Spring 5-2016

Mixed k-means clustering in computer adaptive learning

Camden G. Bock
Bates College, cbock@bates.edu

Bonnie Shulman PhD
Bates College, bshulman@bates.edu

Mixed k-means clustering in computer adaptive learning

Camden Bock

Department of Mathematics

Bates College

Lewiston, ME 04240

Abstract. The ASSISTments project from Worcester Polytechnic Institute provides a free web-based intelligent tutoring system including two levels of differentiation, that are manually programmed by teachers and researchers. Problems assigned through ASSISTments can be programmed in trees, where the sequence of problems adapts to the student's performance on each question. Within each problem, if a student enters an incorrect response the ASSISTments system provides scaffolded feedback to target the student's misconception. This thesis begins to develop an educational data mining algorithm to automate this differentiation. First, an adaption of Alsahaf's mixed k-means clustering algorithm is proposed to handle a mix of categorical and numeric data. Second, the algorithm is implemented in MATLAB and its performance is compared to Alsahaf's results on benchmark data sets. Finally, the MATLAB implementation is applied to ASSISTments data sets from 2009 and 2012 to develop a predictive model.

Mixed k-means clustering in computer adaptive learning

An Honors Thesis
Presented to the Department of Mathematics
Bates College
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
by
Camden Bock
Lewiston, Maine
May 9, 2016

Contents

List of Tables

List of Figures

Acknowledgments

I would first like to thank my thesis advisor Prof. Bonnie Shulman of the Department of Mathematics at Bates College. Prof. Darby Ray of the Harward Center for Community Partnerships at Bates College and Tammy Thilbodeau of Lewiston High School supported this thesis with both funding and advising through the Community Engaged Research Fellowship.

I would also like to thank Dr. Neil Heffernan of the Department of Computer Science at Worcseter Polytechnic Institute for supporting the project, answering questions throughout the year, supplying the data for the application, and for serving as an outside examiner on the thesis evaluation.

I would like to thank Prof. Meredith Greer of the Department of Mathematics at Bates College for advising me throughout my time at Bates (although I wasn't always able to listen), and for examining my thesis as a member of the department. I would also like to acknowledge Prof. Ryan Nelson of the Department of Chemistry at Bates College as the non-department examiner for the thesis evaluation.

Finally, I must express my very profound gratitude to my parents James and Kim Bock and my suite-mates Alex Andonian, Elliot Grace and Seth Dobieski for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Camden Bock '16

Introduction

Intelligent tutoring systems provide instruction that is differentiated to individual student needs. Computer adaptive learning systems, a subset of intelligent tutoring systems, differentiate for individual students by frequently adapting to a student's behavior. While other intelligent tutoring systems may rely on tracking, defining students by demographic labels (e.g. socioeconomic status, ethnicity, IEP or special education designation), computer adaptive learning has the potential for more equitable instruction. One of the motivations for the work in this thesis is to help realize that potential.

Intelligent tutoring systems also have the potential to provide a large number of students with additional support structures and more equitable access to content. Standards based education (SBE) requires frequent and specific intervention and allows for multiple methods for demonstrating proficiency. As a support to the teacher and student, intelligent learning systems can provide struggling students with continuously differentiated skills practice. Additionally, proficiency on standards is assessed while the student is simultaneously learning. This eases the burden on teachers and schools of reassessing failing students both by minimizing time spent grading and generating alternative assessments.

Finally, many intelligent learning systems are web-based, and can be accessed by netbooks, smartphones, tablets and chromebooks in addition to traditional desktops and laptops. With 1-1 technology initiatives, these intelligent learning systems can be accessed by students without additional hardware or software costs to the school districts.

ASSISTments, a web-based intelligent tutoring system from Worcester Polytechnic Institute (WPI), funded by the National Science Foundation, is a free resource for districts, educators and researchers. Although ASSISTments has adaptive differentiation, it requires manual entry by the user. Commercial systems have automated this differentiation of problem sets. The addition of computer adaptive features to the ASSISTments platform would increase its value to educators and researchers and decrease the labor cost of implementing ASSISTments

in local districts. Computer adaptive features would be most useful in either the design of differentiated and adaptive sequences of problems towards proficiency on a skill or in diagnosing misconceptions and appropriate interventions in students' responses.

ASSISTments has collected data from thousands of students over the last decade, focused on secondary mathematics in Maine and Massachusetts. [1] ASSISTments currently allows teachers to assign linear sequences of problems, or a sequence of problems determined by the student's performance (correct/incorrect) on the previous problem. Creating these problem sets is a time-intensive task for the teacher, and does not take advantage of years of student data. ASSISTments provides scaffolding, an additional layer of differentiation within each problem. When a student enters an incorrect response, the system diagnoses the student's error, and asks additional questions that break the problem into smaller pieces around that student's error. If the student has sufficient background knowledge/instruction, this approach should identify the student's misconceptions and correct them with additional questions within the students zone of proximal development.

In order to automate either the problem sequence or problem scaffolding, students or their responses need to be clustered, so that regression models can be made on the smaller partitions, for more precise predictions. Clustering uses unsupervised machine learning, where the k-means algorithm efficiently partitions data that is structured in $m-$dimensional spheres. Points near a cluster center (mean) can be predicted to behave as the cluster center or based on the deviation from the cluster center.

To find spherical groupings, k-means clustering needs a well-defined distance between data points. The data from ASSISTments, as well as many other systems, has a mix of categorical and numeric data. The squared Euclidean distance cannot be directly applied to categorical data. Ahmad et al. propose a measure of distance between categorical variables, and a weighted measure of distance between data points with numeric and categorical attributes [**1**]. Alsahaf implemented Ahmad's algorithm into MATLAB, tailored to a specific set of data [**2**].

In this thesis, we generalize Alsahaf's mixed k-means algorithm. Our algorithm accepts dense matrices as an input. The results of our algorithm agree with those reached by Ahmad on benchmark data sets.

[1] A recent study by WPI, SRI and the University of Maine note both significant gains in the performance of grade seven students in a randomized control trial including 44 schools in Maine.

Outline

Chapter 1 discusses the numeric k-means clustering algorithm. This algorithm is built-in to MATLAB, and is a very efficient algorithm for clustering numeric data sets. However, this algorithm is restricted to numeric data, and does not have a meaningful measure of distance for categorical data.

Chapter 2 discusses the mixed k-means clustering algorithm, that adapts the numeric k-means algorithm to be able to handle categorical data. This includes a definition of 'significance' to weight the distance of each attribute, and a definition of distance between categorical values.

Chapter 3 discusses three tools for error analysis when using these clustering algorithms: silhouette values, performance ratios, and visualizations. Silhouette values measure the average distance of a data point and its' closest cluster center, relative to the second closest cluster center. Performance ratios are used with benchmark sets, where the classification of data is known but hidden to the clustering algorithm. The performance ratio is the percent of data points that are assigned to the appropriate cluster or class. A two-dimensional and three-dimensional visualization are proposed, but the properties of these visualizations have not been fully explored.

Chapter 4 discusses the MATLAB implementation of the mixed k-means algorithm.

Chapter 5 discusses the results of the MATLAB implementation on five UC Irvine benchmark data sets.

Chapter 6 discusses the limitations of the application of the MATLAB implementation of the mixed k-means clustering algorithm to the ASSISTments data set.

Finally, Chapters 7 and 8 integrate the information from previous chapters with conclusions, questions for future research and a reflection on the product and process of this thesis.

CHAPTER 1

Numeric K-means Clustering

Numeric k-means clustering is an algorithm for partitioning N data points with M attributes into K clusters. To achieve this clustering, the algorithm assumes that the data is naturally grouped into M-dimensional spheres, determined by a measure of distance. In this paper, the squared Euclidean distance is used to measure distance between data points and define the spherical groupings.

Numeric k-means clustering is limited by its assumption of the natural M-dimensional spherical structure of data, and the inability to handle categorical attributes.

1. Numeric K-means Clustering Procedure

Given K, the number of clusters, the numeric k-means clustering algorithm randomly assigns each data point to a cluster. For each cluster, the cluster center is a vector of length m. Each entry in the vector is defined as the average value of the corresponding attribute, across all data points in the cluster.

Data points are then assigned to the closest cluster center. Let x_n be the n^{th} data point, and let cc_k be the center of the k^{th} cluster. The squared Euclidian distance $d(x_n, k)$ between x_n and cc_k is given by Equation 1.1, where x_{nm} is the m^{th} component of x_n and cc_{km} is the m^{th} component of cc_k.

$$d(x_n, k) = \sum_{m=1}^{M} (x_{nm} - cc_{km})^2 \tag{1.1}$$

The data point is then assigned to the cluster that minimizes the Euclidean distance, $d(x_n, k)$.

$$d(x_n, k)_{min} = \min\{d(x_n, 1), d(x_n, 2), \ldots, d(x_n, K)\}$$

Using the revised cluster assignments, each new cluster center is defined as the average value of the each attribute, across all data points in the new cluster. The process of cluster assignment and cluster center definition is iterated until no data points are reassigned to a new cluster center.

2. Results of Numeric K-means Clustering

The total distance between data points and their assigned cluster center converges to a local optimum. In order to discuss the convergence, $D(r)$ is defined below to be the total distance after the r^{th} iteration.

For each of the K clusters each iteration minimizes the distance between the data points within the cluster and the cluster center, as defined in 1.1. Let $D(r)$ be the sum of the squared Euclidean distances between each data point, and its assigned (closest) cluster center after the r^{th} iteration, where $cc_{km}(r)$ is the closest center arrived at in the r^{th} iteration. [1]

$$D(r) = \sum_{n=1}^{N}(\min_{k=1}^{K}(\sum_{m=1}^{M}(x_{nm} - cc_{km}(r))^2)) = \sum_{n=1}^{N} d(x_n, k)_{min}$$

2.1. Convergence. An intuitive explanation of the convergence follows, however it is not a proof. [2] This example assumes that only one data point is reassigned on each iteration. This becomes significantly more complex when many data points are simultaneously reassigned (the cluster centers are not recalculated until all data points have been reassigned).

After the r^{th} iteration, if any data point x_n can be reassigned to a closer cluster. In most cases, the reassignment of a single data point will decrease the total distance. If the data point (x_n) is not an outlier, there should be little change to the the locations of the cluster centers.

Let $oldC_1$ be the old position of the old cluster center, as a row vector. Let $oldC_2$ be the old position of the new cluster center. Let $newC_1$ be the new position of the old cluster center. Let $newC_2$ be the new position of the new cluster center. Let n_1 and n_2 be the number of data points in the old and new clusters (excluding x_n). However, if a data point x_n is an outlier, it could shift the new cluster center away from all other data points in the cluster. If the inequality below is true, then it is possible for $D(r) < D(r+1)$.

$$(x_n - oldC_2)^2 - (x_n - oldC_1)^2 \leq n_1(oldC_1 - newC_1)^2 + n_2(oldC_2 - newC_2)^2$$

[1] This can also be expressed with an indicator function [**12**].

[2] The complete proof of convergence is beyond the scope of this thesis, but is discussed by Bottou [**5**] with a proof in Bottou's dissertation at the Universite de Paris [**4**]. This source is written in French, and it was not clear where the proof of convergence was included. A number of other articles note the convergence of the numeric k-means algorithm without citation or proof.

When there are many data points in each cluster, we assume that the expression above will be false. This assumption is reasonable when the data set is normalized and outliers are removed. Then the total distance decreases with each iteration, and is bounded by 0. So, there exists an infimum that is greater or equal to 0 that bounds the sequence $D(r)$, and $D(r)$ is a monotonic sequence. Then, $D(r)$ must converge to the infimum.

2.2. Local Optima. As shown in [**12**], the numeric k-means procedure guarantees a local minimum of the total distance, $D(R)$. However, according to Rogers, the numeric k-means procedure does not guarantee a global minimum of the total distance. [3]

A function is convex if, for any two points on the graph of the function, the line segment connecting those points lies above or on the graph. In a convex function any local minimum is also a global minimum [**15**].

A non-convex function is a function where there exists two points on the graph of the function such that the line segment connecting those points lies below the graph. A local minimum of a non-convex function is not necessarily a global minimum.

The presence of a local or global optimum may be dependent on the random seed of the initial cluster assignments [**12**]. Since not every clustering is an optimal clustering, we run multiple trials of the clustering algorithm, each with a different random seed for the initial assignment of data points to clusters. [4] For some benchmark data sets, there are small deviations in the performance ratio between trials (see section 2). To assess the performance of benchmark data sets, we average the performance ratio over hundreds of trials, rather than selecting the best of the local optima.

In application, the clustering results are used from the trial with the minimum average silhouette value. The silhouette value is a measure of the goodness of fit of a model for a particular data point. In both numeric and mixed k-means clustering, the distance between a data point and each of the cluster centers can be calculated. The silhouette value compares the distance to the two cluster centers with the least distance, as the ratio of their difference divided by their maximum. The

[3]The problem of local optima in the k-means algorithm is known, but has not been extensively studied [**17**]. According to Trivedi et al., these local minima are due to the k-means algorithm optimizing a non-convex distance function [**16**].

[4]Initial assignment of data points can be improved in the numeric k-means algorithm [**6**] to avoid local optima, but this does not make significant gains from random assignment [**17**].

mean of all silhouette values in a data set is a measure of the goodness of fit across the data set. The silhouette value is formally defined in Chapter 3 Section 1.

CHAPTER 2

Mixed K-means Clustering

Similar to the numeric k-means clustering discussed in Chapter 1, the mixed k-means clustering is an algorithm that partitions N data points with M attributes into K clusters. This algorithm also assumes that the data is naturally grouped into M-dimensional spheres. Mixed k-means clustering improves upon the numeric k-means algorithm, by handling both categorical and numeric attributes.

The challenge of mixed k-means clustering is to define a meaningful distance between values of categorical attributes. For many categorical attributes, a well-ordered ranking of values is not obvious. For example, the attribute "animal type" that has a set of 5 possible values {*dog, cat, horse, snake, mouse*} does not have an obvious ranking. In the case of mathematics education, skills for mathematical practices and procedures can be defined in a directed graph of prerequisites [**10**]. However, this directed graph of skills cannot simply be projected into one dimension for a meaningful measure of ranking or distance [1].

