HEBREW UNIVERSITY

AMIRIM PROJECT

December 13, 2010

A convex relaxation to solve the k-medoids problem

Author:
Yaron Margalit

Supervisor:
Dr. Shai Shalev-Shwartz

Contents

1 Introduction

In this project we consider the problem of clustering. The basic goal of clustering is to classify observed objects into groups (clusters), according to a predefined mathematical measure of similarity. Problems of clustering data are prevalent in science, people seek to collect data in order to uncover the underlying structures, derive from them useful insights or discover new knowledge about them. The amount of data collected may vary from large set groups to a small ones.

Clustering algorithms are widely used to manage, classify and summarise many kinds of data. Clustering problems arise in a wide range of application, including: data mining, statistical data analysis, market research, image segmentation, medical diagnosis, pattern recognition, astronomers cluster stars, information retrieval such as document clustering, web access and more.

In general, there is no clear ground truth for how exactly clustering should be. In many of the clustering applications, there is an unknown target or desired grouping. The real goal in these applications is to minimize the clustering error with respect to the target clustering defined.

Machine learning techniques have been widely used in many practical clustering applications. Practical algorithms include: Hierarchical clustering, Linked based clustering, Spectral clustering, Partitioning algorithms and more. In this project we focus on one major clustering algorithms: Partitioning algorithms. The main idea in this approach is to define a distance-based cost function and then design a clustering algorithm that (approximately) optimizes this cost function. Meaning, constructing a partition of the data set into a set of k clusters ("cluster membership") so that the points within a cluster are relatively close using a specific measure. Hopefully, this will result in the minimal cost that is close to the target clustering. Under this paradigm, the clustering problem is turned into an optimization problem.

Intuitively, a clustering algorithm performs well if points that are similar among themselves are lumped together in the same cluster and points that are dissimilar are assigned to different clusters. This leads to the question of, what exactly is an optimal clustering? In this project, we assume that approximately optimizing the objective target function produce a clustering with a low error. Of course, there are other defined measures of the quality of a clustering. For example: [12, 31].

In this project, we study the classic partitional clustering k-mean and k-medoids problems. Also we will formulate and study a new optimization problem that we call *convex relaxation of k-medoids*. In general, it is known that partitional clustering problems are NP-hard to obtain a search heuristic that can compute the minimal cost of the problem in polynomial time [10, 13, 14, 26] and even NP-hard to approximate these problems less than a certain factor times the optimal cost [15]. Partitional clustering algorithms, especially K-means and K-medoids works well on spherical data [11, 30, 31].

Given a data set of n points and an integer k that indicates number of requested

centers, the k-means problem asks us to identify exactly k clusters. The goal is to select the k clusters so as to minimize according to predefined distance measure (Particularly, the sum of the squared distances) between each data point and its cluster. The k-medoids problem is similar to the k-means except the latter's requirement that the center cluster to be one of the data points in the cluster. (Remark: *The medoids* is essentially the discrete analog name of a center centroid, that was first called [8], and is also called *the median*.)

This project focuses on developing approximation algorithms for these problems. It presents a *convex-relaxation* method to solve the k-means and k-medoids problems. We developed algorithms that approximate the k-means and k-medoids problem. This was done by balancing between trying to approximate the K-medoids target function and the use of maximum entropy.

In general, the entropy measures the uncertainty of an observer about the outcome of an experiment. Maximizing the entropy may be viewed as trying to adapt the most non-committal assignment of the distribution that obeys certain constraints [17, 18]. The principle of using the maximum entropy answers the idea of choosing a model consistent with all the constraints, and nothing more than that.

In this project, we discuss the advantages of using the maximum entropy as a factor for our approximation algorithms. We present an implementation of the convex relaxation k-medoids algorithm, application uses and study their performance in practice.

Roadmap. The rest of this project is organized as follows. Section 2 reviews related work. We formally define the clustering problem and present our model in Section 3. Section 4 presents the implementation of our model. Usage applications are presented in section 5. Section 6 presents experimental results. We end with some concluding remarks in Section 7.

2 Related work

Clustering is a form of unsupervised learning that has applications in many problem domains. Approaches to clustering based on minimizing a given objective function has been well-studied [1, 2].

The k-means, an efficient heuristic simple algorithm, is a popular technique to approximate the k-means problem [2, 11]. Given a set of n data points and k number of centers in Euclidean distance, the algorithm first randomly select a set of k initial cluster centers. Then, by iterative process we partition the data points into k groups by associating each point to its closest center. After all points have been assigned, the algorithm recalculate new k centers as the average of each partitioned group. The algorithm ends when there have not been any changes in the partitioned groups. Each iteration in the k-means algorithm requires $O(nk)$ time. Since the cost of the output of the successive iterations of k-means is monotonically decreasing, the algorithm converges to a local minimum solution based on its initial start. A theoretical study

on how close factor times to the optimal objective cost is not known. For a full description of the k-means problem and algorithm, see Appendix A.

The initialization procedure in k-means is very critical to the quality of clusters and also to the running time of the algorithm. The Basic k-means algorithm selects them randomly. We note that it is a feasible initial start for the k-means to give any subset of k data points. We make use of this fact to propose our algorithm as an initialization procedure for k-means.

