SCALABLE ALGORITHMS FOR CONVEX CLUSTERING

Weilian Zhou[⋆] *Haidong Yi*[†] *Gal Mishne*[‡] *Eric Chi*[⋆]

[⋆] Department of Statistics, NC State Univeristy, Raleigh, NC 27607
[†] Department of Computer Science, UNC-Chapel Hill, Chapel Hill, NC 27599
[‡] Halıcıoğlu Data Science Institute, UC San Diego, La Jolla, CA, 92093

ABSTRACT

Convex clustering is an appealing approach to many classical clustering problems. It stands out among standard methods as it enjoys the existence of a unique global optimal solution. Despite this advantage, convex clustering has not been widely adopted, due to its computationally intensive nature. To address this obstacle, especially in the "big data" setting, we introduce a Scalable cOnvex cLustering AlgoRithm via Parallel Coordinate Descent Method (SOLAR-PCDM) that improves the algorithm's scalability by combining a parallelizable algorithm with a compression strategy. This idea is in line with the rise and ever increasing availability of high performance computing systems built around multi-core processors, GPU-accelerators, and computer clusters. SOLAR-PCDM consists of two parts. In the first part, we develop a method called weighted convex clustering to recover the solution path by formulating a sequence of smaller equivalent optimization problems. In the second part, we utilize the Parallel Coordinate Descent Method (PCDM) to solve a specific convex clustering problem. We demonstrate the correctness and scalability of our algorithm on both simulated and real data examples.

***Index Terms*—** Convex optimization, Parallel computing, Sparsity, Unsupervised Learning

1. INTRODUCTION

Clustering is a fundamental problem in many scientific applications. Many clustering algorithms formulate the clustering task as a non-convex optimization problem, for example the widely used k-means method [1, 2] and its various generalizations to mixture models. A vexing challenge with solving non-convex optimization problems is the presence of suboptimal local minima in the objective function landscape which can trap iterative solvers. These shortcomings have been mitigated but not completely eliminated by strategies that include clever initializations, e.g. [3], and annealing schemes that steer solutions away from local minima [4–6]. An alternative but more direct approach to addressing the issue of local minima is to pose the clustering task as a convex optimization problem so that all local minima are global ones.

In this paper, we focus on a convex formulation of the clustering problem introduced in [7–9]. Given n points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^p$, we seek cluster centroids $\mathbf{u}_i \in \mathbb{R}^p$ for each point $\mathbf{x}_i$ that minimize the convex criterion

$$F_\gamma(\mathbf{U}) = \frac{1}{2}\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{u}_i\|_2^2 + \gamma \sum_{(i,j)\in\mathcal{E}} w_{ij}\|\mathbf{u}_i - \mathbf{u}_j\|_2, \quad (1)$$

where γ is a nonnegative tuning parameter, w_{ij} is a positive weight that quantifies the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, $\mathcal{E}$ is a subset of all possible pairs $i, j = 1, \ldots, n$, and $\mathbf{U} \in \mathbb{R}^{p\times n}$ is the matrix whose ith column is $\mathbf{u}_i$. The sum of squares data-fidelity term in (1) quantifies how well the centroids $\mathbf{u}_i$ approximate the data $\mathbf{x}_i$, while the sum of norms regularization term penalizes the differences between pairs of centroids $\mathbf{u}_i$ and $\mathbf{u}_j$. The regularization term encourages sparsity in the pairwise differences of centroid pairs.

The objective $F_\gamma(\mathbf{U})$ in (1) is strongly convex and consequently possesses a unique minimizer $\mathbf{U}(\gamma)$ for every value of γ. The tuning parameter γ trades off data fit and differences between pairs of centroids. When $\gamma = 0$, the minimum is attained when $\mathbf{u}_i = \mathbf{x}_i$, namely when each point occupies a unique cluster. As γ increases, the regularization term coerces centroids to fuse together. Two points $\mathbf{x}_i$ and $\mathbf{x}_j$ with $\mathbf{u}_i = \mathbf{u}_j$ are said to belong to the same cluster. For sufficiently large γ, the $\mathbf{u}_i$ fuse into a single cluster, namely $\mathbf{u}_i = \bar{\mathbf{x}}$, where $\bar{\mathbf{x}}$ is the average of $\{\mathbf{x}_i\}_{i=1}^n$ [10, 11]. Moreover, the unique global minimizer $\mathbf{U}(\gamma)$ is a continuous function of the parameter γ [12]; we refer to the continuous paths $\mathbf{u}_i(\gamma)$, traced out from each $\mathbf{x}_i$ to $\bar{\mathbf{x}}$ as γ increases, collectively as the solution path.

