TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABREVIATIONS

KDE	Kernel Density Estimate
SSL	Semi Supervised Learning
psd	Positive Semi-definite
KKT	Karush-Kuhn-Tucker
kNN	k Nearest Neighbors

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

$\mathbf{x}_p$	Feature vector of a data with index p
N	Number of data points
K	Number of clusters
M	Feature dimension $\mathbf{x} \in \mathbb{R}^M$
D_p	Degree of a data point p
$\mathbf{D}$	Diagonal Degree Matrix with entries from D_p
$\mathbf{S}$	Cluster label assignment matrix of size $N \times K$
$\mathbf{s}_p$	Probability simplex of size $\mathbb{R}^K$ for cluster assignment of $\mathbf{x}_p$
$s_{p,k}$	Probability assignment value for data point p and cluster k
$\mathbf{m}$	Prototype e.g. mean/mode of a certain class or cluster
$\mathbf{W}$	Affinity matrix $\mathbf{W} = [w(\mathbf{x}_p, \mathbf{x}_q)] \in \mathbb{R}^{N \times N}$
$w(\mathbf{x}_p, \mathbf{x}_q)$	pairwise affinity
t	Transpose operator
V_j	Indicating point assignment to demographic group j s.t. $V_j = [v_{j,p}] \in \{0, 1\}^N$
$\mathbf{1}$	Vector containing 1's
U	Target demographic proportion $U = [\mu_j]$
$\mathcal{A}$	Auxiliary Function
$\mathcal{E}$	Objective Function
$\mathcal{F}$	Clustering Objective

$\vert . \vert$	Cardinality operator
tr	Trace operator
f_θ	Feature embedding function
C	Number of few-shot classes
$\mathbb{X}_\text{s}$	Support set in few-shot task
$\mathbb{X}_\text{q}$	Query set in few-shot task
$\mathbb{X}_\text{base}$	Labeled training set with base classes.
$\mathbf{y}_q$	Binary latent assignment for each data q in few-shot task

INTRODUCTION

The advancement of data acquisition technologies through Internet, digital media, social networks, video surveillance and growing business industries have produced massive amounts of high dimensional data. The manipulation of these huge volumes of data is crucial to learn resourceful models, specific to the related applications, for automatic data analysis or predictive analytics. In this regard, machine learning comes into play as a very promising field, where methods are broadly categorized as *supervised* or *unsupervised*. In supervised learning, a predictive model is built given a set of labeled data points whereas, in the unsupervised setting, the data is available, but without the labels. Unsupervised learning methods are difficult to design, and often lead to difficult optimization problems, due to inherent challenges such as the lack of prior knowledge. Supervised techniques, such as those based on deep learning, have achieved outstanding performances on classification tasks, given huge volumes of correctly labeled datasets, such as the well-known ImageNet (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg & Fei-Fei, 2015), along with powerful and intensive computing resources (e.g., GPUs). However, the collection of those colossal amounts of labeled data is often exhaustive, and may not be possible in a breadth of applications, for instance, in medical image analysis, where annotations from domain experts can be prohibitively costly and time consuming. There are also *semi-supervised* learning methods, which leverage unlabeled data, along with small amounts of labeled data samples that guide the training process. Overall, there is a clear consensus among the learning community that the future of artificial intelligence (AI) will transition towards more unsupervised forms of learning, which leverage information from large-scale amounts of unlabeled data.

In this thesis, we investigate bound-optimization based clustering methods, which integrate various constraints and regularizers (e.g. fairness constraints or Laplacian regularization), and can deal efficiently with large-scale applications. Specifically, we propose flexible models, which optimize multi-term functionals, each integrating clustering objectives with some constraints or regularizers. We further derive tight upper bounds (auxiliary functions) of the functionals to provide scalable (parallel-update) solutions, with convergence guarantee and competitive clustering performances. Furthermore, we address the interesting and challenging *few-shot* learning problem, where only a few labeled examples are available for training a model. Specifically, we propose a transductive Laplacian-regularized inference for few-shot tasks, which minimizes a quadratic binary-assignment function integrating supervision information from the few labeled samples and Laplacian regularization over unlabeled samples. Our transductive few-shot inference can be viewed as a constrained graph clustering, and consistently outperforms state-of-the-art methods by significant margins, across different models, settings and data sets. Furthermore, it is very fast, and can be used for large-scale few-shot tasks.

Motivation and Research objectives

This thesis presents effective and scalable (parallel) bound-optimization methods to solve: 1) joint graph clustering and prototype density-mode estimation for large-scale problems (Scalable Laplacian K-modes (SLK)); 2) to provide *fair* clustering solutions by avoiding biases against sensitive demographic groups, in a flexible paradigm that accommodates different clustering objectives and enables to control the trade-off levels between the clustering and fairness terms (Variational Fair Clustering); And 3) to solve widely investigated and challenging few-shot learning problems with a simple and efficient constrained graph clustering approach, without resorting to complex meta-learning strategies (Laplacian Regularized Few-Shot Learning). We highlight the motivations of each these three research contributions of this thesis in the following sections:

1) Scalable Laplacian K-modes (SLK):

We propose Scalable Laplacian K-modes as our *first contribution* of this thesis. In this part, we advocate a model for joint clustering and density mode finding, *Laplacian K-modes*

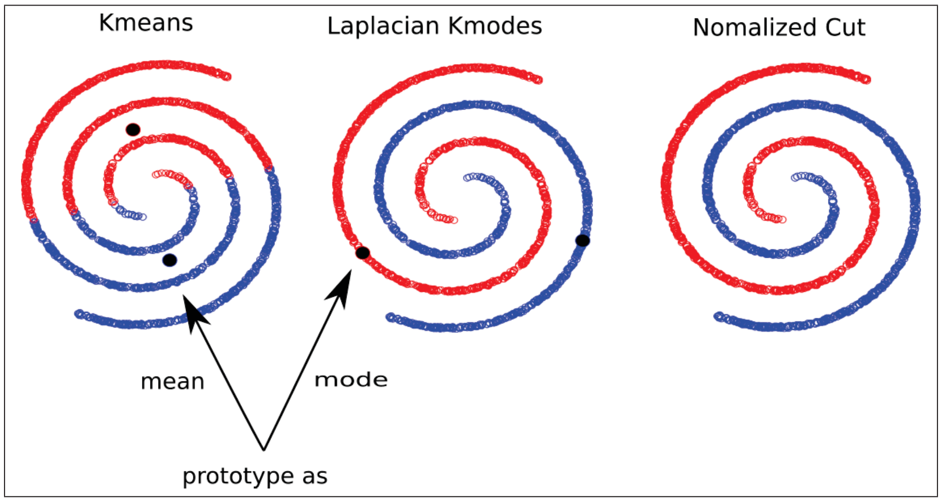


Figure 0.1 Clustering results on a spiral dataset using K-means, Laplacian K-modes and Normalized cut

(Wang & Carreira-Perpinán, 2014), and propose a concave-convex relaxation for the problem, which yields a parallel algorithm that scales up to large datasets and high dimensions. Laplacian K-modes integrates several appealing ideas in clustering. It aims at estimating the mode of the probability density of each cluster, as in the very popular mean-shift algorithm Comaniciu & Meer (2002), while deploying the well-known Laplacian regularizer, widely used in spectral graph clustering (Belkin & Niyogi, 2001; Shi & Malik, 2000; Von Luxburg, 2007b), semi-supervised and representation learning (Belkin, Niyogi & Sindhwani, 2006; Weston, Ratle, Mobahi & Collobert, 2012; Zhu & Ghahramani, 2002). Therefore, the model can handle non-convex (or manifold-structured) clusters, unlike standard prototype-based clustering techniques such as K-means, while finding cluster prototypes as robust density *mode*s, thereby handling outliers better than the *mean*s (see Fig. 0.1). However, optimization of the joint objective is challenging, more so when dealing with large-scale problems, due to the pairwise Laplacian term, the non-linear/non-differentiable dependence between the cluster modes and assignment variables and the integer/simplex constraints.

In fact, it is well known that optimizing the graph Laplacian over discrete simplex variables defined over more than two labels is NP-hard (Tian, Gao, Cui, Chen & Liu, 2014), and it is common to relax the integer constraint. Spectral relaxation (Shi & Malik, 2000) is a dominant technique for solving the problem, subject to balancing constraints, but at the expense of costly eigen-decomposition of the Laplacian matrix is of complexity $O(N^3)$ for a $N \times N$ matrix, which can be prohibitive for large N. Convex relaxation (Wang & Carreira-Perpinán, 2014) of the Laplacian term replaces the integer constraint with a probability-simplex constraint. However, such a convex relaxation requires solving for the assignment variables jointly, with additional simplex-projection overheads, which make it non-applicable for large-scale problems.

Furthermore, optimizing the K-modes over discrete variable is NP-hard (Wang & Carreira-Perpinán, 2014). One way to tackle the problem is to alternate optimization over assignment variables and updates of the modes, with the latter performed as inner-loop mean-shift iterates. Mean-shift moves an initial feature point towards the closest mode via gradient ascent, maximizing the density of feature points. While such a gradient-ascent method has been very popular for low-dimensional distributions over continuous domains, e.g., image segmentation (Comaniciu & Meer, 2002), its use is generally avoided in the context of high-dimensional feature spaces (Chen, Liu, Metaxas & Zhao, 2014). Mean-shift iterates compute expensive summations over feature points, with a complexity that depends on feature-space dimension. Furthermore, the method is not applicable to discrete domains (Chen *et al.*, 2014), as it requires gradient-ascent steps, and its convergence is guaranteed only when the kernels satisfy certain conditions (Comaniciu & Meer, 2002). Finally, the modes obtained at gradient-ascent convergence are not necessarily valid data points in the input set.

We propose a concave-convex relaxation for the problem and derive a tight bound (auxiliary function) for our relaxation, which, at each iteration, amounts to computing an independent update for each cluster-assignment variable, with guaranteed convergence. Therefore, our bound optimizer can be trivially distributed for large-scale data sets. Furthermore, we show that the density modes can be obtained as byproducts of the assignment variables via simple maximum-value operations whose additional computational cost is linear in the number of data points. Our formulation does not need storing a full affinity matrix and computing its eigenvalue decomposition, neither does it perform expensive projection steps and Lagrangian-dual inner iterates for the simplex constraints of each point. Furthermore, unlike mean-shift, our density-mode estimation does not require inner-loop gradient-ascent iterates. It has a complexity independent of feature-space dimension, yields modes that are valid data points in the input set and is applicable to discrete domains as well as arbitrary kernels. We report comprehensive experiments over various data sets, which show that our algorithm yields very competitive performances in term of optimization quality (i.e., the value of the discrete-variable objective at convergence) and clustering accuracy.

2) Variational fair clustering:

The *second contribution* is a general, flexible and scalable framework of fair clustering. Machine learning decisions are already governing day to day activities such as marketing (Perlich, Dalessandro, Raeder, Stitelman & Provost, 2014), awarding home loans (Khandani, Kim & Lo, 2010), and even in sentencing recommendations in courts of law (Kleinberg, Lakkaraju, Leskovec, Ludwig & Mullainathan, 2017). Yet, there is a growing concern that, if not handled consciously, algorithm decisions may lead to discriminatory outcomes for sensitive demographic groups, e.g., gender, race, etc. For example, a higher level of face recognition accuracy may be found with white males (Buolamwini & Gebru, 2018), and a high probability of recidivism tends to be incorrectly predicted for low risk African-Americans (Julia, Larson, Mattu & Kirchner, 2016). These biased decisions may happen due to the biases inherent in the data. This fairness issue has prompted a very active research for designing fair algorithms in the supervised learning setting (Donini, Oneto, Ben-David, Shawe-Taylor & Pontil, 2018; Hardt, Price & Srebro, 2016; Zafar, Valera, Gomez-Rodriguez & Gummadi, 2017). Also, very recently, the research community started to investigate fairness in unsupervised learning (Backurs *et al.*, 2019; Celis, Keswani, Straszak, Deshpande, Kathuria & Vishnoi, 2018; Chierichetti, Kumar, Lattanzi & Vassilvitskii, 2017; Kleindessner *et al.*, 2019; Samadi, Tantipongpipat, Morgenstern, Singh & Vempala, 2018). Specifically, Chierichetti et al. (Chierichetti *et al.*, 2017) pioneered the concept of *fair clustering*, which is still a nascent research area. The problem focuses on how to ensure that clustering solutions have fair (balanced) proportions with respect to certain demographics, and raise many interesting research questions such as: How to embed fairness in popular clustering objectives? What is the cost of fairness with respect to the clustering objective and computational complexity? Can we control the trade-off between some "acceptable" fairness level and the quality of the clustering objective? Can we impose any arbitrary target proportions of the demographics, beyond the balanced proportions sought in (Chierichetti *et al.*, 2017).

Chierichetti et al. (Chierichetti *et al.*, 2017) investigated combinatorial approximation algorithms, which ensure that some fairness measures are within an acceptable range, for K-center and K-median clustering, and for binary demographic groups. They compute *fairlets*, which are groups of points that are fair, and can not be split further into more subsets that are also fair. Then, they consider each fairlet as a data point, and cluster them with approximate K-center or K-median algorithms. Unfortunately, as reported in the experiments in (Chierichetti *et al.*, 2017), obtaining fair solutions with these fairlets-based algorithms comes at the price of a substantial increase in the clustering objectives. Also, the cost for finding fairlets with perfect matching is quadratic with respect to the number of data points, a complexity that increases for more than two demographic groups. Several combinatorial solutions followed-up on the work in (Chierichetti *et al.*, 2017) to reduce this complexity, e.g., (Backurs *et al.*, 2019; Huang, Jiang & Vishnoi, 2019; Schmidt, Schwiegelshohn & Sohler, 2018). Unfortunately, this recent line of combinatorial algorithms is tailored to specific prototype-based objectives. For instance, they are not applicable to very popular graph-clustering objectives such as Normalized Cut

(Von Luxburg, 2007a), which limits applicability in a breadth of graph problems, in which data takes the form of similarities between pairs of points.

Kleindessner et al. (Kleindessner *et al.*, 2019) integrated fairness into graph-clustering objectives, embedding additional linear constraints in popular spectral relaxation. However, it is well-known that spectral relaxation has heavy time and memory loads since it requires storing an $N \times N$ affinity matrix and computing its eigenvalue decomposition, with N the number of data points. The complexity is cubic with respect to N for a straightforward implementation, and super-quadratic for fast implementations (Tian *et al.*, 2014). In the general context of spectral relaxation and graph partitioning, issues related to computational scalability for large-scale problems is driving an active line of recent work (Shaham *et al.*, 2018; Vladymyrov & Carreira-Perpiñán, 2016; Ziko *et al.*, 2018).

The existing fair clustering algorithms, such as the combinatorial or spectral solutions discussed above, do not have mechanisms that control the trade-off levels between the fairness and clustering objectives. Also, they are tailored either to prototype-based (Backurs *et al.*, 2019; Bera, Chakrabarty, Flores & Negahbani, 2019; Chierichetti *et al.*, 2017; Schmidt *et al.*, 2018) or graph-based objectives (Kleindessner *et al.*, 2019). Finally, for a breadth of problems of wide interest, such as pairwise graph data, the computation and memory loads may become an issue for large-scale data sets.

This part of the thesis focuses on a general bound-optimization framework of fair clustering, which integrates an original Kullback-Leibler (KL) fairness term with a large class of clustering objectives, including both prototype-based (e.g., K-means/K-median) and graph-based (e.g., Normalized Cut or Ratio Cut). Fundamentally different from the existing combinatorial and spectral solutions, our variational multi-term approach enables to control the trade-off levels between the fairness and clustering objectives. We derive a general tight upper bound based on a concave-convex decomposition of our fairness term, its Lipschitz-gradient property and the Pinsker inequality. Our tight upper bound can be jointly optimized with various clustering objectives, while yielding a scalable solution, with convergence guarantee. Interestingly, at each iteration, it performs an independent update for each assignment variable. Therefore, it can easily be distributed for large-scale datasets. This scalability is important as it enables to explore different trade-off levels between fairness and the clustering objective. Unlike spectral relaxation, our formulation does not require storing an affinity matrix and computing its eigenvalue decomposition. We report comprehensive evaluations and comparisons with state-of-the-art methods over various fair-clustering benchmarks, which show that our variational method can yield highly competitive solutions in terms of fairness and clustering objectives.

3) Laplacian Regularized Few-shot Learning:

As a *third contribution* of this thesis, we address another very important research problem: Learning classification models using only a few labeled data points from novel (unseen) classes, which is referred to as *few-shot learning*. Unlike the abundant recent few-shot literature, mostly based on complex meta-learning startegies (Finn, Abbeel & Levine, 2017; Snell, Swersky & Zemel, 2017; Sung, Yang, Zhang, Xiang, Torr & Hospedales, 2018; Vinyals, Blundell, Lillicrap, Kavukcuoglu & Wierstra, 2016), we view few-shot inference as a simple *constrained graph clustering* problem, based on Laplacian regularization. Our method provides a new level of state-of-the-art performances, beating significantly a large number of convoluted few-shot learning methods, in all standard public benchmarks.

While deep learning models achieved unprecedented performances, they still have difficulty generalizing to novel classes unseen during training, given only a few labeled instances for these new classes. In contrast, humans can learn new tasks easily from a handful of examples, by leveraging prior experience and related context. Few-shot learning (Fei-Fei, Fergus & Perona, 2006; Miller, Matsakis & Viola, 2000; Vinyals *et al.*, 2016) has emerged as an appealing paradigm to bridge this gap. Under standard few-shot learning scenarios, a model is first trained on substantial labeled data over an initial set of classes, often referred to as the *base* classes. Then, supervision for novel classes, which are unseen during base training, is limited to just one or few labeled examples per class. The model is evaluated over few-shot *tasks*, each one supervised by a few labeled examples per novel class (the *support* set) and containing unlabeled samples for evaluation (the *query* set).

Few-shot learning has recently received substantial research interests in our community, with a large body of work using sophisticated *meta-learning* or *episodic* training strategies (Finn *et al.*, 2017; Snell *et al.*, 2017; Sung *et al.*, 2018; Vinyals *et al.*, 2016). The meta-learning setting uses the base training data to create a set of few-shot tasks (or episodes), with support and query samples that simulate generalization difficulties during test times, and train the model to generalize well on these artificial tasks. Example of very popular methods include matching network, Vinyals *et al.* (2016), which uses an attention mechanism to predict the unknown query samples as a linear combination of the support labels, while using episodic training and memory architectures; prototypical network (Snell *et al.*, 2017), which uses a prototype for each class, and minimize the negative log-probability of the query features with episodic training; the meta-learner in (2017), which views optimization as a model for few-shot learning; and the model-agnostic meta learning method in (2017), which attempts to make a model “easy” to fine-tune. These widely adopted works were recently followed by an abundant meta-learning literature, for instance, (Hou, Chang, Bingpeng, Shan & Chen, 2019; Mishra, Rohaninejad, Chen & Abbeel, 2018; Oreshkin, López & Lacoste, 2018; Rusu, Rao, Sygnowski, Vinyals, Pascanu, Osindero & Hadsell, 2019; Sung *et al.*, 2018; Yanbin, Lee, Park, Kim, Yang, Hwang & Yang, 2019; Ye, Hu, Zhan & Sha, 2020), among many others.

Several recent studies explored *transductive* inference for few-shot tasks, e.g., (Dhillon, Chaudhari, Ravichandran & Soatto, 2020; Hou *et al.*, 2019; Hu, Moreno, Xiao, Shen, Obozinski, Lawrence & Damianou, 2020; Kim, Kim, Kim & Yoo, 2019; Qiao, Shi, Li, Wang, Huang & Tian, 2019; Yanbin *et al.*, 2019), among others. Given a few-shot task at test time, transductive inference performs class predictions jointly for all the unlabeled query samples of the task, rather than one sample at a time as in *inductive* inference. Transductive few-shot methods typically perform better than their inductive counterparts. However, this may come at the price of a much heavier computational complexity during inference. For example, the entropy fine-tuning in (Dhillon *et al.*, 2020) re-trains the network, performing gradient updates over all the parameters during inference. Also, the label propagation in (Yanbin *et al.*, 2019) requires a matrix inversion. The matrix inversion problem is typically solved with algorithms by solving system of linear equations, which has a computational overhead that is cubic or quadratic with better techniques (e.g. CW-like algorithms) with respect to the number of query samples. This may be an impediment for deployment for large-scale few-shot tasks. Also label propagation typically utilizes power method to enhance the speed, however, it does not output the same labeling results as the the optimal solution (Fujiwara & Irie, 2014).

This part of the thesis proposes a transductive Laplacian-regularized inference for few-shot tasks. Given any feature embedding learned from the base data, our method minimizes a quadratic binary-assignment function integrating two types of potentials: (1) unary potentials assigning query samples to the nearest class prototype, and (2) pairwise potentials favoring consistent label assignments for nearby query samples. Our transductive inference can be viewed as a graph clustering of the query set, subject to supervision constraints from the support set, and does not re-train the base model. Following a relaxation of our function, we derive a computationally efficient bound optimizer, which computes independent (parallel) label-assignment updates for each query point, with guaranteed convergence. We conducted comprehensive experiments on five few-shot learning benchmarks, with different levels of difficulties. Using a simple cross-entropy training on the base classes, and without complex meta-learning strategies, our simple clustering method outperforms state-of-the-art methods by significant margins, consistently providing improvements across different settings, data sets, and training models. Furthermore, our transductive inference is very fast, with computational times that are close to inductive inference, and can be used for large-scale few-shot tasks.

Research contributions, organization of the thesis and publications

We start the thesis with a literature review, which focuses on related prototype- and graph-based clustering methods, regularization and fairness constraints for clustering, as well as few-shot learning methods. Thereafter, the research contributions of the thesis are organized chapter-wise as follows:

I Chapter 2 is a detailed description of the proposed scalable Laplacian K-modes method for clustering and density-mode estimation.

   Publication: Scalable Laplacian K-modes, *Neural Information Processing Systems (NeurIPS)*, Vol. 31: pp. 10041–10051, 2018.

II Chapter 3 details the proposed flexible and scalable approach for clustering with fairness constraints.

   Publication: Variational Fair Clustering, Submitted to *Neural Information Processing Systems (NeurIPS)*, 2020.

III Chapter 4 describes our proposed Laplacian regularized few-shot learning model.

   Publication : Laplacian Regularized Few-Shot Learning, Accepted at *International Conference on Machine Learning (ICML)*, 2020; also to appear in the *Proceedings of Machine Learning Research (PMLR)*, Vol. 119, 2020.

Additional research contributions made during this PhD research work:

i Information Maximization for Few-Shot Learning, Submitted to *Neural Information Processing Systems (NeurIPS)*, 2020 – contribution as co-author.

ii Metric learning: cross-entropy vs. pairwise losses, Accepted at *European Conference on Computer Vision (ECCV)*, 2020 – as co-author (equal contribution with the first author).

CHAPTER 1

LITERATURE REVIEW

1.1 Clustering methods

Data clustering methods have been studied as a core part of unsupervised learning. These techniques have been used in a wide range of data analysis applications such as segmentation, classification, business data analysis, market analysis, social network analysis and many other computer vision, machine learning and data mining applications. The aim of a clustering method is to assign similar instances to the same group (cluster), given a pool of instances belonging to different groups. Among numerous popular problems in unsupervised learning, there have been a surge of different clustering algorithms proposed based on exploiting the domain of the particular problem in some specialized field. Thus, one particular clustering algorithm being successful in a specific application may not necessarily succeed in other applications. A large amount of clustering methods appear in the literature, in various fields, including social science, biology, psychology, medical science, mathematics, computer science, to name a few. Therefore, it is very difficult to list, study and link all the existing clustering methods till date in an exhaustive taxonomy. Yet, several surveys of clustering methods are available (Hennig, Meila, Murtagh & Rocci, 2015; Xu & Tian, 2015).

With the very large number of clustering algorithms available in the literature, it is not possible to find a single algorithm capable of yielding the best accuracy in every application. If the clustering method can capture the underlying structure of the data, then one could expect good results. However, it is not possible to know the distribution and nature of the data *a priori*. The complexity of the problem is further compounded in the case of high dimensional applications, as in computer vision problems, where numerous images and videos are involved, even in a single application. There are several major challenges involved in clustering large-scale, high-dimensional data, e.g., images and/or other high-dimensional feature spaces:

- The well-known *curse of dimensionality* problem. It is not yet possible to visualize and represent high-dimensional data accurately. Also, complete enumeration of all sub-spaces in high dimensions is a very challenging task. Finding the nearest (or farthest) feature-space point from another given point becomes difficult as the number of dimensions grows. In fact, the direct use of traditional distance measures between two data points might be irrelevant in feature spaces of very high dimensions;

- The time and memory complexity increases with large-scale and high-dimensional applications, such as high-resolution image/video segmentation, large-scale image clustering, among many other interesting problems;

- The objective of clustering is to gather instances that are neighbors according to observations of their high-dimensional feature-space values. However, when dealing with a large number of attributes (or feature dimensions), some of these might be irrelevant for a given cluster. This is also known as the local feature relevance problem, which states that different clusters might be found in different sub-spaces. Therefore, a *global filtering* of feature dimensions is not sufficient.

This thesis focuses on embedding constraints and regularizers on *graph-based* and *prototype-based* clustering objectives, and on deriving effective and scalable bound optimizers for the ensuing problems. Prototype-based clustering methods such as K-means are simple yet efficient for large-scale data clustering. However, the performance of these relatively simple clustering methods is seriously affected in high dimensions, due to the global filtering of features discussed above. Graph clustering methods such as Normalized cut are based on the affinities between pairs of points, which accounts for local feature relevance of nearby data points, typically yielding much better performances in high-dimensional feature spaces. We discuss prototype- and graph-based clustering in the following sections.

Notations: Suppose that we have N data points $\{\mathbf{x}_p\}_{p=1}^{N} \in \mathbb{R}^M$, which we need to partition into K clusters. Let $\{\mathbf{m}_k\}_{k=1}^{K}$ denote the set of cluster prototypes. Let $\mathbf{S} = \{S_k\}_{k=1}^{K}$ be the sought partition, with each denoting one of the K clusters. We also use $\mathbf{S}_k \in \{0,1\}^N$ to denote the binary assignments of data points withing each cluster.

1.1.1 Prototype based clustering

Prototype-based clustering methods assigns data points to different clusters using some specific distance metric. The representative of each cluster is called prototype. Examples of cluster prototypes include the means (averages), as in the standard K-means algorithm, or the medoid (median), as in K-medoid.

K-means: K-means is one of the most widely used unsupervised clustering algorithms. The basic idea behind the algorithm is to minimize a sum of intra-cluster distances. We need to partition into K clusters based on point-wise similarities with respect to cluster parameters (or prototypes). Let $\{\mathbf{m}_k\}_{k=1}^K$ denote the set of cluster prototypes (one for each cluster), which, as we will see later, would turn out to be data means within the clusters. Let $\mathbf{S} = \{S_k\}_{k=1}^K$ be the sought partition, with each S_k denoting one of the clusters. Then, K-means minimizes the following cost function with respect to both the partition and cluster prototypes.

$$\min_{\mathbf{S}} \sum_{k=1}^{K} \sum_{p \in S_k} \|\mathbf{x}_p - \mathbf{m}_k\|^2 \tag{1.1}$$

Given a randomly initialized set of cluster means $\{\mathbf{m}_k\}_{k=1}^K$ (or an initial partition), the algorithm works by alternating the following two steps until convergence, with each step decreasing the cost function:

1. **Cluster assignment step (Partition update):** This steps minimizes function (1.1) with respect to partition $\mathbf{S}$, with parameters $\{\mathbf{m}_k\}_{k=1}^K$ fixed. It updates clusters $\mathbf{S}$ by assigning each data point $\mathbf{x}_p$ to the cluster having the closest prototype to the point.

2. **Update cluster parameters:** This steps minimizes the function with respect to the parameters, with the partition fixed. Setting the derivative of the function with respect to

each parameter to zero yields closed-form updates, which turn out to be the means within the clusters:

$$\mathbf{m}_k = \frac{1}{|S_k|} \sum_{\mathbf{x}_p \in S_k} \mathbf{x}_p, \; k = 1, 2, \ldots K \tag{1.2}$$

Different initial conditions may lead to different solutions. The algorithm converges to a local optimum, and is sensitive to initialization. One heuristic that works well in practice is to run K-means several times, each corresponding to a different initialization, and to select the solution that correspond to the lowest value of the cost function. The computation complexity of K-means is $O(KNM)$. There are many fast implementations of K-means, which makes it more efficient in the context of large-scale data (Arthur & Vassilvitskii, 2007; Elkan, 2003). Due to its simplicity, K-means is still being used as the most popular clustering technique.

Probabilistic K-means: The distortion of K-means energy can be generalized as follows:

$$\min_{\mathbf{S}} \sum_{k=1}^{K} \sum_{p \in S_k} \|\mathbf{x}_p - \mathbf{m}_k\|_d \tag{1.3}$$

where $\|.\|_d$ is a general distortion measure. When $\|.\|_d$ is the L_2 metric, the function in (1.3) corresponds to *K-means* and the optimal parameter for each cluster is the mean of the cluster. When using different measures, the optimal parameter values may not correpond the means anymore. For instance, the optimal parameters for non-squared L_2 metric are the geometric medians. Exponential distortion measures yield the density modes of the clusters. This corresponds to the *K-modes* (Carreira-Perpiñán, 2015) algorithm, which we will discuss in more details later.

We can further re-write function (1.3) as a *probabilistic K-means* (Kearns, Mansour & Ng, 1998; Tang, Marin, Ayed & Boykov, 2019b) using a general model parameter θ_k and a log-loss function for each cluster:

$$\sum_{k=1}^{K} \sum_{i \in S_k} \|\mathbf{x}_p - \theta_k\|_d = -\sum_{k=1}^{K} \sum_{p \in S_k} \log P(\mathbf{x}_p|\theta_k) \tag{1.4}$$

where $P(.|\theta_k)$ is the density model or probability distribution for for each cluster.

Minimization of the probabilistic K-means objective in (1.4) for different probability distribution models leads to different model fitting clustering methods (Carreira-Perpiñán & Wang, 2013; Rother, Kolmogorov & Blake, 2004; Rousson & Deriche, 2002).

Gaussian Mixture model (GMM) (Rasmussen, 1999) is one of the standard probabilistic mixture model based clustering algorithms, which assumes data points are generated from a mixture of Gaussian distributions. GMM uses the Expectation-Maximization (EM) algorithm for fitting Gaussian models, and can be seen as a generalization of the K-means algorithm to soft assignments. Still, the complexity of GMM is $O(NKM^3)$, making it unpractical for high-dimensional and large-scale applications.

Clustering algorithms based on kernel density estimates have the advantage of being non-parametric, as they do not need to make model assumptions. For instance, in the *Mean-Shift* algorithm, a kernel function is used to estimate the density, as in the well-known Parzen-window approach. GMM and K-means, however, make specific parametric model assumptions as to the distributions of data points withing the cluster. This desirable non-parametric property, and the fact that the algorithm does not require a pre-defined number of clusters, enabled Mean Shift to be widely used for different computer vision applications, such as image filtering, segmentation and tracking, among other applications Comaniciu & Meer (2002); Comaniciu, Ramesh & Meer (2003). In some applications, however, it is important to have a pre-defined number of clusters K. The K-modes algorithm (Ben Salah, Ben Ayed, Yuan & Zhang, 2014; Carreira-Perpiñán, 2015), which runs Mean Shift updates for each cluster prototype, achieves this purpose.

Mean-shift: The Basic idea of the *Mean-shift* algorithm is as follows:

- Run *mean-shift iterations* initialized at every data point;
- All the points that converge to the same mode (high-density point in the input feature space) belong to the same cluster;
- Each mode defines one cluster.

