FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF MASTER'S THESIS

Title:	Active Semi-supervised Clustering
Student:	Bc. Jakub Švehla
Supervisor:	Ing. Tomáš Borovička
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Theoretical Computer Science
Validity:	Until the end of summer semester 2018/19

Instructions

Semi-supervised clustering methods use partially labelled data or constraints in the form of must- or cannot-links to improve the performance of cluster analysis. In an active learning setting, labels and constraints that reduce uncertainty are identified and added iteratively by a domain expert.

1) Review and theoretically describe state of the art semi-supervised clustering approaches with an emphasis on methods that integrate the active learning concept.
2) Use or implement at least three methods and experimentally compare their performance on various data sets. Avoid implementing anew those methods that can be easily taken over from available implementations.
3) Propose directions for further improvements of reviewed methods.

References

Will be provided by the supervisor.

doc. Ing. Jan Janoušek, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague January 16, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Active Semi-Supervised Clustering

Bc. Jakub Švehla

Department of Theoretical Computer Science
Supervisor: Ing. Tomáš Borovička

May 9, 2018

Acknowledgements

I would like to thank my supervisor Ing. Tomáš Borovička for his endless help and support. I am also grateful to my family and friends for support during writing this thesis.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 9, 2018 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Švehla, Jakub. *Active Semi-Supervised Clustering.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Shlukování dat je velice náročný problém, protože v mnoha případech existuje mnoho možných způsobů rozdělení daného datasetu. Shlukování je proto výrazně subjektivní a závislé na daném problému. Aktivní semi-supervisované shlukovací metody aktivně získávají znalost o daném datasetu tak, aby docílily co nejlepšího shlukování pro daný problém.

V této práci analyzujeme několik aktivních semi-supervizovaných shlukovacích metod s důrazem na metody, které využívají informace o omezeních dvojic bodů. Dále analyzujeme tři metody pro aktivní učení těchto omezení. Se všemi metodami byly provedeny experimenty na několika různých datasetech. Výsledky experimentů ukazují, že aktivní semi-supervizované metody výrazně zlepšují kvalitu shlukování oproti běžným shlukovacím metodám. Nicméně, žádná ze zkoumaných metod není lepší než ostatní metody na všech datasetech. V této práci navrhujeme další směry pro vylepšení zkoumaných metod.

Klíčová slova shlukování, semi-supervizované učení, aktivní učení, pairwise constraints

Abstract

Clustering is a challenging problem since there usually exist multiple possible clusterings. This makes it strongly problem dependent and subjective. Active semi-supervised clustering methods are designed to actively ask for background knowledge in order to provide the best clustering for a given problem.

This thesis reviews several state of the art semi-supervised clustering methods with emphasis on methods utilizing pairwise constraints and three schemes for active learning of pairwise constraints. Experiments are conducted to empirically evaluate all reviewed methods on various data sets. Results of the experiments show that active semi-supervised clustering significantly outperforms unsupervised clustering in terms of agreement with a reference clustering. However, none of the methods is superior to the other reviewed methods on all data sets. In the thesis, further directions for extending the current methods are proposed.

Keywords clustering, semi-supervised learning, active learning, pairwise constraints

Contents

List of Figures

List of Tables

Introduction

Clustering is a challenging problem since there always exist multiple possible clusterings of a data set. That means that clustering is strongly problem dependent and subjective. This motivates providing additional information that could steer the clustering in the desirable direction. Semi-supervised learning is a technique that makes use of both labeled and unlabeled data to train a model. It can be used alongside clustering in order to make use of labeled data or other forms of background knowledge. However, obtaining background knowledge from a domain expert is often time consuming and expensive. Hence, it is desirable to ask only about the data points that the clustering algorithm is the least certain about and thus minimize the number of queries needed to improve the clustering. Active learning is a technique that allows the learning algorithm to actively query the domain expert about the information that is the most useful for it. Combining both active learning and semi-supervised methods allows actively asking for background knowledge in order to provide the best clustering for a given problem.

There are two main approaches to semi-supervised clustering. The first approach utilizes partially labeled data similarly to semi-supervised classification. However, obtaining labeled data is often difficult because the domain expert might not know how many and which classes are in the data. This leads to the other approach that utilizes pairwise constraints to aid the clustering. Pairwise constraints provide information about whether a pair of points should be in the same or different clusters.

Pairwise constraints are the most useful background knowledge for clustering, therefore this work focuses on active learning of pairwise constraints. There are two main approaches to active learning of the pairwise constraints. The first one asks for the constraints prior to the clustering. The other one actively asks the domain expert in order to iteratively improve the clustering.

The goal of this thesis is to review several state of the art active semi-supervised clustering methods and experimentally evaluate them on various data sets. The main emphasis is on semi-supervised clustering methods uti-

lizing background knowledge in the form of pairwise constraints and methods for active learning of the pairwise constraints.

The thesis is structured as follows. In Chapter 1, we give an introduction to clustering, distance measures and ways of evaluating clustering quality. In Chapter 2, we review types of background knowledge for clustering and review state of the art methods for both semi-supervised clustering using partially labeled data and pairwise constraints. In Chapter 3, we review the methods for active learning of pairwise constraints. In Chapter 4, we describe our implementation of the reviewed methods and review existing implementations. In Chapter 5, we experimentally evaluate all reviewed semi-supervised clustering and active learning methods on various data sets and discuss the results. Finally, in Chapter 6, we propose directions for further research of the reviewed methods.

Chapter 1

Introduction to clustering

Clustering is an unsupervised machine learning technique whose goal is to group a data set into groups, called clusters, so that objects in the same cluster are as similar to each other as possible and objects in the different cluster as dissimilar as possible. It is used in data mining for data exploration, finding useful patterns and sorting out large data sets into coherent groups. Clustering is used for example for grouping genomes in bioinformatics, customer segmentation in marketing, categorizing documents in text processing, image processing and others.

1.1 Clustering methods

Clustering methods can be categorized based on the model that they use to represent the clusters. The two most common methods are hierarchical and partitional methods that will be described in Section 1.1.1 and 1.1.2, respectively. Beside those two groups, there are other methods that are not so widely used such as distribution-based, density-based or graph-based methods.

Clustering methods can be also divided based on whether they produce a hard/crisp or soft/fuzzy clustering. In case of hard/crips clustering, the data is split into disjoint clusters and each point belongs to exactly one cluster. On the other hand, in case of soft/fuzzy clustering, the clusters may overlap and each point may partially belong to more than one cluster. In this thesis, we will focus only on the hard clustering methods since they are the most widely used.

1.1.1 Hierarchical clustering

Hierarchical clustering organizes a data set into a natural hierarchy. It successively merges data points and smaller clusters into larger clusters or, on the other hand, splits larger clusters into smaller ones, which results into a hierarchy of clusters. At the lowest level of the hierarchy, each data point is in

its own cluster. At the highest level, all the data points are in a single cluster. Each level of the hierarchy represents a grouping into a particular number of disjoint groups.

The hierarchy is usually visualized using a binary tree called dendrogram (Figure 1.1). In the dendrogram, the height of each node represents the dissimilarity of its two child clusters. Based on the dendrogram, we can decide how many clusters are there in the data set and hence where to cut the tree to get disjoint clusters.

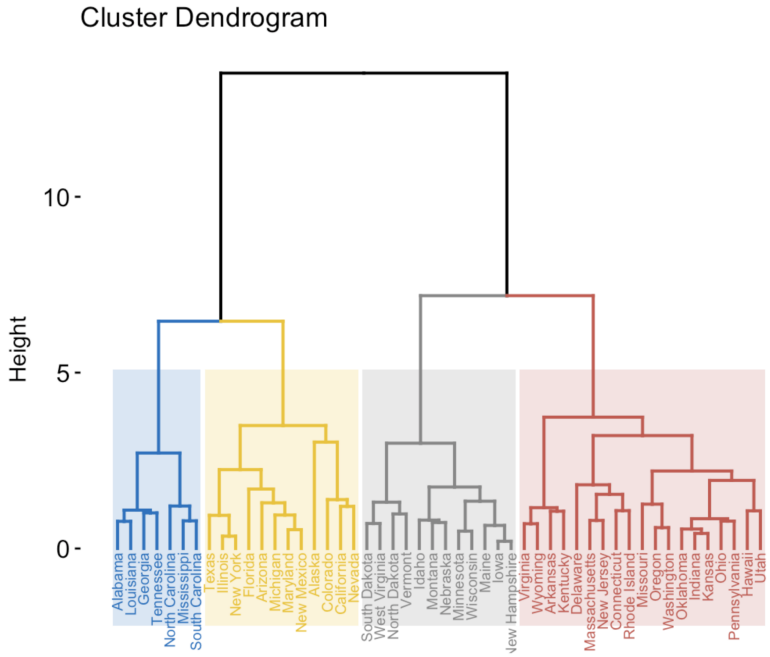

Figure 1.1: Dendrogram; source: `https://www.r-bloggers.com/practical-guide-to-cluster-analysis-in-r-book/`

There are two approaches to hierarchical clustering. The first one is agglomerative hierarchical clustering, which is a bottom-up approach where each data point starts in its own cluster and then they are recursively grouped into larger clusters. The other one is divisive hierarchical clustering, which is a top-down approach and, on the other hand, starts with one cluster containing all the data points and recursively divides it into smaller clusters.

The main advantage of the hierarchical clustering is that we do not need to know the number of clusters beforehand and that the dendrogram might correspond to a meaningful taxonomy of the data set.

The main disadvantage of the hierarchical clustering is its high computational complexity. In general, the agglomerative hierarchical clustering has a time complexity of $\mathcal{O}(n^3)$ and the divisive hierarchical clustering has a time

complexity of $\mathcal{O}(2^n)$, which makes both algorithms infeasible for larger data sets.

1.1.2 Partitional clustering

Partitional clustering groups data into a fixed number of disjoint set of clusters. The main criteria is to minimize a dissimilarity of the samples within each cluster and maximize the dissimilarity between the clusters. It is also referred to as centroid-based clustering since the clusters are typically represented by their centroid.

The most widely used partitional clustering method is K-Means clustering, which will be described in detail in Section 1.4.

1.2 Distance measures

One of the challenging tasks of clustering is defining a distance measure, i.e. defining which objects are similar and which are not. Choosing the right distance measure is difficult because what we consider similar is highly subjective and problem-dependent. For example, one might find two cars similar if they have the same make and others might find them similar if they have the same color. In this section, various distance measures that we can choose from will be described.

The right choice of the distance measure is important because it influences the shape of the clusters and the quality of the resulting clustering. It can be chosen manually by a human expert or learnt by an algorithm from the data. In this section, the most common distance measures will be described.

A distance function, also called a distance metric, is a function $f\colon X \times X \to \mathbb{R}$ that defines a distance between each pair of elements of a set X. A set with a metric is called a metric set. For each $x, y, z \in X$, a metric d satisfies the following properties:

1. $d(x, y) \geq 0$ (nonnegativity),
2. $d(x, y) = 0 \Leftrightarrow x = y$ (identity of indiscernibles),
3. $d(x, y) = d(y, x)$ (symmetry),
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

1.2.1 Minkowski distances

Minkowski distances, also called p-norm distances, are a family of distances induced by ℓ_p norms.

Definition 1.2.1 (Minkowski distance) *The Minkowski distance of two vectors* $x = (x_1, \ldots, x_d)^T$ *and* $y = (y_1, \ldots, y_d)^T$ *is defined as*

$$\|x - y\|_p = \left(\sum_i^d (x_i - y_i)^p \right)^{1/p},$$

where $p \geq 1$ *is a real number. For* $p < 1$ *it violates the triangle inequality and thus is not a metric.*

There are several widely used distance measures that are special cases of the Minkowski distance: Manhattan distance for $p = 1$, Euclidean distance for $p = 2$ and Chebyshev distance, also called maximum metric, for $p \to \infty$.

Definition 1.2.2 (Manhattan distance) *The Manhattan distance of two vectors* $x = (x_1, \ldots, x_d)^T$ *and* $y = (y_1, \ldots, y_d)^T$ *is defined as*

$$\|x - y\| = \sum_i^d |x_i - y_i|.$$

Definition 1.2.3 (Euclidean distance) *The Euclidean distance of two vectors* $x = (x_1, \ldots, x_d)^T$ *and* $y = (y_1, \ldots, y_d)^T$ *is defined as*

$$\|x - y\|_2 = \sqrt{\left(\sum_i^d (x_i - y_i)^2 \right)}.$$

The Euclidean distance can be also defined using vector multiplication as

$$d(x, y) = \sqrt{(x - y)^T (x - y)}.$$

In optimization problems, squared Euclidean distance, which is defined as

$$\|x - y\|_2^2 = \sum_i^d (x_i - y_i)^2 = (x - y)^T (x - y),$$

is sometimes used instead. However, it is not a metric since it does not satisfy the triangular inequality. That is not a problem in optimization problems where the distances have to be compared only with each other.

