Competitive Learning with Pairwise Constraints

Thiago F. Covões*†, Eduardo R. Hruschka*†, Joydeep Ghosh*
*University of Texas (UT) at Austin, USA
†University of São Paulo (USP) at São Carlos, Brazil
{tcovoes,erh}@icmc.usp.br;ghosh@ece.utexas.edu

***Abstract*—Constrained clustering has been an active research topic in the last decade. Most studies focus on batch-mode algorithms. This paper introduces two algorithms for on-line constrained learning, named O-LCVQE (On-line Linear Constrained Vector Quantization Error) and C-RPCL (Constrained Rival Penalized Competitive Learning). The former is a variant of the LCVQE algorithm for on-line settings, while the latter is an adaptation of the (on-line) RPCL algorithm to deal with constrained clustering. The accuracy results — in terms of the normalized mutual information (NMI) — from experiments with nine datasets show that the partitions induced by O-LCVQE are competitive with those found by the (batch-mode) LCVQE. Compared to this formidable baseline algorithm, it is surprising that C-RPCL can provide better partitions (in terms of the NMI) for most of the datasets. Also, experiments on a large dataset show that on-line algorithms for constrained clustering can significantly reduce the computational time.**

I. Introduction

Competitive Learning algorithms are characterized by competitions among k neurons [1]. At each step, an input (object) $\mathbf{x}_i$ is presented, and the neuron that wins the competition learns the input, i.e., that neuron is adapted towards $\mathbf{x}_i$ with a pre-specified learning rate. This approach is known as Winner-Take-All (WTA) learning [2], which has been extensively studied [1,3]. Competitive learning can also be seen as performing clustering in the input space [2,4,5]. In particular, regarding the neurons as cluster prototypes, and making use of the squared Euclidean distance as the competition score, the WTA approach can be seen as an on-line k-means algorithm. In a broader view, competitive learning algorithms belong to the class of stochastic gradient descent algorithms [3].

The use of supervision in the competitive learning framework was considered in the LVQ algorithm [6], where it is assumed that labels are known for the training data. Such labels are used to guide the refinement of the decision boundaries. However, the acquisition of labels is often costly, whereas unlabeled data are abundant in diverse applications. Thus, algorithms capable of using only partial supervision — e.g., in the form of a subset of labeled objects — are needed.

In the last decade, considerable attention has been given to constrained clustering algorithms, which incorporate known information about the desired data partitions into the clustering process [7]. Among the most common types of constraints are those about pairs of objects — specifically the *Must-Link* (ML) and *Cannot-Link* (CL) constraints [7,8]. Considering a set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ of N objects, each one represented by a vector $\mathbf{x}_i \in \mathbb{R}^M$, a ML constraint $c_{=}(i,j)$ indicates that the objects $\mathbf{x}_i$ and $\mathbf{x}_j$ should lie in the same cluster, whereas a CL constraint $c_{\neq}(i,j)$ indicates that $\mathbf{x}_i$ and $\mathbf{x}_j$ should lie in different clusters. One characteristic of ML and CL constraints is that they represent less information than labels. This can be noted by the fact that is possible to deduce constraints from labels, but the inverse does not hold [7].

Several algorithms have been developed for constrained clustering [7]–[11]. Among them, k-means-based algorithms are widely used in practice and thus are of interest to a large audience. In a recent study [11], it has been shown that the clustering results of the Linear Constrained Vector Quantization Error (LCVQE) algorithm [8] are competitive with those achieved by the Constrained Vector Quantization Error (CVQE) algorithm [10], while violating less constraints and being more computationally efficient. Furthermore, in most cases, LCVQE provided better accuracy compared to MPCK-Means [9], which is capable of learning distance metrics. Motivated by these results, as a starting point we decided to extend the (LCVQE) algorithm to deal with on-line settings. Since the proposed extension of LCVQE can be seen as a competitive learning algorithm, extensions of alternative competitive learning algorithms to deal with pairwise constraints can also be potentially helpful.

This paper introduces two on-line constrained clustering algorithms, which are named O-LCVQE (On-line LCVQE) and C-RPCL (Constrained Rival Penalized Competitive Learning). Both algorithms have been designed to incorporate constraint handling without losing the desirable simplicity provided by the competitive learning approach. It is worth mentioning that, despite the increasing interest in constrained clustering over the last decade, there is a lack of studies for on-line settings. The development of on-line algorithms can be helpful to the analysis of large scale data, as well as to the use of constraints in distributed and asynchronous settings [3]. Another attractive characteristic of on-line algorithms is that they are particularly suited to deal with scenarios where the dataset ($\mathcal{X}$) suffers changes (addition/removal of objects), as well as with scenarios where the set of constraints is dynamically changing.

