Unsupervised Learning by Minimal Entropy Encoding

Stefano Melacci, Marco Gori, *Fellow, IEEE*

***Abstract*—Following basic principles of Information-Theoretic Learning, in this paper we propose a novel approach to data clustering, referred to as Minimal Entropy Encoding (MEE), which is based on a set of functions (features) projecting each input onto a minimum entropy configuration (*code*). Inspired by traditional parsimony principles, we seek solutions in Reproducing Kernel Hilbert Spaces (RKHS) and then we prove that the encoding functions are expressed in terms of kernel expansion. In order to avoid trivial solutions, we impose that the developed features must be as different as possible by means of a soft-constraint on the empirical estimation of the entropy associated to the encoding functions. This leads to an unconstrained optimization problem that can be efficiently solved by conjugate gradient. We also investigate an optimization strategy based on concave-convex algorithms. The relationships with Maximum Margin Clustering are studied, showing that MEE overcomes some of its critical issues, such as the lack of a multi-class extension and the need to face problems with a large number of constraints. A massive evaluation on several benchmarks of the proposed approach shows improvements over state-of-the-art techniques, both in terms of accuracy and computational complexity.**

***Index Terms*—Clustering, entropy encoders, information-theoretic learning, kernel methods, unsupervised learning.**

I. Introduction

CLUSTERING is a central topic in machine learning that has a crucial impact in diverse domains, including bioinformatics, medical science, computer vision and information retrieval [1]. Clustering algorithms aim at discovering the underlying structure of the data, by grouping examples into a number of classes, or clusters, such that those belonging to the same cluster are "similar" to each other and "different" from the ones of the other clusters. Popular approaches to clustering include, amongst others, K-Means [4], Mean Shift [5], Mixture Models [1], Spectral Clustering [6]. Many algorithms for the visualization and analysis of high dimensional data follow clustering principles [2], [3].

Kernel machines has been successfully used in clustering algorithms. It is the case of Support Vector Clustering [7] or of kernel-based versions of previously proposed clustering algorithms (see [8] and reference therein). More recently, Maximum Margin Clustering (MMC) has received an increasing interest in the scientific community [9]. Following the success of maximum margin regularized classifiers, MMC extends their principles to unsupervised learning, thus leading to state-of-the art performances in many applications. MMC learns the optimal hyperplane that separates dense regions of points, while maximizing the margin.

S. Melacci and M. Gori are with the Department of Information Engineering, University of Siena, 53100 - Siena, Italy, e-mail: {mela,marco}@dii.unisi.it.

The learning problem behind MMC is intrinsically complicated, so as several relaxations of the original formulation have been recently proposed to make it more affordable in terms of computational resources. The idea of Xu et al. [9] is formalized by an instance of convex semi-definite programming. This is the first popular approach to MMC, but solving the problem is still intractable for many real-word applications, since the number of variables is $O(n^2)$, where n is the number of observations. Valizadegan and Jin [10] propose Generalized Maximum Margin Clustering (GMMC) to reduce the number to $O(n)$, but the algorithm is still limited to several hundreds of points. In order to overcome this issue, Zhang et al. [11] describe an alternative formulation, called IterSVR. The MMC problem is tackled by an iterative approach based on a set of support vector regression problems. The IterSVR objective function is not convex, but its optimization takes place on larger data collections, and it has been shown to be significantly faster than the previous related algorithms. Similarly, a cutting-plane based approach (CPMMC) [12] has been proven to be very efficient for large scale MMC. The optimization is still not-convex, with a large number of constraints, and it is solved using a cutting-plane strategy and a constrained concave-convex procedure (CCCP) [13]. The authors study the case of linear classifiers, and propose to extend the algorithm to the nonlinear setting by explicitly embedding the data in the feature space, using Kernel PCA [14] or a Cholesky factorization of the kernel matrix. In order to preserve convexity, a tighter and convex relaxation (LGMMC) of the original MMC is proposed in [15], with interesting results.

Inspired by the Information-Theoretic Learning (ITL) framework [16], different algorithms for clustering [17]–[21], feature extraction [22], and dimensionality reduction [23]–[25] have been recently proposed. In the ITL framework the *information* carried in the data is transferred onto the model parameters by using cost functions that involve entropy-based measures. The specific choice of the measure and the definition of the learning framework strictly depend on the considered task. For instance, Grandvalet and Bengio [26] propose a semi-supervised learning scheme that is built on the idea of *conditional entropy* minimization. Conditional entropy is shown to be a measure of class overlap [26] and an alternative way to implement the popular "cluster assumption" [27], [28], which drives the decision boundaries to low-density regions. Moving to the unsupervised setting, conditional entropy is directly related to the quality of the data partitioning [29], and this property has been recently exploited in clustering algorithms [20]. When minimized under certain smoothness assumptions, it has been recently shown to produce clustering

outcomes that are close to MMC [19].

In this paper, we introduce Minimal Entropy Encoding (MEE), which is based on a set of functions (features) projecting each input onto a minimum entropy configuration (*code*), of length d, where d is the number of clusters.[1] There are three main contributions in this paper. First, following traditional parsimony principles, we seek solutions in Reproducing Kernel Hilbert Spaces (RKHS) and then we prove that the encoding functions are expressed in terms of kernel expansion. In order to avoid trivial solutions[2], we optimize an index which contains a balancing term aimed at forcing the development of as many features as possible. To some extent, this is related to the introduction of additional constraints in MMC algorithms [9], [10], [12], [15] and to the iterative label-switching mechanisms [19]. Second, we present an efficient way to solve the unconstrained MEE problem that is based on conjugate gradient, and we investigate an alternative optimization scheme based on concave-convex procedures. Using the former, MEE has time complexity $O(n^2dT)$, where T is the number of iterations, that we empirically found to be $T << n$. Due to the non-convex formulation of the problem, we also present an appropriate initialization criterion and a strategy that can be used to filter out sub-optimal solutions. In addition, MEE can naturally handle multi-class tasks, whereas the described MMC implementations are designed for 2-class problems, with the exceptions of Xu et al. [30] and the multiclass version of CPMMC, referred to as CPM3C [31]. The proposed method, however, has a significantly smaller space complexity, both in the linear and nonlinear cases. Third, a massive evaluation on several benchmarks shows that MEE overcomes state-of-the art techniques both in terms of accuracy and complexity.

The paper is organized as follows. Section II presents the MEE approach to clustering and Section III describes efficient ways to train the encoders, followed by a complexity analysis. Section IV provides the assessment of the quality of MEE both in 2-class and multi-class data using a number of different classic benchmarks.

II. Minimal Entropy Encoding

We consider a set of n unlabeled points, $\mathcal{X} = \{\boldsymbol{x}_i \in I\!R^q, i = 1, \ldots, n\}$ that are supposed to be reasonably represented by a codebook of d symbols with corresponding emission probabilities. Following the established framework of regularization used in kernel machines [14], we model the d probabilities using the encoding functions $\boldsymbol{f} = [f_1, \ldots, f_{d-1}]^T$, where $f_i \in \mathcal{H}$, $\mathcal{H}$ being the Reproducing Kernel Hilbert Space (RKHS) [14] induced by the kernel function $k(\cdot, \cdot)$.[3] In order to enforce a probabilistic normalization we use a (multinomial-logit) *softmax* function, so that for $j = 1, \ldots, d-1$ the emission probability of the j-th symbol is

$$p_j(\boldsymbol{x}_i, \boldsymbol{f}) = \frac{e^{f_j(\boldsymbol{x}_i)}}{1 + \sum_{z=1}^{d-1} e^{f_z(\boldsymbol{x}_i)}}, \quad (1)$$

and

$$p_d(\boldsymbol{x}_i, \boldsymbol{f}) = \frac{1}{1 + \sum_{z=1}^{d-1} e^{f_z(\boldsymbol{x}_i)}}. \quad (2)$$

The output *code* $\boldsymbol{o}_i$ associated to an input point $\boldsymbol{x}_i$ is

$$\boldsymbol{o}_i = [p_1(\boldsymbol{x}_i, \boldsymbol{f}), \ldots, p_d(\boldsymbol{x}_i, \boldsymbol{f})].$$

Following Information Theoretic principles [16], the (Shannon) *entropy* of the i-th code is

$$H(Y(\boldsymbol{f})|X = \boldsymbol{x}_i) = -\sum_{j=1}^{d} p_j(\boldsymbol{x}_i, \boldsymbol{f}) \log p_j(\boldsymbol{x}_i, \boldsymbol{f}). \quad (3)$$

The entropy is maximum when all the p_j, $j = 1, \ldots, d$, are equal to $1/d$, whereas it is minimized for "one-hot" configurations in which only one of them is 1 and the remaining ones are 0. In order to emphasize distinctive properties of $\boldsymbol{x}_i$, the encoder must produce vectors with corresponding small entropy configurations. If we know that $\mathcal{X}$ is partitioned into d clusters, we can interpret $p_j(\boldsymbol{x}_i, \boldsymbol{f})$ as the probability of the j-th cluster given $\boldsymbol{x}_i$, and we can get the cluster index of $\boldsymbol{x}_i$ by $y_i(\boldsymbol{f}) = \arg\max_j p_j(\boldsymbol{x}_i, \boldsymbol{f})$, $y_i \in \mathcal{Y} = \{1, 2, \ldots, d\}$ which, of course, depends on $\boldsymbol{f}$.

Now, let X, $Y(\boldsymbol{f})$ be the random variables associated to the data and the clustering outcomes, respectively, so that, for the chosen $\boldsymbol{f}$, we denote by $H(Y(\boldsymbol{f})|X)$ the *conditional entropy* of $Y(\boldsymbol{f})$ given X, which is defined as

$$H(Y(\boldsymbol{f})|X) = -\int \sum_{j=1}^{d} p_j(\boldsymbol{x}, \boldsymbol{f}) \log \left(p_j(\boldsymbol{x}, \boldsymbol{f})\right) p(\boldsymbol{x}) d\boldsymbol{x}.$$

Although $p(\boldsymbol{x})$ is not available, we can follow [26] and apply the plug-in principle to get the empirical estimate of $H(Y(\boldsymbol{f})|X)$. The plug-in principle suggests that the estimator of $H(Y(\boldsymbol{f})|X)$ can be made using a finite quantization of the probability distribution, that in our case leads to the average value of $H(Y(\boldsymbol{f})|X = \boldsymbol{x})$ (3) over $\mathcal{X}$,

$$\begin{aligned} H(Y(\boldsymbol{f})|X) &\stackrel{emp}{=} \frac{1}{n} \sum_{i=1}^{n} H(Y(\boldsymbol{f})|X = \boldsymbol{x}_i) \\ &= -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{d} p_j(\boldsymbol{x}_i, \boldsymbol{f}) \log p_j(\boldsymbol{x}_i, \boldsymbol{f}). \quad (4) \end{aligned}$$

We notice on passing that $p_j(\boldsymbol{x}_i, \boldsymbol{f}) = p_{Y|X}(j|\boldsymbol{x}_i)$, and that we will interchangeably refer to this term using these two definitions, depending on the context. Previous studies [26], [29] have shown that the conditional entropy is an index of the class overlap, and desirable data partitions are related to small $H(Y(\boldsymbol{f})|X)$. Enforcing configurations with small conditional entropy encourages the grouping of dense point clouds in the same cluster, whereas the separation boundaries pass through low density regions. This idea is closely related to the principles of Maximum Margin Clustering (MMC), in such a manner that under some mild approximations MMC can be

[1] In this work we assume that the number of clusters d is known. Some ideas on the relationships between the number of developed clusters and the properties of the proposed algorithm will be sketched in the last section of the paper.