Consider an attribute "vehicle type" that has a set of 4 possible values: *sedan, truck, van,* or *SUV*. A ranking of similarity could be determined as follows:

(1) sedan
(2) SUV
(3) van
(4) truck

What is the distance from a sedan to an SUV relative to the distance from a sedan to a truck? Even with a well-ordered ranking, there is not always an obvious measure of distance between variables.

As the number of values in categorical attributes increases, the problem of meaningful distance can become more complex. In the example of vehicle types, the distance between a crossover and a minivan might differ greatly among different manufacturers. Additionally, their ranking among other vehicles may differ. A Chevrolet Suburban (SUV)

[1] While a distance can be calculated using a directed graph, this is a projection into one dimension that is not necessarily well-ordered. Without a well-ordered ranking, it is difficult to define meaningful cluster center (mean) values.

may be closer to a truck than some vans, while the Honda CRV (SUV) approaches a large sedan. Even if these distances are defined in a one-dimensional system, relative distance between attributes needs to be well defined as well.

The mixed k-means clustering algorithm approaches distance between values of categorical attributes using probabilities to define a measure of distance [**1**]. In this approach, the distance between categorical values is determined by the proportion of equivalent values within the same cluster and a weight of significance for each attribute.

1. Mixed K-means Procedure

The data is encoded as a matrix with N rows (data points) and M columns (attributes). The user defines each of the attributes as either numeric or categorical.

(1) Each numeric attribute is normalized, as a column vector, scaling the output between 0 and 1.
(2) A copy of each numeric attribute is discretized, as described in Section 2.1.
(3) From the discretized numeric attributes and the categorical attributes, a significance weight is assigned to each attribute, as described in Section 2.2.
(4) Similar to the numeric k-means clustering algorithm, each data point is randomly assigned to one of k clusters.
(5) The center or mean of each cluster can then be calculated, as described in Section 2.3.
(6) The distance to each cluster center can then be calculated for each data point, as described in Section 2.4.
(7) Each data point is then reassigned to the closest cluster center.
(8) If at least one data point has changed cluster assignments, this process is repeated, starting at the calculation of centers of each cluster. When cluster assignments are constant between two iterations (or within a predetermined tolerance), the algorithm stops.

2. Calculations for the Mixed K-means Procedure

In this section, we discuss the formal calculations for each step in the mixed k-means procedure. In the next section, we walk through an example data set to make this procedure more concrete.

2.1. Discretization of the data. A continuous numeric attribute may have as many as N unique values. In Section 2.2, the value of

significance is based on the co-occurrence of unique values of two attributes, within a subset of the data points. As the number of unique values of an attribute approaches the number of data points, there is an increasing number of cases where a unique value of one attribute maps to exactly one unique value of another attribute. When there are marginal differences between the unique values in a subset of a numeric attribute, (e.g. duration of 18.00, 18.01, and 17.98 seconds), the value of significance is less meaningful.

To solve this problem in the calculation of significance, numeric attributes are discretized by partitioning unique values of a numeric attribute. The numeric k-means clustering algorithm partitions the unique values such that the ratio of the distance between the closest cluster center and the second closest cluster center is minimized. This decreases the number of unique values of the attribute.

The numeric k-means clustering of one $N \times 1$ attribute vector (all rows and one column of the matrix) returns a clustering that minimizes the distances between values of that one attribute. This gives the necessary grouping of data points of similar value for calculating significance.

Let κ be the number of means for the numeric k-means clustering *only* in the discretization of a given attribute. In this implementation, each attribute uses the value of $\kappa < \kappa_{max}$ that minimizes the average silhouette value, where κ_{max} is a user defined upper bound. In the case that multiple values of $\kappa < \kappa_{max}$ minimize the average silhouette value, the least of these values is used.

A greater value of κ may create tighter clusters that more accurately reflect the structure of the data. However, as κ approaches N, the probability of any value mapping to a subset of the values of another attribute decreases and the discretized attribute is equivalent to the original attribute. Thus, this modification is only suitable for $\kappa < \kappa_{max}$, where $\kappa_{max} << N$. Additionally, as the number of unique values in the discretized data increases, the computational time for the calculation of significance scales poorly.

With this adaptive value of κ, we have meaningful weights for the calculation of distance, within a reasonable amount of computational time. Additionally, the adaptive value of κ does not assume that all attributes in a data set have similar structure.

Note, in Ahmad et al [**1**], a constant κ is used for all attributes. Similarly, in Alsahaf's implementation [**2**], a specific $\kappa = 4$ is used in the numeric k-means clustering to discretize each attribute. The use of a variable κ_i on benchmark data sets has slightly different performance on

those data sets, but reasonably approximates the reported performance of algorithms with constant κ, as described in Chapter 5.

2.2. Significance of Attributes. In the numeric k-means clustering algorithm, each attribute is given equal weight in the calculation of Euclidean distance. In many situations, some attributes are more relevant to the underlying structure of a data set than others. For example, in the Heart Disease benchmark data set in Chapter 5, we would expect that the attributes *chest pain type, sex, exercise induced angina, and maximum heart rate* to be better indicators of absence or presence of heart disease than the attribute *age* or other data that could have been collected on cognitive orientation[2] and stress level. The significance places a weight on each attribute in the calculation of distance to minimize the effect of confounding variables.

The significance or weight of each attribute is calculated from the discretized numeric and categorical data. The discretized numeric data is only used in determining the significance of each variable, and is not used in the clustering or distance computations. The significance is a measure of the similarity of attributes, or more specifically, the correlation of values of partitions of data points between one attribute and all other attributes. The significance allows normalized variables to be weighted in the measure of distance, giving some attributes more impact on the result than others.

Let κ_i be the value of κ used to discretize numeric attribute A_i (in the case of a categorical attribute A_i, the number of unique values). Let u_i be a vector of all unique values of the discretized A_i, where u_{ib} is the b^{th} component of u_i. Then the set $\{(u_{ib}, u_{ic})|b, c \in \mathbb{Z}, c < b \leq \kappa_i\}$ is the set of all unique pairs of unique values of A_i.

Let $P_i(\Omega/x)$ be the the conditional probability that a component having value x in attribute A_i has a value y in attribute A_m such that $y \in \Omega$ where $\Omega \subseteq u_m$ [**1**]. Let $(u_m \setminus \Omega)$ be the set of unique values in u_m (of attribute A_m) that are not in set Ω.

The distance between attribute values a and b of attribute A_i with respect to attribute A_m is defined as

$$\delta^{im}(a, b) = P_i(\omega/a) + P_i((u_m \setminus \omega)/b) - 1.0$$

where $\omega \subseteq u_m$ such that $P_i(\omega/a) + P_i((u_m \setminus \omega)/b)$ is maximized [**1**].

The significance w_i of the numeric attribute A_i is defined by the average value of the distance function δ over all pairs of unique values

[2]In Emergency Medicine, this is often referred to as A&Ox4. The normal, healthy state of cognitive orientation for a patient is alert and oriented to place, time, person, and self.

of attribute A_i with respect to every other attribute [**1**]. Recall, the set $\{(u_{ib}, u_{ic})|b, c \in \mathbb{Z}, c < b \leq \kappa_i\}$ is the set of all unique pairs of unique values of A_i.

$$w_i = \frac{2}{\kappa_i(\kappa_i - 1)(M-1)} \sum_{b=1}^{\kappa_i} \sum_{c=b+1}^{\kappa_i} \sum_{m=1, m\neq i}^{M} \delta^{im}(u_{ib}, u_{ic})$$

Here, we sum the distance function $\delta^{im}(u_{ib}, u_{ic})$ across each other attribute, and each unique pair of values of attribute A_m. To compute an average, it is multiplied by $\frac{1}{\kappa_i(\kappa_i-1)}$ (the reciprocal of the number of unique pairs) and by $\frac{1}{M-1}$ (the reciprocal of the number of other attributes). Substituting for $\delta^{im}(u_{ib}, u_{ic})$, we have the following expression for significance.

(2.1)

$$w_i = \frac{2}{\kappa_i(\kappa_i - 1)(M-1)} \sum_{b=1}^{\kappa_i} \sum_{c=b+1}^{\kappa_i} \sum_{m=1, m\neq i}^{M} [P_i(\omega/u_{ia}) + P_i((u_m \backslash \omega)/u_{ib}) - 1.0]$$

where $\omega \subseteq u_m$ such that $P_i(\omega/a) + P_i((u_m \setminus \omega)/b)$ is maximized. So, the significance w_i is the average probability that any two unique components a, b will have corresponding values in different partitions of the set of unique values of A_m (where A_m is partitioned into two complementary sets that are optimized to maximize that probability), across all other attributes.

The value w_i of significance will not be used in the calculation of cluster centers, but will be used in the calculation of distance between the data points and cluster centers.

2.3. Cluster Centers. For numeric attributes, the cluster center is defined as the mean of the values of the attribute across the data points in the cluster. For categorical attributes, cluster centers are defined by the mode of the values of the attribute across the data points in the cluster.

In the numeric k-means clustering algorithm, since cluster centers are defined by the mean value of each attribute, the distance between a data point and its cluster center is easily calculated. For categorical attributes, the mode is a well defined value. If the value of a data point is equivalent to the value of the mode of its cluster, the distance to the center of the cluster is zero. Otherwise the distance is defined in Section 2.4. In this section, the cluster center is defined to allow for a meaningful calculation of distance for both categorical and numeric attributes.

Let x_n be the data point in row $n \leq N$. Let ϕ_k be the set of data points, x_n such that x_n is in the k^{th} cluster. Let cc_k be the k^{th} cluster center. Let x_{nm} be the value of x_n for attribute A_m.

$$cc_{km} = \begin{cases} \text{mean}_{x_n \in \phi_k}(x_{nm}) & A_m \text{ is a numeric attribute} \\ \text{mode}_{x_n \in \phi_k}(x_{nm}) & A_n \text{ is a categorical attribute} \end{cases}$$

For categorical attributes, the probability of each unique value of the attribute occurring within the cluster is also stored, as defined in Equation 2.3. This will be used to measure the distance from the center for categorical values not equal to the cluster center (mode), in Section 2.4.

Recall, u_m is a list of the unique values of the attribute A_m, where u_{ma} is the a^{th} component of u_m. Let ϕ_{ki} be the set of data points in ϕ_k such that the m^{th} component of the data point has the value of u_{mi}.

$$\phi_{ki} = \{x \in \phi_k | x(m) = u(i)\}$$

Let cd_{km} be a vector representing the probability of a data point in cluster k having a value corresponding to the entry in cd_{km} for categorical attribute A_m in the k^{th} cluster, defined as [**1**]

$$cd_{km}(i) = \frac{|\phi_{ki}|}{|\phi_k|}.$$

2.4. Distance to Cluster Center. In the numeric k-means clustering algorithm, a data point is assigned to the cluster center that minimizes the Euclidean distance. In this section, we define a measure of distance that includes the new definition of cluster center and the weight of significance. This definition of distance allows us determine the new cluster assignments of each data point in each iteration.

Let x_n be the n^{th} data point. The distance between x_n and the k^{th} cluster center, for categorical variable A_m is defined by the Euclidean distance ($\rho_m(x_n, k)$).

$$\rho_m(x_n, k) = (w_m(x_{nm} - cc_{km})) \tag{2.2}$$

The distance between x_n and the k^{th} cluster center, for categorical variable A_m is defined as 0 if the values are equal or as the expression below if the are not equal.

(2.3)

$$\rho_m(x_n, k) = \begin{cases} (\sum_{i=1}^{|u_m|} \frac{cd_{km}(i)}{M-1}[P_m(\omega/x_{nm} + P_m((u_m \setminus \omega)/u_m(i)) - 1.0]), & x_{nm} \neq cc_{km} \\ 0, & x_{nm} = cc_{km} \end{cases}$$

where $\omega \subseteq u_m$ such that $P_m(\omega/x_{nm} + P_m((u_r \setminus \omega)/u_m(i))$ is maximized [**1**]. This measure of distance is similar to significance, but the domain is restricted to the cluster rather than the whole data set. So, the total distance between the point x_n and the k^{th} cluster center is

$$d(x_n, k) = \sum_{m=1}^{M} [\rho_m(x_n, k)]^2. \tag{2.4}$$

3. Results of Mixed K-means

The mixed k-means clustering will minimize D, the sum of the distance between each cluster and its cluster center.

$$D = \sum_{n=1}^{N} \min_{k=1}^{K} \sum_{m=1}^{M} [\rho_m(x_n, k)]^2 = \sum_{n=1}^{N} d(x_n, k)$$

$$\begin{aligned} D_{total} = \sum_{n=1}^{N} \min_{k=1}^{K} [& \sum_{m \in num.}^{M} ((w_m(x_{nm} - cc_{km}))^2 + \\ & z_{nm} \sum_{m \in cat.}^{M} (\sum_{i=1}^{|u_m|} \frac{cd_{km}(i)}{M-1} \\ & [P_m(\omega/x_{nm}) + P_m((u_m \setminus \omega)/u_m(i)) - 1.0])^2] \end{aligned}$$

where $\omega \subseteq u_m$ such that the quantity $P_m(\omega/x_{nm} + P_m((u_m \setminus \omega)/u_m(i))$ is maximized and z_{nm} is an indicator function such that $z_{nm} = \begin{cases} 0, & x_{nm} = cc_{km} \\ 1, & x_{nm} \neq cc_{km} \end{cases}$.