We performed experiments on nine datasets for which the class labels are known. Thus, pairwise constraints can be obtained from some labeled objects, as commonly done in constrained clustering studies [7]. However, there is a caveat on this setting. Since (multimodal) classes can potentially be represented by multiple clusters, the use of ML constraints derived from class labels can be hazardous to the clustering

process. More specifically, if two objects belong to the same class but to different clusters, a ML constraint involving them will guide the clustering algorithm to merge these two clusters. Clearly, this can be detrimental to the clustering process. Therefore, we consider CL constraints only, because these are correct independently of the validity of the so-called *one-cluster-per-class assumption*. However, for cases where ML constraints can be assured to be valid, it is straightforward to incorporate them into the proposed algorithms.

The remainder of this paper is organized as follows. The next section reviews the LCVQE algorithm [8], which serves as the basis to describe the proposed algorithms — presented in Section III and empirically compared in Section IV. Finally, Section V concludes the paper.

Notation. A hard partition of the data is a collection $\mathcal{P} = \{C_i\}_{i=1}^{k}$ of k disjoint clusters such that $\bigcup C_i = \mathcal{X}$ and $C_i \cap C_j = \emptyset, \forall i \neq j$, and $|C_i| \neq 0, \forall i$, where $|C_i|$ denotes the number of objects in cluster C_i. Each cluster C_i is represented by a prototype $\boldsymbol{\mu}_i$. The distance between an object $\mathbf{x}_i$ and a prototype $\boldsymbol{\mu}_j$ is calculated by using the squared Euclidean distance, i.e., $\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 = (\mathbf{x}_i - \boldsymbol{\mu}_j)^T(\mathbf{x}_i - \boldsymbol{\mu}_j)$, where T denotes the transposed matrix. The set of ML and CL constraints are denoted as $\mathcal{M}$ and $\mathcal{C}$, respectively. Using $o_{\mathcal{M}}(l)$ and $o'_{\mathcal{M}}(l)$ for denoting the functions that return the first and second objects of the l^{th} ML constraint, it is possible to define the functions $g_{\mathcal{M}}(l)$ and $g'_{\mathcal{M}}(l)$ that return, respectively, the indices of the clusters that the first and second objects of the l^{th} ML constraint belong to, i.e., $g_{\mathcal{M}}(l) = \{j | o_{\mathcal{M}}(l) \in C_j\}$ and $g'_{\mathcal{M}}(l) = \{t | o'_{\mathcal{M}}(l) \in C_t\}$. Similarly, the functions $o_{\mathcal{C}}(l), o'_{\mathcal{C}}(l), g_{\mathcal{C}}(l)$, and $g'_{\mathcal{C}}(l)$ can be defined for CL constraints. The set of ML constraints being violated is defined as $\mathcal{V}_{\mathcal{M}} = \{i | \mathcal{M}_i \in \mathcal{M}, g_{\mathcal{M}}(i) \neq g'_{\mathcal{M}}(i)\}$, and similarly, the set of CL constraints being violated is defined as $\mathcal{V}_{\mathcal{C}} = \{i | \mathcal{C}_i \in \mathcal{C}, g_{\mathcal{C}}(i) = g'_{\mathcal{C}}(i)\}$. Finally, $\mathbb{1}_{[\text{Condition}]}$ is an indicator function, that is equal to one when the condition is satisfied and 0 otherwise.

II. Review of the Linear Constrained Vector Quantization Error (LCVQE) Algorithm [8]

The LCVQE algorithm is an improved version of the Constrained Vector Quantization Error algorithm [10]. LCVQE uses as its objective function the usual vector quantization error augmented with costs associated with constraint violations. Even though we do not use ML constraints in our experimental setting, for completeness we describe the full LCVQE algorithm. The cost of violating a ML constraint is based on the distances between objects and prototypes. These distances are computed by considering the object of a cluster and the prototype of another (neighbor) cluster. For a CL constraint, the object that is the farthest to the prototype is determined. Then, the distance between this object and its nearest neighbor prototype (second-closest cluster) is used as the violation cost. Thus, total cost = (distortion cost) + (ML violation cost) + (CL violation cost). Mathematicaly, we have:

$$\begin{aligned} J &= \sum_{j=1}^{k}\Big[\Big(\frac{1}{2}\sum_{\mathbf{x}_i \in C_j}\|\boldsymbol{\mu}_j - \mathbf{x}_i\|^2\Big) \\ &+ \Big(\frac{1}{2}\sum_{l \in \mathcal{V}_{\mathcal{M}}, g_{\mathcal{M}}(l)=j}\frac{1}{2}\|\boldsymbol{\mu}_j - o'_{\mathcal{M}}(l)\|^2 \\ &+ \frac{1}{2}\sum_{l \in \mathcal{V}_{\mathcal{M}}, g'_{\mathcal{M}}(l)=j}\frac{1}{2}\|\boldsymbol{\mu}_j - o_{\mathcal{M}}(l)\|^2\Big) \\ &+ \Big(\frac{1}{2}\sum_{l \in \mathcal{V}_{\mathcal{C}}, V(l)=j}\|\boldsymbol{\mu}_j - R_{g_{\mathcal{C}}(l)}(l)\|^2\Big)\Big] \end{aligned} \tag{1}$$