[2] Notice that, regardless of the input, any code of d bits in which only one is "hot" yields the minimal entropy.

[3] We need only $d-1$ functions to model d probabilities, since the d-th one is trivially $p_d = 1 - \sum_{j=1}^{d-1} p_j$.

reformulated using minimum conditional entropy arguments [19].

Unfortunately, the degenerate solution in which all the data points are assigned to the same cluster minimizes the conditional entropy and, therefore, we need to introduce a class balancing constraint. In the case of MMC algorithms, different constrained optimization problems are formulated to balance the class label distribution [9], [10], [15], [31], or a label-switching mechanisms is included in the iterative clustering process [19]. However, in the case of multiple classes ($d > 2$) the number of balancing constraints is d^2, while the label-switching [32] is not straightforwardly generalizable to the multi-class case, since it is specifically designed for 2-class data. As a consequence, the majority of MMC algorithms are limited to the 2-class case. We propose to introduce a regularizer that is based on descriptors from Information Theory. Basically, we favor those configurations in which the d output features are emitted with similar probability $p_Y(j)$. Again, we can empirically estimate the marginal $p_Y(j)$ using the available data and the plug-in principle, such that for $j = 1, \ldots, d$ we have

$$p_Y(j) = \int p_j(\boldsymbol{x}, \boldsymbol{f}) p(\boldsymbol{x}) d\boldsymbol{x} \stackrel{emp}{=} \frac{1}{n} \sum_{i=1}^{n} p_j(\boldsymbol{x}_i, \boldsymbol{f}).$$

Let $H(Y(\boldsymbol{f}))$ be the entropy associated to $p_Y(j)$. It turns out to be

$$H(Y(\boldsymbol{f})) = -\frac{1}{n} \sum_{j=1}^{d} \sum_{i=1}^{n} p_j(\boldsymbol{x}_i, \boldsymbol{f}) \log \left(\frac{1}{n} \sum_{i=1}^{n} p_j(\boldsymbol{x}_i, \boldsymbol{f}) \right). \tag{5}$$

The maximum of $H(Y(\boldsymbol{f}))$ corresponds with configurations for which $p_Y(j) = 1/d$, $j = 1, \ldots, d$, i.e. the average emission probabilities of the d symbols are balanced over X. Following the well established framework of regularization, we seek solutions in the RKHS of $k(\cdot, \cdot)$ such that $\| \boldsymbol{f} \|_K = \sum_{z=1}^{d-1} \| f_z \|_K^2$ is small. Let us introduce

$$D_\mu(X, Y(\boldsymbol{f})) := (1 - \mu) H(Y(\boldsymbol{f})|X) - \mu H(Y(\boldsymbol{f})), \tag{6}$$

$0 < \mu < 1$, to aggregate the two entropy terms of (4) and (5), respectively, where the sign of the second one has been switched. Now, the problem of looking for configurations with small conditional entropy (4) and with a balanced number of features that come from smooth functions is converted into the one of finding $\hat{\boldsymbol{f}} = [f_1, \ldots, f_{d-1}]^T$ such that $\hat{\boldsymbol{f}} = \arg\min C(\boldsymbol{f})$, where

$$C(\boldsymbol{f}) = D_\mu(X, Y(\boldsymbol{f})) + \lambda \| \boldsymbol{f} \|_K . \tag{7}$$

The parameter μ in (6) measures the degree of enforcement of the feature soft-balancing constraint. Larger values of μ favor the development of all the feature functions as well as averagely balanced emission probabilities. Unbalanced solutions are tolerated when μ is small. The different selection of the signs in the two entropies allows us to maximize $H(Y(\boldsymbol{f}))$, while minimizing $H(Y(\boldsymbol{f})|X)$. Moreover, $\lambda > 0$ is the classic regularization parameter that adjusts the degree of smoothness of the solution. Since $-\mu H(Y(\boldsymbol{f}))$ ranges in $[-\mu \log d, 0]$ we add the offset $\mu \log d$ to ensure that (7) is nonnegative. Interestingly, $D_\mu(X, Y(\boldsymbol{f}))$ has an intriguing meaning that is disclosed in the following proposition.

Proposition 1: We have

$$D_\mu(X, Y(\boldsymbol{f})) = -D_{KL}(p_{XY} || (p_Y p_{Y|X})^\mu \cdot p_X), \tag{8}$$

where $D_{KL}(p_{XY} || (p_Y p_{Y|X})^\mu \cdot p_X)$ is the Kullback-Leibler distance between p_{XY} and $(p_Y p_{Y|X})^\mu \cdot p_X$. In particular, if $\mu = 1/2$ then

$$D_\mu(X, Y(\boldsymbol{f})) = -\frac{1}{2} I(X, Y(\boldsymbol{f}))$$

where $I(X, Y(\boldsymbol{f}))$ is the mutual information between X and Y for the given $\boldsymbol{f}$.

Proof: The proof comes out as follows[4]

$$\begin{aligned}
D_\mu &= (1 - \mu) H(Y(\boldsymbol{f})|X) - \mu H(Y(\boldsymbol{f})) \\
&= -\Bigg[(1 - \mu) \frac{1}{n} \sum_{j=1}^{d} \sum_{i=1}^{n} p_j(\boldsymbol{x}_i, \boldsymbol{f}) \log p_j(\boldsymbol{x}_i, \boldsymbol{f}) \\
&\quad - \mu \frac{1}{n} \sum_{j=1}^{d} \sum_{i=1}^{n} p_j(\boldsymbol{x}_i, \boldsymbol{f}) \log \left(\frac{1}{n} \sum_{i=1}^{n} p_j(\boldsymbol{x}_i, \boldsymbol{f}) \right) \Bigg] \\
&= -\frac{1}{n} \sum_{j=1}^{d} \sum_{i=1}^{n} p_{Y|X}(j|\boldsymbol{x}_i) \log \frac{p_{Y|X}(j|\boldsymbol{x}_i)^{1-\mu}}{p_Y(j)^\mu} \\
&= -\sum_{j=1}^{d} \sum_{i=1}^{n} p_{XY}(\boldsymbol{x}_i, j) \\
&\quad \cdot \log \left(\frac{p_{Y|X}(j|\boldsymbol{x}_i)^{1-\mu}}{p_Y(j)^\mu} \cdot \frac{p_{Y|X}(j|\boldsymbol{x}_i)^\mu}{p_{Y|X}(j|\boldsymbol{x}_i)^\mu} \right) \\
&= -\sum_{j=1}^{d} \sum_{i=1}^{n} p_{XY}(\boldsymbol{x}_i, j) \log \frac{p_{XY}(\boldsymbol{x}_i, j) \cdot p_X(\boldsymbol{x}_i)^{-1}}{p_Y(j)^\mu p_{Y|X}(j|\boldsymbol{x}_i)^\mu},
\end{aligned}$$

in which we dropped the dependence on $\boldsymbol{f}$ from the point mass probabilities to simplify the notation. From the last expression we directly derive equation (8). Finally, for $\mu = 1/2$ we have

$$\begin{aligned}
D_{KL}(p_{XY} || (p_Y p_{Y|X})^{\frac{1}{2}} \cdot p_X) = \\
&= D_{KL}(p_{XY} || (p_Y p_{Y|X} p_X)^{\frac{1}{2}} \cdot p_X^{\frac{1}{2}}) \\
&= D_{KL}(p_{XY} || (p_Y p_{XY} p_X)^{\frac{1}{2}}) \\
&= \frac{1}{2} D_{KL}(p_{XY} || p_Y p_X) \\
&= \frac{1}{2} I(X, Y).
\end{aligned}$$

■

The mutual information measures the statistical dependence of two random variables. MEEs maximizes the dependency between X and Y, without making any parametric assumption on the probability density function of X, that is never explicitly estimated. However, from a practical point of view the value of μ must be adjusted to get the best performance, as it will be shown in Section IV.

The optimal solution of the problem is independent of the log base, that we selected to be the natural one (ln). For the

[4]Apart from the dependency on the selected functions $f_j \in \mathcal{H}$, the above property is classic in information theory (see e.g. [33], Ch. 2).

sake of simplicity, we define $p_{ij} := p_j(\boldsymbol{x}_i, \boldsymbol{f})$, and $w_i := 1+\sum_{z=1}^{d-1} e^{f_z(\boldsymbol{x}_i)}$. In turn, the conditional entropy $H(Y(\boldsymbol{f})|X)$ (4) can be rewritten as

$$\begin{aligned} H(Y(\boldsymbol{f})|X) &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d} p_{ij}\ln\left(p_{ij}\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d-1} p_{ij}\ln\left(\frac{e^{f_j(\boldsymbol{x}_i)}}{w_i}\right) - \frac{1}{n}\sum_{i=1}^{n} p_{id}\ln\left(\frac{1}{w_i}\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d-1} p_{ij}\left(f_j(\boldsymbol{x}_i) - \ln\left(w_i\right)\right) + \frac{1}{n}\sum_{i=1}^{n} p_{id}\ln\left(w_i\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d-1} p_{ij} f_j(\boldsymbol{x}_i) + \frac{1}{n}\sum_{i=1}^{n}\ln(w_i)\cdot\left(\sum_{j=1}^{d-1} p_{ij} + p_{id}\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d-1} p_{ij} f_j(\boldsymbol{x}_i) + \frac{1}{n}\sum_{i=1}^{n}\ln\left(w_i\right). \end{aligned} \tag{9}$$

Interestingly, the hypothesis of looking for solutions belonging to the RKHS of $k(\cdot,\cdot)$ has a strong consequence in the optimization of (7).