Since not every clustering is an optimal clustering, we run multiple trials of the clustering algorithm. For some benchmark data sets, there are small deviations in the performance ratio between trials (see section 2). In application, the clustering results are used from the trial with the minimum average silhouette value.

fuel-type	body-style	num-of-cylinders	highway-mpg
gas	convertible	4	27
gas	sedan	6	22
gas	sedan	6	22
diesel	sedan	4	25
diesel	sedan	2	50
diesel	wagon	4	25

TABLE 2.1. Original Auto Data

fuel-type	body-style	num-of-cylinders	highway-mpg
gas	convertible	.5	.1786
gas	sedan	1	0
gas	sedan	1	0
diesel	sedan	.5	.1071
diesel	sedan	0	1
diesel	wagon	.5	.1071

TABLE 2.2. Normalized Auto Data

fuel-type	body-style	num-of-cylinders	highway-mpg
gas	convertible	1	1
gas	sedan	2	2
gas	sedan	2	2
diesel	sedan	1	2
diesel	sedan	3	2
diesel	wagon	1	1

TABLE 2.3. Discretized Auto Data

4. Walk-through with Example Data

Consider the data in Table 2.1, a selection of the benchmark automotive data table [**14**]. The sample size of this data table is too small to make meaningful calculations, but it is used in this section to illustrate the process of the mixed k-means procedure. Each data point has four attributes: two categorical and two numeric.

- Fuel type is a binary categorical variable.
- Body-style has five possible categorical values: *hardtop, wagon, sedan, hatchback, convertible* (although only three values appear in this example).
- The number of cylinders is a discrete numeric variable with meaningful distance and seven possible values: *eight, five,*

four, six, three, twelve, two (although only three values appear in this sample).
- Highway mileage, in miles per gallon, is a continuous variable.

4.1. Normalize the Numeric Data. First, the numeric data is normalized[3] attribute between 0 and 1.[4] This allows us to compare different sets of numeric data with various ranges.

The 'number of cylinders' attribute has three discrete, evenly spaced values, so the scaled result will have values of $\{0, \frac{1}{2}, 1\}$. The range of the highway mileage is 28 miles per gallon, with a minimum value of 22 miles per gallon. Then the normalized entry is the quotient of the difference of the original value and the minimum value between 22 over the range of 28.

$$fuel_{norm}(i) = \frac{fuel(i) - \min(fuel)}{\max(fuel) - \min(fuel)} = \frac{fuel(i) - 22}{28}$$

The normalized data is recorded in Table 2.2.

4.2. Discretize the Numeric Data. To discretize the numeric data, each numeric attribute is run through the numeric k-means clustering algorithm (as a column vector) for an optimal number of clusters κ, where $\kappa < \kappa_{max}$ for some user defined $\kappa_{max} << N$. Since our sample size is very small, we cannot let $\kappa_{max} << N$. For illustrative purposes, we define κ using knowledge of the structure of the data.

The number of cylinders attribute is already discrete, so the value of $\kappa = 3$ gives each unique discrete value its own category. This minimizes the silhouette values from the numeric k-means clustering of the attribute. The values are then replaced with consecutive integers (the output of the numeric k-means algorithm).

The highway mileage is less obvious. Although this sample appears to be discrete, a larger sample would reveal that it is continuous. A choice of κ for this small sample size is largely arbitrary, since κ cannot be much less than the number of data points, N. A more realistic choice of $\kappa = 3$, to limit variance relative to the sample size N, would give a trivial result when finding the significance value in Section 4.3. We continue with $\kappa = 2$ to better illustrate the calculation of significance. In the resulting discretized data there is a larger probability that a data point belongs to a given category.

[3]While this process is not normalizing in the traditional sense, we use this language to be consistent with other current publications on the algorithm.

[4]The minimum value is mapped to 0, the maximum to 1, and all other values are mapped to a proportional distance from 0

4.3. Calculate the Significance of Each Attribute. The significance will place a weight on each of the four attributes, corresponding to the probability of co-occurrence between subsets of unique values. The significance is calculated pairwise between attributes, and then pairwise between unique values within each attribute. The significance will be calculated using the discretized data in Table 2.3.

Consider the attribute 'number of cylinders'. There are three unique values of this attribute, and 3 unique pairs:$(1,2),(1,3),(2,3)$. The vector of unique values of the attribute is defined as $u_{ncyl} = [1,2,3]$.

To calculate the significance of the number of cylinders, we use the formula from Equation 2.1.

$$w_{ncyl} = \frac{2}{\kappa_{ncyl}(\kappa_{ncyl}-1)(M-1)} \sum_{b=1}^{\kappa_{ncyl}} \sum_{c>b}^{\kappa_{ncyl}} \sum_{m=1,m\neq ncyl}^{M} [P_i(\omega/u_{ia}) + P_i((u_m \setminus \omega)/u_{ib}) - 1.0]$$

Recall, $\kappa_{ncyl} = 3$ and $M = 4$.

$$w_{ncyl} = \frac{1}{9} \sum_{b=1}^{3} \sum_{c>b}^{3} \sum_{m=1,m\neq ncyl}^{M} [P_i(\omega/u_{ia}) + P_i((u_m \setminus \omega)/u_{ib}) - 1.0]$$

In the context of the unique pairs of discrete values from u_{ncyl}, the terms $\sum_{b=1}^{3} \sum_{c>b}^{3}$ describe the sum of across all unique pairs of u_{ncyl}

$$\sum_{m=1,m\neq ncyl}^{M} [P_i(\omega/u_{ia}) + P_i((u_m \setminus \omega)/u_{ib}) - 1.0]$$

Recall these unique pairs are $(1,2),(1,3),(2,3)$. Then, the significance of the attribute 'number of cylinders' can be expressed more explicitly, replacing u_{ia} and u_{ib} with the appropriate values of u_{ncyl} as in 2.5.

$$\begin{aligned} w_{ncyl} = \frac{1}{9} \cdot (& \sum_{m=1,m\neq ncyl}^{M} [P_{ncyl}(\omega/1) + P_{ncyl}((u_m \setminus \omega)/2) - 1.0] \\ & + \sum_{m=1,m\neq ncyl}^{M} [P_{ncyl}(\omega/1) + P_{ncyl}((u_m \setminus \omega)/3) - 1.0] \\ & + \sum_{m=1,m\neq ncyl}^{M} [P_{ncyl}(\omega/2) + P_{ncyl}((u_m \setminus \omega)/3) - 1.0]) \end{aligned} \tag{2.5}$$

Consider the first term of 2.5.

$$\sum_{m=1,m\neq ncyl}^{M} [P_{ncyl}(\omega/1) + P_{ncyl}((u_m \setminus \omega)/2) - 1.0] \tag{2.6}$$

This can be expanded to explicitly write the comparison of each attribute in equation 2.7.

$$\begin{aligned} &[P_{ncyl}(\omega_{fuel}/1) + P_{ncyl}((u_{fuel} \setminus \omega_{fuel})/2) - 1.0] \\ &+[P_{ncyl}(\omega_{body}/1) + P_{ncyl}((u_{body} \setminus \omega_{body})/2) - 1.0] \\ &+[P_{ncyl}(\omega_{mpg}/1) + P_{ncyl}((u_{mpg} \setminus \omega_{mpg})/2) - 1.0] \end{aligned} \tag{2.7}$$

The expression for significance is now in a workable state. We now evaluate these using the discretized data in Table 2.3. Consider the first term of 2.7, as listed in 2.8.

$$[P_{ncyl}(\omega_{fuel}/1) + P_{ncyl}((u_{fuel} \setminus \omega_{fuel})/2) - 1.0] \tag{2.8}$$

The first term of 2.8 is $P_{ncyl}(\omega_{fuel}/1)$ which describes the probability that an component with one cylinder has a value that is an component of ω_{fuel}, where ω_{fuel} is a subset of the unique components of the discretized fuel-type attribute. Now, by the definition in Section 2.1, ω_{fuel} is the subset of the unique components of the discretized fuel-type attribute that maximizes the quantity in 2.9.

$$P_{ncyl}(\omega_{fuel}/1) + P_{ncyl}((u_{fuel} \setminus \omega_{fuel})/2) \tag{2.9}$$

Considering Table 2.3, we find the following values of ω and $(u_{fuel} \setminus \omega_{fuel})$.

$$\omega_{fuel} = \{diesel\}$$
$$(u_{fuel} \setminus \omega_{fuel}) = \{gas\}$$

Then the two probabilities can be stated explicitly.

$$P_{ncyl}(\omega_{fuel}/1) = \frac{count(ncyl = 1 \& fuel = diesel)}{count(ncyl = 1)} = \frac{2}{3} \tag{2.10}$$

$$P_{ncyl}((u_{fuel} \setminus \omega_{fuel})/2) = \frac{count(ncyl = 2 \& fuel = gas)}{count(ncyl = 2)} = \frac{2}{3} \tag{2.11}$$

Now, the results of Equation 2.10 and Equation 2.11 can be substituted to find the value of the expression in 2.8.

$$[P_{ncyl}(\omega_{fuel}/1) + P_{ncyl}((u_{fuel} \setminus \omega_{fuel})/2) - 1.0] = \frac{1}{3}$$

This process is iterated across each unique pair and each attribute, with the solutions substituted into 2.5 to find the value of significance

idx	fuel-type	body-style	num-of-cylinders	highway-mpg
1	gas	convertible	.5	.1786
1	gas	sedan	1	0
3	gas	sedan	1	0
2	diesel	sedan	.5	.1071
2	diesel	sedan	0	1
3	diesel	wagon	.5	.1071

TABLE 2.4. Auto Data with Cluster Assignments

idx	fuel-type	body-style	num-of-cylinders	highway-mpg
1	gas	convertible	.5	.1786
1	gas	sedan	1	0

TABLE 2.5. Auto Data - Cluster 1

idx	fuel-type	body-style	num-of-cylinders	highway-mpg
2	diesel	sedan	.5	.1071
2	diesel	sedan	0	1

TABLE 2.6. Auto Data - Cluster 2

idx	fuel-type	body-style	num-of-cylinders	highway-mpg
3	gas	sedan	1	0
3	diesel	wagon	.5	.1071

TABLE 2.7. Auto Data - Cluster 3

of the attribute 'number of cylinders'. These significance values will weight the calculation of distance between each data point and cluster center in Section 4.6.

4.4. Random Assignment of Data Points to Clusters. The user defines K, the number of clusters, as an argument for the mixed k-means clustering. In this example, consider $K = 3$. Each data point is then assigned, at random, to each of the clusters.

Let column 'idx' be the index of the cluster assigned to each data point. Then, a column can be added to the normalized data table for the index. This column is not an additional attribute; it is added to the table for illustrative purposes.

4.5. Calculation of Cluster Centers. Recall, for a numeric attribute, the cluster center is calculated to be the mean of the values of each data point in the cluster.

idx	fuel-type	body-style	num-of-cylinders	highway-mpg
1	gas		.75	.0893
2	diesel	sedan	.25	.5536
3			.75	.5536

TABLE 2.8. Cluster Center Values

Consider the first cluster in Table 2.5. Let cc_1 be a vector that describes the center of the first cluster. Let cc_{1a} be the a^{th} entry of cc_1. Then, the first cluster center value for the attribute 'number of cylinders' is $cc_{13} = mean(\{.5, 1\}) = .75$ and the value for the attribute 'highway mileage' is $cc_{14} = mean(\{.1786, 0\}) = .0893$.

Now, for categorical attributes, the cluster center is defined as the mode, (although the probability of an equivalent value occurring within the cluster is also used to compute the distance between a given point and a cluster center). Clearly, the first cluster center value for the attribute 'fuel type' is $cc_{11} = gas$. However, for the attribute 'body-style' cc_{12} there is no unique mode. The cluster center is left undefined for this attribute, and all calculation of distance will be from the probability of an equivalent value occurring within the cluster. While possible, this is not a likely scenario for clusters in a data set with a sufficiently large number of data points. We continue with the small sample size for illustrative purposes.

The probability of an equivalent value occurring within the cluster is represented as a ratio $cd_{km}(i)$ with respect to each unique value of the categorical attribute in Equation 2.12, where ϕ_k is the set of data points, x_n such that x_n is in the k^{th} cluster and ϕ_{ki} is the set of data points in ϕ_k such that the m^{th} component of the data point has the value of u_{mi}.

$$cd_{km}(i) = \frac{|\phi_{ki}|}{|\phi_k|} \tag{2.12}$$

This process is iterated for each of the three clusters, finding the cluster centers reported in Table 2.8.

4.6. Calculation of the Distance to Each Cluster Center. Consider the distance between the data point in the first row, x_1, and the first cluster center from 2.8. For a categorical attribute, the distance is zero if the value of the data point is equal to the mode of the cluster, otherwise, the distance is computed as in 2.3. For a numeric attribute, the distance between a data point and a cluster is simply the Euclidean distance, as in 2.2. The total distance is the sum of the squares of each numeric and categorical distance, as in 2.4.

Since the fuel type of the first data point and first cluster's mode are equivalent,

$$\rho_{fuel}(x_1, 1) = 0$$

For the attribute body style, there is no defined center value. This value will be calculated from the proportion of equivalent values, as the value of the data point is not equal to the value of the cluster center.

$$\rho_{body}(x_1, 1) = \sum_{i=1}^{|u_m|} \frac{cd_{km}(i)}{M-1}[P_{body}(\omega/x_{nm}(i) + P_{body}((u_m \setminus \omega)/u_m(i)) - 1.0]$$

$$\rho_{body}(x_1, 1) = \sum_{i=1}^{2} \frac{cd_{km}(i)}{3}(P_{body}(\omega/x_{nm}(i)) + P_{body}((u_{body} \setminus \omega)/u_m(i)))$$

Writing the sum explicitly, we have the following expression for $\rho_{body}(x_1, 1)$. Since we have two unique values and two data points, this computation is simple. Additionally, this illustrates the problem with having too many unique values of numeric attributes that was resolved in discretization.[5]

$$\begin{aligned}\rho_{body}(x_1, 1) = cd_{convt}(&P_{body}(\{convert\}/\{convert\}) + \\ &P_{body}(\{sedan\}/\{sedan\}) - 1.0) + \\ cd_{sedan}(&P_{body}(\{sedan\}/\{sedan\}) + \\ &P_{body}(\{convert\}/\{convert\}) - 1.0)\end{aligned}$$

$$\rho_{body}(x_1, 1) = .5 * 1.0 + .5 * 1.0 = 1.0;$$

The attributes number of cylinders and highway mileage are both numeric, and use the Euclidean distance.

$$\rho_{ncyl}(x_1, 1) = (.75 - .5) = .25$$

$$\rho_{mpg}(x_1, 1) = (.0893 -, 1786) = -.0893$$

The total distance is then the sum of the squares of the distance for each attribute.

