Joint Constrained Clustering and Feature Learning based on Deep Neural Networks

by

Xiaoyu Liu

B.Sc., University of Science and Technology of China, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2017

Approval

Name:	**Xiaoyu Liu**
Degree:	**Master of Science**
Title:	***Joint Constrained Clustering and Feature Learning based on Deep Neural Networks***
Examining Committee:	**Chair:** Richard T. Vaughan Associate Professor School of Computing Science

Greg Mori
Senior Supervisor
Professor
School of Computing Science _____

Jiannan Wang
Supervisor
Assistant Professor
School of Computing Science _____

Parmit Chilana
Examiner
Assistant Professor
School of Computing Science _____

Date Defended: August 23rd, 2017

Abstract

We propose a novel method to iteratively improve the performance of constrained clustering and feature learning based on Convolutional Neural Networks (CNNs). There is no effective strategy for neither the constraint selection nor the distance metric learning in traditional constrained clustering methods. In our work, we design an effective constraint selection strategy and combine a CNN-based feature learning approach with the constrained clustering algorithm. The proposed model consists of two iterative steps: First, we replace the random constraint selection strategy with a carefully designed one; based on the clustering result and constraints obtained, we fine tune the CNN and extract new features for distance re-calculation. Our model is evaluated on a realistic video dataset, and the experimental results demonstrate that our method can improve the constrained clustering performance and feature divisibility simultaneously even with fewer constraints.

Keywords: Constrained Clustering; Feature learning with Convolutional Neural Networks; User Feedback; Active Learning

Acknowledgements

First and foremost, I would like to show my deepest thank and respect to my supervisor Dr. Greg Mori because he is a responsible and knowledgeable professor, who provided my valuable and detailed guidance during all my master life. Without his instruction, I could not complete my thesis or enjoy the research experience. I believe his positive attitude and passion to his career will enlighten mine in the future as well.

Then I want to extend my thanks to Dr. Jiannan Wang, Dr. Parmit Chilana and Dr. Richard T. Vaughan, the members of my supervisory committee, for their helpful suggestions and insightful comments.

Last but not least, I appreciate all my friends for their love, encouragement, and support.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

We are overwhelmed by various digital data (e.g., text, image and video etc.) because of the development of high-capacity storage devices and data compression techniques and other technological advances. There are over 300 hours of videos uploaded to YouTube every minute, and it is inefficient to manipulate these videos manually like doing the data search, data retrieval or other operations. For example, when there is a stream of videos in front of you and you are asked to group them into several clusters according to some relationship that is not provided. There are two popular data partition methods, clustering [17] and classification [35], and they are both capable of partitioning data into meaningful sub-groups. But to make use of classification, you need to specify the class labels before seeing the unknown videos. Therefore, in the case above, we can only apply clustering methods, which require no training data.

Because clustering methods work by exploring the potential patterns and relationships among data instances without the supervision from labels, it is easy to return an unexpected clustering. For instance, when the algorithm returns a clustering according to different colors, some people might think the clustering is not perfect enough because they would like to get a clustering based on animal species. To consider the domain knowledge during the clustering procedure, researchers import user feedback in the format of instance-level constraints [42]. The instance-level constraints reflect the relationships among data instances. By forcing the clustering to satisfy all the constraints, we can get a more expected result.

This is exactly the task of constrained clustering [3], which can incorporate limited domain knowledge to fasten and ease the clustering process. However, operating on raw pixels directly can still be time-consuming and misleading, driving the evolution of feature learning techniques.

1.2 Motivation

Among different data formats, videos have much more information than an individual image or a single sentence, and the large amount of information demands compact and meaningful representations. Meanwhile, it is quite appropriate to build clustering methods on videos because of the spatial and temporal regularity contained in videos.

Video clustering can be a technique of great use for easing the subsequent data manipulations, such as video retrieval and video summary. [40] addressed video clustering task using key frames by assuming video clips have been segmented into shots that are represented by a sequence of key frames. This method is considered as a relax solution when the information of the whole video is hard to represent.

Recent achievements in deep representations make it easy to extract high-level and abstract video features, and the temporal relation between video frames can be captured by applying recurrent layers or a temporal pooling.

Given the ability of constrained clustering to incorporate limited domain knowledge, we would like to combine the constrained clustering and deep representations to improve the video clustering performance as much as it can with as few constraints as possible.

We consider the method as a distance-flexible constrained clustering [5], where the distance learning is achieved by fine tuning a CNN to increase the feature divisibility. With the strong ability of deep neural networks to fit high-dimension feature space, we expect that deep neural networks can be better distance functions than previous methods. With a higher feature divisibility, the clustering process can be simplified and accelerated. Conversely, we hope the clustering and constraints can provide helpful guidance for feature learning.

1.3 Outline

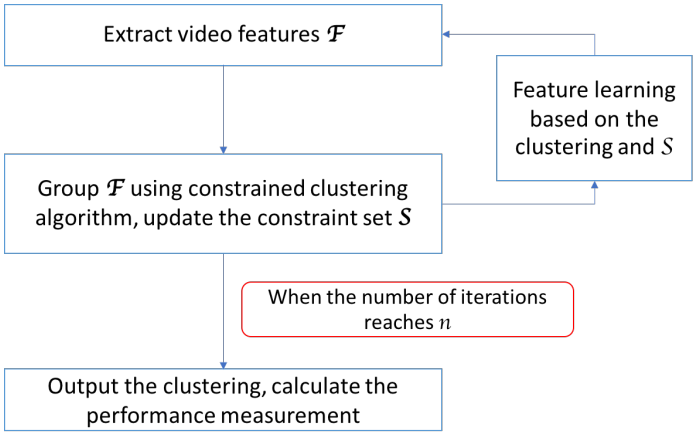


Figure 1.1: Overview of the proposed method

Our method iterates between clustering video features based on the constraint set and refining the feature space based on the clustering result. This is called one iteration, and when the number of iterations reaches a pre-defined value, the method stops and outputs the final clustering result with the clustering measurement. During the constrained clustering procedure, new constraints are obtained by asking whether two data instances belong to the same category. The overview of our method is as Figure 1.1.

The structure of this thesis is organized as follows:

- Chapter 2 introduces several traditional clustering algorithms and corresponding distance-fixed constrained clustering algorithms.
- Chapter 3 explains our method towards designing the constraint selection strategy as well as jointly improving constrained clustering and feature learning performance based on deep neural networks.
- Chapter 4 presents the experimental results under various conditions including different constraint selection strategies, diverse values of hyperparameters and whether feature learning is performed.
- Chapter 5 summarizes our method and proposes future directions for further improvement.

Chapter 2

Previous Work

Clustering is a quite common data partition technique, which explores a partition where the distance between data points inside the same cluster is small while the distance between data points from different clusters is large given a feature space. There are many clustering algorithms that can handle different clustering problems, and almost each of them has the corresponding constrained format. In this chapter, we are going to introduce several popular traditional clustering algorithms including k-means [28], spectral clustering [41], density-based clustering [10], hierarchical clustering [19] and their constrained formats.

During the constrained clustering procedure, new constraints are obtained by asking the source of expertise the relationship between two data instances, which is an active learning [48] way and we are going to take a compact overview about the typical heuristic of active learning.

Instead of operating directly on the pixels, we would like to do some feature extraction at first. In traditional video representations, trajectory-based approaches [45, 46], especially the Dense Trajectory (DT), are the basis of current state-of-the-art hand-crafted algorithms. There are many extended methods [25] about obtaining better DT features.

However, with the arising of deep neural network methods, deep representations extracted by CNNs outperform the traditional ones by a large margin on various datasets. One advantage of CNNs is their strong ability to fit the high-dimension feature space, so it can act as a complicated but powerful distance function.

We expect that the combination of constrained clustering and the deep feature learning method can benefit each other, which is achieved by an iterative process like the Expectation – maximization (EM) algorithm [51]. In each iteration, the algorithm either obtains the constrained clustering result with the feature space fixed or refines the feature space based on the clustering result. When the number of iterations reaches a pre-defined value, the loop will be terminated and the algorithm will output the final clustering as well as the clustering measurement.

2.1 Traditional Clustering Algorithm

2.1.1 K-means

K-means [28] is a popular clustering algorithm that is easy to implement and straightforward to understand. K-means discovers the final partition in an iterative way, which either assigns data points to the closest cluster center or calculates new centers according to the current assignments. The distortion function is defined as follow:

$$J(c,\mu) = \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2 \tag{2.1}$$

where m is the number of data points; c is the cluster index vector; μ is the center vector. $\left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$ is the distance between the ith data point and its corresponding cluster center. Algorithm 1 shows the details of traditional k-means.

Algorithm 1: K-means

Input : Data set X, the number of clusters K
Output: Cluster assignments L

1 Randomly choose a set of K initial centers from all data points
2 For each $x^{(i)} \in X$, $L_i = \arg\min_{c_k} \left\| x^{(i)} - \mu_{c_k} \right\|^2$
3 Compare L with assignments from previous iteration, if same, terminate, else go to step 4
4 For each cluster, re-compute the center $\mu_{c_k} = \frac{1}{|c_k|} \sum_{x^{(i)} \in C_k} x^{(i)}$ and turn to step 2

Step 2 and step 4 in Algorithm 1 are iterated until getting two identical assignment vectors in succession. If too many iterations have been performed and the algorithm still fails to converge, the process will be terminated in case of an endless loop.

The main limitation of k-means is its high sensitivity to the initial centers, because different initialization sets can cause a large performance shift. Usually, researchers will run k-means multiple times with different initialization sets and choose the best one, namely the one with the smallest distortion function value. There are some improved k-means algorithms that can avoid the bad initialization like k-means++ [1] introduced by Arthur et al. in 2007. The intuition behind k-means++ is that spreading initial centers is good for approaching the optimal assignments.

2.1.2 Spectral Clustering

Spectral clustering [41] is an extension of k-means, and the main idea of spectral clustering is to perform dimension reduction using the spectrum of the affinity matrix before standard k-means. According to different similarity graphs and graph Laplacians, spectral clustering is capable of handling kinds of problems efficiently by standard linear algebra methods.