Theorem 1: Let $f_j \in \mathcal{H}$, where $\mathcal{H}$ is the RKHS of kernel $k(\cdot,\cdot)$. Then there exists $\alpha_{ij} \in \mathbb{R}$ such that $\hat{f}_j = \arg\min_{f_j\in\mathcal{H}} C(\boldsymbol{f})$ is given by

$$\hat{f}_j(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_{ij} k(\boldsymbol{x}_i, \boldsymbol{x}) \tag{10}$$

Proof: See Appendix A. ∎

Now, let $K \in \mathbb{R}^{n,n}$ be the Gram matrix associated to the points in $\mathcal{X}$, $\boldsymbol{k}_j \in \mathbb{R}^n$ is j-column of K, and k_{ij} is the element in position (i,j). Analogously, $A \in \mathbb{R}^{n,d-1}$ is the matrix of the coefficients α_{ij}, and $\boldsymbol{\alpha}_j \in \mathbb{R}^n$ is its j-th column. The product $KA \in \mathbb{R}^{n,d-1}$ is the matrix composed of the values of f_j, $j = 1,\ldots,d-1$ on the points in X. $P \in \mathbb{R}^{n,d}$ collects the probabilities p_{ij}. $\mathbf{1}_n$ is the column vector composed of n elements equal to 1. The symbol $\circ$ is the element-wise product between two arrays or matrices, and $\ln(\cdot)$ and $e^{(\cdot)}$ are assumed to operate element-wise on vectorial data. We define $\tilde{d} = d-1$, and we use the notation $\tilde{P}$ to indicate the matrix composed with first $\tilde{d}$ columns of P. Finally, (7) can be written in terms of A using the matrix notation as

$$\begin{aligned} C(A) = \; & \frac{(1-\mu)}{n}\mathbf{1}_n^T\left((-\tilde{P}\circ(KA))\mathbf{1}_{\tilde{d}} + \ln(\mathbf{1}_n + e^{KA}\mathbf{1}_{\tilde{d}})\right) \\ & -\frac{\mu}{n}\left(((-\mathbf{1}_n^T P)\circ(\ln\frac{1}{n}\mathbf{1}_n^T P))\mathbf{1}_d - n\ln d\right) \\ & +\lambda\mathbf{1}_n^T((KA)\circ A)\mathbf{1}_{\tilde{d}}. \end{aligned} \tag{11}$$

III. Training the Encoder

Training an MME consists of minimizing $C(A)$ (11) with respect to the matrix of coefficients A. The objective function is continuous, differentiable but, in general, because of the relation between the probabilities and the coefficients of matrix A, it is not convex. We optimized the problem using a nonlinear Conjugate Gradient (CG) with line search performed by the secant method [34]. We give a sketch of the optimization scheme in Algorithm 1, where $\nabla C(A) \in \mathbb{R}^{n,\tilde{d}}$ is the gradient of (11) with respect to A. The line search procedure is reported in Algorithm 2.

Basically, the algorithm performs a first step that is equivalent to a steepest descent (D_0 is simply the opposite of the gradient), and then the following search directions are computed by Gram-Schmidt conjugation of the residuals [34], [35]. We used the Polak-Ribiere formula with automatic restart to update the search direction ($\beta = \max\{(\tau_{t+1} - \tau_{mix})/\tau_t, 0\}$ in Algorithm 1), that is frequently used in non-linear CG [34], [35]. The max operator allows the algorithm to automatically reset the descent direction in case of round-off errors that may lead to negative β. The line search has been implemented by the secant method [34] since it avoids the computation of second derivatives, and it is generally used for its speed and ease of implementation.

In Appendix B we will describe how to efficiently simplify the gradient expression $\nabla C(A)$ to avoid a costly matrix-by-matrix product, so that Algorithm 1 will be applied using a preconditioned form $\nabla' C(A)$ instead of $\nabla C(A)$. In Section III-A we will analyze the complexity of this approach.

Algorithm 1 Optimization of MEEs by nonlinear Conjugate Gradient (CG) with Polak-Ribier update.

Given: a starting point A_0, a maximum number of iterations t_{max} and a minimum gradient norm ϵ.
Output: the coefficients $A^\star$ at a stationary point of $C(\cdot)$.
Let: $t = 0$, $D_0 = -\nabla C(A_0)$, and $\tau_0 = \|\nabla C(A_0)\|^2 = \mathbf{1}_n^T(\nabla C(A_0)\circ\nabla C(A_0))\mathbf{1}_{\tilde{d}}$.
repeat
  Compute A_{t+1} by line search (Algorithm 2)
  $\tau_{mix} = \mathbf{1}_n^T(\nabla C(A_{t+1})\circ\nabla C(A_t))\mathbf{1}_{\tilde{d}}$
  $\tau_{t+1} = \|\nabla C(A_{t+1})\|^2 = \mathbf{1}_n^T(\nabla C(A_{t+1})\circ\nabla C(A_{t+1}))\mathbf{1}_{\tilde{d}}$
  $\beta = \max\{(\tau_{t+1} - \tau_{mix})/\tau_t, 0\}$
  $D_{t+1} = \nabla C(A_{t+1}) + \beta\cdot D_t$
  $t = t + 1$
until Goal condition ($t = t_{max}$ *or* $\tau_t < \epsilon^2$)
$A^\star = A_t$

Even though in this paper we focus our studies on the CG optimization strategy, we introduce another optimization scheme that allows MEEs to be trained using convex optimization tools. Such tools are largely available and they offer a variety of efficient alternatives [36]. In particular, the structure of the MEE objective function allows us to apply a concave-convex procedure (CCCP) [13], [37] by means of an alternative formulation of the emission probabilities. CCCP allows the MEE problem (that is not-convex) to be minimized by iteratively solving a set of constrained convex problems, subject to linear equality and inequality constraints [37].

Instead of following (1), we can model the $\tilde{d}$ emission probabilities with $p_j(\boldsymbol{x}_i, \boldsymbol{f}) = f_j(\boldsymbol{x}) = \sum_{i=1}^{n}\alpha_{ij}k(\boldsymbol{x}, \boldsymbol{x}_i)$, $j = 1,\ldots,\tilde{d}$, and then enforce the probabilistic relationships using a set of linear constraints, $\sum_{j=1}^{\tilde{d}} f_j(\boldsymbol{x}_i) \le 1$, $f_j(\boldsymbol{x}_i) \ge 0$, with $j = 1,\ldots,\tilde{d}$ and $i = 1,\ldots,n$. The first inequality is due to the fact that the d-th probability term is not modeled

Algorithm 2 Line search performed by the secant method.

Given: A_t, a search direction D_t, an initial step size γ, and a maximum number of line search iterations r_{max} with tolerance ρ.
Output: the optimal A_{t+1} along the line $A_t + \theta \cdot D_t$.
Let: $r = 0$, $\theta^{(0)} = -\gamma$, $A_t^{(0)} = A_t$,
$\phi = \|D_t\|^2 = \mathbf{1}_n^T (D_t \circ D_t) \mathbf{1}_{\tilde{d}}$,
$\Gamma = \nabla C(A_t + \gamma D_t)$,
$\eta = \mathbf{1}_n^T (\Gamma \circ D_t) \mathbf{1}_{\tilde{d}}$.
repeat
$\eta_{new} = \mathbf{1}_n^T (\nabla C(A_t^{(r)}) \circ D_t) \mathbf{1}_{\tilde{d}}$
$\theta^{(r+1)} = \theta^{(r)} \cdot (\eta_{new}/(\eta - \eta_{new}))$
$A_t^{(r+1)} = A_t^{(r)} + \theta^{(r+1)} \cdot D_t$
$\eta = \eta_{new}$
$r = r + 1$
until Goal condition ($r = r_{max}$ *or* $(\theta^{(r)})^2 \phi < \rho^2$)
$A_{t+1} = A_t^{(r)}$

by any f_j.[3] We indicate with $C_{cccp}(A)$ the objective function that is based on such emission probabilities. $C_{cccp}(A)$ and $C(A)$ share exactly the same principles. The only difference is that, in the case of $C_{cccp}(A)$, $p_j(\boldsymbol{x}_i, \boldsymbol{f})$, $j = 1, \ldots, d$ are constrained to fulfill a probabilistic relationship only on a set of n points, whereas they are known to be probabilities for each point in $\mathbb{R}^q$ in the case of $C(A)$.

It is easy to see that $C(\boldsymbol{f})$ (7) is composed by the sum of a concave and a convex term in $p_j(\boldsymbol{x}, \boldsymbol{f})$ (i.e. $H(Y(\boldsymbol{f})|X)$ and $H(Y(\boldsymbol{f}))$, respectively) and a convex term in f_j (i.e. $\|\boldsymbol{f}\|_K$). In the case of $C_{cccp}(A)$, this consideration also holds with respect to the coefficients in A.

Following the notation commonly used in CCCP [37], we indicate with $u(A)$ and $-v(A)$ the convex and concave portions of $C_{cccp}(A)$, so that $C_{cccp}(A) = u(A) - v(A)$. Given the α_{ij} coefficients at a time t, collected in A_t, then,

$$C_{convex}(A|A_t) \quad = \quad u(A) - \mathbf{1}_n^T (A \circ \nabla v(A_t)) \mathbf{1}_{\tilde{d}}, \quad (12)$$

is a convex function obtained by linearizing the concave part of $C_{cccp}(A)$ around A_t. Each step of CCCP requires the minimization of (12) with respect to A (subject to the probabilistic constraints). Then, after setting $A_{t+1} = \arg\min_A C_{convex}(A|A_t)$, another CCCP step is performed, and this procedure is guaranteed to converge to a stationary point of $C_{cccp}(A)$ [37].

Focusing on the CG optimization procedure, now we discuss two critical aspects of the optimization scheme. The first one is how to select the starting point of the optimization A_0, whereas the second one is how to compare a set of solutions A^* (computed from different starting points) selecting the most promising one.