The k-medoids problems were originally studied in the context of resource placement as well as the facility location problem [3, 4, 5, 9]. This can be easily applied to our world context, when we view the cluster centers as resource distribution sites. We focus on the metric case of the problem, that is, when distance base costs satisfy the triangle inequality. Generally, there has been some algorithm techniques used to attack the k-medoids problem, some of them include: partition around medoids, linear programming relaxation.

The Partition Around Medoids (PAM) algorithm [27] is a popular algorithm to approximate the k-medoids problem. It is highly related to the k-means algorithm. PAM attempts to determine k partitions by using the most centrally located point in the cluster instead of the mean cluster as in the k-means algorithm. Similar to the k-means algorithm, PAM starts by selecting a random initial set of medoids, and iteratively replaces each one of the selected medoids by one of the none-selected point in the data set points to be a medoid as long as the sum of dissimilarities of the object is being reduced. PAM works well for small data sets but does not scale well for large data set as the complexity of each change cost $O(k(n-k)^2)$. For a full description of PAM, see Appendix B.

In the linear programming relaxation approach, the solution is based on solving a natural linear programming relaxation of the problem and rounding the fractional solution to a near optimal integer solution. Lin and Vitter [6] gave a polynomial-time algorithm for the metric k-medoids problem that, for any $\epsilon > 0$, finds a solution of cost no more than $2(1+\epsilon)$ times the optimum, while using $(1+1/\epsilon)k$ cluster centers at most. The solution they gave is infeasible: it might open more than k medoids. The first constant factor approximation (and feasible solution) polynomial-time algorithm ($6\frac{2}{3}$-approximation guarantee) LP-based was given by Charikar [1]. Jain and Vazirani [7] give the first nearly linear-time (in the input size) combinatorial algorithms achieving approximation ratio of 6. In our suggested algorithm we make use of a similar idea to the filtering technique of Lin and Vitter [6] for selecting which centers to open.

The *convex relaxation* algorithm, uses a similar approach of the LP relaxation while our problem is translated to a convex relaxation programming that is solved by the interior points method. The solution obtained is only a fractional solution, thus a sort of rounding is needed for a feasible solution.

3 The Model

3.1 Preliminaires

In the k-medoids problem, we are given a set of n data points $X = \{X_1, ..., X_n\}$ from a metric space, an integer bound number k, and a distance cost matrix d, where $d(x, y)$ specifies the distance between each pair of points $x, y \in X$. The k-medoids problem's goal is selecting $T \subset X$, $|T| \leq k$ data points to be the cluster centers. Also, each input point x assigned to the selected center that is the closest to it, so it minimizes the total assignment cost's incurrence. Consequently, the optimization problem we are attempting to solve is:

$$\underset{T \subset X}{\operatorname{argmin}} \sum_{x \in X} d(x, T)$$

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

The k-means problem is the same as k-medoids except the fact that T is a subset of all metric space.

The k-medoids problem can also be stated as the following integer program, where the 0-1 variable y_i, $i \in n$ indicates if the point i is selected as a center centroid, and the 0-1 variables $w_{i,j}$, $i, j \in n$ indicates if point j is assigned to the center centroid i:

$$\underset{w,y}{\text{minimize}} \sum_{i,j \in n} w_{i,j} * x_{i,j} \tag{1}$$

subject to:

$$\sum_{j \in n} w_{i,j} = 1 \tag{2}$$

$$\sum_{i \in n} y_i \leq k \tag{3}$$

$$\forall i, j \in n \; w_{i,j} \leq y_i \tag{4}$$

$$w_{i,j}, y_i \in \{0, 1\} \tag{5}$$

The constraints (2) indicates that a point is selected to one center centroid at most. (3) limits the number of clusters to be at most k. (4) ensures that a point j can be assigned to other point i, only if i is a center centroid.

The integer problem as it stated above is $\mathbb{NP}$-hard. Instead, in order to solve this problem, we focus on developing an approximate algorithm. In the LP-based approximation approach, we consider a linear programming relaxation to the integer program. Where the constraint (5) is replaced with

$$0 \leq w_{i,j}, y_i \leq 1 \tag{6}$$

We note that the problem is turned into a linear program and its solution is not feasible. Meaning, points can be partially attached to several center centroids, and also, there can be existed partial clusters. See Appendix B for a full description and a solution example of a K-medoids LP-based algorithm.

3.2 Convex relaxation and maximum Entropy

The K-means algorithm and the K-medoids algorithms are applicable and used in practice. Though, there are some objections against them.

The K-means method is sensitive to noise and outliers data points since a small number of such data can substantially influence the mean value [19]. The initialization procedure is critical in the K-means algorithm, since it converges to local minimum solutions only. The K-means algorithm does not have any mechanism with which one can get global minimum solution.

We mentioned before that k-means and k-medoids algorithms are suited to find "spherical" clusters, that is, clusters that are roughly ball-shaped. As the dimension of the ball getting higher, the number of points which are located at the center core of the cluster are very limited. Most of the mass will be on the sphere. These gaps are influencing the K-means and K-medoids clustering algorithms, so as it sometimes tries to define centroids by a very rare relation and far from the optimal one. The motivation behind maximum entropy inference is to overcome this problem. We try to find the centers of the clusters even if they don't appear in the original data set.