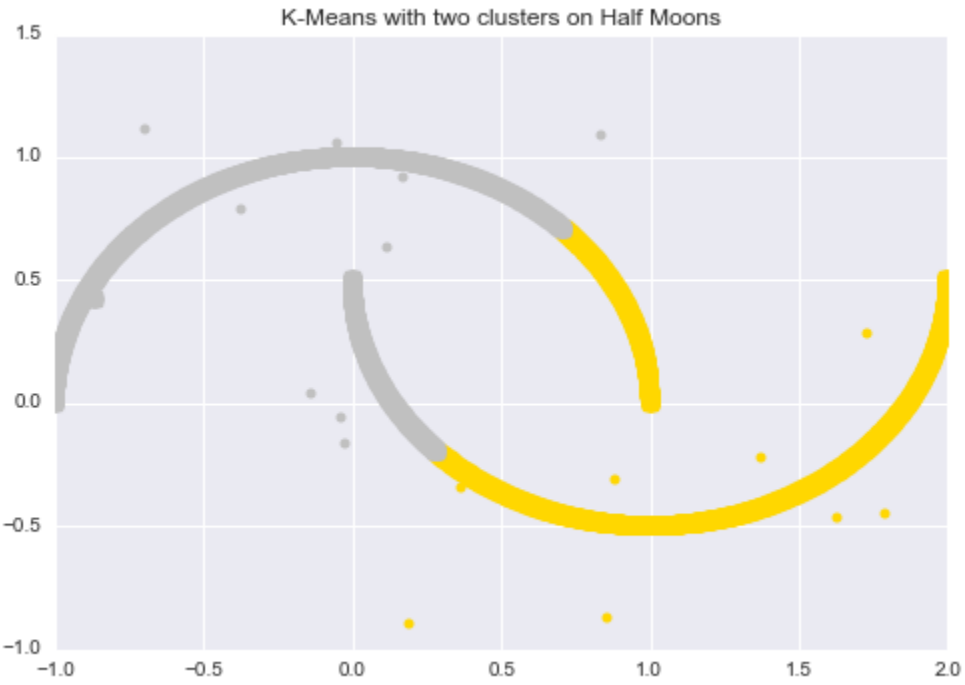


Figure 2.1: Clustering result of k-means on a non-linearly separable dataset

Both unnormalized and normalized spectral clustering need to first construct a similarity matrix based on one of three similarity graphs:

- *ε-neighborhood graph*: Connect all points whose pairwise distances are smaller than ε. Because all distances between connected points are of the roughly same scale, this graph can be nearly treated as an unweighted graph.
- *k-nearest neighborhood graph*:
  - *k-nearest neighborhood graph*: Connect all points using undirected edges as soon as one vertex is among the other vertex's k nearest neighbors.
  - *mutual k-nearest neighborhood graph*: Connect two points only when they are among each other's k nearest neighbors at the same time.
  - In both ways, the distance between connected points can be used to calculate a weight for the edge.
- *fully connected graph*: Simply connect all data points. Only being considered when the similarity function can encode information of local neighbors.

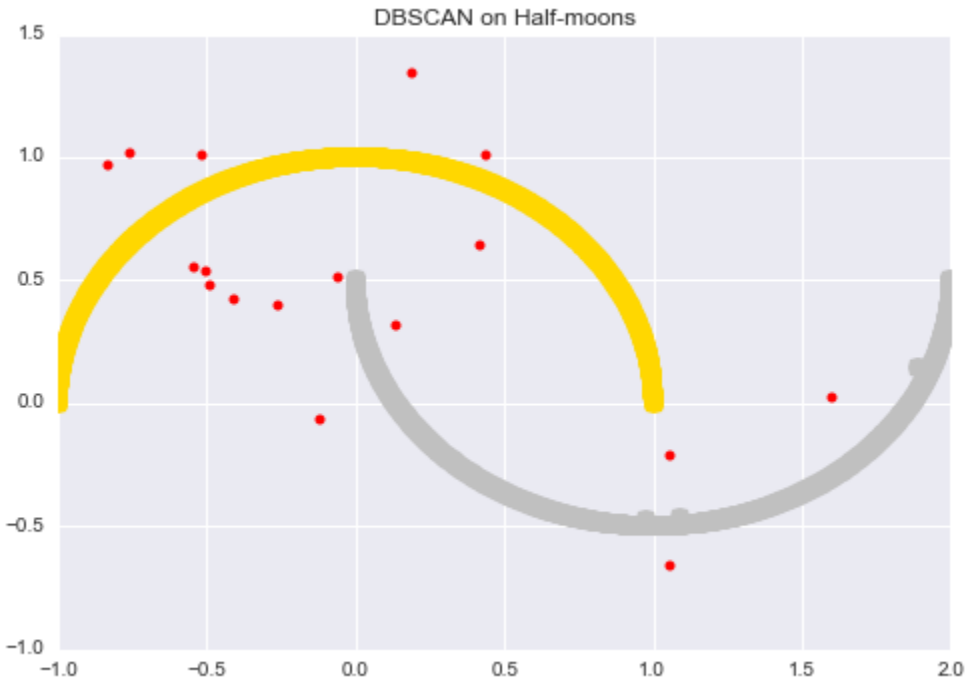


Figure 2.2: Clustering result of DBSCAN on a non-linearly separable dataset

2.1.3 DBSCAN

Previous two clustering algorithms are based on distance, which results in some disadvantages like the disability to handle non-convex data. Figure 2.1 shows the poor performance when applying standard 2-way k-means on non-convex data. Besides, it is required to have the number of clusters as a priori when using k-means, limiting the application of k-means as well.

On account of limitations of distance-based clustering algorithms, density-based clustering methods are proposed to handle arbitrary shape clustering problems and clustering problems where the number of clusters is not known. DBSCAN is a popular density-based clustering algorithm, it is able to find non-linearly separable clusters and is more robust to outliers than distance-based clustering algorithms.

There are two parameters in DBSCAN:

- ε : the radius of the neighborhood searched for directly reached points
- min_{pts} : the minimum number of points required to form a dense region

Algorithm 2 explains how DBSCAN works, where *regionQuery()* returns all points within the n-dimensional sphere and *expandCluster()* expands the cluster from each point in the sphere. Figure 2.2 shows the result of applying DBSCAN on non-convex data.

Algorithm 2: DBSCAN

Input : Data set X, ε, min_{pts}
Output: Cluster assignments L

- DBSCAN(X, ε, min_{pts}):
  C = 0
  for *each unvisited point* $p \in X$ **do**
    mark p as *visited*
    sphere-points = regionQuery(p, ε)
    if *sizeof(sphere-points)* $< min_{pts}$ **then**
      ignore P
    else
      C = next cluster
      expandCluster(p, sphere-points, C, ε, min_{pts})
- expandCluster(p, sphere-points, C, ε, min_{pts}):
  add p to cluster C
  for *each point* $p' \in sphere - points$ **do**
    if p' *is not visited* **then**
      mark p' as visited
      sphere-points' = regionQuery(p', ε)
      if *sizeof(sphere-points')* $\geq min_{pts}$ **then**
        sphere-points = sphere-points $\cup$ sphere-points'
      if $p' \notin$ *any cluster* **then**
        add p' to cluster C
- regionQuery(p, ε):
  return all points within p's ε-neighborhood (including p)

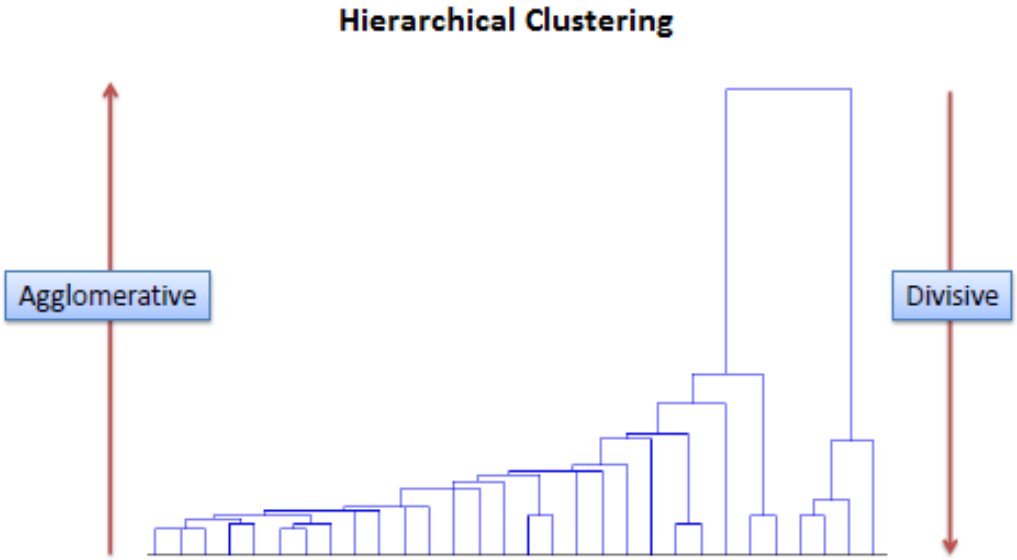


Figure 2.3: Hierarchical clustering

2.1.4 Hierarchical Clustering

Thinking about the way of humans learn, we usually learn things from a coarse level to a fine level or conversely. For example, when learning the definition of "bicycle", it is easier if we have already known the meaning of "transportation" because "bicycle" is a special "transportation", and it must share common properties with other transportation. On the contrary, if we have known the definition of "bicycle", "bus", "skytrain", we can understand "transportation" faster by summarizing their common properties. This is basically how and why hierarchical clustering algorithms work. According to different strategies, hierarchical clustering falls into two types and Figure 2.3 explains how they work separately:

- **Agglomerative clustering**: This is a "bottom up" approach, like "bicycle" → "transportation". In this method, we first treat each instance as a separate cluster, then merge two clusters with the highest similarity. Repeat merging until the number of clusters is equal to the required value.
- **Divisive clustering**: This is a "top down" approach, like "transportation" → "bicycle". In this method, we first group all data instances into one cluster then partition the cluster to several sub-clusters to maximize the sum of distances between clusters.

Either *agglomerative* or *divisive* approach depends on how we define the distance between two clusters. There are three ways to define the distance:

1. Single Link: The distance between two clusters is that between the closest points belonging to two clusters respectively.

2. Average Link: The distance between two clusters is that between two cluster centers.

3. Complete Link: The distance between two clusters is that between the furthest points belonging to two clusters respectively.