[5]In a more realistic example, this expression might not include such trivial probabilities. For example,

$$\begin{aligned}\rho_{body}(x_1, 1) = cd_{convt}(&P_{body}(\{convt\}/\{convt, truck\}) + \\ &P_{body}(\{sedan\}/\{sedan, wagon\}) - 1.0) + \\ cd_{sedan}(&P_{body}(\{sedan\}/\{sedan, truck\}) + \\ &P_{body}(\{convert\}/\{convert, wagon\}) - 1.0)\end{aligned}$$

$$d(x_1, 1) = \sum_{m=1}^{M} [\rho_m(1, x_1)]^2 = 0^2 + 1.0^2 + .25^2 + .0893^2 \approx 1.07$$

This process is iterated for each unique pair of cluster and data point. When we have calculated the distance between each data point and each cluster center, we will be able to determine which cluster center is closest to each data point.

4.7. Cluster Reassignment for Each Data Point. Each data point x_n is reassigned to the cluster k that minimizes $d(k, x_n)$. In the case of x_1, the cluster with index $k = 1$ has the minimum distance $d(x_1, 1)$ as calculated in 4.6. This process of reassigning data points to the closest cluster is iterated for every data point in the data set.

4.8. Loop Conditions and Iteration. In the first iteration of this example, x_3 is reassigned to the first cluster. Since at least one data point has changed cluster assignment, the process is repeated from Section 4.5.

If there are no changes in cluster assignment, the clustering algorithm returns the last iteration's cluster assignments.

CHAPTER 3

Error Analysis

In Chapter 1 and Chapter 2, two algorithms for clustering data are introduced. In this chapter, we will discuss the methods for analyzing the clustering produced by these two algorithms. This discussion is applicable to comparisons between the algorithms and between multiple trials with the same algorithm.

Three processes for error analysis are included in the source to determine the quality of the mixed k-means clustering, especially relative to numeric k-means. Silhouette Values are adapted from their application to numeric k-means [**13**] as both a graphical and numeric representation of the fit of a clustering model. Performance ratios are adapted from Alsahaf's MATLAB implementation [**2**] to compare the result of numeric and mixed k-means clustering on benchmark data sets from SGI [**14**], where the structure of the data is known in an additional attribute, hidden from the clustering algorithm. Finally, two new graphical representations are proposed. The first displays the distance between a given data point and each of the cluster centers, in $\mathbb{R}^2$. The second builds a geometric figure in $\mathbb{R}^3$, where planes on integer coordinates are defined by the $\mathbb{R}^2$ representation for each data point.

1. Silhouette Values

The silhouette value is a measure of the goodness of fit of a model for a particular data point. In both numeric and mixed k-means clustering, the distance between a data point and each of the cluster centers can be calculated. The silhouette value compares the distance to the two cluster centers with the least distance, as the ratio of their difference divided by their maximum. The mean of all silhouette values in a data set is a measure of the goodness of fit across the data set. Assuming a natural structure of M-dimensional spherical partitions of the data set, the number of clusters that maximizes the average silhouette value is considered to be the optimal number of clusters for the data set.

1.1. Definition of Silhouette Value. The silhouette value for a data point x_n within the k^{th} cluster (with center cc_k) is defined in

No. of Centers	Mean Silhouette Value
2.0000	0.6942
3.0000	0.6883
4.0000	0.7090
5.0000	0.8917
6.0000	0.8078

TABLE 3.1. mean Silhouette Values for Synthetic Data

equation 3.1. Let cc_z be the second closest cluster center to x_n. That is, let $cc_z \in \{cc_\zeta, 1 < \zeta < K, \zeta \neq k\}$ that minimizes distance $d(x_n, z)$.

Let $a_n = d(x_n, k)$ be the distance between x_n and cc_k, as defined in Equation 2.4. Let $b_n = d(x_n, z)$ be the distance between x_n and cc_z, as defined in Equation 2.4.

$$s_n = \frac{b_n - a_n}{\max(a_n, b_n)} \tag{3.1}$$

Here, the optimal silhouette value is 1, where the data point is at the cluster center. If the silhouette value is 0, the data point is equidistant from the two closest cluster centers. If the silhouette value is negative, there is a cluster center closer to the data point than its assigned cluster's center.

In both k-means algorithms, we choose the number of clusters that minimizes the difference between the mean silhouette value and 1. That is, the number of clusters that maximizes the mean silhouette value.

1.2. Example with Synthetic Data. We have generated a set of synthetic data with two numeric attributes, and five spherical clusters, and 2,400 data points. Table 3.1 lists the mean silhouette value for each trial of numeric k-means clustering on the synthetic data set.

First, we cluster using $k = 2$ (two centers or means) for the numeric k-means algorithm. Figure 3.1 includes a scatter plot of the synthetic data, coloring each data point by its cluster assignment.

The silhouette values for this clustering can now be visualized with a bar graph in Figure 3.1. In this graph, there are many silhouette values that are significantly less than one. This suggests that there may be another value of k that will better fit the data.

Consider the assignment with $k = 3$ (3 distinct clusters). The scatter plot in Figure 3.2 colors each data point according to its clustering assignment. Here it is more obvious that the clustering assignment

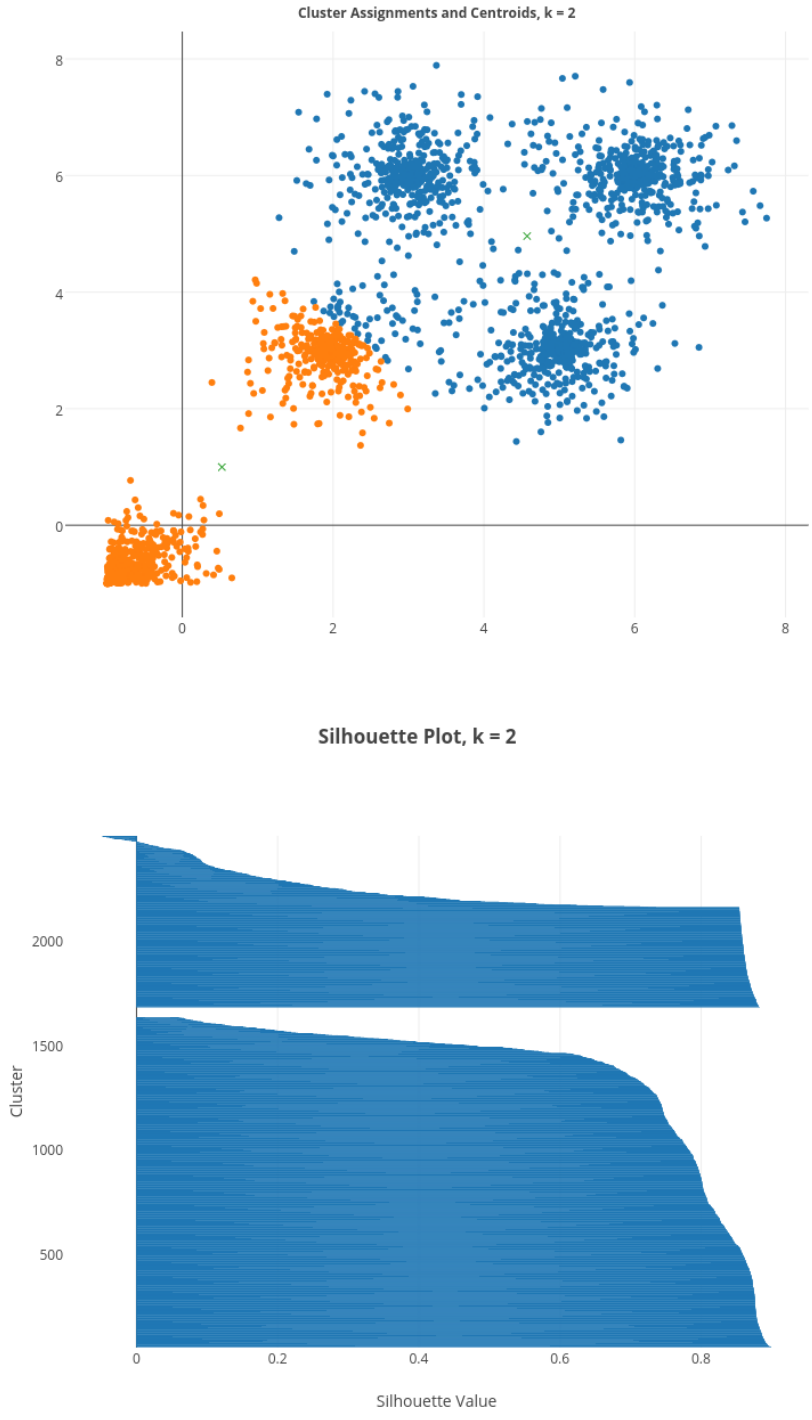

FIGURE 3.1. Scatter Plot & Silhouette Graph, k=2

does not fit the data set. Figure 3.2 confirms the graphical observation, with a large portion of data points with silhouette values near zero in the second cluster.

For $k = 4$ (4 clusters), we have an improved result. Figure 3.3 show the scatter plot and silhouette graph. This has a reasonable mean silhouette graph value, and most clusters have consistently high silhouette values.

When $k = 5$ (the actual number of clusters in the synthetic data set), we have a silhouette graph (Figure 3.4) similar to the $k = 4$ model, but the mean silhouette value is significantly greater. This improvement in fit can be seen clearly in the scatter plot in Figure 3.4, where each cluster center is clearly at the center of a cluster of data points (rather than equidistant to each cluster center. This observation is clear in a scatter plot in $\mathbb{R}^2$, but is not easy to visualize with a greater number of attributes or with categorical attributes.

When a sixth cluster is introduced ($k = 6$), we find that one cluster center again lies between other clusters. The points in the cluster are generally denser at a larger radius from the center of the cluster, with very few points near the center of the cluster (Figure 3.5). While this cluster center is closer to some points, the mean silhouette value has decreased, and there is one cluster where the majority of data points have silhouette values less than 0.7.