Definition 1.2.4 (Chebyshev distance) *The Chebyshev distance of two vectors* $x = (x_1, \ldots, x_d)^T$ *and* $y = (y_1, \ldots, y_d)^T$ *is defined as*

$$\|x - y\|_\infty = \max_i^d |x_i - y_i|.$$

1.2.2 Mahalanobis distance

The Mahalanobis distance is defined as a distance measure between two random vectors of the same distribution. It can also be used to measure a distance between a random vector and a distribution.

Definition 1.2.5 (Mahalanobis distance) *Let S be a covariance matrix. The Mahalanobis distance of two vectors $x = (x_1, \ldots, x_d)^T$ and $y = (y_1, \ldots, y_d)^T$ is defined as*

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}.$$

If the covariance matrix is the identity matrix then the Mahalanobis distance is equal to the Euclidean distance. If the covariance matrix is diagonal the distance is equal to a standardized Euclidean distance which is defined as

$$d(x, y) = \sqrt{\sum_{i}^{d} \frac{(x_i - y_i)^2}{s_i}},$$

where s_i is the standard deviation of the x_i and y_i over the sample set.

1.3 Clustering evaluation

In order to evaluate the quality of a clustering, one has to choose the right evaluation measure. As with other aspects of clustering, choosing the right measure is dependent on a given problem.

There are three main approaches to evaluating clustering results. The first one is internal evaluation where the clustering is summarized into a quality score based only on the data that were used for the clustering. As an example, it can measure how coherent and separate from each other the clusters are. The problem with the internal evaluation is that it may return a high score even though the clustering is not useful for the end user and vice versa. The second one is external evaluation where the clustering is compared to a known ground truth. However, in practice, external evaluation is not used often because usually the ground truth is not known. If it was known we would not have to do the clustering in the first place and could have used a supervised learning instead. Nevertheless, it is useful for comparing different clustering methods on known data sets. Another shortcoming is that the ground truth may not be the only clustering possible, i.e. there may be more plausible clusterings of the same data set. In that case, the external evaluation would assign a low score to the different clustering even though it may be useful as well. The third approach is manual evaluation by a human expert. However, this method is time consuming for larger data sets and may not be objective.

In this section, several internal and external criteria will be described.

1.3.1 Internal criteria

1.3.1.1 Calinski-Harabasz score

Calinski-Harabasz score [1] is a ratio between the within-cluster variability and the between-cluster variability

$$CH = \frac{\text{between-cluster variability}}{\text{within-cluster variability}},$$

which is analogous to the F-test statistics in the one-way analysis of variance (ANOVA).

The between-cluster variability is defined as

$$\sum_{k=1}^{K} n_k \frac{(\mu_k - \bar{X})^2}{K-1}$$

where μ_k is the mean of the cluster k, $\bar{X}$ is the overall data mean, n_k is the number of samples in cluster k and K is number of clusters.

The within-cluster variability is defined as

$$\sum_{k=1}^{K} \sum_{x \in X_k} \frac{(x - \mu_k)^2}{n-K}$$

where X_k is a set of points in the cluster k and n is the total number of data points.

The Calinski-Harabasz score is not bounded and the higher the better the clustering is.

1.3.1.2 Davies-Bouldin index

Davies-Bouldin index [2] is defined as

$$DB = \frac{1}{K} \sum_{k=1}^{K} \max_{k \neq k'} \frac{\sigma_k + \sigma_{k'}}{d(\mu_k, \mu_{k'})}$$

where σ_k is the average distance of all points in cluster k to its centroid. The index is in the range from 0 to 1 and the lower the score the better.

1.3.1.3 Dunn index

Dunn index [3] is a ratio between minimal inter-cluster distance and maximal intra-cluster distance

$$D = \frac{\min_{1 \leq k < k' \leq K} d_{\text{inter}}(X_k, X_{k'})}{\max_{k=1}^{K} d_{\text{intra}}(X_k)}.$$

Both distances may be chosen arbitrarily. For example, distance between cluster centroids can be used as the inter-cluster distance and the maximum

distance between any pair of points in the cluster can be used as the intra-cluster distance.

1.3.1.4 Silhouette coefficient

Silhouette coefficient is calculated for each data point separately and measures how similar a point is to its own cluster compared to other clusters. It is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

where $b(i)$ is the lowest average distance of point i to all points in any other cluster and $a(i)$ is the average distance between i and all other points in the same cluster. The coefficient is in the range from -1 to 1 and the value close to 1 means that the point is correctly clustered.

In order to get the overall clustering score, the coefficients of all points in the data set are averaged.

1.3.2 External criteria

External criteria are used to compare assignment of data points into *classes* according to a known ground truth and assignment into *clusters* determined by a clustering algorithm.

Finding out whether a data point was assigned to the right cluster is difficult even if we have the ground truth because it is not clear which cluster corresponds to which class. However, we can instead look at a pair of points and see whether they are in the same class and in the same cluster. This way we can find out whether the points that are in the same class are also in the same cluster.

For the sake of brevity, let A be a number of pairs of points that are in the same class and in the same cluster, B be a number of pairs of points that are in different classes and in different clusters, C be a number of pairs of points that are in the same class but in different clusters and D be a number of pairs of points that are in different classes but in the same cluster.

Hence, $A + B$ can be viewed as the number of agreements between the ground truth and the clustering and $C + D$ as the number of disagreements between the two. Finally, $A + B + C + D$ is equal to the total number of pair of points which is equal to $\binom{n}{2} = n(n-1)/2$.

1.3.2.1 Rand index

Rand index [4] is defined as

$$R = \frac{A + B}{A + B + C + D} = \frac{A + B}{\binom{n}{2}}$$

and corresponds to the probability that the clustering and the ground truth agrees on a randomly chosen pair of points.

1.3.2.2 Adjusted Rand index

Adjusted Rand index [5] is the corrected-for-chance version of the Rand index.

Given partitionings $\{X_i\}_{i=1}^{r}$ and $\{Y_i\}_{i=1}^{s}$ of a set S and a contingency table

	Y_1	Y_2	$\dots$	Y_s	
X_1	n_{11}	n_{12}	$\dots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\dots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\dots$	n_{rs}	a_r
	b_1	b_2	$\dots$	b_s	

where $n_{ij} = |X_i \cap Y_j|$, the Adjusted Rand index is defined as

$$AR = \frac{\sum_{ij} \binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}.$$

1.4 K-means clustering

K-means clustering is one of the most well known and widely used partitional clustering methods. According to [6], it "is by far the most popular clustering algorithm used in scientific and industrial applications" and one of the ten most used data mining techniques according to [7]. In this section, we will describe two most widely used K-means algorithms and methods for initializing its intitial cluster centroids.

1.4.0.1 Problem definition

In the clustering problem, we are given an input data set

$$X = \{x_1, x_2, \dots, x_n\},$$

where n is number of data points and each element is a d-dimensional vector $x_i \in \mathbb{R}^d$ such that

$$x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{bmatrix},$$

where d is number of variables. As an example, x_{ij} denotes a jth variable of ith data point.

Output of a clustering algorithm is a set of cluster labels for each data point $C = \{c_1, c_2, \ldots, c_n\}$, where $c_i \in \{k\}_{k=1}^{K}$ is a cluster label corresponding to the data point x_i.

Unless otherwise noted, the notation established in this section will be used throughout the whole thesis.

1.4.0.2 Algorithm

K-means' objective is to minimize the within-cluster sum of squares (WCSS), which is defined as a sum of squared distances between each point in the cluster and its cluster mean, called centroid.

Formally, given a data set X, it finds a K-partitioning $\{X_k\}_{k=1}^{K}$ of the data set X minimizing the cost function

$$WCSS = \sum_{k=1}^{K} \sum_{x \in X_k} \|x - \mu_k\|_2 \tag{1.1}$$

where $\mu_k \in \mathbb{R}^d$ is the centroid of the cluster formed by a partition X_k, i.e. mean of all points belonging to a cluster k.

Finding a partitioning that optimizes WCSS is a NP-complete optimization problem, which makes finding optimal solutions infeasible. However, there exist approximate algorithms, commonly referred to as K-means algorithms, that find a partitioning that locally minimizes the WCSS.

One of the approximate algorithms is Lloyd's algorithm [8], which was first introduced in 1957 at Bell Labs but was not published until 1982. The algorithm works as follows. It first initializes the centroids (see Section 1.4.1 for more details on centroid initialization). Then, it iteratively alternates between the following two steps until convergence of the centroids:

1. (Assignment step) Assign each data point x_i to its closest cluster

$$c_i = \arg\min_{k=1}^{K} \|x_i - \mu_k\|_2.$$

2. (Estimation step) Recalculate each cluster centroid μ_k as a mean of its points

$$\mu_k = \frac{1}{|X_k|} \sum_{x \in X_k} x.$$

The algorithm is guaranteed to converge to a local minimum.

The running time of the Lloyd's algorithm is $\mathcal{O}(kndi)$ where k is the number of clusters, n is the number of d-dimensional vectors and i is the number of iterations needed for convergence.

In 1967, James MacQueen published a similar algorithm called MacQueen's algorithm [9] and coined the term K-means for the first time. The only difference between Lloyd's algorithm is that it recalculates the centroids after each point is reassigned to a new cluster. Due to this, the MacQueen's algorithm converges faster but each iteration takes longer. Lloyd's algorithm is also referred to as a batch algorithm as opposed to MacQueen's algorithm which is referred to as an online algorithm. Since Lloyd's algorithm is more popular in practice, when we refer to K-means algorithm in this thesis we refer to Lloyd's algorithm, unless noted otherwise.

1.4.1 Centroid initialization

Initial cluster centroids highly influence the results of the K-means algorithm and hence a lot of research effort was put into finding heuristics for initializing the clusters in a way that would lead to better local optimums.

1.4.1.1 Random initialization

The most common and probably the most simple way is to initialize the centroids randomly. That can be done either by choosing K random data points, sampling the centroids from a multivariate uniform distribution within the ranges of the variables or by perturbing the global mean of X. Of course, it can happen by chance that the centroids will be initialized in a way that leads to a suboptimal local minimum so in practice the clustering algorithm is usually restarted randomly multiple times and the best clustering (in terms of its objective function) is chosen.

1.4.1.2 Farthest first traversal

The farthest first traversal heuristic finds points that are maximally separated from each other and hence well spread across the data space. The algorithm starts by choosing a point randomly and keeps a set of traversed points S. In each following step, it chooses the point that is the farthest from all the already traversed points. The distance of a point and the traversed set S is defined as a distance of the point to the closest point in the set. Formally, it is defined as

$$d(x, S) = \min_{y \in S} d(x, y).$$

The algorithm is illustrated in Figure 1.2.

One of the main disadvantages of this algorithm is that it is sensitive to outliers in the data set.

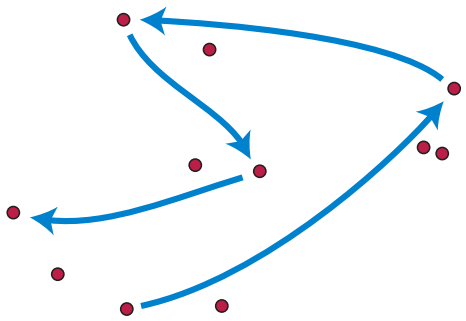

Figure 1.2: Farthest-first traversal; by David Eppstein - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=41569871

1.4.1.3 KMeans++

KMeans++ [10] is an extension of the standard K-means algorithm which uses a heuristic to initialize the cluster centroids similar to the farthest first traversal method. However, instead of choosing the most distant point in each step, it chooses the next point randomly with a probability proportional to the distance of the point to the traversed set. Formally, the probability of choosing a point x is defined as

$$p(x) = \frac{d(x, S)}{\sum_{x' \notin S} d(x', S)}.$$

Authors of the algorithm show that the initialization itself is already a good clustering which gets only better with each iteration.

Chapter 2

Semi-supervised clustering

In this chapter, semi-supervised clustering methods utilizing background knowledge to aid the clustering will be introduced. In Section 2.1, we will explain what background knowledge is and its types. Then, we will introduce the semi-supervised methods. In Section 2.2, we will describe methods using partially labeled data and in Section 2.3, we will describe methods using instance-level constraints.

In this thesis, we focus only on crisp, partitional clustering methods. Specifically, we focus on methods based on the K-means algorithm since it is one of the most widely used clustering algorithms due to its simplicity and efficiency.