2.2 Constrained Clustering

Traditional clustering algorithms obey a pure unsupervised manner to group objects. Though it is efficient to automatically partition data according to some similarity metric, the accuracy can be pretty low. And the clustering result may be meaningful but unexpected if there is no supervision provided. For example, given a dataset of animals, the desired clusters are species of animals but we may get clustering according to different colors. In many real-life problems, limited domain knowledge or expertise is available and should be considered for better clustering result. This is the main idea of constrained clustering [3]. By incorporating incomplete domain knowledge into clustering procedure, constrained clustering algorithms are able to obtain better clustering results.

There are lots of constraint types, such as a *minimum* or *maximum* radius [8] to keep a small distance within clusters and a large distance between clusters, or existential constraints [39] limiting the minimum size of each cluster. And in some cases, labels of a sub-dataset are provided. However, the instanced-based constraint is the most straightforward and simplest constraint type, eventually, other constraints can be expressed using a set of instance-based constraints as well. In this thesis, the domain knowledge is represented in the form of instance-based constraints.

2.2.1 Constraint Definition

There are two kinds of constraints introduced by Wagstaff and Cardie [42]:

- Must-link, *ML(x,y)*, which indicates that x, y must be clustered into the same group.
- Cannot-link, *CL(x,y)*, which indicates that x, y must not be clustered into different groups.

There are several important properties for must-links and cannot-links:

- Both must-links and cannot-links are symmetric.
- Only must-links are transitive, which means ***if*** $(x, y) \in ML$ and $(y, z) \in ML$, ***then*** $(x, z) \in ML$; and ***if*** $(x, y) \in CL$ and $(y, z) \in CL$, we ***cannot*** refer the relationship between x and z.
- Only must-links are reflexive, which means a data instance must always be placed into the same cluster with itself.

We obtain constraints by simulating the process that we ask an expert or someone who knows the correct partition some questions, which might be whether two data instances belong to the same category or which cluster the data point should be assigned to or any other questions that can reflect the true partition.

In the thesis, the constraints are whether two action videos belong to the same class based on the dataset labels. Constraints are highly dependent on the domain knowledge of the source of expertise because different views might give different constraints.

2.2.2 Advantage and Application

It is easy to adapt different domain knowledge using constrained clustering algorithm. For example, when we need to incorporate background information like time in a traditional clustering algorithm, it is hard to design a loss function reflecting the reasonable difference caused by different time. But in constrained clustering algorithms, we can simply encode time information with *must-links* and *cannot-links*. As long as constraints can provide extra information besides the distance metric, they can be useful.

Since constrained clustering was proposed, it has been applied to lots of applications including GPS Lane Finding [3], segmentation of images [47], human tracking based on videos [57], flower measurement [5] and many other tasks. In the following part, we are going to introduce details of several constrained clustering algorithms.

2.3 Constrained Clustering Algorithm

By whether the distance metric is updated, constrained clustering algorithms can be divided into two categories: distance-fixed algorithms and distance-flexible algorithms:

- Distance-fixed constrained clustering algorithm: constraints are considered only as the relations between data points to guide the clustering process or to initialize cluster centers reasonably. The clustering algorithm will not change the distance metric.

- Distance-flexible constrained clustering algorithm: constraints can be viewed as distance indicators that data points belong to a *must-link* are supposed to have a small distance, and data points belong to a *cannot-link set* are supposed to have a large distance. The clustering algorithm will learn an adaptive distance metric based on current constraints.

Because our method is based on a distance-fixed constrained clustering algorithm, here we introduce some well-defined ones.

2.3.1 Constrained K-means

As stated before, traditional k-means algorithm suffers from sensitivity to initial centers, because it can be dramatically slow and return a bad result if starting from a low-quality initialization. Domain knowledge can help get a more meaningful clustering result and fast convergence. Constrained k-means is introduced by Wagstaff et al. [42] to incorporate domain knowledge with k-means. Algorithm 3 explains the detail of constrained k-means.

Algorithm 3: Constrained k-means

Input : Data set X, must-link set ML, cannot-link set CL

Output : Cluster assignments L satisfying ML and CL

1 Let $C_1, ..., C_k$ be the initial cluster centers.

2 For each point x_i in D, assign it to the closest cluster C_j **such that** VIOLATE-CONSTRAINTS(x_i, C_j, ML, CL) returns false. **If no such cluster exists, return null**.

3 For each cluster C_i, update its center by averaging all of points x_i that have been assigned to it.

4 Iterate step 2 and step 3 until convergence.

5 Return $C_1, ..., C_k$.

Function: VIOLATION-CONSTRAINTS(data point x, cluster C, must-link set ML, cannot-link set CL)

6 For each $(x, x_= \in ML)$: If $x_= \notin C$, return true.

7 For each $(x, x_\neq \in CL)$: If $x_\neq \in C$, return true.

8 Otherwise, return false.

From step 2 in Algorithm 3 we can see, instead of assigning each data point to the closest cluster, constrained k-means assigns it to the closest *feasible* cluster. For a data point x, starting from the nearest cluster, if the cluster satisfies one of three following requirements, then the cluster is a feasible one:

1. At least one data instance must-linked with x has been assigned to the cluster;
2. All data instances cannot-linked with x are not assigned to the cluster;
3. None of the data instances in the cluster has any constraint with x.

This algorithm returns a partition of instances in X, satisfying all constraints strictly. It takes advantage of properties of must-links and cannot-links we stated before. For example, step 6 can be explained as: ***if*** *any data point sharing a must-link with x_i has been assigned to another cluster C_j,* ***then*** *x_i must be assigned to cluster C_j,* ***and*** *x_i cannot be assigned to cluster C_i ($i \neq j$).*

Though constrained k-means is able to utilize instance-level constraints, it can be affected by the input order, causing the *constraint violation* problem.

Constraint violation means an instance has got at least one single *cannot-link* in each cluster, so it cannot be assigned to any cluster. As the situation in Figure 2.4, $(x, y) \in CL$,

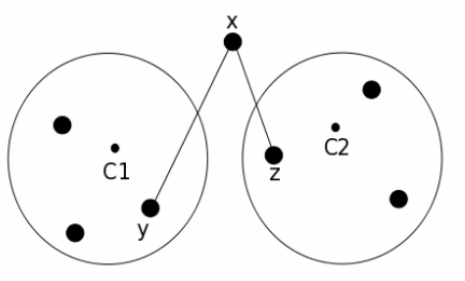


Figure 2.4: Constraint violation for data point x

$(x, z) \in CL$, and the input order is y, z, x. When it comes to x, constraint violation happens because x cannot be grouped into either cluster.

Only *cannot-link* constraints may induce constraint violation because *must-link* constraints are transitive and are not influenced by the input order. CLK-Kmeans [59] is an effective method to solve the problem, it makes use of the conclusion that if the maximum number of *cannot-links* involving one instance is smaller than K (the number of clusters required), there will always be a feasible solution regardless of the input order conditioned on zero noise in constraints. We utilize the principle of CLK-Kmeans in our algorithm to decrease the constraint violation probability.

2.3.2 Constrained Spectral Clustering

Algorithm 4: Constrained Spectral Clustering for K-way Partition

Input : Affinity matrix A, constraint matrix Q, hyperparameter β, the number of clusters K

Output: Cluster assignments $u*$

$vol \leftarrow \sum_{i=1}^{N} \sum_{j=1}^{N} A_{ij}$, $D \leftarrow diag(\sum_{j=1}^{N} A_{ij})$

$\overline{L} \leftarrow I - D^{-1/2} A D^{-1/2}$, $\overline{Q} \leftarrow D^{-1/2} Q D^{-1/2}$

$\lambda_{max} \leftarrow$ the largest eigenvalue of $\overline{Q}$

if $\beta \geq \lambda_{K-1} vol$ **then**

  return $u* = \emptyset$

else

  solve he generalized eigenvalue system in $\overline{L}v = \lambda(\overline{Q} - \frac{\beta}{vol} I)v$

  remove eigenvectors associated with non-positive eigenvalues and normalize the rest by $v \leftarrow \frac{v}{||v||}\sqrt{vol}$

  $V* \leftarrow argmin_{V \in R^{(N \times (K-1))}} trace(V^T \overline{L} V)$, where the columns of V are a subset of feasible eigenvectors generated in the previous step

  return $u* \leftarrow$kmeans$(D^{-1/2} V*, K)$

Constrained spectral clustering can be divided into two categories according to how they make use of constraints. The former [44] manipulates the graph Laplacian matrix or the affinity matrix directly according to the given constraints. The latter [27] restricts the feasible solution space based on constraints. For constrained spectral clustering, Wang et al. [47] came up with an effective and principled framework described in Algorithm 4.

Here A is the similarity graph, and β is related to eigenvalues of $\overline{Q}$. For each element $q_{ij} \in Q$, $q_{ij} = 1$ means the ith and the jth points are must-linked, $q_{ij} = -1$ means the ith and the jth points are cannot-linked, $q_{ij} = 0$ means there is no constraint between the ith and the jth points.

2.3.3 Constrained Density-based Clustering

Ruiz et al. [32] introduced Density-based Clustering with Constraints (C-DBSCAN), which incorporates instance-level constraints after partitioning the data space into sub-spaces. Algorithm 5 explains how C-DBSCAN works.

Algorithm 5: C-DBSCAN

Input : Data set D, must-kink set ML, cannot-link set CL

Output: Cluster assignments L satisfying ML and CL

```
1  kdtree := BuildKDTree(D)
2  repeat
3      for all unlabeled points in a leaf X do
4          select an arbitrary point p_i from X
5          X_{p_i} ← all points in X that are within Eps radius of p_i
6          if X_{p_i} contains less than MinPts points then
7              Label p_i as NOISE. Break.
8          if exists a Cannot-Link constraint in CL among points in X_{p_i} then
9              create a local cluster for each point in X. Break.
10         else
11             Label p_i as CORE. Label X_{p_i} as LOCAL CLUSTER.
12 until all leaves of the kdtree have been processded;
13 for each constraint m ∈ ML do
14     Add clusters involved in constraint m into cluster Y
15     Label Y as CORE LOCAL CLUSTER
16 for each core(local) cluster Y do
17     while number of local clusters NLC decreases do
18         closestCluster ← closest local cluster to Y
19         if ∃ Cannot-Link constraint in CL between points of Y and closestCluster
            then
20             Y ← Y∪ closestCluster. Label Y as CORE CLUSTER.
21             NLC -= 1
```

2.4 Active Learning

In practical applications, it is usual that constraints are provided by referring a source of expertise. Active Learning [48] is a form of learning that queries an expert for domain knowledge to improve the learning performance. It is different from semi-supervised learning because active learning needs an exterior source of expertise while semi-supervised learning does not need artificial intervention.