The kernel density estimate (KDE) is evaluated as follows:

$$P(\mathbf{x}) = \frac{1}{N}\sum_{p=1}^{N} w(\mathbf{x}, \mathbf{x}_p) \tag{1.5}$$

Different kernel functions $w(.)$ could be used. A standard choice is the Gaussian kernel with scale parameter σ:

$$w(t) = e^{-(t^2/2\sigma^2)}$$

The *mean-shift iteration* is derived by evaluating the derivative of $P(\mathbf{x})$, setting it equal to 0 and rearranging the terms. This leads to the following fixed-point iterates maximizing density function P: $\mathbf{x}^{(i+1)} = f(\mathbf{x}^i)$, with i the iteration index and f given by:

$$f(\mathbf{x}) = \sum_{p=1}^{N} \frac{w^{'}(\mathbf{x}, \mathbf{x}_p)}{\sum_{q=1}^{N} w^{'}(\mathbf{x}, \mathbf{x}_q)}\mathbf{x}_p \qquad \text{where } w^{'} = dw/dt \tag{1.6}$$

Each data point $\mathbf{x}_p$ is assigned to the mode to which it converged via the mean-shift iteration, $f^{\infty}(\mathbf{x}_p)$. The algorithm is stopped when the relative change in the value of $\mathbf{x}$ is smaller than some non-negative tolerance value.

The advantages of the mean-shift algorithm are:

- There are no model assumptions (other than using a specific kernel), unlike Gaussian mixture models or K-means;
- The algorithm is able to model complex clusters having non-convex (manifold-structured) shapes, unlike K-means. This, however, does not imply that all shapes can be modeled well;
- It enables to determine automatically the number of clusters but depending on the kernel width for e.g. σ in case of Gaussian.
- It handle outliers well as the KDE does not get affected by such outliers significantly.

Mean-shift has a few important limitations. While such derivative based ascent method has been very popular for low-dimensional distributions over continuous domains, e.g., image segmentation (Comaniciu & Meer, 2002), its use is generally avoided in the context of high-dimensional feature spaces (Chen *et al.*, 2014). Mean-shift iterates compute expensive summations over feature points, with a complexity that depends on feature-space dimension. Furthermore, the method is not applicable to discrete domains (Chen *et al.*, 2014), as it requires derivatives, and its convergence is guaranteed only when the kernels satisfy certain conditions (Comaniciu & Meer, 2002). Finally, the modes obtained at convergence are not necessarily valid data points in the input set. In general, while widely used, the most successful applications of mean-shift have been in low-dimensional problems, such as image segmentation problems using a few features (e.g., color and pixel coordinates). Another important limitation is that the number of clusters may change abruptly with the scale parameter of the KDE.

K-modes: The K-modes algorithm in Carreira-Perpiñán & Wang (2013) is a natural combination of two ideas:

1. The cluster assignment idea of K-means.
2. The density maximization idea of mean-shift.

The objective function is defined as follows:

$$\max_{\mathbf{S}} \sum_{k=1}^{K} \sum_{p \in S_k} w(\mathbf{x}_p, \mathbf{m}_k) \tag{1.7}$$

Equation (1.7) provides two interesting limit cases based on kernel bandwidth σ, which appears in the Gaussian kernel:

- when $\sigma \to \infty$, the model becomes the K-means objective;
- when $\sigma \to 0$, the model becomes the K-medoids objective, i.e., the centroids are data points within the input set of samples.

Maximization of (1.7) is NP hard! The authors of Carreira-Perpiñán & Wang (2013) proposed a homotopy algorithm, used over a kernel-width interval, in the case of the Gaussian Kernel. The optimization process in Carreira-Perpiñán & Wang (2013) alternates two steps for each fixed σ:

- Assignment step for fixed centroids $\mathbf{m}_k$;
- Mode-finding step for fixed Assignments $\mathbf{S}$. This step is achieved with mean-shift iterations, with cluster assignments $\mathbf{S}$ fixed from the previous step.

1.1.2 Graph Based Clustering

Most of the clustering methods discussed earlier use standard dissimilarity measures between data points and prototypes, such as the Euclidean distance, so as to achieve a partition of the data. However, standard distance measures may not be effective in high-dimensional feature spaces. The use of *graph Laplacian* from spectral graph theory has led to several widely used graph clustering methods, which can deal effectively with high dimensions, noise and outliers. This category of methods builds an affinity/similarity matrix between pairs of data points, and uses the eigenvectors of the affinity matrix to find the clusters. The eigen decomposition of the Laplacian matrix enables to capture the intrinsic non-linearity of the data, preserving the locality of each cluster. There are very popular methods from this class of *spectral* clustering techniques (Von Luxburg, 2007b), such as Normalized cut (NC), which has been widely used in a breadth of computer vision problems, e.g., unsupervised image segmentation. These spectral clustering techniques are also closely related to very popular non-linear dimensionality reduction methods such as Laplacian Eigenmap (Belkin & Niyogi, 2001).

Normalized Cut (NC): (Shi & Malik, 2000) method is one of the most popular clustering methods based on graph-theoretic formulations and pairwise affinities. NC is inspired by graph-partitioning methods based on the minimum cut (min-cut) criterion, such as (Wu & Leahy, 1993). Let $G(\mathbf{X}, \mathbf{E})$ denotes a graph, where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2..\mathbf{x}_N\}$ is the set of N vertices (or nodes), each corresponding to a data point, and $\mathbf{E}$ is the set of weighted edges, each connecting a

Table 1.1 Graph clustering objectives.

Name	**Objective**
Association	$-\mathbf{S}_k^t \mathbf{W} \mathbf{S}_k$
Min-cut	$\mathbf{S}_k^t \mathbf{W}(\mathbf{1} - \mathbf{S}_k)$
Average-cut (AC)	$\frac{\mathbf{S}_k^t \mathbf{W}(\mathbf{1}-\mathbf{S}_k)}{\mathbf{1}^t \mathbf{S}_k}$
Normalized-cut (NC)	$\frac{\mathbf{S}_k^t \mathbf{W}(\mathbf{1}-\mathbf{S}_k)}{D^t \mathbf{S}_k}$
Self Balanced (Chen, Zhexue Haung, Nie, Chen & Wu, 2017)	$\mathbf{S}_k^t \mathbf{W}(\mathbf{1} - \mathbf{S}_k) + \lambda(\|\mathbf{S}_k\|)^2$
Modularity (Boyd, Bae, Tai & Bertozzi, 2018; Newman, 2006)	$-\mathbf{S}_k^t \mathbb{M} \mathbf{S}_k$, with Modularity matrix $\mathbb{M} \in \mathbb{R}^{N \times N}$: $\mathbb{M}(p,q) = w(\mathbf{x}_p, \mathbf{x}_q) - \frac{d_p d_q}{\sum_p d_p}$

pair of vertices. Each edge connecting $\mathbf{x}_p$ and $\mathbf{x}_q$ carries a non-negative weight based on some kernel-based similarity measure $w(\mathbf{x}_p, \mathbf{x}_q)$. The similarity matrix is constructed as $\mathbf{W} = \{w(\mathbf{x}_p, \mathbf{x}_q)\} \in \mathbb{R}^{N \times N}$.

The task is to partition $G(\mathbf{X}, \mathbf{E})$ into disjoint sets of nodes or clusters S_k. The objective function for NC can be written in terms of *association* as follows:

$$\mathcal{E}_{nc}(\mathbf{S}) = -\sum_{k=1}^{K} \frac{assoc(S_k, S_k)}{assoc(S_k, \mathbf{X})} = -\sum_{k=1}^{K} \frac{\mathbf{S}_k^t \mathbf{W} \mathbf{S}_k}{\mathbf{1}^t \mathbf{W} \mathbf{S}_k} \tag{1.8}$$

The *association* is defined as follows:

$$assoc(S_k, S_k) = \sum_{p \in S_k, q \in S_k} w(\mathbf{x}_p, \mathbf{x}_q) = \mathbf{S}_k^t \mathbf{W} \mathbf{S}_k \tag{1.9}$$

The denominator in (1.8) can be written as follows $\mathbf{1}^T W \mathbf{S}_k = D\mathbf{S}_k = \sum_{p \in S_k} d_p$, where d_p denotes the degree of data point $\mathbf{x}_p$, defined as:

$$d_p = \sum_{q=1}^{N} w(\mathbf{x}_p, \mathbf{x}_q), \tag{1.10}$$

and D is the diagonal degree matrix given by: $D = \mathbf{1}^t\mathbf{W} = diag(d_p)$. Association (AA) is another popular graph clustering objective closely related to NC. AA is also based on min-cut criteria of a graph, given by:

$$\mathcal{E}_{aa}(\mathbf{S}) = -\sum_{k=1}^{K} \frac{assoc(S_k, S_k)}{|S_k|} = -\frac{\mathbf{S}_k^t\mathbf{W}\mathbf{S}_k}{\mathbf{1}^t\mathbf{S}_k} \tag{1.11}$$

Minimizing the NC objective in (1.9) is NP-hard. A generalized eigen value problem is rather formulated to minimize the criterion. Optimization is carried out by minimizing the following relaxed version of the NC objective (Von Luxburg, 2007b):

$$\mathcal{E}_{nc}(\mathbf{S}) = -\mathrm{tr}(\mathbf{Z}^t\mathbf{W}\mathbf{Z});\ \mathbf{Z} = \left[.., \frac{\mathbf{S}_k}{\sqrt{\mathbf{S}_k^t D\mathbf{S}_k}}, ..\right], \tag{1.12}$$

where $\mathbf{Z} \in \mathbb{R}^{N\times K}$ is a relaxed assignment matrix containing normalized indicator vectors. From the expression of relaxed assignment variables $\mathbf{Z}$, one can easily verify the following:

$$\mathbf{Z}^t D\mathbf{Z} = I_K$$

By putting $\mathbf{Z}_r = D^{\frac{1}{2}}\mathbf{Z}$, the problem becomes a trace optimization, subject to orthogonality constraints:

$$\mathcal{E}_{nc}(\mathbf{S}) = -\mathrm{tr}(\mathbf{Z}_r D^{-\frac{1}{2}}\mathbf{W}D^{-\frac{1}{2}}\mathbf{Z}_r);\ s.t.\ \mathbf{Z}_r^t\mathbf{Z}_r = I_K \tag{1.13}$$

To obtain the clustering solution based on spectral relaxation, we compute the eigen value decomposition of the Laplacian matrix, $\mathbf{L} = D^{-\frac{1}{2}}\mathbf{W}D^{-\frac{1}{2}}$. Few eigen vectors of $\mathbf{L}$ are selected based on the corresponding lowest eigenvalues, so as to get an embedding. Finally, K clusters are achieved by applying any general clustering algorithm such as K-means. The difference between several spectral clustering methods actually stems from the different ways of constructing the Laplacian matrix (Belkin & Niyogi, 2001; Fowlkes, Belongie, Chung & Malik, 2004; Meila & Shi, 2001; Ng, Jordan, Weiss et al., 2002; Shi & Malik, 2000). There are different variants of graph clustering methods with different objectives. Some well-known graph clustering objectives are highlighted in Table 1.1.

It is well-known that spectral relaxation has high computational and memory load for large N as one has to store the $N \times N$ affinity matrix and compute explicitly its eigenvalue decomposition, which has a complexity that is cubic with respect to N for a straightforward implementation and, to our knowledge, super-quadratic for fast implementations (Tian *et al.*, 2014). In case of sparse matrix the time and memory complexity may reduce, however, selective eigenvectors computation based on approximate eigenvalue estimation by Lanczos algorithm is still hard computationally which is prone to numerical instability (Arora, Hazan & Kale, 2005). In fact, investigating the scalability of spectral relaxation for large-scale problems is an active research subject (Shaham *et al.*, 2018; Tian *et al.*, 2014; Vladymyrov & Carreira-Perpiñán, 2016). For instance, the studies in (Shaham *et al.*, 2018; Tian *et al.*, 2014) investigated deep learning approaches to spectral clustering, so as to ease the scalability issues for large data sets, and the authors of (Vladymyrov & Carreira-Perpiñán, 2016) examined the variational Nyström method for large-scale spectral problems, among many other efforts on the subject. In general, computational scalability is attracting significant research interest with the overwhelming widespread of interesting large-scale problems (Gong, Pawlowski, Yang, Brandy, Bourdev & Fergus, 2015). Such issues are being actively investigated even for the basic K-means algorithm (Gong *et al.*, 2015; Newling & Fleuret, 2016).

Kernel K-means:

We include kernel K-means as a graph-based clustering method, as it uses pairwise affinity kernels. K-means cannot find non-linearly separable clusters. The idea of kernel K-means is to map the original data points to a high-dimensional feature space using some non-linear function. Thus, the data points become linearly separable in the newly mapped feature space.

The objective function is given by:

$$\min_{\mathbf{S}} \sum_{k=1}^{K} \sum_{p \in S_k} \|\phi(\mathbf{x}_p) - \mathbf{m}_k\|^2 \tag{1.14}$$

where ϕ is the mapping. Setting the derivatives of the objective with respect parameters $\mathbf{m}_k$ equal to zero, it is straightforward to see that the optimal parameters correspond to the means of the clusters in the new space:

$$\tilde{\mathbf{m}}_k = \frac{\sum_{p \in S_k} \phi(\mathbf{x}_p)}{|S_k|}$$

Plugging these optimal means in the kernel K-means objective in (1.14), expanding the Euclidean distances and omitting a constant independent of clustering variable **S**, the objective can be expressed solely in term of dot products $\phi(\mathbf{x}_p)^t \phi(\mathbf{x}_q)$. Therefore, using the Mercer theorem, which states that any kernel function is a dot product in some higher-dimensional feature space, one do not need to know explicitly mapping function ϕ. Instead, we can use the *kernel trick*, replacing the dot products by kernel functions, e.g., polynomial, Gaussian, Sigmoid, etc. In this case, the distances in (1.14) are expressed in terms of pairwise affinities, similarly to the graph-clustering objectives discussed earlier. Interestingly, one can show that the kernel K-means objective in (1.14) is equivalent to the average association (AA) objective discussed earlier (Dhillon, Guan & Kulis, 2004; Tang *et al.*, 2019b). Note, however, the optimization procedure is different from spectral relaxation. Kernel K-means follows an iterative procedure similar to K-means, except that the distances are kernel-induced.

The time complexity of kernel K-means increases quadratically with respect to the number of data points. This is due to the required computation of full $N \times N$ kernel matrix, and the fact that the distances computed for each data point require summations over all points, unlike the standard K-means. There are workarounds, for instance, by randomly selecting n samples among the full data set of N points, such that $n << N$. This is followed by finding the optimal cluster centers based on the sampled data points and, finally, assigning the unsampled data points to the nearest cluster centers. In practice, this naive approach does not perform well, in contrast to the approach computing the full kernel matrix. In Kernel K-means, it is observed that the cluster centers are from a subspace spanned by all the data points. By noting that, approximate Kernel K-means, proposed in (Chitta, Jin, Havens & Jain, 2011), seeks a subspace smaller than the subspace spanned by all data points, while providing clustering results comparable to using the full kernel matrix.

1.1.3 Clustering with constraints

Integrating clustering objectives with additional regularization constraints can achieve outstanding performances. Recently, (Meng Tang,Ben Ayed & Boykov, 2014) proposed to integrate the very popular Normalized Cut (NC) clustering objective along with different Markov Random Field (MRF) constraints. They jointly optimized the ensuing objective function with bound optimization and well-known combinatorial graph cut techniques, such as α-expansion (Boykov & Kolmogorov, 2004; Boykov, Veksler & Zabih, 2001). The method has shown promising results in several computer vision applications, such as image/video segmentation and image clustering.

Laplacian K-modes (Wang & Carreira-Perpinán, 2014) is another example of regularized clustering objectives. As discussed earlier, K-modes clustering uses the same assignment rule as K-means, i.e., it assigns each data point to the closest prototype. Due to this fact, the K-modes algorithm is not able to capture clusters having non-convex shapes (or manifold structures). Thus, the authors of (Wang & Carreira-Perpinán, 2014) aimed at finding manifold-structured clusters by enhancing the K-modes objective with the regularization effect of the graph Laplacian. The latter encourages data points that are close in the feature space to have similar cluster assignments.

Laplacian K-modes: Laplacian K-modes (Wang & Carreira-Perpinán, 2014) adds a graph Laplacian regularization term to the K-modes cost:

$$\min_{\mathbf{S}} \; -\sum_{k=1}^{K}\sum_{p\in S_k} w(\mathbf{x}_p,\mathbf{m}_k) + \frac{\lambda}{2}\sum_{p,q} w(\mathbf{x}_p,\mathbf{x}_q)\|\mathbf{s}_p-\mathbf{s}_q\|^2 \tag{1.15}$$

where, for each point p, $\mathbf{s}_p = [s_{p,1}, \ldots, s_{p,K}]^t$ denotes a binary assignment vector, which is constrained to be within the K-dimensional simplex: $s_{p,k} = 1$ if p belongs to cluster k and $s_{p,k} = 0$ otherwise. The first term is the K-modes term, and the second term is the well-known Laplacian regularization. The graph Laplacian, as discussed earlier, is widely used in the context of spectral clustering (Shi & Malik, 2000; Von Luxburg, 2007b). It is also widely used in semi-supervised learning (SSL) (Weston *et al.*, 2012; Zhu & Ghahramani, 2002) and representation learning (Belkin & Niyogi, 2001; Belkin *et al.*, 2006). This Laplacian term encourages nearby data points to have similar cluster assignments. Also the addition of the Laplacian term enables to handle non-convex (manifold-structured) clusters, which is not possible when using prototype-based clustering objectives, such as K-means or K-modes, alone. However, the model in (1.15) yields a challenging optimization problem, due the discrete pairwise (quadratic) Laplacian term and the simplex/integer constraints, more so when dealing with large-scale problems. Well-known spectral relaxation, which was discussed in the previous section for NC, might be be possible, although it is not clear/straightforward how to handle the first K-modes term, in a way that enables to write the overall problem as a trace optimization. Also, spectral relaxation has scalability issues due to the need of expensive eigen decompositions for large matrices. Relaxation of the integer constraints to probability-simplex constraints can provide a convex relaxation (when the modes or prototypes are fixed). For instance, the authors of (Wang & Carreira-Perpinán, 2014) used such a convex relaxation, and proposed a proximal gradient method. Unfortunately, the method requires solving over $N \times K$ variables altogether, with an additional overhead for simplex projection steps. This makes the computational load prohibitive for large-scale clustering problems.

Clustering with Fairness constraints: Clustering under fairness constraints has triggered significant research interest recently, and is a nascent field of ethical machine learning. It has been shown recently that machine learning models may exhibit biases towards specific demographic groups, due to biases inherent in the data itself. For example, a higher level of face recognition accuracy may be found with white males (Buolamwini & Gebru, 2018), and a high probability of recidivism tends to be incorrectly predicted for low-risk African-Americans (Julia *et al.*, 2016). Recently, the community started to investigate fairness constraints in unsupervised learning (Backurs *et al.*, 2019; Celis *et al.*, 2018; Chierichetti *et al.*, 2017; Kleindessner *et al.*, 2019; Samadi *et al.*, 2018). Specifically, the authors of Chierichetti *et al.* (2017) pioneered the concept of *fair clustering*. The problem consists of embedding fairness constraints, which encourage clusters to have balanced demographic groups pertaining to some sensitive attributes (e.g., sex, gender, race, etc.), so as to counteract any form of data-inherent bias.

Assume that we are given N data points to be assigned to a set of K clusters, and let $\mathbf{S}_k \in \{0,1\}^N$ denotes a binary indicator vector, whose components take value 1 when the point is within cluster k, and 0 otherwise. Also, suppose that the data contains J different demographic groups, with $V_j \in \{0,1\}^N$ denoting a binary indicator vector of demographic group j. The authors of (Chierichetti *et al.*, 2017; Kleindessner *et al.*, 2019) suggested to evaluate fairness in terms of cluster-balance measures, which take the following form:

$$\text{balance}(\mathbf{S}_k) = \min_{j \neq j'} \frac{V_j^t \mathbf{S}_k}{V_{j'}^t \mathbf{S}_k} \in [0,1] \tag{1.16}$$

The higher this measure, the fairer the cluster. The overall clustering balance is defined by the minimum of Eq. (1.16) over k. This notion of fairness in clusters has given rise to a new line of research that was introduced, mostly, for prototype-based clustering (e.g., K-center and K-median and K-means) (Backurs *et al.*, 2019; Bera *et al.*, 2019; Chierichetti *et al.*, 2017; Schmidt *et al.*, 2018) and, very recently, for spectral graph clustering (Kleindessner *et al.*, 2019). It raises several interesting questions. How to embed fairness in popular clustering objectives? Can we control the trade-off between some "acceptable" fairness level and the quality of the clustering objective? What is the cost of fairness with respect to the clustering objective and computational complexity?

(Chierichetti *et al.*, 2017) investigated combinatorial approximation algorithms, which ensure the fairness measures in Eq. (1.16) are within an acceptable range, for K-center and K-median clustering, and for binary demographic groups ($J = 2$). They compute *fairlets*, which are groups of points that are fair, and can not be split further into more subsets that are also fair. Then, they consider each fairlet as a data point, and cluster them with approximate K-center or K-median algorithms. Unfortunately, as reported in the experiments in (Chierichetti *et al.*, 2017), obtaining fair solutions with these fairlets-based algorithms comes at the price of a substantial increase in the clustering objectives. Also, the cost for finding fairlets with perfect matching is quadratic w.r.t the number of data points, a complexity that increases for more than two demographic groups. Several combinatorial solutions followed-up on the work in (Chierichetti *et al.*, 2017) to reduce this complexity. For instance, (Backurs *et al.*, 2019) proposed a solution to make the fairlet decomposition in (Chierichetti *et al.*, 2017) scalable for $J = 2$, by embedding the input points in a tree metric. (Rösner & Schmidt, 2018) designed a 14-approximate algorithm for fair K-center. (Huang *et al.*, 2019; Schmidt *et al.*, 2018) proposed fair K-means/K-median based on coreset – a reduced proxy set for the full dataset. Bera *et al.* (2019) provided a bi-criteria approximation algorithm for fair prototype-based clustering, enabling multiple groups ($J > 2$). It is worth noting that, for large-scale data sets, (Bera *et al.*, 2019; Chierichetti *et al.*, 2017; Rösner & Schmidt, 2018) sub-sample the inputs to mitigate the quadratic complexity with respect to N.

More importantly, the combinatorial algorithms discussed above are tailored for specific prototype-based objectives. For instance, they are not applicable to the very popular graph-clustering objectives discussed earlier, e.g., Normalized Cut (Von Luxburg, 2007a), which limits applicability in a breadth of graph problems, in which data takes the form of pairwise affinities. (Kleindessner *et al.*, 2019) integrated fairness into graph-clustering objectives. They embedded linear constraints on the assignment matrix in spectral relaxation. Then, they solved a constrained trace optimization via finding the K smallest eigenvalues of some transformed Laplacian matrix. However, it is well-known that spectral relaxation has heavy time and memory loads since it requires storing an $N \times N$ affinity matrix and computing its eigenvalue decomposition.

The existing fair clustering algorithms, such as the combinatorial or spectral solutions discussed above, do not have mechanisms that control the trade-off levels between the fairness and clustering objectives. Also, they are tailored either to prototype-based (Backurs *et al.*, 2019; Bera *et al.*, 2019; Chierichetti *et al.*, 2017; Schmidt *et al.*, 2018) or graph-based objectives

(Kleindessner *et al.*, 2019). Finally, for a breadth of problems of wide interest, such as pairwise graph data, the computation and memory loads may become an issue for large-scale data sets.

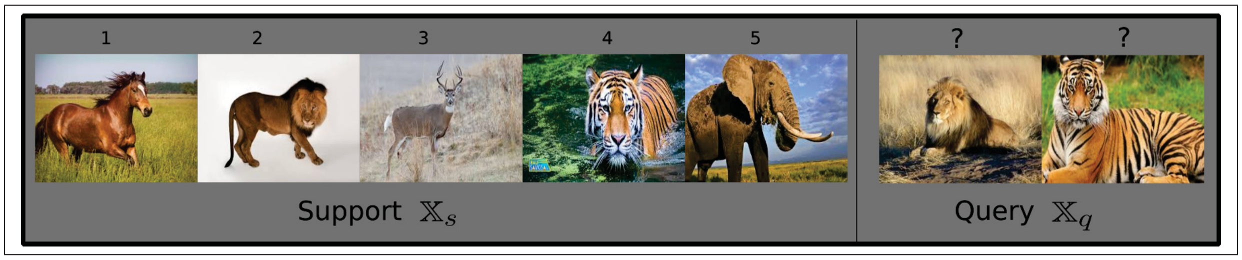


Figure 1.1 A few-shot task with support examples and unknown query samples.

1.2 Few-shot Learning

In the few-shot setting, we are given a labeled support set $\mathbb{X}_\text{s}$, where each few-shot class has a few (e.g. 1 or 5) labeled examples. The objective of few-shot learning is, therefore, to accurately classify unlabeled unseen query sample set $\mathbb{X}_\text{q}$ from these classes; see the example in Fig. 1.1. Generally, few-shot settings assume that we are also given a training set $\mathbb{X}_\text{base}$, which contains enough labeled data points, with base classes $\{1, \dots A\}$ that are different from the test classes of a few-shot task. An initial embedding function f_θ is learned over $\mathbb{X}_\text{base}$, typically with a deep convolutional neural network, with parameters θ and $\mathbf{x} = f_\theta(x) \in \mathbb{R}^M$ is the encoded features of a given input data point x. Most approaches in the few-shot literature use *meta-learning* (Finn *et al.*, 2017; Snell *et al.*, 2017; Sung *et al.*, 2018; Vinyals *et al.*, 2016), which differs from regular supervised training based on the standard softmax classification loss. Episodic training stems from the idea of emulating the testing condition of a few-shot task, but during the base learning of f_θ. More specifically, it splits the training set $\mathbb{X}_\text{base}$ into *episodes*. An *episode* is created by first randomly subsampling some classes from the training set. This is followed by randomly sampling a batch $\mathbb{X}_\text{s}$, taking a few examples from each class. A fraction of the rest of samples from the respective classes are selected as query samples, so as to form a query set $\mathbb{X}_\text{q}$. Thus, an episode consists of support set $\mathbb{X}_\text{s}$ and query set $\mathbb{X}_\text{q}$. The set of all $\mathbb{X}_\text{s}$ and $\mathbb{X}_\text{q}$ is called the *meta-train* set, when composed from base set $\mathbb{X}_\text{base}$. The learning model obtained from the meta-train set is typically called *meta-learner*. The idea is to penalize the model trained on the support set $\mathbb{X}_s$ for not being able to classify query samples $\mathbb{X}_q$ properly during episodic training. The way of penalizing mis-classification is what makes the difference among the different approaches in the literature. The few-shot task with $\mathbb{X}_s$ and $\mathbb{X}_q$ from the *novel classes* are called *meta-test* set. Most of the few-shot learning papers based on meta-learning follows this setting (see Fig. 1.2). We discuss some of the well-known works in the following sections.

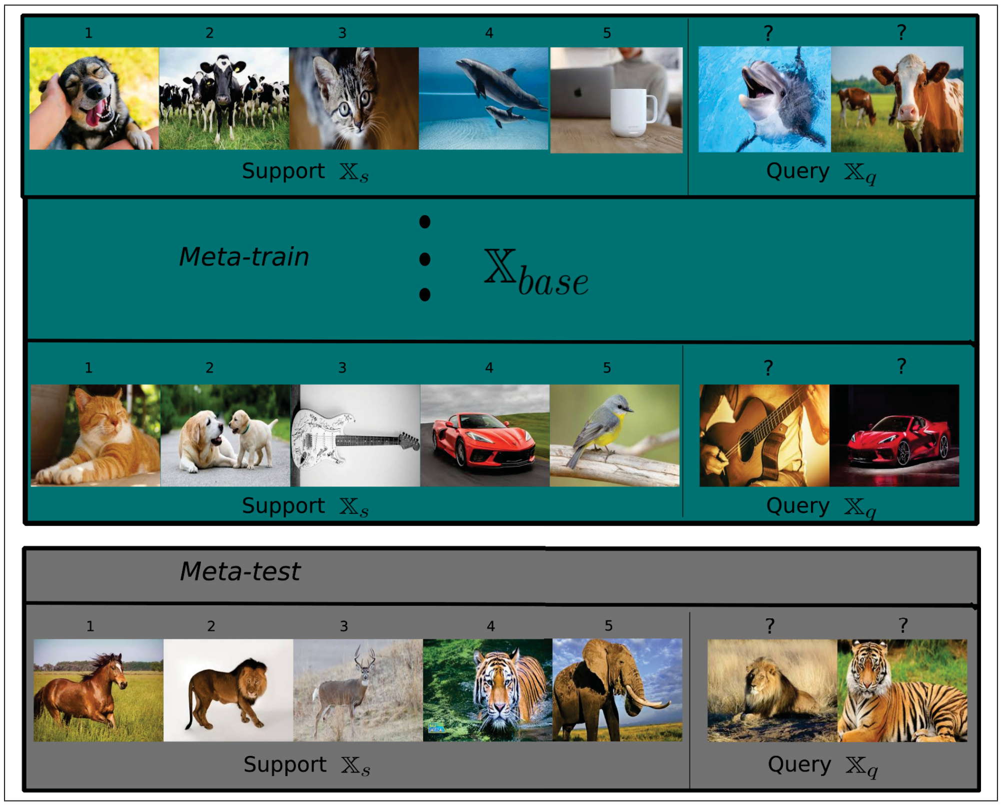


Figure 1.2 Meta-learning framework using episodic training.

1.2.1 Non parametric meta-learning

Matching networks: Matching networks (Vinyals *et al.*, 2016) use episodic training. They utilize an attention kernel w to measure the similarity between a query point $\mathbf{x}_q \in \mathbb{X}_q$ and the points in the support set $\mathbf{x}_s \in \mathbf{X}_s$. More specifically, the probability of a query point belonging to class c is computed as follows:

$$P(c|\mathbf{x}_q, \mathbb{X}_\mathrm{s}) = \sum_{\mathbf{x}_s \in c} a(\mathbf{x}_q, \mathbf{x}_s) \tag{1.17}$$

In the paper, kernel w is chosen to be the softmax over cosine similarity in an embedded feature space. Using $cos(p, q)$ to denote the cosine similarity between two feature points $\mathbf{x}_p$ and $\mathbf{x}_q$, this is given by:

$$a(\mathbf{x}_p, \mathbf{x}_q) = \frac{e^{cos(p,q)}}{\sum_{q'} e^{cos(p,q')}} \tag{1.18}$$

Thus, the training loss becomes just the standard cross-entropy based on the predicted labels in (1.17), given the ground-truth labels:

$$\mathcal{L}_{matching} = -\sum_{\mathbf{x}_q \in \mathbb{X}_\mathrm{q}} \sum_{c} \log P(c|\mathbf{x}_q, \mathbb{X}_\mathrm{s})$$

Predictions for the test few-shot task are simply obtained from (1.17).

