Kernel Methods for Minimum Entropy Encoding

Stefano Melacci and Marco Gori
Department of Information Engineering, University of Siena, 53100 Siena, Italy
{mela,marco}@dii.unisi.it

***Abstract*—Following the basic principles of Information-Theoretic Learning (ITL), in this paper we propose Minimum Entropy Encoders (MEEs), a novel approach to data clustering. We consider a set of functions that project each input point onto a minimum entropy configuration (*code*). The encoding functions are modeled by kernel machines and the resulting code collects the cluster membership probabilities. Two regularizers are included to balance the distribution of the output features and favor smooth solutions, respectively, thus leading to an unconstrained optimization problem that can be efficiently solved by conjugate gradient or concave-convex procedures. The relationships with Maximum Margin Clustering algorithms are investigated, which show that MEEs overcomes some of the 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.**

I. Introduction

Clustering is a central topic in machine learning that has a crucial impact in diverse domains, ranging from bioinformatics, medical science, computer vision to information retrieval [1]. Clustering algorithms aim at discovering the underlying structure of the data, grouping examples into a number of classes, or clusters, such that the entities belonging to the same cluster are "similar" to each other and "different" from the ones of the other clusters. Popular examples include K-Means, Mixture Models [1], and Spectral Clustering [2].

More recently, Maximum Margin Clustering (MMC) has received an increasing interest in the scientific community [3]. MMC extends the maximum margin principle to unsupervised learning, 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. The learning problem behind MMC is intrinsically complicated and several relaxations of the original formulation have been recently proposed to make it more affordable. Xu et al. [3] formulate a convex semi-definite programming problem, whereas Valizadegan and Jin [4] propose Generalized Maximum Margin Clustering (GMMC) that reduces the number of variables from $O(n^2)$ to $O(n)$, where n is the data set size. Zhang et al. [5] propose IterSVR, an iterative solution based on a set of support vector regression problems. Similarly, the cutting-plane based approach (CPMMC) in [6] has been proven to be an efficient method 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) [7]. Finally, in order to preserve convexity, a tighter and convex relaxation (LGMMC) of the original MMC is proposed in [8].

Inspired by the Information-Theoretic Learning (ITL) framework [9], different algorithms for feature extraction, dimensionality reduction, and clustering have been recently studied [10]–[12]. In particular, the conditional entropy can be interpreted as a measure of class overlap [13], which is related to the quality of the data partitioning. The minimization of the conditional entropy under a smoothness assumption converges to solutions that are close to MMC [12].

In this paper, we introduce Minimum Entropy Encoders (MEEs), that are designed to learn a set of functions projecting each input point onto a minimum entropy configuration (*code*) of length d, where d is the number of clusters. The code collects the cluster membership probabilities, and the encoding functions are modeled by means of kernel machines. The optimization problem of MEEs consists of the sum of three terms that embed three crucial properties in the encoders: minimize the conditional entropy, ensure that the distribution of the output features is balanced, and enforce smooth solutions.

There are three main contributions in this paper. First, we introduce MEEs as the solution of a minimization problem based on a suitable regularization of the entropy, and show that such a solution is given in terms of kernel machines. MEEs learn cluster separation hyperplanes in low-density regions and balance the distribution of the output features. This avoids any additional constraints [3], [4], [6], [8] or iterative label-switching mechanisms [12]. Second, we present efficient ways to solve the unconstrained MEE problem by means of conjugate gradient or CCCP. 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. Third, we show that MEE can handle multi-class tasks, whereas the described MMC implementations are designed for 2-class problems, with the exceptions of Xu et al. [14] and the multiclass version of CPMMC, referred to as CPM3C [15]. The proposed method, however, is better suited to face problems with a large number of classes. Finally, a massive evaluation on several benchmarks shows that MEEs overcomes state-of-the art techniques both in terms of accuracy and complexity.