Result from different expected values of sample information contained in each data point, active learning can reduce annotation cost and time expense by selecting the valuable part.

Pseudo codes in Algorithm 6 explain the typical heuristic of active learning.

Algorithm 6: Active Learning

```
Given a pool of unlabeled data and a source of expertise
Choose some points randomly and query their labels or relations
repeat
    Learn a classifier based on the labels known so far
    Choose other points according to some rule (like data points that are most likely
      to increase the overall performance) and query their labels
until some end condition is satisfied;
```

2.5 Deep Representation Learning

Deep features extracted by CNNs attract attention from various traditional areas like computer vision and natural language processing. They achieve significant improvement on multiple tasks including speech recognition [15, 11, 7, 2], image classification [23, 37, 14], and object recognition [38, 31, 30] compared to hand-crafted features.

Researchers propose that the larger capacity the network has, the higher illustrative ability the learned representation owns. Inspired by the strong ability of AlexNet [23] to capture potential input distribution, [37] proposed a convolutional neural network with an incredible depth and achieved the state-of-the-art performance on multiple computer vision tasks at that time. Furthermore, recurrent connections [33] and residual building blocks [14] are proposed to improve the network's capacity and decrease the model complexity simultaneously. Based on [14], [56] improves classification accuracy by increasing the width of the residual building block instead, but it still follows the principle of enlarging the model's capacity.

When neural networks grow larger, the need towards more complicated annotated datasets increases sharply. At least one large dataset is collected for each task, e.g. ImageNet [23] for image classification, UCF-101 [36] for action-related tasks, Caltech-256 [12] for object detection. However annotating such a large dataset is too expensive and there is more unlabeled data that supervised learning cannot make use of. Therefore, researchers try to find unsupervised ways to obtain representations using deep neural networks.

There are some works [4, 22] focusing on learning features in an unsupervised way from unlabeled images by minimizing the loss function of reconstruction tasks. [55] conducted image clustering and code-book learning iteratively on SIFT features. [16] made use of the Deep Belief Network (DBN) to learn features and applied a maximum margin clustering on the outputs of the DBN, then the author fine-tuned the DBN's last layer according to clustering results. Due to the excellent performance of deep representations learned by neural

networks on supervised tasks, more recent works turn to jointly learn deep representations and run clustering, which is similar to the target of traditional distance-flexible constrained clustering algorithms. [58] combines feature learning and agglomerative clustering into a recurrent framework. By using agglomerative clustering, they can make sure in the early stage clusters are highly pure and can provide correct supervision for fine-tuning networks. While this method is proved to be effective when applying to small datasets like MNIST [26], it is hard to scale to real-life scenarios especially videos because the agglomerative clustering needs to form initial clusters based on raw pixels.

Chapter 3

Joint Feature Learning and Constrained Clustering based on Deep Neural Network

With a target to combine the advantages of constrained clustering and deep representations, we iteratively optimize feature learning and constrained clustering. A specially designed constraint selection strategy is applied in an active learning way, and the model is evaluated to be effective on a real-life unconstrained video dataset, UCF-101 [36]. The key innovation of the thesis is the combination of constrained clustering with deep feature learning. Also while the task of the thesis is video clustering, our method is applicable broadly for many other tasks as long as a correct loss function for feature learning is defined.

3.1 Outline of Model

Algorithm 7: Outline of Model

Input : Video set X, pre-trained network N, clustering algorithm A

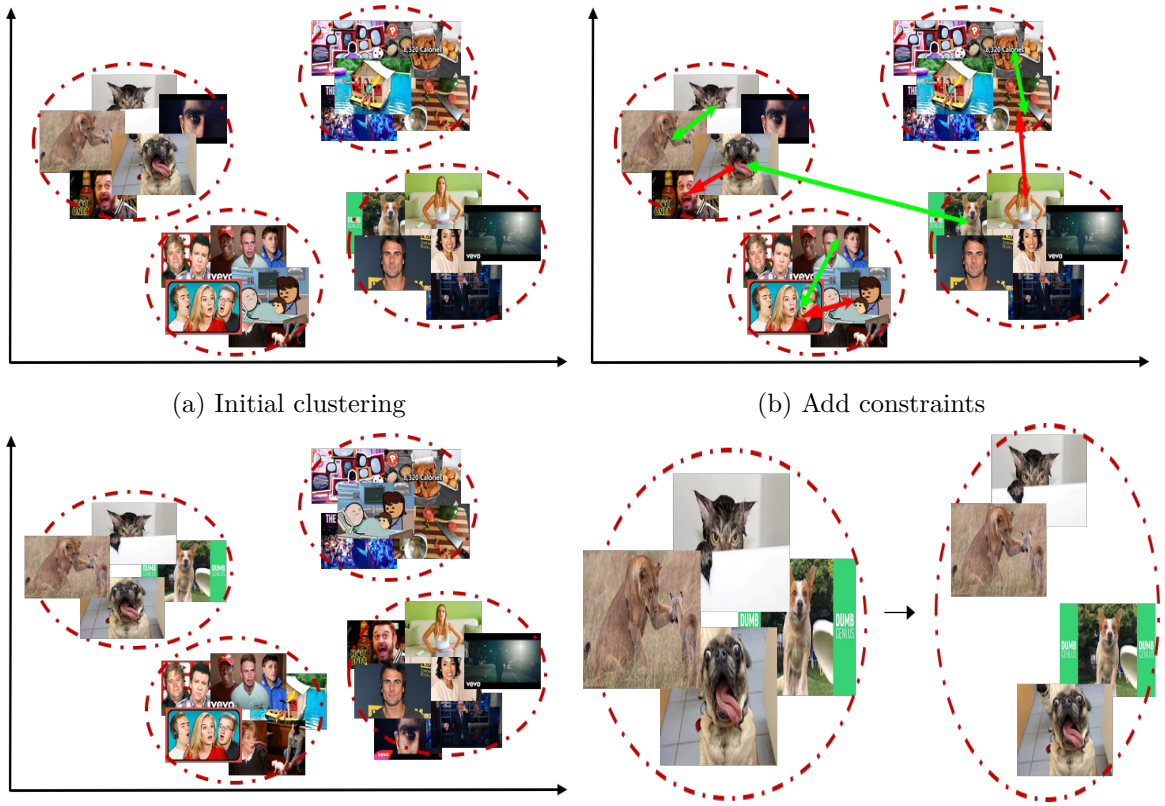
Output: Cluster assignments L

repeat

- Extract video feature x_i using N
- Cluster x_i using A to get the clustering c based on the current constraint set S
- Ask the source of expertise to correct some cluster labels in c and store the new clustering as c^*
- Update the constraint set S according to domain knowledge from the inquiry
- Replace N with N^*, which is fine-tuned with c^* and S

until *the performance of c exceeds a threshold OR the number of iterations reaches a required value*;

We assume a network pre-trained on the ImageNet dataset [23] is available. The network is considered as a feature extractor as well as a distance function, because it is going to not only extract features from the input, but also be fine-tuned with the current constraint set. Algorithm 7 gives the outline of our method.



(a) Initial clustering

(b) Add constraints

(c) Updated clustering

(d) Feature learning

Figure 3.1: Illustrations for step 3 – 5 in Algorithm 7.

To explain the model clearly, we provide an example in Figure 3.1:

- Figure 3.1a is the result of step 3 in Algorithm 7. It is the initial clustering of given videos based on the feature extracted by a CNN and the current constraint set.
- Figure 3.1b represents the process in step 4. It displays that the method adds must-links and cannot-links among data instances according to the answers from the source of expertise. The green double-sided arrows are must-links and red double-sided arrows are cannot-links.
- Figure 3.1c is the updated clustering which satisfies all constraints in Figure 3.1b.
- Figure 3.1d shows the effect of feature learning in step 5. It shows that by applying feature learning methods on the updated clustering and the current constraint set, the feature distribution can be more divisible.

3.2 Format of Constraints

Different from previous works like [21], where the constraints are between data points from different clusters, our constraints are between data points belonging to the same cluster because our constraints are relationships between the data point and the center. Therefore, we do not care about the specific cluster label and instead, we only care about whether the data point and its current center belong to the same category. This is the first strategy in our method to avoid the *cluster label reassignment* problem, which will be discussed later. Therefore, the constraints preserved in our algorithm are like:

$$Q_{i,j} = \begin{cases} 0, (i,j) \notin M, (i,j) \notin C \\ 1, (i,j) \in M \\ -1, (i,j) \in C \end{cases} \tag{3.1}$$

Where Q is an $N \times N$ matrix, and N is the number of data points. M is the set of $Must\text{-}Link$ constraints, and C is the set of $Cannot\text{-}Link$ constraints. $Q_{i,j}$ indicates the constraint between the ith data point and the jth data point. When $Q_{i,j} = 1$, the ith and the jth data point must be assigned to one cluster; when $Q_{i,j} = -1$, the ith and the jth data point must be assigned to different clusters; when $Q_{i,j} = 0$, there is no constraint between the ith and the jth data point.

3.3 Loss Function

The loss function is a critical part in designing an algorithm, it represents how you hope the data are used. There are a lot of loss functions utilized in different deep neural networks for various aims, such as Mean Square Error (MSE) Loss [53], Cross Entropy Loss [50] and Hinge Loss [52]. As we mentioned before, there is a problem in the iterative clustering process and we call it *cluster label reassignment*. Take k-means as an example, when we apply k-means on exactly the same dataset multiple times, we will get lots of different results. Because k-means initializes cluster centers at random, and the clustering depends a lot on the initialization.

We notice the reassignment problem when using the softmax loss to compute the multinomial logistic loss between network predictions and clustering labels. The clustering label for the same data point can be completely different each time, which confuses the network during the subsequent fine-tuning. We try to use a Confusion Matrix [49] to find the co-relation between labels from two continuous assignments and to alleviate the problem. The results are not optimal because it is hard to decide which data instances should be used to maintain the cluster label. Inspired by [34], we notice that the *triplet loss* is a loss function that ignores exact label values and can help solve the cluster reassignment problem. As

figure 3.2 shows, the triplet loss minimizes the distance between an *anchor* and a *positive*, which belong to the same cluster; and maximizes the distance between an *anchor* and a *negative*, which belong to different clusters.