ProtoNet: Prototypical networks (Snell *et al.*, 2017) uses distance metric $d(\mathbf{x}_q, \mathbf{m}_c)$ (such as Euclidean) of data $\mathbf{x}_q$ to prototype $\mathbf{m}_c$, to predict the posterior probability of the point belonging to class c.

$$P(c|\mathbf{x}_q, \{\mathbf{m}_c\}) = \frac{e^{-d(\mathbf{x}_q, \mathbf{m}_c)}}{\sum_{c'} e^{d(\mathbf{x}_q, \mathbf{m}_{c'})}} \tag{1.19}$$

Prototypes $\mathbf{m}_c$ can be simply estimated as the mean features of support examples for each class c. This amounts to a soft nearest-prototype classification utilized during meta-training, so as to learn a proper distance metric. The negative log-posterior based on probability $P(c|.)$ in (1.19) is used as a loss to optimize the network parameters θ:

$$\mathcal{L}_{proto} = -\sum_{\mathbf{x}_q \in \mathbb{X}_\mathrm{q}} \sum_{c} \log P(c|\mathbf{x}_q, \{\mathbf{m}_c\})$$

Relation networks: (Sung *et al.*, 2018) also use episodic training. The idea is very similar to ProtoNet, with minor differences. First, instead of minimizing the distance of each point to the prototypes, they minimize the distance to every data point in the support set. Second, the distance metric is learnt via an additional network. In more details, for every pair of points $(\mathbf{x}_s, \mathbf{x}_q) \in \mathbb{X}_\mathrm{s} \times \mathbb{X}_\mathrm{q}$, a score r_{sq} is computed:

$$r_{sq} = g_\Phi(\mathbf{x}_s, \mathbf{x}_q) \tag{1.20}$$

where $g_\Phi \longrightarrow [0, 1]$ is the distance network. Then the loss is computed as:

$$\mathcal{L}_{relation} = \sum_{\mathbf{x}_s \in \mathbb{X}_\mathrm{s}} \sum_{\mathbf{x}_q \in \mathbb{X}_\mathrm{s}} (\mathbf{1}(y_s = y_q) - r_{sq})^2$$

where y_s and y_q are the ground truth labels of $\mathbf{x}_s$ and $\mathbf{x}_q$ from base classes in $\mathbb{X}_\mathrm{base}$, which are used during episodic base training. Finally the prediction during inference (or test) is done as as follows:

$$P(c|x, W) \propto e^{g_\Phi(W_c, f_\phi(x))}$$

Where $\mathbf{m}_c \leftarrow \sum_{\mathbf{x}_s \in \mathbb{X}_\mathrm{s} : y_s = c} \mathbf{x}_s$

1.2.2 Optimization based meta-learning

MAML: Model Agnostic Meta-Learning (MAML) (Finn *et al.*, 2017) is an optimization-based inference model, which, for a target few-shot task, optimizes parameter θ from an initial prior parameter ϕ in a network model, as follows:

$$\max_\theta \log P(\mathbf{X}_s|\theta) + \log P(\theta|\phi) \tag{1.21}$$

A typical prior could be the initialization from fine-tuning:

$$\theta \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}(\phi, \mathbb{X}_\mathrm{s}) \tag{1.22}$$

Where the fine-tuning is done over many gradient steps optimizing loss $\mathcal{L}$, which is computed over support set $\mathbb{X}_\text{s}$. Here, α is the learning rate (or step size) parameter for gradient descent. Meta-learning in MAML amounts to updating ϕ by optimizing:

$$\min_{\phi} \sum_{\mathbf{X}_s} \mathcal{L}(\phi - \alpha \nabla_{\phi} \mathcal{L}(\phi, \mathbb{X}_\text{s}), \mathbb{X}_\text{q}) \tag{1.23}$$

Therefore, the main idea is to learn a set of parameters ϕ over many tasks, which can be transferable to target few-shot task parameters θ, via fine-tuning. This overly complicated procedure actually does not work for deeper networks, and has a very slow convergence speed. In fact, for a network with only 4 convolutional layers, MAML takes almost 60k iterations, which takes almost 30 hours on a Nvidia V100 GPU.

There is a other variant called **Meta-LSTM** (Ravi & Larochelle, 2017), which replaces the term $\nabla_{\phi} \mathcal{L}(\phi, \mathbb{X}_\text{s})$ in (1.22) by a black-box learned network $f(\phi, \mathbb{X}_\text{s}, \nabla_{\phi} \mathcal{L})$, such as cell state update of LSTM.

1.2.3 Baselines based on regular training

Taking a few steps backwards, several very recent works (Chen, Liu, Kira, Wang & Huang, 2019; Dhillon *et al.*, 2020; Wang *et al.*, 2019) have highlighted that very simple baselines actually outperform the overly convoluted episodic-training methods. All these works rely on the same idea of using regular training with simple cross-entropy on the support set, in order to initialize the classifier. The classifier is composed of an encoder f_θ, and a linear classifier layer W, b, with a network posterior probability output defined as a softmax function:

$$P_\theta(y|x) \propto e^{W_y f_\theta(x) + b_y}$$

where y is the ground truth class label: $y \in \{1, \ldots A\}$ from the base classes for training. Therefore, the first part of the training just minimizes the cross-entropy defined over the full base training data:

$$\theta, W, b \leftarrow \underset{\theta,W,b}{\arg\min} \sum_{x \in \mathbb{X}_{\text{base}}} -\log P_\theta(y|x) \tag{1.24}$$

The differences between these baselines comes at test time, i.e., in the methods' ways of using the samples from support set $\mathbb{X}_{\text{s}}$, so as to maximize accuracy on query set $\mathbb{X}_{\text{q}}$. Different variants are briefly discussed below:

Baseline (Chen *et al.*, 2019) uses support samples from $\mathbb{X}_{\text{s}}$ to re-train the final linear classifier from scratch, while keeping the encoder f_θ fixed. Two versions of this baseline are proposed:

- *Baseline:*

$$\textbf{Adaptation: } W, b \leftarrow \underset{W,b}{\arg\min} \sum_{x \in \mathbb{X}_{\text{s}}} -\log P_\theta(y|x) \tag{1.25}$$

$$\textbf{Testing: } P_\theta(y|x) \propto \exp(W_y^t f_\theta(x)) \tag{1.26}$$

- *Baseline++:* In this version, features are normalized, so that the softmax classifier becomes equivalent to a distance-based classifier:

$$\textbf{Testing: } P_\theta(y|x) \propto \exp(\frac{W_y^t f_\theta(x)}{\|W_y\| \, \|f_\theta(x)\|}) \tag{1.27}$$

SimpleShot (Wang *et al.*, 2019) uses a nearest-neighbor approach directly on the features learnt from base classes. The support set is used to compute prototypes $\mathbf{m}_c$ for each class c. The prediction for a test feature point $\mathbf{x}_q$ is then performed using distances to prototypes. More precisely, they propose three approaches based on different normalization techniques:

- *Unnormalized:* Features are kept as provided by the feature extractor f_θ.

$$P_\theta(c|x) \propto \exp(-\|f_\theta(x) - \mathbf{m}_c\|^2)$$

- *L2-normalized:* Features provided by the feature extractor are L2-normalized, in which case the distance-based classifier becomes equivalent to the following:

$$P_\theta(c|x) \propto \exp \frac{\mathbf{m}_c^t f_\theta(x)}{\|\mathbf{m}_c\| \, \|f_\theta(x)\|}$$

- *L2-normalized and centered:* Centering using prototypes from the Base classes is used before L2-normalization

$$\textbf{Base mean } \bar{\mathbf{x}} = \mathbf{x} - \frac{1}{|\mathbb{X}_{\text{base}}|} \sum_{\mathbf{x} \in \mathbb{X}_{\text{base}}} \mathbf{x}$$

$$P_\theta(c|x) \propto \exp \frac{\mathbf{m}_c^t \bar{\mathbf{x}}}{\|\mathbf{m}_c\| \, \|\bar{\mathbf{x}}\|}$$

A baseline for few shot (Dhillon *et al.*, 2020) differs from those discussed above in several implementation details. The first-stage of learning from the base classes is almost the same. Then, the first difference in the adaptation phase is that the logits of the previously learnt model are used as features: $\mathbf{x} = W f_\theta(x) + b \in \mathbb{R}^{|A|}$, where A is the number of base classes in $\mathbb{X}_{\text{base}}$. An additional linear layer is trained on top of these features: $\{W', b'\}$, $W' \in \mathbb{R}^{|C| \times |A|}$, $b' \in \mathbb{R}^{|C|}$ where C is the number of novel classes for a target few-shot task. This is quite unusual, as the common choice is to simply drop the previous classifier and use the features $\mathbf{x} = f_\theta(x)$. Then they propose two baseline methods:

- A support based initialization that uses the prototypes from the support set to build the following distance-based classifier:

$$P_\theta(c|x) \propto \exp \frac{(W_c)_+^T (f_\phi(x))_+}{\|(W_k)_+\| \, \left\|(f_\phi(x))_+\right\|}$$

  where $(.)_+$ is the ReLU non-linearity.

- A transductive fine-tuning approach by utilizing the query set $\mathbb{X}_\text{q}$: During this phase, all the parameters (including feature extractor f_θ) are trained. Noting $\theta = \{\theta', W', b'\}$ and

$\phi' = \{\theta, W, b\}$, their semi-supervision like objective reads:

$$\textbf{Fine-tuning: } \theta \leftarrow \arg\min_{\theta} \frac{1}{|\mathbb{X}_s|} \sum_{(x,y)\in\mathbb{X}_s} -\log P_\theta(y|x) + \frac{1}{|\mathbb{X}_q|} \sum_{x\in\mathbb{X}_q} \mathrm{H}(P_\theta(.|x))$$

$$\textbf{Testing: } P_\theta(c|x) \propto \exp((W_c')^t f_{\theta'}(x))$$

Where H represents the entropy of the predicted distribution for a sample. Just like in semi-supervised learning, this amounts to encouraging peaked distributions for unlabelled query samples. Note that transductive fine-tuning is very slow, as predictions require updating the model parameters using the query set during the inference.

1.2.4 Inductive vs Transductive inference

It is worth mentioning that few-shot learning approaches can be further categorized into *inductive* and *transductive* inference methods. *Transductive* inference methods utilize the available unlabeled query samples of a given few-shot task at once, instead of one sample at a time as in *inductive* inference methods. Classical transductive methods (Dengyong, Bousquet, Lal, Weston & Schölkopf, 2004; Joachims, 1999; Vapnik, 1999) has already been shown to outperform inductive methods on small training sets. This trend is again confirmed in recent few-shot learning approaches, where transductive inference has emerged as an appealing approach to tackle few-shot tasks (Dhillon *et al.*, 2020; Hou *et al.*, 2019; Kim *et al.*, 2019; Nichol, Achiam & Schulman, 2018; Qiao *et al.*, 2019; Yanbin *et al.*, 2019), showing performance improvements over *inductive* inference. In few-shot learning, (Nichol *et al.*, 2018) used information of unlabeled query samples via batch normalization. TPN (Yanbin *et al.*, 2019) utilizes popular label-propagation ideas (Zhu & Ghahramani, 2002), along with episodic training. However, there is an inherent matrix inversion overhead for this method which requires solving system of linear equations having computational overhead that is cubic or quadratic with better techniques (e.g. CW-like algorithms) with respect to the number of query samples. This may be an impediment for deployment for large-scale few-shot tasks. Also label propagation typically utilizes power method to enhance the speed, however, it does not output the same labeling results as the the optimal solution (Fujiwara & Irie, 2014). CAN-T (Hou *et al.*, 2019) is a meta-learning based transductive method, which use attention mechanisms to propagate labels to unlabeled query samples. The transductive fine-tuning by (Dhillon *et al.*, 2020), discussed above, updates the network parameters with an entropy loss, defined over the unlabeled query samples of a few-shot task, thereby encouraging peaked posteriors (i.e., confident predicitions). The performance of (Dhillon *et al.*, 2020) is in line with established results in the context of semi-supervised learning, where entropy minimization is widely used (Grandvalet & Bengio, 2005; Miyato, Maeda, Koyama & Ishii, 2018). However, due to the need of gradient updates in fine-tuning, this method is very slow during inference, which is almost 300 times slower than simple inductive approaches, such as (Snell *et al.*, 2017; Vinyals *et al.*, 2016; Wang *et al.*, 2019).

CHAPTER 2

SCALABLE LAPLACIAN K-MODES

Imtiaz Masud Ziko [a], Eric Granger [b], Ismail Ben Ayed [c]

[a, b, c] Department of Systems Engineering, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article published in «Neural Information Processing Systems», vol. 31: pp. 10041–10051, 2018.

Abstract

We advocate Laplacian K-modes for joint clustering and density mode finding, and propose a concave-convex relaxation of the problem, which yields a parallel algorithm that scales up to large datasets and high dimensions. We optimize a tight bound (auxiliary function) of our relaxation, which, at each iteration, amounts to computing an independent update for each cluster-assignment variable, with guaranteed convergence. Therefore, our bound optimizer can be trivially distributed for large-scale data sets. Furthermore, we show that the density modes can be obtained as byproducts of the assignment variables via simple maximum-value operations whose additional computational cost is linear in the number of data points. Our formulation does not need storing a full affinity matrix and computing its eigenvalue decomposition, neither does it perform expensive projection steps and Lagrangian-dual inner iterates for the simplex constraints of each point. Furthermore, unlike mean-shift, our density-mode estimation does not require inner-loop gradient-ascent iterates. It has a complexity independent of feature-space dimension, yields modes that are valid data points in the input set and is applicable to discrete domains as well as arbitrary kernels. We report comprehensive experiments over various data sets, which show that our algorithm yields very competitive performances in term of optimization quality (i.e., the value of the discrete-variable objective at convergence) and clustering accuracy.

2.1 Introduction

We advocate Laplacian K-modes for joint clustering and density mode finding, and propose a concave-convex relaxation of the problem, which yields a parallel algorithm that scales up to large data sets and high dimensions. Introduced initially in the work of Wang and Carreira-Perpinán (Wang & Carreira-Perpinán, 2014), the model solves the following constrained optimization problem for K clusters and data points $\mathbf{X} = \{\mathbf{x}_p \in \mathbb{R}^M, p = 1, \ldots, N\}$:

$$\begin{aligned}
\min_{\mathbf{S}} \quad & \left\{ \mathcal{E}(\mathbf{S}) := -\sum_{p=1}^{N}\sum_{k=1}^{K} s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k) + \frac{\lambda}{2}\sum_{p,q} w(\mathbf{x}_p, \mathbf{x}_q)\|\mathbf{s}_p - \mathbf{s}_q\|^2 \right\} \\
& \mathbf{m}_k = \arg\max_{\mathbf{y}\in\mathbf{X}} \sum_{p} s_{p,k} w(\mathbf{x}_p, \mathbf{y}) \\
& \mathbf{1}^t\mathbf{s}_p = 1, \ \mathbf{s}_p \in \{0,1\}^K \ \forall p
\end{aligned} \tag{2.1}$$

where, for each point p, $\mathbf{s}_p = [s_{p,1}, \ldots, s_{p,K}]^t$ denotes a binary assignment vector, which is constrained to be within the K-dimensional simplex: $s_{p,k} = 1$ if p belongs to cluster k and $s_{p,k} = 0$ otherwise. $\mathbf{S}$ is the $N \times K$ matrix whose rows are given by the $\mathbf{s}_p$'s. $w(\mathbf{x}_p, \mathbf{x}_q)$ are pairwise affinities, which can be either learned or evaluated in an unsupervised way via a kernel function.

Model (2.1) integrates several powerful and well-known ideas in clustering. First, it identifies density modes (Comaniciu & Meer, 2002; Li, Ray & Lindsay, 2007), as in popular mean-shift. Prototype $\mathbf{m}_k$ is a cluster mode and, therefore, a valid data point in the input set. This is important for manifold-structured, high-dimensional inputs such as images, where simple parametric prototypes such as the means, as in K-means, may not be good representatives of the data; see Fig. 2.1. Second, the pairwise term in $\mathcal{E}$ is the well-known graph Laplacian regularizer, which can be equivalently written as $\mathrm{tr}(\mathbf{S}^t\mathbf{L}\mathbf{S})$, with $\mathbf{L}$ the Laplacian matrix corresponding to affinity matrix $\mathbf{W} = [w(\mathbf{x}_p, \mathbf{x}_q)]$. Laplacian regularization encourages nearby data points to have similar latent representations (e.g., assignments) and is widely used in spectral clustering (Shaham *et al.*, 2018; Von Luxburg, 2007b) as well as in semi-supervised and/or representation learning (Belkin *et al.*, 2006). Therefore, the model can handle non-convex (or manifold-structured) clusters, unlike standard prototype-based clustering techniques such as K-means. Finally, the explicit cluster assignments yield straightforward out-of-sample extensions, unlike spectral clustering (Bengio, Paiement, Vincent, Delalleau, Roux & Ouimet, 2004).

Optimization problem (2.1) is challenging due to the simplex/integer constraints and the non-linear/non-differentiable dependence of modes $\mathbf{m}_k$ on assignment variables. In fact, it is well known that optimizing the pairwise Laplacian term over discrete variables is NP-hard (Tian *et al.*, 2014), and it is common to relax the integer constraint. For instance, (Wang & Carreira-Perpinán, 2014) replaces the integer constraint with a probability-simplex constraint, which results in a convex relaxation of the Laplacian term. Unfortunately, such a direct convex relaxation requires solving for $N \times K$ variables all together. Furthermore, it requires additional projections onto the K-dimensional simplex, with a quadratic complexity with respect to K. Therefore, as we will see in our experiments, the relaxation in (Wang & Carreira-Perpinán, 2014) does not scale up for large-scale problems (i.e., when N and K are large). Spectral relaxation (Shi & Malik, 2000; Von Luxburg, 2007b) widely dominates optimization of the Laplacian term subject to balancing constraints in the context of graph clustering[1]. It can be expressed in the form of a generalized Rayleigh quotient, which yields an exact closed-form solution in terms of the K largest eigenvectors of the affinity matrix. It is well-known that spectral relaxation has high computational and memory load for large N as one has to store the $N \times N$ affinity matrix and compute explicitly its eigenvalue decomposition, which has a complexity that is cubic with respect to N for a straightforward implementation and, to our knowledge, super-quadratic for fast implementations (Tian *et al.*, 2014). In fact, investigating the scalability of spectral relaxation for large-scale problems is an active research subject (Shaham *et al.*, 2018; Tian *et al.*, 2014; Vladymyrov & Carreira-Perpiñán, 2016). For instance, the studies in (Shaham *et al.*, 2018; Tian *et al.*, 2014) investigated deep learning approaches to spectral clustering, so as to ease the scalability issues for large data sets, and the authors of (Vladymyrov & Carreira-Perpiñán, 2016) examined the variational Nyström method for large-scale spectral problems, among many other efforts on the subject. In general, computational scalability is attracting significant research

[1] Note that spectral relaxation is not directly applicable to the objective in (2.1) because of the presence of the K-mode term.

interest with the overwhelming widespread of interesting large-scale problems (Gong *et al.*, 2015). Such issues are being actively investigated even for the basic K-means algorithm (Gong *et al.*, 2015; Newling & Fleuret, 2016).

The K-modes term in (2.1) is closely related to kernel density based algorithms for mode estimation and clustering, for instance, the very popular mean-shift (Comaniciu & Meer, 2002). The value of $\mathbf{m}_k$ globally optimizing this term for a given fixed cluster k is, clearly, the mode of the kernel density of feature points within the cluster (Tang *et al.*, 2019b). Therefore, the K-mode term, as in (Carreira-Perpiñán & Wang, 2013; Salah, Ayed, Yuan & Zhang, 2014), can be viewed as an energy-based formulation of mean-shift algorithms with a fixed number of clusters (Tang *et al.*, 2019b). Optimizing the K-modes over discrete variable is NP-hard (Wang & Carreira-Perpinán, 2014), as is the case of other prototype-based models for clustering[2]. One way to tackle the problem is to alternate optimization over assignment variables and updates of the modes, with the latter performed as inner-loop mean-shift iterates, as in (Carreira-Perpiñán & Wang, 2013; Salah *et al.*, 2014). Mean-shift moves an initial random feature point towards the closest mode via gradient ascent iterates, maximizing at convergence the density of feature points. While such a gradient-ascent approach has been very popular for low-dimensional distributions over continuous domains, e.g., image segmentation (Comaniciu & Meer, 2002), its use is generally avoided in the context of high-dimensional feature spaces (Chen *et al.*, 2014). Mean-shift iterates compute expensive summations over feature points, with a complexity that depends on the dimension of the feature space. Furthermore, the method is not applicable to discrete domains (Chen *et al.*, 2014) (as it requires gradient-ascent steps), and its convergence is guaranteed only when the kernels satisfy certain conditions; see (Comaniciu & Meer, 2002). Finally, the modes obtained at gradient-ascent convergence are not necessarily valid data points in the input set.

We optimize a tight bound (auxiliary function) of our concave-convex relaxation for discrete problem (2.1). The bound is the sum of independent functions, each corresponding to a data point p. This yields a scalable algorithm for large N, which computes independent updates for assignment variables $\mathbf{s}_p$, while guaranteeing convergence to a minimum of the relaxation.

[2] In fact, even the basic K-means problem is NP-hard.

Therefore, our bound optimizer can be trivially distributed for large-scale data sets. Furthermore, we show that the density modes can be obtained as byproducts of assignment variables $\mathbf{s}_p$ via simple maximum-value operations whose additional computational cost is linear in N. Our formulation does not need storing a full affinity matrix and computing its eigenvalue decomposition, neither does it perform expensive projection steps and Lagrangian-dual inner iterates for the simplex constraints of each point. Furthermore, unlike mean-shift, our density-mode estimation does not require inner-loop gradient-ascent iterates. It has a complexity independent of feature-space dimension, yields modes that are valid data points in the input set and is applicable to discrete domains and arbitrary kernels. We report comprehensive experiments over various data sets, which show that our algorithm yields very competitive performances in term of optimization quality (i.e., the value of the discrete-variable objective at convergence)[3] and clustering accuracy, while being scalable to large-scale and high-dimensional problems.

2.2 Concave-convex relaxation

We propose the following concave-convex relaxation of the objective in (2.1):

$$\min_{\mathbf{s}_p \in \nabla_K} \left\{ \mathcal{R}(\mathbf{S}) := \sum_{p=1}^{N} \mathbf{s}_p^t \log(\mathbf{s}_p) - \sum_{p=1}^{N} \sum_{k=1}^{K} s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k) - \lambda \sum_{p,q} w(\mathbf{x}_p, \mathbf{x}_q) \mathbf{s}_p^t \mathbf{s}_q \right\} \tag{2.2}$$

where ∇_K denotes the K-dimensional probability simplex $\nabla_K = \{\mathbf{y} \in [0,1]^K \mid \mathbf{1}^t\mathbf{y} = 1\}$. It is easy to check that, at the vertices of the simplex, our relaxation in (2.2) is equivalent to the initial discrete objective in (2.1). Notice that, for binary assignment variables $\mathbf{s}_p \in \{0,1\}^K$, the first term in (2.2) vanishes and the last term is equivalent to Laplacian regularization, up to an additive constant:

$$\mathrm{tr}(\mathbf{S}^t\mathbf{L}\mathbf{S}) = \sum_{p} \mathbf{s}_p^t \mathbf{s}_p D_p - \sum_{p,q} w(\mathbf{x}_p, \mathbf{x}_q) \mathbf{s}_p^t \mathbf{s}_q = \sum_{p} D_p - \sum_{p,q} w(\mathbf{x}_p, \mathbf{x}_q) \mathbf{s}_p^t \mathbf{s}_q, \tag{2.3}$$

[3] We obtained consistently lower values of function $\mathcal{E}$ at convergence than the convex-relaxation proximal algorithm in (Wang & Carreira-Perpinán, 2014).

where the last equality is valid only for binary (integer) variables and $D_p = \sum_q w(\mathbf{x}_p, \mathbf{x}_q)$. When we replace the integer constraints $\mathbf{s}_p \in \{0, 1\}$ by $\mathbf{s}_p \in [0, 1]$, our relaxation becomes different from direct convex relaxations of the Laplacian (Wang & Carreira-Perpinán, 2014), which optimizes $\mathrm{tr}(\mathbf{S}^t\mathbf{L}\mathbf{S})$ subject to probabilistic simplex constraints. In fact, unlike $\mathrm{tr}(\mathbf{S}^t\mathbf{L}\mathbf{S})$, which is a convex function[4], our relaxation of the Laplacian term is concave for positive semi-definite (psd) kernels w. As we will see later, concavity yields a scalable (parallel) algorithm for large N, which computes independent updates for assignment variables $\mathbf{s}_p$. Our updates can be trivially distributed, and do not require storing a full $N \times N$ affinity matrix. These are important computational and memory advantages over direct convex relaxations of the Laplacian (Wang & Carreira-Perpinán, 2014), which require solving for $N \times K$ variables all together as well as expensive simplex projections, and over common spectral relaxations (Von Luxburg, 2007b), which require storing a full affinity matrix and computing its eigenvalue decomposition. Furthermore, the first term we introduced in (2.2) is a convex negative-entropy barrier function, which completely avoids expensive projection steps and Lagrangian-dual inner iterations for the simplex constraints of each point. First, the entropy barrier restricts the domain of each $\mathbf{s}_p$ to non-negative values, which avoids extra dual variables for constraints $\mathbf{s}_p \geq 0$. Second, the presence of such a barrier function yields closed-form updates for the dual variables of constraints $\mathbf{1}^t\mathbf{s}_p = 1$. In fact, entropy-like barriers are commonly used in Bregman-proximal optimization (Yuan, Yin, Bai, Feng & Tai, 2017), and have well-known computational and memory advantages when dealing with the challenging simplex constraints (Yuan *et al.*, 2017). Surprisingly, to our knowledge, they are not common in the context of clustering. In machine learning, such entropy barriers appear frequently in the context of conditional random fields (CRFs) (Krähenbühl & Koltun, 2011; Krähenbühl & Koltun, 2013), but are not motivated from optimization perspective; they result from standard probabilistic and mean-field approximations of CRFs (Krähenbühl & Koltun, 2011).

[4] For relaxed variables, $\mathrm{tr}(\mathbf{S}^t\mathbf{L}\mathbf{S})$ is a convex function because the Laplacian is always positive semi-definite.

2.3 Bound optimization

In this section, we derive an iterative bound optimization algorithm that computes independent (parallel) updates of assignment variables $\mathbf{s}_p$ ($\mathbf{s}$-updates) at each iteration, and provably converges to a minimum of relaxation (2.2). As we will see in our experiments, our bound optimizer yields consistently lower values of function $\mathcal{E}$ at convergence than the proximal algorithm in (Wang & Carreira-Perpinán, 2014), while being highly scalable to large-scale and high-dimensional problems. We also show that the density modes can be obtained as byproducts of the $\mathbf{s}$-updates via simple maximum-value operations whose additional computational cost is linear in N. Instead of minimizing directly our relaxation $\mathcal{R}$, we iterate the minimization of an auxiliary function, i.e., an upper bound of $\mathcal{R}$, which is tight at the current solution and easier to optimize.

Definition 1. $\mathcal{A}_i(\mathbf{S})$ *is an auxiliary function of* $\mathcal{R}(\mathbf{S})$ *at current solution* $\mathbf{S}^i$ *if it satisfies:*

$$\mathcal{R}(\mathbf{S}) \leq \mathcal{A}_i(\mathbf{S}), \forall \mathbf{S} \tag{2.4a}$$

$$\mathcal{R}(\mathbf{S}^i) = \mathcal{A}_i(\mathbf{S}^i) \tag{2.4b}$$

In (2.4), i denotes the iteration counter. In general, bound optimizers update the current solution $\mathbf{S}^i$ to the optimum of the auxiliary function: $\mathbf{S}^{i+1} = \arg\min_{\mathbf{S}} \mathcal{A}_i(\mathbf{S})$. This guarantees that the original objective function does not increase at each iteration: $\mathcal{R}(\mathbf{S}^{i+1}) \leq \mathcal{A}_i(\mathbf{S}^{i+1}) \leq \mathcal{A}_i(\mathbf{S}^i) = \mathcal{R}(\mathbf{S}^i)$. Bound optimizers can be very effective as they transform difficult problems into easier ones (Zhang, Kwok & Yeung, 2007). Examples of well-known bound optimizers include the concave-convex procedure (CCCP) (Yuille & Rangarajan, 2001), expectation maximization (EM) algorithms and submodular-supermodular procedures (SSP) (Narasimhan & Bilmes, 2005), among others. Furthermore, bound optimizers are not restricted to differentiable functions[5], neither do they depend on optimization parameters such as step sizes.

[5] Our objective is not differentiable with respect to the modes as each of these is defined as the maximum of a function of the assignment variables.

Proposition 1. *Given current solution* $\mathbf{S}^i = [s^i_{p,k}]$ *at iteration i, and the corresponding modes* $\mathbf{m}^i_k = \arg\max_{\mathbf{y}\in\mathbf{X}} \sum_p s^i_{p,k} w(\mathbf{x}_p, \mathbf{y})$*, we have the following auxiliary function (up to an additive constant) for the concave-convex relaxation in* (2.2) *and psd*[6] *affinity matrix* $\mathbf{W}$*:*

$$\mathcal{A}_i(\mathbf{S}) = \sum_{p=1}^{N} \mathbf{s}^t_p(\log(\mathbf{s}_p) - \mathbf{a}^i_p - \lambda \mathbf{b}^i_p) \tag{2.5}$$

where $\mathbf{a}^i_p$ *and* $\mathbf{b}^i_p$ *are the following K-dimensional vectors:*

$$\mathbf{a}^i_p = [a^i_{p,1}, \ldots, a^i_{p,K}]^t, \text{ with } a^i_{p,k} = w(\mathbf{x}_p, \mathbf{m}^i_k) \tag{2.6a}$$

$$\mathbf{b}^i_p = [b^i_{p,1}, \ldots, b^i_{p,K}]^t, \text{ with } b^i_{p,k} = \sum_q w(\mathbf{x}_p, \mathbf{x}_q) s^i_{q,k} \tag{2.6b}$$

Proof 1. *See Supplemental section 2.6.1.*

Notice that the bound in Eq. (2.5) is the sum of independent functions, each corresponding to a point p. Therefore, both the bound and simplex constraints $\mathbf{s}_p \in \nabla_K$ are separable over assignment variables $\mathbf{s}_p$. We can minimize the auxiliary function by minimizing independently each term in the sum over $\mathbf{s}_p$, subject to the simplex constraint, while guaranteeing convergence to a local minimum of (2.2):

$$\min_{\mathbf{s}_p \in \nabla_K} \mathbf{s}^t_p(\log(\mathbf{s}_p) - \mathbf{a}^i_p - \lambda \mathbf{b}^i_p), \ \forall p \tag{2.7}$$

Note that, for each p, negative entropy $\mathbf{s}^t_p \log \mathbf{s}_p$ restricts $\mathbf{s}_p$ to be non-negative, which removes the need for handling explicitly constraints $\mathbf{s}_p \geq 0$. This term is convex and, therefore, the problem in (2.7) is convex: The objective is convex (sum of linear and convex functions) and constraint $\mathbf{s}_p \in \nabla_K$ is affine. Therefore, one can minimize this constrained convex problem for each p by solving the Karush-Kuhn-Tucker (KKT) conditions[7]. The KKT conditions yield a closed-form

[6] We can consider $\mathbf{W}$ to be psd without loss of generality. When $\mathbf{W}$ is not psd, we can use a diagonal shift for the affinity matrix, i.e., we replace $\mathbf{W}$ by $\tilde{\mathbf{W}} = \mathbf{W} + \delta \mathbf{I}_N$. Clearly, $\tilde{\mathbf{W}}$ is psd for sufficiently large δ. For integer variables, this change does not alter the structure of the minimum of discrete function $\mathcal{E}$.