2.1 Types of background knowledge

Clustering is an unsupervised learning technique that tries to find useful patterns in data. Unsupervised learning means that there are no correct answers / labels given to the learning algorithm, as opposed to supervised learning algorithms. However, sometimes, there is some background knowledge about the structure of the data that can help the algorithm. This background knowledge is usually provided by a domain expert. Problems where we are given such background knowledge are called semi-supervised learning problems. In case of clustering, there are three main types of background knowledge that might be given to the algorithm. The first one is partial class labels, the second one is cluster-level constraints and the third one is instance-level constraints. In this section, we will look into all three types in detail.

2.1.1 Partially labeled data

Background knowledge in the form of partially labeled data is similar to the information given to supervised learning algorithms because it is given information about classes that the data points belong to. However, unlike in the

supervised setting, the labels are given only to a small subset of the whole data set.

An example of partially labeled data, consider a problem of classifying emails as spam or not spam. There is a vast amount of emails sent every day but only a small fraction of them are classified as spam or not spam by a human. However, we can make use of both the labeled and not labeled data to find a structure in them.

2.1.2 Cluster-level constraints

Another type of background knowledge, which is specific to clustering, is cluster-level constraints. Cluster-level constraints provide information about attributes of the clusters such as a number, size or shape of the clusters in the data. For example, the user might know that there are four clusters or that each cluster should have at most a certain number of items. In this section, we will describe few common cluster-level constraints.

2.1.2.1 Number of clusters

A constraint on number of clusters provides information about the number of clusters that are present in the data set. For example, a user might know that there are four types of customers.

2.1.2.2 Absolute/relative maximal/minimal size

This constraint provides background knowledge about the size of the clusters. It gives information about the maximum or minimum number of points in each cluster as an absolute number or relatively to a size of the whole data set.

2.1.2.3 Maximum diameter

Maximum diameter γ is defined as the maximum distance of any two points in the same cluster. It provides information about the intraclass similarity, i.e. about how much should be the points in the same cluster similar to each other.

2.1.2.4 δ-constraint

δ-constraint or minimum separation constraint [11] specifies that clusters must be separated by a distance of at least δ. It provides information about inter-class dissimilarity, i.e. how much should individual clusters be dissimilar to each other.

2.1.2.5 ϵ-constraint

ϵ-constraint [11] specifies that for each point in a cluster there must be another point in the same cluster that is within a distance of ϵ. Formally, for any point $x_a \in X_k$ there must exist a point $x_b \in X_k$ such that $d(x_a, x_b) < \epsilon$.

2.1.3 Instance-level constraints

Instance-level constraints [12], also called pairwise constraints, provide background knowledge about a pair of points from the data set. There are two main types of the instance-level constraints. The first type is must-link (ML) constraints that indicate that a pair of points should be in the same cluster. The other type is cannot-link (CL) constraints that, on the other hand, indicate that two points should not be in the same cluster.

Formally, sets of must- and cannot-link constraints $M \subset X \times X$ and $C \subset X \times X$, respectively, define a symmetric binary relation over the set X. Both the must- and cannot-link constraints have different properties. In case of must-link constraints, the relation is transitive, i.e. if $(x, y) \in M$ and $(y, z) \in M$ then $(x, z) \in M$. In case of cannot-link constraints, the relation is not transitive. However, $(x, y) \in M$ and $(x, z) \in C$ implies $(y, z) \in C$. This is referred to as an entailment property.

Instance-level constraints, and binary relations in general, might be viewed as a graph $G = (V, E)$ where vertices $V \in X$ and an edge $e \in E$ between two vertices indicates whether the two points (represented by the vertices) should be in the same or different clusters in case of ML and CL constraints, respectively.

Typically, instance-level constraints are preprocessed before being used by a clustering algorithm. First, the transitive closure of the ML constraints is taken and added to the set of constraints. Second, inferred constraints based on the entailment property are added to the set of CL constraints.

Background knowledge provided by instance-level constraints is more general than that provided by partially labeled data. A set of labeled data points can be expressed using a set of ML constraints but not vice versa. In order to express the partially labeled data using the instance-level constraints, we can impose the ML constraint between each pair of points from the same class and the CL constraint between each pair of points from different classes.

Note that the instance-level constraints can be used to express some of the cluster-level constraints. The maximum diameter γ constraint can be expressed by setting cannot-link constraints for all pairs of points that have a distance greater or equal to γ. The δ-constraint (minimum separation constraint) can be expressed by setting must-link constraints for all pairs of points which distance is less than δ. The ϵ-constraint can be expressed by setting at least one constraint $(x_i, x_j) \in M$ for each data point x_i such that $d(x_i, x_j) \leq \epsilon$.

Input: Set of data points X, set of seed points $S = \cup_{k=1}^{K} S_k$, number of clusters K
Output: Disjoint K-partitioning of X

for $k \leftarrow 1, \ldots, K$ **do**
  $\mu_k \leftarrow \frac{1}{|S_k|} \sum_{x \in S_k} x$
end
repeat
  for $x_i \in X$ **do**
    $c_i \leftarrow \arg\min_{k=1}^{K} \|x_i - \mu_k\|_2$
  end
  for $k = 1, \ldots, K$ **do**
    $\mu_k \leftarrow \frac{1}{|X_k|} \sum_{x \in X_k} x$
  end
until *convergence*;

Algorithm 1: Seeded-KMeans algorithm

In this thesis, we are mainly focusing on the pairwise constraints since in practice, they are easier to obtain than the class labels because they do not require the domain expert to know about the underlying classes in the data set — the domain expert has to only be able to tell whether two instances should be in the same cluster or in different ones.

2.2 Methods using partially labeled data

This section describes clustering methods utilizing partially labeled data. Specifically, two algorithms based on K-means are presented: Seeded-KMeans and Constrained-KMeans. They are using a labeled subset of X to aid the clustering. Let $S \subset X$ be a seed set. For each point $x_i \in S$, called a seed point, a cluster index $c_i \in \{k\}_{k=1}^{K}$ to which the point x_i belongs is provided. This gives us a disjoint partitioning $\{S_k\}_{k=1}^{K}$, called seed clustering, which is used to guide the clustering. It is further assumed that at least one seed point corresponding to each partition X_i of X is provided.

2.2.1 Seeded-KMeans Algorithm

Seeded-KMeans [13] is using the seed set to initialize the cluster centroids instead of choosing them randomly. For each partition X_i, its centroid μ_i is calculated as the mean of all seed points $x \in S_i$, where S_i is a seed set corresponding to the partition X_i. The rest of the algorithm stays the same as in the K-means algorithm. The pseudocode of the algorithm is presented in Algorithm 1.

Input: Set of data points X, set of seed points $S = \cup_{k=1}^{K} S_k$, number of clusters K
Output: Disjoint K-partitioning of X

for $k \leftarrow 1, \ldots, K$ **do**
  $\mu_k \leftarrow \frac{1}{|S_k|} \sum_{x \in S_k} x$
end
repeat
  for $x_i \in X$ **do**
    if $\exists j \in \mathbb{N}, x_i \in S_j$ **then**
      $c_i \leftarrow j$
    else
      $c_i \leftarrow \arg\min_{k=1}^{K} \|x_i - \mu_k\|_2$
    end
  end
  for $k = 1, \ldots, K$ **do**
    $\mu_k \leftarrow \frac{1}{|X_k|} \sum_{x \in X_k} x$
  end
until *convergence*;

Algorithm 2: Constrained-KMeans

Seeded-KMeans assumes that there is at least one seed point for each cluster. However, it might be used even if we do not have a seed point for each cluster. In this case, cluster centroids are initialized randomly or by perturbing the mean of all points.

One of the main advantages of this algorithm is that it does not strictly enforce the labels because it is using them only for the initialization. This fact is useful when we have a weak supervision, i.e. a lot of incorrectly labeled data.

2.2.2 Constrained-KMeans Algorithm

In Constrained-KMeans [13], the cluster centroids are initialized the same way as in the Seeded-KMeans algorithm. However, in the assignment step, only the points $x \notin S$ are assigned to its closest cluster. Cluster membership of the seed points is kept unchanged. The pseudocode of the algorithm is presented in Algorithm 2.

One of the main disadvantages of the algorithm is that it does not handle noisy labels. In case of noisy labels, the algorithm keeps the incorrectly labeled points assigned to the wrong clusters and could steer the algorithm in an unintended way. In this case, Seeded-KMeans might be more suitable since it does not enforce assignment of the points to the clusters corresponding to its labels.

2.3 Methods using instance-level constraints

There are three main approaches to utilizing instance-level constraints by semi-supervised clustering methods. The first approach modifies the clustering algorithm itself in order to satisfy the constraints. Methods taking this approach are referred to as constraint-based methods and are described in detail in Section 2.3.1. The second approach is learning a suitable distance measure from the background knowledge and then using a standard unsupervised clustering algorithm. These methods are referred to as metric-based methods and are described in Section 2.3.2. The third approach combines constraint- and metric-based methods and is described in Section 2.3.3.

2.3.1 Constraint-based methods

Constraint-based methods modify the clustering algorithm itself in order to satisfy all or as many constraints as possible. Usually, in case of K-means algorithm, they do so by modifying the assignment step. There are two main ways to accomplish this. The first way is to enforce the constraints during the assignment step. An example of an algorithm from this group is COP-KMeans which will be described in Section 2.3.1.1. The other way is modifying the objective function so that violations of the constraints are penalized but not strictly enforced. An example of an algorithm from this group is PCK-Means which will be described in Section 2.3.1.2.

2.3.1.1 COP-KMeans algorithm

COP-KMeans [12] enforces the instance-level constraints during the assignment step. It does so by assigning each point to the closest cluster that does not violate any constraint. If, at any point, any legal cluster cannot be found for some point then no clustering is returned. This can happen even though there might exist a feasible clustering for the given constraints since finding a feasible clustering depends on the order in which the points are assigned to the clusters.

One of the main disadvantages of the COP-KMeans is that it does not handle noisy constraints. If there are any incorrect constraints (for example a must-link constraint on two points that should be in different clusters) the algorithm will enforce them even though it would lead to a worse clustering.

2.3.1.2 PCK-Means algorithm

PCK-Means [14] (pairwise constrained K-means) utilizes the instance-level constraints in two ways. First, it uses them in the initialization step to initialize the centroids. Second, it uses them in the assignment step to penalize any constraint violations.

In the initialization step, the algorithm uses the constraints to obtain λ neighbour sets $\{N_p\}_{p=1}^{\lambda}$ by taking the transitive closure of the ML constraints as described in Section 2.1.3. If $\lambda \geq K$, the algorithm selects K largest neighborhood sets and initialize K cluster centroids with the centroids of these sets. If $\lambda < K$, the algorithm initializes λ cluster centroids with the centroids of the sets. Then, it looks for a point x that is connected by cannot-links to every neighborhood set. If such a point exists, it is used to initialize $(\lambda+1)^{\text{th}}$ cluster. The remaining uninitialized cluster centroids are initialized by random perturbations of the mean of X.

In the assignment step, instead of minimizing WCSS (Equation 1.1), the algorithm minimizes the following objective function:

$$J_{\text{pckm}} = \frac{1}{2} \sum_{x_i \in X} \|x_i - \mu_{c_i}\|_2 \tag{2.1}$$

$$+ \sum_{(x_i,x_j)\in M} w_{ij} \mathbb{1}[c_i \neq c_j] \tag{2.2}$$

$$+ \sum_{(x_i,x_j)\in C} \overline{w}_{ij} \mathbb{1}[c_i = c_j], \tag{2.3}$$

where $w_{i,j}$ and $\overline{w}_{i,j}$ are costs for violating $(x_i, x_j) \in M$ and $(x_i, x_j) \in C$, respectively, and $\mathbb{1}$ is an indicator function which value is 1 if the condition is satisfied and 0 otherwise. The first term in the Equation 2.1 is equal to WCSS (Equation 1.1). The second and third term penalize ML and CL constraint violations, respectively.

The choice of the w and $\overline{w}$ parameters significantly influences behaviour of the algorithm. If the parameters are set to 0, the constraints are used only to initialize the centroids but not to penalize the violations. It is hence similar to the Seeded-KMeans algorithm. On the other hand, if the parameters are set to high values then the constraints become hard constraints (no violations are allowed) since the violations terms have much higher weight than the distance component. In this case, the algorithm is similar to the Constrainted-KMeans and COP-KMeans algorithms. If the parameters are set to an intermediate value, the algorithm makes a trade-off between minimizing the total distance between the points and clusters and the cost of violating the constraints.

One of the main advantages of the algorithm is that it can handle noisy constraints (since it does not enforce them, only penalize them) and always returns a solution, unlike COP-KMeans. Pseudocode of the algorithm is presented in Algorithm 3.