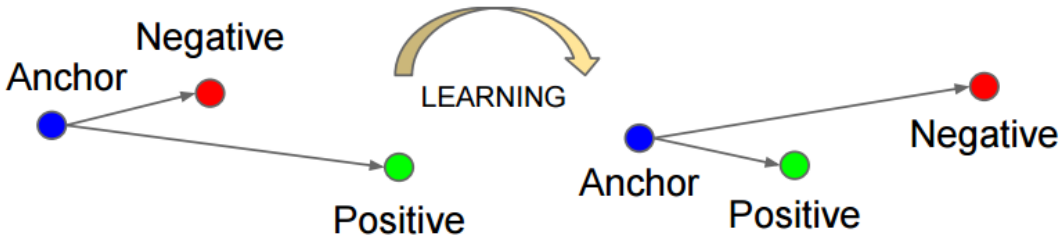


Figure 3.2: The triplet loss minimizes the distance between an anchor and a positive that have the same identity; it maximizes the distance between the anchor and a negative that have different identities [34].

The triplet loss function to be minimized looks like:

$$L = \sum_{i}^{N}[||f(x_i^a) - f(s_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2] + \alpha \tag{3.2}$$

And it is usually changed into a hinged format:

$$L = \max(\sum_{i}^{N}[||f(x_i^a) - f(s_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2] + \alpha, 0) \tag{3.3}$$

Here α is the enforced margin between positive and negative pairs. $||f(x_i^a) - f(s_i^p)||_2^2$ represents the distance between an *anchor* and a *positive*, and $||f(x_i^a) - f(x_i^n)||_2^2$ represents the distance between an *anchor* and a *negative*. We can also understand the network in the view of distance metric learning. It means the network is trying to learn an embedding or a metric function to map the raw input space to a linearly separable one.

By applying the triplet loss, we can exclude the influence of the label reassignment problem and take advantage of relationships among *anchors*, *positives* and *negatives*. The relationship between an *anchor* and a *positive* can presented as a *must-link*, and that of an *anchor* and a *negative* can be presented as a *cannot-link*. Conversely, any data point involved in a *must-link* is a potential *anchor* or *positive*, and any data point involved in a *cannot-link* with a *anchor* or a *positive* is a *negative* candidate.

3.4 Hard Triplets

Negative examples are beneficial for feature learning, but the amount of useful information contained in different negative examples is not equal. In order to maximize the effect of

different examples, we select hard negative and hard positive examples other than random ones. For example, actions with different backgrounds are hard examples compared to actions with a similar background. They can provide extra supervision and lead to a more robust model when used for training a network, here "more robust" means a higher "background independence". Generally speaking, hard examples contribute mainly to deciding the accurate boundary between groups in a partition problem.

In our method, the constrained clustering algorithm will choose data points according to their distances from the corresponding center. If the data point is not involved in any existing constraint, the algorithm will query its relationship with the center and add a new constraint to the current constraint set. The process will continue until it finds a certain number of new constraints. According to different constraints between the data point and the center, the data point can be a positive or a negative candidate and our algorithm will automatically add a must-link or cannot-link between them. In this way, we can maintain a negative data set and a positive data set for building hard triplets.

3.5 Proposed Method

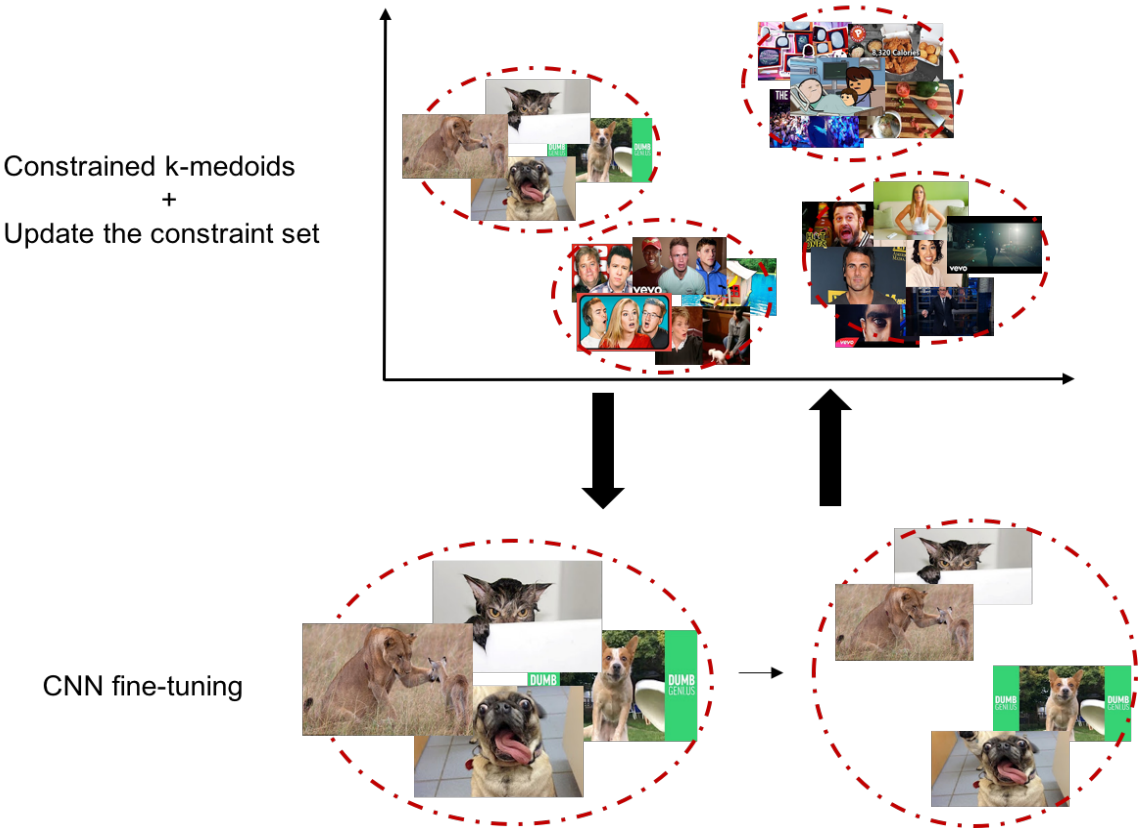


Figure 3.3: Illustration of the overall method

Our method iterates between the constrained k-medoids algorithm and the CNN fine-tuning.

The constrained clustering used in the thesis is called the constrained k-medoids algorithm. In k-means, a cluster center is the average of all data points that belong to the same cluster, so the center is not necessarily a real data point. We make a minor modification by always selecting data points as cluster centers, which is similar to the k-medoids algorithm [20]. Therefore, we call our constrained clustering algorithm "constrained k-medoids" that works with the l_2 distance and instance-level constraints.

Our constraints are obtained by actively querying the source of expertise according to Chapter 3.4, and different sources of expertise can lead to different constraints. To reduce the probability of constraint violation happening, we utilize the principle of CLK-Kmeans by limiting the number of cannot-links that contain the same data point.

Once the constrained partition process completes, our algorithm searches for "hard" positive and "hard" negative data points according to the distance to the center as Figure 4.5b. Here negative data points are instances that belong to a different category with the center and positive data points are instances that belong to the same category with the center.

For each cluster, starting from the nearest data point to the center, the algorithm will ask whether it belongs to the same category with the center if the data point is not contained in any existing constraint yet. If "yes", the algorithm will add a must-link between the data point and the center; or it will add a cannot-link between them and update the data point's label to be the opposite value. When the number of new constraints reaches a pre-defined value, the algorithm will stop searching the nearest data points and turn to the furthest points. Follow the similar procedure, our algorithm will add the same number of constraints to the current constraint set.

The clustering result along with the current constraint set will be used to form triplets for fine-tuning the network. When building triplets, only data points that are involved in at least one constraint will be considered as training data. Our algorithm treats all data points with the positive label as candidates of anchors or positives, and data points with the negative label are candidates of negatives.

While the convergence property of the traditional k-means has been proved in [6], the convergence of the combination of constrained k-medoids algorithm and feature learning method is not certain because of constraints and changeable feature distribution.

After finishing the fine-tuning, we can extract new representations using the fine-tuned network and perform the constrained k-medoids algorithm on the new features again. The process will iterate until one end requirement is met. In our experiments, the end requirement is when the number of iterations reaches a pre-defined value.

Our method is described in Algorithm 8 in detail:

Algorithm 8: Proposed Method

Input : Video set X, pre-trained network N, CLK-Kmedoids algorithm A, the number of new constraints n in each cycle

Output: Cluster assignments L

repeat
  Extract feature x_i of each video X_i using N
  Cluster video features using A to get clustering c_1 and create a constraint set
  Record indexes of data points affected by any user feedback in U
  for *each data point* $p \in P$ **do**
    if p *is the nearest or furthest point that* $(p, c_p) \notin M$ *and* $(p, c_p) \notin C$ **then**
      Ask the expert whether p and its center c_p belong to a same action category
      if *YES* **then**
        add (p, c_p) to M
      else
        add (p, c_p) to C, $l_p = -l_a$
      Add p and c_p to U, until n constraints from nearest and furthest data points are discovered respectively
  Build a triplet set T in each mini batch during training according to:
  for *each* i *in* U **do**
    if $l_i > 0$ **then**
      build a triplet with the i-th data point as the anchor,
      the j-th ($j \in U$ and $j \neq i$ and $l_j > 0$) data point as the positive,
      the k-th ($k \in U$ and $k \neq i$ and $l_k == -l_i$) data point as the negative,
      add (x_i, x_j, x_k) to T
  Fine tune network N using T
until *the number of iterations reaches a pre-defined value*;

Chapter 4

Experiments

4.1 Dataset

In this thesis, we apply our method on a subset of UCF-101 [36], which is a challenging dataset of action videos. UCF-101 is collected based on YouTube videos, containing up to 101 different action categories. The large number of clips and the unconstrained nature of videos increase the difficulty of obtaining a good performance on it. In Figure 4.1, some sample frames from the first 36 categories of UCF-101 are displayed. The order is provided by the dataset. In the thesis, we select 1374 videos from the 10 first categories according to the same order as our dataset.