The solution path to (1) possesses several notable properties. First, simple data-driven choices for the weights w_{ij} ensure that the solution path is a tree and thus a valid hierarchical clustering of the data [13]. Second, the solution path is a 1-Lipschitz function of the data $\{\mathbf{x}_i\}_{i=1}^n$ [14] and consequently small perturbations in the data *cannot* lead to disproportionately large fluctuations in the recovered tree. Third,

GM acknowledges support from the National Institutes of Health (R01EB026936). ECC acknowledges support from the National Science Foundation (DMS-1752692).

hard clustering assignments obtained by selecting a single γ have also been shown to come with cluster recovery guarantees [11, 15–17].

Given its desirable features, there has been a steady progression of approaches for solving (1) efficiently ranging from a variety of first order methods, [9, 10, 16], to second order methods [18], as well as a novel algorithm regularization path approach which remarkably is able to approximate the solution path arbitrarily well with extremely inexact alternating direction method of multiplier updates [19].

In this paper, we propose a two component framework: Scalable cOnvex cLustering AlgoRithm via Parallel Coordinate Descent Method (SOLAR-PCDM) to efficiently compute the solution path $\mathbf{U}(\gamma)$ in the big data setting.

The first component involves solving a sequence of smaller Weighted Convex Clustering variants of (1). Due to the tree recovery properties of convex clustering under appropriate w_{ij} choices [13], if two centroids coincide under a tuning parameter γ_0 they will continue to coincide for all γ greater than γ_0. Thus, as γ increases the number of distinct variables in (1) decreases, and we can express (1) as an equivalent weighted convex clustering problem with a smaller number of variables. Thus, as γ increases, we solve a sequence of increasingly smaller optimization problems.

The second component involves solving the weighted convex clustering problem with a stochastic parallel algorithm, namely the Parallel Coordinate Descent Method (PCDM) [20]. We derive a dual problem of weighted convex clustering which has a partially separable structure, making the dual problem a good candidate to be solved by the scalable algorithm PCDM which can be mapped naturally onto high-performance computing architectures.

The rest of this paper is organized as follows. In Section 2, we derive the weighted convex clustering problem and its dual. In Section 3, we review PCDM and adapt PCDM for our dual problem. In Section 4, we present SOLAR-PCDM by combining the results derived from the previous two sections. In Section 5, we present numerical examples to demonstrate the scalability of SOLAR-PCDM.

2. WEIGHTED CONVEX CLUSTERING

We first formulate the primal weighted convex clustering problem and its dual. Due to space limitations, some details of the derivation have been deferred to the supplement.

Suppose for a given tuning parameter γ in (1), the data has been clustered into K clusters $C_1, \dots, C_K$. For each cluster C_k, we have $\mathbf{u}_i = \bar{\mathbf{u}}_k$ for all $\mathbf{x}_i \in C_k$, where $\bar{\mathbf{u}}_k$ represents the centroid of the kth cluster. Then (1) can be written as

$$F_\gamma^{(w)}(\bar{\mathbf{U}}) = \frac{1}{2}\sum_{k=1}^{K} |C_k| \|\bar{\mathbf{u}}_k - \tilde{\mathbf{x}}_k\|_2^2 + \gamma \sum_{(k,l)\in\mathcal{E}} \hat{w}_{kl}\|\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_l\|_2, \tag{2}$$

where $\hat{w}_{kl}$ is the sum of all weights between points belonging to C_k, C_l, $\tilde{\mathbf{x}}_k = \frac{1}{|C_k|}\sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$, the mean of points in C_k, and $|C_k|$ denotes the number of points in C_k. The function $F_\gamma^{(w)}$ shares a similar form as the function F_γ in (1); the only difference is that the data-fidelity term in (2) is a weighted sum of squares as opposed to the unweighted sum of squares in (1). We call problem (2) a weighted convex clustering problem.

In expressing the convex clustering problem (1) as an equivalent weighted convex clustering problem (2), the dimensions of $\mathbf{U}$ change from $\mathbb{R}^{p\times n}$ to $\mathbb{R}^{p\times K}$. Since K decreases rapidly as γ increases, the dimensions of the weighted convex problem rapidly decreases as γ increases.

Before deriving the dual problem to minimizing (2), we note that [9] built this reweighting scheme into the updates of subgradient updates for the same purpose as we do here. By deriving the weighted objective function explicitly, however, we can set up a dual constrained least squares problem that can be solved by the more scalable PCDM algorithm.