[7] Note that strong duality holds since the objectives are convex and the simplex constraints are affine. This means that the solutions of the (KKT) conditions minimize the auxiliary function.

solution for both primal variables $\mathbf{s}_p$ and the dual variables (Lagrange multipliers) corresponding to simplex constraints $\mathbf{1}^t\mathbf{s}_p = 1$. Each closed-form update, which globally optimizes (2.7) and is within the simplex, is given by:

$$\mathbf{s}_p^{i+1} = \frac{\exp(\mathbf{a}_p^i + \lambda \mathbf{b}_p^i)}{\mathbf{1}^t \exp(\mathbf{a}_p^i + \lambda \mathbf{b}_p^i)} \; \forall p \tag{2.8}$$

Algorithm 2.1 SLK algorithm

```
**Input:** $\mathbf{X}$, Initial seeds
**Output:** $\mathbf{S}$ and modes $\{\mathbf{m}_k\}_{k=1}^K$ and clustering $labels \in \{1, .., K\}^N$
Initialize $i = 1$.
Initialize modes $\{\mathbf{m}_k\}_{k=1}^K$ from initial seeds.
Initialize $labels$ from initial seeds.
Initialize $\mathbf{S}$ from $labels$.
**repeat**
   $\{\mathbf{m}_l^i\}_{k=1}^K \leftarrow \{\mathbf{m}_k\}_{k=1}^K$
   Compute $\mathbf{a}_p^i$ from $\mathbf{S}$ from (2.6a).
   Initialize $\mathbf{s}_p^i = \frac{\exp(\mathbf{a}_p^i)}{\mathbf{1}^t \exp(\mathbf{a}_p^i)}$.
   **repeat**
      Compute $\mathbf{s}_p^{i+1}$ using (2.8).
      $\mathbf{s}_p^i \leftarrow \mathbf{s}_p^{i+1}$.
   **until** $\mathcal{A}_i(\mathbf{S})$ in (2.5) does not change
   $\mathbf{S} = [\mathbf{s}_p^i]; \; \forall p$.
   $i = i + 1$.
   **if** SLK-MS **then**
      update $\mathbf{m}_k$ using (2.9) until converges
   **else if** SLK-BO **then**
      $\mathbf{m}_k \leftarrow \arg\max_{\mathbf{x}_p} [s_{p,k}^i]$
   **end if**
**until** $\mathcal{E}(\mathbf{S})$ in (2.1) does not change
$l_p = \arg\max_k \mathbf{s}_p; \forall p$.
$labels = \{l_p\}_{p=1}^N$.
```

The pseudo-code for our Scalable Laplacian K-modes (SLK) method is provided in Algorithm 2.1. The complexity of each inner iteration in $\mathbf{s}$-updates is $\mathcal{O}(N\rho K)$, with ρ the neighborhood size for the affinity matrix. Typically, we use sparse matrices ($\rho << N$). Note that the complexity becomes $O(N^2K)$ in the case of dense matrices in which all the affinities are non-zero. However, the update of each $\mathbf{s}_p$ can be done independently, which enables parallel implementations.

Our SLK algorithm alternates the following two steps until convergence (i.e. until the modes $\{\mathbf{m}_k\}_{k=1}^K$ do not change):

1. **s**-*updates*: update cluster assignments using expression (2.8) with the modes fixed and
2. *Mode-updates*: update the modes $\{\mathbf{m}_k\}_{k=1}^K$ with the assignment variable **S** fixed; see the next section for further details on mode estimation.

2.3.1 Mode updates

To update the modes, we utilize two options: modes via mean-shift or as byproducts of the **s**-updates.

Modes via mean-shift: This amounts to updating each mode $\mathbf{m}_k$ by running inner-loop mean-shift iterations until convergence, using the current assignment variables:

$$\mathbf{m}_k = \frac{\sum_p s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k)\mathbf{x}_p}{\sum_p s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k)} \tag{2.9}$$

Modes as byproducts of the **s**-*updates:* We also propose an efficient alternative to mean-shift. Observe the following: For each point p, $b^i_{p,k} = \sum_q w(\mathbf{x}_p, \mathbf{x}_q) s^i_{q,k}$ is proportional to the kernel density estimate (KDE) of the distribution of features within current cluster k at point p. In fact, the KDE at a feature point $\mathbf{y}$ is:

$$\mathcal{P}^i_k(\mathbf{y}) = \frac{\sum_q w(\mathbf{y}, \mathbf{x}_q) s^i_{q,k}}{\sum_q s^i_{q,k}}.$$

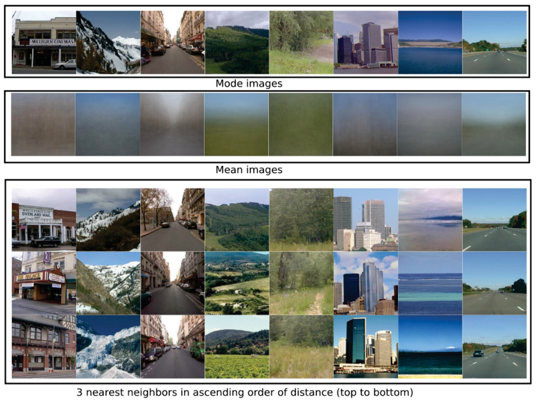
Therefore, $b^i_{p,k} \propto \mathcal{P}^i_k(\mathbf{x}_p)$. As a result, for a given point p within the cluster, the higher $b^i_{p,k}$, the higher the KDE of the cluster at that point. Notice also that $a^i_{p,k} = w(\mathbf{x}_p, \mathbf{m}^i_k)$ measures a proximity between point $\mathbf{x}_p$ and the mode obtained at the previous iteration. Therefore, given the current assignment $\mathbf{s}_p^i$, the modes can be obtained as a proximal optimization, which seeks a high-density data point that does not deviate significantly from the mode obtained at the previous iteration:

$$\max_{\mathbf{y}\in\mathbf{X}} [\underbrace{w(\mathbf{y}, \mathbf{m}_k^i)}_{\text{proximity}} + \underbrace{\sum_p s_{p,k} w(\mathbf{x}_p, \mathbf{y})}_{\text{density}}] \tag{2.10}$$

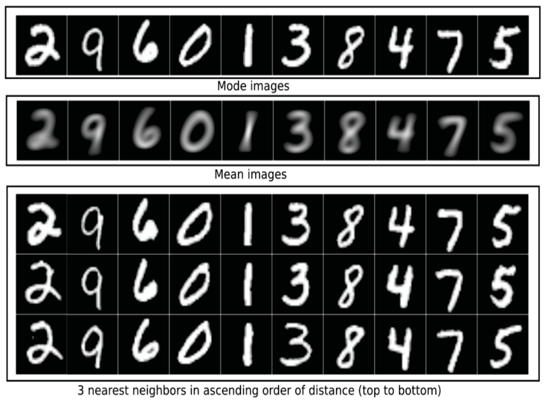
Now observe that the **s**-updates we obtained in Eq. (2.8) take the form of *softmax* functions. Therefore, they can be used as soft approximations of the hard max operation in Eq. (2.10):

$$\mathbf{m}_k^{i+1} = \mathbf{x}_p, \text{ with } p = \arg\max_q [s_{q,k}]^i \tag{2.11}$$

This yields modes as byproducts of the **s**-updates, with a computational cost that is linear in N. We refer to the two different versions of our algorithm as SLK-MS, which updates the modes via mean-shift, and SLK-BO, which updates the modes as byproducts of the **s**-updates.


Mode images
Mean images
3 nearest neighbors in ascending order of distance (top to bottom)

a) LabelMe modes


Mode images
Mean images
3 nearest neighbors in ascending order of distance (top to bottom)

b) MNIST Modes

Figure 2.1 Examples of mode images obtained with our SLK-BO, mean images and the corresponding 3-nearest-neighbor to the mode images within each cluster.

2.4 Experiments

We report comprehensive evaluations of the proposed algorithm[8] as well as comparisons to the following related baseline methods: Laplacian K-modes (LK) (Wang & Carreira-Perpinán, 2014), K-means, NCUT (Shi & Malik, 2000), K-modes (Carreira-Perpiñán, 2015; Salah *et al.*, 2014), Kernel K-means (KK-means) (Dhillon *et al.*, 2004; Tang *et al.*, 2019b) and Spectralnet (Shaham *et al.*, 2018). Our algorithm is evaluated in terms of performance and optimization quality in various clustering datasets.

Table 2.1 Datasets used in the experiments.

Datasets	Samples (N)	Dimensions (M)	Clusters (K)	Imbalance
MNIST (small)	2,000	784	10	1
MNIST (code)	70,000	10	10	~ 1
MNIST	70,000	784	10	~ 1
MNIST (GAN)	70,000	256	10	~ 1
Shuttle	58,000	9	7	$4,558$
LabelMe (Alexnet)	2,688	4,096	8	1
LabelMe (GIST)	2,688	44,604	8	1
YTF	10,036	9,075	40	13
Reuters (code)	685,071	10	4	~ 5

2.4.1 Datasets and evaluation metrics

We used image datasets, except Shuttle and Reuters. The overall summary of the datasets is given in Table 2.1. For each dataset, imbalance is defined as the ratio of the size of the biggest cluster to the size of the smallest one. We use three versions of MNIST (LeCun, Bottou, Bengio & Haffner, 1998). MNIST contains all the $70,000$ images, whereas MNIST (small) includes only $2,000$ images by randomly sampling 200 images per class. We used small datasets in order to compare to LK (Wang & Carreira-Perpinán, 2014), which does not scale up for large datasets. For MNIST (GAN), we train the GAN from (Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville & Bengio, 2014) on $60,000$ training images and extract the 256-dimensional features

[8] Code is available at: https://github.com/imtiazziko/SLK

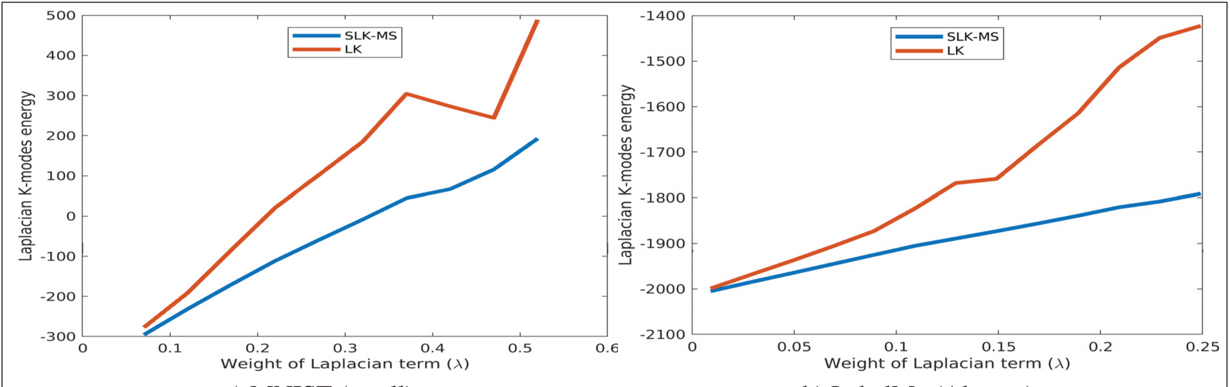


Figure 2.2 Discrete-variable objective (2.1): Comparison of the objectives obtained at convergence for SLK-MS (ours) and LK (Wang & Carreira-Perpinán, 2014). The objectives at convergence are plotted versus different values of parameter λ.

from the discriminator network for the 70, 000 images. The publicly available autoencoder in (Jiang, Zheng, Tan, Tang & Zhou, 2017) is used to extract 10-dimensional features as in (Shaham *et al.*, 2018) for MNIST (code) and Reuters (code). LabelMe (Oliva & Torralba, 2001) consists of 2, 688 images divided into 8 categories. We used the pre-trained AlexNet (Krizhevsky, Sutskever & Hinton, 2012) and extracted the 4096-dimensional features from the fully-connected layer. To show the performances on high-dimensional data, we extract 44604-dimensional GIST features (Oliva & Torralba, 2001) for the LabelMe dataset. Youtube Faces (YTF) (Wolf, Hassner & Maoz, 2011) consists of videos of faces with 40 different subjects.

To evaluate the clustering performance, we use two well adopted measures: Normalized Mutual Information (NMI) (Strehl & Ghosh, 2002) and Clustering Accuracy (ACC) (Ghasedi Dizaji, Herandi, Deng, Cai & Huang, 2017; Shaham *et al.*, 2018). The optimal mapping of clustering assignments to the true labels are determined using the Kuhn-Munkres algorithm (Munkres, 1957).

2.4.2 Implementation details

We built kNN affinities as follows: $w(\mathbf{x}_p, \mathbf{x}_q) = 1$ if $\mathbf{x}_q \in \mathcal{N}_p^{k_n}$ and $w(\mathbf{x}_p, \mathbf{x}_q) = 0$ otherwise, where $\mathcal{N}_p^{k_n}$ is the set of the k_n nearest neighbors of data point $\mathbf{x}_p$. This yields a sparse affinity matrix, which is efficient in terms of memory and computations. In all of the datasets, we fixed $k_n = 5$. For the large datasets such as MNIST, Shuttle and Reuters, we used the *Flann* library (Muja & Lowe, 2014) with the KD-tree algorithm, which finds approximate nearest neighbors. For the other smaller datasets, we used an efficient implementation of exact nearest-neighbor computations. We used the Euclidean distance for finding the nearest neighbors. We used the same sparse $\mathbf{K}$ for the pairwise-affinity algorithms we compared with, i.e., NCUT, KK-means, Laplacian K-modes. Furthermore, for each of these baseline methods, we evaluated the default setting of affinity construction with tuned σ, and report the best result found. Mode estimation is based on the Gaussian kernel $w(\mathbf{x}, \mathbf{y}) = e^{-(\|\mathbf{x}-\mathbf{y})\|^2/2\sigma^2)}$, with σ^2 estimated as: $\sigma^2 = \frac{1}{Nk_n} \sum_{\mathbf{x}_p \in \mathbf{X}} \sum_{\mathbf{x}_q \in \mathcal{N}_p^{k_n}} \|\mathbf{x}_p - \mathbf{x}_q\|^2$. Initial centers $\{\mathbf{m}_l^0\}_{k=1}^K$ are based on K-means++ seeds (Arthur & Vassilvitskii, 2007). We choose the best initial seed and regularization parameter λ empirically based on the accuracy over a validation set (10% of the total data). The λ is determined from tuning in a small range from 1 to 4. In SLK-BO, we take the starting mode $\mathbf{m}_k$ for each cluster from the initial assignments by simply following the mode definition in (2.1). In Algorithm 2.1, all assignment variables $\mathbf{s}_p$ are updated in parallel. We run the publicly released codes for K-means (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot & Duchesnay, 2011), NCUT (Shi & Malik, 2000), Laplacian K-modes (Carreira-Perpiñán, 2007), Kernel K-means[9] and Spectralnet (Shaham *et al.*, 2018).

2.4.3 Clustering results

Table 2.2 reports the clustering results, showing that, in most of the cases, our algorithms SLK-MS and SLK-BO yielded the best NMI and ACC values. For MNIST with the raw intensities as features, the proposed SLK achieved almost 80% NMI and ACC. With better learned features for MNIST (code) and MNIST (GAN), the accuracy (ACC) increases up to 95%. For the MNIST (code) and Reuters (code) datasets, we used the same features and Euclidean distance based affinity as Spectralnet (Shaham *et al.*, 2018), and obtained better NMI/ACC

[9] https://gist.github.com/mblondel/6230787

Table 2.2 Clustering results as NMI/ACC in the upper half and average elapsed time in seconds (s). (*) We report the results of Spectralnet for MNIST (code) and Reuters (code) from (Shaham *et al.*, 2018).

Algorithm	MNIST	MNIST (code)	MNIST (GAN)	LabelMe (Alexnet)	LabelMe (GIST)	YTF	Shuttle	Reuters
K-means	0.53/0.55	0.66/0.74	0.68/0.75	0.81/0.90	0.57/0.69	0.77/0.58	0.22/0.41	0.48/0.73
K-modes	0.56/0.60	0.67/0.75	0.69/0.80	0.81/0.91	0.58/0.68	0.79/0.62	0.33/0.47	0.48/0.72
NCUT	0.74/0.61	0.84/0.81	0.77/0.67	0.81/0.91	0.58/0.61	0.74/0.54	0.47/0.46	-
KK-means	0.53/0.55	0.67/0.80	0.69/0.68	0.81/0.90	0.57/0.63	0.71/0.50	0.26/0.40	-
LK	-	-	-	0.81/0.91	0.59/0.61	0.77/0.59	-	-
Spectralnet*	-	0.81/0.80	-	-	-	-	-	0.46/0.65
SLK-MS	0.80/0.79	0.88/0.95	0.86/0.94	0.83/0.91	0.61/0.72	0.82/0.65	0.45/0.70	0.43/0.74
SLK-BO	0.77/0.80	0.89/0.95	0.86/0.94	0.83/0.91	0.61/0.72	0.80/0.64	0.51/0.71	0.43/0.74
K-means	119.9s	16.8s	51.6s	11.2s	132.1s	210.1s	1.8s	36.1s
K-modes	90.2s	20.2s	20.3s	7.4s	12.4s	61.0s	0.5s	51.6s
NCUT	26.4s	28.2s	9.3s	7.4s	10.4s	19.0s	27.4s	-
KK-means	2580.8s	1967.9s	2427.9s	4.6s	17.2s	40.2s	1177.6s	-
LK	-	-	-	33.4s	180.9s	409.0s	-	-
Spectralnet*	-	3600.0s	-	-	-	-	-	9000.0s
SLK-MS	101.2s	82.4s	37.3s	4.7s	37.0s	83.3s	3.8s	12.5s
SLK-BO	14.2s	23.1s	10.3s	1.8s	7.1s	12.4s	1.3s	53.1s

performances. The Shuttle dataset is quite imbalanced and, therefore, all the baseline clustering methods fail to achieve high accuracy. Notice that, in regard to ACC for the Shuttle dataset, we outperformed all the methods by a large margin.

One advantage of our SLK-BO over standard prototype-based models is that the modes are valid data points in the input set. This is important for manifold-structured, high-dimensional inputs such as images, where simple parametric prototypes such as the means, as in K-means, may not be good representatives of the data; see Fig. 2.1.

Table 2.3 Discrete-variable objectives at convergence for LK (Wang & Carreira-Perpinán, 2014) and SLK-MS (ours).

Datasets	LK (Wang & Carreira-Perpinán, 2014)	SLK-MS (ours)
MNIST (small)	273.25	67.09
LabelMe (Alexnet)	$-1.513\,84 \times 10^3$	$-1.807\,77 \times 10^3$
LabelMe (GIST)	$-1.954\,90 \times 10^3$	$-2.024\,10 \times 10^3$
YTF	$-1.000\,32 \times 10^4$	$-1.000\,35 \times 10^4$

2.4.4 Comparison in terms of optimization quality

To assess the optimization quality of our optimizer, we computed the values of discrete-variable objective $\mathcal{E}$ in model (2.1) at convergence for our concave-convex relaxation (SLK-MS) as well as for the convex relaxation in (Wang & Carreira-Perpinán, 2014) (LK). We compare the discrete-variable objectives for different values of λ. For a fair comparison, we use the same initialization, σ, $w(\mathbf{x}_p, \mathbf{x}_q)$, λ and mean-shift modes for both methods. As shown in the plots in Figure 2.2, our relaxation consistently obtained lower values of discrete-variable objective $\mathcal{E}$ at convergence than the convex relaxation in (Wang & Carreira-Perpinán, 2014). Also, Table 2.3 reports the discrete-variable objectives at convergence for LK (Wang & Carreira-Perpinán, 2014) and SLK-MS (ours). These experiments suggest that our relaxation in Eq. (2.2) is tighter than the convex relaxation in (Wang & Carreira-Perpinán, 2014). In fact, Eq. (2.3) also suggests that our relaxation of the Laplacian term is tighter than a direct convex relaxation (the expression in the middle in Eq. (2.3)) as the variables in term $\sum_p D_p \mathbf{s}_p^t \mathbf{s}_p$ are not relaxed in our case.

2.4.5 Running Time

The running times are given at the bottom half of Table 2.2. All the experiments (our methods and the baselines) were conducted on a machine with Xeon E5-2620 CPU and a Titan X Pascal GPU. We restrict the multiprocessing to at most 5 processes. We run each algorithm over 10 trials and report the average running time. For high-dimensional datasets, such as LabelMe (GIST) and YTF, our method is much faster than the other methods we compared to. It is also interesting to see that, for high dimensions, SLK-BO is faster than SLK-MS, which uses mean-shift for mode estimation.

2.5 Conclusion

We presented Scalable Laplacian K-modes (SLK), a method for joint clustering and density mode estimation, which scales up to high-dimensional and large-scale problems. We formulated a concave-convex relaxation of the discrete-variable objective, and solved the relaxation with an iterative bound optimization. Our solver results in independent updates for cluster-assignment variables, with guaranteed convergence, thereby enabling distributed implementations for large-scale data sets. Furthermore, we showed that the density modes can be estimated directly from the assignment variables using simple maximum-value operations, with an additional computational cost that is linear in the number of data points. Our solution removes the need for storing a full affinity matrix and computing its eigenvalue decomposition. Unlike the convex relaxation in (Wang & Carreira-Perpinán, 2014), it does not require expensive projection steps and Lagrangian-dual inner iterates for the simplex constraints of each point. Furthermore, unlike mean-shift, our density-mode estimation does not require inner-loop gradient-ascent iterates. It has a complexity independent of feature-space dimension, yields modes that are valid data points in the input set and is applicable to discrete domains as well as arbitrary kernels. We showed competitive performances of the proposed solution in term of optimization quality and accuracy. It will be interesting to investigate joint feature learning and SLK clustering.

2.6 Supplemental

2.6.1 Proof of Proposition 1

In this supplemental material, we give a detailed **proof of Proposition 1**. Recall that our concave-convex relaxation of the discrete Laplacian K-modes objective is:

$$\mathcal{R}(\mathbf{S}) = \sum_{p=1}^{N} \mathbf{s}_p^t \log(\mathbf{s}_p) - \sum_{p=1}^{N}\sum_{k=1}^{K} s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k) - \lambda \sum_{p,q} w(\mathbf{x}_p, \mathbf{x}_q)\mathbf{s}_p^t \mathbf{s}_q$$

The proposition states that, given current solution $\mathbf{S}^i = [s^i_{p,k}]$ at iteration i, and the corresponding modes $\mathbf{m}^i_k = \arg\max_{\mathbf{y}} \sum_p s^i_{p,k} w(\mathbf{x}_p, \mathbf{y})$, we have the following auxiliary function (up to an additive constant) for concave-convex relaxation (2.2) and PSD affinity matrix $\mathbf{W}$:

$$\mathcal{A}_i(\mathbf{S}) = \sum_{p=1}^{N} \mathbf{s}_p^t (\log(\mathbf{s}_p) - \mathbf{a}^i_p - \lambda \mathbf{b}^i_p)$$

where $\mathbf{a}_p^i$ and $\mathbf{b}_p^i$ are the following K-dimensional vectors:

$$\mathbf{a}_p^i = [a_{p,1}^i, \ldots, a_{p,K}^i]^t, \; with \; a_{p,k}^i = w(\mathbf{x}_p, \mathbf{m}_k^i)$$
$$\mathbf{b}_p^i = [b_{p,1}^i, \ldots, b_{p,K}^i]^t, \; with \; b_{p,k}^i = \sum_q w(\mathbf{x}_p, \mathbf{x}_q) s_{q,k}^i$$

Proof:

Instead of $N \times K$ matrix $\mathbf{S}$, let us represent our assignment variables with a vector $\mathbf{s} \in [0, 1]^{LN}$, which is of length K multiplied by N and takes the form $[\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N]$. As in the paper, each $\mathbf{s}_p$ is a vector of dimension K containing the probability variables of all labels for point p: $\mathbf{s}_p = [s_{p,1}, \ldots, s_{p,K}]^t$.

Let $\Psi = -\mathbf{W} \otimes \mathbf{I}_N$, where $\otimes$ denotes the Kronecker product and $\mathbf{I}_N$ the $N \times N$ identity matrix. Now, observe that we can write the relaxed Laplacian term in (2.2) in the following convenient form:

$$-\lambda \sum_{p,q} w(\mathbf{x}_p, \mathbf{x}_q) \mathbf{s}_p^t \mathbf{s}_q = \lambda \mathbf{s}^t \Psi \mathbf{s} \tag{2.12}$$

Notice that Kronecker product Ψ is negative semi-definite when $\mathbf{W}$ is positive semi-definite. In this case, function $\mathbf{s}^T \psi \mathbf{s}$ is concave and, therefore, is upper bounded by its first-order approximation at current solution $\mathbf{s}^i$ (iteration i). In fact, concavity arguments are standard in deriving auxiliary functions for bound-optimization algorithms (Lange, Hunter & Yang, 2000a). With this condition, we have the following auxiliary function for the Laplacian-term relaxation in (2.2):

$$-\sum_{p,q} w(\mathbf{x}_p, \mathbf{x}_q) \mathbf{s}_p^t \mathbf{s}_q \leq (\mathbf{s}^i)^t \Psi \mathbf{s}^i + (\Psi \mathbf{s}^i)^t (\mathbf{s} - \mathbf{s}^i) \tag{2.13}$$

Now, notice that, for each cluster k, the mode is by definition: $\mathbf{m}_k = \arg\max_{\mathbf{y} \in \in \mathbf{X}} \sum_p s_{p,k} w(\mathbf{x}_p, \mathbf{y})$. Therefore, $\forall \mathbf{y} \in \mathbf{X}$, we have $-\sum_{p=1}^N s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k) \leq -\sum_{p=1}^N s_{p,k} w(\mathbf{x}_p, \mathbf{y})$. Applying this result to $\mathbf{y} = \mathbf{m}_k^i$, we obtain the following auxiliary function on the K-mode term :

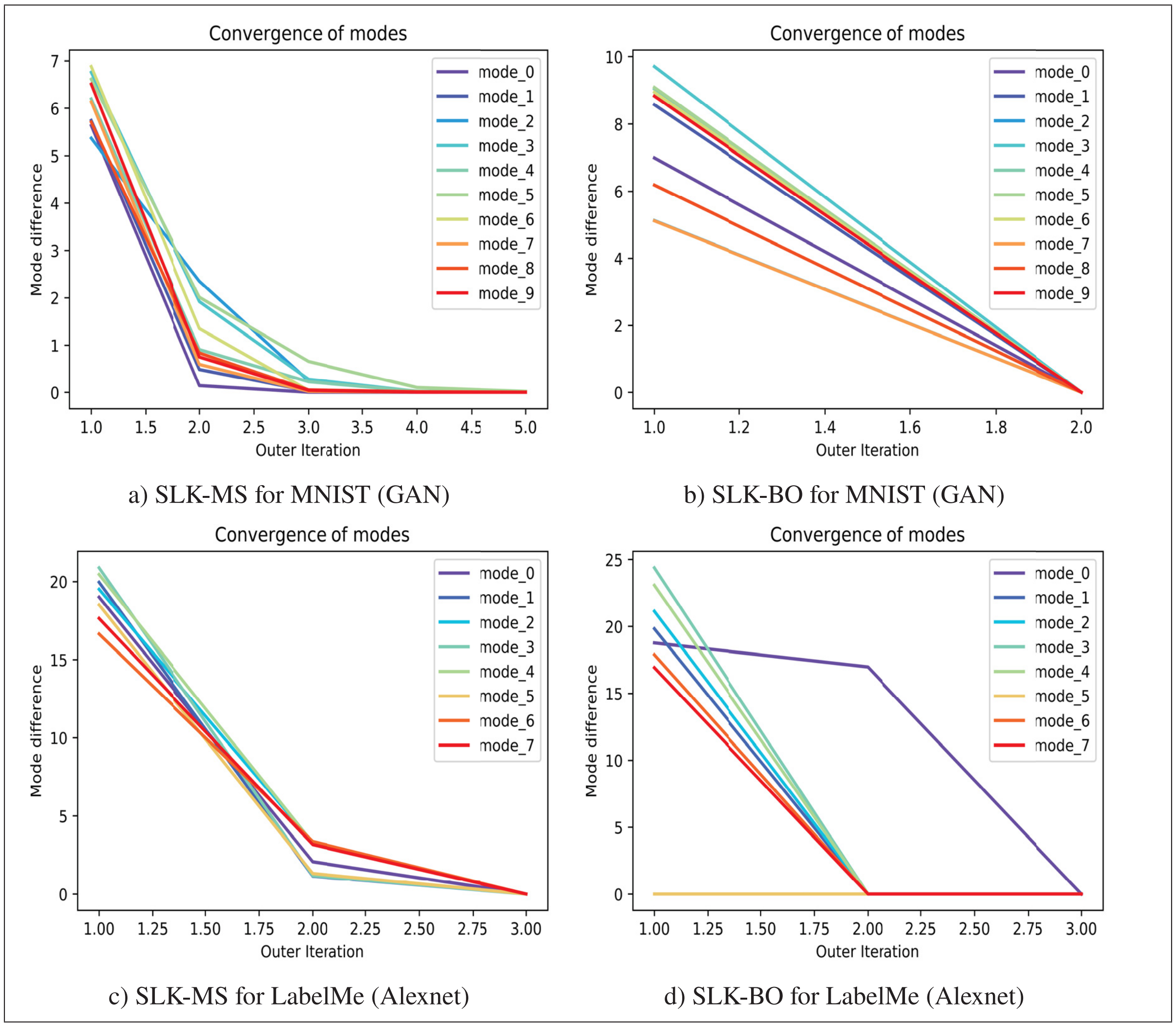
$$-\sum_{p=1}^{N} s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k) \leq -\sum_{p=1}^{N} s_{p,k} w(\mathbf{x}_p, \mathbf{m}_k^i) \tag{2.14}$$

Combining (2.13) and (2.14), it is easy to see that (2.5) is an upper bound on our concave-convex relaxation in (2.2), up to an additive constant[10]. It easy to check that both bounds in (2.13) and (2.14) are tight at the current solution. This complete the proof that (2.5) is an auxiliary function for our concave-convex relaxation, up to an additive constant.

2.6.2 Convergence of SLK

Figure 2.3 show the convergence of the outer iterations of SLK-BO and SLK-MS using MNIST (GAN) and LabelME (Alexnet) datasets. For each cluster, the convergence of the outer loop (mode updates) is shown as the difference in mode values within two consecutive outer iterations. Notice that both SLK-BO and SLK-MS converge within less than 5 outer iterations, with SLK-MS typically taking more outer iterations. This might be due to the fact that SLK-BO updates the modes from valid data points within the input set, whereas SLK-MS updates the modes as local means via mean-shift iterations.

[10] The additive constant depends only on the $\mathbf{s}^i$'s, the assignment variables computed at the previous iteration. This additive constant is ignored in the expression of the auxiliary function in Eq. (2.5).



a) SLK-MS for MNIST (GAN)

b) SLK-BO for MNIST (GAN)

c) SLK-MS for LabelMe (Alexnet)

d) SLK-BO for LabelMe (Alexnet)

Figure 2.3 Convergence of the outer iterations (mode updates): For each cluster, the convergence of the outer loop is shown as the difference in mode values within two consecutive outer iterations. The plots are for MNIST (GAN) and LabelMe (Alexnet) datasets.

CHAPTER 3

VARIATIONAL FAIR CLUSTERING

Imtiaz Masud Ziko [a], Eric Granger [b], Jing Yuan [c], Ismail Ben Ayed [d]

[a, b, d] Department of Systems Engineering, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[c] School of Math. and Statistics, Xidian University, China

Article submitted in «Neural Information Processing Systems» 2020.

Abstract

We propose a general variational framework of fair clustering, which integrates an original Kullback-Leibler (KL) fairness term with a large class of clustering objectives, including prototype or graph based. Fundamentally different from the existing combinatorial and spectral solutions, our variational multi-term approach enables to control the trade-off levels between the fairness and clustering objectives. We derive a general tight upper bound based on a concave-convex decomposition of our fairness term, its Lipschitz-gradient property and the Pinsker inequality. Our tight upper bound can be jointly optimized with various clustering objectives, while yielding a scalable solution, with convergence guarantee. Interestingly, at each iteration, it performs an independent update for each assignment variable. Therefore, it can easily be distributed for large-scale datasets. This scalability is important as it enables to explore different trade-off levels between fairness and the clustering objective. Unlike spectral relaxation, our formulation does not require storing an affinity matrix and computing its eigenvalue decomposition. We report comprehensive evaluations and comparisons with state-of-the-art methods over various fair-clustering benchmarks, which show that our variational method can yield highly competitive solutions in terms of fairness and clustering objectives.