Different A_0 can lead to different suboptimal results A^*. A a good choice A_0 can reduce the training times and improve the quality of the solution. For instance, a random initialization is the most immediate choice, but it may not lead to satisfying clustering outcomes. We also notice that each permutation of the columns of A_0 will lead to the same solution (the permutation will be applied to the output codes), and that selecting $A_0 = 0 \cdot \mathbf{1}_n \mathbf{1}_{\tilde{d}}^T$ (the null matrix) leads to $f_1, \ldots, f_{\tilde{d}}$ that are equivalent and constant, where each input point is projected into the same output code $[1/d, \ldots, 1/d]$. This is a degenerate solution of the problem and it must be avoided. In order to get a more appropriate initial guess of the data partitioning, we suggest to run a few K-Means iterations. This choice is motivated by the simplicity and speed of K-Means clustering, but the essence of what follows also holds for other clustering algorithms. We denote with $L \in \{-1, 1\}^{n,\tilde{d}}$ the label matrix generated by K-Means, where the element in position (i, j) is 1 if $\boldsymbol{x}_i$ was predicted to belong to cluster j (when it belongs to cluster d, then the i-th line is composed only of -1's). We can solve a regression problem to fit the targets of L with the functions $f_1, \ldots, f_{\tilde{d}}$. Then, given the probabilities (1) and (2), it is easy to verify that $y_i(\boldsymbol{f}) = \arg\max_j p_j(\boldsymbol{x}_i, \boldsymbol{f})$ will match the K-Means prediction for all i. In detail, we can solve a regularized least-square problem [38], [39], that corresponds with minimizing the objective function given by

$$C_{init}(\boldsymbol{f}) = \sum_{j=1}^{d-1} \sum_{i=1}^{n} (f_j(\boldsymbol{x}_i) - l_{ij})^2 + \lambda \|\boldsymbol{f}\|_K, \quad (13)$$

where l_{ij} is the element of L in position (i, j). When passing to the finite dimensional case (10), the solution of (13) is $A_0 = (K + \lambda I)^{-1} L$, where I is the identity matrix [38]. We notice that an alternative approach to get the initial coefficients is to solve a regularized logistic regression problem using the (properly scaled) targets from the K-Means output. This approach would directly fit the emission probabilities to the K-Means targets. However, solving (13) is simpler, and it leads to an essentially equivalent result, if we consider that we are only roughly initializing the MEE algorithm.

We can collect a set of promising starting points A_0 with different criteria or with multiple runs of K-Means, and then select the best solution A^*. Such selection process can be tackled by considering that once an MEE is trained we have the use of the conditional entropy $H(Y(\boldsymbol{f})|X)$ associated to the given solution without performing any additional machinery, since $H(Y(\boldsymbol{f})|X)$ is one of the terms that compose (7). We already discussed how $H(Y(\boldsymbol{f})|X)$ is popularly used as an index of clustering quality [29]. Hence, given a set of Minimal Entropy Encoders trained from different starting points A_0, we can filter out the not promising solutions by selecting the one that corresponds to the smallest $H(Y(\boldsymbol{f})|X)$. We notice that this criterion is heuristic, since we have no guarantees that we are not selecting a degenerate solution (without performing other checks). The soft nature of the constraining scheme in $C(\boldsymbol{f})$ does not allow us to have formal guarantees about this point. On the other hand, $C(\boldsymbol{f})$ is strongly oriented in avoiding degenerate solutions and we successfully used this criterion in several benchmarks, as we will investigate in Section IV.

A. Complexity Analysis

In this work we present MEE in the case of a generic kernel function $k(\cdot, \cdot)$, that can be linear or nonlinear without any distinctions. We coherently follow the same approach in this complexity analysis. However, we notice that in the linear case

the complexity can be reduced, since there is no need to store the Gram matrix K. We leave to future work the study of the optimization and of the other details of linear MEE.

When an MEE is trained using CG, each iteration requires to store the matrix A, the preconditioned gradient $\nabla' C(A)$, and the current gradient descent direction. Each of those matrices belongs to $\mathbb{R}^{n,\tilde{d}}$ (for simplicity, we discard the bias term in this discussion). If the matrices $KA \in \mathbb{R}^{n,\tilde{d}}$ and $P \in \mathbb{R}^{n,d}$ are stored when evaluating the objective function (11), then the computation of the gradient $\nabla'(A)$ can be significantly accelerated. This result is remarkable, since the space complexity scales linearly with the number of classes d. Optionally, also the Gram matrix $K \in \mathbb{R}^{n,n}$ can precomputed. Hence, in the worst case, the space complexity is bounded $O(n^2)$ ($d \leq n$).

Differently, CPM3C [31], the most efficient multiclass MMC algorithm, has a space complexity of $O((n+d)D + ndm)$. D is the dimensionality of the input data in the case of linear kernels. In the nonlinear case, the authors suggest to exploit an explicit embedding with Kernel PCA [14], and $D \leq n$, so that the worst case is $O(n^2 + ndm)$. The variable m is the average number of cutting-plane iterations, and it is problem dependent for multiclass data (theoretical results are given only in the 2-class case [31]). Moreover it depends on the degree of fulfillment of the problem constraints, that are an exponential number (i.e. $(d+1)^n$ [31]). In Section IV, we show that this space requirements can make CPM3C impractical in some benchmarks, whereas we have been able to efficiently complete them with MEE.

Training an MEE has time complexity $O(n^2 \tilde{d} T)$ due to the KA product, where T is the number of CG iterations. We experimentally evaluated that we can find a solution in a just few steps, as we investigate in Section IV. Differently, from a practical point of view CPM3C scales roughly linearly with n when using a linear kernel, whereas in the nonlinear case it becomes $O(n^3)$ due to computational burden of the data embedding process.

IV. Experimental Results

Before discussing more detailed comparisons, the outcome of MEE-based clustering on some synthetic datasets is reported in Fig. 1, showing the capability of MEE to correctly separate clusters in non-dense regions. The first three figures are from popular data sets that are commonly used to assess the performances of manifold regularization algorithms [40]. In those algorithms the geometry of the data is estimated from the Laplacian matrix, that must be explicitly computed beforehand. MEE achieves similar results only using entropy-related information indexes. The rightmost picture is the outcome of clustering 3 Gaussian distributions in $\mathbb{R}^{10}$ that are linearly separable only on the first dimension. MEE correctly defines the separation boundaries along such dimension.

We tested MEE in several UCI data sets[5] [41], in the digit recognition benchmark USPS[6] (selecting the test portion, USPST), in the object recognition data set Coil20[7], as well as in the 20-Newsgroup data[8]. We used the original, not preprocessed data collections (if not differently specified), and the details of each data set are reported in Table I. The UCI data sets are of heterogeneous nature (medical diagnosis, house prices, ...) and the reader can find all the details in the provided reference to the UCI repository. USPS is a collection of handwritten digit pictures from the US Postal System, at the resolution of 16×16 pixels. Image rows were concatenated to generate vectors of 256 elements, and scaled in $[0, 1]$. We used the 10-class data as well as two binary versions, that discriminates between 1-7 and 3-8 (USPST1vs7, USPST3vs8, respectively). Coil20 is composed of images of 20 object categories acquired at different rotation angles, that we rescaled to the resolution of 32×32. The 20-Newsgroup data is composed of ≈ 20000 posts from 20 different newsgroups. We applied the Martin Porter's stemming algorithm [42] and extracted the $tf-idf$ (term frequency - inverse document frequency [43]) of each document from the resulting vocabulary of 44989 words. A binary version that discriminates between posts on Pc and Mac is also exploited (PcMac, preprocessed as in [40]).

We compared MEE with the related clustering algorithms, using publicly accessible code and, when possible, the implementation of the respective authors: K-Means (random initialization), Mean Shift (MS) [5], Spectral Clustering[9] (normalized cut, NC) [6], GMMC[10] [10], ISVR[11] [11], CPMMC[12] [12] and CPM3C[13] [31], (that we compactly refer to as CPM(M/3)C), LGMMC[14] [15]. MEE has been implemented in MATLAB 7.11 and all the experiments are ran on a machine with a 2.33 GHz Intel Core 2 Duo and 4GB Ram.

To assess the clustering accuracy, we followed the strategy of [9], used in most of the existing experimental comparisons involving MMC. In particular, we removed the class-labels, ran the clustering algorithms, labeled each of the resulting clusters with the majority class according to the original training labels, and finally measured the percentage of correct classifications made by each clustering. We also measured the Adjusted Rand Index (ARI), that considers the pairwise assignments of the data points. The Rand Index (RI) [45] is the number of pairs with the same label that are assigned to the same cluster (N_{11}) plus the number of the pairs with different labels that are assigned to different clusters (N_{00}), over all the possible pairs. ARI is the RI adjusted for chance, and it ranges in $[0, 1]$ where 0 means that RI corresponds to its expected value (the importance of the use of adjusted measures in discussed in [46]). More formally,

$$ARI = \frac{2(N_{00}N_{11} - N_{01}N_{10})}{(N_{00}+N_{01})(N_{01}+N_{11}) + (N_{00}+N_{10})(N_{10}+N_{11})},$$

where N_{01} is the number of pairs that are assigned to same cluster but with different labels, and N_{10} is the number of pairs that are in different clusters but with the same label.

[5] http://archive.ics.uci.edu/ml/

[6] http://www-i6.informatik.rwth-aachen.de/~keysers/usps.html

[7] http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

[8] http://people.csail.mit.edu/jrennie/20Newsgroups/

[9] http://alumni.cs.ucsb.edu/~wychen/sc.html

[10] http://www.cs.pitt.edu/~valizadegan/

[11] http://www.cse.ust.hk/~twinsen/

[12] https://sites.google.com/site/binzhao02/

[13] http://binzhao02.googlepages.com/Code_MMC_v1.rar, as in [44]

[14] http://lamda.nju.edu.cn/code_LGMMC.ashx

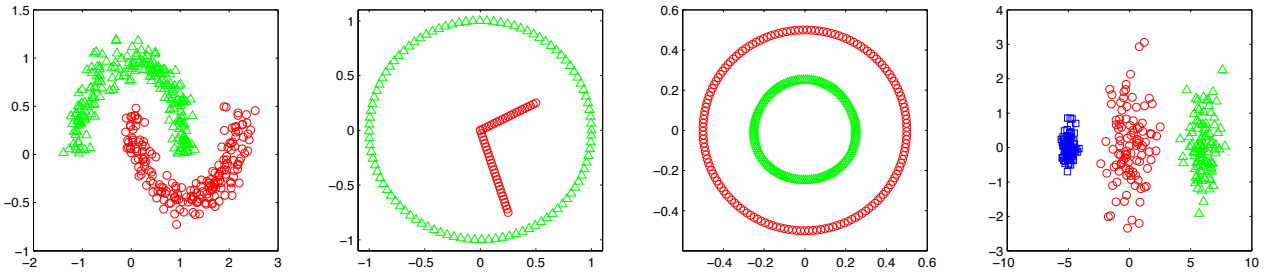

Fig. 1. Examples of clustering results produced by MEE (different color/markers represent data from different classes). The rightmost data set is composed of 3 Gaussian distributions in $\mathbb{R}^{10}$, separable only on the first dimension (the first 2 ones are plotted).