Figure 4.1: Sample frames for the first 36 action categories from UCF-101 [36]

4.2 Experimental Results

At first, we would like to figure out whether our constraint selection strategy can improve the performance of constrained k-medoids algorithm under different conditions.

There are lots of measurements to evaluate the performance of a clustering, such as *purity* [13], *normalized mutual information* [13], *random index* [54] and *f measure* [13].

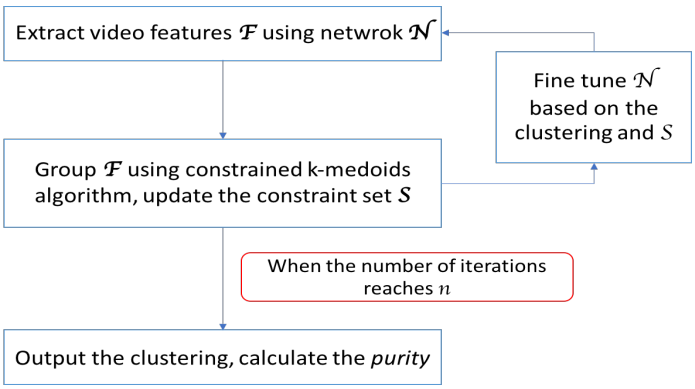


Figure 4.2: Experiment process

Among them, *purity* is a simple and transparent external evaluation measurement, so we utilize *purity* to evaluate the clustering result. *Purity* represents the percent of the total number of data points that are classified correctly, ranging continuously from 0 to 1. It is defined as:

$$purity(\Omega, C) = \frac{1}{N}\sum_{i=1}^{K}\max_{j}|\omega_i \cap c_j| \tag{4.1}$$

where $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ is the set of clusters and $C = \{c_1, c_2, ..., c_J\}$ is the set of classes.

4.2.1 Effectiveness of User Feedback Only

To investigate the effectiveness of user feedback only, we run constrained k-medoids algorithm starting from 10 random initialization sets, and keep adding user feedback through different strategies. Every time we add a fixed number of new constraints to the current constraint set randomly or according to Algorithm 8. The total number of constraints in the dataset is $1374 \times 1373 = 1886502$. Given Figure 4.3 and Figure 4.4, after adding $20 \times 20 = 400$ constraints, the average purity reaches 0.88 and after adding $40 \times 20 = 800$ constraints the average purity reaches 0.96. It takes several minutes for the constrained k-medoids algorithm to converge when the initialization is correctly selected. Here a correct initialization means that any two of the 10 initial centers are not involved in a must-link.

From Figure 4.3 and Figure 4.4 we can see the purity difference between random constraints and chosen ones. The difference maintains when we tried 10 different initialization sets. Compared to that there is no apparent improvement in the clustering performance from random constraints, constraints selected by our algorithm can make sure the increase trend, proving the strategy's effectiveness.

The first reason behind the bad performance of random constraints is that the random selection strategy could easily inquiry the relationship between the cluster center and "easy"

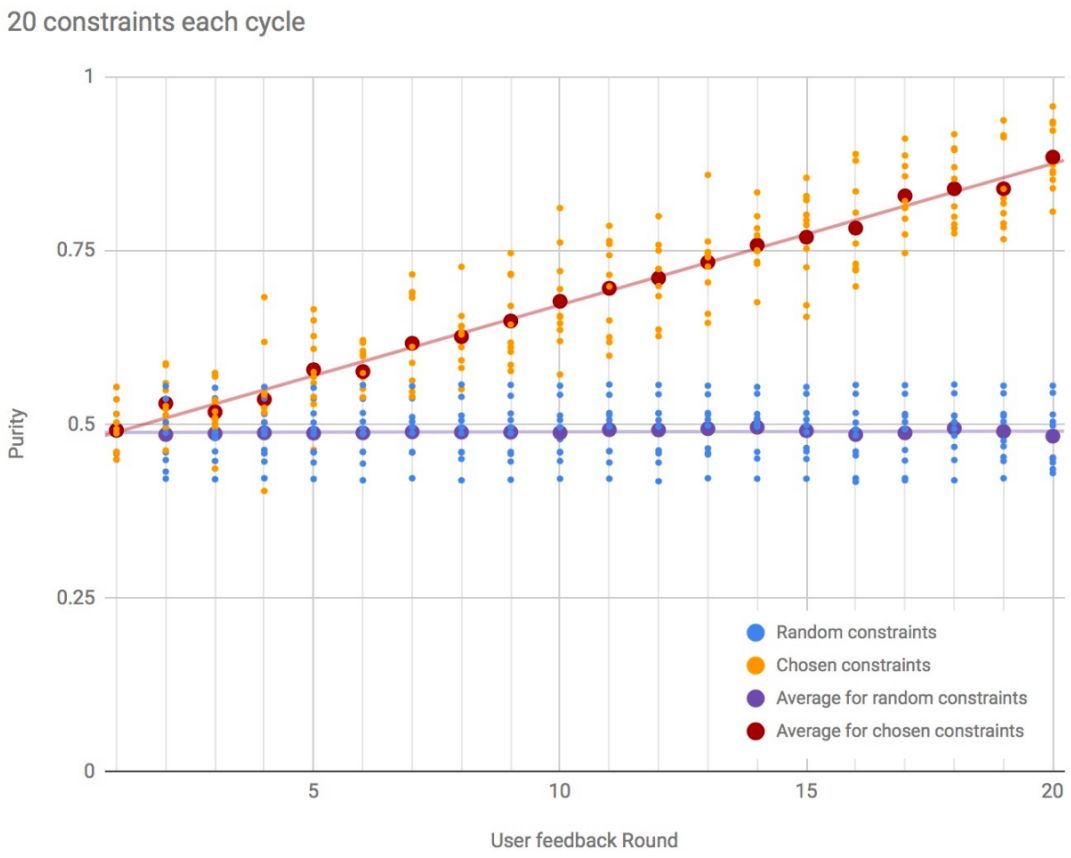


Figure 4.3: Clustering performance with 20 random or chosen constraints added in each cycle starting from 10 random initialization sets, where orange points represent results of random constraints, blue points represent results of chosen constraints. Dark red and dark blue points represent the average from two algorithms with the corresponding trend line respectively.

data points. "Easy" means the algorithm can correctly cluster these data points on itself. Therefore, these constraints can hardly help strengthen the clustering algorithm. Nevertheless, our algorithm can provide more useful information to avoid assigning negatives with a small distance from the center to the same cluster with the center or assigning positives with a large distance from the center to a different cluster.

The second reason that random constraints are much worse than chosen constraints is shown in Figure 4.5. In Figure 4.5a, the constraints are selected randomly and it is hard to apply the *transitive* property because they are related to totally different data instances. However, when the constraints are selected according to our strategy like in Figure 4.5b, according to Chapter 2.2.1, must-links are transitive. The constraints selected by our strategy are all related to the same point, the cluster center, therefore, they can refer much more constraints than random ones like Figure 4.5c.

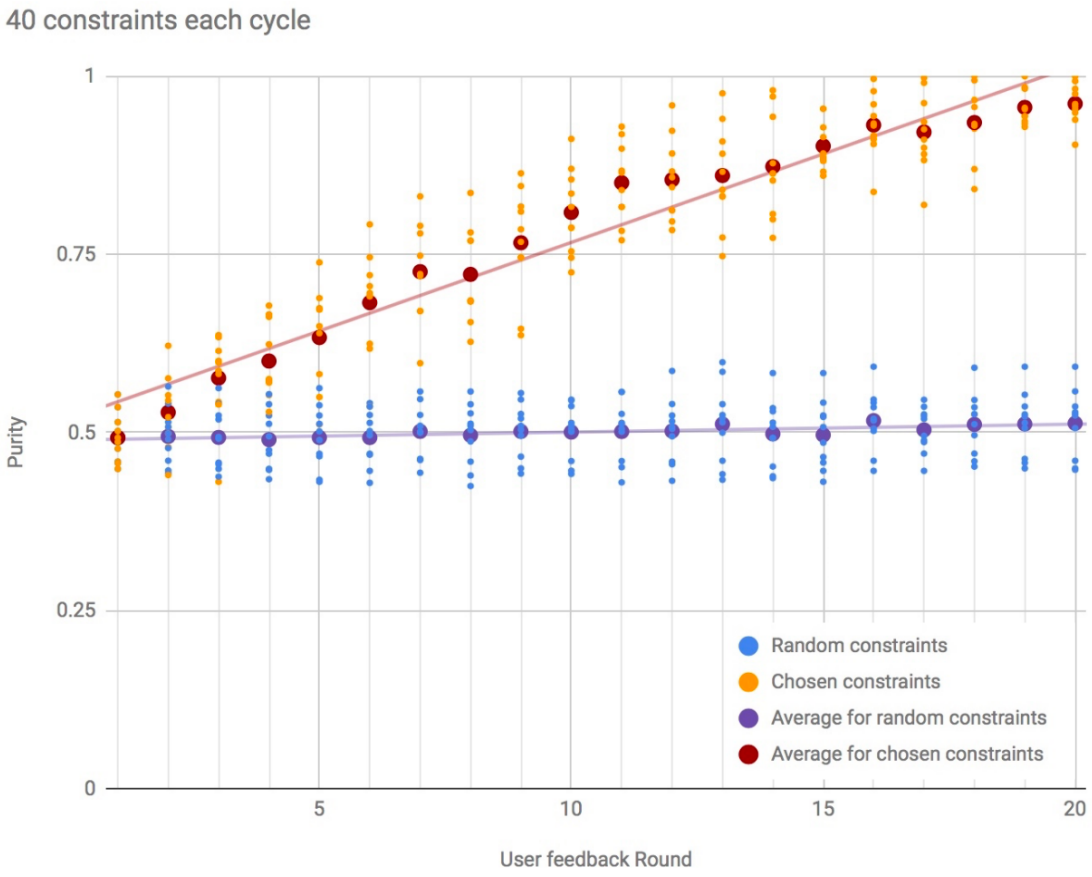
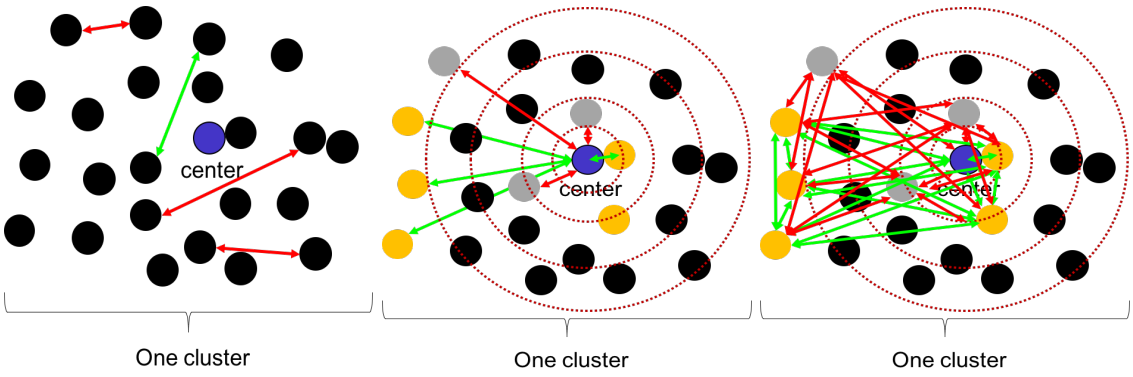


Figure 4.4: Clustering performance with 40 random or chosen constraints added in each cycle starting from 10 random initialization sets, where orange points represent results of random constraints, blue points represent results of chosen constraints. Dark red and dark blue points represent the average from two algorithms with the corresponding trend line respectively.