3.1 Introduction

Machine learning models are impacting our daily life, for instance, in marketing, finance, education, and even in sentencing recommendations (Kleinberg *et al.*, 2017). However, these models may exhibit biases towards specific demographic groups due to, for instance, the biases that exist within the data. For example, a higher level of face recognition accuracy may be found with white males (Buolamwini & Gebru, 2018), and a high probability of recidivism tends to be incorrectly predicted for low-risk African-Americans (Julia *et al.*, 2016). These biases have recently triggered substantial interest in designing fair algorithms for the supervised learning setting (Donini *et al.*, 2018; Hardt *et al.*, 2016; Zafar *et al.*, 2017). Also, very recently, the community started to investigate fairness constraints in unsupervised learning (Backurs *et al.*, 2019; Celis *et al.*, 2018; Chierichetti *et al.*, 2017; Kleindessner *et al.*, 2019; Samadi *et al.*, 2018). Specifically, Chierichetti et al. (Chierichetti *et al.*, 2017) pioneered the concept of *fair clustering*. The problem consists of embedding fairness constraints that encourage clusters to have balanced demographic groups pertaining to some sensitive attributes (e.g., sex, gender, race, etc.), so as to counteract any form of data-inherent bias.

Assume that we are given N data points to be assigned to a set of K clusters, and let $S_k \in \{0,1\}^N$ denotes a binary indicator vector whose components take value 1 when the point is within cluster k, and 0 otherwise. Also suppose that the data contains J different demographic groups, with $V_j \in \{0,1\}^N$ denoting a binary indicator vector of demographic group j. The authors of (Chierichetti *et al.*, 2017; Kleindessner *et al.*, 2019) suggested to evaluate fairness in terms of cluster-balance measures, which take the following form:

$$\text{balance}(S_k) = \min_{j \neq j'} \frac{V_j^t S_k}{V_{j'}^t S_k} \in [0,1] \tag{3.1}$$

The higher this measure, the fairer the cluster. The overall clustering balance is defined by the minimum of Eq. (3.1) over k. This notion of fairness in clusters has given rise to a new line of research that was introduced, mostly, for prototype-based clustering (e.g., K-center and K-median and K-means) (Backurs *et al.*, 2019; Bera *et al.*, 2019; Chierichetti *et al.*, 2017;

Schmidt *et al.*, 2018) and, very recently, for spectral graph clustering (Kleindessner *et al.*, 2019). It raises several interesting questions. How to embed fairness in popular clustering objectives? Can we control the trade-off between some ‘’acceptable” fairness level and the quality of the clustering objective? What is the cost of fairness with respect to the clustering objective and computational complexity?

Chierichetti et al. (Chierichetti *et al.*, 2017) investigated combinatorial approximation algorithms, which ensure the fairness measures in Eq. (3.1) are within an acceptable range, for K-center and K-median clustering, and for binary demographic groups ($J = 2$). They compute *fairlets*, which are groups of points that are fair, and can not be split further into more subsets that are also fair. Then, they consider each fairlet as a data point, and cluster them with approximate K-center or K-median algorithms. Unfortunately, as reported in the experiments in (Chierichetti *et al.*, 2017), obtaining fair solutions with these fairlets-based algorithms comes at the price of a substantial increase in the clustering objectives. Also, the cost for finding fairlets with perfect matching is quadratic w.r.t the number of data points, a complexity that increases for more than two demographic groups. Several combinatorial solutions followed-up on the work in (Chierichetti *et al.*, 2017) to reduce this complexity. For instance, Backurs et al. (Backurs *et al.*, 2019) proposed a solution to make the fairlet decomposition in (Chierichetti *et al.*, 2017) scalable for $J = 2$, by embedding the input points in a tree metric. Rösner and Schmidt (Rösner & Schmidt, 2018) designed a 14-approximate algorithm for fair K-center. (Huang *et al.*, 2019; Schmidt *et al.*, 2018) proposed fair K-means/K-median based on coreset – a reduced proxy set for the full dataset. Bera et al. (Bera *et al.*, 2019) provided a bi-criteria approximation algorithm for fair prototype-based clustering, enabling multiple groups ($J > 2$). It is worth noting that, for large-scale data sets, (Bera *et al.*, 2019; Chierichetti *et al.*, 2017; Rösner & Schmidt, 2018) sub-sample the inputs to mitigate the quadratic complexity w.r.t N. More importantly, the combinatorial algorithms discussed above are tailored for specific prototype-based objectives. For instance, they are not applicable to the very popular graph-clustering objectives, e.g., Ratio Cut or Normalized Cut (Von Luxburg, 2007a), which limits applicability in a breadth of graph problems, in which data takes the form of pairwise affinities.

Kleindessner et al. (Kleindessner *et al.*, 2019) integrated fairness into graph-clustering objectives. They embedded linear constraints on the assignment matrix in spectral relaxation. Then, they solved a constrained trace optimization via finding the K smallest eigenvalues of some transformed Laplacian matrix. However, it is well-known that spectral relaxation has heavy time and memory loads since it requires storing an $N \times N$ affinity matrix and computing its eigenvalue decomposition – the complexity is cubic w.r.t N for a straightforward implementation, and super-quadratic for fast implementations (Tian *et al.*, 2014). In the general context of spectral relaxation and graph partitioning, issues related to computational scalability for large-scale problems is driving an active line of recent work (Shaham *et al.*, 2018; Vladymyrov & Carreira-Perpiñán, 2016; Ziko *et al.*, 2018).

The existing fair clustering algorithms, such as the combinatorial or spectral solutions discussed above, do not have mechanisms that control the trade-off levels between the fairness and clustering objectives. Also, they are tailored either to prototype-based (Backurs *et al.*, 2019; Bera *et al.*, 2019; Chierichetti *et al.*, 2017; Schmidt *et al.*, 2018) or graph-based objectives (Kleindessner *et al.*, 2019). Finally, for a breadth of problems of wide interest, such as pairwise graph data, the computation and memory loads may become an issue for large-scale data sets.

Contributions: We propose a general bound-optimization framework of fair clustering, which integrates an original Kullback-Leibler (KL) fairness term with a large class of clustering objectives, including both prototype-based (e.g., K-means/K-median) and graph-based (e.g., Normalized Cut or Ratio Cut). Fundamentally different from the existing combinatorial and spectral solutions, our variational multi-term approach enables to control the trade-off levels between the fairness and clustering objectives. We derive a general tight upper bound based on a concave-convex decomposition of our fairness term, its Lipschitz-gradient property and the Pinsker inequality. Our tight upper bound can be jointly optimized with various clustering objectives, while yielding a scalable solution, with convergence guarantee. Interestingly, at each iteration, our general variational fair-clustering algorithm performs an independent update for each assignment variable. Therefore, it can easily be distributed for large-scale datasets. This scalibility is important as it enables to explore different trade-off levels between fairness and the clustering objective. Unlike the constrained spectral relaxation in (Kleindessner *et al.*, 2019), our formulation does not require storing an affinity matrix and computing its eigenvalue decomposition. We report comprehensive evaluations and comparisons with state-of-the-art methods over various fair-clustering benchmarks, which show that our variational method can yield highly competitive solutions in terms of fairness and clustering objectives, while being scalable and flexible.

Table 3.1 Auxiliary functions of several well-known clustering objectives. Details on how to derive auxiliary functions for several prototype- or graph-based objectives can be found in (Tang *et al.*, 2019a; Ziko *et al.*, 2018).

Clustering	$\mathcal{F}(\mathbf{S})$	$\mathbf{a}_p^i = [a_{p,k}^i],\ \forall k$	**Where**
K-means	$\sum_N \sum_k s_{p,k}(\mathbf{x}_p - \mathbf{c}_k)^2$	$a_{p,k}^i = (\mathbf{x}_p - \mathbf{c}_k^i)^2$	$\mathbf{c}_k^i = \frac{\mathbf{X}^t S_k^i}{\mathbf{1}^t S_k^i}$
K-median	$\sum_N \sum_k s_{p,k} \mathrm{d}(\mathbf{x}_p, \mathbf{c}_k)$	$a_{p,k}^i = \mathrm{d}(\mathbf{x}_p, \mathbf{c}_k^i)$	$\mathbf{c}_k^i = \arg\min_{p \neq q} \mathrm{d}(\mathbf{x}_p, \mathbf{x}_q)$, d is a distance metric
Ncut	$K - \sum_k \frac{S_k^t \mathbf{W} S_k}{\mathbf{d}^t S_k}$	$a_{p,k}^i = d_p z_k^i - \frac{2\sum_q w(\mathbf{x}_p, \mathbf{x}_q) s_{p,k}^i}{\mathbf{d}^t S_k^i}$	$z_k^i = \frac{(S_k^i)^t \mathbf{W} S_k^i}{\mathbf{d}^t S_k^i}$ $\mathbf{d} = [d_p]$, with $d_p = \sum_q w(\mathbf{x}_p, \mathbf{x}_q); \forall p$ $\mathbf{W} = [w(\mathbf{x}_p, \mathbf{x}_q)]$ is an affinity matrix

Let $\mathbf{X} = \{\mathbf{x}_p \in \mathbb{R}^M, p = 1, \ldots, N\}$ denote a set of N data points to be assigned to K clusters, and $\mathbf{S}$ is a soft cluster-assignment vector: $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_N] \in \{0,1\}^{NK}$. For each point p, $\mathbf{s}_p = [s_{p,k}] \in [0,1]^K$ is the probability simplex vector verifying $\sum_k s_{p,k} = 1$. Suppose that the data set contains J different demographic groups, with vector $V_j = [v_{j,p}] \in \{0,1\}^N$ indicating point assignment to group j: $v_{p,j} = 1$ if data point p is in group j and 0 otherwise. We propose the following general variational formulation for optimizing any clustering objective $\mathcal{F}(\mathbf{S})$ with a fairness penalty, while constraining each $\mathbf{s}_p$ within the K-dimensional probability simplex $\nabla_K = \{\mathbf{y} \in [0,1]^K \mid \mathbf{1}^t \mathbf{y} = 1\}$:

$$\min_{\mathbf{S}} \mathcal{F}(\mathbf{S}) + \lambda \sum_k \mathcal{D}_{\text{KL}}(U||P_k) \quad \text{s.t.} \quad \mathbf{s}_p \in \nabla_K\ \forall p \tag{3.2}$$

$\mathcal{D}_{\text{KL}}(U||P_k)$ denotes the Kullback-Leibler (KL) divergence between the given (required) demographic proportions $U = [\mu_j]$ and the marginal probabilities of the demographics within cluster k:

$$P_k = [P(j|k)];\ P(j|k) = \frac{V_j^t S_k}{\mathbf{1}^t S_k} \forall j, \tag{3.3}$$

where $S_k = [s_{p,k}] \in [0,1]^N$ is the N-dimensional vector [1] containing point assignments to cluster k, and t denotes the transpose operator. Notice that, at the vertices of the simplex (i.e., for hard binary assignments), $V_j^t S_k$ counts the number of points within the intersection of demographic j and cluster k, whereas $\mathbf{1}^t S_k$ is the total number of points within cluster k.

Parameter λ controls the trade-off between the clustering objective and fairness penalty. The problem in (3.2) is challenging due to the ratios of summations in the fairness penalty and the simplex constraints. Expanding KL term $\mathcal{D}_{\text{KL}}(U||P_k)$ and discarding constant $\mu_j \log \mu_j$, our objective in (3.2) becomes equivalent to minimizing the following functional with respect to the relaxed assignment variables, and subject to the simplex constraints:

$$\mathcal{E}(\mathbf{S}) = \underbrace{\mathcal{F}(\mathbf{S})}_{\text{clustering}} + \underbrace{\lambda \sum_k \sum_j -\mu_j \log P(j|k)}_{\text{fairness}} \tag{3.4}$$

Observe that, in Eq. (3.4), the fairness penalty becomes a cross-entropy between the given (target) proportion U and the marginal probabilities P_k of the demographics within cluster k. Notice that our fairness penalty decomposes into convex and concave parts:

$$-\mu_j \log P(j|k) = \underbrace{\mu_j \log \mathbf{1}^t S_k}_{\text{concave}} \underbrace{- \mu_j \log V_j^t S_k}_{\text{convex}}.$$

This enables us to derive tight bounds (auxiliary functions) for minimizing our general fair-clustering model in (3.4) using a a quadratic bound and Lipschitz-gradient property of the convex

[1] The set of N-dimensional vectors S_k and the set of simplex vectors $\mathbf{s}_p$ are two equivalent ways for representing assignment variables. However, we use S_k here for a clearer presentation of the problem, whereas, as will be clearer later, simplex vectors $\mathbf{s}_p$ will be more convenient in the subsequent optimization part.

part, along with Pinsker inequality, and a first-order bound on the concave part. This will be discussed in more details in the following sections for various clustering objectives.

3.2 Proposed bound optimization

Definition 2. $\mathcal{A}_i(\mathbf{S})$ *is an* auxiliary function *of objective* $\mathcal{E}(\mathbf{S})$ *if it is a tight upper bound at current solution* $\mathbf{S}^i$*, i.e., it satisfies the following conditions:*

$$\mathcal{E}(\mathbf{S}) \leq \mathcal{A}_i(\mathbf{S}), \forall \mathbf{S} \tag{3.5a}$$

$$\mathcal{E}(\mathbf{S}^i) = \mathcal{A}_i(\mathbf{S}^i) \tag{3.5b}$$

where i is the iteration index.

Bound optimizers, also commonly referred to as Majorize-Minimize (MM) algorithms (Zhang *et al.*, 2007), update the current solution $\mathbf{S}^i$ to the next by optimizing the auxiliary function:

$$\mathbf{S}^{i+1} = \arg\min_{\mathbf{S}} \mathcal{A}_i(\mathbf{S})$$

These updates guarantee that the original objective function does not increase at each iteration:

$$\mathcal{E}(\mathbf{S}^{i+1}) \leq \mathcal{A}_i(\mathbf{S}^{i+1}) \leq \mathcal{A}_i(\mathbf{S}^i) = \mathcal{E}(\mathbf{S}^i)$$

This general principle is widely used in machine learning as it transforms a difficult problem into a sequence of easier sub-problems (Zhang *et al.*, 2007). Examples of well-known bound optimizers include concave-convex procedures (CCCP) (Yuille & Rangarajan, 2001), expectation maximization (EM) algorithms and submodular-supermodular procedures (SSP) (Narasimhan & Bilmes, 2005), among others. The main technical difficulty in bound optimization is how to derive an auxiliary function. In the following, we derive auxiliary functions for our general fairness-clustering objectives in (3.4).

Proposition 2 (Bound on the fairness penalty)**.** *Given current clustering solution $\mathbf{S}^i$ at iteration i, we have the following auxiliary function on the fairness term in* (3.4)*, up to additive and multiplicative constants, and for current solutions in which each demographic is represented by at least one point in each cluster (i.e., $V_j^t S_k^i \geq 1 \,\forall\, j,k$):*

$$\begin{aligned} \mathcal{G}_i(\mathbf{S}) \quad &\propto \textstyle\sum_{p=1}^N \mathbf{s}_p^t(\mathbf{b}_p^i + \log \mathbf{s}_p - \log \mathbf{s}_p^i) \\ \text{with} \qquad &\mathbf{b}_p^i = [b_{p,1}^i, \dots, b_{p,K}^i] \\ &b_{p,k}^i = \frac{1}{L}\sum_j \left(\frac{\mu_j}{\mathbf{1}^t S_k^i} - \frac{\mu_j v_{j,p}}{V_j^t S_k^i} \right) \end{aligned} \tag{3.6}$$

where L is some positive Lipschitz-gradient constant verifying $L \leq N$

Proof: We provide a detailed proof in the supplemental material. Here, we give the main technical ingredients for obtaining our bound. We use a quadratic bound and a Lipschitz-gradient property for the convex part, and a first-order bound on the concave part. We further bound the quadratic distances between simplex variables with Pinsker inequality (Csiszar & Körner, 2011), which is is well known in information theory. This step avoids completely point-wise Lagrangian-dual projections and inner iterations for handling the simplex constraints, yielding scalable (parallel) updates, with convergence guarantee.

Proposition 3 (Bound on the clustering objective)**.** *Given current clustering solution $\mathbf{S}^i$ at iteration i, we can derive auxiliary functions for several popular clustering objectives $\mathcal{F}(\mathbf{S})$. These auxiliary functions take the following general form (see Table 3.1):*

$$\mathcal{H}_i(\mathbf{S}) = \textstyle\sum_{p=1}^N \mathbf{s}_p^t \mathbf{a}_p^i \tag{3.7}$$

where point-wise (unary) potentials $\mathbf{a}_p^i$ are given in Table 3.1 for several popular clustering objectives.

Proofs: See the corresponding references in Table 3.1.

Algorithm 3.1 Proposed Fair-clustering

```
**Input:** $\mathbf{X}$, Initial seeds, $\lambda$, $U$, $\{V_j\}_{j=1}^{J}$
**Output:** Clustering $labels \in \{1, .., K\}^N$
Initialize $labels$ from initial seeds.
Initialize $\mathbf{S}$ from $labels$.
Initialize $i = 1$.
**repeat**
    Compute $\mathbf{a}_p^i$ from $\mathbf{S}$ (see Table 3.1).
    Initialize $\mathbf{s}_p^i = \frac{\exp(-\mathbf{a}_p^i)}{\mathbf{1}^t \exp(-\mathbf{a}_p^i)}$.
    **repeat**
        Compute $\mathbf{s}_p^{i+1}$ using (3.10).
        $\mathbf{s}_p^i \leftarrow \mathbf{s}_p^{i+1}$.
        $\mathbf{S} = [\mathbf{s}_p^i];\ \forall p$.
    **until** $\mathcal{A}_i(\mathbf{S})$ in (3.8) does not change
    $i = i + 1$.
**until** $\mathcal{E}(\mathbf{S})$ in (3.4) does not change
$l_p = \arg\max_k s_{p,k}; \forall p$.
$labels = \{l_p\}_{p=1}^N$.
```

Proposition 4 (Bound on the fair-clustering functional)**.** *Given current clustering solution* $\mathbf{S}^i$*, the bound on clustering objective* $\mathcal{H}_i$ *and the bound on fairness penalty* $\mathcal{G}_i$ *at iteration i. We have the following auxiliary function for general fair-clustering objective* $\mathcal{E}(\mathbf{S})$ *in* (3.4)*:*

$$\mathcal{A}_i(\mathbf{S}) = \sum_{p=1}^{N} \mathbf{s}_p^t(\mathbf{a}_p^i + \mathbf{b}_p^i + \log \mathbf{s}_p - \log \mathbf{s}_p^i) \tag{3.8}$$

Proof: It is straightforward to check that sum of auxiliary functions, each corresponding to a term in the objective, is also an auxiliary function of the overall objective.

Notice that, at each iteration, our auxiliary function in (3.8) is the sum of *independent* functions, each corresponding to a single data point p. Therefore, our minimization problem in (3.4) can be tackled by optimizing each term over $\mathbf{s}_p$, subject to the simplex constraint, and independently of the other terms, while guaranteeing convergence:

$$\min_{\mathbf{s}_p \in \nabla_K} \mathbf{s}_p^t(\mathbf{a}_p^i + \mathbf{b}_p^i + \log \mathbf{s}_p - \log \mathbf{s}_p^i), \; \forall p \tag{3.9}$$

Also, notice that, in our derived auxiliary function, we obtained a convex negative entropy barrier function $\mathbf{s}_p \log \mathbf{s}_p$, which comes from the convex part in our fairness penalty. This entropy term is very interesting as it avoids completely expensive projection steps and Lagrangian-dual inner iterations for the simplex constraint of each point: As we will see later, it yields closed-form updates for the dual variables of constraints $\mathbf{1}^t\mathbf{s}_p = 1$ and restricts the domain of each $\mathbf{s}_p$ to non-negative values, avoiding extra dual variables for constraints $\mathbf{s}_p \geq 0$. Interestingly, entropy-based barriers are commonly used in Bregman-proximal optimization (Yuan *et al.*, 2017), and have well-known computational benefits when handling difficult simplex constraints (Yuan *et al.*, 2017). However, they are not very common in the general context of clustering.

The objective in (3.9) is the sum of convex functions with affine simplex constraints $\mathbf{1}^t\mathbf{s}_p = 1$. As strong duality holds for the convex objective and the affine simplex constraints, the solutions of the Karush-Kuhn-Tucker (KKT) conditions minimize the auxiliary function. The KKT conditions yield a closed-form solution for both primal variables $\mathbf{s}_p$ and the dual variables (Lagrange multipliers) corresponding to simplex constraints $\mathbf{1}^t\mathbf{s}_p = 1$.

$$\mathbf{s}_p^{i+1} = \frac{\mathbf{s}_p^i \exp(\mathbf{a}_p^i + \lambda \mathbf{b}_p^i)}{\mathbf{1}^t[\mathbf{s}_p^i \exp(\mathbf{a}_p^i + \lambda \mathbf{b}_p^i)]} \; \forall p \tag{3.10}$$

Notice that each closed-form update in (3.10), which globally optimizes (3.9), is within the simplex. We give the pseudo-code of the proposed fair-clustering in **Algorithm 3.1**. The algorithm can be used for any specific clustering objective, e.g., K-means or Ncut, among others, by providing the corresponding $\mathbf{a}_p^i$. The algorithm consists of an inner and an outer loop. The inner iterations updates $\mathbf{s}_p^{i+1}$ using (3.10) until $\mathcal{A}_i(\mathbf{S})$ does not change, with the clustering term $\mathbf{a}_p^i$ fixed from the outer loop. The outer iteration re-computes $\mathbf{a}_p^i$ from the updated $\mathbf{s}_p^{i+1}$. The time complexity of each inner iteration is $\mathcal{O}(NKJ)$. Also, the updates are independent for each data

p and, thus, can be efficiently computed in parallel. In the outer iteration, the time complexity of updating $\mathbf{a}_p^i$ depends on the chosen clustering objective. For instance, for K-means, it is $O(NKM)$, and, for Ncut, it is $O(N^2K)$ for full affinity matrix $\mathbf{W}$ or much lesser for a sparse affinity matrix. Note that $\mathbf{a}_p^i$ can be computed efficiently in parallel for all the clusters.

Table 3.2 Comparison of our proposed fair algorithm with respect to (Backurs *et al.*, 2019).

Datasets	**Fair K-median**			
	Objective		Fairness error / Balance	
	Backurs et. al.	Ours	Backurs et. al.	Ours
Synthetic ($N = 400,\ J = 2$)	140.2	86.03	22.39/0.25	0.18/0.45
Synthetic-unequal ($N = 400,\ J = 2$)	71.63	60.36	11.22/0.21	0.07/0.32
Adult ($N = 32,561,\ J = 2$)	2.38	2.14	0.41/0.16	0.38/0.17
Bank ($N = 41,108,\ J = 3$)	N/A	116.03	N/A	0.02/0.16
Census II ($N = 2,458,285,\ J = 2$)	431714.52	326882.07	0.42/0.36	0.10/0.66

Table 3.3 Comparison of our proposed fair algorithm with respect to (Kleindessner *et al.*, 2019).

Datasets	**Fair NCUT**			
	Objective		Fairness error / Balance	
	Kleindessner et al.	Ours	Kleindessner et al.	Ours
Synthetic ($N = 400,\ J = 2$)	0.0	0.0	22.39/0.0	0.0/1
Synthetic-unequal ($N = 400,\ J = 2$)	0.03	0.06	0.00/0.33	0.00/0.33
Adult ($N = 32,561,\ J = 2$)	2.38	4.48	0.26/0.28	0.32/0.21
Bank ($N = 41,108,\ J = 3$)	N/A	2.36	N/A	0.3/0.11
Census II ($N = 2,458,285,\ J = 2$)	N/A	0.52	N/A	0.41/0.43

3.3 Experiments

In this section, we present comprehensive empirical evaluations of the proposed fair-clustering algorithm, along with comparisons with state-of-the-art fair-clustering techniques. We choose three well-known clustering objectives: K-means, K-median and Normalized cut (Ncut), and integrate our fairness-penalty bound with the corresponding clustering bounds $\mathbf{a}_p$ (see Table 3.1). We refer to our bound-optimization versions as: Fair K-means, Fair K-median and Fair

Ncut. Note that our formulation can be used for other clustering objectives (if a bound could be derived for the objective).

We investigate the effect of fairness on the original hard clustering objectives, and compare with the existing methods in terms of fairness and clustering objectives. For a fair comparison with combinatorial and discrete methods, we use the hard (binary assignments) versions of the soft solutions obtained by our variational method at convergence. In regard to the hard fairness objective (i.e., w.r.t. binary assignment variables), we evaluate the results in terms of the balance of each cluster S_k in (3.1), and define the overall *balance* of the clustering as balance = $\min_{S_k}$ balance(S_k). We further propose to evaluate the *fairness error*, which is the KL divergence $\mathcal{D}_{\text{KL}}(U||P_k)$ in (3.2). This KL measure becomes equal to zero when the proportions of the demographic groups within all the output clusters match the target distribution. For Ncut, we use 20-nearest neighbor affinity matrix, $\mathbf{W}$: $w(\mathbf{x}_p, \mathbf{x}_q) = 1$ if data point $\mathbf{x}_q$ is within the 20-nearest neighbors of $\mathbf{x}_p$, and equal to 0 otherwise. In all the experiments, we fixed $L = 1$ and found that this does not increase the objective (see the detailed explanation in the supplemental material). We performed L2-normalization of the features, and used the standard K-means++ (Arthur & Vassilvitskii, 2007) to generate initial partitions for all the models.

3.3.1 Datasets

Synthetic datasets. We created two types of synthetic datasets according to the proportions of the demographics, each having two clusters and a total of 400 data points in 2D features (figures in supplemental). The *Synthetic* dataset contains two perfectly balanced demographic groups, each having an equal number of 200 points. For this data set, we imposed target target proportions $U = [0.5, 0.5]$. To experiment with our fairness penalty with unequal proportions, we also used *Synthetic-unequal* dataset with 300 and 100 points within each of the two demographic groups. In this case, we imposed target proportions $U = [0.75, 0.25]$.

Real datasets. We use three datasets from the UCI machine learning repository (Dua & Graff, 2017), one large-scale data set whose demographics are balanced (Census), along with two other data sets with various demographic proportions:

Bank [2] dataset contains 41188 number of records of direct marketing campaigns of a Portuguese banking institution corresponding to each client contacted (Moro, Cortez & Rita, 2014). We utilize the marital status as the sensitive attribute, which contains three groups ($J = 3$) – single, married and divorced – and removed the ''Unknown" marital status. Thus, we have $41,108$ records in total. We chose 6 numeric attributes (age, duration, euribor of 3 month rate, no. of employees, consumer price index and number of contacts performed during the campaign) as features, set the number of clusters $K = 30$, and impose the target proportions of three groups $U = [0.28, 0.61, 0.11]$ within each cluster.

Adult[3] is a US census record data set from 1994. The dataset contains $32,561$ records. We used the gender status as the sensitive attribute, which contains 10771 females and 21790 males. We chose the 4 numeric attributes as features, set the number of clusters to $K = 30$, and impose proportions $U = [0.33, 0.67]$ within each cluster.

Census[4] is a large-scale data set corresponding to a US census record data from 1990. The dataset contains $2,458,285$ records. We used the gender status as the sensitive attribute, which contains $1,191,601$ females and $1,266,684$ males. We chose the 25 numeric attributes as features, similarly to (Backurs *et al.*, 2019). We set the number of clusters to $K = 20$, and imposed proportions $U = [0.48, 0.52]$ within each cluster.

3.3.2 Results

In this section, we discuss the results of the different experiments to evaluate the proposed algorithm for Fair Ncut, Fair K-means and Fair K-median clustering. We further report

[2] https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

[3] https://archive.is.uci/ml/datasets/adult

[4] https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990)

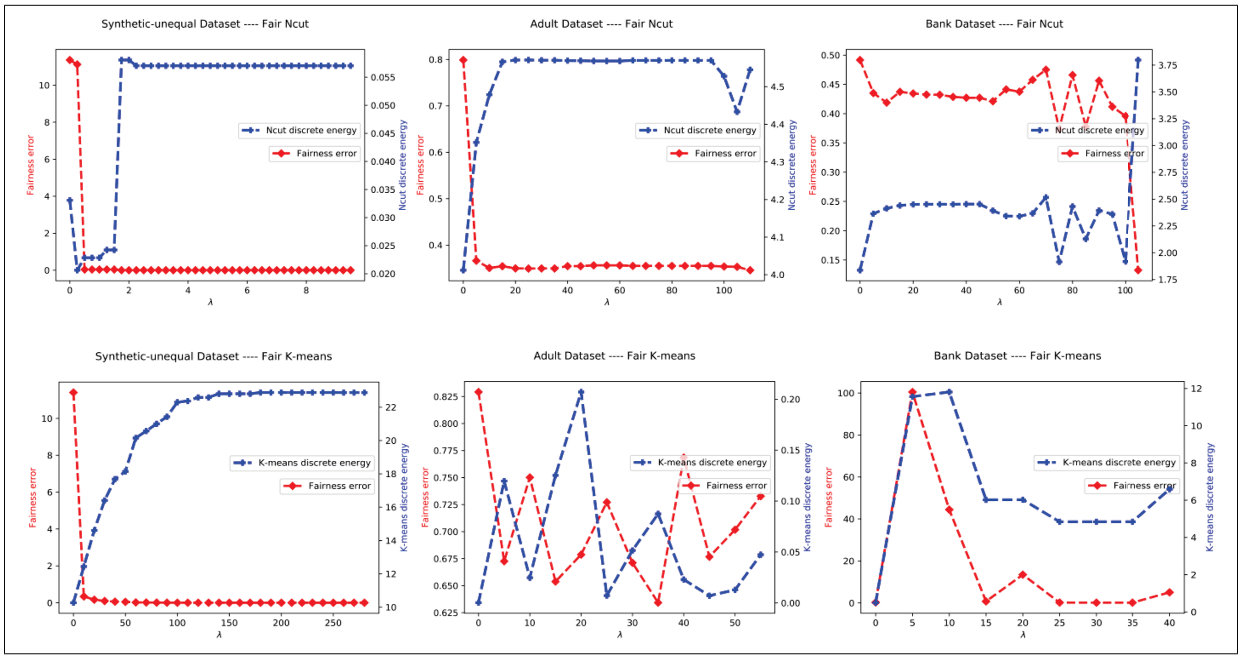


Figure 3.1 The clustering objective vs. λ vs. fairness error on the *Synthetic-unequal*, *Adult* and *Bank* datasets. The first row shows the results with Fair Ncut, while the second row shows the results with Fair K-means.

comparisons with (Backurs *et al.*, 2019) and (Kleindessner *et al.*, 2019) in terms of discrete fairness measures and clustering objectives.

Trade-off between clustering and fairness objectives. We assess the effect of incorporating fairness constraints on the original clustering objectives. In each plot in Fig. 3.1, the blue curve depicts the discrete-valued clustering objective $\mathcal{F}(\mathbf{S})$ (K-means or Ncut) obtained at convergence as a function of λ, the weight of the fairness term. On each of these plots, we also show the fairness errors, depicted in red. Observe that the behaviour of our models is consistent with previous fair-clustering works, i.e., the discrete clustering objective increases with decreased fairness error, which is intuitive. On the one hand, the best (lowest) fairness error is, typically, obtained for several values of λ greater than a certain value. On the other hand, the smaller the value of λ, the better the clustering objective at convergence. Therefore, there is a value of λ, which yields the best trade-off between fairness and clustering objectives. This makes the scalability of our model highly relevant because we can explore several solutions of our algorithm, each corresponding to a different value of λ, and choose the best solution in terms the clustering objective and a desired fairness level at convergence. For instance, we can run our model for different values of λ, and choose the smallest λ corresponding the best fairness error obtained at convergence. This flexibility enables us to obtain better solutions, in terms of fairness and clustering objectives, than several recent fair clustering methods.

Convergence. In Fig. 3.2, we plot the fair-clustering objectives for Fair K-means, Fair Ncut in (3.4) in each outer iteration of our algorithm. Observe that the objective decreases with each outer iteration, and converges.