The number of clusters was set to the number of classes of each data collection. In all the kernel-based algorithms we used a Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right)$, with σ selected from $\{0.25\sigma_0, 0.33\sigma_0, 0.5\sigma_0, \sigma_0, 2\sigma_0, 3\sigma_0, 4\sigma_0\}$, σ_o being the average pairwise distance of $\mathcal{X}$. Then λ has been selected from $\{10^{-5}, 10^{-4}, 10^{-3}, \ldots, 10^{2}\}$ ($C = 1/\lambda$ in GMMC, ISVR, CPM(M/3)C), whereas μ has been set to 0.5 for binary data sets and $\mu = 0.7$ for multi-class collections. MEE and ISVR were initialized using the K-Means algorithm (if not differently specified), whereas the other non-convex algorithms lacks of a specific strategy, and their starting point was randomly generated. The bandwidth of the MS (flat) kernel was tuned in different ranges, depending on the data set, in order to produce the appropriate number of clusters. NC was ran using different number of neighbors to compute the Laplacian, $\{6, 12, 24, 48\}$. Other algorithm-specific parameters have been set following similar strategies to the ones suggested by the respective authors. In particular: $\epsilon = 0.01$, $\ell \in \{0.03n, 0.15n\}$ in the case of ISVR [11]; $\beta \in \{0.03n, 0.3n\}$ for LGMMC [15]; $\epsilon = 0.01$, $\ell \in \{0, 1, 10, 20\}$ in the case of CPM(M/3)C [31]. In the case of the 20 Newsgroups data, we set $\sigma = 200$, $\lambda = 10^{-30}$, and initialized MEE using the NC outcome, where the optimal number of neighbors to compute the Laplacian was found to be 2000. For each method and each data set, we report the result with the best parameter values belonging to the listed sets, as commonly done in the related literature [11].

TABLE II reports the details and the accuracies of our experimental comparison. The algorithms that may incur in suboptimal solutions have been ran 20 times for each parameter set; in the table we show the average accuracies (and the standard deviations). We can see that in many cases MEE has better performance than exiting state-of-the-art algorithms, while in other cases, the performance is at least comparable to the other methods.

Similar observations can be made about the ARI metric from Table (TABLE III). Notice that it was not possible to execute CPM3C in our machine in the case of data sets with a large number of classes and points (out of memory, even after slightly increasing ϵ), due to issue discussed in Section III-A, whereas we efficiently completed the experiments with MEE.

The impact of the initialization strategy in the MEE training stage is investigated in TABLE IV. Interestingly, seeding the algorithm with K-Means leads to better clustering accuracies.

TABLE I
DETAILS OF THE CONSIDERED DATA SETS.

Data	Size	Dimensions	Classes
Ionosphere	351	34	2
Heart	270	13	2
Musk1	476	166	2
German	1000	24	2
Breast	569	30	2
Diabetes	768	8	2
Echocg	132	8	2
Uspst1vs7	446	256	2
Uspst3vs8	338	256	2
Pcmac	1946	7511	2
Iris	150	4	3
Balance	625	4	3
Wine	178	13	3
Boston	506	13	3
Coil20	1440	1024	20
Uspst	2007	256	10
20News	18774	44989	20

The standard deviation is generally reduced with respect to a random initialization, but this depends on the stability of K-Means over multiple executions. For each run, we also tried to repeat the MEE training 10 times, and to select the most promising solution using the strategy described in Section III (the columns marked with "+" in TABLE IV). In the case of random initialization, this approach is very effective in avoiding inaccurate solutions, and this approach is also preferable in the case of k-means based initialization.

We compared the training times (including the time spent in kernel evaluations) of MEE with the publicly available implementation of ISVR, one of the faster MMC algorithms. From TABLE V , we observe that MEE converges faster than ISVR on average. We also report the number of CG iterations required to converge to a solution with gradient norm smaller that 10^{-10} (ϵ), that we found to be always significantly smaller than the number of data points. Finally, Fig. 2 illustrates the sensitivity of MEE to the parameter μ. Small values of μ can lead to degenerate solutions with one or more empty clusters, whereas too large values may degrade the quality of the data partitioning. The figure gives an idea of the optimal choice to achieve a good trade-off between the two entropies in (7), so as to get accurate groupings and to avoid trivial solutions. In the case of multi-class data (bottom row), a larger values

TABLE II
CLUSTERING ACCURACIES (%) OF MEE AND RELATED ALGORITHMS, AVERAGED OVER MULTIPLE RUNS FOR NON-CONVEX APPROACHES (STANDARD DEVIATION IN BRACKETS). SOME ALGORITHMS ARE LIMITED TO 2-CLASS DATA (GMMC REQUIRED MORE THAN 5 HOURS IN PCMAC). CPM(M/3)C HAD TOO LARGE MEMORY REQUIREMENTS ON THE LAST 3 BENCHMARKS.

Data	KM	MS	NC	GMMC	ISVR	CPM(M/3)C	LGMMC	MEE
Ionosphere	71.23 (0.00)	64.39 (0.00)	70.66	77.49	77.32 (0.15)	71.23 (0.00)	74.07	**88.03 (0.00)**
Heart	76.07 (7.59)	**78.98 (3.38)**	78.89	77.04	74.00 (9.53)	77.33 (6.41)	72.59	77.70 (3.80)
Musk1	56.51 (0.00)	56.13 (0.13)	56.51	56.51	62.14 (9.27)	56.85 (0.74)	62.82	**63.32 (2.69)**
German	70.00 (0.00)	70.10 (0.00)	70.00	70.00	70.00 (0.00)	70.00 (0.00)	70.00	**70.70 (0.00)**
Breast	92.79 (0.00)	92.13 (0.00)	**94.73**	85.06	87.35 (0.37)	92.97 (0.14)	81.37	92.97 (0.00)
Diabetes	66.02 (0.00)	65.16 (0.07)	65.10	66.15	65.10 (0.00)	65.46 (1.11)	**68.75**	66.21 (1.85)
Echocg	81.82 (0.00)	81.82 (0.00)	81.82	81.82	81.82 (0.00)	81.82 (0.00)	81.82	**82.35 (0.72)**
Uspst1vs7	98.65 (0.00)	94.44 (11.77)	99.10	98.65	90.27 (0.22)	93.77 (12.33)	97.98	**99.78 (0.00)**
Uspst3vs8	89.76 (1.19)	67.93 (16.36)	95.86	88.76	92.25 (0.57)	88.82 (2.02)	87.28	**97.04 (0.00)**
Pcmac	81.93 (15.23)	57.52 (0.18)	90.08	-	**94.36 (0.57)**	86.84 (13.14)	56.17	94.02 (0.52)
Iris	82.53 (10.95)	85.27 (9.95)	92.00	-	-	86.32 (4.45)	-	**96.67 (0.00)**
Balance	65.71 (3.72)	63.63 (9.62)	62.72	-	-	66.45 (7.57)	-	**68.54 (2.65)**
Wine	69.66 (1.18)	67.08 (1.82)	72.47	-	-	70.42 (0.99)	-	**73.03 (0.00)**
Boston	65.02 (0.00)	65.59 (0.15)	65.02	-	-	65.42 (0.00)	-	**65.61 (0.21)**
Coil20	66.57 (2.52)	30.10 (1.41)	**84.79**	-	-	-	-	73.50 (3.74)
Uspst	72.50 (1.51)	36.29 (5.87)	77.43	-	-	-	-	**77.57 (0.42)**
20News	13.00 (3.32)	12.17 (8.44)	39.02	-	-	-	-	**48.91 (0.00)**

TABLE III
ADJUSTED RAND INDEX (ARI) OF MEE AND RELATED ALGORITHMS, AVERAGED OVER MULTIPLE RUNS FOR NON-CONVEX APPROACHES (STANDARD DEVIATION IN BRACKETS). SOME ALGORITHMS ARE LIMITED TO 2-CLASS DATA (GMMC REQUIRED MORE THAN 5 HOURS IN PCMAC). CPM(M/3)C HAD TOO LARGE MEMORY REQUIREMENTS ON THE LAST 3 BENCHMARKS.

Data	KM	MS	NC	GMMC	ISVR	CPM(M/3)C	LGMMC	MEE
Ionosphere	0.18 (0.00)	0.00 (0.00)	0.17	0.30	0.30 (0.00)	0.18 (0.00)	0.21	**0.58 (0.00)**
Heart	0.29 (0.12)	**0.34 (0.08)**	0.33	0.29	0.26 (0.15)	0.31 (0.10)	0.20	0.31 (0.07)
Musk1	0.00 (0.00)	0.00 (0.00)	0.00	0.00	**0.09 (0.13)**	0.00 (0.01)	0.06	0.07 (0.03)
German	0.03 (0.00)	0.01 (0.00)	0.03	0.00	0.01 (0.00)	0.00 (0.00)	0.02	**0.06 (0.00)**
Breast	0.73 (0.00)	0.74 (0.00)	**0.80**	0.49	0.56 (0.01)	0.74 (0.00)	0.38	0.74 (0.00)
Diabetes	0.07 (0.00)	0.02 (0.00)	0.00	**0.10**	0.02 (0.01)	0.04 (0.05)	0.08	0.08 (0.05)
Echocg	0.12 (0.00)	0.12 (0.00)	0.12	0.00	0.02 (0.00)	0.00 (0.00)	0.00	**0.17 (0.08)**
Uspst1vs7	0.95 (0.00)	0.96 (0.03)	0.96	0.95	0.65 (0.01)	0.82 (0.33)	0.92	**0.99 (0.00)**
Uspst3vs8	0.63 (0.04)	0.45 (0.03)	0.84	0.60	0.71 (0.02)	0.60 (0.06)	0.55	**0.88 (0.00)**
Pcmac	0.49 (0.32)	0.01 (0.02)	0.64	-	**0.79 (0.02)**	0.60 (0.24)	0.01	0.78 (0.02)
Iris	0.64 (0.15)	0.71 (0.08)	0.79	-	-	0.73 (0.07)	-	**0.90 (0.00)**
Balance	0.14 (0.05)	0.16 (0.16)	0.09	-	-	0.18 (0.14)	-	**0.20 (0.05)**
Wine	0.36 (0.02)	0.39 (0.04)	0.38	-	-	0.37 (0.03)	-	**0.42 (0.00)**
Boston	0.09 (0.04)	0.13 (0.00)	0.00	-	-	**0.14 (0.00)**	-	0.13 (0.00)
Coil20	0.55 (0.03)	0.18 (0.01)	**0.76**	-	-	-	-	0.68 (0.03)
Uspst	0.53 (0.01)	0.38 (0.00)	0.58	-	-	-	-	**0.65 (0.01)**
20News	0.05 (0.02)	0.05 (0.03)	0.21	-	-	-	-	**0.34 (0.00)**

of μ is generally preferable, as expected, since it reduces the probability of generating empty clusters. When no prior knowledge is available, this experimental analysis suggests that setting μ to values slightly larger than 0.5 (for instance $\mu = 0.7$) can be a proper choice.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose Minimal Entropy Encoding (MEE), a clustering algorithm which bridges naturally information-theoretic principles and kernel methods. MEE detects distinctive properties of the patterns thanks to the minimization of an appropriate entropy measure, that is balanced in such a way to avoid trivial solutions. This is made possible by adding to the conditional entropy another entropy-based term to favor the development of a balanced number of features. It is shown that the proposed information-based objective function can nicely be expressed in terms of a proper KL-distance measure, which reduces to the mutual information between the developed code and the input in the case in which the two entropy terms are equally weighed. The problem of developing the coding functions is studied under the framework of regularization, which leads to their expression in terms of kernel expansion.