In order to get advantage of the K-medoids method, but still have the ability of robust clusters and small error rate, we added maximum entropy inference. Our model falls between two extremes. On the one hand, we would like to have an optimal solution for the K-medoids problem. On the other hand, unlike the purley K-medoids models, we require that the center centroids distribution chosen at each step will be the most uniform model subject to a set of constraints. This way, we could get some conclusion on where the center of the cluster is placed even if in the original data, there are no points placed near the center. The idea is to look at a solution obtained by our model. Now, each point might be assigned to a mixture of centroids decided by the k-medoids optimal solution and the entropy. The *expected points* that derived from this assignment, might be closer to the center than any other points that appear in the data set. It takes one good point that placed in the core of the cluster, to have a robust and quality clustering for K-medoids. We make use of this feature, by running our algorithm with new points that we call *expected points*. The concept and definition of *expected points* is further explained in the application section [5].

Let us define the *convex relaxation k-medoids approach* . The problem is the same as LP-based relaxation but we replace the objective (1) with:

$$minimize \sum_{i,j \in n} w_{i,j} * x_{i,j} + \eta H(Y) \tag{7}$$

where H(Y) is the maximum entropy of random variable Y: $\sum_{i \in Y} y_i * log(y_i)$
The problem is turned into a convex optimization problem, since maximum entropy function is convex. The parameter η unveils the trade-off between a minimal cost which is based on the standard k-medoids solution and imposing the fewest additional constraints on the center centroids. Note that, assigning η to be zero return the

objective to be as the original LP-based relaxation problem. We further modify the solution to obtain a feasible solution (integer clusters).

Regarding the distance-based of our algorithm, the input data simply consist of a dissimilarity matrix, without any measurement values. Meaning, the dissimilarities may have been computed form different types of distance-based on the attributes of the objects. However, to compare our results and examine them in a theoretical view, we limited our experiments only to the Euclidean distance.

4 Implementation

In Section 3, we defined the problem we want to optimize. In this section we will focus on how to solve the convex optimization problem (7). The problem includes inequality and equality constraints that can be stated as follows:

minimize (over $w \in \mathbb{R}_{++}^{nxn}, y \in \mathbb{R}_{++}^{n}$):

$$\sum_{i,j} w_{i,j} * d_{i,j} + \eta \sum_{i=1}^{n} y_i \log y_i$$

subject to:

$$Ax = b$$

$$Cx \leq d$$

The equality constraints are:

$$\sum_{j} w_{i,j} = 1$$

The inequality constraints are:

$$\sum_{i} y_i \leq k \ , \ \forall i, j \ w_{i,j} \leq y_j \ , \ 0 \leq w_{i,j} \leq 1 \ , \ 0 \leq y_j \leq 1$$

We notice that all the equality and inequality constraints are linear and the objective is convex.

Over the last years, an extensive research in interior points method for linear and convex optimization has been done. One result of this research is the development of a primal-dual interior point algorithm [22, 23, 24, 25]. Here we present an efficient primal-dual interior point method for solving the convex-relaxation k-medoids problem.

Primal-dual interior-point method:

given x that satisfies $Cx \prec 0$, $\lambda \succ 0$, $\mu = 10$, $\epsilon_{fes} > 0$, $\epsilon > 0$

Repeat :

1. Set $t = \mu * m \setminus \hat{\eta}$
2. Compute primal-dual search direction $\Delta y_{pd} = (\Delta x, \Delta\lambda, \Delta\nu)$
3. Line search and update: Determine step length $s > 0$ and set $y = y + s\Delta y_{pd}$.
4. Stopping criterion. Compute updated r_{pri}, r_{dual} (primal and dual residuals), $\hat{\mu}$, quit if $\| r_{pri} \|_2 \leq \epsilon_{fes}$, $\| r_{dual} \|_2 \leq \epsilon_{fes}$, and $\hat{\eta} \leq \epsilon$

λ presents the inequality constraints, ν presents the equality constraints. m is defined as the number of constraints.

We start the algorithm by obtaining a starting point x_0 that satisfies $Ax_0 = b$ and $Cx \prec 0$ (Meaning, holds all equality and inequality constraints). This can easily be done by picking k randomly medoids from the set points and assign each point to the closest cluster. Notice that we start our algorithm with an infeasible solution. The infeasible start of the primal-dual method will eventually converge as it takes one direct step, and thereafter we will have a strictly feasible x primal-dual solution. Also, we initialize $\lambda_0 = -1/Cx_0$, thus the initial surrogate gap is $\hat{\eta} = 100$.

Determine t gives the accuracy sub-optimal problem as a factor of the surrogate gap.

A Primal-dual search direction is obtained by Newton step. Given the current point $y = (x, \lambda, \nu)$ we try to find the search direction $\Delta y = (\Delta x, \Delta\lambda, \Delta\nu)$ by finding a solution that holds all constraints. This is the main effort of the algorithm and can be done by solving linear equations. In our implementation, we take advantage of the sparse structure to have an efficient solver.

The line search implementation is as a standard backtracking line search: first modified to ensure that updated x satisfies $Cx \prec 0$ and $\lambda \succ 0$ (each time the solution changes as a factor of β). Then we continue searching for s that satisfies:

$$\| r_t(updated_x, updated_\lambda, updated_\nu) \|_2 \leq (1 - \alpha s) \| r_t(x, \lambda, \nu) \|_2$$

We chose $\alpha = 0.05$ and $\beta = 0.6$.