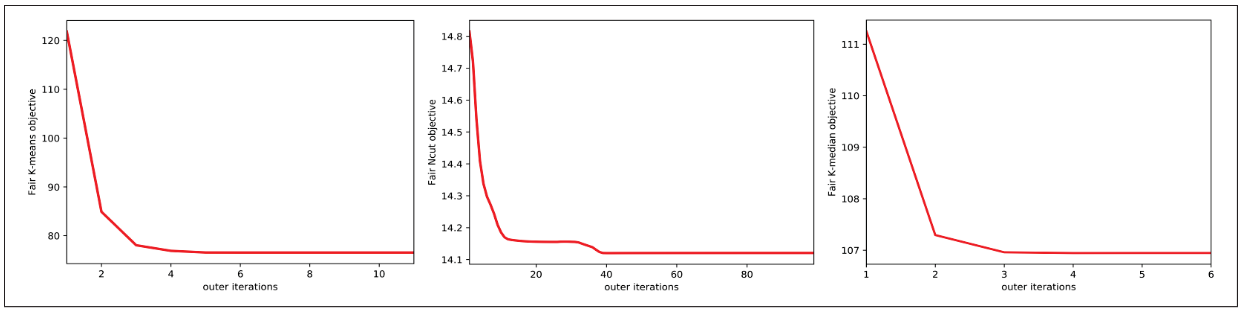


Figure 3.2 The convergence of the proposed bound optimizer for minimizing general fair-clustering objective in (3.4), for Fair K-means and Fair Ncut, and Fair K-median on the *Synthetic* dataset.

Comparison of Fair K-median to (Backurs *et al.*, 2019) and Fair Ncut to (Kleindessner *et al.*, 2019). Our algorithm is flexible as it can be used in conjunction with different well-known clustering objectives. This enabled us to compare our Fair K-median and Fair Ncut versions to (Backurs *et al.*, 2019) and (Kleindessner *et al.*, 2019), respectively. Tables 3.2 and 3.3 report comparisons in terms of the original clustering objectives, achieved minimum balances and fairness errors, for Fair K-median and Fair NCUT. For our model, we run the algorithm for different values of λ, and choose the smallest λ corresponding the best fairness error obtained at convergence. This flexibility and scalability enabled us to obtain significantly better clustering objectives and fairness/minimum-balance measures in comparisons to (Backurs *et al.*, 2019); See Table 3.2. It is worth noting that, for the *Bank* dataset, we were unable to run (Backurs *et al.*, 2019) as the number of demographic group is 3 (i.e. $J > 2$).

For fair Ncut, in the case of the *Synthetic* dataset, we achieved the desired balance whereas (Kleindessner *et al.*, 2019) obtained a high fairness error with a minimum balance equal to zero. Both our method and (Kleindessner *et al.*, 2019) achieved the same Ncut clustering objective on the *Synthetic* dataset. In the *Adult* dataset, (Kleindessner *et al.*, 2019) achieved better Ncut and fairness objectives than our model. However, we were unable to run the spectral solution of (Kleindessner *et al.*, 2019) for large-scale *Census II* data set, and for *Bank*, due to its computational and memory load (as it requires computing the eigen values of the square affinity matrix).

Scalability. On the large Census II dataset, we achieved the fair clustering result in 632.5 seconds, while (Backurs *et al.*, 2019) took 860.5 seconds. Note that all the methods were compared on the same computing environment, with the same initialization and data. Our fair K-median clustering achieved better K-median clustering objectives, with reduced fairness errors in comparison to (Backurs *et al.*, 2019), while being faster. Also, our algorithm scales up to more than two demographic groups, i.e. when $J > 2$ (e.g. *Bank*), unlike many of the existing approaches. Furthermore, Ncut graph clustering, our bound optimizer can deal with large-scale data sets, unlike (Kleindessner *et al.*, 2019), which requires eigen decomposition for large affinity matrices. Finally, the parallel structure of our algorithm within each iteration (i.e., independent updates for each assignment variable) enables to explore different values of λ, thereby choosing the best trade-off between the clustering objective and fairness error.

3.4 Conclusion

We presented a variational, bound-optimization formulation that integrates fairness with various well-known clustering objectives. It enables to control the trade-off between the clustering objective and fairness criterion: we can choose a trade-off that yields a given acceptable fairness level, while yielding the best possible clustering objective. This yielded competitive solutions in terms of clustering and fairness objectives in comparisons to state-of-the-art methods over various fair-clustering benchmarks. Furthermore, our method enables parallel updates of cluster assignments for each data point, with convergence guarantee, yielding a scalable and computationally efficient solution, in terms of the number of data points and demographic groups.

3.5 Supplemental

Definition 3. *A convex function f defined over a convex set $\Omega \in \mathbb{R}^l$ is* L-smooth *if the gradient of f is Lipschitz (with a Lipschitz constant $L > 0$): $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L.\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \Omega$. Equivalently, there exists a strictly positive L such that the Hessian of f verifies: $\nabla^2 f(\mathbf{x}) \preceq L\mathbf{I}$ where $\mathbf{I}$ is the identity matrix.*

Remark 1. *Let $\sigma_{max}(f)$ denotes the maximum Eigen value of $\nabla^2 f(\mathbf{x})$ is a valid Lipschitz constant for the gradient of f because $\nabla^2 f(\mathbf{x}) \preceq \sigma_{max}(f)\mathbf{I}$*

Lipschitz gradient implies the following bound[5] on $f(\mathbf{x})$

Lemma 1 (Quadratic upper bound)**.** *If f is* L-smooth*, then we have the following quadratic upper bound:*

$$f(\mathbf{x}) \leq f(\mathbf{y}) + [\nabla f(\mathbf{y})]^t(\mathbf{x} - \mathbf{y}) + L.\|\mathbf{x} - \mathbf{y}\|^2 \tag{3.11}$$

Proof: The proof of this lemma is straightforward. It suffices to start from convexity condition $f(\mathbf{y}) \geq f(\mathbf{x}) + [\nabla f(\mathbf{x})]^t(\mathbf{y} - \mathbf{x})$ and use Cauchy-Schwarz inequality and the Lipschitz gradient condition:

$$\begin{aligned} f(\mathbf{x}) &\leq f(\mathbf{y}) + [\nabla f(\mathbf{x})]^t(\mathbf{x} - \mathbf{y}) \\ &= f(\mathbf{y}) + [\nabla f(\mathbf{y})]^t(\mathbf{x} - \mathbf{y}) + [\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})]^t(\mathbf{x} - \mathbf{y})\rangle \\ &\leq f(\mathbf{y}) + [\nabla f(\mathbf{y})]^t(\mathbf{x} - \mathbf{y}) + \|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|.\|\mathbf{x} - \mathbf{y}\| \\ &\leq f(\mathbf{y}) + [\nabla f(\mathbf{y})]^t(\mathbf{x} - \mathbf{y}) + L.\|\mathbf{x} - \mathbf{y}\|^2 \end{aligned} \tag{3.12}$$

[5] This implies that the distance between the $f(\mathbf{x})$ and its first-order Taylor approximation at $\mathbf{y}$ is between 0 and $L.\|\mathbf{x} - \mathbf{y}\|^2$. Such a distance is the Bregman divergence with respect to the l_2 norm.

Lemma 2. *For any* $\mathbf{x}$ *and* $\mathbf{y}$ *belonging to the* K*-dimensional probability simplex* $\mathcal{S} = \{\mathbf{x} \in [0,1]^K \mid \mathbf{1}^t\mathbf{x} = 1\}$*, we have the following inequality:*

$$\mathcal{D}_k(\mathbf{x}||\mathbf{y}) \geq \frac{1}{2}||\mathbf{x} - \mathbf{y}||^2 \tag{3.13}$$

where $\mathcal{D}_k$ *is the Kullback-Leibler divergence:*

$$\mathcal{D}_k(\mathbf{x}||\mathbf{y}) = \sum_k x_k \log \frac{x_k}{y_k} \tag{3.14}$$

Proof: Let $q_{\mathbf{o}}(\mathbf{x}) = \mathcal{D}_k(\mathbf{x}||\mathbf{o})$. The Hessian of $q_{\mathbf{o}}$ is a diagonal matrix whose diagonal elements are given by: $\frac{1}{x_k}, k = 1, | \dots K$. Now because $\mathbf{x} \in \mathcal{S}$, we have $\frac{1}{x_i} > 1 \quad \forall i$. Therefore, $q_{\mathbf{o}}$ is 1-*strongly* convex: $\nabla^2 q_{\mathbf{o}}(\mathbf{x}) \succeq \mathbf{I}$. This is equivalent to:

$$q_{\mathbf{o}}(\mathbf{x}) \geq q_{\mathbf{o}}(\mathbf{y}) + [\nabla q_{\mathbf{o}}(\mathbf{y})]^t(\mathbf{x} - \mathbf{y}) + \frac{1}{2}||\mathbf{x} - \mathbf{y}||^2 \tag{3.15}$$

The gradient of $q_{\mathbf{o}}$ is given by:

$$\nabla q_{\mathbf{o}}(\mathbf{y}) = (1 + \log \frac{\mathbf{y}_1}{\mathbf{o}_1}, \dots, 1 + \log \frac{\mathbf{y}_k}{\mathbf{o}_k})^t. \tag{3.16}$$

Applying this expression to $\mathbf{o} = \mathbf{y}$, notice that $\nabla q_{\mathbf{o}}(\mathbf{y}) = \mathbf{1}$. Using these in expression (3.15) for $\mathbf{o} = \mathbf{y}$, we get:

$$\mathcal{D}_k(\mathbf{x}||\mathbf{y}) \geq \mathbf{1}^t(\mathbf{x} - \mathbf{y}) + \frac{1}{2}||\mathbf{x} - \mathbf{y}||^2 \tag{3.17}$$

Now, because $\mathbf{x}$ and $\mathbf{y}$ are in $\mathcal{S}$, we have $\mathbf{1}^t(\mathbf{x} - \mathbf{y}) = \sum_k \mathbf{x}_k - \sum_k \mathbf{y}_k = 1 - 1 = 0$. This yields the result in Lemma 2.

3.5.1 Proof of Proposition 2

We present a detailed proof of **Proposition 2 (Bound on fairness)**. Recall that, we wrote the fairness clustering problem in the following form:

$$\mathcal{E}(\mathbf{S}) = \underbrace{\mathcal{F}(\mathbf{S})}_{\text{clustering}} \underbrace{+\lambda \sum_k \sum_j -\mu_j \log P(j|k)}_{\text{fairness}} \tag{3.18}$$

The proposition for the bound on the fairness penalty states the following: Given current clustering solution $\mathbf{S}^i$ at iteration i, we have the following tight upper bound (auxiliary function) on the fairness term in (3.4), up to additive and multiplicative constants, and for current solutions in which each demographic is represented by at least one point in each cluster (i.e., $V_j^t S_k^i \geq 1 \,\forall\, j, k$):

$$\begin{aligned} \mathcal{G}_i(\mathbf{S}) &\propto \textstyle\sum_{p=1}^N \mathbf{s}_p^t(\mathbf{b}_p^i + \log \mathbf{s}_p - \log \mathbf{s}_p^i) \\ \textit{with} \quad & \mathbf{b}_p^i = [b_{p,1}^i, \dots, b_{p,K}^i] \\ & b_{p,k}^i = \frac{1}{L} \sum_j \left(\frac{\mu_j}{\mathbf{1}^t S_k^i} - \frac{\mu_j v_{j,p}}{V_j^t S_k^i} \right) \end{aligned} \tag{3.19}$$

where L is some positive Lipschitz-gradient constant verifying $L \leq N$

Proof: We can expand each term in the fairness penalty in (3.18), and write it as the sum of two functions, one is convex and the other is concave:

$$\begin{aligned} -\mu_j \log P(j|k) &= \mu_j \log \mathbf{1}^t S_k - \mu_j \log V_j^t S_k \\ &= g_1(S_k) + g_2(S_k) \end{aligned} \tag{3.20}$$

Let us represent the $N \times K$ matrix $\mathbf{S} = \{S_1, \dots, S_K\}$ in its equivalent vector form $\mathbf{S} = [\mathbf{s}_1, \dots, \mathbf{s}_N] \in [0,1]^{NK}$, where $\mathbf{s}_p = [s_{p,1}, \dots, s_{p,K}] \in [0,1]^K$ is the probability simplex assignment vector for point p. We shall see later, this equivalent simplex-variable representation will be convenient for deriving our bound.

Bound on $\tilde{g}_1(\mathbf{S}) = \sum_k g_1(S_k)$:
For concave part g_1, we can get a tight upper bound (auxiliary function) by its first-order approximation at current solution S_k^i:

$$\begin{aligned} g_1(S_k) &\leq g_1(S_k^i) + [\nabla g_1(S_k^i)]^t(S_k - S_k^i) \\ &= [\nabla g_1(S_k^i)]^t S_k + const \end{aligned} \tag{3.21}$$

where gradient vector $\nabla g_1(S_k^i) = \frac{\mu_j}{\mathbf{1}^t S_k^i}\mathbf{1}$ and $const$ is the sum of all the constant terms. Now consider $N \times K$ matrix $\mathbf{T}_1 = \{\nabla g_1(S_1^i), \dots \nabla g_1(S_K^i)\}$ and its equivalent vector representation $\mathbf{T}_1 = [\mathbf{t}_1^1, \dots, \mathbf{t}_1^N] \in \mathbb{R}^{NK}$, which concatenates rows $\mathbf{t}_1^p \in \mathbb{R}^K$, $p \in \{1, \dots N\}$, of the $N \times K$ matrix into a single NK-dimensional vector. Summing the bounds in (3.21) over $k \in \{1, \dots K\}$ and using the NK-dimensional vector representation of both $\mathbf{S}$ and $\mathbf{T}_1$, we get:

$$\tilde{g}_1(\mathbf{S}) \leq \mathbf{T}_1^t \mathbf{S} + const \tag{3.22}$$

Bound on $\tilde{g}_2(\mathbf{S}) = \sum_k g_2(S_k)$:
For convex part g_2, the upper bound (auxiliary function) can be found by using the **Lemma 1** and **Definition 3**:

$$\begin{aligned} g_2(S_k) &\leq g_2(S_k^i) + [\nabla g_2(S_k^i)]^t(S_k - S_k^i) + L\|S_k - S_k^i\|^2 \\ &= [\nabla g_2(S_k^i)]^t S_k + L\|S_k - S_k^i\|^2 + const \end{aligned} \tag{3.23}$$

where gradient vector $\nabla g_2(S_k^i) = -\frac{\mu_j V_j}{V_j^t S_k^i} \in \mathbb{R}^N$ and L is a valid Lipschitz constant for the gradient of g_2. Similarly to earlier, consider $N \times K$ matrix $\mathbf{T}_2 = \{\nabla g_2(S_1^i), \dots \nabla g_2(S_K^i)\}$ and it equivalent vector representation $\mathbf{T}_2 = [\mathbf{t}_2^1, \dots, \mathbf{t}_2^N] \in \mathbb{R}^{NK}$. Using this equivalent vector representations for matrices $\mathbf{T}_2$, $\mathbf{S}$ and $\mathbf{S}^i$, and summing the bounds in (3.23) over k, we get:

$$\tilde{g}_2(\mathbf{S}) \quad \leq \quad \mathbf{T}_2^t\mathbf{S} + L\|\mathbf{S} - \mathbf{S}^i\|^2 + const \tag{3.24}$$

In our case, the Lipschitz constant is: $L = \sigma_{max}$, where σ_{max} is the maximum eigen value of the Hessian matrix:

$$\nabla^2(g_2(S_k^i)) = \frac{\mu_j}{(V_j^t S_k^i)^2} V_j V_j^t.$$

Note that, $\|\mathbf{S} - \mathbf{S}^i\|^2$ is defined over the simplex variable of each data point $\mathbf{s}_p$. Thus, we can utilize the **Lemma 2 (Pinsker inequality)** and get the following bound on $\tilde{g}_2(\mathbf{S})$:

$$\tilde{g}_2(\mathbf{S}) \quad \leq \quad \mathbf{S}^t[\mathbf{T}_2 + L\log\mathbf{S} - L\log\mathbf{S}^i] \tag{3.25}$$

Total bound on the Fairness term:

By taking into account the sum over all the demographics j and combining the bounds for $\tilde{g}_1(\mathbf{S})$ and $\tilde{g}_2(\mathbf{S})$, we get the following bound for the fairness term:

$$\begin{aligned}
\mathcal{G}_i(\mathbf{S}) &= \mathbf{S}^t\left[\sum_j(\mathbf{T}_1 + \mathbf{T}_2) + L\log\mathbf{S} - L\log\mathbf{S}^i\right] \\
&\propto \sum_{p=1}^{N} \mathbf{s}_p^t(\mathbf{b}_p^i + \log\mathbf{s}_p - \log\mathbf{s}_p^i) \\
\textit{with} \quad & \mathbf{b}_p^i = [b_{p,1}^i, \ldots, b_{p,K}^i] \\
& b_{p,k}^i = \frac{1}{L}\sum_j\left(\frac{\mu_j}{\mathbf{1}^t S_k^i} - \frac{\mu_j v_{j,p}}{V_j^t S_k^i}\right)
\end{aligned} \tag{3.26}$$

Note that for current solutions in which each demographic is represented by at least one point in each cluster (i.e., $V_j^t S_k^i \geq 1 \, \forall \, j, k$), the maximum eigen value of the Hessian $\nabla^2(g_2(S_k^i))$ is bounded by N, which means $L \leq N$. Note that, in our case, typically the term $\frac{\mu_j}{(V_j^t S_k^i)^2}$ in the Hessian is much smaller than 1. Therefore, in practice, setting a suitable positive $L << N$ does not increase the objective.

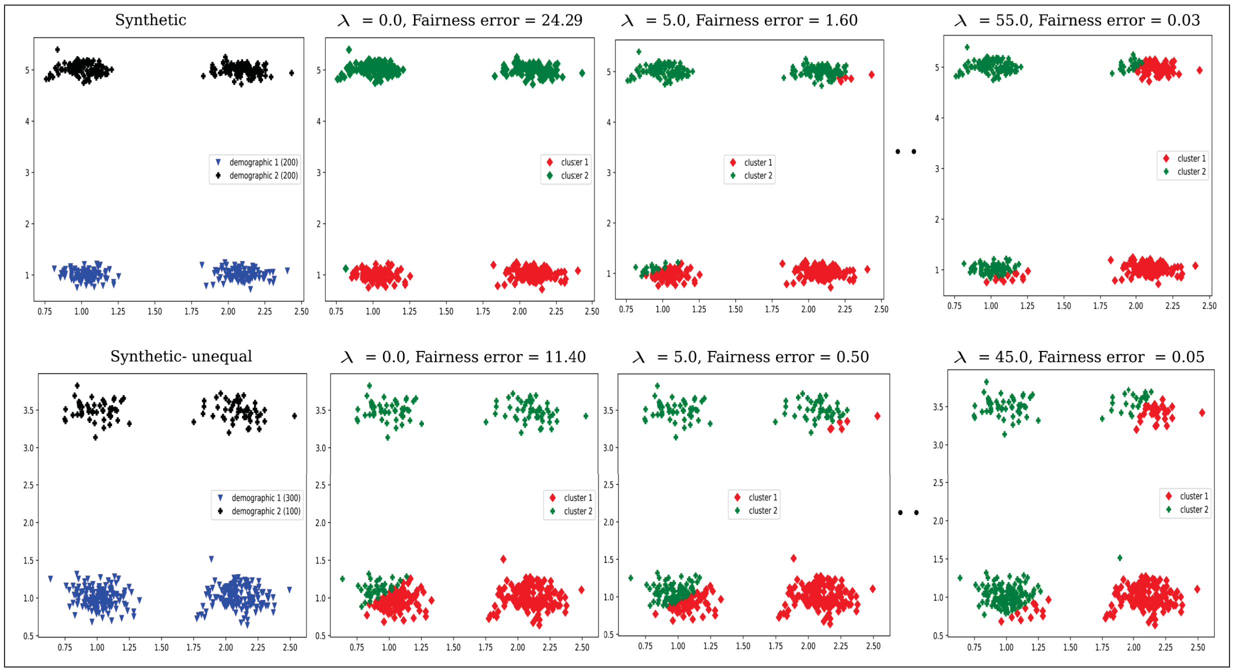


Figure 3.3 Output clusters of Fair K-means with respect to λ on synthetic datasets. Demographics are colored in either black or blue and the output clusters are colored in either red or green. *First row* – 1st: *Synthetic* dataset with two equal demographics. (2nd-4th): With the increased λ parameter, the output clusters get balanced demographics. *Second row* – 1st: *Synthetic-unequal* dataset with different demographic proportions $U = [0.75, 0.25]$. (2nd-4th): output clusters colored in either red or green. With the increased λ parameter, the output clusters are according to the given proportions of demographics, with almost 0 fairness error.

3.5.2 Output clusters with respect to λ.

In Fig.3.3, we plot the output clusters of Fair K-means with respect to an increased value of λ, for the synthetic data sets. When $\lambda = 0$, we get the traditional clustering results of K-means without fairness. The result clearly has biased clusters, each corresponding fully to one the demographic groups, with a balance measure equal 0. In the *Synthetic* dataset, the balance increases with increased value of parameter λ and eventually gain the desired equal balance with a certain increased value of λ. We also observe the same trend in the *Synthetic-unequal* dataset, where the output clusters are found according to prior demographic distribution $U = [0.75, 0.25]$, with almost a null fairness error starting from a certain value of λ.

CHAPTER 4

LAPLACIAN REGULARIZED FEW-SHOT LEARNING

Imtiaz Masud Ziko [a], Jose Dolz [b], Eric Granger [c], Ismail Ben Ayed [d]

[a, b, c, d] Department of Systems Engineering, École de technologie supérieure,
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Article accepted in «International Conference on Machine Learning» 2020,
To appear in the «Proceedings of Machine Learning Research», vol. 119, 2020.

Abstract

We propose a transductive Laplacian-regularized inference for few-shot tasks. Given any feature embedding learned from the base classes, we minimize a quadratic binary-assignment function containing two terms: (1) a unary term assigning query samples to the nearest class prototype, and (2) a pairwise Laplacian term encouraging nearby query samples to have consistent label assignments. Our transductive inference does not re-train the base model, and can be viewed as a graph clustering of the query set, subject to supervision constraints from the support set. We derive a computationally efficient bound optimizer of a relaxation of our function, which computes independent (parallel) updates for each query sample, while guaranteeing convergence. Following a simple cross-entropy training on the base classes, and without complex meta-learning strategies, we conducted comprehensive experiments over five few-shot learning benchmarks. Our LaplacianShot consistently outperforms state-of-the-art methods by significant margins across different models, settings, and data sets. Furthermore, our transductive inference is very fast, with computational times that are close to inductive inference, and can be used for large-scale few-shot tasks.

4.1 Introduction

Deep learning models have achieved human-level performances in various tasks. The success of these models rely considerably on exhaustive learning from large-scale labeled data sets.

Nevertheless, they still have difficulty generalizing to novel classes unseen during training, given only a few labeled instances for these new classes. In contrast, humans can learn new tasks easily from a handful of examples, by leveraging prior experience and related context. Few-shot learning (Fei-Fei *et al.*, 2006; Miller *et al.*, 2000; Vinyals *et al.*, 2016) has emerged as an appealing paradigm to bridge this gap. Under standard few-shot learning scenarios, a model is first trained on substantial labeled data over an initial set of classes, often referred to as the base classes. Then, supervision for novel classes, which are unseen during base training, is limited to just one or few labeled examples per class. The model is evaluated over few-shot tasks, each one supervised by a few labeled examples per novel class (the support set) and containing unlabeled samples for evaluation (the query set).

The problem has recently received substantial research interests, with a large body of work based on complex meta-learning and episodic-training strategies. The meta-learning setting uses the base training data to create a set of few-shot tasks (or episodes), with support and query samples that simulate generalization difficulties during test times, and train the model to generalize well on these artificial tasks. For example, (Vinyals *et al.*, 2016) introduced matching network, which employs an attention mechanism to predict the unknown query samples as a linear combination of the support labels, while using episodic training and memory architectures. Prototypical networks (Snell *et al.*, 2017) maintain a single prototype representation for each class in the embedding space, and minimize the negative log-probability of the query features with episodic training. Ravi & Larochelle (2017) viewed optimization as a model for few-shot learning, and used an LSTM meta-learner to update classifier parameters. Finn et al. (2017) proposed MAML, a meta-learning strategy that attempts to make a model "easy" to fine-tune. These widely adopted works were recently followed by an abundant meta-learning literature, for instance, (Hou *et al.*, 2019; Mishra *et al.*, 2018; Oreshkin *et al.*, 2018; Rusu *et al.*, 2019; Sung *et al.*, 2018; Yanbin *et al.*, 2019; Ye *et al.*, 2020), among many others.

Several recent studies explored transductive inference for few-shot tasks, e.g., (Dhillon *et al.*, 2020; Hou *et al.*, 2019; Hu *et al.*, 2020; Kim *et al.*, 2019; Qiao *et al.*, 2019; Yanbin *et al.*, 2019), among others. Given a few-shot task at test time, transductive inference performs class predictions jointly for all the unlabeled query samples of the task, rather than one sample at a time as in inductive inference. For instance, TPN (Yanbin *et al.*, 2019) used label propagation (Dengyong *et al.*, 2004) along with episodic training and a specific network architecture, so as to learn how to propagate labels from labeled to unlabeled samples. CAN-T (Hou *et al.*, 2019) is another meta-learning based transductive method, which uses attention mechanisms to propagate labels to unlabeled query samples. The transductive fine-tuning method by (Dhillon *et al.*, 2020) re-train the network by minimizing an additional entropy loss, which encourages peaked (confident) class predictions at unlabeled query points, in conjunction with a standard cross-entropy loss defined on the labeled support set.

Transductive few-shot methods typically perform better than their inductive counterparts. However, this may come at the price of a much heavier computational complexity during inference. For example, the entropy fine-tuning in (Dhillon *et al.*, 2020) re-trains the network, performing gradient updates over all the parameters during inference. Also, the label propagation in (Yanbin *et al.*, 2019) requires a matrix inversion, which has a computational overhead that is cubic with respect to the number of query samples. This may be an impediment for deployment for large-scale few-shot tasks.

We propose a transductive Laplacian-regularized inference for few-shot tasks. Given any feature embedding learned from the base data, our method minimizes a quadratic binary-assignment function integrating two types of potentials: (1) unary potentials assigning query samples to the nearest class prototype, and (2) pairwise potentials favoring consistent label assignments for nearby query samples. Our transductive inference can be viewed as a graph clustering of the query set, subject to supervision constraints from the support set, and does not re-train the base model. Following a relaxation of our function, we derive a computationally efficient bound optimizer, which computes independent (parallel) label-assignment updates for each query point, with guaranteed convergence. We conducted comprehensive experiments on five few-shot learning benchmarks, with different levels of difficulties. Using a simple cross-entropy training on the base classes, and without complex meta-learning strategies, our LaplacianShot outperforms state-of-the-art methods by significant margins, consistently providing improvements across

Algorithm 4.1 Proposed Algorithm for LaplacianShot

```
**Input:** $\mathbb{X}_{\text{s}}$, $\mathbb{X}_{\text{q}}$, $\lambda$, $f_\theta$
**Output:** $Labels \in \{1, .., C\}^N$ for $\mathbb{X}_{\text{q}}$
Get prototypes $\mathbf{m}_c$.
Compute $\mathbf{a}_q$ using (4.8a) $\forall \mathbf{x}_q \in \mathbb{X}_{\text{q}}$.
Initialize $i = 1$.
Initialize $\mathbf{y}_q^i = \frac{\exp(-\mathbf{a}_q)}{\mathbf{1}^t \exp(-\mathbf{a}_q)}$.
**repeat**
  Compute $\mathbf{y}_q^{i+1}$ using (4.12)
  $\mathbf{y}_q^i \leftarrow \mathbf{y}_q^{i+1}$.
  $\mathbf{Y} = [\mathbf{y}_q^i];\ \forall q$.
  $i = i + 1$.
**until** $\mathcal{B}_i(\mathbf{Y})$ in (4.7) does not change
$l_q = \arg\max_c \mathbf{y}_q;\ \forall \mathbf{y}_q \in \mathbf{Y}$.
$Labels = \{l_q\}_{q=1}^N$
```

different settings, data sets, and training models. Furthermore, our transductive inference is very fast, with computational times that are close to inductive inference, and can be used for large-scale tasks.

4.2 Laplacian Regularized Few-Shot Learning

4.2.1 Proposed Formulation

In the few-shot setting, we are given a labeled support set $\mathbb{X}_{\text{s}} = \bigcup_{c=1}^{C} \mathbb{X}_{\text{s}}^c$ with C test classes, where each novel class c has $|\mathbb{X}_{\text{s}}^c|$ labeled examples, for instance, $|\mathbb{X}_{\text{s}}^c| = 1$ for 1-shot and $|\mathbb{X}_{\text{s}}^c| = 5$ for 5-shot. The objective of few-shot learning is, therefore, to accurately classify unlabeled unseen query sample set $\mathbb{X}_{\text{q}} = \bigcup_{c=1}^{C} \mathbb{X}_{\text{q}}^c$ from these C test classes. This setting is referred to as the $|\mathbb{X}_{\text{s}}^c|$-shot C-way few-shot learning.

Let f_θ denotes the embedding function of a deep convolutional neural network, with parameters θ and $\mathbf{x}_q = f_\theta(\mathbf{s}_q) \in \mathbb{R}^M$ encoding the features of a given data point $\mathbf{s}_q$. Embedding f_θ is learned from a labeled training set $\mathbb{X}_{\text{base}}$, with base classes that are different from the few-shot classes of

$\mathbb{X}_s$ and $\mathbb{X}_q$. In our work, parameters θ are learned through a basic network training with the standard cross-entropy loss defined over $\mathbb{X}_{base}$, without resorting to any complex episodic-training or meta-learning strategy. For each query feature point $\mathbf{x}_q$ in a few-shot task, we define a latent binary assignment vector $\mathbf{y}_q = [y_{q,1}, \ldots, y_{q,C}]^t \in \{0,1\}^C$, which is within the C-dimensional probability simplex $\nabla_C = \{\mathbf{y} \in [0,1]^C \mid \mathbf{1}^t\mathbf{y} = 1\}$: binary $y_{q,c}$ is equal to 1 if $\mathbf{x}_q$ belongs to class c, and equal to 0 otherwise. t is used as the transpose operator. Let $\mathbf{Y}$ denotes the $N \times C$ matrix whose rows are formed by $\mathbf{y}_q$, where N is the number of query points in $\mathbb{X}_q$. We propose a transductive few-shot inference, which minimizes a Laplacian-regularization objective for few-shot tasks w.r.t assignment variables $\mathbf{Y}$, subject to simplex and integer constraints $\mathbf{y}_q \in \nabla_C$ and $\mathbf{y}_q \in \{0,1\}^C, \forall q$:

$$\begin{aligned}
\mathcal{E}(\mathbf{Y}) &= \mathbb{N}(\mathbf{Y}) + \frac{\lambda}{2}\mathbb{L}(\mathbf{Y}) \\
\mathbb{N}(\mathbf{Y}) &= \sum_{q=1}^{N}\sum_{c=1}^{C} y_{q,c}\, d(\mathbf{x}_q - \mathbf{m}_c) \\
\mathbb{L}(\mathbf{Y}) &= \frac{1}{2}\sum_{q,p} w(\mathbf{x}_q, \mathbf{x}_p)\|\mathbf{y}_q - \mathbf{y}_p\|^2
\end{aligned} \tag{4.1}$$

In (4.1), the first term $\mathbb{N}(\mathbf{Y})$ is minimized globally when each query point is assigned to the class of the nearest prototype $\mathbf{m}_c$ from the support set, using a distance metric $d(\mathbf{x}_q, \mathbf{m}_c)$, such as the Euclidean distance. In the **1-shot** setting, prototype $\mathbf{m}_c$ is the support example of class c, whereas in **multi-shot**, $\mathbf{m}_c$ can be the mean of the support examples. In fact, $\mathbf{m}_c$ can be further rectified by integrating information from the query features, as we will detail later in our experiments.