The auxiliary functions, $R_{g_{\mathcal{C}}(l)}(l)$ and $V(l)$, are defined in Equations (2) and (3), respectively. Intuitively, the former — $R_{g_{\mathcal{C}}(l)}(l)$ — returns the object of the l^{th} CL constraint farthest to the prototype $\boldsymbol{\mu}_{g_{\mathcal{C}}(l)}$, while the latter — $V(l)$ — returns the index of the nearest neighbor prototype to the object $R_{g_{\mathcal{C}}(l)}(l)$.

$$R_{g_{\mathcal{C}}(l)}(l) = \begin{cases} o_{\mathcal{C}}(l) & \text{if } \|o_{\mathcal{C}}(l) - \boldsymbol{\mu}_{g_{\mathcal{C}}(l)}\|^2 > \|o'_{\mathcal{C}}(l) - \boldsymbol{\mu}_{g_{\mathcal{C}}(l)}\|^2 \\ o'_{\mathcal{C}}(l) & \text{otherwise,} \end{cases} \tag{2}$$

$$V(l) = \underset{m \in \{1,\ldots,k\} - \{g_{\mathcal{C}}(l)\}}{\arg\min} \|R_{g_{\mathcal{C}}(l)}(l) - \boldsymbol{\mu}_m\|^2. \tag{3}$$

For the assignment of which cluster each object belongs to, initially every object is assigned to the closest cluster. Then, for each ML constraint being violated, only three assignment possibilities are examined: (i) maintain the violation; (ii) assign the two objects to the cluster whose prototype is the nearest to the first object ($o_{\mathcal{M}}(l)$); (iii) assign the two objects to the cluster whose prototype is the nearest to the second object ($o'_{\mathcal{M}}(l)$). For each CL constraint being violated, only two cases are checked: (i) maintain the violation; (ii) keep the object that is the closest to the cluster prototype as it is, and assign the farthest object ($R_{g_{\mathcal{C}}(l)}(l)$) to the cluster with second-closest prototype ($V(l)$). Then, the prototypes are updated as:

$$\boldsymbol{\mu}_j = \frac{\mathbf{y}_j}{z_j}, \tag{4}$$

$$\begin{aligned} \mathbf{y}_j &= \sum_{\mathbf{x}_i \in C_j}\mathbf{x}_i + \frac{1}{2}\sum_{l \in \mathcal{V}_{\mathcal{M}}, g_{\mathcal{M}}(l)=j} o'_{\mathcal{M}}(l) \\ &+ \frac{1}{2}\sum_{l \in \mathcal{V}_{\mathcal{M}}, g'_{\mathcal{M}}(l)=j} o_{\mathcal{M}}(l) \\ &+ \sum_{l \in \mathcal{V}_{\mathcal{C}}, V(l)=j} R_{g_{\mathcal{C}}(l)}(l) \end{aligned} \tag{5}$$

$$\begin{aligned} z_j &= |C_j| + \frac{1}{2}\sum_{l \in \mathcal{V}_{\mathcal{M}}}\mathbb{1}_{[g_{\mathcal{M}}(l)=j]} \\ &+ \frac{1}{2}\sum_{l \in \mathcal{V}_{\mathcal{M}}}\mathbb{1}_{[g'_{\mathcal{M}}(l)=j]} + \sum_{l \in \mathcal{V}_{\mathcal{C}}}\mathbb{1}_{[V(l)=j]} \end{aligned} \tag{6}$$

The update rule can be interpreted as follows. Let l be the index of a ML constraint that is being violated, i.e., $o_{\mathcal{M}}(l) \in$

C_j and $o'_{\mathcal{M}}(l) \in C_n$ with $j \neq n$. Then, the prototype $\boldsymbol{\mu}_j$ is moved towards the object $o'_{\mathcal{M}}(l)$ and the prototype $\boldsymbol{\mu}_n$ is moved towards the object $o_{\mathcal{M}}(l)$. Now consider the case of a CL constraint being violated, i.e., $o_{\mathcal{C}}(l) \in C_j$ and $o'_{\mathcal{C}}(l) \in C_j$. Consider also that $\|\boldsymbol{\mu}_j - o'_{\mathcal{C}}(l)\|^2 > \|\boldsymbol{\mu}_j - o_{\mathcal{C}}(l)\|^2$ and that $\boldsymbol{\mu}_n$ is the second-closest prototype of $o'_{\mathcal{C}}(l)$. Then, $\boldsymbol{\mu}_n$ is moved towards $o'_{\mathcal{C}}(l)$. The whole process (assignment and prototype update) is repeated until a convergence criterion is satisfied.