Figure 1 shows the progress of the primal-dual interior points method on a specific example. In this example, we consider $m = 7600$ constraints, and 50 data points. The figure shows that the surrogate gap $\hat{\mu}$ and the norm of primal and dual residuals $\gamma = \sqrt{(\|r_{pri}\|_2^2 + \|r_{dual}\|_2^2)}$ converge to zero rapidly even when the initial point is infeasible.

5 Applications

In this section, we will focus on the possible usage of our suggested approach into real applications. We will take advantage of our *convex relaxation* algorithm, denote it as the *solver* algorithm.

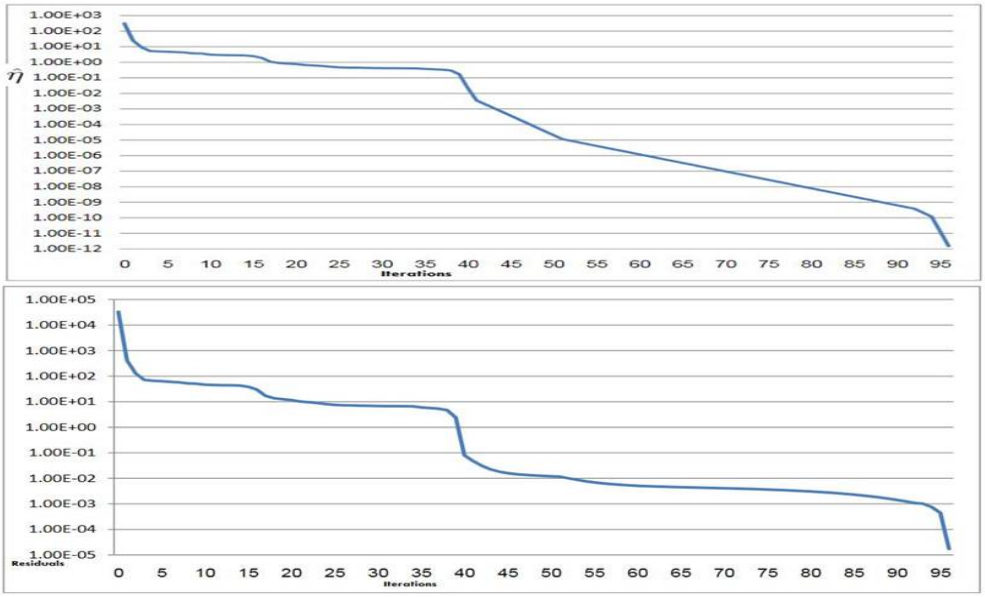

Figure 1: Progress of the primal-dual interior-point method as an example usage in the convex relaxation problem, showing surrogate duality gap $\hat{\mu}$ and the norm of the primal and dual residuals, as a function of number of iterations.

We use the idea of *expected points* that was first introduced by Lin and Vitter [6, 29]. the *solver* algorithm will output points that might be spread over several centers. We can look at each output point i that is assigned to several points as the probability distribution over centers for the output point i. Under this distribution, we denote the *expected point* to be the expected distance of a center from output point i. See, for instance Figure 2. Formally, *expected point* C_i of point i:

$$C_i = \sum_j D_{i,j} * W_{i,j}$$

The expected points might lead us to points that are located closer to the center of the cluster. This can be done when at each step we receive new expected points, which are closer to the core of the centers than any other point in the data set, and add these points to the data set. Run the algorithm for a certain number of times until we have a new strong centers. The choice of having more or less *expected points* comes directly from the weight given on the maximum entropy in the objective. Assign high weight on the maximum entropy, will have a solution that prefers having points that are assigned to more than one cluster. Thus, we receive *expected points* that might not appear in the data set and are close to the core of the clustering. The need for *expected points* may vary between the data sets, and so in our applications, we give the user the ability to choose the weight of the maximum entropy.

We will use the *solver* algorithm to solve the K-medoids problem. Our applications consist of two phases. In the *first* phase, we try to locate a fractional solution of the

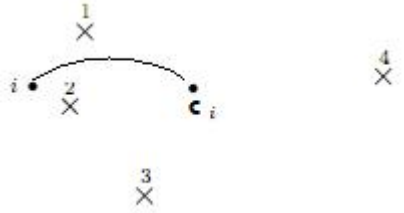

Figure 2: An expected point C_i calculated from the assignment of point i to 4 different points.

k-medoids problem. This can be done by iteratively running *solver* algorithm with a new data set derives from earlier iterations. In the *second* phase, given a fractional solution to the k-medoids instance, we round the fractional solution into one that has k centers at most.

We present our first phase algorithms:

Algorithm 1: Convex relaxation K-medoids first phase
Input: D - distance matrix, X- data set, k - number of max clusters,η - entropy factor, T- nof iteartions
Output: W - assign matrix, Y - clusters, X_{new} - new data points
(1) $X_{new} \leftarrow \emptyset, j \leftarrow 0, V \leftarrow \emptyset$
(2) **while** $j \leq T \wedge |X_{new} \cup V| > |X_{new}|$
(3) $X_{new} \leftarrow X_{new} \cup V$
(4) $W, Y \leftarrow solver\ (X \cup X_{new}, k, \eta)$
(5) $V \leftarrow$ Add new points according to new expected points derive from W, Y, X, X_{new}
(6) $j \leftarrow j + 1$
(7) (option 2: $\eta \leftarrow \frac{\eta}{2}$)
(8) **return** W, X_{new}, Y

The choice of new points are concluded from two aspects:

1. A new point is at least Δ norm far from any other point in the data set. We used $\Delta = 0.01$.

2. A new point is at least ϕ percent considered as a cluster. We used $\phi = 0.2$.

Also, we have limited the number of new points in the original data set to be 10 percent of the data set at most. This is done because we don't want to create a new data set that is not similar to the source data set.