II. Minimum Entropy Encoding

We consider a set of n unlabeled points, $X = \{\boldsymbol{x}_i \in \mathbb{R}^q, i = 1, \ldots, n\}$ that are supposed to be reasonably rep-

resented by a codebook of d symbols with corresponding emission probabilities. Following the established framework of learning with kernel machines [16], we model the d probabilities by a set of $d-1$ functions $F = \{f_z \in \mathcal{H},\ z = 1,\ldots,d-1\}$, where $\mathcal{H}$ is the Reproducing Kernel Hilbert Space (RKHS) induced by the kernel function $k(\cdot,\cdot)$.[1] In order to enforce a probabilistic normalization, we use the *softmax* function, so that for $j = 1,\ldots,d-1$ we have

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

and

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

Following Information Theoretic principles [9], the (Shannon) *entropy* of the i-code is $H(\boldsymbol{x}_i, F) = -\sum_{j=1}^{d} p_j(\boldsymbol{x}_i, F) \log p_j(\boldsymbol{x}_i, F)$. 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 only the most distinctive property of $\boldsymbol{x}_i$ or just a few of them, the encoder must produce output vectors that are close to minimum entropy configurations. For instance, if we know that the data in X is partitioned into d clusters, we can interpret $p_j(\boldsymbol{x}_i)$ as the probability of the j-th cluster given $\boldsymbol{x}_i$ and in case of hard-partitioning we can get the cluster index of $\boldsymbol{x}_i$ as $y_i = \arg\max_j p_j(\boldsymbol{x}_i, F)$, $y_i \in Y$. If we average $H(\boldsymbol{x}_i, F)$ over X, we get the empirical estimate of the *conditional entropy* [13], that do not require any assumptions on the distribution of the data,

$$H(Y|X,F) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d} p_j(\boldsymbol{x}_i, F) \log p_j(\boldsymbol{x}_i, F), \tag{3}$$

where, for simplicity, we overloaded the notation X and Y to also indicate the random variables associated to the data and the clustering outcomes. Previous studies [13], [17] have shown that the conditional entropy is an index of the class overlap, and desirable data partitions are associated with small $H(Y|X,F)$. This idea is closely related to the principles of Maximum Margin Clustering (MMC), that under some mild approximations can be reformulated using minimum conditional entropy arguments [12].

Unfortunately, the degenerate solutions in which all the data points are assigned to the same clusters minimizes the conditional entropy, so as 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 [3], [4], [8], [15], or to rely on label-switching mechanisms in the iterative clustering process [12]. However, in the case of multiple classes the number of balancing constraints is d^2, and the application of the label-switching is not straightforward. We propose to introduce a regularizer that is based on descriptors from Information Theory. If we indicate with $\hat{X}$ the set of the output codes associated to the data in X, $\hat{X} = \{\boldsymbol{o}_i, i = 1,\ldots,n\}$, we aim at ensuring that the d output features shares the same average probability over $\hat{X}$. Now, let us define $\hat{p}_j(X,F) = \frac{1}{n}\sum_{i=1}^{n} p_j(\boldsymbol{x}_i, F)$, $j = 1,\ldots,d$. We can immediately see that $\hat{p}_j(X,F)$ are probabilities and the corresponding entropy $H(\hat{X},F)$ (overloading the notation $\hat{X}$, as previously done), turns out to be

$$H(\hat{X},F) = -\frac{1}{n}\sum_{j=1}^{d}\sum_{i=1}^{n} p_j(\boldsymbol{x}_i, F) \log\left(\frac{1}{n}\sum_{i=1}^{n} p_j(\boldsymbol{x}_i, F)\right). \tag{4}$$

The maximum of $H(\hat{X},F)$ corresponds with configurations that are averagely balanced with $\hat{p}_j(X,F) = 1/d$, $j = 1,\ldots,d$. Without making any additional hypothesis, the problem is still ill-posed, since there are many configurations that minimize the conditional entropy while maximizing the class balancing term. Following the framework of regularization we can face the problem by enforcing smooth solutions, thus converting the problem to the minimization of

$$C(F) = (1-\mu)H(Y|X,F) - \mu H(\hat{X},F) + \lambda\Omega(F), \tag{5}$$

where $\Omega(F) = \sum_{z=1}^{d-1} \|f_z\|_k^2$ is the norm associated with the kernel k and the parameter, μ, $0 < \mu < 1$ is a weight that measures the combination of the two entropies. Notice the different selection of the signs in the two terms that allows us to maximize $H(\hat{X},F))$, while minimizing $C(F)$. Moreover, $\lambda > 0$ is the classic regularization parameter to enforce the smoothness of the solution. Since $-\mu H(\hat{X},F)$ ranges in $[-\mu\log d, 0]$ we can eventually add the offset $\mu\log d$ to ensure that (5) is nonnegative. Since we do not know in advance if the d clusters are balanced, a soft-constraining scheme with a tunable parameter μ appears an appropriate choice. When scaling the terms of (4) properly, we can easily realize that the optimal solution of the problem is independent of the $\log$ base. For the sake of simplicity, we define $p_{ij} := p_j(\boldsymbol{x}_i, F)$, and $w_i := 1 + \sum_{z=1}^{d-1} e^{f_z(\boldsymbol{x}_i)}$. In turn, the conditional entropy $H(Y|X,F)$ (3) can be rewritten as