2.3.2 Metric-based methods

Metric-based methods utilize a technique known as metric learning which learns a distance measure from background knowledge.

Input: Set of data points X, set of must-link constraints M, set of cannot-link constraints C, number of clusters K
Output: Disjoint K-partitioning of X

create λ neighborhoods $\{N_p\}_{p=1}^{\lambda}$ from M and C;
if $\lambda \geq K$ **then**
  initialize the cluster centroids with the centroids of the K largest neighborhoods;
else
  initialize λ cluster centroids with the centroids of the neighborhoods;
  if there is a point x that is cannot-linked to all neighborhoods initialize $(\lambda+1)^{\text{th}}$ cluster centroid with x;
  initialize the remaining clusters at random;
end
repeat
  for $x_i \in X$ **do**
    $c_i \leftarrow \arg\min_{k=1}^{K} \frac{1}{2} \sum_{x_i \in X} \|x_i - \mu_{c_i}\|_2 + \sum_{(x_i,x_j)\in M} w_{ij} \mathbb{1}[c_i \neq c_j] + \sum_{(x_i,x_j)\in C} \overline{w}_{ij} \mathbb{1}[c_i = c_j]$
  end
  for $k = 1, \ldots, K$ **do**
    $\mu_k \leftarrow \frac{1}{|X_k|} \sum_{x \in X_k} x$
  end
until *convergence*;

Algorithm 3: PCK-Means algorithm

2.3.2.1 MK-Means algorithm

Utilizing metric learning alongside the standard K-means algorithm was first proposed in [15]. We refer to this method as MK-Means (metric K-means).

By using the metric learning, a metric is learned such that points that should be in the same cluster are close to each other and points that should not be in the same cluster are further apart. Let d_A be a distance measure in the form

$$d_A(x, y) = \|x - y\|_A = \sqrt{(x - y)^T A (x - y)},$$

where A must be a positive semidefinite matrix, $A \succeq 0$, in order for d_A to be a metric (otherwise it would not satisfy nonnegativity and the triangular inequality). The metric learning algorithm finds A by minimizing squared distances between the points that are linked with the must-link constraint, $\min_A \sum_{(x,y)\in M} \|x - y\|_A^2$. In order to avoid the trivial solution $A = 0$, which is not useful, the constraint $\sum_{(x,y)\in C} \|x - y\|_A \geq 1$ is added. If the set of cannot-link constraints is empty, we can use a set of pairs that are not in M

instead. This gives us the following optimization problem:

$$\begin{aligned} \min_{A} & \sum_{(x,y)\in M} \|x-y\|_A^2 \\ \text{s.t.} & \sum_{(x,y)\in C} \|x-y\|_A \geq 1 \\ & A \succeq 0. \end{aligned}$$

Minimizing A with respect to those constraints is a convex optimization problem. Two cases of the problem are considered separately: learning a diagonal A and learning a full A. If only a diagonal A is considered, an efficient algorithm using the Newton-Raphson method is used. The learned metric is equivalent to the weighted Euclidean distance. In case of learning a full A, the Newton-Raphson method becomes computationally too expensive. Hence, gradient ascent combined with the iterative projection algorithm [16] is used to optimize the following equivalent problem:

$$\begin{aligned} \max_{A} & \sum_{(x,y)\in C} \|x-y\|_A \\ \text{s.t.} & \sum_{(x,y)\in M} \|x-y\|_A^2 \leq 1 \\ & A \succeq 0. \end{aligned}$$

As we can see from the equality

$$\begin{aligned} d_A(x,y) &= \sqrt{(x-y)^T A(x-y)} \\ &= \sqrt{(x-y)^T L^T L(x-y)} \\ &= \sqrt{(Lx-Ly)^T(Lx-Ly)}, \end{aligned}$$

the distance measure d_A is equivalent to the Euclidean distance after applying the linear projection L to its inputs. Hence, after learning the distance measure, we can use the transformation matrix L to project the input data X and use the standard K-means algorithm (using the Euclidean distance).

2.3.3 Methods combining constraint- and metric-based methods

2.3.3.1 MPCK-Means algorithm

MPCK-Means [17] (metric pairwise constrained K-means) integrates both constraint- and metric-based approach. Same as pairwise constrained K-means, it allows constraint violations if it leads to a more cohesive clustering

(soft constraint satisfaction) so it is not prone to noisy supervision. It integrates metric learning into the clustering so that it can make use of both the instance-level constraints and unlabeled data to learn the metric, unlike MK-Means, which utilizes only the constraints. Moreover, it can learn a separate metric for each cluster, which allows different shapes of each cluster.

The objective function utilized by MPCK-Means is similar to J_{pckm} (Equation 2.1) that is utilized by PCK-Means. However, instead of using the standard Euclidean distance it uses the same parametrized Euclidean distance d_A as MK-Means. However, unlike MK-Means, it learns a separate metric d_{A_i} for each cluster. That is equivalent to a generalized K-means model where points in each cluster come from a multivariate Gaussian distribution with a covariance matrix A_i^{-1}. It can be shown that maximizing complete data log-likelihood, which is the data likelihood if we knew the true labels, under this model is equivalent to minimizing the objective function

$$J = \sum_{x_i \in X} \|x_i - \mu_i\|^2_{A_i} - \log\det A_i.$$

In PCK-Means, each constraint violation is penalized the same way regardless of the severeness of the violation. However, intuitively, violating a must-link constraint of distant points should be penalized more than violating a must-link constraint of nearby points because it means that a proper metric was not learned. To achieve that, each penalty for violating a ML constraint is multiplied by the average distance of the two points according to metrics of both their clusters. Similarly, violating a cannot-link constraint of nearby points should be penalized more than violating a cannot-link constraint of distant points. To achieve that, each penalty for violating a CL constraint is multiplied by a difference between the distance of the two most distant points and the distance of the points that violated the constraint. That guarantees that the penalty is always nonnegative and increasing with decreasing distance of the violating points.

Combining the objective function and constraint violation penalties described in previous paragraphs gives us the following objective function:

$$J_{\text{mpckm}} = \sum_{x_i \in X} \|x_i - \mu_{c_i}\|^2_{A_{c_i}} - \log\det A_{c_i} \tag{2.4}$$

$$+ \sum_{(x_i,x_j)\in M} w_{ij} f_M(x_i, x_j) \mathbb{1}[c_i \neq c_j] \tag{2.5}$$

$$+ \sum_{(x_i,x_j)\in C} \overline{w}_{ij} f_C(x_i, x_j) \mathbb{1}[c_i = c_j], \tag{2.6}$$

where f_M is the cost of violating a ML constraint

$$f_M(x_i, x_j) = \frac{1}{2}\|x_i - x_j\|^2_{A_{c_i}} + \frac{1}{2}\|x_i - x_j\|^2_{A_{c_j}}$$

and f_C is the cost of violating a CL constraint

$$f_C(x_i, x_j) = \|x'_{c_i} - x''_{c_i}\|^2_{A_{c_i}} - \|x_i - x_j\|^2_{A_{c_i}},$$

where (x'_k, x''_k) is the pair of the most distant points according to the metric of cluster k.

MPCK-Means locally minimizes the J_{mpckm} objective function by first utilizing the instance level constraints to initialize the cluster centroids and then alternating between the assignment step and the estimation step where it estimates both the cluster centroids and the cluster metrics. Each step is described in detail in the following paragraphs.

In the initialization step, MPCK-Means is using instance-level constraints to initialize the cluster centroids in a similar way to PCK-Means with few minor differences. It is using the must-link constraints to obtain λ neighbour sets $\{N_p\}_{p=1}^{\lambda}$ as in PCK-Means. If $\lambda \leq K$, it uses λ neighborhood sets to initialize cluster centroids with the neighborhood centroids. If $\lambda < K$, it initializes the remaining cluster centroids by random perturbations of the global mean of X, same way as in PCK-Means. If $\lambda > K$, instead of selecting K largest neighborhoods, the neighborhood sets are selected using the weighted farthest first traversal method. The weighted farthest first traversal method works the same way as the standard farthest first traversal but each point's distance to the traversed set is multiplied by the point's weight. In this case, points are the centroids of the neighborhoods and the weights are sizes of the neighborhoods in order to prefer choosing larger neighborhoods.

In the original paper, it is not mentioned how to initialize each cluster's metric. However, the simplest way would be to initialize each weight matrix with the identity matrix so that each cluster is initialized with the standard Euclidean distance.

In the assignment step, each point x_i is assigned to a cluster c_i that minimizes the objective function J_{mpckm}. Before each assignment step, all points are randomly reordered. The reordering is done due to the fact that the value of the objective function is order-dependent (since preceding assignments influence which constraints are being violated by following assignments). Authors experimented with the random ordering as well as a greedy strategy that first assigned points that are closest to the centroid and involved in a minimal number of constraints. According to the experiments, there were no significant differences in the clustering quality.

In the estimation step, the cluster centroids are first re-estimated as in standard K-means. Then, a metric d_{A_i} is re-estimated for each cluster i. The matrices A_i for each metric are obtained by taking partial derivative $\frac{\partial J_{\text{mpckm}}}{\partial A_i}$ and setting it to zero, which minimizes the objective function with respect to metric parameters. The resulting equation for updating the A_i is as follows:

$$A_k = |X_k|\Bigg(\sum_{x \in X_k} (x - \mu_k)(x - \mu_k)^T + \sum_{(x_i, x_j) \in M_k} \frac{1}{2} w_{ij} (x_i - x_j)(x_i - x_j)^T \mathbb{1}[c_i \neq c_j] + \sum_{(x_i, x_j) \in C_k} \overline{w}_{ij} \left((x'_k - x''_k)(x'_k - x''_k)^T - (x_i - x_j)(x_i - x_j)^T \right) \mathbb{1}[c_i = c_j] \Bigg)^{-1},$$

where M_k and C_k are sets of must- and cannot-link constraints that contain points that are assigned to cluster k.

In some cases, A_k^{-1} may not be invertible (singular). If that is the case, it can be made invertible by adding the identity matrix multiplied by a small fraction of the trace of A_k^{-1} [18]:

$$A_k^{-1} \leftarrow A_k^{-1} + \epsilon \operatorname{tr}(A_k^{-1}) I.$$

If the resulting weight matrix is not positive semidefinite, a projection of A, $\operatorname{Proj}_C(A)$, onto a set of positive semidefinite matrices $C = \{A : A \succeq 0\}$ is done as follows. First, we do eigen (spectral) decomposition of the weight matrix $A = U \operatorname{diag}(\lambda_1, \ldots, \lambda_d) U^T$, where $\lambda_1 \geq \cdots \geq \lambda_d$, are sorted eigenvalues of A and U is a corresponding orthonormal matrix of eigenvectors of A. Then, we can do the projection as follows:

$$\operatorname{Proj}_C(A) = U \operatorname{diag}(\max(0, \lambda_1), \ldots, \max(0, \lambda_d))\, U^T.$$

In other words, it takes all its negative eigenvalues and sets them to zero in order for $\operatorname{Proj}_C(A)$ to be positive semidefinite. (A matrix is positive semidefinite if and only if all its eigenvalues are nonnegative.)

Due to the time complexity of learning full weight matrix for each cluster, authors of the algorithm also propose modifications of the algorithm that learn only a diagonal weight matrix for each cluster, a single full weight matrix for all clusters or a single diagonal matrix for all clusters.

Pseudocode of the algorithm is presented in Algorithm 4.

Input: Set of data points X, set of must-link constraints M, set of cannot-link constraints C, number of clusters K
Output: Disjoint K-partitioning of X

create λ neighborhoods $\{N_p\}_{p=1}^{\lambda}$ from M and C;
if $\lambda \geq K$ **then**
    initialize cluster centroids using weighted farthest-first traversal of the neighborhoods centroids;
else
    initialize λ cluster centroids with centroids of the neighborhoods;
    initialize remaining clusters at random;
end
repeat
    for $x^{(i)} \in X$ **do**
        $c_i \leftarrow \arg\min_{k=1}^{K} \sum_{x_i \in X} \|x_i - \mu_k\|_{A_k}^2 - \log\det A_k$
        $+ \sum_{(x_i,x_j)\in M} w_{ij} f_M(x_i, x_j) \mathbb{1}[k \neq c_j]$
        $+ \sum_{(x_i,x_j)\in C} \overline{w}_{ij} f_C(x_i, x_j) \mathbb{1}[k = c_j]$
    end
    for $k = 1, \ldots, K$ **do**
        $\mu_k \leftarrow \frac{1}{|X_k|} \sum_{x \in X_k} x$
    end
    for $k = 1, \ldots, K$ **do**
        $A_k^{-1} = \sum_{x \in X_k} (x - \mu_k)(x - \mu_k)^T$
        $+ \sum_{(x_i,x_j)\in M_k} \frac{1}{2} w_{ij} (x_i - x_j)(x_i - x_j)^T \mathbb{1}[c_i \neq c_j]$
        $+ \sum_{(x_i,x_j)\in C_k} \overline{w}_{ij} \left((x'_k - x''_k)(x'_k - x''_k)^T - (x_i - x_j)(x_i - x_j)^T \right) \mathbb{1}[c_i = c_j]$;
        $A_k = |X_k| (A_k^{-1})^{-1}$
    end
until *convergence*;

Algorithm 4: MPCK-Means algorithm

Chapter 3

Active learning of pairwise constraints

In the semi-supervised clustering setting, getting background knowledge in the form of pairwise constraints might be expensive and time consuming. Moreover, improper selection of constraints might actually degrade the resulting clustering instead of improving it as shown in [19] and [20]. Hence, it is desirable to ask for as few constraints as possible that would improve the clustering the most. This setting where the system actively queries an oracle, which is usually a domain expert, in order to obtain the most informative background knowledge is called active learning. In case of active learning of pairwise constraints for semi-supervised clustering, the oracle is queried about a pair of points and answers whether they should be in the same or different clusters, i.e. whether they are must- or cannot-linked.