2.1. Dual Problem of Weighted Convex Clustering

Problem (2) can be equivalently formulated as the following equality constrained problem by introducing variables for pairwise differences $\mathbf{v}_{kl} = \bar{\mathbf{u}}_k - \bar{\mathbf{u}}_l$

$$\begin{aligned} \min_{\bar{\mathbf{u}}, \mathbf{v}_{kl}} \ & \frac{1}{2}\|\tilde{\mathbf{D}}(\bar{\mathbf{x}} - \bar{\mathbf{u}})\|_2^2 + \gamma \sum_{(k,l)\in\mathcal{E}} \hat{w}_{kl}\|\mathbf{v}_{kl}\|_2, \\ \text{s.t.} \quad & \mathbf{v}_{kl} = \mathbf{A}_{kl}\bar{\mathbf{u}}, \end{aligned} \tag{3}$$

where $\mathbf{A}_{kl} = (\mathbf{e}_k - \mathbf{e}_l)^\mathsf{T} \otimes \mathbf{I}$, $\mathbf{e}_k$ is the kth standard basis vector in $\mathbb{R}^n$, $\tilde{\mathbf{D}} = \mathbf{D} \otimes \mathbf{I}$, and $\mathbf{D}$ is a diagonal matrix with $d_{kk} = \sqrt{|C_k|}$. The vector $\bar{\mathbf{x}}$ is obtained by stacking the $\bar{\mathbf{x}}_k$ on top of each other; $\bar{\mathbf{u}}$ is similarly obtained from the $\bar{\mathbf{u}}_k$.

The Lagrangian dual problem to (3) is

$$\begin{aligned} \min_{\boldsymbol{\lambda}} \ & \mathcal{D}_\gamma(\boldsymbol{\lambda}) = \frac{1}{2}\|\tilde{\mathbf{D}}\bar{\mathbf{x}} - \tilde{\mathbf{D}}^{-1}\mathbf{A}^\mathsf{T}\boldsymbol{\lambda}\|_2^2, \\ \text{s.t.} \quad & \|\boldsymbol{\lambda}_{kl}\|_2 \le \hat{w}_{kl}\gamma, \quad \forall (k,l) \in \mathcal{E}, \end{aligned} \tag{4}$$

where $\mathbf{A}$ is the matrix obtained by stacking the $\mathbf{A}_{kl}$ matrices on top of each other and the vector $\boldsymbol{\lambda}$ is the vector obtained by stacking the vectors $\boldsymbol{\lambda}_{kl}$ in a matching order.

3. PARALLEL COORDINATE DESCENT METHOD

We have shown that the dimensions of the original problem (1) will be reduced by reformatting it as the weighted version (2). The effect of the dimension reduction, however, depends on the tuning parameter γ. The scale of the problem is still close to its original scale for small γ. Consequently, the weighted convex clustering (2) still faces computational challenges in "big data" scenarios and motivates the need for a scalable algorithm to solve the problem (2).

A highly scalable stochastic parallel coordinate descent method is introduced in [20] for solving problems of the form

$$\text{minimize } f(\boldsymbol{\lambda}) + \Omega(\boldsymbol{\lambda}), \tag{5}$$

where f is a partially separable smooth convex function and Ω is a simple separable function. We call a function f is partially separable if $f(\mathbf{x}) = \sum_{J \in \mathcal{J}} f_J(\mathbf{x})$, where $f_J(\mathbf{x})$ only depends on a block of $\mathbf{x}$. The basic idea is that a separable optimization problem presents an embarrassingly parallel computational task. If the optimization is nearly separable, it turns out that the optimization problem can be solved in parallel with some care. PCDM maps naturally to parallel architectures and can consequently greatly reduce the computing time when using a multi-core computer. In this section, we show the dual problem (4) can be solved by the PCDM.

3.1. PCDM on the Dual Problem

The dual problem (4) has the form defined in (5) with $f(\boldsymbol{\lambda}) = \mathcal{D}_\gamma(\boldsymbol{\lambda})$ and $\Omega(\boldsymbol{\lambda}) = \sum_{(k,l)\in\mathcal{E}} \iota_{C_{kl}}(\boldsymbol{\lambda}_{kl})$ where $C_{kl} = \{\boldsymbol{\lambda}_{kl} : \|\boldsymbol{\lambda}_{kl}\|_2 \leq \hat{w}_{kl}\gamma\}$ and $\iota_C(\boldsymbol{\lambda})$ is an indicator function which vanishes when $\boldsymbol{\lambda} \in C$ and is infinity otherwise.

The details of the verification that $f(\boldsymbol{\lambda})$ and $\Omega(\boldsymbol{\lambda})$ satisfy the prerequisites of PCDM are in the supplement. Algorithm 1 summarizes PCDM for solving the problem (2); we give detailed commentary on Algorithm 1 in the supplement.