$$\begin{aligned} H(Y|X,F) &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d} p_{ij}\ln\left(p_{ij}\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d-1} p_{ij}\ln\left(\frac{e^{f_j(\boldsymbol{x}_i)}}{w_i}\right) - \frac{1}{n}\sum_{i=1}^{n} p_{id}\ln\left(\frac{1}{w_i}\right) \\ &= -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d-1} p_{ij} f_j(\boldsymbol{x}_i) + \frac{1}{n}\sum_{i=1}^{n}\ln\left(w_i\right). \end{aligned} \tag{6}$$

Interestingly, the hypothesis of looking for solutions belonging into the RKHS of k has a strong consequence in the optimization of 4.

Theorem 2.1: Let $\overline{f}_j \in \mathcal{H}$, where $\mathcal{H}$ is the RKHS of kernel $k(\cdot,\cdot)$. Then $f_j = \arg\min_{\overline{f}_j \in \mathcal{H}} C(F)$ is given by

$$f_j(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_{ij} k(\boldsymbol{x}_i, \boldsymbol{x})$$

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

Proof: The proof can easily be given by following the classic scheme reported in [16] p. 90-91. ∎

Now, let $K \in I\!R^{n,n}$ be the Gram matrix associated to the points in X, $\boldsymbol{k}_j \in I\!R^n$ is j-column of K, and k_{ij} is the element in position (i,j). Analogously, $A \in I\!R^{n,d-1}$ is the matrix of the coefficients α_{ij}, and $\boldsymbol{\alpha}_j \in I\!R^n$ is its j-th column. $P \in I\!R^{n,d}$ 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, (5) can be written in matrix notation as

$$\begin{aligned} C(A) \;=\; & \frac{(1-\mu)}{n}\mathbf{1}_n^T\left((-\tilde{P}\circ(KA))\mathbf{1}_{\tilde{d}} + \ln(\mathbf{1}_n + \mathbf{1}_{\tilde{d}}e^{KA})\right) \\ & -\frac{\mu}{n}\left(((-\mathbf{1}_n^T P)\circ(\ln\frac{1}{n}\mathbf{1}_n^T P))\mathbf{1}_d - n\ln d\right) \\ & +\lambda\mathbf{1}_n^T((KA)\circ A)\mathbf{1}_{\tilde{d}}. \end{aligned} \tag{7}$$

III. Training the Encoder

Training an MME consists in minimizing $C(A)$ (7) with respect to the matrix of coefficients A. The objective function is continuous, differentiable, and it is composed of the sum of a concave part, i.e. $(1-\mu)H(Y|X,F)$ (bounded in $[0, \ln d]$), and two convex ones, i.e. $-\mu H(\hat{X},F)$ and $\lambda\Omega(F)$, so that it is bounded below and not convex.

We optimized the problem using a nonlinear Conjugate Gradient (CG) with line search performed by the secant method and Polak-Ribiere update [18], that avoids the computation of second derivatives, and it is popularly used for its speed and easy implementation. In Section III-A we will describe how to efficiently simplify the gradient expression to avoid a costly matrix-by-matrix product. Alternatively, MEEs can be trained using convex optimization tools by means of a concave-convex procedure (CCCP) [7]. CCCP allows $C(A)$ to be minimized by iteratively solving a set of convex problems. Given A^t at time $t = 0$, we indicate with $u(A)$ and $-v(A)$ the convex and concave portions of $C(A)$, so that $C(A) = u(A) - v(A)$, and

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

is a convex function obtained by linearizing the concave part of $C(A)$ around A^t. Finally $A^{t+1} = \arg\min C_{convex}(A, A^t)$.