There are two main approaches to active learning of pairwise constraints. The first one makes all queries prior to the clustering in order to obtain a set of the most informative constraints. The other one queries the oracle iteratively based on the previous clustering solution and current constraints in order to further improve it in each iteration. The latter approach has been shown to outperform the noniterative methods in terms of clustering performance [21]. On the other hand, it is more computationally expensive since it has to run the clustering in each iteration. Hence, one can combine both approaches to obtain a good initial set of constraints and then iteratively improve the clustering upon them. Both approaches will be described in detail in the following sections.

All the active learning schemes described in this chapter are based on a notion of neighborhood. A neighborhood is a subset of data points that are known to belong to the same class, i.e. all pairs of points in a neighborhood are must-linked to each other. Furthermore, points in different neighborhoods are known to belong to different classes, i.e. all pairs of points from different neighborhoods are connected by cannot-link constraints. Note that not all

the constraints have to be provided explicitly by the oracle but rather inferred from the existing constraints thanks to the transitive and entailment property of ML and CL constraints. Note also, that the notion of the neighborhood differs from the one used by the semi-supervised clustering methods where more than one neighborhood can correspond to a single class. Because of this, all the described schemes return a set of neighborhoods that can be used to infer the constraints instead of returning sets of ML and CL constraints.

Finally, all the methods in this chapter do not make any assumptions about the underlying semi-supervised clustering method so they can be used alongside any algorithm presented in Chapter 2.

3.1 Active learning prior to clustering

In this section, we will describe two methods for active learning prior to clustering. First, we describe Explore & Consolidate framework for active learning and then Min-max method that builds on top of the same framework and further improves it.

3.1.1 Explore & Consolidate method

A framework for active learning of pairwise constraints prior to the clustering was first proposed by Base et al. in [14]. The active learning schema is based on the farthest first traversal method and consists of two phases, explore and consolidate, and hence we refer to it as Explore & Consolidate.

The first, exploration phase tries to obtain K disjoint nonempty neighborhoods corresponding to the underlying classes. To accomplish that, farthest first traversal method is used. At each step, the farthest point from all current neighborhoods (a distance of a point from a neighborhood is calculated as the minimum distance of the point and all points in the neighborhood) is queried against a random point of each neighborhood. If it is must-linked to any of the neighboords it is added to that neighborhood. Otherwise, a new neighborhood containing that point is created. Assuming that the point is more likely to belong to a closer neighborhood, the neighborhoods are queried in the order of ascending distances in order to minimize the number of queries needed to find out the neighborhood membership. The result of this phase is called a skeleton.

In the second phase, called consolidation phase, the skeleton is iteratively expanded by randomly chosen points. In each step, it selects a random point and adds it to its corresponding neighborhood by querying all neighborhoods from the closest one to the farthest one. At most $K - 1$ queries are required because if it does not get a must-link response in $K - 1$ queries it must inevitably belong to the remaining neighborhood. This phase lasts until a maximum number of queries is exceeded.

3.1.2 Min-max method

In [22], an enhancement of the Explore & Consolidate method was proposed. Specifically, instead of choosing points in the consolidate phase randomly, they propose a heuristic to choose the most informative point to query at each step. The exploration phase stays unchanged.

In the consolidation phase they are using Min-max heuristic in order to query the points that it is least certain about which neighborhood they belong to. They assume that the point uncertainty is inversely proportional to the maximum similarity of the point to all points in the skeleton X_s. This makes intuitive sense since the less similar the point is to any point in the skeleton the less certain we are about which neighborhood it belongs to. Putting it all together, we can choose the least certain point as the point with the minimal similarity to the skeleton (maximum similarity to any point in the skeleton). Hence, they call the method Min-max. Finally, after the point is selected, the consolidate phase continues the same way as in Explore & Consolidate, i.e. it is queried against all neighborhoods and included in the neighborhood that it is must-linked to.

Formally, the most uncertain point is defined as

$$x^* = \arg\min_{x_i} \max_{x_j \in X_s} s(x_i, x_j).$$

As the similarity measure authors propose using the Gaussian kernel, hence

$$s(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}},$$

where σ is the kernel width parameter.

3.2 Iterative active learning

In this section, we describe a state of the art iterative active learning scheme that selects the most informative points to query based on the neighborhood membership uncertainty and the expected number of queries needed to find it out.

3.2.1 Normalized point-based uncertainty method

Normalized point-based uncertainty (NPU) method [21] is an iterative algorithm that in each step builds upon a previous set of constraints and current clustering and adds new constraints in order to improve the clustering as much as possible. In order to query the most informative point at each iteration it builds on top of the concept of point-based uncertainty. The uncertainty is defined in terms of probabilities of the point belonging to the known neighborhoods which are estimated using a novel approach utilizing random forests.

In general, the higher the uncertainty of a point the more queries are needed to find out its neighborhood membership which leads to a trade-off between the amount of acquired information and number of queries required. Hence, the amount of uncertainty is normalized by the expected number of queries to obtain the point that would provide the most information per query.

The algorithm starts by initializing the neighborhoods by selecting a random point and making it a new neighborhood. Then, in each iteration, it performs a semi-supervised clustering using the constraints inferred from the current neighborhoods. After that, it selects the most informative point to query given the current clustering, which will be described in the following paragraph. As in the other methods, the point is then queried against all neighborhoods and either added to an existing neighborhood that it belongs to or used to initialize a new neighborhood. In order to minimize the number of queries needed, the neighborhoods are queried in the decreasing order of the probability of the point belonging to a given neighborhood.

The most important part of the algorithm is selecting the most informative point. The idea is based on the observation that if we can predict with high certainty to which neighborhood the instance belongs to, based on the current clustering, then querying about that instance will not lead to much additional / useful information. In order to measure the uncertainty, they first learn a similarity measure using supervised learning. Then, they calculate probabilities of each point belonging to each neighborhood and use it to calculate the uncertainty of each point. Finally, they normalize the uncertainty by the expected number of queries to find out the neighborhood membership. Each step is described in detail in the subsequent paragraphs.

Supervised learning of similarities In order to estimate similarities between each pair of points, they leverage random forests. First, they train a random forest to predict the current cluster assignments. Then, the similarity of each pair of points is calculated by predicting the cluster assignment of both points and counting how many times do both points end up in the same leaf of each tree and divide it by the total number of trees in the forest. That gives them a number between 0 and 1, where 0 means no similarity and 1 maximum similarity.

Estimating neighborhood membership probability The probability of point x belonging to a neighborhood N_i is assumed to be proportional to the average similarity of the point to all points in the neighborhood. Formally, it is calculated as

$$p(x \in N_i) = \frac{\frac{1}{|N_i|} \sum_{x' \in N_i} s(x, x')}{\sum_{p=1}^{l} \frac{1}{|N_p|} \sum_{x' \in N_p} s(x, x')},$$

where $|N_i|$ is the number of points in the neighborhood N_i and l is the total number of neighborhoods.

Calculating uncertainty The uncertainty of each point is calculated as the entropy of the neighborhood membership probabilities which is defined as

$$H(N|x) = -\sum_{i=1}^{l} p(x \in N_i) \log_2 p(x \in N_i).$$

Normalizing uncertainty Finally, the uncertainty is normalized by the expected number of queries needed to find out the neighborhood membership. We can look at the expected number of queries as the cost of obtaining information about the point. Let $q(x)$ denote a random variable of the number of queries needed to obtain the neighborhood membership. Assuming that we query the neighborhoods in order of their decreasing probability, then $p(q(x) = i) = p(x \in N_i)$ and hence expected number of queries equals to

$$\mathbb{E}[q(x)] = \sum_{i=1}^{l} i \times p(x \in N_i).$$

Putting all this together, we can choose the most informative point as

$$x^* = \arg\max_{x \in N} \frac{H(N|x)}{\mathbb{E}[q(x)]},$$

where $N = \cup_i^l N_i$.

Chapter 4

Implementation

In this chapter, we describe our active semi-supervised clustering library that implements all methods reviewed in this thesis. First, we start with a motivation for implementing the methods ourselves, review existing implementations and finally describe the programming interface and structure of the library.

4.1 Motivation

There are two main reasons for implementing all the methods ourselves. First, to the best of our knowledge, there do not exist any available open-source implementations of many of the reviewed methods. Moreover, some of the existing implementations are no longer maintained or have functional issues. Second, one of our main requirements was for the implementations to have a uniform application programming interface (API) which would allow easy integration with other machine learning libraries and allow us to run all the algorithms in a single environment.

4.2 Design and implementation

One of the main requirements for the library was to be able to integrate it with an existing machine learning library so that we do not have to implement basic functionality, such as cross-validation or evaluation measures, anew. The machine learning library of our choice is scikit-learn [23], which is probably the most popular machine learning library for Python. It includes a vast amount of supervised and unsupervised learning algorithms, tools for data preprocessing, model selection and evaluation, and a framework for building machine learning pipelines. Implementing their API allows us to make use of all its functionality.

scikit-learn defines a uniform basic API that all objects should conform to. There are three main interfaces: an *estimator* interface for building and fitting

```
clusterer = MPCKMeans(n_clusters=3)
clusterer.fit(X, ml=[(10, 20), (30, 40)], cl=[(40, 80)])
clusterer.labels_
```

Listing 4.1: Semi-supervised clustering example

```
oracle = Oracle(labels=y, max_queries_cnt=20)
active_learner = NPU(clusterer=clusterer, n_clusters=3)
active_learner.fit(X, oracle=oracle)
active_learner.pairwise_constraints_
```

Listing 4.2: Active learning of pairwise constraints example

models, a *predictor* interface for making predictions, a *transformer* interface for transforming data.

All supervised and unsupervised learning algorithms should implement the estimator interface. The estimator interface requires a `fit` method that accepts training data and optional data-dependent parameters and does the model training. For clustering algorithms, scikit-learn defines a mixin that requires that the cluster assignments are stored in a `labels_`[1] instance variable.

Hence, all implemented semi-supervised clustering algorithms implement the `fit` method that accepts the data and background knowledge in the form of either paritally labeled data or pairwise constraints, does the clustering and stores the assignments to the `labels_` instance variable. The active learning algorithms implement only the estimator interface and expect an oracle object as a parameter. The oracle object implements a single `query` method that given two objects returns whether the objects should be in the same cluster or not. An example usage of the semi-supervised clustering and active learning methods can be seen in Listing 4.1 and 4.2, respectively.

4.3 Implemented methods

All implemented methods are organized into packages based on their functionality. There are two main packages. The first package contains methods for semi-supervised clustering and is itself divided into two subpackages. The first subpackage contains methods utilizing partially labeled data and the other one methods utilizing pairwise constraints. The second package contains all methods for active learning. Beside these two packages, there is a package with helper functions that are used by semi-supervised clustering and active learning methods. Below is a comprehensive list of all packages and the classes and functions that they include.

[1] By convention, variables estimated from data must end with trailing underscore

- `semi_supervised_clustering.partially_labeled_data`
  - `SeededKMeans`
  - `ConstrainedKMeans`
- `semi_supervised_clustering.pairwise_constraints`
  - `COPKMeans`
  - `PCKMeans`
  - `MKMeans`
  - `MPCKMeans`
- `active_learning`
  - `Random`
  - `ExploreConsolidate`
  - `MinMax`
  - `NPU`
- `helpers`
  - `farthest_first_traversal`

4.4 Dependencies

Our library depends on the following packages:

- NumPy [24] for working with N-dimensional arrays, linear algebra operations and random number capabilities.
- scikit-learn for their implementation of random forest classifier.
- metric-learn[2] for their implementation of the distance metric learning.