Based on the experimental results and the analysis above, our constraint selection algorithm is effective and correct to select more valuable constraints for improving the clustering performance. It is noted that the experiments are based on the condition that the constraints from the source of expertise are totally correct. And the advantage of random constraints is they might be more robust when there is any wrong user feedback provided.

In addition, from the result we find that extra constraints are not always beneficial to the clustering performance, especially when they are randomly selected. This leads to our next discussion about measuring the constraints.



(a) Random constraints (b) Chosen constraints (c) Transitive constraints

Figure 4.5: Difference between random constraints and chosen constraints. The dark blue point represents the cluster center. The red double-sided arrows represent cannot-links and the green double-sided arrows represent must-links. Figure 4.5a displays the random constraints among data points, then Figure 4.5b is the result of chosen constraints and Figure 4.5c is the transitive result of constraints in 4.5b. In Figure 4.5b and Figure 4.5c, red dashed circles represent different distances from the cluster center. Grey points are data instances belonging to the different category with the center, and orange points are those belonging to the same category with the center.

4.2.2 Measurement of User Feedback

Researchers have found there can be a large variation when utilizing constraints to improve the clustering performance. [43] and [9] proposed two constraint measures that are strongly correlated with the constrained clustering algorithm performance:

- *Informativeness*: The amount of information in the constraint set that the algorithm cannot determine on its own.
- *Coherence*: The amount of internal agreement between constraints in the constraint set, given a distance metric.

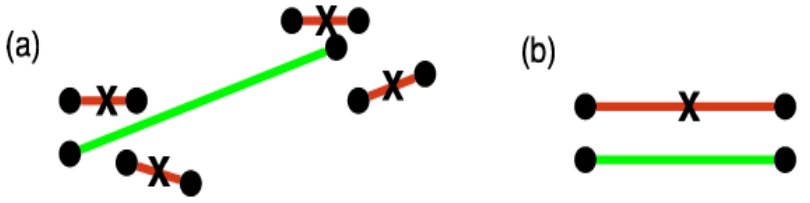


Figure 4.6: Illustrative examples of constraints with (a)high informativeness and (b)low coherence, given a Euclidean distance metric [9]. Green lines represent must-links, and red lines with "X" represent cannot-links.

Figure 4.6 illustrates the examples of two situations. In Figure 4.6, must-links are depicted as solid lines; red cannot-links have an 'X' through them. Figure 4.6(a) shows

constraints with high informativeness because the distance of the must-link is larger than that of cannot-links, which is against the distance metric and the clustering algorithm cannot get the conclusions by itself. Figure 4.6(b) shows two different constraints that are very close and parallel, so the constraints are incoherent.

According to [9] that *constraint sets with high informativeness and high coherence are most likely to provide performance gains.* On one hand, our algorithm to obtain the user feedback can greatly improve the proportion of constraints with higher informativeness from two aspects:

- (Hard negative) Finding a data point not belonging to the same category with the center while close to the center.
- (Hard positive) Finding a data point belonging to the same category with the center but far from the center.

These constraints are of high informativeness because the clustering algorithm cannot determine them on its own. At the same time, due to the transitiveness of must-links, one hard negative point can lead to multiple constraint pairs with high informativeness. This explains why constraints selected by our algorithm can be much more powerful than random ones, which is coherent with our experimental results.

On the other hand, the proportion of constraints with low coherence can be affected as well. When a negative data point a and a positive data point b are found around the boundary of this cluster, two constraints formed by a and b with the center respectively are nearly parallel while having the opposite meaning. These constraints with low coherence can decrease the clustering performance.

4.2.3 Effectiveness of Feature Learning

After analyzing the reason that extra constraints cannot always lead to performance improvement, which is mainly caused by incoherent constraints. We find this drawback can be overcome by refining the distance metric, for example, when the distance difference between must-links and cannot-links is large enough, the constraints will be of high coherence.

In this part, we explore how feature learning based on CNNs can improve the clustering performance by refining the distance metric, and whether the constrained clustering result can guide feature learning in return. Our experiments are based on 1374 videos from UCF-101 [36] dataset, feature learning is based on a hybrid network between Caffe net [18] and Zeiler & Fergus [60]. The network is pre-trained on ImageNet [24], and we use the constrained k-medoids algorithm mentioned before.

Because of the different number of constraints in each fine-tuning, it is hard to decide a unified set of hyperparameters for all experiments if we want to achieve the best state in each fine-tuning. Therefore, we run a 10-fold cross-validation experiment to select an

Figure 4.7: Clustering performance with 20 constraints added in each cycle performing feature learning or not based on 10 random initialization sets, where blue points represent results of constraints only, orange points represent results of the combination of constraints and feature learning. Dark blue and dark red points represent the average from two algorithms with the corresponding trend line.

effective but small learning rate, a margin α in the triplet loss as well as the number of training epochs, which can push the feature learning slightly in case of over-fitting. In the end, the network is fine-tuned on a single GeForce GTX 970 GPU machine using the stochastic gradient descent with a learning rate of 10^{-9} and a margin of 1000 in the triplet loss for only 1 epoch each time. In addition, as the number of constraints keeps increasing, the time of fine-tuning increases as well. The first fine-tuning costs around half of an hour to complete while the last one costs a couple of hours.

Follow the diagram in Figure 4.2, we alternatively update the clustering and the network. We call the process that clusters the feature and updates the network for one time a "cycle". In each cycle, the network is only fine-tuned for 1 epoch, meaning that each training frame will be seen only once by the network. In order to be comparable with the experiments without feature learning, we run 20 cycles under all conditions, whose performances are displayed in Figure 4.7, Figure 4.8 and Figure 4.10. Here the value of 20 can be any number, we simply select one intuitively.

We compare the clustering performance between the constrained clustering with or without feature learning in Figure 4.7 and 4.8, adding 20 and 40 constraints each cycle respec-

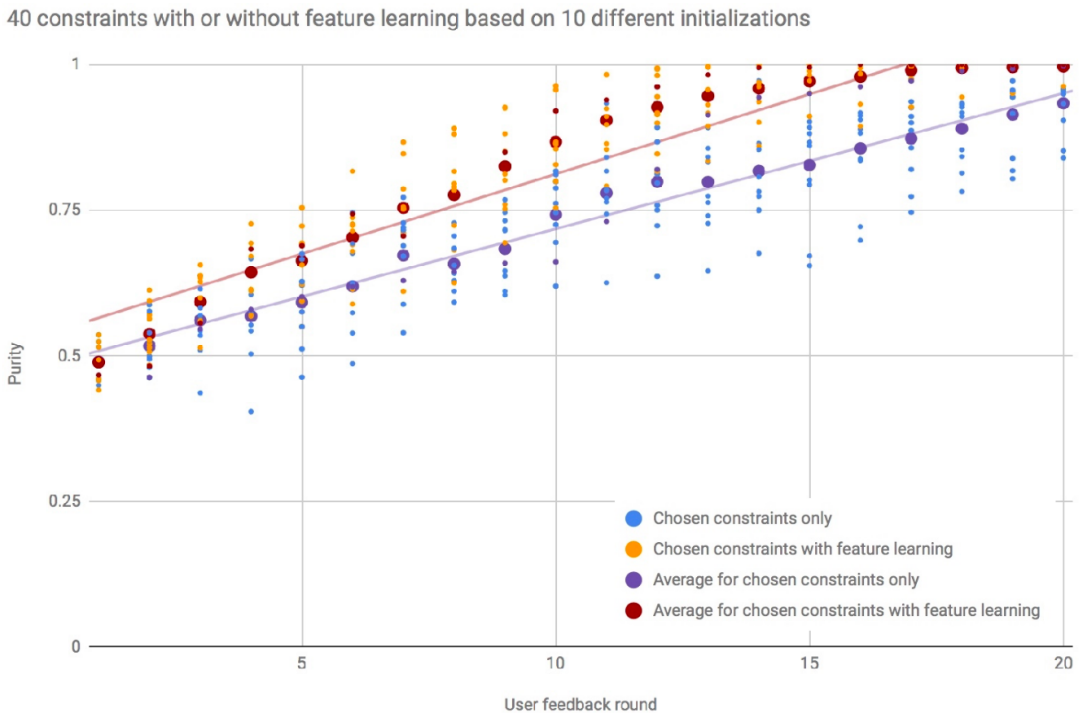
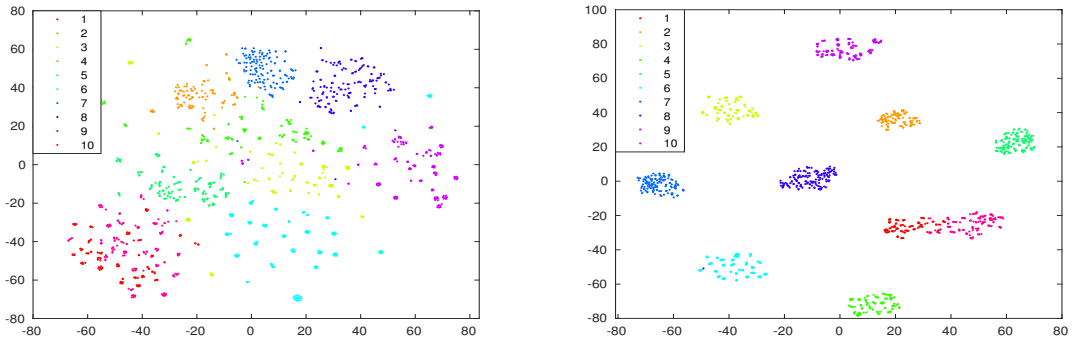


Figure 4.8: Clustering performance with 40 constraints added in each cycle performing feature learning or not based on 10 random initialization sets, where blue points represent results of constraints only, orange points represent results of the combination of constraints and feature learning. Dark blue and dark red points represent the average from two algorithms with the corresponding trend line.

tively. The results demonstrate that feature learning is capable of accelerating the clustering performance improvement and obtaining higher performance with fewer constraints. And when more constraints are added, the average clustering performance is higher.