Now we discuss two important issues of both the optimization schemes. Firstly, different starting points of the optimization, A^0, can lead to different suboptimal results. A a good choice A^0 can reduce the optimization times and improve the quality of the solution. Second, when evaluating different solutions, smaller values of the objective function may not always correspond to a better data partitioning, and a criterion to select the most promising one must be devised.

Beyond a random initialization of A^0, a we suggest to run a few K-Means iterations to get an initial guess of the data partitioning. We denote with $Y_{km} \in \{-1, 1\}^{n,\tilde{d}}$ 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 -1s). If we fit the targets of Y_{km} with the functions $f_1, \ldots, f_{\tilde{d}}$, then $y_i = \arg\max_j p_j(\boldsymbol{x}_i, F)$ will match the K-Means prediction. For instance, we can solve a regularized least-square problem, leading to $A^0 = (K + \lambda I)^{-1}Y_{km}$ [19]. We notice that each permutation of the columns of A^0 will lead to equivalent solutions (the permutation will be applied to the output codes). Selecting $A^0 = 0\cdot\mathbf{1}_n\mathbf{1}_{\tilde{d}}^T$ leads to $f_1, \ldots, f_{\tilde{d}}$ that are equivalent and constant, and each input point is projected into the same output code $[1/d, \ldots, 1/d]$. It is a degenerate solution of the problem, it must be avoided.

Moreover, without performing any additional machinery we have the use of the conditional entropy $H(Y|X,F)$ associated to each solution, that is one of the three terms that compose (5). Hence, once μ and λ have been fixed, we can optimize the algorithm from different starting points and select the solution that corresponds to the smallest $H(Y|X,F)$.

A. Gradient Computation

In this section we describe how to compute the gradient matrix $\nabla C(A) \in I\!R^{n,\tilde{d}}$ and how to dramatically simplify its computation avoid a product by K. $\nabla C(A)$ is the sum of three elements,

$$\nabla C(A) = (1-\mu)\nabla H(Y|X,F) - \mu\nabla H(\hat{X},F) + \lambda\nabla\Omega(F). \tag{8}$$

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

$$\begin{aligned} \frac{\partial H(Y|X,F)}{\partial\alpha_{zh}} \;=\; & -\sum_{i=1}^{n}\sum_{j=1}^{\tilde{d}}\left(\frac{\partial p_{ij}}{\partial\alpha_{zh}}\cdot\boldsymbol{k}_i^T\boldsymbol{\alpha}_j + p_{ij}\cdot\frac{\partial\boldsymbol{k}_i^T\boldsymbol{\alpha}_j}{\partial\alpha_{zh}}\right) \\ & +\sum_{i=1}^{n}p_{ih}\cdot\frac{\partial\boldsymbol{k}_i^T\boldsymbol{\alpha}_h}{\partial\alpha_{zh}}. \end{aligned} \tag{9}$$

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},

$$\frac{\partial p_{ij}}{\partial\alpha_{zh}} = (p_{ij}\delta_{hj} - p_{ij}p_{ih})\cdot k_{iz}, \quad \frac{\partial\boldsymbol{k}_i^T\boldsymbol{\alpha}_j}{\partial\alpha_{zh}} = \delta_{hj}k_{iz}. \tag{10}$$

Plugging them in (9), we get

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

and, switching to matrix notation,[2]

$$\nabla H(Y|X,F) = -K\left(\tilde{P}\circ\left(KA - (\tilde{P}\circ(KA))\mathbf{1}_{\tilde{d}}\mathbf{1}_{\tilde{d}}^T\right)\right). \tag{11}$$

We follow the same procedure to compute the gradient of $H(\hat{X},F)$. In detail,

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

[2]Due to space constraints, we do not report all the derivations.

After we introduce $s_j = \ln\left(\frac{1}{n}\sum_{r=1}^{n} p_{rj}\right) + 1$ and we use the results of (10), we get

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

Finally, passing to matrix notation,[2]

$$\nabla H(\hat{X}, F) = -K\left(\tilde{P} \circ (\mathbf{1}_n(\ln\frac{1}{n}\mathbf{1}_n^T P)^T - P(\ln\frac{1}{n}\mathbf{1}_n^T P)\mathbf{1}_{\tilde{d}}^T)\right). \quad (12)$$