In one of our applications, we tried to see what happens when we reduce the effectiveness of a high entropy as the process continues. This is needed for reducing

the effectiveness of a high entropy on the problem as we are getting further at each step since we might converge to the high entropy distribution and not to our main goal: optimizing the target function.

In the *second* phase, we try to round the fractional solution into one that has k centers at most. A main method that was taken is spotting the clusters by their higher rate consideration from the first-phase fractional-solution. In the experiments we used this method. There are other methods that we could have considered (See Appendix B for the Lin & Vitter approach).

For our experiments we have tried two applications for the k-medoids problem:

1. Algorithm that includes first and second main phases. Denote it as $IterKMedoids$
2. Algorithm that is the same as the previous, but reduced the entropy on each iteration in the first phase. Denote it as $IterKMedoidsReduced$.

As to solving the k-means problem, rather than using the K-means algorithm, whose performance can heavily depend on initial starting centers that are usually randomly generated, we sought an algorithm that is using the K-medoids applications. The idea is as follows:

1. Run K-Medoids algorithm on a random fractional data set, receives Y as the solution given from the application.
2. Run the K-means algorithm, as Y is the initial start.

6 Numerical Results

We have tested our applications as suggested in section 5 on a real dataset. We compared the performance of our application in comparison with k-means (See Appendix A), and k-medoids algorithms: PAM and linear relaxation, when $\eta = 0$ (See Appendix B). Cluster analysis is based on two measures: error rate of the objective function and error rate compared to the real data label.

In order to try a variety of parameters, clustering was performed with request of different cluster sizes, each time being run 5 times for the LP and Convex based algorithm and converge at least 20 times for others. We used three classification (labeled) datasets: Iris, Wine and Isolet datasets from the UCI Meaching learning archive [28].

As results in figures 3, 4 show, the clusters corresponded pretty well to a real data set. As compared with other applications, our application mostly provide low error rates when using the medium rate η i.e. when having a medium interfere by the maximum entropy in the optimization problem. The higher the data set's dimension gets, the lower the need to interfere with maximum entropy is.

As figure 5 states, on some data sets declaring medium (Wine,Isolate) or high (Isolate) weight on the maximum entropy might give better results. This implies that

Application Data set	K-Medoids problem	IterKMedoids			IterKMedoidsReduced			PAM
		weight: low (0-0.05)	meduim(0.06-0.12)	high(0.13-0.2)	low (0-0.05)	meduim(0.06-0.12)	high(0.13-0.2)	
Iris, K = 3	Target function avg.	32.08	36.84	45.22	32.13	31.61	34.51	34.6
	Error rate avg.	8.41	9.57	13.86	8.58	7.46	9.04	10.86
Iris, K = 6	Target function avg.	29.72	42	31.93	24.1	39.69	33.87	24.95
	Error rate avg.	7.09	6.14	6.41	6.05	7.21	6.65	5.38
Wine, K = 3	Target function avg.	4376	4270	4511	4376	4300	4351	4587
	Error rate avg.	25.62	26.65	25.9	25.62	26.65	25.84	26.25
Isolet, K = 6	Target function avg.	360.85	366.42	367.11	361.26	361.3	359.13	393.84
	Error rate avg.	26.47	24.04	24.9	26.31	25.06	24.47	26.2

Figure 3: Results obtained from Iris, Wine and Isolet datasets for the K-medoids problem.

Application Data set	K-means problem	K-means - partial data set							K-means - full data set						
		random start	IterKMedoids start			Reduced start			random start	IterKMedoids start			Reduced start		
			low	med.	high	low	med.	high		low	med.	high	low	med.	high
Iris, K = 3	Target function avg.	32.99	31.609	31.81	33.35	31.64	31.15	31.94	102.44	97.32	100.07	103.92	102.4	105.03	101.6
	Error rate avg.	10.2	8.92	8.86	10.97	8.95	8.02	9.04	11.61	9.29	10.45	12.31	9.29	9.29	9.29
Iris, K = 6	Target function avg.	23.2	22.34	23.21	22.98	22.17	22.9	22.93	73.76	71.87	74.02	73.6	72.11	73.43	73.41
	Error rate avg.	5.62	5.75	4.81	5.85	5.34	4.82	5.13	5.76	5.45	5.54	6.5	5.25	5.81	5.89
Wine, K = 3	Target function avg.	4551	4477	4395	4504	4477	4449	4471	16885	16593	16555	16713	16593	16555	16713
	Error rate avg.	26.32	25.58	27.2	26.78	25.58	27.2	26.78	27.28	27.41	27.63	27.5	27.41	27.63	27.5
Isolet, K = 6	Target function avg.	314.87	317.13	316.72	318.04	318.08	316.59	317.28	1996.02	2008.34	2000.03	2010.09	2012.53	2001.33	2005.31
	Error rate avg.	26.46	24.93	20.92	23.63	26.31	25.06	24.47	24.96	24.24	25.22	22.09	24.59	25.17	20.01

Figure 4: Results obtained from Iris, Wine and Isolet datasets for the K-means problem.

compared to the Linear Relaxation original algorithm (when $\eta = 0$), on some datasets we gain benefit from declaring weight on the maximum entropy.