Algorithm 1 PCDM for weighted convex clustering

1: Initialize $k = 1$, $\boldsymbol{\lambda}^{(1)} = \mathbf{0} \in \mathbb{R}^{|\mathcal{E}|p\times 1}$, $\bar{\mathbf{u}}^{(1)} = \bar{\mathbf{x}} \in \mathbb{R}^{np\times 1}$ and $d^{(1)} = |F_\gamma^{(w)}(\bar{\mathbf{U}}^{(1)}) - \mathcal{D}_\gamma(\boldsymbol{\lambda}^{(1)})|$
2: **while** $d^{(k)} > \epsilon$ and $k < n_{\max}$ **do**
3: Randomly generate blocks $S^{(k)} \subset \{1, 2, \ldots, |\mathcal{E}|\}$
4: Update $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + (\mathbf{h}(\boldsymbol{\lambda}^{(k)}))_{[S^{(k)}]}$
5: Update $\bar{\mathbf{u}}^{(k+1)} = \bar{\mathbf{x}} - (\tilde{\mathbf{D}}^\mathsf{T}\tilde{\mathbf{D}})^{-1}\mathbf{A}^\mathsf{T}\boldsymbol{\lambda}^{(k+1)}$
6: Update $d^{(k+1)} = |F_\gamma^{(w)}(\bar{\mathbf{U}}^{(k+1)}) - \mathcal{D}(\boldsymbol{\lambda}^{(k+1)})|$
7: $k \leftarrow k + 1$
8: **end while**
9: Calculate $\mathbf{V}$

3.2. Computational Complexity of PCDM

The complexity for the PCDM [20] is $\mathcal{O}\left(\frac{|\mathcal{E}|\beta}{\tau}\frac{1}{\epsilon}\log\left(\frac{1}{\rho}\right)\right)$, where ϵ and ρ quantify the convergence of the result. For each iteration, it would take $\mathcal{O}(np^2|\mathcal{E}|)$ operations to compute $\mathbf{A}^\mathsf{T}\boldsymbol{\lambda}$ in step 5 of Algorithm 1. However, the update in $\boldsymbol{\lambda}$ only requires modifying a few coordinates, which means that the vast majority of the intermediate terms in computing the product $\mathbf{A}^\mathsf{T}\boldsymbol{\lambda}$ stay the same. This observation motivates a simple way to efficiently update $\bar{\mathbf{u}}$. Note that the matrix version of step 5 is $\bar{\mathbf{U}} = \bar{\mathbf{X}} - \boldsymbol{\Lambda}\boldsymbol{\Phi}\mathbf{D}^{-2}$. Moreover, we have the following identity,

$$\boldsymbol{\Lambda}\boldsymbol{\Phi} \quad = \quad \boldsymbol{\Lambda}_{[\mathcal{I}]}(\boldsymbol{\Phi}_{[\mathcal{I}]}^\mathsf{T})^\mathsf{T} + \boldsymbol{\Lambda}_{[\mathcal{I}^c]}(\boldsymbol{\Phi}_{[\mathcal{I}^c]}^\mathsf{T})^\mathsf{T},$$

where $\mathcal{I}$ is the set of column indices to be updated in the iteration. Since $\boldsymbol{\Lambda}_{\mathcal{I}^c}^{(k)}(\boldsymbol{\Phi}_{\mathcal{I}^c}^\mathsf{T})^\mathsf{T}$ is fixed during the iteration, we may more efficiently update $\bar{\mathbf{U}}$ in Algorithm 1 as follows:

$$\bar{\mathbf{U}}^{(k+1)} \quad = \quad \bar{\mathbf{U}}^{(k)} - \mathbf{H}_{[S^{(k)}]}^{(k)}(\boldsymbol{\Phi}_{[S^{(k)}]}^\mathsf{T})^\mathsf{T}\mathbf{D}^{-2},$$

where $\mathbf{H}^{(k)}$ is the matrix form of $\mathbf{h}(\boldsymbol{\lambda}^{(k)})$. This update only needs $\mathcal{O}(pn\tau)$ operations. We will discuss τ in more detail in Section 5.2, but briefly in this paper τ is the number of cores used. Consequently, the more careful $\bar{\mathbf{u}}$ update is an obvious improvement since typically we have $\tau \ll |\mathcal{E}|$. Algorithm 2 summarizes the modified version of the PCDM algorithm.

