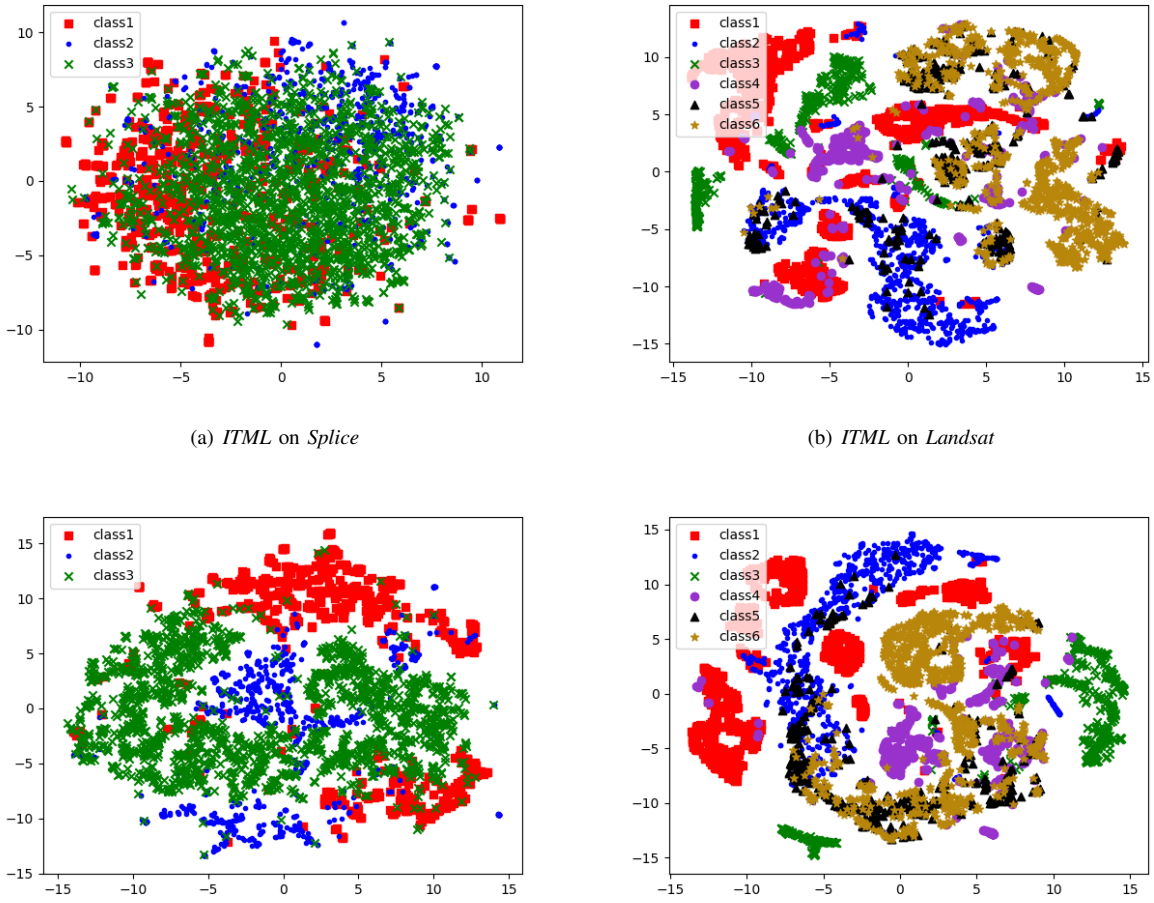


Fig. 4. Sensitivity of *MSAEClust* w.r.t. the ensemble size for all benchmark datasets.



(a) *ITML* on *Splice*

(b) *ITML* on *Landsat*

(c) *MSAEClust* on *Splice*

(d) *MSAEClust* on *Landsat*

Fig. 5. Visual projection of the embeddings learnt by ITML and *MSAEClust* on *Splice* and *Landsat* data. *(Best viewed in color version, available in electronic format.)*

E. Sensitivity analysis

To evaluate how the performances of *MSAEClust* change according to the chosen number of semi-supervised autoencoders, we vary this parameter between 10 to 50 with a step of 10. In this case, for each configuration (dataset, ensemble size) we generate ten sets of embeddings and, for each of them, we run *k-means* 30 times. The values reported in Figure 4 constitute the average of such runs.

We observe that the ensemble size has little or no influence on the performances of *MSAEClust*. The only exception is related to *Spambase*. However, variations of the ensemble size only result in slight fluctuations of the NMI (from 0.32 with 10 as the ensemble size, to 0.36 for an ensemble size equal to 30 or 40). This analysis allows us to conclude that the ensemble size parameter does not influence the performance of *MSAEClust* significantly, at least for the benchmark datasets considered in our study. We also observe that setting the size of the ensemble to a value between 30 and 40 help achieving good performances for all the datasets.

F. Qualitative investigation of the embedded space

The aim of this analysis is to provide a qualitative evaluation of the embedding learnt by our approach. Figure 5 shows two-dimensional projections of the embeddings learnt by *MSAEClust* and *ITML* on datasets *Splice* and *Landsat* when the number of labeled examples per class is equal to 10. To obtain the two dimensional representations, we apply principal component analysis (PCA) on the embeddings to extract the first ten principal components and, successively, we apply the t-distributed stochastic neighbor embedding (*TSNE*) approach [24] to obtain the final bidimensional space used for the visualization. PCA is applied because the representations learnt by both methods are high-dimensional and *TSNE* fails to manage such kind of data directly.