III. Constrained Competitive Learning

As stated in Section I, competitive learning can be seen as an on-line approach to clustering because the prototype updates are incrementally made after each input is observed. This approach is different from the one performed by batch mode algorithms, which take one full pass through the data before an update is made.

The simplest competitive learning algorithm is based on the Winner-Take-All (WTA) approach, in which the closest prototype (neuron) to the input moves towards the input at a learning rate α, i.e., $\Delta\boldsymbol{\mu}_j = r_j\alpha(\mathbf{x}_i - \boldsymbol{\mu}_j), i \in \{1,\ldots,N\}, j \in \{1,\ldots,k\}$, where r_j is one if $\boldsymbol{\mu}_j$ is the closest prototype to $\mathbf{x}_i$, and zero otherwise. Despite its simplicity, WTA presents the *neuron underutilization* problem [1], which arises when some neurons are not properly initialized and thereby making them unable to win even once. For instance, if all neurons are initialized in a region far from the data, it is likely that the winner for the first input will also win the remaining competitions.

Several competitive learning algorithms have been developed to address the neuron underutilization problem — see [1] for a thorough discussion. A simple approach is to consider the winning frequency for each neuron, i.e., weighting the distances between prototypes and objects by the neuron winning frequency. This is known as frequency sensitive competitive learning [12]. An extension of this approach, which can also be seen as an extension of the LVQ2 algorithm [6], is the Rival Penalized Competitive Learning (RPCL) [13]. In this algorithm, the second nearest prototype to the input, called rival, is penalized by moving it away from that object. This penalization is proportional to a different learning rate (β) — usually $\beta \ll \alpha$. For simplicity, we refer to β as the *unlearning rate*. Now we are in position to describe the proposed constrained competitive learning algorithms, namely: O-LCVQE and C-RPCL.

A. On-line LCVQE

O-LCVQE can also be seen as a WTA approach that deals with CL constraints. We do not adopt ML constraints in our framework due to the reasons discussed in Section I. Therefore, we omit the procedure that handles them[1]. We note that the procedure used to handle CL constraints is similar in spirit to the LCVQE's procedure, i.e., the winner prototype is defined as in LCVQE. Specifically, if the input is involved in a CL constraint, and the same prototype ($\boldsymbol{\mu}_j$) is the nearest one for the two objects in question, then two cases are assessed: (i) use the same prototype to represent both objects; (ii) adjust the second nearest prototype ($\boldsymbol{\mu}_n$) to the object of the constraint that is farthest from $\boldsymbol{\mu}_j$. Each case has an associated cost, which is derived from the objective function of LCVQE — Eq. (1). After computing them, the case that incurs the smallest cost is chosen. For the second case, if the farthest object is not the current input ($\mathbf{x}_i$), an additional step is performed so that $\boldsymbol{\mu}_j$ moves away from the farthest object and towards $\mathbf{x}_i$. The main steps of O-LCVQE are presented in Algorithm 1.

Note that this algorithm handles CL constraints in a greedy manner. In particular, for each constraint it assesses every possible assignment by considering only the costs imposed by that particular constraint — without taking into account the costs that may be imposed by other constraints related to the current input. This procedure is related to the competitive learning approach, as the constraints are always processed in an on-line fashion. Also, when no constraints are provided this algorithm reduces to the WTA approach (see Section III).

Algorithm 1 On-line LCVQE (O-LCVQE).

function O-LCVQE($\mathcal{X}, \{\boldsymbol{\mu}_i\}_{i=1}^k, \alpha, \beta, \mathcal{C}$)
  Randomly take an object $\mathbf{x}_i$ from $\mathcal{X}$
  $j \leftarrow \arg\min_{j\in\{1,\ldots,k\}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$
  $\mathcal{S}_{\mathcal{C}} \leftarrow \{l | o_{\mathcal{C}}(l) = i\} \cup \{l | o'_{\mathcal{C}}(l) = i\}$
  if $\mathcal{S}_{\mathcal{C}} = \emptyset$ **then**
    $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \alpha(\mathbf{x}_i - \boldsymbol{\mu}_j)$
  else
    for each $l \in \mathcal{S}_{\mathcal{C}}$ **do**
      $o \leftarrow \{o_{\mathcal{C}}(l) \cup o'_{\mathcal{C}}(l)\} \backslash \{i\}$
      $r \leftarrow \arg\min_{r\in\{1,\ldots,k\}} \|\mathbf{x}_o - \boldsymbol{\mu}_r\|^2$
      if $r \neq j$ **then**
        $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \alpha(\mathbf{x}_i - \boldsymbol{\mu}_j)$
      else
        $c \leftarrow \arg\min_{c\in\{i,o\}} \|\mathbf{x}_c - \boldsymbol{\mu}_j\|^2$
        $f \leftarrow \{i, o\} \backslash \{c\}$
        $n \leftarrow \arg\min_{n\in\{1,\ldots,k\}\backslash\{j\}} \|\mathbf{x}_f - \boldsymbol{\mu}_n\|^2$
        $cVio \leftarrow \frac{1}{2}(\|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 + \|\mathbf{x}_o - \boldsymbol{\mu}_j\|^2 + \|\boldsymbol{\mu}_n - \mathbf{x}_f\|^2)$
        $cNeighbor \leftarrow \frac{1}{2}(\|\mathbf{x}_c - \boldsymbol{\mu}_j\|^2 + \|\mathbf{x}_f - \boldsymbol{\mu}_n\|^2)$
        if $cVio < cNeighbor$ **then**
          $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \alpha(\mathbf{x}_i - \boldsymbol{\mu}_j)$
          $\boldsymbol{\mu}_n \leftarrow \boldsymbol{\mu}_n + \alpha(\mathbf{x}_f - \boldsymbol{\mu}_n)$
        else
          $\boldsymbol{\mu}_n \leftarrow \boldsymbol{\mu}_n + \alpha(\mathbf{x}_f - \boldsymbol{\mu}_n)$
          if f = o **then**
            $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j - \beta(\mathbf{x}_o - \boldsymbol{\mu}_j)$
          $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \alpha(\mathbf{x}_i - \boldsymbol{\mu}_j)$
  If there are still objects to process, go to Step 2