Now we can plug (11), (12) and $\nabla\Omega(F) = 2KA$ into (8), and complete the derivation of the gradient,

$$\begin{aligned}\nabla C(A) = &\frac{\mu - 1}{n} K\left(\tilde{P} \circ \left(KA - (\tilde{P} \circ (KA))\mathbf{1}_{\tilde{d}}\mathbf{1}_{\tilde{d}}^T\right)\right) \\ &+\frac{\mu}{n} K\left(\tilde{P} \circ (\mathbf{1}_n(\ln\frac{1}{n}\mathbf{1}_n^T P)^T - P(\ln\frac{1}{n}\mathbf{1}_n^T P)\mathbf{1}_{\tilde{d}}^T)\right) \\ &+2\lambda KA. \end{aligned} \quad (13)$$

Each stationary point of $C(A)$ satisfies $\nabla C(A) = 0 \cdot \mathbf{1}_n\mathbf{1}_{\tilde{d}}^T$. This set of equations can be simplified by solving their preconditioned form $\nabla' C(A) = K^{-1}\nabla C(A) = 0 \cdot \mathbf{1}_n\mathbf{1}_{\tilde{d}}^T$, where $\nabla' C(A)$ can be computed without performing any matrix inversions, since K is a factor of $\nabla C(A)$.[3] This preconditioned form of the gradient avoids the need of a costly matrix-by-matrix product, and we can efficiently apply the CG algorithm. More details are discussed in Section III-B.

Finally, the expansion of $f_j(\boldsymbol{x})$ can be augmented with a not-regularized bias b_j, i.e. $f_j(\boldsymbol{x}_i) = \boldsymbol{k}_i^T\boldsymbol{\alpha}_j + b_j$. The vector $\boldsymbol{b}$ collects b_j, $j = 1, \ldots, \tilde{d}$. Following the same strategy that we used for A, we can get that the gradient of the objective function with respect to $\boldsymbol{b}$ is simply the vector that collect the sum (column-wise) of the preconditioned gradient with respect to A, $\nabla'_A C(A)$, computed with $\lambda = 0$,

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

B. Complexity Analysis

When MEEs are trained using CG, each iteration requires to store the matrix A, $\nabla' C(A)$, and the current descent direction, that belong to $\mathbb{R}^{n,\tilde{d}}$ (we discard the bias term, for simplicity).

If the matrices $KA \in \mathbb{R}^{n,\tilde{d}}$ and $P \in \mathbb{R}^{n,d}$ are stored when evaluating the objective function (7), then the computation of the gradient $\nabla'(A)$ can be significantly accelerated. Optionally, also the Gram matrix $K \in \mathbb{R}^{n,n}$ can precomputed. Hence, in the worst case, the space complexity is bounded $O(n^2)$ (since $d \leq n$).[4]

CPM3C [15], 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 [16], 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 [15]). Moreover it depends on the degree of fulfillment of the problem constraints, that are an exponential number (i.e. $(d + 1)^n$ [15]). 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 MEEs.

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 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.

[3]We are assuming that in this expression K is non singular, otherwise we can suppose that a small ridge is added to fix it.

[4]This analysis considers a generic kernel function (linear, nonlinear), without any distinctions. However, in the linear case the complexity can be reduced, and we leave to future work this specific study.

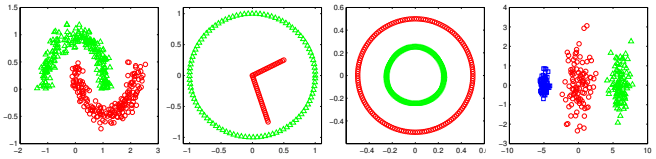

Fig. 1. Examples of clustering results produced by MEEs (different color/markers represent data from different classes). The rightmost data set is composed of 3 Gaussian distributions in $\mathbb{R}^{10}$, linearly separable only on the first dimension (the first 2 ones are plotted).

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 MEEs to correctly separate clusters in non-dense regions. We tested MEEs in several UCI data sets, in the digit recognition benchmark USPS (selecting the test portion, USPST) and in the object recognition data set Coil20. We compared MEEs with the related clustering algorithms, using publicly accessible code and, when possible, the implementation of the respective authors: K-Means (random initialization), Spectral Clustering (normalized cut, NC) [2], GMMC [4], ISVR [5], CPMMC [6] and CPM3C [15], (that we compactly refer to as CPM(M/3)C), LGMMC [8]. MEEs have 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 [14], used in most of the experimental comparisons involving MMC. In detail, 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) is the number of pairs with the same label that are assigned to the same cluster plus the number of the pairs with different labels that are assigned to

TABLE II
ADJUSTED RAND INDEX (ARI) OF MEES, COMPARED WITH THE BEST ARI (ARI*) AMONG THE OTHER ALGORITHMS LISTED IN TABLE I (ALGORITHM* IS THE TECHNIQUES THAT SCORED SUCH RESULT).