The second term $\mathbb{L}(\mathbf{Y})$ is the well-known Laplacian regularizer, which can be equivalently written as $\mathrm{tr}(\mathbf{Y}^t\mathbf{L}\mathbf{Y})$, where $\mathbf{L}$ is the Laplacian matrix[1] corresponding to affinity matrix $\mathbf{W} = [w(\mathbf{x}_q, \mathbf{x}_p)]$, and tr denotes the trace operator. Pairwise potential $w(\mathbf{x}_q, \mathbf{x}_p)$ evaluates the similarity between

[1] The Laplacian matrix corresponding to affinity matrix $\mathbf{W} = [w(\mathbf{x}_q, \mathbf{x}_p)]$ is $\mathbf{L} = \mathbf{D} - \mathbf{W}$, with $\mathbf{D}$ the diagonal matrix whose diagonal elements are given by: $D_q = \sum_p w(\mathbf{x}_q, \mathbf{x}_p)$.

feature vectors $\mathbf{x}_q$ and $\mathbf{x}_p$, and can be computed using some kernel function. The Laplacian term encourages nearby points ($\mathbf{x}_q$, $\mathbf{x}_p$) in the feature space to have the same latent label assignment, thereby regularizing predictions at query samples for few-shot tasks. As we will show later in our comprehensive experiments, the pairwise Laplacian term complements the unary potentials in $\mathbb{N}(\mathbf{Y})$, substantially increasing the predictive performance of few-shot learning across different networks, and various benchmark datasets with different levels of difficulty.

More generally, Laplacian regularization is widely used in the contexts of graph clustering (Shi & Malik, 2000; Von Luxburg, 2007a; Wang & Carreira-Perpinán, 2014; Ziko *et al.*, 2018) and semi-supervised learning (Belkin *et al.*, 2006; Weston *et al.*, 2012). For instance, popular spectral graph clustering techniques (Shi & Malik, 2000; Von Luxburg, 2007a) optimize the Laplacian term subject to partition-balance constraints. In this connection, our transductive inference can be viewed as a graph clustering of the query set, subject to supervision constraints from the support set.

Regularization parameter λ controls the trade-off between the two terms. It is worth noting that the recent nearest-prototype classification in (Wang *et al.*, 2019) corresponds to the particular case of $\lambda = 0$ of our model in (4.1). It assigns a query sample $\mathbf{x}_q$ to the label of the closest support prototype in the feature space, thereby minimizing $\mathbb{N}(\mathbf{Y})$:

$$y_{q,c^*} = 1 \quad \text{if} \quad c^* = \underset{c \in \{1,\ldots,C\}}{\arg\min}\; d(\mathbf{x}_q, \mathbf{m}_c) \tag{4.2}$$

4.2.2 Optimization

In this section, we propose an efficient bound-optimization technique for solving a relaxed version of our objective in (4.1), which guarantees convergence, while computing independent closed-form updates for each query sample in few-shot tasks. It is well known that minimizing pairwise functions over binary variables is NP-hard (Tian *et al.*, 2014), and a standard approach in the context of clustering algorithms is to relax the integer constraints, for instance, using a convex (Wang & Carreira-Perpinán, 2014) or a concave relaxation (Ziko *et al.*, 2018). In fact, by relaxing integer constraints $\mathbf{y}_q \in \{0,1\}^C$, our objective in (4.1) becomes a convex quadratic problem. However, this would require solving for the $N \times C$ assignment variables all together, with additional projections steps for handling the simplex constraints. In this work, we use a concave relaxation of the Laplacian-regularized objective in (4.1), which, as we will later show, yields fast independent and closed-form updates for each assignment variable, with convergence guarantee. Furthermore, it enables us to draw interesting connections between Laplacian regularization and attention mechanisms in few-shot learning (Vinyals *et al.*, 2016).

It is easy to verify that, for binary (integer) simplex variables, the Laplacian term in (4.1) can be written as follows, after some simple manipulations:

$$\mathbb{L}(\mathbf{Y}) = \sum_q D_q - \sum_{q,p} w(\mathbf{x}_q, \mathbf{x}_p)\mathbf{y}_q^t \mathbf{y}_p \tag{4.3}$$

where $D_q = \sum_p w(\mathbf{x}_q, \mathbf{x}_p)$ denotes the *degree* of query sample $\mathbf{x}_q$. By relaxing integer constraints $\mathbf{y}_q \in \{0,1\}^C$, the expression in Eq. (4.3) can be viewed as a *concave* relaxation[2] for Laplacian term $\mathbb{L}(\mathbf{Y})$ when symmetric affinity matrix $\mathbf{W} = [w(\mathbf{x}_q, \mathbf{x}_p)]$ is positive semi-definite. As we will see in the next paragraph, concavity is important to derive an efficient bound optimizer for our model, with independent and closed-form updates for each query sample. Notice that the first term in relaxation (4.3) is a constant independent of the soft (relaxed) assignment variables.

We further augment relaxation (4.3) with a convex negative-entropy barrier function $\mathbf{y}_q^t \log \mathbf{y}_q$, which avoids expensive projection steps and Lagrangian-dual inner iterations for the simplex constraints of each query point. Such a barrier[3] removes the need for extra dual variables for constraints $\mathbf{y}_q \geq 0$ by restricting the domain of each assignment variable to non-negative values, and yields closed-form updates for the dual variables of constraints $\mathbf{1}^t\mathbf{y}_q = 1$. Notice that this barrier function is null at the vertices of the simplex. Putting all together, and omitting the

[2] Equality (4.3) holds in for points on the vertices of the simplex, i.e., $\mathbf{y}_q \in \{0,1\}^C$, but is an approximation for points within the simplex (soft assignments), i.e., $\mathbf{y}_q \in]0,1[^C$.

[3] Note that entropy-like barriers are known in the context of Bregman-proximal optimization (Yuan *et al.*, 2017), and have well-known computational benefits when dealing with simplex constraints.

additive constant $\sum_q D_q$ in (4.3), we minimize the following concave-convex relaxation of our objective in (4.1) w.r.t soft assignment variables $\mathbf{Y}$, subject to simplex constraints $\mathbf{y}_q \in \nabla_C, \forall q$:

$$\mathcal{R}(\mathbf{Y}) = \mathbf{Y}^t \log \mathbf{Y} + \mathbb{N}(\mathbf{Y}) + \frac{\lambda}{2}\tilde{\mathbb{L}}(\mathbf{Y}) \tag{4.4}$$

where $\tilde{\mathbb{L}}(\mathbf{Y}) = -\sum_{q,p} w(\mathbf{x}_q, \mathbf{x}_p)\mathbf{y}_q^t\mathbf{y}_p$.

Bound optimization: In the following, we detail an iterative bound-optimization solution for relaxation (4.4). Bound optimization, often referred to as MM (Majorize-Minimization) framework (Lange, Hunter & Yang, 2000b; Zhang *et al.*, 2007), is a general optimization principle[4]. At each iteration, it updates the variable as the minimum of a *surrogate function*, i.e., an upper bound on the original objective, which is tight at the current iteration. This guarantees that the original objective does not increase at each iteration.

Re-arranging the soft assignment matrix $\mathbf{Y}$ in vector form $\mathbf{Y} = [\mathbf{y}_q] \in \mathbb{R}^{NC}$, relaxation $\tilde{\mathbb{L}}(\mathbf{Y})$ can be written conveniently in the following form:

$$\tilde{\mathbb{L}}(\mathbf{Y}) = -\sum_{q,p} w(\mathbf{x}_q, \mathbf{x}_p)\mathbf{y}_q^t\mathbf{y}_p = \mathbf{Y}^t\Psi\mathbf{Y} \tag{4.5}$$

with $\Psi = -\mathbf{W} \otimes \mathbf{I}$, where $\otimes$ denotes the Kronecker product and $\mathbf{I}$ is the $N \times N$ identity matrix. Note that Ψ is negative semi-definite for a positive semi-definite $\mathbf{W}$. Therefore, $\mathbf{Y}^t\Psi\mathbf{Y}$ is a concave function, and the first-order approximation of (4.5) at a current solution $\mathbf{Y}^i$ (i is the iteration index) gives the following tight upper bound on $\tilde{\mathbb{L}}(\mathbf{Y})$:

$$\tilde{\mathbb{L}}(\mathbf{Y}) = \mathbf{Y}^t\Psi\mathbf{Y} \leq (\mathbf{Y}^i)^t\Psi\mathbf{Y}^i + 2\,(\Psi\mathbf{Y}^i)^t(\mathbf{Y} - \mathbf{Y}^i) \tag{4.6}$$

[4] The general MM principle is widely used in machine learning in various problems as it enables to replace a difficult optimization problem with a sequence of easier sub-problems (Zhang *et al.*, 2007). Examples of well-known bound optimizers include expectation-maximization (EM) algorithms, the concave-convex procedure (CCCP) (Yuille & Rangarajan, 2001) and submodular-supermodular procedures (SSP) (Narasimhan & Bilmes, 2005), among many others.

Therefore, using unary potentials $\mathbb{N}(\mathbf{Y})$ and the negative entropy barrier in conjunction with the upper bound in (4.6), we obtain the following surrogate function $\mathcal{B}_i(\mathbf{Y})$ for relaxation $\mathcal{R}(\mathbf{Y})$ at current solution $\mathbf{Y}^i$:

$$\mathcal{R}(\mathbf{Y}) \leq \mathcal{B}_i(\mathbf{Y}) \overset{\text{c}}{=} \sum_{q=1}^{N} \mathbf{y}_q^t (\log(\mathbf{y}_q) + \mathbf{a}_q - \lambda \mathbf{b}_q^i) \tag{4.7}$$

where $\overset{\text{c}}{=}$ means equality up to an additive constant[5] that is independent of variable $\mathbf{Y}$, and $\mathbf{a}_q$ and $\mathbf{b}_q^i$ are the following C-dimensional vectors:

$$\mathbf{a}_q = [a_{q,1}, \dots, a_{q,C}]^t; \; a_{q,c} = d(\mathbf{x}_q, \mathbf{m}_c) \tag{4.8a}$$

$$\mathbf{b}_q^i = [b_{q,1}^i, \dots, b_{q,C}^i]^t; \; b_{q,c}^i = \sum_p w(\mathbf{x}_q, \mathbf{x}_p) y_{p,c}^i \tag{4.8b}$$

It is straightforward to verify that upper bound $\mathcal{B}_i(\mathbf{Y})$ is tight at the current iteration, i.e., $\mathcal{B}_i(\mathbf{Y}^i) = \mathcal{R}(\mathbf{Y}^i)$. This can be seen easily from the first-order approximation in (4.6). We iteratively optimize the surrogate function at each iteration i:

$$\mathbf{Y}^{i+1} = \arg\min_{\mathbf{Y}} \mathcal{B}_i(\mathbf{Y}) \tag{4.9}$$

Because of upper-bound condition $\mathcal{R}(\mathbf{Y}) \leq \mathcal{B}_i(\mathbf{Y}), \forall \mathbf{Y}$, tightness condition $\mathcal{B}_i(\mathbf{Y}^i) = \mathcal{R}(\mathbf{Y}^i)$ at the current solution, and the fact that $\mathcal{B}_i(\mathbf{Y}^{i+1}) \leq \mathcal{B}_i(\mathbf{Y}^i)$ due to minimization (4.9), it is easy to verify that updates (4.9) guarantee that relaxation $\mathcal{R}(\mathbf{Y})$ does not increase at each iteration:

$$\mathcal{R}(\mathbf{Y}^{i+1}) \leq \mathcal{B}_i(\mathbf{Y}^{i+1}) \leq \mathcal{B}_i(\mathbf{Y}^i) = \mathcal{R}(\mathbf{Y}^i)$$

Closed-form solutions of the surrogate functions: Notice that $\mathcal{B}_i(\mathbf{Y})$ is a sum of independent functions of each assignment variable. Therefore, we can solve (4.9) for each $\mathbf{y}_q$ independently,

[5] The additive constant in $\mathcal{B}_i(\mathbf{Y})$ is a term that depends only on $\mathbf{Y}^i$. This term comes from the Laplacian upper bound in (4.6).

while satisfying the simplex constraint:

$$\min_{\mathbf{y}_q \in \nabla_C} \mathbf{y}_q^t(\log(\mathbf{y}_q) + \mathbf{a}_q - \lambda \mathbf{b}_q^i), \, \forall q \tag{4.10}$$

The negative entropy barrier term $\mathbf{y}_q^t \log \mathbf{y}_q$ in (4.10) restricts $\mathbf{y}_q$ to be non-negative, removing the need of extra dual variables for the constraints $\mathbf{y}_q > 0$. Also, simplex constraint $\mathbf{1}^t \mathbf{y}_q = 1$ is affine. Thus, the solution of the following Karush-Kuhn-Tucker (KKT) condition provide the minimum of (4.10):

$$\log \mathbf{y}_q + \mathbf{a}_q - \lambda \mathbf{b}_q^i + \beta \mathbf{1} = 0 \tag{4.11}$$

with β the Lagrange multiplier for the simplex constraint. This provides, for each q, closed-form solutions for both the primal and dual variables, yielding the following independent updates of the assignment variables:

$$\mathbf{y}_q^{i+1} = \frac{\exp(-\mathbf{a}_q^i + \lambda \mathbf{b}_q^i)}{\mathbf{1}^t \exp(-\mathbf{a}_q^i + \lambda \mathbf{b}_q^i)} \; \forall \, q \tag{4.12}$$

4.2.3 Proposed Algorithm

The overall proposed algorithm is simplified in Algorithm 4.1. Once the network function f_θ is learned using the base dataset $\mathbb{X}_{\text{base}}$, our algorithm proceeds with the extracted features $\mathbf{x}_q$. Before the iterative bound updates, each soft assignment $\mathbf{y}_q^1$ is initialized as a softmax probability of $\mathbf{a}_q$, which is based on the distances to prototypes $\mathbf{m}_c$. The iterative bound optimization is guaranteed to converge, typically less than 15 iterations in our experiments (Figure 4.2). Also the independent point-wise bound updates yield a parallel structure of the algorithm, which makes it very efficient (and convenient for large-scale few-shot tasks). We refer to our method as LaplacianShot in the experiments.

Link to attention mechanisms: Our Laplacian-regularized model has interesting connection to the popular attention mechanism in (Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser & Polosukhin, 2017). In fact, MatchingNet (Vinyals *et al.*, 2016) predicted the labels of the query samples $\mathbf{x}_q$ as a linear combination of the support labels. The expression of $b_{q,c}^i$

Table 4.1 Average accuracy (in %) in *mini*ImageNet and *tiered*ImageNet.

Methods	Network	***mini*ImageNet**		***tiered*ImageNet**	
		1-shot	**5-shot**	**1-shot**	**5-shot**
MAML (Finn *et al.*, 2017)	ResNet-18	49.61 ± 0.92	65.72 ± 0.77	-	-
Chen (Chen *et al.*, 2019)	ResNet-18	51.87 ± 0.77	75.68 ± 0.63	-	-
RelationNet (Sung *et al.*, 2018)	ResNet-18	52.48 ± 0.86	69.83 ± 0.68	-	-
MatchingNet (Vinyals *et al.*, 2016)	ResNet-18	52.91 ± 0.88	68.88 ± 0.69	-	-
ProtoNet (Snell *et al.*, 2017)	ResNet-18	54.16 ± 0.82	73.68 ± 0.65	-	-
Gidaris (Gidaris & Komodakis, 2018)	ResNet-15	55.45 ± 0.89	70.13 ± 0.68	-	-
SNAIL (Mishra *et al.*, 2018)	ResNet-15	55.71 ± 0.99	68.88 ± 0.92	-	-
AdaCNN (Munkhdalai, Yuan, Mehri & Trischler, 2018)	ResNet-15	56.88 ± 0.62	71.94 ± 0.57	-	-
TADAM (Oreshkin *et al.*, 2018)	ResNet-15	58.50 ± 0.30	76.70 ± 0.30	-	-
CAML (Jiang, Havaei, Varno, Chartrand, Chapados & Matwin, 2019)	ResNet-12	59.23 ± 0.99	72.35 ± 0.71	-	-
TPN (Yanbin *et al.*, 2019)	ResNet-12	59.46	75.64	-	-
TEAM (Qiao *et al.*, 2019)	ResNet-18	60.07	75.90	-	-
MTL (Sun, Liu, Chua & Schiele, 2019)	ResNet-18	61.20 ± 1.80	75.50 ± 0.80	-	-
VariationalFSL (Zhang, Zhao, Ni, Xu & Yang, 2019)	ResNet-18	61.23 ± 0.26	77.69 ± 0.17	-	-
Transductive tuning (Dhillon *et al.*, 2020)	ResNet-12	62.35 ± 0.66	74.53 ± 0.54	-	-
MetaoptNet (Lee, Maji, Ravichandran & Soatto, 2019)	ResNet-18	62.64 ± 0.61	78.63 ± 0.46	65.99 ± 0.72	81.56 ± 0.53
SimpleShot (Wang *et al.*, 2019)	ResNet-18	63.10 ± 0.20	79.92 ± 0.14	69.68 ± 0.22	84.56 ± 0.16
CAN+T (Hou *et al.*, 2019)	ResNet-12	67.19 ± 0.55	80.64 ± 0.35	73.21 ± 0.58	84.93 ± 0.38
LaplacianShot (ours)	ResNet-18	72.11 ± 0.19	82.31 ± 0.14	78.98 ± 0.21	86.39 ± 0.16
Qiao (Qiao, Liu, Shen & Yuille, 2018)	WRN	59.60 ± 0.41	73.74 ± 0.19	-	-
LEO (Rusu *et al.*, 2019)	WRN	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09
ProtoNet (Snell *et al.*, 2017)	WRN	62.60 ± 0.20	79.97 ± 0.14	-	-
CC+rot (Gidaris, Bursuc, Komodakis, Pérez & Cord, 2019)	WRN	62.93 ± 0.45	79.87 ± 0.33	70.53 ± 0.51	84.98 ± 0.36
MatchingNet (Vinyals *et al.*, 2016)	WRN	64.03 ± 0.20	76.32 ± 0.16	-	-
FEAT (Ye *et al.*, 2020)	WRN	65.10 ± 0.20	81.11 ± 0.14	70.41 ± 0.23	84.38 ± 0.16
Transductive tuning (Dhillon *et al.*, 2020)	WRN	65.73 ± 0.68	78.40 ± 0.52	73.34 ± 0.71	85.50 ± 0.50
SimpleShot (Wang *et al.*, 2019)	WRN	65.87± 0.20	82.09 ± 0.14	70.90 ± 0.22	85.76 ± 0.15
SIB (Hu *et al.*, 2020)	WRN	70.0 ± 0.6	79.2 ± 0.4	-	-
BD-CSPN (Liu, Song & Qin, 2019)	WRN	70.31 ± 0.93	81.89 ± 0.60	78.74 ± 0.95	86.92 ± 0.63
LaplacianShot (ours)	WRN	74.86 ± 0.19	84.13 ± 0.14	80.18 ± 0.21	87.56± 0.15
SimpleShot (Wang *et al.*, 2019)	MobileNet	61.55 ± 0.20	77.70 ± 0.15	69.50 ± 0.22	84.91 ± 0.15
LaplacianShot (ours)	MobileNet	70.27 ± 0.19	80.10 ± 0.15	79.13 ± 0.21	86.75 ± 0.15
SimpleShot (Wang *et al.*, 2019)	DenseNet	65.77 ± 0.19	82.23 ± 0.13	71.20 ± 0.22	86.33 ± 0.15
LaplacianShot (ours)	DenseNet	75.57 ± 0.19	84.72 ± 0.13	80.30 ± 0.22	87.93 ± 0.15

Table 4.2 Results for CUB and cross-domain results on *mini*Imagenet → CUB.

Methods	**Network**	**CUB**		***mini*Imagenet → CUB**	
		1-shot	**5-shot**	**1-shot**	**5-shot**
MatchingNet (Vinyals *et al.*, 2016)	ResNet-18	73.49	84.45	-	53.07
MAML (Finn *et al.*, 2017)	ResNet-18	68.42	83.47	-	51.34
ProtoNet (Snell *et al.*, 2017)	ResNet-18	72.99	86.64	-	62.02
RelationNet (Sung *et al.*, 2018)	ResNet-18	68.58	84.05	-	57.71
Chen (Chen *et al.*, 2019)	ResNet-18	67.02	83.58	-	65.57
SimpleShot (Wang *et al.*, 2019)	ResNet-18	70.28	86.37	48.56	65.63
LaplacianShot(ours)	ResNet-18	80.96	88.68	55.46	66.33

Table 4.3 Average accuracy (in %) in iNat benchmark for SimpleShot (Wang *et al.*, 2019) and the proposed LaplacianShot. The best results are reported in bold font. Note that, for iNat, we do not utilize the rectified prototypes. [The best reported result of (Wertheimer & Hariharan, 2019) with ResNet50 is: Per Class: 46.04%, Mean: 51.25%.]

Methods	**Network**	**UN**		**L2**		**CL2**	
		Per Class	**Mean**	**Per Class**	**Mean**	**Per Class**	**Mean**
SimpleShot	ResNet-18	55.80	58.56	57.15	59.56	56.35	58.63
LaplacianShot	ResNet-18	62.80	66.40	58.72	61.14	58.49	60.81
SimpleShot	ResNet-50	58.45	61.07	59.68	61.99	58.83	60.98
LaplacianShot	ResNet-50	65.96	69.13	61.40	63.66	61.08	63.18
SimpleShot	WRN	62.44	65.08	64.26	66.25	63.03	65.17
LaplacianShot	WRN	71.55	74.97	65.78	67.82	65.32	67.43

that we obtained in (4.8b), which stems from our bound optimizer and the concave relaxation of the Laplacian, also takes the form of a combination of labels at each iteration i in our model: $b^i_{q,c} = \sum_p w(\mathbf{x}_q, \mathbf{x}_p) y^i_{p,c}$. However, there are important differences with (Vinyals *et al.*, 2016): First, the attention in our formulation is non-parametric as it considers only the feature relationships among the query samples in $\mathbb{X}_\text{q}$, not the support examples. Second, unlike our approach, the attention mechanism in (Vinyals *et al.*, 2016) is employed during training for learning embedding function f_θ with a meta-learning approach.

4.3 Experiments

In this section, we describe our experimental setup. An implementation of our LaplacianShot is publicly available[6].

4.3.1 Datasets

We used five benchmarks for few-shot classification: *mini*ImageNet, *tiered*ImageNet, CUB, cross-domain CUB (with base training on *mini*ImageNet) and iNat.

[6] https://github.com/imtiazziko/LaplacianShot

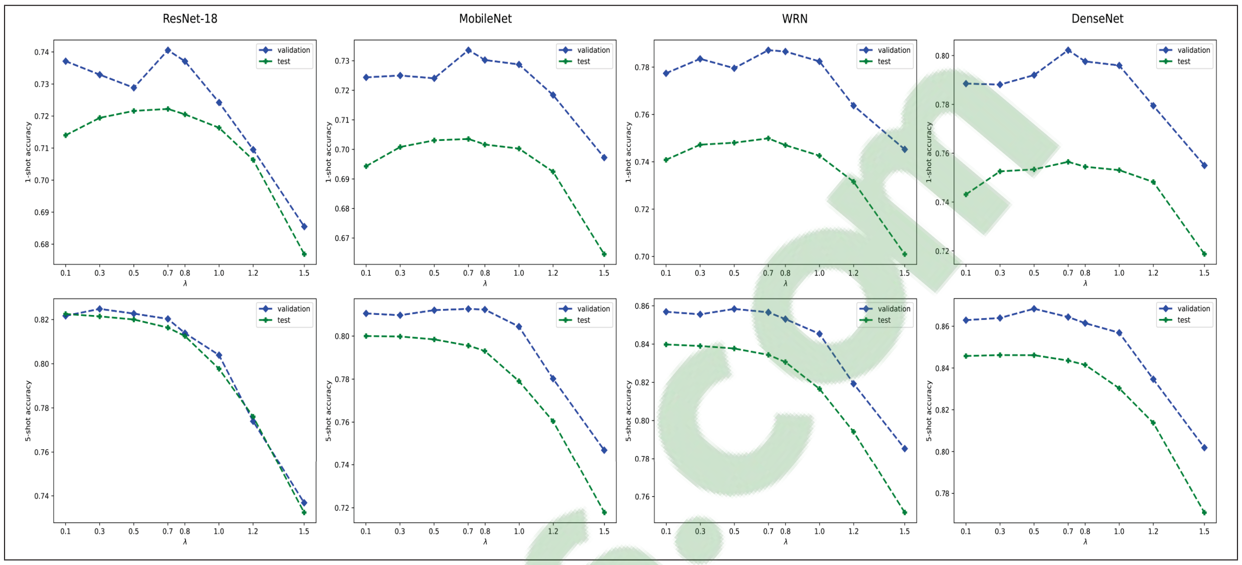


Figure 4.1 We tune regularization parameter λ over values ranging from 0.1 to 1.5. In the above plots, we show the impact of choosing λ on both validation and test accuracies. The values of λ based on the best validation accuracies correspond to good accuracies in the test classes. The results are shown for different networks on *mini*ImageNet dataset, for both 1-shot (top row) and 5-shot (bottom row).

The ***mini*ImageNet** benchmark is a subset of the larger ILSVRC-12 dataset (Russakovsky *et al.*, 2015). It has a total of 60,000 color images with 100 classes, where each class has 600 images of size 84×84, following (Vinyals *et al.*, 2016). We use the standard split of 64 base, 16 validation and 20 test classes (Ravi & Larochelle, 2017; Wang *et al.*, 2019). The ***tiered*ImageNet** benchmark (Ren, Triantafillou, Ravi, Snell, Swersky, Tenenbaum, Larochelle & Zemel, 2018) is also a subset of ILSVRC-12 dataset but with 608 classes instead. We follow standard splits with 351 base, 97 validation and 160 test classes for the experiments. The images are also resized to 84×84 pixels. **CUB**-200-2011 (Wah, Branson, Welinder, Perona & Belongie, 2011) is a fine-grained image classification dataset. We follow (Chen *et al.*, 2019) for few-shot classification on CUB, which splits into 100 base, 50 validation and 50 test classes for the experiments. The images are also resized to 84×84 pixels, as in *mini*ImageNet. The **iNat** benchmark, introduced recently for few-shot classification in (Wertheimer & Hariharan, 2019), contains images of 1,135 animal species. It introduces a more challenging few-shot scenario, with different numbers of support examples per class, which simulates more realistic class-imbalance scenarios, and with semantically related classes that are not easily separable. Following (Wertheimer & Hariharan, 2019), the dataset is split into 908 base classes and 227 test classes, with images of size 84×84.

4.3.2 Evaluation Protocol

In the case of ***mini*ImageNet**, **CUB** and ***tiered*ImageNet**, we evaluate 10,000 five-way 1-shot and five-way 5-shot classification tasks, randomly sampled from the test classes, following standard few-shot evaluation settings (Rusu *et al.*, 2019; Wang *et al.*, 2019). This means that, for each of the five-way few-shot tasks, $C = 5$ classes are randomly selected, with $|\mathbb{X}_s^c| = 1$ (1-shot) and $|\mathbb{X}_s^c| = 5$ (5-shot) examples selected per class, to serve as support set $\mathbb{X}_s$. Query set $\mathbb{X}_q$ contains 15 images per class. Therefore, the evaluation is performed over $N = 75$ query images per task. The average accuracy of these 10,000 few shot tasks are reported along with the 95% confidence interval. For the **iNat** benchmark, the number of support examples $|\mathbb{X}_s^c|$ per class varies. We performed 227-way multi-shot evaluation, and report the top-1 accuracy averaged over the test images per class (Per Class in Table 4.3), as well as the average over all test images (Mean in Table 4.3), following the same procedure as in (Wang *et al.*, 2019; Wertheimer & Hariharan, 2019).

4.3.3 Network Models

We evaluate LaplacianShot on four different backbone network models to learn feature extractor f_θ:

ResNet-18/50 is based on the deep residual network architecture (He, Zhang, Ren & Sun, 2016), where the first two down-sampling layers are removed, setting the stride to 1 in the first convolutional layer and removing the first max-pool layer. The first convolutional layer is used with a kernel of size 3×3 instead of 7×7. ResNet-18 has 8 basic residual blocks, and ResNet-50 has 16 bottleneck blocks. For all the networks, the dimension of the extracted features is 512.
MobileNet (Howard, Zhu, Chen, Kalenichenko, Wang, Weyand, Andreetto & Adam, 2017) was initially proposed as a light-weight convolutional network for mobile-vision applications. In our

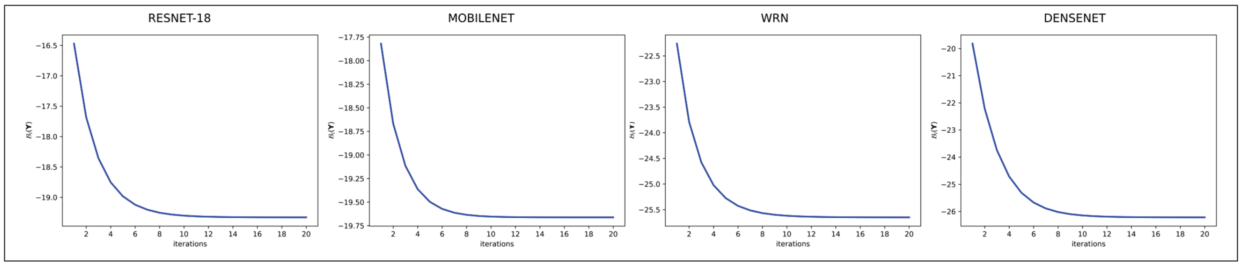


Figure 4.2 Convergence of Algorithm 4.1: Bounds $\mathcal{B}_i(\mathbf{Y})$ vs. iteration numbers for features from different networks. Here, the plots are produced by setting $\lambda = 1.0$, for a single 5-way 5 shot task from the *mini*ImageNet test set.

setting, we remove the first two down-sampling operations, which results in a feature embedding of size 1024. **WRN** (Zagoruyko & Komodakis, 2016) widens the residual blocks by adding more convolutional layers and feature planes. In our case, we used 28 convolutional layers, with a widening factor of 10 and an extracted-feature dimension of 640. Finally, we used the standard 121-layer **DenseNet** (Huang, Liu, Van Der Maaten & Weinberger, 2017), omitting the first two down-sampling layers and setting the stride to 1. We changed the kernel size of the first convolutional layer to 3×3. The extracted feature vector is of dimension 1024.

4.3.4 Implementation Details

Network model training: We trained the network models using the standard cross-entropy loss on the base classes, with a label-smoothing (Szegedy, Vanhoucke, Ioffe, Shlens & Wojna, 2016) parameter set to 0.1. Note that the base training did not involve any meta-learning or episodic-training strategy. We used the SGD optimizer to train the models, with mini-batch size set to 256 for all the networks, except for WRN and DenseNet, where we used mini-batch sizes of 128 and 100, respectively. We used two 16GB P100 GPUs for network training with base classes. For this base training, we used the data augmentation procedure of (He *et al.*, 2016), along with color jitter, similarly to (Chen *et al.*, 2019). For *mini*ImageNet, CUB and *tiered*ImageNet, we used early stopping by evaluating the the nearest-prototype classification accuracy on the validation classes, with **L2** normalized features.