Our applications' success of having a low error rate compared to PAM can be mainly concluded from two elements: First, PAM is only an approximation application, while our application sometimes succeed to find the optimal solution. Second, we added new points to the data sets (that are later considered to be center clusters points) that are close to the center centroid. Also, the higher the data dimension gets, PAM has a higher error rate. That can tell us that on some data sets using maximum entropy can obtain a benefit of being closer to the center of the centroid.

The generated centers from our applications can be well used as a starting guess for the K-means algorithm. When we already have some clue on the data, in most cases, the K-means algorithm will get a low error rate and a high prediction percentage. This is also obtained even if we are using a subset of the data set and not using all the data set points. This is useful, since our applications can handle only a limited number of points.

In the Isolate data set, the test results suggests that obtaining the objective function of the k-means and k-medoids problem, do not necessarily translate into better classification. Each instance in the isolet data has 617 attributes. Considering all attributes in the data, create very high entropy as the similarity between data points need to be counted by all attributes. As figure 6 shows, we observe that our appli-

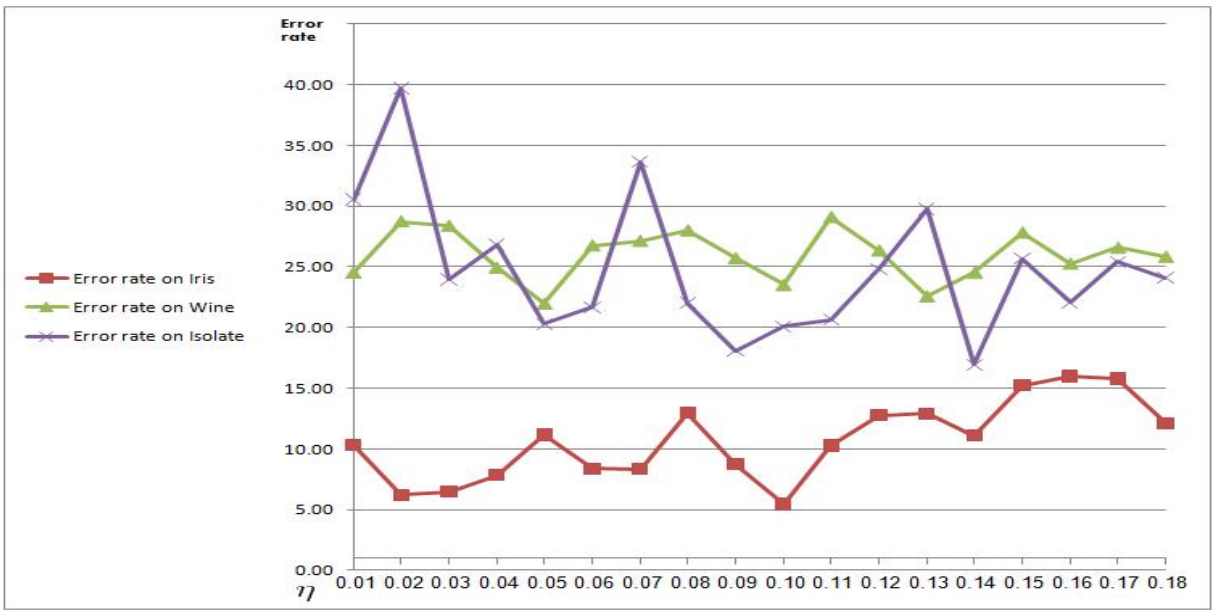

Figure 5: Error rate of Iris, Wine and Isolet datasets for the K-medoids problem as a function of η - weight of maximum entropy.

cations have large deviation rate on the results of the Isolet data sets. This can be explained by the fact that, when the data set has already high entropy and we request from our applications also to consider the maximum entropy in the equation it might leads us to two extremes. One, the algorithm will not converge and be far from the optimal solution (high error rate). Second, the algorithm would get much better points that might lead us stronger to the core of the centers (low error rate). High deviation show high error rate on average on our applications, but in the usual case we obtain low error rate than the other applications - as one can see in the average and median columns of Isolet results.

At first glance, it seemed like that our model could have helped to find the parameter k - number of clusters without the need of knowing it from advance. This is since our model only asks us to have at most k and not exactly the number of clusters. However, from the experiments results, the number of clusters that was insert as input to the model, was the same as the output of the *solver* algorithm. What is interesting, is that for all applications the clusters error rate were low even if we assigned an arbitrary k (For example, Iris dataset with $k = 6$). This showed us, how complicated is to search for the right k and that it is very data depended. Also, we observe that on most cases, in our experiments, assigning k which was different from the real k of the data, brought high entropy on the clusters distribution even if η weight was low.