4.5 Existing implementations

In this section, we review existing implementations in Python, R and Java programming languages:

- conclust [25], which is pairwise constraints clustering library written in R, that includes an implementations of the COP-KMeans and MPCK-Means algorithms.

[2] `https://github.com/metric-learn/metric-learn`

- WekaUT package[3] is a modified version of Weka [26] (machine learning library for Java) that includes an implementation of the MPCK-Means algorithm.
- copkmeans [27] package that implements COP-KMeans in Python.

[3] https://www.cs.utexas.edu/users/ml/risc/code/

Chapter 5

Experiments

In this chapter, we experimentally compare studied semi-supervised clustering and active learning methods on data sets of various sizes and dimensionalities. First, we evaluate how well can the semi-supervised clustering methods utilize provided background knowledge and compare the performance to an unsupervised clustering method. Second, we evaluate methods for active learning of pairwise constraints to see whether they can lead to a better clustering with fewer queries compared to a random selection of the constraints.

5.1 Design of experiments

In this section, we describe the design of experiments. First, we describe the method for evaluating the algorithms and how is the background knowledge being generated. Then, we describe the chosen evaluation measure and the baseline method that we use to compare the reviewed methods with.

5.1.1 Learning curves

Each algorithm is experimentally evaluated using learning curves. A learning curve shows a dependence of an evaluation measure on the amount of background knowledge provided to the algorithm. Intuitively, the more background knowledge is provided the better the clustering results should be. In case of methods using partially labeled data, the points on the learning curve correspond to the number of seed points. In case of methods using instance-level constraints, the points of the learning curve correspond to the number of instance-level constraints.

The learning curves are generated using stratified 5-fold cross validation. The clustering algorithms are run on the whole data set but the background knowledge is generated only from the training folds and the external evaluation measure is calculated only on the test fold to see if the algorithms can

generalize on the new data that was not used to generate the background knowledge. Results from all testing folds are averaged.

Finally, since the algorithms themselves and the sampling of background knowledge is random, we run each experiment 20 times and average the results.

5.1.2 Generating the background knowledge

The background knowledge used for evaluating the semi-supervised methods is generated randomly from the testing data sets. In case of partially labeled data, a given fraction of labeled points is sampled using stratified sampling to keep the same ratio of the labels as in the original data set. In case of instance-level constraints, pairs of points are randomly sampled from the data set. If they belong to the same class then the ML constraint is added. If they belong to a different class then the CL constraint is added.

5.1.3 Evaluation measure

In order to compare the clustering solution to the ground truth (known classes) we use the Adjusted Rand index, which is a standard external measure, as described in Section 1.3.2.

5.1.4 Baseline method

Besides comparing the methods to each other, we also compare all methods to a baseline unsupervised clustering method. We chose K-Means++ since it is a state of the art partitional clustering method.

5.2 Data sets

For the experiments, we chose 6 data sets from the UCI Machine Learning Repository [28]. All data sets have only numerical features (which is an assumption of K-means algorithm) and known classes of each instance, i.e. they are classification data sets. In order to evaluate the algorithms on wide range of data sets, we chose data sets of various complexities in terms of size, dimensionality and number of classes. Also, all the selected data sets are commonly used for benchmarking semi-supervised methods in literature.

Before doing the experiments, all data set features were standardized by subtracting the mean and scaling to unit variance so that all features have the same range. This is important because K-means and all methods based on K-means that do not learn its own metric would give higher weight (importance) to the features with larger range.

For the Digits data set, only digits 3, 8, 9 were chosen since they are visually similar and that makes it a more difficult clustering problem.

Data set	# of instances	# of attributes	# of classes
Breast Cancer (UCI)	569	30	2
Digits-389 (UCI)	537	64	10
Glass (UCI)	214	9	6
Ionosphere (UCI)	351	34	2
Iris (UCI)	150	4	3
Wine (UCI)	178	13	3

Table 5.1: Data sets

All data sets are listed in Table 5.1 along with number of instances, number of attributes and number of classes.

5.3 Experiments implementation

All experiments were implemented in the Python[4] programming language, specifically in Python 3.6.5. Python is a high-level open-source programming language that is commonly used for data mining and machine learning thanks to its ease of use and a wide range of scientific libraries.

For prototyping the experiments, we used Jupyter Notebook[5], which is an open-source web application for interactive computing.

For handling data and evaluating and visualizing the results of the experiments, we used the following open-source libraries:

- NumPy [24] — a package for scientific computing.
- Pandas [29] — a package for loading and analyzing data.
- scikit-learn [23] — a popular machine learning library. We use its implementation of cross-validation, evaluation measures and others.
- Matplotlib [30] — a data visualization library. We use it for visualizing the results of the experiments and plotting all graphs in this chapter.

5.4 Results analysis

In this section, we describe design and results of the experiments we conducted in order to evaluate semi-supervised clustering and active learning methods reviewed in this thesis.

[4] `https://www.python.org`
[5] `http://jupyter.org`

5.4.1 Experiments with semi-supervised clustering methods using partially labeled data

In this experiment, in order to evaluate how increasing amount of labeled data points influences quality of clustering and to see whether these methods provide better clustering than the unsupervised one, we generated learning curves for each algorithm where on the x-axis there is a fraction of training data set that was used as labeled data points given to the algorithm and on the y-axis there is the value of the Adjusted Rand index. We experimented with the following fractions:

$$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}.$$

Note that the fraction is relative to the size of the training set used for sampling the labeled data, not to the whole set used for the clustering. So for example, if the value of the fraction is 1 then we use $^4/_5$ (in case of the 5-fold cross validation) of the whole data set as the background knowledge.

We experimented with both algorithms described in Section 2.2, specifically with Seeded-KMeans and Constrainted-KMeans.

As we can see from Figure 5.1, both algorithms outperform the baseline unsupervised clustering on all testing data sets given sufficient amount of labeled data.

In case of the Digits-389, Iris and Wine data sets, the learning curve is very steep in the beginning and then slows down, which means that even a small fraction (0.1–0.2) of the data set is sufficient to significantly improve the results and providing additional background knowledge gives only minor improvements (relative to the improvements at the beginning). On the other hand, in case of the Breast Cancer, Glass and Ionosphere data sets, providing additional background knowledge improves the clustering at a relatively steady rate.

We can notice that on the Iris and Wine data set, K-Means++ is significantly better than the semi-supervised methods given no or very little background knowledge. This is probably due to the fact that the K-Means++ initialization heuristic works significantly better than the random initialization utilized by the semi-supervised methods. On the other hand, random initialization seems to work better on the Glass and Digits-389 data sets.

When comparing Seeded-KMeans and Constrained-KMeans, we can see that after a certain point on the learning curve, the clustering quality of the Seeded-KMeans stops increasing while the quality of the Constrained-KMeans keeps increasing. This is not surprising since the Seeded-KMeans is using the labeled data only for the initialization but does not take them into further consideration, unlike the Constrained-KMeans that takes the labels into account in the assignment step and prevents them from changing cluster assignments.

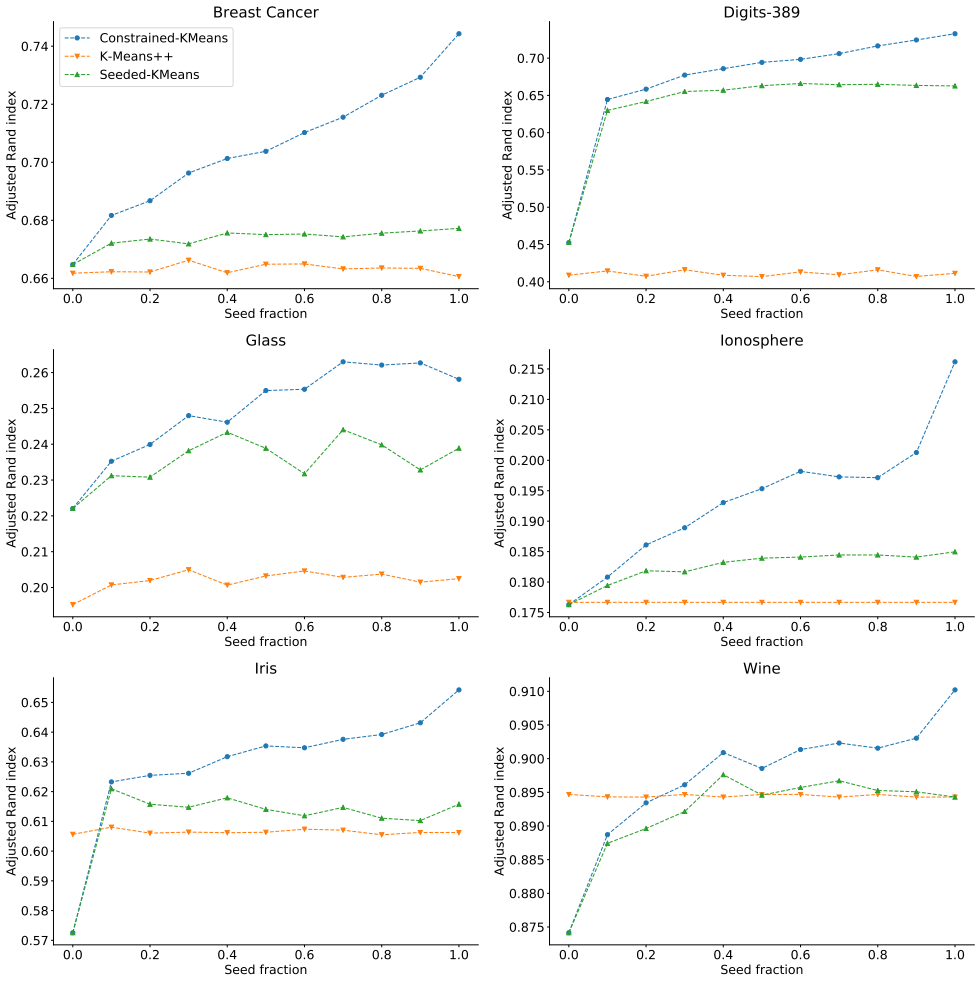

Figure 5.1: Results of semi-supervised clustering methods using partially labeled data

5.4.2 Experiments with semi-supervised methods using pairwise constraints

The main goal of this experiment is to empirically evaluate how increasing amount of background knowledge in the form of the pairwise constraints influences clustering quality of the semi-supervised methods. We generated learning curves of each method where on the x-axis is the number of randomly chosen pairwise constraints provided to the algorithm and on the y-axis is the value of the Adjusted Rand index. In order to see how many constraints would have to be provided by a domain expert to significantly improve the clustering, the number of constraints refers to the number of explicitly provided constraints, as opposed to the actual number of constraints that were used by the algorithm thanks to the transitive and entailment properties of

the ML and CL constraints, respectively. We experimented with the following values:

$$\{0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}.$$

Methods

In the experiment, we have evaluated all methods described in Section 2.3, specifically:

- K-Means++, which is an unsupervised clustering algorithm and we used it as a baseline. We chose K-Means++ over the random K-means since it is the state of the art clustering algorithm and has been shown to outperform the standard K-means [10].
- COP-KMeans, which does hard constraint enforcement and does not utilize the constraints for the centroid initialization. We consider this algorithm as the baseline method of semi-supervised clustering algorithms.
- PCK-Means, which does soft constraint enforcement by penalization and utilizes the constraints for the centroid initialization.
- MK-Means, which utilizes metric learning prior to clustering and utilizes only the constraints to learn the metric (unlike MPCK-Means which uses all data to learn the metric). We use the variant that learns only a diagonal weight matrix (effectively doing feature weighting) since learning the full matrix is computationally infeasible.
- MPCK-Means, which combines soft constraint enforcement and iterative metric learning. We chose the variant that learns a single diagonal weight matrix, same as in MK-Means, since learning multiple / full matrices is computationally infeasible and moreover it is the variant that the authors of the algorithm used to evaluate and compare to other methods.

Parameters

PCK-Means and MPCK-Means depend on the constraint weight (w) parameter that is used for penalizing constraint violations in order to do soft constraint enforcement. As such, it considerably influences behaviour of the algorithms. Low values of w forces them to use the constraints mostly only for the centroid initialization since it does not almost penalize the constraints and hence does not enforce them. On the other hand, high values of w makes them behave more like COP-KMeans in that it strictly penalizes constraint violations and hence enforces them. Hence, we experimented with various values $(0.001, 0.01, 0.1, 1, 10)$ of the parameter on each dataset and in the subsequent

Data set	Constraint weight w	
	PCK-Means	MPCK-Means
Breast Cancer	10	0.01
Digits-389	10	0.1
Glass	0.1	0.001
Ionosphere	1	1
Iris	1	0.01
Wine	1	10

Table 5.2: Constraint weights w used for each data set and algorithm

experiments we use the best values for each algorithm and data set. The summary of the best parameters is shown in Table 5.2 and the full learning curves can be seen in Figure B.1 and B.2.