Algorithm 2 Modified PCDM for weighted convex clustering

1: Initialize $k = 1$, $\boldsymbol{\lambda}^{(1)} = \mathbf{0} \in \mathbb{R}^{|\mathcal{E}|p\times 1}$, $\bar{\mathbf{u}}^{(1)} = \bar{\mathbf{x}} \in \mathbb{R}^{np\times 1}$ and $d^{(1)} = |F_\gamma^{(w)}(\bar{\mathbf{U}}^{(1)}) - \mathcal{D}_\gamma(\boldsymbol{\lambda}^{(1)})|$
2: **while** $d^{(k)} > \epsilon$ and $k < n_{max}$ **do**
3: Randomly generate blocks $S^{(k)} \subset \{1, 2\cdots|\mathcal{E}|\}$
4: Update $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + (\mathbf{h}(\boldsymbol{\lambda}^{(k)}))_{[S^{(k)}]}$
5: Update $\bar{\mathbf{U}}^{(k+1)} = \bar{\mathbf{U}}^{(k)} - \mathbf{H}_{[S^{(k)}]}^{(k)}(\boldsymbol{\Phi}_{[S^{(k)}]}^\mathsf{T})^\mathsf{T}\mathbf{D}^{-2}$
6: Update $d^{(k+1)} = |F_\gamma^{(w)}(\bar{\mathbf{U}}^{(k+1)}) - \mathcal{D}(\boldsymbol{\lambda}^{(k+1)})|$
7: $k \leftarrow k + 1$
8: **end while**
9: Update: Difference matrix $\mathbf{V}$

4. SOLAR-PCDM

The basic idea of SOLAR-PCDM is to iteratively formulate a weighted convex clustering problem as γ increases and apply Algorithm 2 on it to get $\mathbf{U}$ and corresponding difference matrix $\mathbf{V}$. Clusters are identified from connected components which are computed from $\mathbf{V}$ [10]. Algorithm 3 presents SOLAR-PCDM. Steps 6 - 10 in Algorithm 3 detail the updates for the corresponding quantities when cluster information changes. A unique aspect of our algorithm is step 9, we need to find the maximum weight between two clusters. This quantity is used as a threshold to determine whether two clusters have coalesced together.

5. EXPERIMENTAL RESULTS

In this section, we demonstrate (1): The consistency of the solution paths recovered by SOLAR-PCDM; (2): PCDM's speedup factor; (3): SOLAR-PCDM's scalability on large real-world data.

Algorithm 3 SOLAR-PCDM

Initialize $\mathbf{X}^{(0)} = \mathbf{X} \in \mathbb{R}^{p\times n}$, $n_0 = n$, clusters $C_j^{(0)} = \{j\}$, cluster size: $N_j^{(0)} = 1$, for $j = 1, 2 \cdots n_0$. Edge weights: $\{\hat{\mathbf{w}}_{jk}^{(0)}\} = \{\mathbf{w}_{jk}\}$, maximal weight between clusters $\{\mathbf{w}_{jk}^{\star,(0)}\} = \{\mathbf{w}_{jk}\}$, for $1 \le j < k \le n_0$
for $i = 0, \ldots, m-1$ **do**
  Apply Algorithm 2 to problem (2) to compute $\mathbf{U}(\gamma_{i+1}, \{\hat{\mathbf{w}}^{(i)}\})$ and $\mathbf{V}(\gamma_{i+1}, \mathbf{w}^{\star,(i)})$
  for $j = 1, \ldots, n_{i+1}$ **do**
    Get connected components I_j in $\mathbf{V}(\gamma_{i+1}, \mathbf{w}^{\star,(i)})$
    Update clusters $C_j^{(i+1)} = \{l : l \in C_k^{(i)}, k \in I_j\}$
    Update cluster size $N_j^{(i+1)} = \sum_{k\in\ I_j} N_k^{(i)}$
    Update data matrix $\mathbf{X}_j^{(i+1)} = \frac{1}{N_j^{(i+1)}} \sum_{l\in C_j^{(i+1)}} \mathbf{X}_l$
    Update maximum weight for $j < k \le n_{i+1}$ $\mathbf{w}_{jk}^{\star,(i+1)} = \max_{l_1\in C_j^{(i+1)}, l_2\in C_k^{(i+1)}} \mathbf{w}_{l_1 l_2}$
    Update edge weight for $j < k \le n_{i+1}$ $\hat{\mathbf{w}}_{jk}^{(i+1)} = \sum_{l_1\in C_j^{(i+1)}, l_2\in C_k^{(i+1)}} \mathbf{w}_{l_1 l_2}$
  end for
end for

5.1. Dentition of Mammals

To see if the solution paths of the convex clustering algorithm and of SOLAR-PCDM coincide, we consider the problem of clustering mammals based on dentition [10]. Eight different kinds of teeth are tallied for each mammal: the number of top incisors, bottom incisors, top canines, bottom canines, top premolars, bottom premolars, top molars, and bottom molars.