Prototype estimation and feature transformation: During the inference on test classes, SimpleShot (Wang *et al.*, 2019) performs the following feature transformations: **L2** normalization, $\mathbf{x}_q := \mathbf{x}_q / \|\mathbf{x}_q\|_2$ and **CL2**, which computes the mean of the base class features $\bar{\mathbf{x}} = \frac{1}{|\mathbb{X}_{\text{base}}|} \sum_{\mathbf{x} \in \mathbb{X}_{\text{base}}} \mathbf{x}$ and centers the extracted features as $\mathbf{x}_q := \mathbf{x}_q - \bar{\mathbf{x}}$, which is followed by an L2 normalization. We report the results in Table 4.1 and 4.2 with CL2 normalized features. In Table 4.3 for the iNat dataset, we provide the results with both normalized and unnormalized (**UN**) features for a comparative analysis. We reproduced the results of SimpleShot with our trained network models. In the **1-shot** setting, prototype $\mathbf{m}_c$ is just the support example $\mathbf{x}_q \in \mathbb{X}_{\text{s}}^c$ of class c, whereas in **multi-shot**, $\mathbf{m}_c$ is the simple mean of the support examples of class c. Another option is to use *rectified* prototypes, i.e., a weighted combination of features from both the support examples in $\mathbb{X}_{\text{s}}^c$ and query samples in $\mathbb{X}_{\text{q}}^c$, which are initially predicted as belonging to class c using Eq. (4.2):

$$\tilde{\mathbf{m}}_c = \frac{1}{|\mathbb{X}_{\text{s}}^c| + |\mathbb{X}_{\text{q}}^c|} \sum_{\mathbf{x}_p \in \{\mathbb{X}_{\text{s}}^c, \mathbb{X}_{\text{q}}^c\}} \frac{\exp(cos(\mathbf{x}_p, \mathbf{m}_c))}{\sum_{c=1}^{C} \exp(cos(\mathbf{x}_p, \mathbf{m}_c))} \mathbf{x}_p,$$

where cos denotes the cosine similarity. And, for a given few-shot task, we compute the cross-domain shift Δ as the difference between the mean of features within the support set and the mean of features within the query set: $\Delta = \frac{1}{|\mathbb{X}_{\text{s}}|} \sum_{\mathbf{x}_p \in \mathbb{X}_{\text{s}}} \mathbf{x}_p - \frac{1}{|\mathbb{X}_{\text{q}}|} \sum_{\mathbf{x}_q \in \mathbb{X}_{\text{q}}} \mathbf{x}_q$. Then, we rectify each query point $\mathbf{x}_p \in \mathbb{X}_{\text{q}}$ in the few-shot task as follows: $\mathbf{x}_p = \mathbf{x}_p + \Delta$. This shift correction is similar to the prototype rectification in (Liu *et al.*, 2019). Note that our LaplacianShot model in Eq. (4.1) is agnostic to the way of estimating the prototypes: It can be used either with the standard prototypes ($\mathbf{m}_c$) or with the rectified ones ($\tilde{\mathbf{m}}_c$). We report the results of LaplacianShot with the rectified prototypes in Table 4.1 and 4.2, for *mini*Imagenet, *tiered*Imagenet and CUB. We do not report the results with the rectified prototypes in Table 4.3 for iNat, as rectification drastically worsen the performance.

For $\mathbf{W}$, we used the k-nearest neighbor affinities as follows: $w(\mathbf{x}_q, \mathbf{x}_p) = 1$ if $\mathbf{x}_p$ is within the k nearest neighbor of $\mathbf{x}_q$, and $w(\mathbf{x}_q, \mathbf{x}_p) = 0$ otherwise. In our experiments, k is simply chosen from three typical values (3, 5 or 10) tuned over 500 few-shot tasks from the base training classes (i.e., we did not use test data for choosing k). We used $k = 3$ for *mini*ImageNet, CUB and *tiered*ImageNet and $k = 10$ for iNat benchmark. Regularization parameter λ is chosen based on the validation class accuracy for *mini*ImageNet, CUB and *tiered*ImageNet. This will be discussed in more details in section 4.3.6. For the iNat experiments, we simply fix $\lambda = 1.0$, as there is no validation set for this benchmark.

Table 4.4 Ablation study on the effect of each term corresponding to nearest prototype $\mathbb{N}(\mathbf{Y})$, Laplacian $\mathbb{L}(\mathbf{Y})$ and rectified prototype $\tilde{\mathbf{m}}_c$. Results are reported with ResNet-18 network. Note that, the Laplacian regularization $\mathbb{L}(\mathbf{Y})$ improve the results consistently.

			***mini*-ImageNet**		***tiered*-ImageNet**		**CUB**	
$\mathbb{N}(\mathbf{Y})$	$\mathbb{L}(\mathbf{Y})$	$\tilde{\mathbf{m}}_c$	1-shot	5-shot	1-shot	5-shot	1shot	5-shot
✓	✗	✗	63.10	79.92	69.68	84.56	70.28	86.37
✓	✓	✗	66.20	80.75	72.89	85.25	74.46	86.86
✓	✗	✓	69.74	82.01	76.73	85.74	78.76	88.55
✓	✓	✓	72.11	82.31	78.98	86.39	80.96	88.68

4.3.5 Results

We evaluated LaplacianShot over five different benchmarks, with different scenarios and difficulties: Generic image classification, fine-grained image classification, cross-domain adaptation, and imbalanced class distributions. We report the results of LaplacianShot for *mini*ImageNet, *tiered*ImageNet, CUB and iNat datasets, in Tables 4.1, 4.2 and 4.3, along with comparisons with state-of-the-art methods.

Generic image classification: Table 4.1 reports the results of generic image classification for the standard *mini*ImageNet and *tiered*ImageNet few-shot benchmarks. We can clearly observe that LaplacianShot outperforms state-of-the-art methods by large margins, with gains that are consistent across different settings and network models. It is worth mentioning that, for challenging scenarios, e.g., 1-shot with low-capacity models, LaplacianShot outperforms complex meta-learning methods by more than 9%. For instance, compared to well-known MAML (Finn *et al.*, 2017) and ProtoNet (Snell *et al.*, 2017), and to the recent MetaoptNet (Lee *et al.*, 2019), LaplacianShot brings improvements of nearly 22%, 17%, and 9%, respectively, under the same evaluation conditions. Furthermore, it outperforms the very recent transductive approaches in (Dhillon *et al.*, 2020; Liu *et al.*, 2019; Yanbin *et al.*, 2019) by significant margins. With better learned features with WRN and DenseNet, LaplacianShot brings significant performance boosts, yielding state-of-the art results in few-shot classification, without meta-learning.

Fine-grained image classification: Table 4.2 reports the results of fine-grained few-shot classification on CUB, with Resnet-18 network. LaplacianShot outperforms the best performing method in this setting by a 7% margin.

Cross-domain (mini-ImageNet → CUB): We perform the very interesting few-shot experiment, with a cross-domain scenario, following the setting in (Chen *et al.*, 2019). We used the ResNet-18 model trained on the *mini*Imagenet base classes, while evaluation is performed on CUB few-shot tasks, with 50 test classes. Table 4.2 (rightmost column) reports the results. In this cross-domain setting, and consistently with the standard settings, LaplacianShot outperforms complex meta-learning methods by substantial margins.

Imbalanced class distribution: Table 4.3 reports the results for the more challenging, class-imbalanced iNat benchmark, with different numbers of support examples per class and, also, with high visual similarities between the different classes, making class separation difficult. To our knowledge, only (Wang *et al.*, 2019; Wertheimer & Hariharan, 2019) report performances on this benchmark, and SimpleShot (Wang *et al.*, 2019) represents the state-of-the-art. We compared with SimpleShot using unnormalized extracted features (UN), L2 and CL2 normalized features. Our Laplacian regularization yields significant improvements, regardless of the network model and feature normalization. However, unlike SimpleShot, our method reaches its best performance with the unnormalized features. Note that, for iNat, we did not use the rectified prototypes. These results clearly highlight the benefit Laplacian regularization brings in challenging class-imbalance scenarios.

Table 4.5 Average inference time (in seconds) for the 5-shot tasks in *mini*Imagenet dataset.

Methods	**inference time**
SimpleShot (Wang *et al.*, 2019)	0.009
Transductive tuning (Dhillon *et al.*, 2020)	20.7
LaplacianShot (ours)	0.012

4.3.6 Ablation Study

Choosing the Value of λ: In LaplacianShot, we need to choose the value of regularization parameter λ, which controls the trade-off between the nearest-prototype classifier term $\mathbf{a}_q$ and Laplacian regularizer $\mathbf{b}_q^i$. We tuned this parameter using the validation classes by sampling 500 few-shot tasks. LaplacianShot is used in each few-shot task with the following values of λ: $[0.1, 0.3, 0.5, 0.7, 0.8, 1.0, 1.2, 1.5]$. The best λ corresponding to the best average 1-shot and 5-shot accuracy over validation classes/data is selected for inference over the test classes/data. To examine experimentally whether the chosen values of λ based on the best validation accuracies correspond to good accuracies in the test classes, we plotted both the validation and test class accuracies vs. different values of λ for *mini*ImageNet (Figure 4.1). The results are intuitive, with a consistent trend in both 1-shot and 5-shot settings. Particularly, for 1-shot tasks, $\lambda = 0.7$ provides the best results in both validation and test accuracies. In 5-shot tasks, the best test results are obtained mostly with $\lambda = 0.1$, while the best validation accuracies were reached with higher values of λ. Nevertheless, we report the results of LaplacianShot with the values of λ chosen based on the best validation accuracies.

Effects of Laplacian regularization: We conducted an ablation study on the effect of each term in our model, i.e., nearest-prototype classifier $\mathbb{N}(\mathbf{Y})$ and Laplacian regularizer $\mathbb{L}(\mathbf{Y})$. We also examined the effect of using prototype rectification, i.e., $\tilde{\mathbf{m}}_c$ instead of $\mathbf{m}_c$. Table 4.4 reports the results, using the ResNet-18 network. The first row corresponds to the prediction of the nearest neighbor classifier ($\lambda = 0$), and the second shows the effect of adding Laplacian regularization. In the 1-shot case, the latter boosts the performances by at least 3%. Prototype rectification (third and fourth rows) also boosts the performances. Again, in this case, the improvement that the Laplacian term brings is significant, particularly in the 1-shot case (2 to 3%).

Convergence of transductive LaplacianShot inference: The proposed algorithm belongs to the family of bound optimizers or MM algorithms. In fact, the MM principle can be viewed as a generalization of expectation-maximization (EM). Therefore, in general, MM algorithms inherit the monotonicity and convergence properties of EM algorithms (Vaida, 2005), which are well-studied in the literature. In fact, Theorem 3 in (Vaida, 2005) states a simple condition for convergence of the general MM procedure, which is almost always satisfied in practice: The surrogate function has a unique global minimum. In Fig. 4.2, we plotted surrogates $\mathcal{B}_i(\mathbf{Y})$, up to a constant, i.e., Eq. (4.7), as functions of the iteration numbers, for different networks. One can see that the value of $\mathcal{B}_i(\mathbf{Y})$ decreases monotonically at each iteration, and converges, typically, within less than 15 iterations.

Inference time: We computed the average inference time required for each 5-shot task. Table 4.5 reports these inference times for *mini*ImageNet with the WRN network. The purpose of this is to check whether there exist a significant computational overhead added by our Laplacian-regularized transductive inference, in comparison to inductive inference. Note that the computational complexity of the proposed inference is $O(NkC)$ for a few-shot task, where k is the neighborhood size for affinity matrix $\mathbf{W}$. The inference time per few-shot task for LaplacianShot is close to inductive SimpleShot run-time (LaplacianShot is only 1-order of magnitude slower), and is 3-order-of-magnitude faster than the transductive fine-tuning in (Dhillon *et al.*, 2020).

4.4 Conclusion

Without meta-learning, we provide state-of-the-art results, outperforming significantly a large number of sophisticated few-shot learning methods, in all benchmarks. Our transductive inference is a simple constrained graph clustering of the query features. It can be used in conjunction with any base-class training model, consistently yielding improvements. Our results are in line with several recent baselines (Chen *et al.*, 2019; Dhillon *et al.*, 2020; Wang *et al.*, 2019) that reported competitive performances, without resorting to complex meta-learning strategies. This recent line of simple methods emphasizes the limitations of current few-shot benchmarks, and questions the viability of a large body of convoluted few-shot learning techniques in the recent literature. As pointed out in Fig. 1 in (Dhillon *et al.*, 2020), the progress made by an abundant recent few-shot literature, mostly based on meta-learning, may be illusory. Classical and simple regularizers, such as the entropy in (Dhillon *et al.*, 2020) or our Laplacian term, well-established in semi-supervised learning and clustering, achieve outstanding performances. We do not claim to hold the ultimate solution for few-shot learning, but we believe that our model-agnostic transductive inference should be used as a strong baseline for future few-shot learning research.

CONCLUSION AND RECOMMENDATIONS

In conclusion, the contributions of the thesis are elegant and flexible approaches with an efficient and scalable bound optimization algorithm for joint clustering and prototype estimation including density mode updates for large scale clustering applications; to provide a variational fair clustering method which flexibly avoid bias against sensitive demographic groups during clustering of both prototype based and graph based in a single formulation; And to solve the challenging few-shot learning problems with simple Laplacian regularized prototype estimation approach without resorting to complicated meta-learning process. The proposed methods are shown to be very effective and efficient in terms of performance measures validated with comprehensive experiments in different benchmarks.

The first contribution: Scalable Laplacian K-modes (SLK) is a joint graph clustering and density mode estimation method. A concave-convex relaxation of the problem proposed to yields parallel thus distributed algorithm for large scale and high dimensional clustering datasets. The proposed bound optimizer can be trivially distributed with guaranteed convergence properties. The formulation also avoid the need of storing a full square affinity matrix and computing its eigenvalue decomposition unlike spectral relaxation. Also the expensive projection steps and Lagrangian-dual inner iterates for the simplex constraints of each point are avoided unlike convex relaxation. Furthermore, unlike mean-shift, our density-mode estimation does not require inner-loop gradient-ascent iterates which yields modes that are valid data points in the input set as a byproduct of bound updates. We report comprehensive experiments over various data sets, which show that our algorithm achieve very competitive performances in term of both optimization and clustering quality.

As a second contribution, we investigate a general variational formulation of fair clustering, which can integrate fairness constraints with a large class of clustering objectives. Unlike the existing existing combinatorial and spectral fair clustering approaches, the proposed variational formulation enables to control the trade-off between the fairness and clustering terms. We derive a general tight upper bound based on a concave-convex decomposition of our fairness term, its Lipschitz-gradient property and the Pinsker inequality. The upper bound can be optimized jointly with various clustering objectives, including prototype-based, such as K-means and K-median, or graph-based such as Normalized Cut. The algorithm can be easily distributed for large-scale data sets due to the independent update for each assignment variable. We show the effectiveness, flexibility and scalability of our approach through comprehensive evaluations and comparisons to the existing methods over several data sets.

As a third contribution, we investigate few-shot learning problem. We propose a Laplacian-regularization objective for few-shot tasks, which integrates two types of potentials having nearest prototype classification combined with pairwise Laplacian potentials encouraging nearby query samples to have consistent predictions. Following the standard experimental setting for few-shot learning, our LaplacianShot technique outperforms state-of-the-art methods significantly, while using simple cross-entropy training on the base classes instead of meta-learning. The significant boost in accuracy is consistent on standard few-shot benchmarks: *mini*/*tiered* ImageNet, CUB and on the recent challenging iNat benchmark, across various network models.

REFERENCES

Arora, S., Hazan, E. & Kale, S. (2005). Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pp. 339–348.

Arthur, D. & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. *ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035.

Backurs, A., Indyk, P., Onak, K., Schieber, B., Vakilian, A. & Wagner, T. (2019). Scalable fair clustering. *International conference on machine learning (ICML)*, 405–413.

Belkin, M. & Niyogi, P. (2001). Laplacian eigenmaps and spectral techniques for embedding and clustering. *NIPS*, 14, 585–591.

Belkin, M., Niyogi, P. & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7, 2399-2434.

Ben Salah, M., Ben Ayed, I., Yuan, J. & Zhang, H. (2014). Convex-relaxed kernel mapping for image segmentation. *IEEE Transactions on Image Processing*, 23(3), 1143–1153.

Bengio, Y., Paiement, J.-f., Vincent, P., Delalleau, O., Roux, N. L. & Ouimet, M. (2004). Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Neural Information Processing Systems (NIPS)*, pp. 177–184.

Bera, S., Chakrabarty, D., Flores, N. & Negahbani, M. (2019). Fair algorithms for clustering. *Advances in Neural Information Processing Systems*, pp. 4955–4966.

Boyd, Z. M., Bae, E., Tai, X.-C. & Bertozzi, A. L. (2018). Simplified energy landscape for modularity using total variation. *SIAM Journal on Applied Mathematics*, 78(5), 2439–2464.

Boykov, Y. & Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9), 1124–1137.

Boykov, Y., Veksler, O. & Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11), 1222–1239.

Buolamwini, J. & Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. *Conference on Fairness, Accountability and Transparency*, pp. 77–91.

Carreira-Perpiñán, M. Á. (2007). Gaussian mean-shift is an EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5), 767-776.

Carreira-Perpiñán, M. Á. (2015). A review of mean-shift algorithms for clustering. *arXiv preprint arXiv:1503.00687*.

Carreira-Perpiñán, M. Á. & Wang, W. (2013). The K-modes algorithm for clustering. *arXiv preprint arXiv:1304.6478*.

Celis, L. E., Keswani, V., Straszak, D., Deshpande, A., Kathuria, T. & Vishnoi, N. K. (2018). Fair and Diverse DPP-Based Data Summarization. *International Conference on Machine Learning (ICML)*, pp. 715–724.

Chen, C., Liu, H., Metaxas, D. & Zhao, T. (2014). Mode estimation for high dimensional discrete tree graphical models. *Neural Information Processing Systems (NIPS)*, pp. 1323–1331.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F. & Huang, J.-B. (2019). A Closer Look at Few-shot Classification. *International Conference on Learning Representations*.

Chen, X., Zhexue Haung, J., Nie, F., Chen, R. & Wu, Q. (2017). A self-balanced min-cut algorithm for image clustering. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2061–2069.

Chierichetti, F., Kumar, R., Lattanzi, S. & Vassilvitskii, S. (2017). Fair Clustering Through Fairlets. *Neural Information Processing Systems (NeurIPS)*, pp. 5036–5044.

Chitta, R., Jin, R., Havens, T. C. & Jain, A. K. (2011). Approximate kernel k-means: Solution to large scale kernel clustering. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 895–903.

Comaniciu, D. & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5), 603-619.

Comaniciu, D., Ramesh, V. & Meer, P. (2003). Kernel-based object tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 25(5), 564–577.

Csiszar, I. & Körner, J. (2011). *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press.

Dengyong, Z., Bousquet, O., Lal, T. N., Weston, J. & Schölkopf, B. (2004). Learning with local and global consistency. *Advances in neural information processing systems (NeurIPS)*.

Dhillon, G. S., Chaudhari, P., Ravichandran, A. & Soatto, S. (2020). A Baseline for Few-Shot Image Classification. *International Conference on Learning Representations*.

Dhillon, I. S., Guan, Y. & Kulis, B. (2004). Kernel k-means: spectral clustering and normalized cuts. *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 551-556.

Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J. & Pontil, M. (2018). Empirical Risk Minimization Under Fairness Constraints. *Neural Information Processing Systems (NeurIPS)*, pp. 2796–2806.

Dua, D. & Graff, C. (2017). UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences.

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. *ICML*, 3, 147–153.

Fei-Fei, L., Fergus, R. & Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4), 594–611.

Finn, C., Abbeel, P. & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *ICML*.

Fowlkes, C., Belongie, S., Chung, F. & Malik, J. (2004). Spectral grouping using the Nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2), 214–225.

Fujiwara, Y. & Irie, G. (2014). Efficient label propagation. *International Conference on Machine Learning*, pp. 784–792.

Ghasedi Dizaji, K., Herandi, A., Deng, C., Cai, W. & Huang, H. (2017). Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. *International Conference on Computer Vision (ICCV)*, pp. 5747-5756.

Gidaris, S. & Komodakis, N. (2018). Dynamic few-shot visual learning without forgetting. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4367–4375.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P. & Cord, M. (2019). Boosting Few-Shot Visual Learning with Self-Supervision. *Proceedings of the IEEE International Conference on Computer Vision*.

Gong, Y., Pawlowski, M., Yang, F., Brandy, L., Bourdev, L. & Fergus, R. (2015). Web scale photo hash clustering on a single machine. *Computer Vision and Pattern Recognition (CVPR)*, pp. 19–27.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. *Neural Information Processing Systems (NIPS)*, pp. 2672–2680.

Grandvalet, Y. & Bengio, Y. (2005). Semi-supervised learning by entropy minimization. *Advances in neural information processing systems (NeurIPS)*.

Hardt, M., Price, E. & Srebro, N. (2016). Equality of Opportunity in Supervised Learning. *Neural Information Processing Systems (NeurIPS)*, pp. 3315–3323.

He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hennig, C., Meila, M., Murtagh, F. & Rocci, R. (2015). *Handbook of cluster analysis*. CRC Press.

Hou, R., Chang, H., Bingpeng, M., Shan, S. & Chen, X. (2019). Cross Attention Network for Few-shot Classification. *Advances in Neural Information Processing Systems (NeurIPS)*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, S. X., Moreno, P. G., Xiao, Y., Shen, X., Obozinski, G., Lawrence, N. D. & Damianou, A. (2020). Empirical Bayes Transductive Meta-Learning with Synthetic Gradients. *International Conference on Learning Representations (ICLR)*.

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708.

Huang, L., Jiang, S. & Vishnoi, N. (2019). Coresets for clustering with fairness constraints. *Advances in Neural Information Processing Systems*, pp. 7587–7598.

Jiang, X., Havaei, M., Varno, F., Chartrand, G., Chapados, N. & Matwin, S. (2019). Learning to Learn with Conditional Class Dependencies. *International Conference on Learning Representations*.

Jiang, Z., Zheng, Y., Tan, H., Tang, B. & Zhou, H. (2017). Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1965–1972. Consulted at http://dl.acm.org/citation.cfm?id=3172077.3172161.

Joachims, T. (1999). Transductive Inference for Text Classification Using Support Vector Machines. *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*.

Julia, A., Larson, J., Mattu, S. & Kirchner, L. (2016). Propublica – machine bias.

Kearns, M., Mansour, Y. & Ng, A. Y. (1998). An information-theoretic analysis of hard and soft assignment methods for clustering. In *Learning in graphical models* (pp. 495–520). Springer.

Khandani, A. E., Kim, A. J. & Lo, A. W. (2010). Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11), 2767–2787.

Kim, J., Kim, T., Kim, S. & Yoo, C. D. (2019). Edge-labeling graph neural network for few-shot learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kleinberg, J., Lakkaraju, H., Leskovec, J., Ludwig, J. & Mullainathan, S. (2017). Human decisions and machine predictions. *The quarterly journal of economics*, 133(1), 237–293.

Kleindessner, M., Samadi, S., Awasthi, P. & Morgenstern, J. (2019). Guarantees for Spectral Clustering with Fairness Constraints. *International Conference of Machine Learning (ICML)*, pp. 3458–3467.

Krähenbühl, P. & Koltun, V. (2011). Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. *Neural Information Processing Systems (NIPS)*, pp. 109–117.

Krähenbühl, P. & Koltun, V. (2013). Parameter learning and convergent inference for dense random fields. *International Conference on Machine Learning (ICML)*, pp. 513–521.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems (NIPS)*, pp. 1097–1105.

Lange, K., Hunter, D. R. & Yang, I. (2000a). Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics*, 9(1), 1–20.

Lange, K., Hunter, D. R. & Yang, I. (2000b). Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics*, 9(1), 1–20.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324. doi: 10.1109/5.726791.

Lee, K., Maji, S., Ravichandran, A. & Soatto, S. (2019). Meta-learning with differentiable convex optimization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665.

Li, J., Ray, S. & Lindsay, B. G. (2007). A nonparametric statistical approach to clustering via mode identification. *Journal of Machine Learning Research*, 8, 1687–1723.

Liu, J., Song, L. & Qin, Y. (2019). Prototype Rectification for Few-Shot Learning.

Meila, M. & Shi, J. (2001). A random walks view of spectral segmentation.

Meng Tang,Ben Ayed, I. & Boykov, Y. (2014). Normalized cut meets MRF. *ECCV submission*.

Miller, E., Matsakis, N. & Viola, P. (2000). Learning from One Example through Shared Densities on Transforms. *Computer Vision and Pattern Recognition (CVPR)*, 1, 464–71.

Mishra, N., Rohaninejad, M., Chen, X. & Abbeel, P. (2018). A Simple Neural Attentive Meta-Learner. *International Conference on Learning Representations*.

Miyato, T., Maeda, S.-i., Koyama, M. & Ishii, S. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*.

Moro, S., Cortez, P. & Rita, P. (2014). A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62, 22–31.

Muja, M. & Lowe, D. G. (2014). Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11), 2227–2240.

Munkhdalai, T., Yuan, X., Mehri, S. & Trischler, A. (2018, 10–15 Jul). Rapid Adaptation with Conditionally Shifted Neurons. *Proceedings of the 35th International Conference on Machine Learning*, 80(Proceedings of Machine Learning Research), 3664–3673.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1), 32–38.

Narasimhan, M. & Bilmes, J. (2005). A Submodular-supermodular Procedure with Applications to Discriminative Structure Learning. *Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 404–412.

Newling, J. & Fleuret, F. (2016). *Nested Mini-Batch K-Means*. Poster presented in Neural Information Processing Systems (NIPS) (pp. 1352-1360).

Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23), 8577–8582.

Ng, A. Y., Jordan, M. I., Weiss, Y. et al. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2, 849–856.

Nichol, A., Achiam, J. & Schulman, J. (2018). On First-Order Meta-Learning Algorithms. *ArXiv*, abs/1803.02999.

Oliva, A. & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3), 145–175.

Oreshkin, B., López, P. R. & Lacoste, A. (2018). Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in Neural Information Processing Systems*, pp. 721–731.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Perlich, C., Dalessandro, B., Raeder, T., Stitelman, O. & Provost, F. (2014). Machine learning for targeted display advertising: Transfer learning in action. *Machine learning*, 95(1), 103–127.

Qiao, L., Shi, Y., Li, J., Wang, Y., Huang, T. & Tian, Y. (2019). Transductive Episodic-Wise Adaptive Metric for Few-Shot Learning. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Qiao, S., Liu, C., Shen, W. & Yuille, A. L. (2018). Few-shot image recognition by predicting parameters from activations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7229–7238.

Rasmussen, C. E. (1999). The infinite Gaussian mixture model. *NIPS*, 12, 554–560.

Ravi, S. & Larochelle, H. (2017). Optimization as a Model for Few-Shot Learning. *ICLR*.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H. & Zemel, R. S. (2018). Meta-Learning for Semi-Supervised Few-Shot Classification. *Proceedings of 6th International Conference on Learning Representations ICLR*.

Rösner, C. & Schmidt, M. (2018). Privacy preserving clustering with constraints. *ICALP*.

Rother, C., Kolmogorov, V. & Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3), 309–314.

Rousson, M. & Deriche, R. (2002). A variational framework for active and adaptative segmentation of vector valued images. *Motion and Video Computing, 2002. Proceedings. Workshop on*, pp. 56–61.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 211-252. doi: 10.1007/s11263-015-0816-y.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S. & Hadsell, R. (2019). Meta-Learning with Latent Embedding Optimization. *International Conference on Learning Representations*.

Salah, M. B., Ayed, I. B., Yuan, J. & Zhang, H. (2014). Convex-relaxed kernel mapping for image segmentation. *IEEE Transactions on Image Processing*, 23(3), 1143–1153.

Samadi, S., Tantipongpipat, U. T., Morgenstern, J. H., Singh, M. & Vempala, S. (2018). The Price of Fair PCA: One Extra dimension. *Neural Information Processing Systems (NeurIPS)*, pp. 10999–11010.

Schmidt, M., Schwiegelshohn, C. & Sohler, C. (2018). Fair Coresets and Streaming Algorithms for Fair k-Means Clustering. *arXiv 1304.6478*, abs/1812.10854.

Shaham, U., Stanton, K., Li, H., Basri, R., Nadler, B. & Kluger, Y. (2018). SpectralNet: Spectral Clustering using Deep Neural Networks. *International Conference on Learning Representations (ICLR)*. Consulted at https://openreview.net/forum?id=HJ_aoCyRZ.

Shi, J. & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 888–905.

Snell, J., Swersky, K. & Zemel, R. (2017). Prototypical Networks for Few-shot Learning. *Advances in Neural Information Processing Systems*.

Strehl, A. & Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3(12), 583–617.

Sun, Q., Liu, Y., Chua, T. & Schiele, B. (2019, June). Meta-Transfer Learning for Few-Shot Learning. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

Tang, M., Marin, D., Ayed, I. B. & Boykov, Y. (2019a). Kernel Cuts: Kernel and Spectral Clustering Meet Regularization. *International Journal of Computer Vision*, 127, 477-511.

Tang, M., Marin, D., Ayed, I. B. & Boykov, Y. (2019b). Kernel Cuts: Kernel and Spectral Clustering Meet Regularization. *International Journal of Computer Vision*, 127, 477-511.

Tian, F., Gao, B., Cui, Q., Chen, E. & Liu, T.-Y. (2014). Learning deep representations for graph clustering. *AAAI Conference on Artificial Intelligence*, pp. 1293–1299.

Vaida, F. (2005). Parameter convergence for EM and MM algorithms. *Statistica Sinica*, 15, 831–840.

Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u. & Polosukhin, I. (2017). Attention is All You Need. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, (NIPS'17), 6000–6010.

Vinyals, O., Blundell, C., Lillicrap, T. P., Kavukcuoglu, K. & Wierstra, D. (2016). Matching Networks for One Shot Learning. *NIPS*.

Vladymyrov, M. & Carreira-Perpiñán, M. (2016). The Variational Nystrom method for large-scale spectral problems. *International Conference on Machine Learning (ICML)*, pp. 211–220.

Von Luxburg, U. (2007a). A tutorial on spectral clustering. *Statistics and computing*, 17(4), 395–416.

Von Luxburg, U. (2007b). A tutorial on spectral clustering. *Statistics and computing*, 17(4), 395–416.

Wah, C., Branson, S., Welinder, P., Perona, P. & Belongie, S. (2011). The caltech-ucsd birds-200-2011 dataset.

Wang, W. & Carreira-Perpinán, M. A. (2014). The Laplacian K-modes algorithm for clustering. *arXiv preprint arXiv:1406.3895*.

Wang, Y., Chao, W.-L., Weinberger, K. Q. & van der Maaten, L. (2019). SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning. *arXiv preprint arXiv:1911.04623*.

Wertheimer, D. & Hariharan, B. (2019). Few-shot learning with localization in realistic settings. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6558–6567.

Weston, J., Ratle, F., Mobahi, H. & Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade* (pp. 639–655). Springer.

Wolf, L., Hassner, T. & Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. *Computer Vision and Pattern Recognition (CVPR)*, pp. 529–534.

Wu, Z. & Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11), 1101–1113.

Xu, D. & Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2), 165–193.

Yanbin, L., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S. & Yang, Y. (2019). Learning to Propagate Labels: Transductive Propagation Network for Few-shot Learning. *International Conference on Learning Representations*.

Ye, H.-J., Hu, H., Zhan, D.-C. & Sha, F. (2020). Few-Shot Learning via Embedding Adaptation with Set-to-Set Functions. *Computer Vision and Pattern Recognition (CVPR)*.

Yuan, J., Yin, K., Bai, Y., Feng, X. & Tai, X. (2017). Bregman-Proximal Augmented Lagrangian Approach to Multiphase Image Segmentation. *Scale Space and Variational Methods in Computer Vision (SSVM)*, pp. 524–534.

Yuille, A. L. & Rangarajan, A. (2001). The Concave-Convex Procedure (CCCP). *Neural Information Processing Systems (NIPS)*, pp. 1033–1040.

Zafar, M. B., Valera, I., Gomez-Rodriguez, M. & Gummadi, K. P. (2017). Fairness Constraints: Mechanisms for Fair Classification. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 962–970.

Zagoruyko, S. & Komodakis, N. (2016, September). Wide Residual Networks. *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 87.1-87.12. doi: 10.5244/C.30.87.

Zhang, J., Zhao, C., Ni, B., Xu, M. & Yang, X. (2019). Variational Few-Shot Learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1685-1694.

Zhang, Z., Kwok, J. T. & Yeung, D.-Y. (2007). Surrogate maximization/minimization algorithms and extensions. *Machine Learning*, 69, 1-33.

Zhu, X. & Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation.

Ziko, I., Granger, E. & Ayed, I. B. (2018). Scalable Laplacian K-modes. *Advances in Neural Information Processing Systems*, pp. 10041–10051.