B. Constrained RPCL

The RPCL algorithm [13] can also be modified to deal with CL constraints. The intuition behind this modified algorithm, named C-RPCL, is that if a CL constraint gets violated by

[1]Nevertheless, it is straightforward to consider them if so desired.

assigning the object to a given prototype, then we search for the nearest rival that does not cause any constraint violations. This nearest rival becomes the winner, and the previous winner prototype is moved away from that object. For the degenerate case where no nearest rival can be found, the original RPCL procedure is used. The variable $\boldsymbol{\gamma}$ is introduced to avoid *neuron underutilization*, as mentioned in Section III. More specifically, $\boldsymbol{\gamma}$ holds the (normalized) winning count for each neuron. The distance between objects and prototypes are scaled by $\boldsymbol{\gamma}$, so that previously frequent winners have less chance to win the subsequent competition. The C-RPCL's main steps are described in Algorithm 2. Differently from O-LCVQE, the proposed C-RPCL takes into account all the available constraints in order to define the winner and rival neurons. By doing that, C-RPCL can avoid some unnecessary prototype updates that would be made by O-LCVQE. When no constraints are provided, C-RPCL reduces to RPCL, which is capable of estimating the number of clusters. The adoption of constraints does not make C-RPCL lose this feature.

Algorithm 2 Constrained RPCL (C-RPCL).

function C-RPCL($\mathcal{X}, \{\boldsymbol{\mu}_i\}_{i=1}^k, \alpha, \beta, \mathcal{C}$)
  $\mathbf{w} \leftarrow \mathbf{1}$ //$\mathbf{w} = [w_i]_{i=1}^k$
  $\boldsymbol{\gamma} \leftarrow \mathbf{1} \times k^{-1}$ //$\boldsymbol{\gamma} = [\gamma_j]_{j=1}^k$
  Randomly take an object $\mathbf{x}_i$ from $\mathcal{X}$
  $j \leftarrow \arg\min_{j \in \{1,\ldots,k\}} \gamma_j \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$
  $\mathcal{S}_\mathcal{C} \leftarrow \{l | o_\mathcal{C}(l) = i\} \cup \{l | o'_\mathcal{C}(l) = i\}$
  $\mathcal{F} \leftarrow \emptyset$
  for each $l \in \mathcal{S}_\mathcal{C}$ **do**
    $o \leftarrow \{o_\mathcal{C}(l) \cup o'_\mathcal{C}(l)\} \backslash \{i\}$
    $r \leftarrow \arg\min_{r \in \{1,\ldots,k\}} \gamma_r \|\mathbf{x}_o - \boldsymbol{\mu}_r\|^2$
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{r\}$
  if $\mathcal{S}_\mathcal{C} = \emptyset$ or $j \notin \mathcal{F}$ or $|\mathcal{F}| = k$ **then**
    $n \leftarrow \arg\min_{n \in \{1,\ldots,k\} \backslash \{j\}} \gamma_n \|\mathbf{x}_i - \boldsymbol{\mu}_n\|^2$
    $\boldsymbol{\mu}_n \leftarrow \boldsymbol{\mu}_n - \beta(\mathbf{x}_i - \boldsymbol{\mu}_n)$
    $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \alpha(\mathbf{x}_i - \boldsymbol{\mu}_j)$
    $w_j \leftarrow w_j + 1$
  else
    $n \leftarrow \arg\min_{n \in \{1,\ldots,k\} \backslash \mathcal{F}} \gamma_n \|\mathbf{x}_i - \boldsymbol{\mu}_n\|^2$
    $\boldsymbol{\mu}_n \leftarrow \boldsymbol{\mu}_n + \alpha(\mathbf{x}_i - \boldsymbol{\mu}_n)$
    $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j - \beta(\mathbf{x}_i - \boldsymbol{\mu}_j)$
    $w_n \leftarrow w_n + 1$
  for each $j \in \{1, \ldots, k\}$ **do**
    $\gamma_j \leftarrow w_j \times (\sum_{i=1}^k w_i)^{-1}$
  If there are still objects to process, go to Step 4