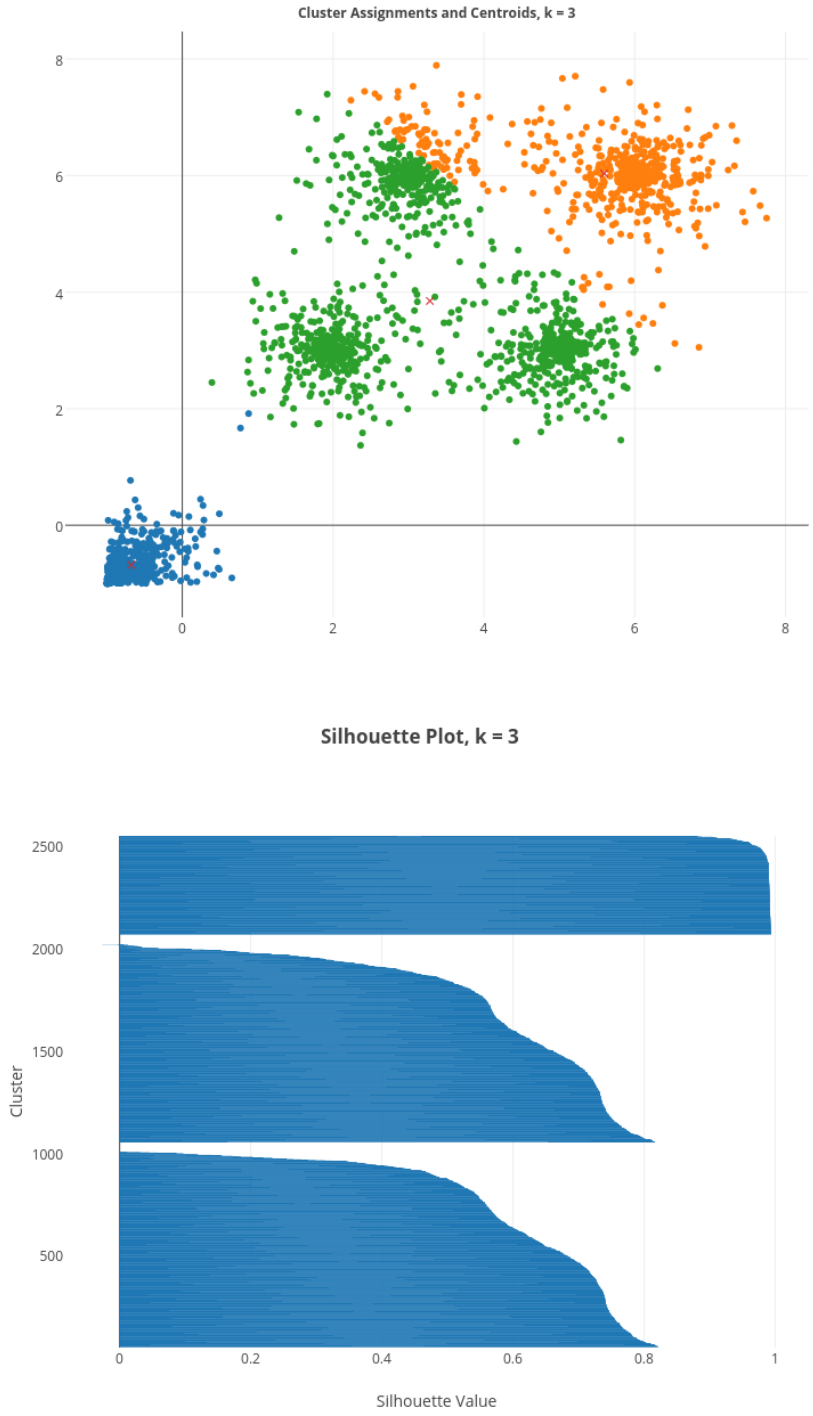

FIGURE 3.2. Scatter Plot & Silhouette Graph, k=3

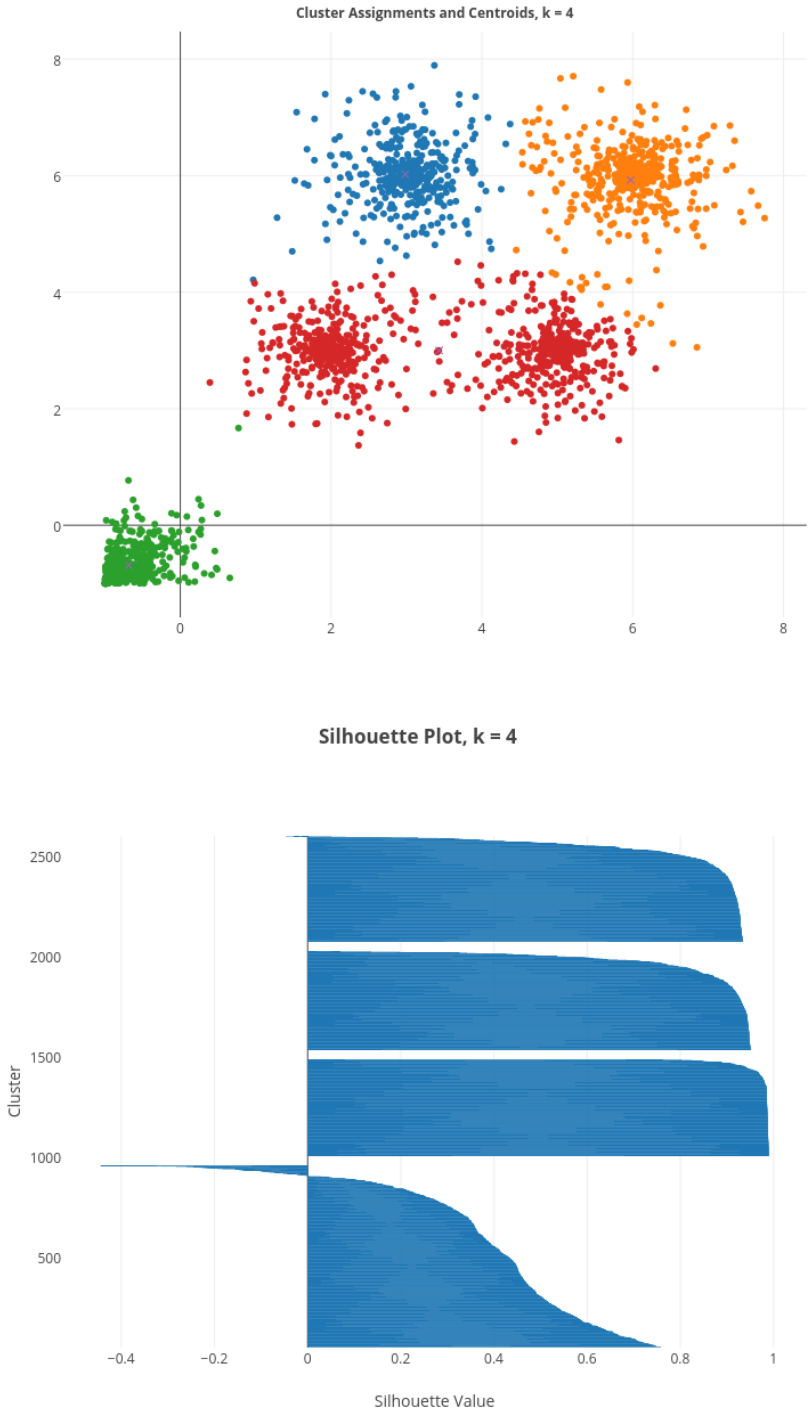

FIGURE 3.3. Scatter Plot & Silhouette Graph, k=4

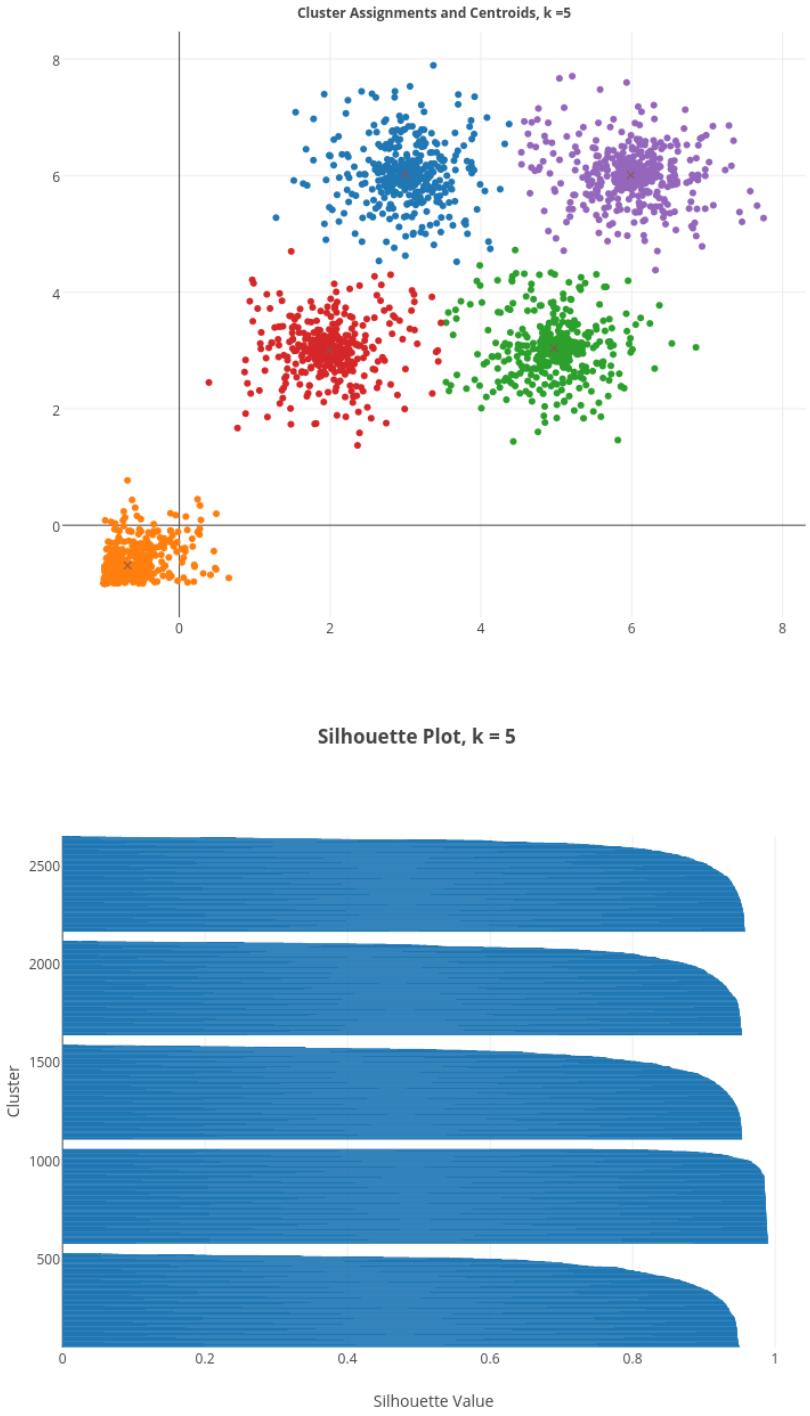

FIGURE 3.4. Scatter Plot & Silhouette Graph, k=5

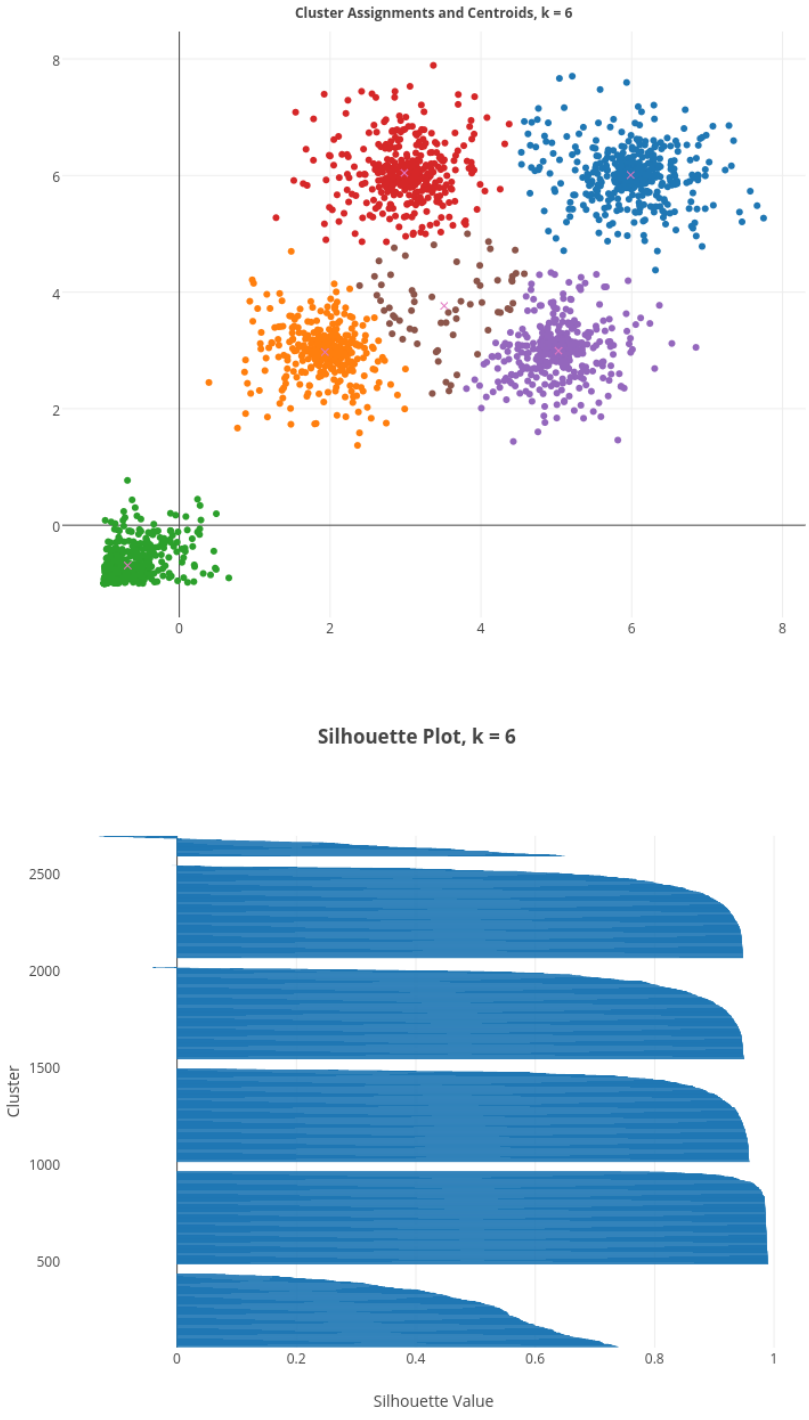

FIGURE 3.5. Scatter Plot & Silhouette Graph, k=6

2. Performance Ratios

The performance ratio is a comparison between the clustering produced by the algorithm and the "true" categorization for benchmark data sets. This ratio can be compared with other unsupervised clustering algorithms as a measure of accuracy. The performance ratio is not used in application, where the true categorization of data is unknown.

The performance ratio is defined as the maximum percentage of common assignments between a permutation of the clustering assignments and the"true" categorization, as defined in 3.2. Let x_n be the n^{th} data point. Let cc_k be the k^{th} cluster center.

Let Υ be a $k! \times k$ permutation matrix where each row of Υ is a unique permutation of the indexes of the clustering assignments. Let $\Upsilon_{p,k}$ be the p^{th} row of the k^{th} column of Υ. Let θ_n be the assigned index of x_n in the clustering. Let ϕ_n be the assigned index of x_n in the the"true" categorization. Let γ_n be an index function, to denote a common assignment between the permutation of the clustering and the "true" categorization.

$$\gamma_n = \begin{cases} 1 & \text{if } \Upsilon_{p,theta_n} = \phi_n \\ 0 & \text{if } \Upsilon_{p,\theta_n} \neq \phi_n \end{cases}$$

Then R, the performance ratio, can be defined as average of the values of the indicator function γ, for the permutation that maximizes this sum.

$$R = \max_{p=1}^{K!}\left(\frac{1}{N} \cdot \sum_{n=1}^{N} \gamma_n\right) \tag{3.2}$$

3. Visualization of Clustering

When there are many attributes, data points, and clusters it becomes difficult to visualize the location of data points in relation to their cluster centers. With three numeric attributes, data points and cluster centers can be visualized in $\mathbb{R}^3$. With additional numeric attributes, locations can be projected into fewer dimensions to give a visual representation. These relationships in $\mathbb{R}^3$ are not well defined for categorical attributes, especially categorical attributes with many unique values.

We propose a visualization that compares the distances between each data point and each cluster center. Figure 3.6 is a visualization of mixed k-means clustering in both $\mathbb{R}^2$ and $\mathbb{R}^3$, using the 'lenses' data set from Chapter 5 Section 5.

This visualization is not used formally in our analysis, but is useful for building intuition in certain concepts around error analysis. They can be used to demonstrate some properties of clustering algorithms on simple synthetic data sets. We do not have evidence of the usefulness of theses figures or their geometric properties in application to larger data sets.

A data point can be represented in $\mathbb{R}^2$. The visual representation of data point x_n is defined as a polygon with verticies in the set V_n, as defined in Equation 3.3. Let $d(x_n, k)$ be the distance from the data point x_n to the k^{th} cluster center. Let K be the total number of clusters. We use polar coordinates here for convenience.

$$V_2 = \bigcup_{n=1}^{N} \{(r, \theta) | r = d(x_n, k) \text{ and } \theta = \frac{2\pi k}{K}\} \tag{3.3}$$

where $1 < n < N$ and $1 < k < K$.

A cluster of data points or an entire data set can be represented in $\mathbb{R}^3$. We define a polygon in $\mathbb{R}^3$, by the set of points V_3 in equation 3.4. First, we assign each of the data points an height h_n, using the following procedure.