Data	MEE ARI	ARI*	Algorithm*
Ionosphere	**0.58 (0.00)**	0.30 (0.00)	ISVR
Heart	0.31 (0.07)	**0.33**	NC
Musk1	0.07 (0.03)	**0.09 (0.13)**	ISVR
German	**0.06 (0.00)**	0.02	LGMMC
Breast	0.74 (0.00)	**0.80**	NC
Diabetes	0.08 (0.05)	**0.10**	GMMC
Echocg	**0.17 (0.08)**	0.12 (0.00)	KM
Uspst1vs7	**0.99 (0.00)**	0.96	NC
Uspst3vs8	**0.88 (0.00)**	0.84	NC
Pcmac	0.78 (0.02)	**0.79 (0.02)**	ISVR
Iris	**0.90 (0.00)**	0.79	NC
Balance	**0.20 (0.05)**	0.18 (0.14)	CPM3C
Wine	**0.42 (0.00)**	0.38	NC
Boston	0.13 (0.00)	**0.14 (0.00)**	CPM3C
Coil20	0.68 (0.03)	**0.76**	NC
Uspst	**0.65 (0.01)**	0.58	NC

different clusters, 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 [20].

Parameter selection has been performed by an exhaustive search of the optimal values, selecting the best configurations. 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 $\{0.25\sigma_0, 0.33\sigma_0, 0.5\sigma_0, \sigma_0, 2\sigma_0, 3\sigma_0, 4\sigma_0\}$, being σ_0 the average pairwise distances of X. λ has been selected from $\{10^{-5}, 10^{-4}, \ldots, 10^{2}\}$ ($C = 1/\lambda$ in GMMC, ISVR, CPM(M/3)C), whereas μ in $\{0.5, 0.7\}$. MEE and ISVR were initialized using the K-Means algorithm, whereas the other non-convex algorithms lacks of a specific strategy, and their starting point was randomly generated. NC was ran using different number of neighbors to compute the Laplacian, $\{6, 12, 24, 48\}$. Other algorithm-specific parameters are: $\epsilon = 0.01$, $\ell \in \{0.03n, 0.15n\}$ in the case of ISVR; $\beta \in \{0.03n, 0.3n\}$ for LGMMC; $\epsilon = 0.01$, $\ell \in \{0, 1, 10, 20\}$ in the case of CPM(M/3)C.

TABLE I 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 MEEs overcomes state of the art algorithms while in others, at least, it exhibits accuracies that are almost comparable. The same hold when measuring the ARI (TABLE II). 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-B, whereas we efficiently completed the experiments with MEEs.

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

TABLE III
ACCURACY OF MEES 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+	KMeans	KMeans+
Ionos.	68.03 (5.96)	74.93 (8.50)	87.92 (0.15)	**88.03 (0.00)**
Breast	88.01 (9.30)	90.93 (3.81)	**92.97 (0.00)**	**92.97 (0.00)**
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)

TABLE IV
AVERAGE TRAINING TIMES (SECONDS) OF MEES AND ISVR, AND THE NUMBER OF CONJUGATE GRADIENT (CG) ITERATIONS.

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)

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 III). In the case of a random initialization, this approach resulted very efficient to avoid inaccurate solutions. Even if a detailed time comparison goes beyond the scope of this paper, 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 IV we can appreciate that MEE resulted faster than ISVR. We also report the number of CG iterations, that we found to be always significantly smaller than the number of data points. Finally, Fig. 2 illustrates the sensitivity of MEEs 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 (5), so as to get accurate groupings and to avoid trivial solutions.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce Minimum Entropy Encoder (MEEs) clustering algorithms, which derive from bridging

TABLE I

CLUSTERING ACCURACIES (%) OF MEES 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). CPM3C HAD TOO LARGE MEMORY REQUIREMENTS ON THE LAST TWO BENCHMARKS. MEES COMPARES FAVORABLY WITH MOST OF THE COMPETITORS.