IV. Empirical Evaluation

In order to compare the algorithms O-LCVQE and C-RPCL, experiments were performed on eight benchmarks datasets. Most of them are available at the UCI Repository [14]. In addition, we have used the *9Gauss* dataset [15], which is formed by nine balanced clusters arranged according to Gaussian distributions that have some degree of overlapping, as well as the *Protein* dataset [9]. Following [9], for the *Letters* dataset only classes I, J, and L were considered and, for Pendigits, only classes 3, 8, and 9 were considered. According to [9] these classes represent difficult classification problems. The characteristics of the datasets are summarized on Table I.

As stated in Section I, we consider the scenario where constraints are derived from class labels. For generating the constraints, we consider cases where samples of labeled objects are available. In particular, we adopt a variety of different quantities of labeled objects — i.e., 5, 10, 15, and 20 randomly selected objects per class. From these samples, all possible CL constraints are deduced. For instance, for the Iris dataset, which has three classes, by sampling 5 objects per class a set of 75 constraints is obtained. The adopted methodology follows from the scenario where a domain expert provides labels for some objects. In this scenario, one should take as much information as possible from the labeled objects by considering all the induced constraints.

Each algorithm was run for 100 epochs (full passes through the data). Initial prototypes were sampled by a Gaussian distribution, with mean and covariance estimated from 20% of the data. Due to the sensitivity to initialization and processing order of the objects, each algorithm was run five times with different initial prototypes and processing orders. The whole process was repeated ten times to get better estimates of the statistics under interest. As done in [13], the learning and unlearning rates were set to 0.05 and 0.002, respectively.

The quality of the obtained partitions is assessed by means of *Normalized Mutual Information* (NMI) [16] by taking into account the reference partitions provided by the known classes/clusters. As a baseline[2], we considered the results obtained by the (batch mode) LCVQE algorithm [8]. Since batch mode algorithms have access to more information to perform prototypes updates than on-line algorithms, it is expected that their partitions serve as approximations to those achieved by the batch mode algorithms. Thus, we assess the relative performance of the proposed on-line algorithms by analyzing the differences between the NMI values computed for the partitions obtained by the on-line and batch algorithms.

The average differences between the NMI values obtained by the on-line algorithms and those obtained by LCVQE are reported in Table I, where a positive number means that the corresponding on-line algorithm obtained better results than LCVQE. It can be seen that C-RPCL obtained the best results, with NMI values equal to or greater than those obtained by LCVQE in more than 67% of the cases (27 out of 40). For O-LCVQE, the results were less favorable — specifically, the results obtained were equal or better than LCVQE in only 47% of the cases. Comparing the on-line algorithms, C-RPCL obtained better results than O-LCVQE in 75% of the cases.

The results for Pendigits are particularly appealing. In this dataset, C-RPCL presented the largest observed positive difference (0.229 on average). In order to better understand this result, the average NMI values obtained by each algorithm are

[2]Note that this is a very competitive baseline as LCVQE typically outperforms alternatives such as MPCK-Means [9] and CVQE [10] — see [11] (due to space limitations we do not reproduce such results).

Table I: Differences between Normalized Mutual Information values obtained by on-line algorithms and LCVQE (greater is better) — #LO denotes the number of labeled objects used to generate constraints.

Algorithm / # LO / Dataset				RPCL	C-RPCL					WTA	O-LCVQE				
				0	5	10	15	20	Average	0	5	10	15	20	Average
Name	N	M	k												
9Gauss	900	2	9	-.010	.005	-.015	-.005	-.007	-.006	-.017	-.032	-.101	-.100	-.102	-.071
Ionosphere	351	34	2	-.010	.004	-.004	.000	.017	.001	-.010	-.007	-.028	-.077	-.057	-.036
Iris	150	4	3	-.007	-.015	.001	-.037	-.019	-.015	.000	.013	.010	-.027	-.024	-.006
Wine	178	13	3	.003	.008	.030	.035	.038	.023	.000	.012	.020	.025	.028	.017
Breast Cancer	683	9	2	.000	.016	-.011	.007	.027	.008	-.018	.016	.013	-.023	-.013	-.005
Pendigits	3165	16	3	.204	.184	.227	.280	.250	.229	.052	.031	.173	.128	.111	.099
Letters	2263	16	3	.013	.010	.000	.008	.020	.010	-.006	.000	.010	.024	.052	.016
Pima	768	8	2	.001	.000	-.011	-.005	.001	-.003	-.003	-.004	-.019	-.027	-.019	-.014

Table II: Averages and standard deviations for the NMI values for different amounts of labeled objects (#LO) — Pendigits.