The superiority of feature learning appears at a very young stage, and it is enlarged gradually when the number of constraints increases. To explain this phenomenon, we visualize the feature without feature learning and after the final fine-tuning in Figure 4.9 using t-SNE [29].

In Figure 4.9a the initial feature distribution is highly disordered has no pattern to follow. If we do not apply feature learning to refine the feature distribution, every time the algorithm will try to find 10 clusters based the chaotic distribution, and it will be hard even with lots of constraints. In this case, the probability that the constraints selected by our algorithm are not coherent enough is much higher than Figure 4.9b. After feature learning, the distance between data points belonging to the same category tends to decrease and that of data points belonging to different categories will increase. The feature distribution becomes cleaner and easier to find potential patterns. This change leads to the increase in the coherence of selected constraints, contributing to the larger improvement.



(a) Initial feature (b) Final feature

Figure 4.9: Feature visualizations

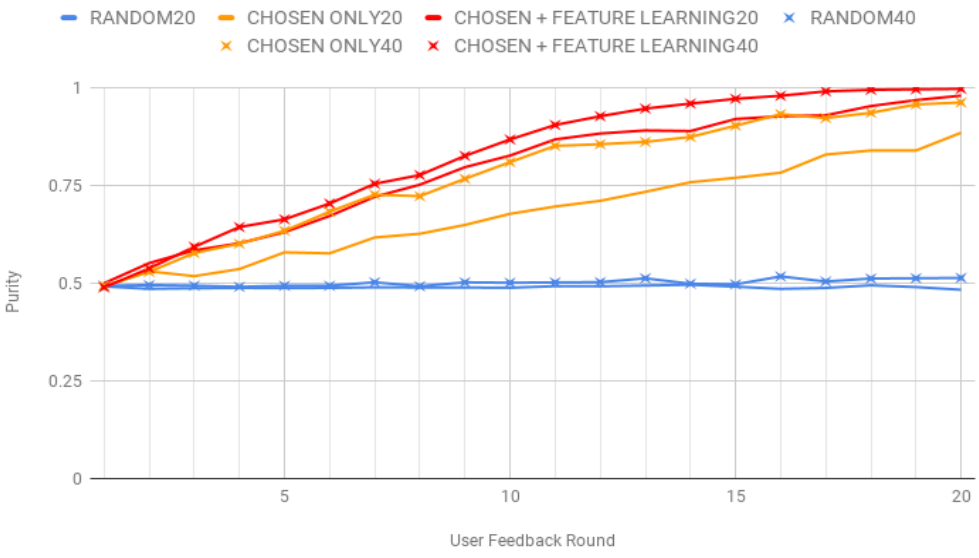


Figure 4.10: Average performances under different conditions from 10 random initialization sets

From Figure 4.10 the constrained clustering method with 40 constraints added in each cycle and feature learning gains the best average performance. According to our experimental results and analysis, we conclude that:

- Our constraint selection algorithm can find more valuable constraints and utilize transitiveness of constraints better than random selection strategy.
- Feature learning can overcome the low coherence of constraints resulted from the disordered feature distribution.
- With all other conditions same, more constraints can lead to a better performance.

Chapter 5

Summary

5.1 Conclusion

In this thesis, we came up with a solution to improve the performance of constrained k-medoids algorithm and feature learning based on the CNN fine-tuning.

Starting from regular clustering algorithms, we choose constrained clustering because of its ability to combine the limited domain knowledge. We design an algorithm to effectively select more valuable constraints than random ones in an active learning way.

The experimental results demonstrate that the constraints selected by our strategy can accelerate the clustering performance improvement, and this conclusion is on condition that all constraints provided by the source of expertise are correct.

Because the constraint quality is affected by the feature divisibility, we utilize the clustering and constraints to refine the feature by fine-tuning a CNN. A special loss function, triplet loss, is chosen to avoid the cluster reassignment problem during the fine-tuning. While the constrained clustering process takes only several minutes, it can take up to a couple of hours for fine tuning a CNN.

In the end, we evaluate the integrated method on a challenging action video dataset, and the experimental results prove our method is able to improve the constrained clustering performance and increase the feature divisibility efficiently.

5.2 Future Work

There are several possible directions that could be explored in the future work to improve the algorithm further.

At first, our method works based on the hypothesis that the user feedback is completely correct, and the effectiveness of constraints selected by our strategy highly depends on the fact as well. In the next step, we can add some noise to the experiment by importing wrong constraints to explore the robustness of random constraint select strategy and our strategy.

Then the time expense of our method is high because the CNN fine-tuning is slow. In future experiments, we can try smaller networks or apply network prune to decrease the network complexity. However, there should be a trade-off between the improvement over speed and the decrease in performance.

Besides, the necessity of iterations needs to be explored as well by providing all the constraints at once and watching the difference in the clustering performance.

Also, we can select constraints with high informativeness and high coherence only. And if we change the method of obtaining new user feedback from cluster verification to cluster relabel method, we can get more information, but still, there needs to be a trade-off between the performance improvement and the complexity increase.

Finally, we can obtain more accurate video features by concentrating on areas where the action happens, which can be achieved by placing an action detector before the feature extractor.

Therefore, we believe the performance can be advanced again when heading towards these directions.

Bibliography

[1] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[2] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4945–4949. IEEE, 2016.

[3] Sugato Basu, Ian Davidson, and Kiri Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press, 2008.

[4] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[5] Mikhail Bilenko, Sugato Basu, and Raymond J Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM, 2004.

[6] Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Advances in neural information processing systems*, pages 585–592, 1995.

[7] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[8] Ian Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 138–149. SIAM, 2005.

[9] Ian Davidson, Kiri L Wagstaff, and Sugato Basu. Measuring constraint-set utility for partitional clustering algorithms. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 115–126. Springer, 2006.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[11] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[12] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[13] Stanford NLP Group. Evaluation of clustering.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[15] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[16] Minh Hoai and Fernando De la Torre. Maximum margin temporal clustering.

[17] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[18] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[19] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[20] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[21] Mehran Khodabandeh, Arash Vahdat, Guang-Tong Zhou, Hossein Hajimirsadeghi, Mehrsan Javan Roshtkhari, Greg Mori, and Stephen Se. Discovering human interactions in videos with limited data labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–18, 2015.

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[25] Zhengzhong Lan, Ming Lin, Xuanchong Li, Alex G Hauptmann, and Bhiksha Raj. Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 204–212, 2015.

[26] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[27] Zhenguo Li, Jianzhuang Liu, and Xiaoou Tang. Constrained clustering via spectral regularization. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 421–428. IEEE, 2009.

[28] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[29] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[32] Carlos Ruiz, Myra Spiliopoulou, and Ernestina Menasalvas. C-dbscan: Density-based clustering with constraints. In *International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing*, pages 216–223. Springer, 2007.

[33] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014.

[34] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[35] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[36] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[38] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013.

[39] Anthony KH Tung, Jiawei Han, Laks VS Lakshmanan, and Raymond T Ng. Constraint-based clustering in large databases. In *ICDT*, volume 1, pages 405–419. Springer, 2001.

[40] Aditya Vailaya, Anil K. Jain, and Hongjiang Zhang. Video clustering.

[41] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[42] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[43] Kiri L Wagstaff, Sugato Basu, and Ian Davidson. When is constrained clustering beneficial, and why? *Ionosphere*, 58(60.1):62–63, 2006.

[44] Fei Wang, Chris Ding, and Tao Li. Integrated kl (k-means–laplacian) clustering: a new clustering approach by combining attribute data and pairwise relations. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 38–48. SIAM, 2009.

[45] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3169–3176. IEEE, 2011.

[46] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE international conference on computer vision*, pages 3551–3558, 2013.

[47] Xiang Wang and Ian Davidson. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 563–572. ACM, 2010.

[48] David Weltman. *A comparison of traditional and active learning methods: An empirical investigation utilizing a linear mixed model*. The University of Texas at Arlington, 2007.

[49] Wikipedia. Confusion matrix — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Confusion%20matrix&oldid=764152469`, 2017. [Online; accessed 27-February-2017].

[50] Wikipedia. Cross entropy — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Cross%20entropy&oldid=753407109`, 2017. [Online; accessed 25-February-2017].

[51] Wikipedia. ExpectationâĂŞmaximization algorithm — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Expectation%E2%80%93maximization%20algorithm&oldid=794220949`, 2017. [Online; accessed 28-August-2017].

[52] Wikipedia. Hinge loss — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Hinge%20loss&oldid=741695220`, 2017. [Online; accessed 25-February-2017].

[53] Wikipedia. Mean squared error — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Mean%20squared%20error&oldid=767240164`, 2017. [Online; accessed 25-February-2017].

[54] Wikipedia. Rand index — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Rand%20index&oldid=765944540`, 2017. [Online; accessed 12-March-2017].

[55] Pengtao Xie and Eric P Xing. Integrating image clustering and codebook learning. 2015.

[56] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

[57] Rong Yan, Jian Zhang, Jie Yang, and Alexander G Hauptmann. A discriminative learning framework with pairwise constraints for video object classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):578–593, 2006.

[58] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.

[59] Yan Yang, Tonny Rutayisire, Chao Lin, Tianrui Li, and Fei Teng. An improved copkmeans clustering for solving constraint violation based on mapreduce framework. *Fundamenta Informaticae*, 126(4):301–318, 2013.

[60] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.