We used sparse Gaussian kernel weights with five nearest neighbors to set the weights between the nodes, namely

$$w_{ij} \quad = \quad \iota_{\{i,j\}}^{(5)} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma}\right), \qquad (6)$$

where $\iota_{\{i,j\}}^{(5)}$ is an indicator function that is 1 when jth point is among the 5-nearest neighbors of ith point and is 0 otherwise. The bandwidth σ is a data-driven parameter. We want the weight for two nodes in the same cluster to be large, whereas the weight between nodes in different clusters should tend to zero. In this experiment, our main focus is on the correctness of the algorithm so we set $\sigma = 2$. Such weights ensure that the solution path will be a tree [13].

We compute the solution path over a range of 100 equally spaced values of γ between 0 and 43. To visualize results we project the data and the solution paths onto the first two principal components of the data. Figure 1 shows that the solution paths recovered by SOLAR-PCDM (upper panel) and convex clustering (lower panel) are virtually identical. To quantify the differences between the two solution paths, Figure 2 plots the computed the max element-wise difference of $\mathbf{U}(\gamma)$ computed by each algorithm at each of the 100 values of γ. The differences are almost all entirely within the stopping condition error tolerance (0.001), confirming that the weighted convex clustering algorithm recovers the solution path generated by convex clustering within designed tolerances.

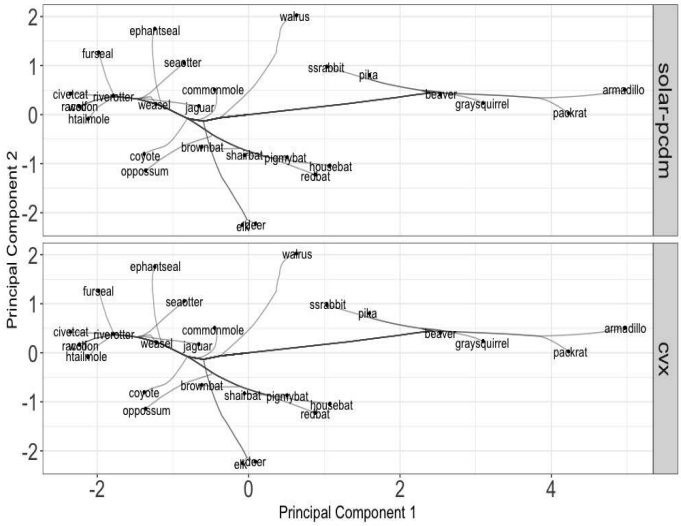

Fig. 1: Comparing the solution paths between convex and weighted convex clustering

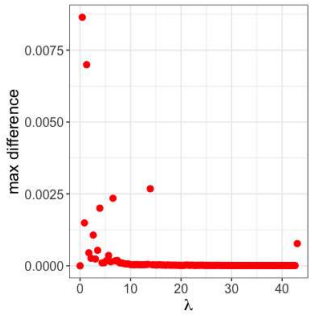

Fig. 2: Max difference of $\mathbf{U}(\gamma)$ in the solution path

5.2. Simulated data

We next investigate the computational gains in using the PCDM algorithm as a function of the number of cores used. Two data sets with different graph structures are simulated in this study. For each data set, we apply the PCDM algorithm under two sampling schemes, τ-nice sampling and fully parallel sampling. For the τ-nice sampling scheme, we randomly update τ edges (same as the number of cores) in each iteration of PCDM and the number of cores varies in $\{1, 2, 4, 8, 16\}$. For the fully parallel sampling, we do not have a large number of cores to update every edge simultaneously. Nonetheless, we can approximate the fully parallel sampling by updating all edges in the same iteration with 16 cores. More specifically, we evenly assign $|\mathcal{E}|$ edges to 16 cores and each core

updates about $[\frac{|\mathcal{E}|}{16}]$ edges each iteration.

Chain Graph: We sample 1000 points from a standard multivariate normal distribution in $\mathbb{R}^7$, taking $\mathcal{E}$ to be a chain graph, namely each node is connected to two nodes with node numbers immediately preceding and following its node number. $w_{ij} \in \mathcal{E}$ is randomly assigned uniformly between 0 and 1.

Gaussian Mixture with kernel weights: We first sample five vectors uniformly in $[-10, 10]^{30}$ as the five cluster centroids $\mathbf{c}_1, \ldots, \mathbf{c}_5$ in a Gaussian mixture. We then sample 1000 points from a 5-component mixture as follows. For each data point $\mathbf{x}_i$, we assign it to one of the 5 components with equal probability, and each component is normally distributed centered in $\mathbf{c}_k$ with an identity covariance matrix, so $\mathbf{x}_i \sim \mathcal{N}(\mathbf{c}_k, \mathbf{I})$, if $\mathbf{x}_i$ was assigned to the kth cluster. We use a 5-nearest neighbors graph with kernel weights (6). Since the expectation of square distance of two nodes is 60 in this case, we set $\sigma = 30$.