# LO	C-RPCL	O-LCVQE	LCVQE
0	0.69 (0.00)	0.53 (0.08)	0.48 (0.00)
5	0.68 (0.00)	0.52 (0.04)	0.49 (0.05)
10	0.71 (0.02)	0.66 (0.05)	0.49 (0.06)
15	0.76 (0.00)	0.61 (0.07)	0.48 (0.06)
20	0.77 (0.00)	0.63 (0.13)	0.52 (0.09)

presented in Table II, where there are two salient aspects to be observed. Firstly, the small variance of the results obtained by C-RPCL indicates some level of robustness with respect to the initialization of prototypes. Secondly, for all quantities of labeled objects the partitions obtained by C-RPCL presented high NMI values (≈ 0.72, on average). These results suggest that C-RPCL is not only obtaining better results than the other algorithms, but also it has found very good data partitions.

Table III presents the average differences between the number of constraints violated by on-line algorithms compared to LCVQE. In this case, small numbers are better, indicating that the on-line version violated less constraints than LCVQE. From these results, it is possible to see that, as expected, LCVQE, which can observe all the data at the same time, violated less constraints than the on-line algorithms in 75% of the cases (24 out of 32). However, some results deserve further attention. In particular, for the Ionosphere and Pendigits datasets, C-RPCL violated less constraints than LCVQE.

For a better understanding of the results obtained for Pendigits, we performed experiments by using only its (two) most informative features — with the same constraints as derived from 15 labeled objects used in the previous experiments. These two features were selected using the well-known Naïve Bayes Wrapper and provide an average classification error of 7% (10-fold cross-validation). Note that this is not a practical procedure. Instead, the use of the known labels for feature selection is only justified by our interest in understanding the obtained results. From this standpoint, Figure 1a illustrates the partitions obtained from the trial with the largest NMI difference between C-RPCL and LCVQE. The cluster and class centroids, as well as the objects used for deriving the constraints (shown as inverted triangles) are highlighted. The density of each class is shown by its contours. One can see that the classes overlap in this subspace and that the class represented by the number "9" has a small cluster to the left hand-side of the larger density area. From the obtained centroids, it can be noted that C-RPCL was less affected by such an "outlier" cluster, which allows it to obtain a better estimate of the cluster mean than LCVQE. For this reason, the partition obtained by LCVQE has more errors for objects of class "8" compared to C-RPCL. To better illustrate this, we present in Figure 1b three objects of class "8" that were correctly classified by C-RPCL only — the remaining algorithms misclassified them. It can be seen that the object on the top-left is a reasonably well-formed "8", whereas the others are harder to classify. Noting that digit recognition is a hard problem, the capability of correctly identifying numbers with deformities (like those in Fig. 1b) is relevant. Also, we speculate that the structure observed in the subspace shown in Fig. 1a is, to some extent, similar to the one existing in the full 16-dimensional space.

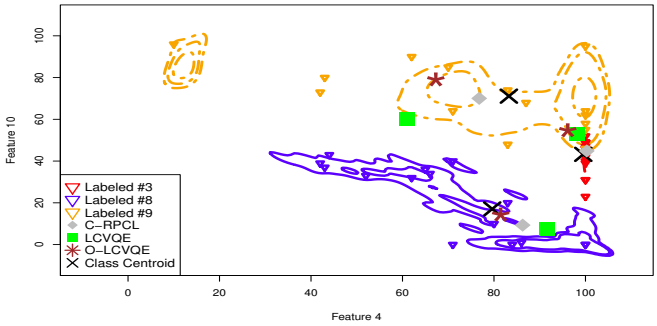
To compare on-line algorithms against batch algorithms on a large dataset, we performed experiments on the KDDCUP'99 Network Intrusion Detection dataset, commonly used to assess streaming clustering algorithms [17]. We considered only classes with more than 1,000 objects, reducing the problem from 23 to 8 types of connections. The dataset comprises 492,368 objects represented by 34 features. We generated constraints using 20 labeled objects per class and repeated the experiments ten times. Performance comparisons are summarized in Table IV. The on-line algorithms were run in a stream fashion, i.e., they make only one pass through the data. Note that intrusion detection applications need a just-in-time processing, making an on-line algorithm even more suitable. From Table IV, one can note that the NMI values obtained by the algorithms are similar. However, the amount of computation time needed by the on-line algorithms is significantly smaller than for the batch algorithm. More specifically, C-RPCL used approximately 33% of the processing time needed by LCVQE, while O-LCVQE used about 23% of the same amount of time.