Unlike most of the Maximum Margin Clustering algorithms, MEE naturally handles multi-class data sets. In addition it makes use of optimization schemes based on non-linear conjugate gradient and concave-convex procedures, which make it very efficient. Our massive experimental analysis shows that MEE overcomes state-of-the art techniques on many of the selected benchmarks and, moreover, it exhibits remarkable

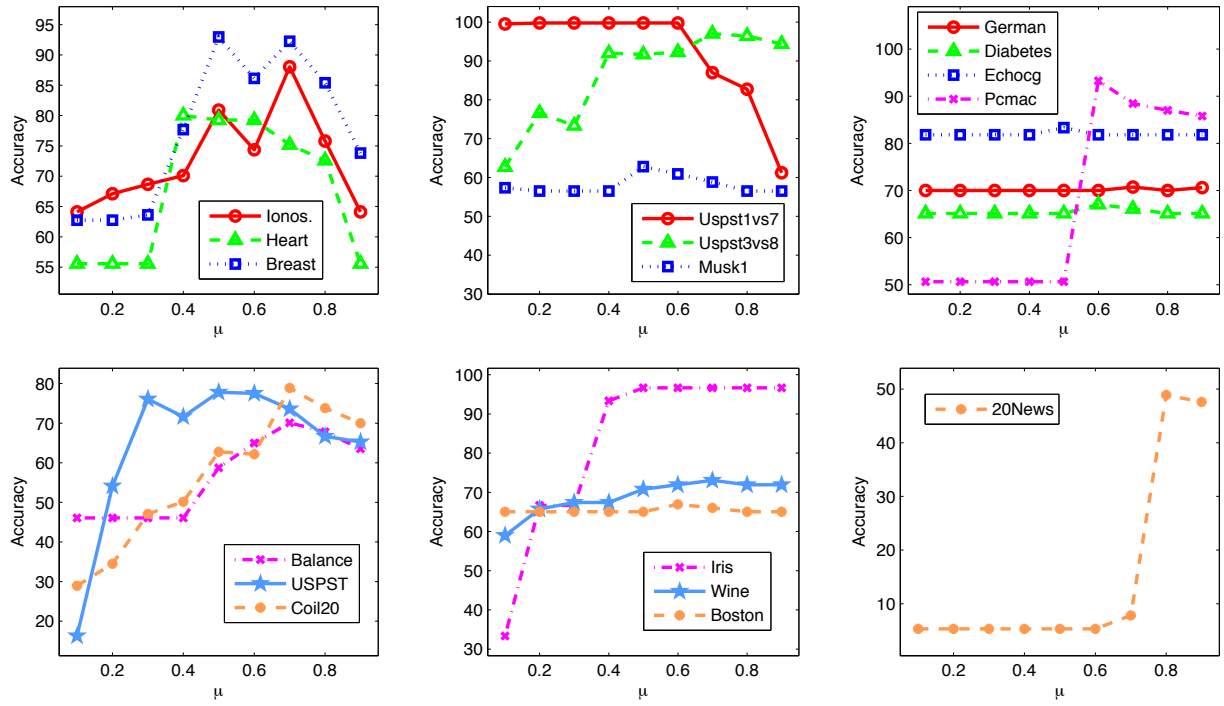

Fig. 2. Accuracy of MEE in function of the parameter μ, in the case of 2-class (top row) and multi class data (bottom row). In the latter case, larger values of μ is generally preferable in order to reduce the probability of generating empty clusters.

TABLE IV
ACCURACY OF MEE WITH RANDOM AND KMEANS-BASED INITIALIZATIONS. "+" INDICATES THAT FOR EACH RUN, MEE ARE TRAINED 10 TIMES, RETAINING THE SOLUTION WITH MINIMAL CONDITIONAL ENTROPY.

Data	Random	Random+	K-Means	K-Means+
Ionos.	68.03 (5.96)	74.93 (8.50)	87.92 (0.15)	**88.03 (0.00)**
Heart	74.70 (7.06)	77.33 (3.55)	61.19 (3.35)	**77.70 (3.80)**
Musk1	58.93 (4.56)	57.46 (1.52)	58.40 (3.04)	**63.32 (2.69)**
German	70.02 (0.06)	70.02 (0.06)	**70.70 (0.00)**	**70.70 (0.00)**
Breast	88.01 (9.30)	90.93 (3.81)	**92.97 (0.00)**	**92.97 (0.00)**
Diabetes	65.52 (0.90)	65.10 (0.00)	65.10 (0.00)	**66.21 (1.85)**
Echocg	81.89 (0.24)	81.97 (0.48)	81.82 (0.00)	**82.35 (0.72)**
Uspst17	**99.78 (0.00)**	**99.78 (0.00)**	**99.78 (0.00)**	**99.78 (0.00)**
Uspst38	96.66 (1.22)	96.98 (0.12)	71.12 (22.33)	**97.04 (0.00)**
Pcmac	53.58 (8.75)	53.79 (7.55)	92.92 (0.82)	**94.02 (0.52)**
Iris	85.87 (8.64)	94.13 (4.08)	92.00 (5.19)	**96.67 (0.00)**
Balance	58.59 (8.52)	59.07 (7.74)	63.62 (6.30)	**68.54 (2.65)**
Wine	67.81 (8.25)	71.40 (3.69)	70.00 (3.48)	**73.03 (0.00)**
Boston	65.22 (0.00)	65.22 (0.00)	65.22 (0.00)	**65.61 (0.21)**
Coil20	71.94 (2.83)	**73.51 (5.72)**	70.99 (3.06)	73.50 (3.74)
Uspst	72.90 (4.55)	**78.08 (1.21)**	75.76 (2.26)	77.57 (0.42)
20News	7.31 (1.3)	8.40 (2.35)	16.96 (5.01)	**18.93 (3.06)**

TABLE V
AVERAGE TRAINING TIMES (SECONDS) OF MEE AND ITERSVR (ISVR), AND THE NUMBER OF CONJUGATE GRADIENT (CG) ITERATIONS. THE TRAINING TIMES INCLUDE THE TIME SPENT IN KERNEL EVALUATIONS (I.E. TO COMPUTE THE GRAM MATRIX)

Data	ISVR Time	MEE Time	MEE CG Iters
Ionosphere	0.25 (0.05)	0.06 (0.01)	45.00 (0.00)
Heart	0.07 (0.01)	0.02 (0.00)	15.20 (0.42)
Musk1	1.92 (0.18)	0.14 (0.07)	99.30 (59.24)
German	2.07 (0.16)	0.14 (0.01)	10.00 (0.00)
Breast	2.97 (0.45)	0.18 (0.01)	50.00 (0.00)
Diabetes	0.93 (0.12)	0.08 (0.00)	7.00 (0.00)
Echocg	0.07 (0.01)	0.01 (0.00)	4.00 (0.00)
Uspst1vs7	1.13 (0.10)	0.05 (0.00)	29.00 (0.00)
Uspst3vs8	0.96 (0.13)	0.11 (0.03)	110.30 (28.85)
Pcmac	56.11 (1.03)	10.99 (5.14)	334.80 (215.88)
Iris	-	0.14 (0.07)	153.60 (95.03)
Balance	-	0.45 (0.21)	172.60 (90.18)
Wine	-	0.04 (0.01)	52.30 (12.88)
Boston	-	0.09 (0.00)	28.90 (1.10)
Coil20	-	7.62 (1.81)	280.00 (70.17)
Uspst	-	7.97 (3.22)	236.40 (102.30)
20News	-	852.12 (0.04)	45.00 (0.00)

performance also for those tasks in which it does not reach the best score.

A limitation of MEE is that we need to choose the number of clusters to develop in advance. However, the effect of the regularization term is that of favoring a sort of "natural" number of non-null features, since the regularization penalizes the development of a high number of features. We illustrate this point in Fig. 3. This is in fact a strong property of the proposed entropy encoding scheme that needs further investigation.

APPENDIX A
PROOF OF THEOREM 1

Proof: The proof can easily be given by following the classic scheme reported in [14] p. 90-91. We show that the kernel-based representation of the solution is essentially the outcome of evaluating D_μ over a sample of n elements

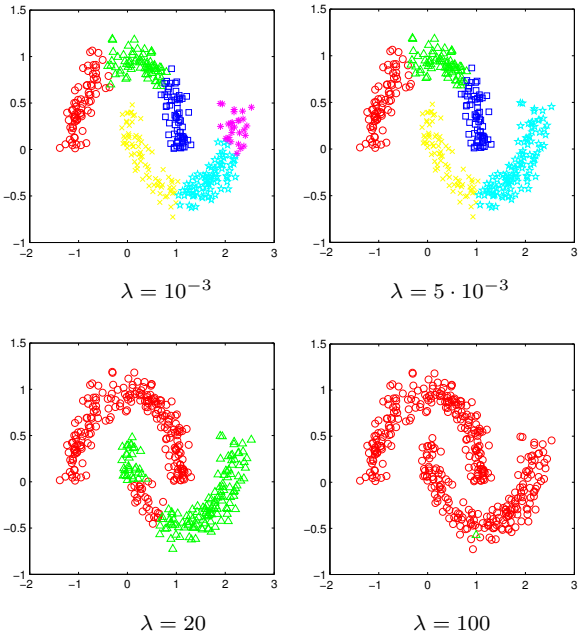

Fig. 3. The role of λ on the clustering outcomes. This example shows an instance of MEE-based clustering setting a number of clusters $d = 6$ that is different from the "optimal" one ($d_{opt} = 2$). The entropy balancing term was set to $\mu = 0.5$ whereas the weight λ of the RKHS norm of f_j, $j = 1, \ldots, d-1$ is iteratively increased from 10^{-3} to 10^2. Larger values of λ produces a larger number of empty cluster due to the small norm of several f_j. Within a certain range of λ values, we can get clusters that are closer to the ideal case.