Data	Size	Dim	Classes	KM	NC	GMMC	ISVR	CPM(M/3)C	LGMMC	MEE
Ionosphere	351	34	2	71.23 (0.00)	70.66	77.49	77.32 (0.15)	71.23 (0.00)	74.07	**88.03 (0.00)**
Heart	270	13	2	76.07 (7.59)	**78.89**	77.04	74.00 (9.53)	77.33 (6.41)	72.59	77.70 (3.80)
Musk1	476	166	2	56.51 (0.00)	56.51	56.51	62.14 (9.27)	56.85 (0.74)	62.82	**63.32 (2.69)**
German	1000	24	2	70.00 (0.00)	70.00	70.00	70.00 (0.00)	70.00 (0.00)	70.00	**70.70 (0.00)**
Breast	569	30	2	92.79 (0.00)	**94.73**	85.06	87.35 (0.37)	92.97 (0.14)	81.37	92.97 (0.00)
Diabetes	768	8	2	66.02 (0.00)	65.10	66.15	65.10 (0.00)	65.46 (1.11)	**68.75**	66.21 (1.85)
Echocg	132	8	2	81.82 (0.00)	81.82	81.82	81.82 (0.00)	81.82 (0.00)	81.82	**82.35 (0.72)**
Uspst1vs7	446	256	2	98.65 (0.00)	99.10	98.65	90.27 (0.22)	93.77 (12.33)	97.98	**99.78 (0.00)**
Uspst3vs8	338	256	2	89.76 (1.19)	95.86	88.76	92.25 (0.57)	88.82 (2.02)	87.28	**97.04 (0.00)**
Pcmac	1946	7511	2	81.93 (15.23)	90.08	-	**94.36 (0.57)**	86.84 (13.14)	56.17	94.02 (0.52)
Iris	150	4	3	82.53 (10.95)	92.00	-	-	86.32 (4.45)	-	**96.67 (0.00)**
Balance	625	4	3	65.71 (3.72)	62.72	-	-	66.45 (7.57)	-	**68.54 (2.65)**
Wine	178	13	3	69.66 (1.18)	72.47	-	-	70.42 (0.99)	-	**73.03 (0.00)**
Boston	506	13	3	65.02 (0.00)	65.02	-	-	65.42 (0.00)	-	**65.61 (0.21)**
Coil20	1440	1024	20	66.57 (2.52)	**84.79**	-	-	-	-	73.50 (3.74)
Uspst	2007	256	10	72.50 (1.51)	77.43	-	-	-	-	**77.57 (0.42)**

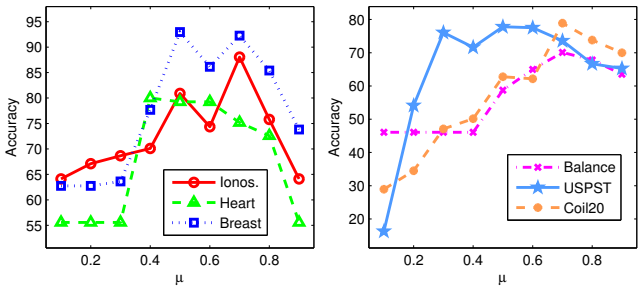

Fig. 2. Accuracy of MEEs in function of the parameter μ, in the case of 2-class (left) and multi class data (right).

Information-Theoretic principles and kernel methods. MEEs are shown to be the optimal solution of a proper regularization problem, which is based on an appropriate balancing and smoothing of the developed features f_j. This makes it possible to devise learning algorithms that re-use the kernel mathematical apparatus. Unlike most of the Maximum Margin Clustering algorithms, MEEs naturally handle multi-class data sets. Two efficient solutions of the MEE problem are presented, which are based on non-linear conjugate gradient and concave-convex procedures. In many tasks, MEEs overcome state-of-the art techniques on a variety of benchmarks and, moreover, they exhibit remarkable performance also for tasks in which they do not reach the best score.

REFERENCES

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

[2] 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.

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

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

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems*, vol. 17, 2004, pp. 529–536.

[14] 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.

[15] 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.

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

[17] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[18] J. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Tech. Rep., 1994.

[19] R. Rifkin and R. Lippert, "Notes on regularized least squares," MIT, Cambridge, MA, Tech. Rep. CBCL Paper, Tech. Rep., 2007.

[20] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 12, pp. 2837–2854, 2010.