Figure 5(a) and 5(c) depict the visual results obtained by processing *Splice* data as described beforehand. This dataset involves instances belonging to three different classes. While the two dimensional projection of *ITML* (Figure 5(a)) seems not adequate to distinguish between the three different classes,

MSAEClust (Figure 5(a)) supplies an embedding that naturally allows to distinguish the original classes by simply looking at the resulting visual patterns. In Figure 5(b) and 5(d) we show the embeddings produced by both approaches on *Landsat* data. As in the previous case, we observe that the representation learnt by *MSAEClust* induces a better discrimination compared to the one supplied by *ITML*. While, in this case, the representation learnt by *ITML* (Figure 5(b)) allows to visually distinguish some cluster structures, we still observe very few coherent visual clusters; the majority of the visual groups are heterogeneous and involve examples of different classes. Conversely, the embeddings produced by *MSAEClust* (Figure 5(d)), unless some small confusion regarding the black triangle class, generally produce visual groups that seem more coherent and exhibit more clear visual patterns. The visual results we obtain are consistent with the performances in terms of NMI reported in Section V-D: the higher the visual separability induced by a method, the higher the values of Normalized Mutual Information achieved by the semi-supervised clustering method.

VI. Conclusions

In this work we have introduced *MSAEClust*, a semi-supervised clustering algorithm that directly exploits labeled examples to produce the final data partitioning. Our approach leverages an ensemble of semi-supervised autoencoder to produce a new representation space that is successively used to perform clustering. The ensemble construction leverages a multiresolution strategy that enables the injection of diversity in the set of network models generated. Extensive experimentations in comparison with state-of-the-art semi-supervised clustering algorithms have shown the benefit of the framework proposed in this paper. As future work we will study how to couple the embedding learnt by our framework with constrained clustering algorithms in order to exploit the background knowledge at both levels of the semi-supervised clustering process.

VII. Acknowledgments

The authors acknowledge the support of the National Research Agency within the framework of the program "Investissements d'Avenir" for the GEOSUD project (ANR-10-EQPX-20).

References

[1] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *Journal of Machine Learning Research*, vol. 6, pp. 1705–1749, 2005.

[2] S. Basu, A. Banerjee, and R. J. Mooney, "Semi-supervised clustering by seeding," in *Proceedings ICML 2002*, Sydney, Australia, 2002, pp. 27–34.

[3] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings ACM SIGKDD 2004*, Seattle, USA, 2004, pp. 59–68.

[4] S. Basu, I. Davidson, and K. W. (Editors), *Constrained Clustering: Advances in Algorithms, Theory and Applications*. Chapman & Hall/CRC Press, Data Mining and Knowledge Discovery Series, 2008.

[5] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering." in *Proceedings ICML 2004*, Banff, Canada, 2004, pp. 81–88.

[6] J. Chen, S. Sathe, C. C. Aggarwal, and D. S. Turaga, "Outlier detection with autoencoder ensembles," in *SDM*, 2017, pp. 90–98.

[7] M. Cucuringu, I. Koutis, S. Chawla, G. L. Miller, and R. Peng, "Simple and scalable constrained clustering: a generalized spectral method," in *Proceedings of AISTATS 2016, Cadiz, Spain*, ser. JMLR Workshop and Conference Proceedings, vol. 51. JMLR.org, 2016, pp. 445–454.

[8] I. Davidson and S. S. Ravi, "Identifying and generating easy sets of constraints for clustering," in *AAAI*, 2006, pp. 336–341.

[9] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *ICML*, 2007, pp. 209–216.

[10] M. Ganji, J. Bailey, and P. J. Stuckey, "Lagrangian constrained clustering," in *Proceedings of SIAM SDM 2016, Miami, Florida, USA*. SIAM, 2016, pp. 288–296.

[11] A. Gogna and A. Majumdar, "Semi supervised autoencoder," in *ICONIP*, 2016, pp. 82–89.

[12] F. Gullo, C. Domeniconi, and A. Tagarelli, "Metacluster-based projective clustering ensembles," *Machine Learning*, vol. 98, no. 1-2, pp. 181–216, 2015.

[13] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[14] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[15] W. Kalintha, S. Ono, M. Numao, and K. Fukui, "Kernelized evolutionary distance metric learning for semi-supervised clustering," in *Proceedings of AAAI 2017*. AAAI Press, 2017, pp. 4945–4946.

[16] D. Klein, S. D. Kamvar, and C. D. Manning, "From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering," in *Proceedings ICML 2002*, Sydney, Australia, 2002, pp. 307–314.

[17] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of ICML 2010*. Omnipress, 2010, pp. 807–814.

[18] B. M. Nogueira, Y. K. B. Tomas, and R. M. Marcacini, "Integrating distance metric learning and cluster-level constraints in semi-supervised clustering," in *Proceedings of IJCNN 2017*. IEEE, 2017, pp. 4118–4125.

[19] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *NIPS*, 2015, pp. 3546–3554.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[21] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] A. Strehl and J. Ghosh, "Cluster ensembles — A knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[23] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," COURSERA: Neural Networks for Machine Learning, 2012.

[24] L. van der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.

[25] K. Wagstaff and C. Cardie, "Clustering with instance-level constraints." in *Proceedings ICML 2000*, Standford, USA, 2000, pp. 1103–1110.

[26] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *Proceedings ICML 2001*, Williamstown, USA, 2001, pp. 577–584.

[27] X. Wang, B. Qian, and I. Davidson, "Labels vs. pairwise constraints: A unified view of label propagation and constrained spectral clustering," in *12th IEEE International Conference on Data Mining, ICDM 2012, Brussels, Belgium, December 10-13, 2012*, 2012, pp. 1146–1151.

[28] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NIPS*. MIT Press, 2003, pp. 321–328.

[29] X. Zhu, C. C. Loy, and S. Gong, "Constrained clustering with imperfect oracles," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 27, no. 6, pp. 1345–1357, 2016.

[30] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*. AAAI Press, 2003, pp. 912–919.