V. Final Remarks

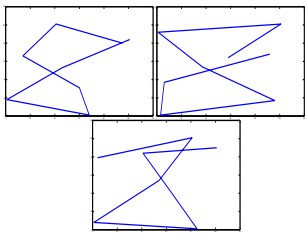
We introduced two competitive learning algorithms for on-line constrained clustering, namely: the Constrained Rival Penalized Competitive Learning (C-RPCL) and the On-line Linear Constrained Vector Quantization Error (O-LCVQE).

Table III: Differences between the number of constraints violated by on-line algorithms and LCVQE (smaller is better).

Algorithm	C-RPCL					O-LCVQE				
# LO / Dataset	5	10	15	20	Average	5	10	15	20	Average
9Gauss	10.8	39.8	75.8	108.5	58.7	12.6	115.2	225.4	398.6	188.0
Ionosphere	1.1	-8.8	-9.3	-45.6	-15.7	1.5	-7.6	9.9	26.6	7.6
Iris	5.6	17.3	42.7	51.3	29.2	3.4	9.7	35.3	57.7	26.5
Wine	4.5	7.1	6.4	1.3	4.8	3.8	5.0	-6.1	94.5	24.3
Breast Cancer	2.0	1.4	14.4	-22.0	-1.0	2.0	-3.4	14.2	4.2	4.2
Pendigits	-6.5	-22.8	-81.9	-108.8	-55.0	10.1	-16.6	-43.0	-7.7	-14.3
Letters	4.3	9.6	22.0	19.6	13.9	4.8	15.7	11.6	16.1	12.1
Pima	3.8	7.2	12.4	13.8	9.3	4.2	4.2	-6.2	-5.0	-0.7



(a) Centroids, labeled objects, and classes — using the most discriminative features only.



(b) Three objects found by C-RPCL in the cluster of class 8 — only C-RPCL correctly classified these objects.

Figure 1: Detailed results on Pendigits dataset.

Table IV: Results obtained on the KDDCUP'99 Network Intrusion Detection dataset (standard deviations in parentheses).

Algorithm	NMI	Time (seconds)
C-RPCL	0.82 (0.03)	40.2 (0.2)
O-LCVQE	0.84 (0.02)	28.5 (0.2)
LCVQE	0.83 (0.02)	121.2 (26.6)

Experimental results show that C-RPCL not only provides better results than O-LCVQE, but also that it can often provide more accurate partitions than the well-known (batch mode) LCVQE algorithm [8]. Moreover, results on a large dataset highlight the computational and memory advantages of using on-line algorithms. As future work, the study of C-RPCL for distributed and asynchronous competitive learning [3], as well as algorithms capable of identifying clusters with different shapes (as in [18]), are promising.

Acknowledgments

This work has been supported by NSF Grants (IIS-0713142 and IIS-1016614) and by FAPESP and CNPq.

References

[1] C. S.-T. Choy and W.-C. Siu, "A class of competitive learning models which avoids neuron underutilization problem," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1258–1269, 1998.

[2] T. Hofmann and J. Buhmann, "Competitive learning algorithms for robust vector quantization," *Signal Processing, IEEE Transactions on*, vol. 46, no. 6, pp. 1665 –1675, 1998.

[3] B. Patra, "Convergence of distributed asynchronous learning vector quantization algorithms," *JMLR*, vol. 12, pp. 3431–3466, 2011.

[4] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel least mean square algorithm," *IEEE Trans. Neural Netw. Learning Syst.*, pp. 22–32, 2012.

[5] T. C. Silva and Z. Liang, "Stochastic competitive learning in complex networks," *IEEE Trans. Neural Netw. Learning Syst.*, pp. 385–398, 2012.

[6] T. Kohonen, "Improved versions of learning vector quantization," in *IJCNN*, 1990, pp. 545–550.

[7] S. Basu, I. Davidson, and K. Wagstaff, *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 2008.

[8] D. Pelleg and D. Baras, "*K*-means with large and noisy constraint sets," in *ECML*, 2007, pp. 674–682.

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

[10] I. Davidson and S. S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm," in *SDM*, 2005, pp. 138–149.

[11] T. F. Covões, E. R. Hruschka, and J. Ghosh, "A study of k-means-based algorithms for constrained clustering," *Intelligent Data Analysis*, no. 3, 2013.

[12] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton, "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3, no. 3, pp. 277 – 290, 1990.

[13] L. Xu, A. Krzyzak, and E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 636 –649, 1993.

[14] A. Asuncion and D. Newman, "UCI machine learning repository," http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007.

[15] R. J. G. B. Campello, E. R. Hruschka, and V. S. Alves, "On the efficiency of evolutionary fuzzy clustering," *Journal of Heuristics*, vol. 15, pp. 43–75, 2009.

[16] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining multiple partitions," *JMLR*, vol. 3, pp. 583–617, 2003.

[17] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *VLDB*, 2003, pp. 81–92.

[18] L. Xu, "A unified perspective and new results on RHT computing, mixture based learning, and multi-learner based problem solving," *Pattern Recogn.*, vol. 40, no. 8, pp. 2129–2153, 2007.