$\{\boldsymbol{x}_i,\ i = 1, \ldots, n\}$. Any solution $f_j \in \mathcal{H}$ can be written as

$$f_j(\boldsymbol{x}) = f_j^{\parallel}(\boldsymbol{x}) + f_j^{\perp}(\boldsymbol{x}),$$

where $f_j^{\parallel}(\cdot) \in \mathcal{K}_n = span\{k(\cdot, \boldsymbol{x}_1), \ldots, k(\cdot, \boldsymbol{x}_n)\} \subset \mathcal{H}$, while $f_j^{\perp}(\cdot)$ is in the orthogonal complement of $\mathcal{K}_n$. Now, in order to compute $D_\mu(X, Y(\boldsymbol{f}))$ we only need the values $f_j(\boldsymbol{x}_i)$. Now, $\forall \kappa = 1, \ldots, n$:

$$\begin{aligned} f_j(\boldsymbol{x}_\kappa) &= < f_j(\cdot), k(\boldsymbol{x}_\kappa, \cdot) > \\ &= \sum_{i=1}^{n} \alpha_{ij} < k(\boldsymbol{x}_i, \cdot), k(\boldsymbol{x}_\kappa, \cdot) > + < f_j^{\perp}(\cdot), k(\boldsymbol{x}_\kappa, \cdot) > \\ &= \sum_{i=1}^{n} \alpha_{ij} < k(\boldsymbol{x}_i, \boldsymbol{x}_\kappa) > . \end{aligned}$$

For the regularization term we have

$$\begin{aligned} \| f_j \| &= < \sum_{i=1}^{n} \alpha_{ij} k(\boldsymbol{x}_i, \cdot) + f_j^{\perp}(\cdot), \sum_{i=1}^{n} \alpha_{ij} k(\boldsymbol{x}_i, \cdot) + f_j^{\perp}(\cdot) > \\ &= \| \sum_{i=1}^{n} \alpha_{ij} k(\boldsymbol{x}_i, \cdot) \|^2 + \| f_j^{\perp}(\cdot) \|^2 \\ &\geq \| \sum_{i=1}^{n} \alpha_{ij} k(\boldsymbol{x}_i, \cdot) \|^2 . \end{aligned}$$

Finally, for any $\alpha_{ij} \in \mathbb{R}$ the minimization of $C(\boldsymbol{f})$ is reached for $\boldsymbol{f}^{\perp} = 0$, from which the thesis follows. ■

APPENDIX B
GRADIENT COMPUTATION

In this appendix we describe how to compute the gradient matrix $\nabla C(A) \in \mathbb{R}^{n,\tilde{d}}$ and how to simplify the optimization avoiding a product by K. We recall that $\boldsymbol{f}$ depends on A, and that $\nabla C(A)$ is the sum of three elements,

$$\begin{aligned} \nabla C(A) &= (1-\mu)\nabla H(Y(\boldsymbol{f})|X) \\ &\quad -\mu \nabla H(Y(\boldsymbol{f})) + \lambda \nabla \|\boldsymbol{f}\|_K . \end{aligned} \tag{14}$$

Considering that $f_j(\boldsymbol{x}_i) = \boldsymbol{k}_i^T \boldsymbol{\alpha}_j$ (10), we have

$$\begin{aligned} \frac{\partial H(Y(\boldsymbol{f})|X)}{\partial \alpha_{zh}} &= -\sum_{i=1}^{n}\sum_{j=1}^{\tilde{d}} \left(\frac{\partial p_{ij}}{\partial \alpha_{zh}} \cdot \boldsymbol{k}_i^T \boldsymbol{\alpha}_j + p_{ij} \cdot \frac{\partial \boldsymbol{k}_i^T \boldsymbol{\alpha}_j}{\partial \alpha_{zh}} \right) \\ &\quad + \sum_{i=1}^{n} p_{ih} \cdot \frac{\partial \boldsymbol{k}_i^T \boldsymbol{\alpha}_h}{\partial \alpha_{zh}} . \end{aligned} \tag{15}$$

The derivatives of p_{ij} and $\boldsymbol{k}_i^T \boldsymbol{\alpha}_j$ with respect to α_{zh} can be straightforwardly computed, and they can be expressed using the Kronecker delta δ_{hj} ($\delta_{hj} = 1$ if $h = j$, otherwise $\delta_{hj} = 0$),

$$\frac{\partial p_{ij}}{\partial \alpha_{zh}} = (p_{ij}\delta_{hj} - p_{ij}p_{ih}) \cdot k_{iz}, \tag{16}$$

$$\frac{\partial \boldsymbol{k}_i^T \boldsymbol{\alpha}_j}{\partial \alpha_{zh}} = \delta_{hj} k_{iz}. \tag{17}$$

Plugging them into (15), we get

$$\frac{\partial H(Y(\boldsymbol{f})|X)}{\partial \alpha_{zh}} = -\sum_{j=1}^{\tilde{d}}\sum_{i=1}^{n} k_{iz}(p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j\delta_{hj} - p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j p_{ih}).$$

Switching to matrix notation, we indicate with $[a_{ih}]_{h=1,\ldots,d}^{i=1,\ldots,n}$ the matrix with n rows and d columns whose elements are given by varying the indexes i and h of a_{ih}. We have

$$\begin{aligned} &\nabla H(Y(\boldsymbol{f})|X) = \\ &= -\left[\sum_{j=1}^{\tilde{d}}\sum_{i=1}^{n} k_{iz}(p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j\delta_{hj} - p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j p_{ih})\right]_{h=1,\ldots,\tilde{d}}^{z=1,\ldots,n} \\ &= -\sum_{j=1}^{\tilde{d}} K\left[p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j\delta_{hj} - p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j p_{ih}\right]_{h=1,\ldots,\tilde{d}}^{i=1,\ldots,n} \\ &= -K\left(\sum_{j=1}^{\tilde{d}}\left[p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j\delta_{hj}\right]_{h=1,\ldots,\tilde{d}}^{i=1,\ldots,n} - \sum_{j=1}^{\tilde{d}}\left[p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j p_{ih}\right]_{h=1,\ldots,\tilde{d}}^{i=1,\ldots,n}\right) \\ &= -K\left(\tilde{P}\circ(KA) - \sum_{j=1}^{\tilde{d}}\left(\left[p_{ij}\boldsymbol{k}_i^T\boldsymbol{\alpha}_j\right]_{i=1,\ldots,n}^{T}\mathbf{1}_{\tilde{d}}^T\right)\circ\tilde{P}\right) \\ &= -K\left(\tilde{P}\circ(KA) - \tilde{P}\circ\left((\tilde{P}\circ(KA))\mathbf{1}_{\tilde{d}}\mathbf{1}_{\tilde{d}}^T\right)\right) \\ &= -K\left(\tilde{P}\circ\left(KA - (\tilde{P}\circ(KA))\mathbf{1}_{\tilde{d}}\mathbf{1}_{\tilde{d}}^T\right)\right). \end{aligned} \tag{18}$$

We follow the same procedure to compute the gradient of $H(Y(\boldsymbol{f}))$. In detail,

$$\frac{\partial H(Y,F)}{\partial \alpha_{zh}} = -\sum_{j=1}^{d}\sum_{i=1}^{n}\frac{\partial p_{ij}}{\partial \alpha_{zh}}\left(\ln\left(\frac{1}{n}\sum_{r=1}^{n} p_{rj}\right) + 1\right).$$

After we introduce $s_j := \ln\left(\frac{1}{n}\sum_{r=1}^{n} p_{rj}\right) + 1$ (that we eventually collect in the vector $\boldsymbol{s} \in \mathbb{R}^d$), and we use the results of (16) and (17), we get

$$\frac{\partial H(Y(\boldsymbol{f}))}{\partial \alpha_{zh}} = -\sum_{j=1}^{d}\sum_{i=1}^{n}(s_j p_{ij}\delta_{hj} - s_j p_{ij} p_{ih}) \cdot k_{iz}.$$

Finally, passing to matrix notation,

$$\begin{aligned}
&\nabla H(Y(\boldsymbol{f})) = \\
&= -\left[\sum_{j=1}^{d}\sum_{i=1}^{n}(s_j p_{ij}\delta_{hj} - s_j p_{ij} p_{ih})k_{iz}\right]_{h=1,\dots,\tilde{d}}^{z=1,\dots,n} \\
&= -\sum_{j=1}^{d} K\left[s_j p_{ij}\delta_{hj} - s_j p_{ij} p_{ih}\right]_{h=1,\dots,\tilde{d}}^{i=1,\dots,n} \\
&= -K\left(\sum_{j=1}^{d}\left[s_j p_{ij}\delta_{hj}\right]_{h=1,\dots,\tilde{d}}^{i=1,\dots,n} - \sum_{j=1}^{d}\left[s_j p_{ij} p_{ih}\right]_{h=1,\dots,\tilde{d}}^{i=1,\dots,n}\right) \\
&= -K\left(\tilde{P}\circ(\mathbf{1}_n\tilde{\boldsymbol{s}}^T) - \sum_{j=1}^{d}\left(\left[s_j p_{ij}\right]_{i=1,\dots,n}^{T}\mathbf{1}_{\tilde{d}}^T\right)\circ\tilde{P}\right) \\
&= -K\left(\tilde{P}\circ(\mathbf{1}_n\tilde{\boldsymbol{s}}^T) - \tilde{P}\circ(P\boldsymbol{s}\mathbf{1}_{\tilde{d}}^T)\right) \\
&= -K\left(\tilde{P}\circ(\mathbf{1}_n(\ln\frac{1}{n}\mathbf{1}_n^T P)^T - P(\ln\frac{1}{n}\mathbf{1}_n^T P)\mathbf{1}_{\tilde{d}}^T)\right). \quad (19)
\end{aligned}$$

Now we can plug (18), (19) and $\nabla(\|\boldsymbol{f}\|_K) = 2KA$ into (14), and complete the derivation of the gradient,

$$\begin{aligned}
\nabla C(A) = &\frac{\mu-1}{n}K\left(\tilde{P}\circ\left(KA - (\tilde{P}\circ(KA))\mathbf{1}_{\tilde{d}}\mathbf{1}_{\tilde{d}}^T\right)\right) \\
&+\frac{\mu}{n}K\left(\tilde{P}\circ(\mathbf{1}_n(\ln\frac{1}{n}\mathbf{1}_n^T P)^T - P(\ln\frac{1}{n}\mathbf{1}_n^T P)\mathbf{1}_{\tilde{d}}^T)\right) \\
&+2\lambda KA. \quad (20)
\end{aligned}$$

Each stationary point of $C(A)$ satisfies $\nabla C(A) = 0\cdot\mathbf{1}_n\mathbf{1}_{\tilde{d}}^T$. We notice that the matrix K is a factor of all the three terms of (20). Following the strategy used in other kernel-based algorithm optimized by conjugate gradient [40], we can speed up the gradient computation and improve the optimization process if we select K^{-1} as a preconditioner of $\nabla C(A)$ [34]. In particular, we indicate with $\nabla' C(A) = K^{-1}\nabla C(A)$ the gradient preconditioned by K^{-1}. The matrix K^{-1} does not need to be explicitly calculated, since K is a factor of $\nabla C(A)$, and we can simply build $\nabla' C(A)$ without computing $\nabla C(A)$ first. This allows us to avoid a costly matrix-by-matrix product by K, and to speed up the gradient computation.[15] The set of equations of the stationary points becomes $\nabla' C(A) = 0\cdot\mathbf{1}_n\mathbf{1}_{\tilde{d}}^T$, and we can efficiently apply the CG procedure of Algorithm 1 replacing $\nabla C(A)$ with $\nabla' C(A)$. The details on the complexity of this solution are discussed in Section III-A.