In general, choosing the right η has a major influence on our results. For large values of η (0.13-0.2), it can be noticed that the algorithm will focus on solving the

	Isolet Data K = 5			
Algorithm	Weights	Best value	Average	Median
IterKMedoidsReduced	Low weight	14.98	26.31	22.75
	Medium weight	16	25.06	21.8
	High weight	13.75	24.47	22.71
PAM	-	15.55556	26.2	27.04
K-means - random start	-	12.39	26.46	26.7
K-means - IterKMedoidsReduced start	Low weight	15.15	26.31	27.25
	Medium weight	12.39	25.06	23.62
	High weight	10.78	24.47	24.16
K-means full - random start	-	15.91	24.96	24.92
K-means full - IterKMedoidsReduced start	Low weight	17.38	24.59	25.34
	Medium weight	16.9	25.17	23.63
	High weight	15.91	20.01	17.6

Figure 6: Best, Average, Median results of different applications for the Isolet dataset.

maximum entropy on the clusters' distribution and not on our main goal, optimizing the LP relaxation K-medoids problem. We can see from the results that, when sometimes the maximum entropy overweights, it denies the algorithm from converging. Also, from this reason, using the option of reducing the entropy at each step gives better results.

7 Summary and Conclusion

In this project, we proposed a novel algorithm to attack the k-means and k-medoids problems. We came up with several applications and checked the results compared to other similar algorithms. The algorithm uses the primal-dual method to solve a convex relaxation k-medoids problem as introduced. We created a few applications that use this algorithm. We also demonstrated that these applications can be applied to real data sets.

The main results show that in some cases, by finding the balance η that fits the data set, the model succeeded to have a low error rate. Also, we showed that our method will serve as a good reference in evaluating various heuristic approaches as K-means.

7.1 Future Work

In the future, our applications will most certainly need to be improved. The main areas for improvement involve:

1. The algorithm runs on a reasonable time for small data sets but does not for large data sets. When a number of data points grow, the algorithm performs very slowly as it tries to solve a big set of linear equations. The algorithm performs a polynomial run-time as respect to the input data set. In our implementation, we tried to be efficient by using the interior points method. The algorithm converges rapidly, even though, the initial input is not a feasible solution (coverage varies up to 100 iterations). Furthermore, though the sparsity of the Hessian matrix was in our favor, it seems that a future work is required in order to improve the algorithm's efficiency.

2. The rounding solution in our algorithm is limited. Picking a good rounding has a major influence on the results of the algorithm. For example, Lin & Vitter approach (See Appendix B).

3. One more area that we can improve is in the criteria of picking new points from the obtained *expected points*. By that, we can gain a better performance and reduction of the entropy on next steps in the algorithm.

4. Theoretical aspect to the maximum entropy.

A The K-means algorithm

In the k-means problem, we accepts data set X and integer k represents the number of clusters. We try to partition the data into k disjoint sets $(S_1, ..., S_k)$ where each set is represented by center. We want to find the k centers so as to minimize the sum of the square of the Euclidean distance of each point in X to its nearest center. Formally, our objective:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in S_i} d(x_j, \mu_i) \tag{8}$$

where μ_i is the mean of points in S_i, and $d(x, y) = \| x - y \|_2^2$.

It is known that finding the optimal K-Means solution is NP-hard and even NP-hard to approximate. As an alternative,an efficient heuristic simple algorithm, The k-means, is used to find an approximation solution [2]. The K-means algorithm converge to the minimum local whose performance depend heavily on its initial starting clusters that are usually randomly generated.

The K-means algorithm:

Input: Data points X, number of clusters k.

1. Randomly choose k cluster center locations $\mu_1, ..., \mu_k$ and make them as initial centroids.

2. Repeat until converge:

   Assignment step: Assign each data point to its closest mean cluster.

   $$\forall i = 1...k \; set \; S_i = \{x \in X : \min_j \| x - \mu_j \| = \| x - \mu_i \|\}$$

   Update step: Calculate the new means to be the centroids of the data points.

   $$\forall i = 1...k \; set \; \mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

We will use this algorithm to compare the results of our applications.

B The K-medoids algorithm

The K-medoids problem is related to the k-means problem. The K-medoids ask us to partition the data set up into separate clusters as the same as the K-means except that the cluster centroid being selected as one of the data points in the cluster and not the mean. We will show two examples of K-medoids algorithms.

B.1 PAM

The algorithm PAM is an adaption of the k-means algorithm [21]. The algorithm take advantage that the distance matrix is only computed once (unlike the k-means algorithm). The algorithm runs as follows: *The Partition Around Medoids : (PAM)*
Input: Data points X, Distance matrix D, number of clusters k.

1. Randomly pick k data points to serve as initial medoids: $y_1, ..., y_k$

   Assigning step: Assign each data point to its closest medoids using the distance matrix.

   $$\forall i = 1...k \; set \; S_i = \{x \in X : \min_j \parallel x - y_j \parallel = \parallel x - y_i \parallel\}$$

   Update step: Calculate the objective cost function.

   Pick up step: Randomly guess another data point, non medoid, to replace a current medoid, assign and calculate the cost again. If the new cost is greater than the old cost then stop the algorithm.

2. Repeat until there is no change in the medoid.

B.2 Linear relaxation algorithms

As similar to our convex relaxation algorithm, follows an algorithm that take advantage of the liner programming relaxation approach. The algorithm starts by running the linear programming to solve the linear relaxation of the k-medoids problem (1) and then translate the fractional solution to a feasible one.