(1) Let $k = 1$.
    (a) Order the data points in cluster k by their row number.
    (b) Assign each data point x_n in cluster k a consecutive integer value h_n, where the least value of h_n is 1.
    (c) Let $k = k + 1$.
(2) While $k < K$ do
    (a) Let ξ be the greatest value of h_n assigned in cluster $(k-1)$.
    (b) Order the data points in cluster k by their row number.
    (c) Assign each data point x_n in cluster k a consecutive integer value h_n, where the least value of h_n is $\xi + 1$. Let $k = k + 1$.

Now, we can define V_3, using cylindrical coordinates for convenience.

$$V_3 = \bigcup_{n=1}^{N} \{(r, \theta, h) | r = d(x_n, k) \text{ and } \theta = \frac{2\pi k}{K} \text{ and } h = h_n\} \tag{3.4}$$

where $1 < n < N$ and $1 < k < K$.

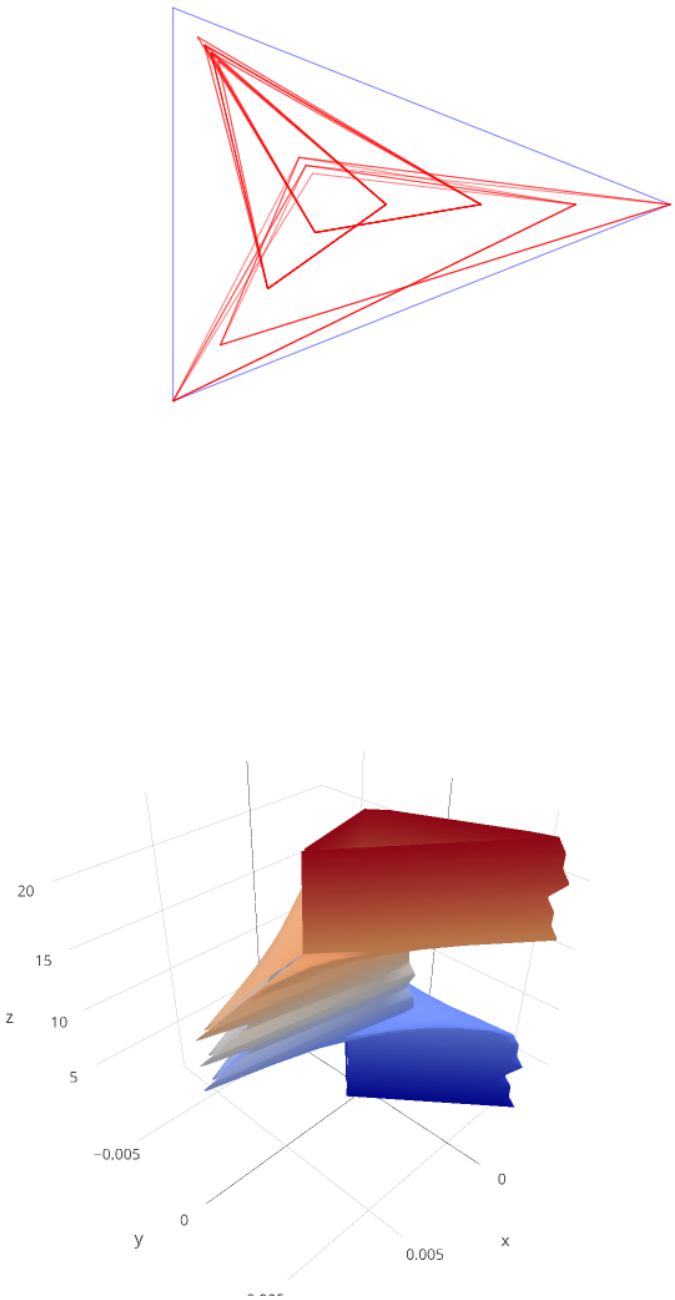

FIGURE 3.6. Visualization of mixed clustering on lenses data set

CHAPTER 4

MATLAB Implementation

Chapters 2 discussed an algorithm for clustering mixed data sets. Chapter 3 discussed methods for analyzing clustering algorithms. In this chapter, the challenges of implementing both of these chapters are discussed through a series of examples. The source code for all of these implementations is included in Appendixes B, C and D.[1]

This chapter highlights four adaptations of our implementation from Alsahaf's MATLAB implementation [**2**] and Ahmad's algorithm [**1**]. This selection focuses on the challenges that required the most creative solutions or problems that assisted in the understanding of the function of the algorithm. First, Section 1 describes adaptations that improve the overall efficiency of the implementation and the process of debugging. Second, Section 2 discusses an adaptation that deviates from both Alsahaf's implementation and Ahmad's original algorithm to avoid assumptions about the structure of numeric attributes. Section 3 discusses efforts to debug unintended implications of the previous section's adaptive value for κ_i. Finally, Section 4 addresses the need for an efficient calculation of error for increasingly complex benchmark data sets. This chapter gives a brief overview of the type of work required for writing and adapting the source code which composes the bulk of the thesis.

1. Object Oriented Programming

The original MATLAB implementation written by Alsahaf [**2**] first needed to be generalized to accept a generic $n \times m$ matrix and labels to designate categorical attributes. Once generalized, the program was inefficient when running hundreds of trials on benchmark data sets. Across each of these trials, the values for the discretized data and significance were the same, but they were re-computed for every trial on the data set. The program was first adapted to move normalization, discretization and significance calculations outside of the loop that iterates trials.

[1] A current verison of the source code is available at https://github.com/cam3715/mkmeans

```
methods
    function obj = mixedclust(data, k, max_iter, ...
        inputType,trialsNo)

        [dn, ¬] = size(data);
        [¬, ¬] = size(inputType);
        if nargin < 4
            trialsNo = 1;
            inputType = [];
        elseif nargin < 3
            max_iter = 1000;
        elseif nargin < 2
            display('Not Enough Arguments')
        end
        obj.tempvar.dn = dn;
        obj.trialsNo = trialsNo;
        obj.data = data;
        obj.k = k;
        obj.max_iter = max_iter;
        obj.inputType = inputType;

        % replace NaN entrieies
        obj.data(isnan(obj.data)) = 1;
        obj = normalize(obj);
        obj = discretize(obj);
        obj = sigpairs(obj);
```

FIGURE 4.1. Objects in mixedclust.m

This increased efficiency, but the problem of reducing runtime when debugging errors when testing benchmark categorical data sets remained. Some errors would appear after a large number of trials, which seemed to depend on the random seed for the trial (see Section 3). A *try-catch* statement block[2] was added within the loop, where the trial would be discounted and repeated if any error appeared. This

[2]a try-catch statement block gives instructions if an error is encountered during the specified section of code. For example, the conditional instructions might be *print 'Error in Section 5a'*. Then the system would print 'Error in Section 5a' if and only if an error is encountered in the section. Try-catch statement blocks can be used in this way to flag errors, which is useful for debugging, but they can also be used to develop temporary workarounds for known bugs.

workaround made it possible to find preliminary benchmark results, but did not identify the underlying errors in the algorithm or source code.

To minimize repeated computation when debugging, the program was rewritten to pass an object that included the original and discretized data matrices and other temporary variables. First, Alsahaf's implementation was rewritten to minimize the number of sub-functions and to contain those sub-functions within a single m-file. Second, a function was created to handle data input from a comma separated values (CSV) file and another function to handle input/output and the user interface for the clustering algorithm. The serial program was then rewritten to input and output structs[3], and finally those structs were used as properties of the objects. This required that many of the dependencies written by Alsahaf be retrofitted to be passed objects instead of vector arguments, which enabled additional sections of the code to be adapted for parallel processing but increased overhead. Finally, the data input and data output functions were rewritten as classes, which made it easier to output and graphically display various results for a sequence of benchmark tests.

The result is not a true object oriented design. The original serial implementation was adapted to operate on objects instead of passing large structs (as shown in Figure 4.1). That is, the MATLAB source is an objected oriented program, but it does not necessarily follow object oriented design principles. This could be improved by rewriting the entire program with object oriented design in mind. It would be optimal to redesign the implementation for true object oriented design when simultaneously vectorizing the code for CUDA optimization and parallel processing. Additionally, a handle class should be used for many of the sub-functions rather than a value class (as it is currently) which would diminish the overhead and bottlenecks during parallel processing.

2. Discretization: Adaptive Number of Means

The adaptive number of means for the discretization is the most significant deviation from Ahmad's algorithm. Both Ahmad and Alsahaf use constant values of k_i for all numeric attributes [**1**]. This

[3]A struct is a group of variables that can be accessed through a structure tag, using a member access operator. For example, the variable *idx* might have the tag *mixedclust*. In MATLAB, the member access operator is '.', so *idx* is accessed as *mixedclust.idx*. Variables with the same tag are physically grouped together in memory.

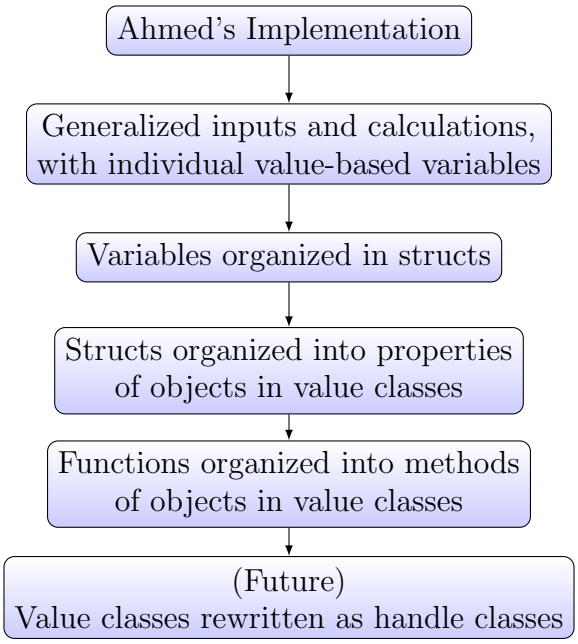

FIGURE 4.2. Use of variables in each version of the MATLAB source

assumes that every numeric attribute has the same structure, which is not a reasonable assumption for many complex data sets. Additionally, this assumes that the structure of the numeric attributes of a data set can be determined by the user's observation or some other simple test. Clustering is used when there is much unknown about the underlying structure of the data, and it is often impractical to spend much time investigating the structure of each attribute. This is especially true in the case of our application to ASSISTments, where there are over one million data points (responses) in the smaller of the two data sets.

The procedure for this adaptation is described in Chapter 2, Section 1. For limitations of computational time, this implementation samples a user defined range of values of κ_i for each numeric attribute A_i, where $2 \leq \kappa_i << N$. The silhouette value of each potential value of κ_i is stored, where value of κ_i minimizes the difference between 1 and the respective silhouette value. An additional conditional, described in Section 4.4, further restricts this value of κ_i to avoid a trivial result (which causes bugs in the calculation of significance).

This impact of this adaptive value of κ_i has not been fully explored. It is included in our implementation to avoid the use of a constant value of κ_i. Future research may indicate a more efficient method of selection for κ_i or a method for a selection of κ_i that improves the clustering

```
obj.tempvar.idx_cat = find(-1*obj.inputType+1);
for i=1:numel(obj.tempvar.idx_cat)
    silh_avg = zeros(max_k,1);
    data_num = ...
       obj.data(:,obj.tempvar.idx_cat(i));
    for k_iter=1:max_k

        [idx,¬,¬,D]=...
            kmeans(data_num,k_iter+1,'dist', ...
                ...
            'sqeuclidean','MaxIter',100,...
            'Options',statset('UseParallel',1));
        [Drow,¬] = size(D);
        silh = zeros(1,Drow);
        for drow = 1:Drow
            [a_drow,excl_D] = min(D(drow,:));
            b_drow = ...
               min(D(drow,[1:(excl_D-1),...
                (excl_D+1):end]));
            silh(drow) = (b_drow-a_drow)...
                /max(a_drow,b_drow);
        end
```

FIGURE 4.3. Discretization in mixedclust.m

results and better reflects the underlying structure of the data. The emphasis of this modification is not the particular method of selection of κ_i, but the use of an adaptive value.

3. Hidden Bugs: Unique Values After Discretization

The bug that persisted longest in the source turned out to be a lack of unique values in a discretized numeric attribute; the bug took over 4 months to resolve. This bug presented itself with error messages nested about 4-5 function calls deep, in MATLAB's *nchoosek*, which outputs a matrix of unique pairs (n=2) of k unique values (in this context k is not the number of clusters). If $k = 1$, *nchoosek* returns an error, however the error notes that 'n must be an integer'. Consistently, n was input as an integer, which could be confirmed by MATLAB's debugger.

The $try - catch$ statement block was successful as a temporary workaround for this bug, as a different random seed for the k-means in the discretization function usually returned more unique values. When

```
        %Ensure selection has >1 unique value
        if numel(unique(idx))>1
            silh_avg(k_iter) = mean(silh);
        else
            silh_avg(k_iter) = -10000000;
        end

    end
    [¬,k_best] = max(silh_avg);
    k_best = k_best+1;
    obj.data_discrete(:,obj.tempvar.idx_cat(i)) ...
       = ...
        kmeans(obj.data(:,obj.tempvar.idx_cat(i)),...
        k_best);
```

FIGURE 4.4. Unique Values after discretization in mixedclust.m

rewriting the source to pass objects, and breaking the clustering algorithm into multiple method functions, it became clear that the error 'n must be an integer' was also reported when k is not an integer or $k < 2$. A conditional was added to the discretization method (lines $164 - 169$ of *mixedclust.m* in Figure 4.4) that omits values of κ_i that output a single cluster. This conditional replaces the recorded average silhouette value with a number significantly less than -1 so that these values of κ_i will not be selected for the discretization of the attribute.

An additional conditional was added to give a more descriptive error message when n choose k is called and $k < 2$. An additional case of single unique values may occur during computations of distance on a cluster with a subset of discretized values, (although for any data point within the cluster, its value would be equal to the mode, and the distance would be defined as 0).

4. Performance Ratios and the Hungarian Algorithm

Performance ratios were a challenge in early benchmark tests. The performance ratio compares permutations of the output to the 'known' classification of the data for benchmark sets. Since the output varies by the initial random seed (that first assigns data points to clusters), distinct trials may result in equivalent permutations of the output but have significant differences in output labels.