For the two settings above, we repeat the simulation for a given number of cores 10 times. We report the mean and standard errors of the wall clock time and the number of iterations for the Chain Graph and Gaussian Mixture settings respectively in Table 1 and Table 2. We observe an anticipated decrease in wall clock time and iteration number as the number of cores increases. There is an approximate linear trend in the number of iterations, which agrees with the theoretical speed-up factor for PCDM [20]. The wall clock time exhibits an expected sublinear trend due to communication costs incurred during parallel computing. The phenomenon is especially severe when the computation task for each core is not large. Both experiments illustrate the speed up gains of applying the PCDM with larger number of cores.

Core	Time (sec)		Iteration	
1	88.46	± 3.84	6557.00	±286.05
2	68.13	± 4.18	3592.40	± 200.59
4	54.41	± 2.12	1816.20	± 90.19
8	46.93	± 2.12	905.60	± 40.95
16	35.23	± 1,75	442.80	± 18.20
16 (fully)	27.28	± 0.03	5.00	± 0.00

Table 1: The computing time and the number of iterations for different number of cores for the Chain Graph.

Core	Time (sec)		Iteration	
1	831.83	± 29.35	25991.20	± 849.30
2	424.79	± 10.11	12386.00	± 207.79
4	233.89	± 8.99	6130.40	± 225.20
8	144.89	± 2.00	3184.80	± 42.21
16	90.27	± 1.14	1578.80	± 27.78
16 (fully)	74.88	± 0.17	20.00	± 0.00

Table 2: The computing time and the number of iterations for different number of cores for the Gaussian Mixture.

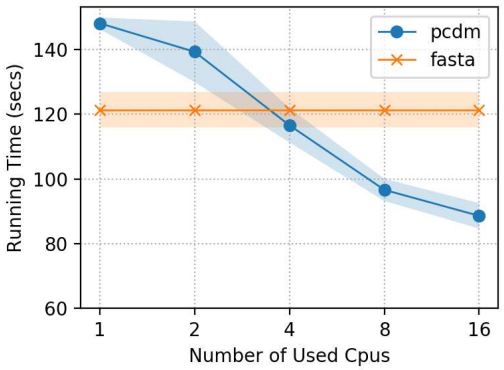

Fig. 3: The computation time for MNIST data

5.3. MNIST handwritten digits

In this section, we use the MNIST handwritten data [21] to show the performance of SOLAR-PCDM on computing the solution path of the convex clustering problem. We compare SOLAR-PCDM to the Fast Adaptive Shrinkage/Thresholding Algorithm (FASTA) [22] applied to the dual problem (2).

We sample 10,000 images from the MNIST dataset. Each sample in the MNIST data is an image data with 28×28 pixels and can be reshaped into a column-major vector in $\mathbb{R}^{784}$. We use a common strategy in pre-processing of MNIST data and apply fast PCA [23] for dimensionality reduction [24], reserving the first 20 principle components. We use the wall clock time for the same solution path on the MNIST data as the comparison measure, and we repeatedly compute it 10 times. The solution path is computed over $\gamma \in \{5, 10, 30, 50, 70, 90, 110, 130\}$. For each γ, the algorithm stops when the relative change of the primal function value is less than 10^{-4}. To get the best performance of PCDM, we use the fully parallel sampling in step 3 of Algorithm 1.

We again use sparse Gaussian kernel weights but instead of setting σ to be a fixed global scale parameter we use a commonly used data-driven strategy of choosing a local scale parameter σ_{ij} that is pair dependent [25], namely

$$w_{ij} \quad = \quad \iota^{(k)}_{\{i,j\}} \exp\left(-\frac{\|x_i - x_j\|_2^2}{\sigma_{ij}}\right).$$

We first compute a local measure of scale σ_i, which is the median Euclidean distance between the ith point x_i and its 5-nearest neighbors. We then set $\sigma_{ij} = \sigma_i \sigma_j$.

Figure 3 shows the wall clock times for computing the solution path by SOLAR-PCDM (blue) and FASTA (orange) over 1, 2, 4, 8, and 16 cores. The shadow around each point represents the variance of wall clock time from repeated experiments. We see that when the number of cores becomes large enough (4 in this case), SOLAR-PCDM outperforms FASTA in wall clock time. Based on the scalable nature of the PCDM the speed-up gains will be even bigger for machines with additional cores.

6. REFERENCES

[1] James MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Berkeley, Calif., 1967, pp. 281–297, University of California Press.

[2] Stuart P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[3] David Arthur and Sergei Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, USA, 2007, SODA '07, p. 1027–1035, Society for Industrial and Applied Mathematics.