Finally, the expansion of $f_j(\boldsymbol{x})$ can be augmented with a not-regularized bias b_j, i.e. $f_j(\boldsymbol{x}_i) = \boldsymbol{k}_i^T\boldsymbol{\alpha}_j + b_j$. Care must be taken when including the bias term, since we recall that all the constant solutions with $A = 0\cdot\mathbf{1}_n\mathbf{1}_{\tilde{d}}^T$ and $b_j = b_z$ for all j, z, are bad stationary points and must be avoided. For completeness, we report the essential points of the gradient computation, since it follows the same strategy that we previously used. We indicate the criterion function (11) with $C(A, \boldsymbol{b})$, where the vector $\boldsymbol{b}$ collects b_j, $j = 1,\dots,\tilde{d}$. We use the notation $\nabla_A C(A,\boldsymbol{b})$ and $\nabla_{\boldsymbol{b}} C(A,\boldsymbol{b})$ to distinguish between the gradient of $C(A,\boldsymbol{b})$ with respect to A and $\boldsymbol{b}$, respectively. Now we have to compute $\nabla_{\boldsymbol{b}} H(Y(\boldsymbol{f})|X)$ and $\nabla_{\boldsymbol{b}} H(Y(\boldsymbol{f}))$, whereas $\nabla_{\boldsymbol{b}}\|\boldsymbol{f}\|_K$ is zero, due to the not regularized nature of the bias terms. Using the intermediate result

$$\frac{\partial p_{ij}}{\partial b_h} = p_{ij}\delta_{hj} - p_{ij}p_{ih}, \quad (21)$$

it is easy to find that $\nabla_{\boldsymbol{b}} H(Y(\boldsymbol{f})|X)$ and $\nabla_{\boldsymbol{b}} H(Y(\boldsymbol{f}))$ correspond to simple variants of (18) and (19), respectively, where the first occurrence of K is substituted with $\mathbf{1}_n^T$. As a consequence, $\nabla_{\boldsymbol{b}} C(A,\boldsymbol{b})$ is the vector that collects the sum (column-wise) of the preconditioned gradient with respect to A, $\nabla'_A C(A,\boldsymbol{b})$, computed with $\lambda = 0$,

$$\nabla_{\boldsymbol{b}} C(A,\boldsymbol{b}) = \left(\mathbf{1}_n^T\cdot\nabla'_A C(A,\boldsymbol{b})\big|_{\lambda=0}\right)^T.$$

[15] We are assuming that in this expression K is non singular, otherwise we can suppose that a small ridge is added to fix it, as commonly done in other kernel-based approaches [40].

REFERENCES

[1] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[2] N. Manukyan, M. Eppstein, and D. Rizzo, "Data-driven cluster reinforcement and visualization in sparsely-matched self-organizing maps," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 846–852, 2012.

[3] C. Hsu and S. Lin, "Visualized analysis of mixed numeric and categorical data via extended self-organizing map," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 1, pp. 72–86, 2012.

[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of Berkeley Symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, 1967.

[5] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[6] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14: Proceeding of the 2001 Conference*, 2001, pp. 849–856.

[7] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik, "Support vector clustering," *The Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2002.

[8] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern recognition*, vol. 41, no. 1, pp. 176–190, 2008.

[9] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans, "Maximum margin clustering," in *Advances in Neural Information Processing Systems*, vol. 17, 2004, pp. 1537–1544.

[10] H. Valizadegan and R. Jin, "Generalized maximum margin clustering and unsupervised kernel learning," in *Neural Information Processing Systems*, 2006, pp. 1417–1424.

[11] K. Zhang, I. Tsang, and J. Kwok, "Maximum margin clustering made practical," *Neural Networks, IEEE Transactions on*, vol. 20, no. 4, pp. 583–596, 2009.

[12] F. Wang, B. Zhao, and C. Zhang, "Linear time maximum margin clustering," *Neural Networks, IEEE Transactions on*, vol. 21, no. 2, pp. 319–332, 2010.

[13] A. Yuille and A. Rangarajan, "The concave-convex procedure (cccp)," in *Advances in Neural Information Processing Systems*, vol. 2, 2002, pp. 1033–1040.

[14] B. Schoelkopf. and A. Smola, *Learning with kernels*. MIT Press, 2002.

[15] Y. Li, I. Tsang, J. Kwok, and Z. Zhou, "Tighter and convex maximum margin clustering," in *Proceeding of the 12th International Conference on Artificial Intelligence and Statistics*, 2009, pp. 344–351.

[16] J. Principe, *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Verlag, 2010.

[17] N. Vinh and J. Epps, "mincentropy: A novel information theoretic approach for the generation of alternative clusterings," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 521–530.
[18] D. Zhang, S. Chen, and Z. Zhou, "Entropy-inspired competitive clustering algorithms," *Int J Software Informatics*, vol. 1, no. 1, 2007.
[19] B. Dai and B. Hu, "Minimum conditional entropy clustering: A discriminative framework for clustering," *Journal of Machine Learning Research - Proceedings Track*, vol. 13, pp. 47–62, 2010.
[20] L. Faivishevsky and J. Goldberger, "A nonparametric information theoretic clustering algorithm," in *International Conference on Machine Learning*. ACM, 2010, pp. 351–358.
[21] L. Song, A. Smola, A. Gretton, and K. Borgwardt, "A dependence maximization view of clustering," in *International conference on Machine learning*. ACM, 2007, pp. 815–822.
[22] X.-T. Yuan and B.-G. Hu, "Robust feature extraction via information theoretic learning," in *Proceedings of the 26th International Conference on Machine Learning*, 2009, pp. 1193–1200.
[23] N. Lawrence, "Spectral dimensionality reduction via maximum entropy," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011, pp. 51–59.
[24] R. Jenssen, "Kernel entropy component analysis," *IEEE transactions on pattern analysis and machine intelligence*, pp. 847–860, 2009.
[25] R. He, B. Hu, X. Yuan, and W. Zheng, "Principal component analysis based on non-parametric maximum entropy," *Neurocomputing*, vol. 73, no. 10-12, pp. 1840–1852, 2010.
[26] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems*, vol. 17, 2004, pp. 529–536.
[27] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006. [Online]. Available: http://www.kyb.tuebingen.mpg.de/ssl-book
[28] Y. Wang, S. Chen, and Z. Zhou, "New semi-supervised classification method based on modified cluster assumption," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 689–702, 2012.
[29] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.
[30] L. Xu and D. Schuurmans, "Unsupervised and semi-supervised multi-class support vector machines," in *Proceedings Of The National Conference On Artificial Intelligence*, vol. 20, no. 2, 2005, pp. 904–910.
[31] B. Zhao, F. Wang, and C. Zhang, "Efficient multiclass maximum margin clustering," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1248–1255.
[32] T. Joachims, "Transductive inference for text classification using support vector machines," in *International Conference on Machine Learning*, 1999, pp. 200–209.
[33] T. Cover and J. Thomas, *Elements of Information Theory*. John Wiley & Sons, Inc, 1991.
[34] J. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Tech. Rep., 1994.
[35] Y. Saad, *Iterative methods for sparse linear systems*. Society for Industrial Mathematics, 2003.
[36] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Pr, 2004.
[37] B. Sriperumbudur and G. Lanckriet, "On the convergence of the concave-convex procedure," *Advances in neural information processing systems*, vol. 22, pp. 1759–1767, 2009.
[38] R. Rifkin and R. Lippert, "Notes on regularized least squares," MIT, Cambridge, MA, Tech. Rep. CBCL Paper, Tech. Rep., 2007.
[39] A. Caponnetto and E. De Vito, "Optimal rates for the regularized least-squares algorithm," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 331–368, 2007.
[40] S. Melacci and M. Belkin, "Laplacian Support Vector Machines Trained in the Primal," *Journal of Machine Learning Research*, vol. 12, pp. 1149–1184, March 2011.
[41] A. Frank and A. Asuncion, "UCI repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml
[42] K. Jones, *Readings in information retrieval*. Morgan Kaufmann, 1997.
[43] H. Wu, R. Luk, K. Wong, and K. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, pp. 1–37, 2008.
[44] F. Wang, X. Wang, and T. Li, "Maximum margin clustering on data manifolds," in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, 2009, pp. 1028–1033.
[45] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.
[46] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. of Mach. Learn. Res.*, vol. 12, pp. 2837–2854, 2010.

Stefano Melacci received the MS Laurea degree (cum Laude) in Computer Engineering from the University of Siena, Italy, in 2006, and the Ph.D. in Information Engineering (Adaptive Systems for Information Processing) from the same university in April 2010. From April to October 2009 he was a visiting scientist at the Department of Computer Science and Engineering, Ohio State University, Columbus, OH, USA. He is currently a Research Associate at the Department of Information Engineering, University of Siena, Italy. His research interests include machine learning and pattern recognition, mainly focused on regularization theory and learning from constraints with applications to computer vision. He is an active reviewer for many international conferences and journals, including IEEE Transaction on Neural Networks and Learning Systems, IEEE Transactions on Pattern Analysis and Machine Intelligence, Journal of Machine Learning Research.

Marco Gori received the Ph.D. degree in 1990 from University of Bologna, Italy. From October 1988 to June 1989 he was a visiting student at the School of Computer Science (McGill University, Montreal). In 1992, he became an Associate Professor of Computer Science at University of Firenze and, in November 1995, he joint the University of Siena, where he is currently full professor of computer science. His main interests are in machine learning, with applications to pattern recognition, Web mining, and game playing. He is especially interested in the formulation of relational machine learning schemes in the continuum setting. Prof. Gori serves (has served) as an Associate Editor of a number of technical journals related to his areas of expertise, including IEEE Transaction on Neural Networks, Pattern Recognition, Neural Networks, Neurocomputing, Pattern Analysis and Application, the International Journal of Document Analysis and Recognition, and the International Journal on Pattern Recognition and Artificial Intelligence. He has been the recipient of best paper awards and keynote speakers in a number of international conferences. He is the Chairman of the Italian Chapter of the IEEE Computational Intelligence Society, a fellow of the ECCAI and of the IEEE.