Results

As we can see from Figure 5.2, performance of each algorithm differs based on the data set. We analyse the results on each data set in the subsequent paragraphs.

Breast Cancer All semi-supervised algorithms outperform unsupervised clustering more and more with increasing number of constraints. MPCK-Means also significantly outperforms other semi-supervised clustering algorithms, which suggests that combining metric learning with soft constraint enforcement is beneficial for the clustering quality on this data set. Moreover, we can notice that MPCK-Means performs better even without any background knowledge, which is due to the fact that MPCK-Means does metric learning even without any constraints (it weights each feature by the inverse of its variation, similarly to Mahalanobis distance).

Digits-389 All semi-supervised clustering algorithms significantly outperform unsupervised clustering. They all have similar results with MPCK-Means being the best one.

Glass All the semi-supervised clustering methods perform better than the unsupervised clustering even without any background knowledge which suggests that K-Means++ heuristic works worse than the random initialization utilized by the other algorithms. From the semi-supervised clustering algorithms, both methods that do metric learning perform better, which suggests that feature weighting is beneficial on this data set. In the early stage of the learning curve, PCK-Means does not improve at all and at that point works worse than COP-KMeans but it starts improving after 300 constraints have been provided and gradually starts outperforming the COP-KMeans.

Ionosphere We can see that the methods that enforce constraints eventually improve the clustering over the unsupervised one. However, in the beginning with only few constraints the clustering does not bring any improvements and in case of COP-KMeans there is a significant dip in the clustering quality which might suggest that due to the complexity of the data set, the initial set of constraints might be unreliable and actually worsen the clustering. Regarding the MK-Means, which does not enforce the constraints and learns the metric solely from the constraints prior to the clustering and thus not considering the clustering performance, leads to learning a metric that results in worse clustering.

Iris On the Iris data set, metric learning (feature weighting) seems to be highly beneficial (similarly to the Glass data set) since methods utilizing it significantly outperform other methods. Also, we can notice a very steep learning curve that flats out very fast which means that very few constraints are necessary for significant improvement and that additional constraints do not add much benefit.

Wine As opposed to the Glass and Iris data set, on the Wine data set, learning a metric leads to worse clustering quality in terms of the agreement with the ground truth. The inability to learn a useful metric might be caused by the relatively high dimensionality compared to the data set size or due to correlated features. Without learning any metric, the semi-supervised clustering methods work better than the unsupervised clustering.

5.4.3 Experiments with methods for active learning of pairwise constraints

The goal of this experiment is to empirically evaluate whether using an active learning of pariwise constraints leads to a better clustering quality than if we chose the pairwise constraints randomly. Also, we want to see how do the individual reviewed methods compare to each other on each data set. In order to do that, we generated learning curves for each method where on the x-axis is the maximum number of queries that the method is allowed to ask the oracle and on the y-axis is the value of the Adjusted Rand index. We experimented with the following numbers of maximum allowed queries:

$$\{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}.$$

In the experiment, we evaluated all methods reviewed in Chapter 3. Specifically, we evaluated the following methods:

- Random method, which selects random pairwise constraints, as a baseline method. It is a common baseline method for active learning studies.

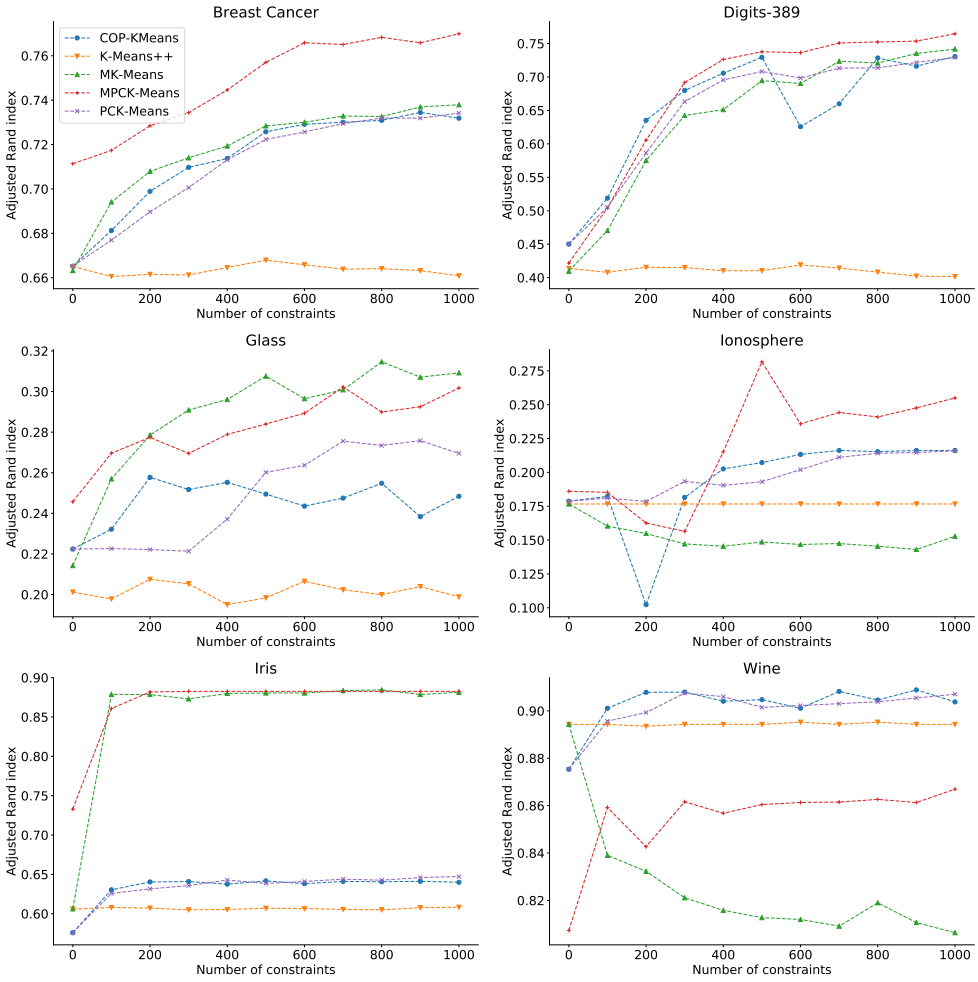

Figure 5.2: Results of semi-supervised clustering methods using pairwise constraints

- Explore & Consolidate method, which is a baseline of the active learning schemes.

- Min-Max method, which is an improved version of the Explore & Consolidate method. As the kernel width parameter of its similarity measure, we used the 20^{th} percentile of the distribution of pairwise Euclidean distances of all pairs of points as suggested by the authors of the algorithm.

- NPU, which is an iterative active learning method. NPU depends on a random forest classifier to estimate a similarity measure. We used a scikit-learn implementation of the random forest with 50 trees. We kept the default values of other parameters.

All the active learning methods are independent of the underlying semi-supervised clustering algorithm. For the experiment, we used the PCK-Means algorithm, since it performed consistently well on all data sets, with the same parameters as in Section 5.4.2.

As we can see from the results in Figure 5.3, all the active learning algorithms outperform the baseline method on all data sets, except for one case that we will look into in the following paragraphs.

We can see that on most of the data sets, the iterative active learning method gives better or similar results as the noniterative methods. However, on the Wine data set it works worse than the noniterative methods and on the Ionosphere data set it works even worse than the random choice at some points of the learning curve. In case of the Ionosphere data set, we can notice that both Min-max and NPU, which depend on a similarity measure to query the most informative points, work worse than Explore & Consolidate, which queries the points randomly. This might suggest that the estimated similarities are not reliable on this data set and that providing constraints based on that measure does not bring any benefit to the clustering algorithm. In both cases, however, experimenting with the parameters of the similarity measures might lead to improved results.

Similarly, we can see that the Min-max method, which is an improved version of the Explore & Consolidate method, works indeed better than the Explore & Consolidate on most data sets.

Finally, on the Digits-389 data set, we can notice a dip in the performance of the methods that depend on the explore phase to initialize neighborhoods. That might suggest that the initial neighborhoods obtained from the explore phase are unreliable and the consolidate phase is necessary to get good results. Unfortunately, at the beginning of the learning curve, where only a few queries are allowed, it uses all the queries for the exploration phase and it will not get to the consolidate phase. Even worse, it might happen that it will not be even able to finish the exploration phase before it exceeds the limit, leaving it with incomplete neighborhoods. The explore phase is utilizing the farthest first traversal method, which is sensitive to outliers. Consequently, the unreliability of the explore phase might be caused by outliers in the data set.

5.4.4 Summary

Results of the experiments show that both semi-supervised clustering and active learning considerably outperform unsupervised clustering and random selection of pairwise constraints, respectively, in terms of clustering quality. However, none of the reviewed methods was superior to the other methods on all testing data sets, which makes choosing the right method difficult.

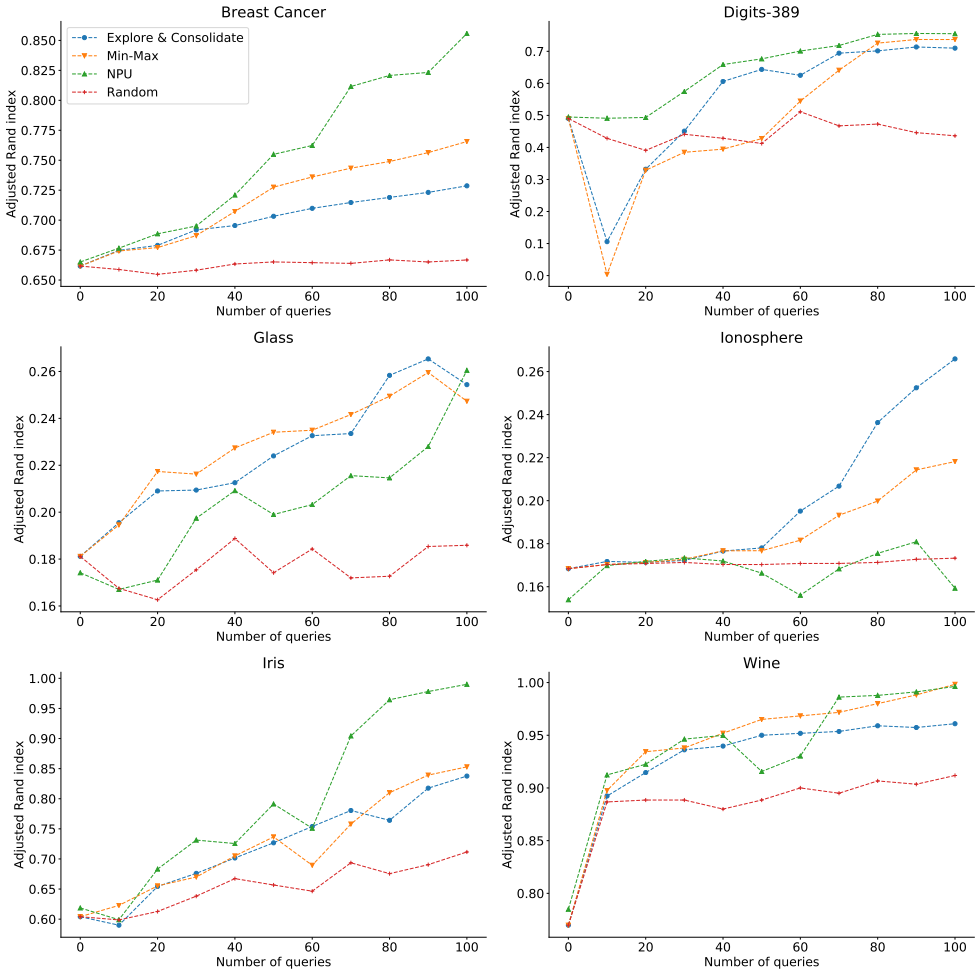

Figure 5.3: Results of active learning methods

Chapter 6

Future work

In this chapter, we propose several future directions for extending/improving the reviewed methods.

Based on the empirical evaluations of the reviewed methods, we can conclude that there is no single method that would be superior in all circumstances but rather that each approach is beneficial in different situations. To address this problem, one could consider utilizing ensemble learning. Specifically, a technique known as consensus clustering, also known as aggregation of clusterings, whose goal is finding a single (consensus) clustering based on multiple different clusterings of the same data set.