[4] Hua Zhou and Kenneth L. Lange, "On the bumpy road to the dominant mode," *Scandinavian Journal of Statistics*, vol. 37, no. 4, pp. 612–631, 2010.

[5] Jason Xu and Kenneth Lange, "Power k-means clustering," Long Beach, California, USA, 09–15 Jun 2019, vol. 97 of *Proceedings of Machine Learning Research*, pp. 6921–6931, PMLR.

[6] Saptarshi Chakraborty, Debolina Paul, Swagatam Das, and Jason Xu, "Entropy weighted power k-means clustering," 26–28 Aug 2020, vol. 108 of *Proceedings of Machine Learning Research*, pp. 691–701, PMLR.

[7] K. Pelckmans, J. De Brabanter, J. Suykens, and B. De Moor, "Convex clustering shrinkage," in *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

[8] Fredrik Lindsten, Henrik Ohlsson, and Lennart Ljung, "Just Relax and Come Clustering! A Convexification of k-Means Clustering," Tech. Rep., Linköpings Universitet, 2011.

[9] Toby Hocking, Jean-Philippe Vert, Francis R. Bach, and Armand Joulin, "Clusterpath: An algorithm for clustering using convex fusion penalties," *Proceedings of the 28th International Conference on Machine Learning*, pp. 745–752, 2011.

[10] Eric C Chi and Kenneth Lange, "Splitting methods for convex clustering," *Journal of Computational and Graphical Statistics*, vol. 24, no. 4, pp. 994–1013, 2015.

[11] Kean Ming Tan and Daniela Witten, "Statistical properties of convex clustering," *Electronic Journal of Statistics*, vol. 9, pp. 2324–2347, 2015.

[12] Eric C. Chi, Genevera I. Allen, and Richard G. Baraniuk, "Convex biclustering," *Biometrics*, vol. 73, no. 1, pp. 10–19, 2017.

[13] Eric C. Chi and Stefan Steinerberger, "Recovering trees with convex clustering," *SIAM Journal on Mathematics of Data Science*, vol. 1, no. 3, pp. 383–407, 2019.

[14] Eric C. Chi, Brian J. Gaines, Will Wei Sun, Hua Zhou, and Jian Yang, "Provable convex co-clustering of tensors," *Journal of Machine Learning Research*, vol. 21, no. 214, pp. 1–58, 2020.

[15] Changbo Zhu, Huan Xu, Chenlei Leng, and Shuicheng Yan, "Convex optimization procedure for clustering: Theoretical revisit," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., pp. 1619–1627. 2014.

[16] Ashkan Panahi, Devdatt Dubhashi, Fredrik D. Johansson, and Chiranjib Bhattacharyya, "Clustering by sum of norms: Stochastic incremental algorithm, convergence and cluster recovery," Sydney, Australia, 06–11 Aug 2017, vol. 70 of *Proceedings of Machine Learning Research*, pp. 2769–2777, PMLR.

[17] Binhuan Wang, Yilong Zhang, Will Wei Sun, and Yixin Fang, "Sparse convex clustering," *Journal of Computational and Graphical Statistics*, vol. 27, no. 2, pp. 393–403, 2018.

[18] Yancheng Yuan, Defeng Sun, and Kim-Chuan Toh, "An efficient semismooth Newton based algorithm for convex clustering," in *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause, Eds., Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018, vol. 80 of *Proceedings of Machine Learning Research*, pp. 5718–5726, PMLR.

[19] Michael Weylandt, John Nagorski, and Genevera I. Allen, "Dynamic visualization and fast computation for convex clustering via algorithmic regularization," *Journal of Computational and Graphical Statistics*, 2019.

[20] Peter Richtárik and Martin Takáč, "Parallel coordinate descent methods for big data optimization," *Mathematical Programming*, vol. 156, no. 1, pp. 433–484, Mar 2016.

[21] Yoshua Bengio Yann LeCun, Léon Bottou and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] Tom Goldstein, Christoph Studer, and Richard Baraniuk, "A Field guide to forward-backward splitting with a FASTA implementation," 2016.

[23] Huamin Li, George C. Linderman, Arthur Szlam, Kelly P. Stanton, Yuval Kluger, and Mark Tygert, "An implementation of a randomized algorithm for principal component analysis," *ACM Trans. Math*, 2017.

[24] Ariel Jaffe, Yuval Kluger, George C. Linderman, Gal Mishne, and Stefan Steinerberger, "Randomized near-neighbor graphs, giant components and applications in data science," *Journal of Applied Probability*, vol. 57, no. 2, pp. 458, 2020.

[25] Lihi Zelnik-Manor and Pietro Perona, "Self-tuning spectral clustering," in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds., pp. 1601–1608. MIT Press, 2005.