1. Lin & Vitter translate the fractional solution into a feasible one using the following approach:

   Algorithm 2: Lin & Vitter rounding solution - second phase
   Input: $C_1, ..., C_n$ - Expected points from the first phase solution, $S = \{1, ..., n\}$
   Output: T - Indexes of the clusters
   (1) $T \leftarrow \{\}$
   (2) **while** $S \neq \{\}$
   (3) $i = \operatorname{argmin}_S C_i$
   (4) $T \leftarrow T \cup \{i\}$
   (5) $S \leftarrow S - \{i' \in S | B(i, 2C_i) \cap B(i', 2C_{i'}) \neq \emptyset\}$
   (6) **return** T

While Lin & Vitter round solution consider the distances between each expected points, their algorithm might expand the number of cluster to at most $2k$ medoids. Thus, we can use this algorithm with $\frac{k}{2}$ centers.

2. Greedy algorithm that picks the k most points that defined by the fractional solution as the best chances as clusters. (Same as the convex relaxation algorithm with $\eta = 0$)

References

[1] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In Proceedings of the 31st Annual ACM Symposium on Theory of Computing, pages 1-10, May 1999.

[2] J. MacQueen. Some methods for classification and analysis of multivariate observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, volume 1, pages 281-297, 1967.

[3] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In Proc. of the 5th International Workshop on Approximation Algorithms for Combinatorical Optimization Problems (APPROX), pages 229-242, 2002.

[4] D.B. Shmoys, E. Tardos, and K.I. Aardal. Approximation algorithms for facility location problems. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pages 265-274, 1997.

[5] S. Arora, P. Raghavan, S. Rao. Approximation schemes for Euclidean k-medians and related problems , Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 106-113, 1998.

[6] J.-H. Lin and J. S. Vitter. Approximation algorithms for geometric median problems. Information Processing Letters, 44:245-249, 1992.

[7] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. Journal of the ACM, 48:274-296, 2001.

[8] C. D. Manning and H. Schutze. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, 1999.

[9] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey. The uncapacitated facility location prob- lem. In P. Mirchandani and R. Francis, editors, Discrete Location Theory, pages 119-171. John Wiley and Sons, Inc., New York, 1990.

[10] M. Mahajan, P. Nimbhorkar, K. Varadarajan. "The Planar k-Means Problem is NP-Hard". Lecture Notes in Computer Science 5431: 274-285, 2009.

[11] A. Jain, R. Dubes. Algorithms for Clustering Data. Prentice Hall, 1988.

[12] R. Kannan, S. Vempala, A. Vetta. On clusterings - good, bad and spectral. In FOCS, pages 367-377, 2000.

[13] O. Kariv, S Hakimi. An algorithmic approach to network location problems. II: The p-medians, SIAM Journal on Applied Mathematics, pages 539-560. 1979.

[14] N. Megiddo, K. Supowit. On the complexity of some common geometric location problems, SIAM Journal on Computing, 13, pages 182-196. 1984.

[15] J. Lin, J.S Vitter. Approximations with minimum packing constraint violation, Proceedings of the 24th Annual ACM Symposium on Theory of Computing, pages. 771-782. 1992.

[16] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit. Local search heuristics for k-median and facility location problems, Proceedings of the ACM Symposium on Theory of Computing, pages 21-29. 2001.

[17] A. Ratnaparkhi. A simple introduction to maximum entropy models for natural language processing. Technical Report 97-08, Institute for "Research in Cognitive Science, University of Pennsylvania. 1997.

[18] A. Berger, S. Pietra, V. Pietra. A maximum entropy approach to natural language processing. Computational Linguistics, 22 pages 39-71. 1996.

[19] M. Balcan, H. Roeglin, S. Teng. Agnostic clustering. In Proceedings of the 20th International Conference on Algorithmic Learning Theory. 2009

[20] J. Han, M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, San Francisco, pages 403-420, 2000.

[21] A. P. Reynolds, G. Richards, and V. J. Rayward-Smith, "The application of k-medoids and pam to the clustering of rules", in Intelligent Data Engineering and Automated Learning, ser. Lecture Notes in Computer Science. Springer Berlin. pages 173-178, 2004.

[22] R. D. Monteiro, and I. Adler. Interior path following primal-dual algorithms. part I: Linear programming. Mathematical Programming, Vol. 44, No. 1. pages 27-41, 1989.

[23] Y. Nesterov, and A. Nemirovskii. Interior-Point Polynomial Methods in Convex Programming. Society for Industrial and Applied Mathematics. 1994.

[24] S. Boyd, and L. Vandenberghe. Convex Optimization. Cambridge University Press, Cambridge, UK. Chapter 11, 2004.

[25] Y. Nesterov, and M. J. Todd. Self-scaled barriers and interior-point methods for convex programming. Math. Oper. Res., 22(1). pages 1-42, 1997.

[26] F. Maffioli, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi. Combinatorial Optimization. Wiley, New York. Chapter 5, 1979.

[27] L. Kaufman, and P. Rousseeuw. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons. 1990.

[28] UCI repository, http://archive.ics.uci.edu/ml/datasets.html.

[29] S. Dasgupta. Unsupervised learning course - Lecture notes, CSE 291. UCSD, University, Computer Science Department. Available at http://cseweb.ucsd.edu/ dasgupta/291/, Spring 2008.

[30] M. Su and C. Chou, âĂIJA modified version of the K-means algorithm with a distance based on cluster symmetry. IEEE Trans, Pattern Anal.Mach. Intell., vol. 23, no. 6, pp. 674-680, 2001.

[31] P. Tan, M. Steinbach and V. Kumar, Introduction to Data mining, Chapter 8 - Cluster Analysis: basic concepts and algorithms. Addison Wesley, 2005.