Alsahaf's implementation [**2**] worked with binary permutations. This was first adapted to handle more unique values in the output

```
for i=1:obj.trialsNo
    idx = obj.(name).idx(:,:,i);
    k = numel(unique(obj.output));
    ErrorMatrix = zeros(k);
    output_values = unique(obj.output);
    for emCol = 1:k
        for emRow = 1:k
            output_emRow = ...
               output_values(emRow);
            for oRow = 1:length(obj.output)
                if (obj.output(oRow) ≠ ...
                   output_emRow)...
                        && (idx(oRow) ...
                           == emCol);
                    ErrorMatrix(emCol, ...
                       emRow)...
                        = ...
                           ErrorMatrix(emCol,emRow)+1;
                end
            end
        end
    end
    [mEM, nEM] = size(ErrorMatrix);
    if mEM≠nEM
        display('Warning, matrix must ...
           be square');
    end
    [¬, count] = ...
       assignmentoptimal(ErrorMatrix);
    obj.(name).performance(i) = ...
       1-count/length(idx);
end
```

FIGURE 4.5. Performance ratios in clusteringCompare.m

using MATLAB's *perms* function. If a data point x_n is assigned to cluster k, the n^{th} output value is k. Using *perms* the value k in the output vector would be substitutes for the k^{th} term of $perms(1 : numel(unique(output)))$. In serial applications, before the introduction of MATLAB's Parallel Processing Toolbox, this was computationally expensive for large numbers of unique values of the output. Additionally, before upgrading RAM storage, MATLAB was unable to store the

permutation matrix when $K \geq 11$, where K is the number of clusters and the number of unique output values.

In an effort to minimize the computational time an 'error matrix' was created, where each column index represented the original output value and each row index was a substituted value, of the same set of values. In the error matrix, the entry $EM(row, col)$ is the count of data points that were clustered into the row^{th} cluster that are not equal to the col^{th} value of the known classification.

The permutation that minimizes the error will have the minimal sum of K elements of EM, where no two elements in the sum have the same row or column index. In other words, the permutation will substitute the value of each pair of column and row indicies in the minimal sum those K elements of EM. Initially this sum was found with a brute force method, which was did not scale significantly better than the *perms* function. In a personal correspondence, Andonian suggested the use of the Hungarian algorithm as a computationally efficient method for minimizing these sums[**3**]. The source was then adapted to take the output of Buehren's implementation[**7**] of the Hungarian Algorithm, and determine the performance ratio (as shown in figure 4.5).

This method using the Hungarian algorithm and the 'error matrix' was confirmed to find equal performance ratios to the *perms* substitution method, over many trials of multiple benchmark data sets.

CHAPTER 5

Results of MATLAB Implementation

Chapter 4 discussed the MATLAB implementation of the mixed k-means algorithm (introduced in chapter 2). In this chapter, the results of this implementation are analyzed using the methods discussed in Chapter 3.

In this chapter we compare the results from our MATLAB implementation of the mixed k-means algorithm with the results from the numeric k-means algorithm and the results published by Ahmad et al., to test the algorithm we clustered five benchmark data set with 'known' classifications. The clustering algorithm performed well on four of the five data sets, with higher performance ratios than the numeric k-means clustering algorithm. One of the data sets is known to perform well for hierarchical clustering algorithms and perform poorly for variations of the k-means clustering algorithm. Our implementation of the mixed k-means clustering algorithm performs poorly here as well, possibly due to the assumption of a m-dimensional spherically structured data set.

The clusteringCompare function in Appendix C was run on five benchmark data sets to compare the accuracy of this implementation with Ahmad's results. Each of the data sets was obtained from Silicon Graphics International (SGI) [**14**], where the attribute classification (as categorical or numeric) was listed. Table 5.1 summarizes the results of the benchmark tests and detailed outputs are included in Appendix A.

Data Set	Mixed		Numeric		Ahmad
	10 Trials	100 Trials	10 Trials	100 Trials	
Iris	88.6%	85.4%	82.3%	84.2%	95%
Vote	87.3%	87.3%	86.9%	85.3%	87%
Lenses	58.3%	51.4%	46.5%	48.8%	
Heart	84.0%	84.0%	76.3%	75.6%	83%
Australian	78.2%	75.3%	61.2%	59.2%	85%

TABLE 5.1. Benchmark average performance ratio

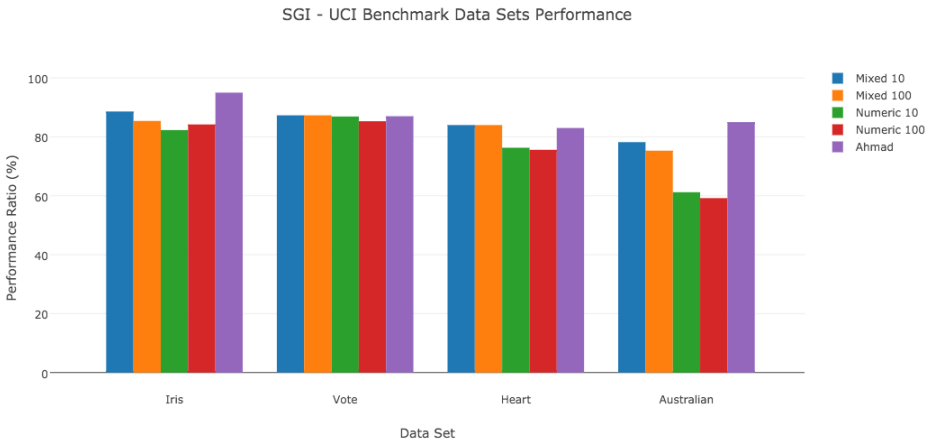

FIGURE 5.1. Benchmark average performance ratio

1. Iris Plants - Numeric Data Set

The Iris data set demonstrates that the mixed k-means algorithm will perform at least as well as the numeric k-means algorithm, on numeric data sets. The mixed k-means algorithm is almost identical to the numeric k-means algorithm for numeric data sets, with the exception of significance weights placed on each attribute and the normalization (scaling) within each attribute. In this case, the use of the significance improves average performance by 1.2% — a small but consistent gain.

The Iris plants data set has four continuous attributes describing sepal length, sepal width, petal length and petal width for three classifications of Iris plants. There is an equal count of each class of Iris plant in the data set.

According to SGI, one of the classes is linearly separable, but the remaining classes are not linearly separable [**14**]. That is, a convex region can be defined such that all of the points of one class fall in that convex region, and no points from the other classes fall in the convex region. Algebraically, there exist weights w_1, w_2, w_3, w_4 and constant q such that for all data points x_n within the linearly separable class, we have

$$\sum_{m=1}^{4} w_m x_{nm} < q.$$

For all data points x_n not in the linearly separable class, we have

$$\sum_{m=1}^{4} w_m x_{nm} \geq q.$$

In the examples with synthetic data in Chapter 3 Section 1, the cluster in the lower left portion of the scatter plots is linearly separable from all of the other clusters.

Ahmad et al. report gains of 7% on the Iris data set [**1**] with their implementation of the mixed k-means algorithm (using a fixed value of κ in the discretization of each numeric attribute).

2. Voting Records - Categorical Data Set

The Voting Records data set demonstrates that the mixed k-means algorithm performs as well as Ahmed and the numeric k-means algorithm on data sets with purely categorical data. Numeric k-means is applied using an integer representation of categorical labels, although this does not give a meaningful measure of rank or distance. The numeric k-means algorithm was not expected to perform well on this data set, and would possibly perform differently on other categorical data sets that have more unique values (a larger range of integer labels).

The Voting Records data set reports 16 votes for U.S. House of Representatives Congressmen from the Congressional Quarterly Almanac of 1985 [**14**]. Numeric k-means performs better than expected on this categorical data set. This performance is assisted by two factors, the distribution of attribute values and the binary output.

This data set has sixteen categorical variables, and two classes: Republican and Democrat. Each vote is listed as an attribute with a value (disposition) of yea, nay, unknown. The majority of the values of each attribute are either yea or nay. The undecided dispositions range from 0% to 23.9%, with an average of just 5% of the data points in the Voting Records data set. Thus the behavior of most attributes approaches that of a binary categorical attribute. Since binary attributes can be represented with meaningful numeric measures of distance and rank, most attributes are handled well by the numeric k-means algorithm.

Finally, the binary classification increases the chance that a given clustering assignment will match the "true" classification.

Ahmad et al. report a performance ratio of 87% [**1**], without a comparison to the numeric k-means algorithm. Over 100 trials, our average performance ratio confirmed Ahmad et al.'s result with 87.3%, a 2.0% gain over numeric k-means.

3. Heart Disease - Mixed Data Set

The Heart Disease data set shows improvement over the numeric k-means algorithm with a mix of categorical and numeric attributes. The heart disease data set from SGI has a mix of 13 categorical and numeric attributes across 270 data points. Five attributes of 13 have categorical values, and two numeric attributes have discrete values [**14**]. This data set is a realistic application of the mixed k-means algorithm, with attributes that have a good mix of categorical, discrete and continuous values.

Confirming Ahmed's result, we find a performance ratio of 84% over 100 trials, a gain of 8.4% over numeric k-means. Ahmed et al. reported an average performance ratio of 83.0% over 100 trials [**1**].

4. Australian Credit Approval - Mixed Data Set

The Australian credit approval data set shows improvement over the numeric k-means algorithm with a mix of categorical and numeric attributes. The Australian credit approval data set has 14 attributes, 8 categorical and 6 numeric. The number of unique values in categorical variables ranges from 2 to 14. Missing values have been replaced with the mean of the attribute [**14**], so we have no NaN values[1]. This is another realistic application of the mixed k-means algorithm.

We perform significantly below Ahmad's implementation on the Australian Credit Approval data set, with a performance ratio of 75.3% compared to Ahmad's performance ratio of 85% [**1**]. This may be due to the selection of the value of κ for the discretization of numeric attributes and calculation of significance. Rather than using an adaptive value of κ_i, Ahmad et al., may have improved their clustering by letting the user define κ_i with some knowledge of the structure of the data. Alternatively, the result of a performance ratio of 85% could be the performance of a best trial rather than the average over many trials. We were not able to replicate an average performance ratio of 85% with a fixed value for κ. We still have a 16.1% gain over the numeric k-means algorithm, which has a performance ratio of only 59.2%.

5. Limitations: Fitting Contact Lenses

This data set compares four attributes to the type of contact lenses that a patient is prescribed. This data set does not have the assumed

[1]A NaN value is a value that is not a number. In some cases this includes values of infinity, but for the purposes of this thesis, NaN values are 'char' (character) values or any other non-numeric representation of the data.

m-dimensional spherical structure. Instead, it is known that this data set performs well with hierarchical clustering algorithms that iteratively sub-partition clusters [**14**]. Here, the mixed k-means algorithm outperforms the numeric k-means algorithm (as expected), but does not perform well enough for use in practice.

The attributes are all categorical, where the attribute age has three possible values and the remaining attributes (spectacle prescription, astigmatic, and tear production rate) have binary values. These attributes behave similarly to the Voting Records data set in Section 2.

This data set does not perform well for either mixed or numeric k-means clustering algorithms, although the mixed k-means algorithm has a 2.6% gain over numeric k-means, the performance ratio of each remains close to 50%. Our result here are trivial, the clustering either appears to converge in two iterations or loses a cluster in the first iteration. Hierarchical clustering algorithms perform significantly better on this data set [**8**]. The results for this data set were not published by Ahmad et al.

6. Conclusion

The mixed k-means clustering algorithm was effective, and consistently had a greater performance ratio than the numeric k-means algorithm for four of the five data sets. In the Iris data set, while falling short of the 95% performance ratio published by Ahmad et al, our implementation of the mixed k-means algorithm averaged slightly higher than the numeric k-means algorithm. Since this is a numeric data set, we did not expect significant gains, and were confirming that the mixed k-means algorithm does not perform any worse. Any gains in this data set should be limited to the effects of the weighted distance from significance.

The Voting Records data set also had small gains over numeric k-means. All attributes in this data set are categorical, so the gains in performance were expected to be higher. Each attribute behaves similar to binary attributes, which may give the numeric k-means algorithm some advantage for this data set.

The Heart Disease data set and Australian Credit Card data set had the largest gains over numeric k-means. The mix of categorical and numeric attributes in these data sets most closely reflects realistic applications of the algorithm.

The Lenses data set, a data set with all categorical attributes, performed poorly with both numeric and mixed k-means clustering algorithms. While the mixed k-means clustering algorithm performed better than the numeric k-means clustering algorithm, a performance ratio of 51.4% is not high enough for use in practical application. This data set is known to perform well with hierarchical clustering algorithms and not with k-means clustering algorithms, possibly because it lacks a structure of m-dimensional spherical groupings.

CHAPTER 6

Application to ASSISTments

Chapters 2 through 5 discussed the development and testing of a mixed k-means clustering algorithm. As noted in the introduction, the algorithm was developed with the hope of automating differentiation in the ASSISTments system. The first step towards this automation is to use the mixed k-means clustering algorithm to cluster students, in order to find more accurate regressions on smaller samples of students.

In this chapter, we discuss some of the challenges in the application of the mixed k-means algorithm to the ASSISTments data set. The initial objective was simply to gain an elementary understanding of the structure of the data in this initial clustering of the students' responses, and then to design an experiment that might yield meaningful predictions. Due to limitations on computational time, we were unable to complete this initial clustering. These limits appear in the stages of preprocessing the data and in the calculation of significance. While some improvements were made, an extensive redesign of the implementation would be necessary to get meaningful results. Since clustering is an NP-hard problem, a redesigned implementation (that takes advantage of object oriented design and vectorized computations, optimized for parallel processing and/or CUDA) may not sufficiently reduce computational time.

1. Preprocessing for Significance: N Choose K

The attributes *order id*, *assistment id*, and *problem id* had to be excluded because of the computational time required for the MATLAB function *nchoosek*, which writes a matrix with all possible pairings of elements of the attribute. The function *nchoosek* is used in both the calculation of significance and the calculation of distance between a data point and a cluster center. The function *nchoosek* takes 1.71 seconds for 1,000 unique values, 15.526 seconds for 2,000 unique values, and 127.309 seconds for 5,000 unique values.[1] The function *nchoosek* is impractical when an attribute has more than 10,000 unique values.