Another extension of the current methods could be automated estimation of the number of clusters. Background knowledge provided in the form of pairwise constraints could be used to infer the number of clusters implicitly. A simple approach to do that could be taking the number of neighborhoods inferred from the constraints and use it to estimate the number of clusters. This approach would not work in case of very few constraints, which leads to many sparse neighborhoods, but our assumption is that the number of neighborhoods converges to the true number of clusters with increasing number of constraints.

Regarding the active learning of pairwise constraints, the empirical evaluations show that the iterative clustering methods, in our case the NPU method, in many cases outperform active learning prior to clustering. However, this approach is computationally expensive since the clustering must be rerun after each query. This shortcoming could be solved by querying the oracle in batches. A simple approach would be querying first n most informative points. However, as the authors of the algorithm note, this could lead to redundant queries. Hence, it would be interesting to investigate ways of obtaining a more diverse set of the most informative points.

Conclusion

Clustering is a challenging problem since there are often multiple possible clusterings and as such it is strongly problem dependent. Hence, it can benefit from semi-supervised learning in order to make use of prior knowledge from a domain expert. However, obtaining background knowledge from the domain expert is often time consuming and expensive. This motivates use of active learning in order to minimize the number of queries. In this thesis, we review state of the art methods for active semi-supervised clustering.

First, we review types of background knowledge for clustering and methods utilizing them. We focus on methods that utilize pairwise constraints, which provide an information about whether two points should be in the same cluster or in different ones. There are three main approaches to integrating the constraints. The first approach modifies a clustering algorithm itself in order to satisfy all or as many constraints as possible. The second approach learns a distance measure so that the points that should be in the same cluster are close to each other and the points that should be in different clusters are far apart. The third approach combines the two approaches. We reviewed state of the art algorithms from each approach.

Second, we reviewed approaches to active learning of pairwise constraints. There are two main approaches to learning the constraints. The first approach learns all the constraints before the clustering and the other iteratively queries the domain expert in order to improve the clustering in each iteration. We reviewed several active learning schemes that can be integrated with all reviewed semi-supervised clustering algorithms.

In the practical part, we implemented and experimentally evaluated all reviewed methods on various data sets. Results of the experiments show that both semi-supervised clustering and active learning considerably outperform unsupervised clustering and random selection of pairwise constraints, respectively, in terms of clustering quality. However, none of the reviewed methods was superior to the other methods on all testing data sets.

Finally, we discussed three directions for further improvements of the re-

viewed methods. The most interesting direction could be exploring possibilities of utilizing pairwise constraints for automated estimation of number of clusters in a data set.

Bibliography

[1] Caliński, T.; Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, volume 3, no. 1, 1974: pp. 1–27.

[2] Davies, D. L.; Bouldin, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, , no. 2, 1979: pp. 224–227.

[3] Dunn, J. C. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, volume 4, no. 1, 1974: pp. 95–104.

[4] Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, volume 66, no. 336, 1971: pp. 846–850.

[5] Hubert, L.; Arabie, P. Comparing partitions. *Journal of classification*, volume 2, no. 1, 1985: pp. 193–218.

[6] Berkhin, P. A survey of clustering data mining techniques. In *Grouping multidimensional data*, Springer, 2006, pp. 25–71.

[7] Wu, X.; Kumar, V.; et al. Top 10 algorithms in data mining. *Knowledge and information systems*, volume 14, no. 1, 2008: pp. 1–37.

[8] Lloyd, S. Least squares quantization in PCM. *IEEE transactions on information theory*, volume 28, no. 2, 1982: pp. 129–137.

[9] MacQueen, J.; et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, Oakland, CA, USA, 1967, pp. 281–297.

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

[11] Davidson, I.; Ravi, S. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining*, SIAM, 2005, pp. 138–149.

[12] Wagstaff, K.; Cardie, C.; et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, 2001, pp. 577–584.

[13] Basu, S.; Banerjee, A.; et al. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*, Citeseer, 2002.

[14] Basu, S.; Banerjee, A.; et al. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, SIAM, 2004, pp. 333–344.

[15] Xing, E. P.; Jordan, M. I.; et al. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, 2003, pp. 521–528.

[16] Rockafellar, R. T. *Convex analysis*. Princeton university press, 2015.

[17] Bilenko, M.; Basu, S.; et al. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 11.

[18] Saul, L. K.; Roweis, S. T. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, volume 4, no. Jun, 2003: pp. 119–155.

[19] Davidson, I.; Wagstaff, K. L.; et al. Measuring constraint-set utility for partitional clustering algorithms. In *European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2006, pp. 115–126.

[20] Wagstaff, K. L. Value, cost, and sharing: Open issues in constrained clustering. In *International workshop on knowledge discovery in inductive databases*, Springer, 2006, pp. 1–10.

[21] Xiong, S.; Azimi, J.; et al. Active learning of constraints for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, volume 26, no. 1, 2014: pp. 43–54.

[22] Mallapragada, P. K.; Jin, R.; et al. Active query selection for semi-supervised clustering. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, IEEE, 2008, pp. 1–4.

[23] Pedregosa, F.; Varoquaux, G.; et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, volume 12, 2011: pp. 2825–2830.

[24] Oliphant, T. E. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[25] Hiep, T. K.; Duc, N. M. *conclust: Pairwise Constraints Clustering*. 2016, r package version 1.1. Available from: `https://CRAN.R-project.org/package=conclust`

[26] Frank, E.; Hall, M.; et al. The WEKA workbench. *Data mining: Practical machine learning tools and techniques. Burlington: Morgan Kaufmann*, 2016.

[27] Babaki, B. COP-Kmeans version 1.5. July 2017, doi:10.5281/zenodo.831850. Available from: `https://doi.org/10.5281/zenodo.831850`

[28] Dheeru, D.; Karra Taniskidou, E. UCI Machine Learning Repository. 2017. Available from: `http://archive.ics.uci.edu/ml`

[29] McKinney, W.; et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, Austin, TX, 2010, pp. 51–56.

[30] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in science & engineering*, volume 9, no. 3, 2007: pp. 90–95.

Appendix A

Acronyms

API Application Programming Interface

CL Cannot-link

ML Must-link

NPU Normalized point-based uncertainty

WCSS Within-cluster sum of squares

Appendix B

Results

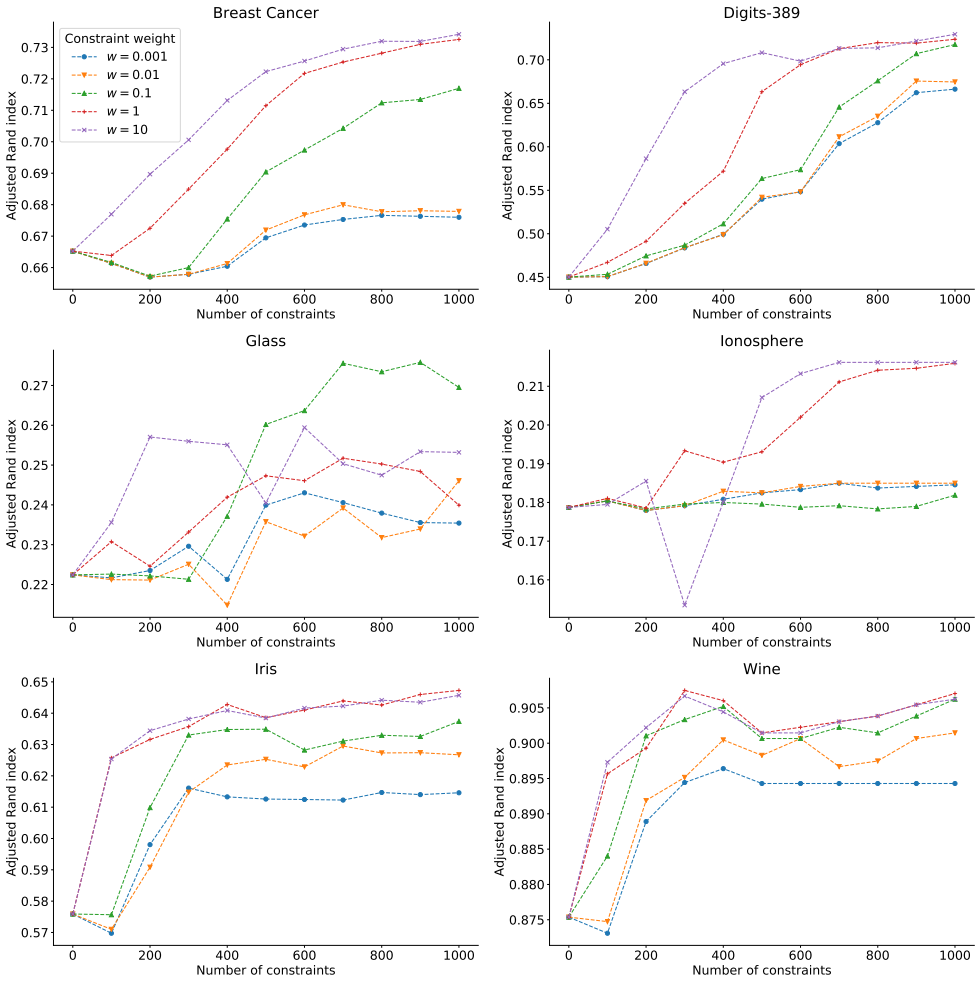

Figure B.1: Influence of the constraint weight parameter on PCK-Means

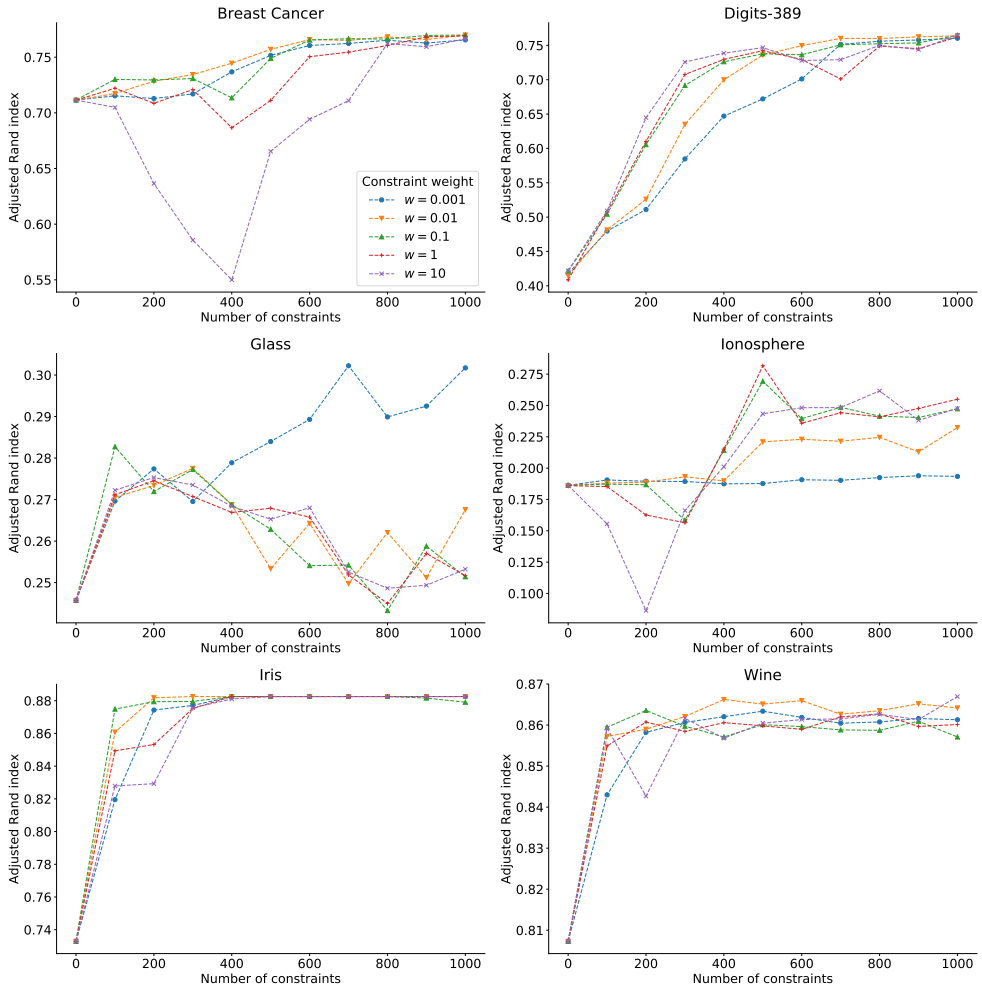

Figure B.2: Influence of the constraint weight parameter on MPCK-Means

Appendix C

Contents of enclosed CD

- `README.md` ..... the file with CD contents description
- `src` ..... the directory of source codes
  - `implementation` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format