[1]MATLAB v8.5, Windows 10 x64, Intel Core i7-6700k (4.00 GHz)

Attribute	Type	Unique Values
order id	Categorical	1011079
assignment id	Categorical	6163
user id	Categorical	8519
assistment id	Categorical	22039
problem id	Categorical	35978
original	Numeric	2
correct	Numeric	10
attempt count	Numeric	270
ms first response time	Numeric	155586
tutor mode	Categorical	5
answer type	Categorical	7
sequence id	Categorical	1552
student class id	Categorical	435
position	Numeric	297
problem set type	Categorical	4
base sequence id	Categorical	1128
list skill ids	Categorical	348
list skills	Categorical	337
teacher id	Categorical	242
school id	Categorical	109

TABLE 6.1. Significance Values for 2009-10 Data Set

The inclusion of the attributes *assignment id* and *user id* increased the computational time for the significance algorithm, such that the cooling system of a personal computer is not sufficient to maintain a safe operating temperature after many hours. These attributes have also been excluded, but may be practical to execute on a dedicated server.

2. Calculation of Significance

In the attribute with the greatest remaining number of unique values *sequence id*, there are $2,407,152$ unique pairs of values to be compared across 14 other attributes and $1,011,097$ data points. This takes approximately $5,000$ hours for this one attribute on a personal computer.

The number of unique values of each categorical attribute leads to problems with both computational time and storage in RAM. This is not surprising, as expectation-maximization machine learning algorithms are NP-hard [**9**]. In this case, to calculate the significance of

an attribute, each unique pair of unique values of the element must be compared. On a personal computer, this comparison takes 85 seconds.

Let $u(A_m)$ be the number of unique elements in attribute A_m. Counting the comparison in n choose k and omitting permutations of order, we find that there are $u(A_m)^2 - u(A_m)$ comparisons. Then, on a personal computer, the approximate time to complete the computation is expressed below, in hours.

$$t = \frac{85}{3600} \sum_{m=1}^{M} [\mathrm{u}(A_m)^2 - \mathrm{u}(A_m)]$$

Considering all of the attributes would take approximately 2.4×10^{10} hours. Considering the 9 attributes that minimize computational time would take 310 hours, and 8 attributes that minimize computational time would take 31 hours. Removing more attributes would likely yield a trivial result, as the majority of the data is not considered. It is not practical to complete these calculations on a personal computer.

3. Conclusions from Application

The source for the calculation of significance is written in serial, assigning each Central Processing Unit (CPU) core to one comparison of unique values, computing each of the steps to find the significance, and returning the result. Vectorization of that code may significantly decrease the necessary computational time, and would allow for a Graphics Processing Unit (GPU) to process the code. MATLAB is generally optimized for vectorized code, and includes GPU parallel computing with NVIDIA CUDA graphics cards in the parallel computing toolbox. This would be the next step in improving this algorithm, but there is not sufficient time to complete this within the project.

An improved source may be run on the MATLAB Distributed Computing Server for Amazon EC2. This would allow for up to 256 workers in the parallel processing instead of the four workers in an Intel Core i7 processor. In addition to dramatic performance enhancements, a cloud computing solution would eliminate temperature concerns with an extended run time, as professional server systems are better suited for temperature management. Unfortunately, cloud computing is not available on the Student License of MATLAB.

While the mixed k-means algorithm allows for additional types of data to be considered for clustering of data sets, it may not be practical for very large data sets, especially data sets with large numbers of unique values of categorical attributes. Even with an improved source, it may not be practical to update clustering daily or weekly in order to

keep up with an expanding data base of problems, scaffolding questions, and student responses. The mixed k-means algorithm may be best suited for small data sets, or to determine clustering rules for a system where new types of problems or users are not likely to be encountered regularly. Without the introduction of new types of problems or users, the clustering could be run monthly or annually, and regression models on each cluster may be updated more frequently.

CHAPTER 7

Conclusions and Further Research

We have shown that the mixed k-means algorithm performs at least as well as the numeric k-means algorithm on numeric data sets, and improves performance on categorical data sets. The mixed k-means algorithm is limited by two assumptions common to the numeric k-means algorithm: (a) that there are no NaN entries, and (b) that the data set can be structed into sum number (k) of m-dimensional spheres. For benchmark data sets, two methods of error analysis can be used to compare the goodness of fit of a clustering model: performance ratios and silhouette values. Finally, the proposed mixed k-means implementation is not practical for application to a data set with categorical attributes that have a large number of unique values.

1. Questions for Future Research

There are three immediate extensions of this implementation of Ahmad's mixed k-means algorithm. First, this MATLAB implementation should be tested on a larger variety of data sets, in an effort to demonstrate a broader utility. Additional benchmark tests should be compared to a larger variety of clustering algorithms including both modifications of k-means and hierarchical algorithms. This experimentation may also yield some intuition for the types of data sets where the mixed k-means algorithm performs well and the types where it is prone to error or trivial results.

Second, the MATLAB implementation should be rewritten with attention to object oriented design, as well as vectorizing the process in whole or in part to take advantage of CUDA and parallel processing for large data sets. In this redesign of the implementation, a number of the objects and functions should be written as handle classes rather than as value classes to decrease general overhead and eliminate bottlenecks in parallel processing. The larger classes dealing with data input and output from the CSV file and interacting with the user would likely function well as value classes, while the class 'mixedclust' that is used to execute the clustering (and repeatedly manipulates large matrices) might function better as a handle class.

Third, the optimization of κ_i in the discretization should be studied further. There is likely a better measure than silhouette value for the goodness of fit of a clustering or the number of clusters, for the discretization of a single attribute. Silhouette values do not always behave well at boundaries, as κ_i approaches N or 1, where the number of clusters or number of data points per cluster (respectively) approaches 1. A hierarchial clustering algorithm might perform well instead of the numeric k-means for determining the appropriate number of clusters for discretization. Optimally the selection of κ_i should reflect the underlying structure of the attribute A_i. In the case that A_i is a continuously distributed numeric attribute, there may be an optimal ratio of κ_i to N.

After further study of this implementation, there are a number of questions that extend the application to the ASSISTments data set. The original problem that inspired this thesis has been solved in part by Piech et al. [**11**], specifically building directed graphs that give trajectories for students between skills. After consideration of the objectives of current users of the ASSISTments application, it may be sufficient to use skill-builders with randomly selected problems and differentiate with trajectories built from directed graphs of skills rather than directed graphs of problems within each skill.

An algorithm that considers both categorical and numeric attributes may provide additional insight into the ASSISTments data set. This would require substantial improvements in the performance of this implementation, as well as additional computing resources. If this is achieved, it would be interesting to investigate the inclusion or exclusion of specific attributes in the clustering (and the impact on the significance values). Additionally, if the data set is clustered by individual responses (in its current for as a CSV file, rather than building a matrix with metrics for students or problems), is a pattern revealed by a partition of groups of students, problems or sequences? Finally, do different data sets (specifically 2009-2010 vs. 2012-2013) impact this result? Further research in these questions of application should carefully consider existing literature on clustering within the ASSISTments data sets.

CHAPTER 8

Reflection

This thesis was not able to meet the original goals and objectives for a value added feature for the ASSISTments application. The product of this thesis, the MATLAB source, contributes both a functional implementation and a suggestion of adaptive discretization. More significantly, this thesis has developed personal skill, interest and understanding in three key areas: the process of inquiry in mathematics, the design of computer programs, and quantitative analysis of educational systems. Finally the work of this thesis has impacted my work as a pre-service teacher, in time-management, philosophy of curriculum and assessment design, and in developing understanding of and experience in mathematics research.

1. Inquiry in mathematics

This thesis developed my understanding for the process of inquiry in mathematics, how existing research is reviewed and how new applications are explored. In particular, the connections between graph theory, real analysis, probability, and computer science were reinforced as well as the connection between pure and applied mathematics. In this thesis, as I found bugs and explored the implications of my own adaptations of the mixed k-means algorithm, I was forced to review and connect content from previous and current study in each of these fields.

Mathematical inquiry requires not only problem solving skills but also flexibility among multiple representations and sub-fields. Additionally, mathematical inquiry requires flexibility in process and procedure, investigating connections between new content areas, scaling solutions from specific cases to general applications, and adjusting the project's time-line for unexpected digressions. In connection to mathematics education, an understanding of and experience with the method of inquiry in mathematical science are essential to build authentic content and to model authentic problem solving.

2. Design of computer programs

A lack of formal computer science coursework has significantly affected this thesis. My background in computer science can be more accurately be described as experience 'coding' in a number of languages. This allows me to quickly become comfortable in new environments and with new languages , and continue to learn and troubleshoot in forums. However, some of the significant challenges in this thesis would be resolved with a deeper theoretical understanding of principles of program design and the use and role of hardware.

Restrictions on data input and output, and design for parallel processing, object oriented design and CUDA optimization would all have significantly improved constraints on memory and computational time experienced throughout this thesis (in the process of this thesis, I upgraded the hardware of my personal computer a half dozen times, overheated the CPU, and replaced the laptop with a desktop). A better understanding of computer science may have given me an early idea of the realistic hardware requirements for this thesis. This has motivated me to integrate computer science coursework into my graduate studies, along with further study in mathematics and statistics. Additionally, this has reinforced the importance of integrating introductory computer science concepts and tools into K-16 science and mathematics curriculum.

3. Quantitative analysis of educational systems

The most significant impact of this thesis is an understanding of the complexity and difficulty of the problem of quantitative analysis of educational systems. I had hoped that methods of educational data mining would quickly yield results that either give an accurate assessment of students' progress and needs (which can then be used to evaluate the effectiveness of particular programs) or results that would provide fast and accurate differentiation for students (which could be powerful for underserved demographics). Unfortunately this process of quantitative analysis is much more complex, and the development and benchmark testing of an algorithm took longer that I had anticipated.

I will continue to study quantitative analysis of educational systems in graduate school. This thesis has given me perspective on the role of quantitative assessment in education, which may serve better as a research tool than a definitive diagnostic. Large educational data sets are complex, and if data is collected on many attributes with a fine level of precision, it can be computationally difficult to find meaningful results; smaller data sets may not be able to scale across diverse populations.

Current methods of quantitative assessment may be better served for curriculum development and internal assessments than driving policy decisions. This experience of research in quantitative analysis has improved my teaching, better integrating and analyzing qualitative and quantitative assessment of my students and lessons, especially in questions of difference in performance of specific socioeconomic groups.

APPENDIX A

Benchmark Performance

1. Performance on Benchmark Data Sets, 10 Trials

```
nTrials = 10
-------------------------
Iris
Elapsed time is 480.705623 seconds.
-------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance     0.88591         0.82282
    Silhouette      0.78531         0.81208

-------------------------
Lenses
Elapsed time is 26.449903 seconds.
-------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance     0.58261         0.46522
    Silhouette      0.58863         0.58624

-------------------------
Heart
Elapsed time is 838.774515 seconds.
-------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance     0.83963         0.76296
    Silhouette      0.45377         0.81353

-------------------------
Vote
```

```
Elapsed time is 670.403463 seconds.
-------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance    0.87327         0.86866
    Silhouette     0.75569         0.61724

-------------------------
Australian
Elapsed time is 164.631749 seconds.
-------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance      0.782         0.61248
    Silhouette     0.58958         0.89507

-------------------------
```

2. Performance on Benchmark Data Sets, 100 Trials

```
nTrials = 100
-------------------------
Iris
Elapsed time is 4277.812547 seconds.
-------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance    0.85416         0.84174
    Silhouette     0.79697         0.81175

-------------------------
Lenses
Elapsed time is 254.807561 seconds.
-------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance    0.51391         0.48783
    Silhouette     0.59164         0.58876

-------------------------
```

```
Heart
Elapsed time is 8931.101548 seconds.
--------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance     0.83981         0.75615
    Silhouette      0.45644         0.79679

--------------------------
Vote
Elapsed time is 7893.835162 seconds.
--------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance     0.87327         0.85276
    Silhouette      0.75569         0.61025

--------------------------
Australian
Elapsed time is 7232.194358 seconds.
--------------------------
                    Mixed_kMeans    Numeric_kMeans
                     ------------    --------------

    Performance     0.75392         0.59216
    Silhouette      0.59283         0.86832

--------------------------
```

APPENDIX B

Object Oriented Clustering - MATLAB Source

```
classdef mixedclust
    %MIXEDCLUST is a class for copmuting kmeans clustering
    %for data sets with numeric and categorical variables.
    %
    % Other m-files required:
    %     assignmentoptimal.m, Markus Buehren ...
    %         http://www.mathworks.com/matlabcentral/...
    %           fileexchange/6543-functions-for-the-...
    %           rectangular-assignment-problem ...
    %            /content/assignmentoptimal.m
    %     This function implements the Hungarian Algorithm.
    %
    % Subfunctions:
    %     significance       (C) Ahmad Alsahaf - GNU GPL
    %     cluster_center     (C) Ahmad Alsahaf - GNU GPL
    %     algo_dist          (C) Ahmad Alsahaf - GNU GPL
    %     dist_to_center     (C) Ahmad Alsahaf - GNU GPL
    %
    %     These subfunctions can also be found as part ...
       of Ahmad Alsahaf's...
    %       amjams/mixedkmeans package on MATLAB ...
       Central/FileExchange.  They ...
    %       have been modified for use, and are inluded ...
       in mixedclust.m.
    %       http:///www.mathworks.com/...
    %           matlabcentral/fileexchange
    %
    % Author: Camden Glenn Bock
    % 598 Bates College, Lewistion, ME 04240
    % cbock@bates.edu, camdenbock@gmail.com
    % http://www.camdenbock.com
    % December 2015; Last Revision: 12/30/2015
    %
    %% Copyright (C) 2016  Camden Bock - GPL v. 3.0
```

```
32      %
33      % 'This program is liscensed under GPL v3.0'
34      % 'This program is modified from Ahmad Alsahaf`s ...
           package: ...
35      % amjams/mixedkmeans'.
36      %
37      % This program is free software: you can ...
           redistribute it and/or modify
38      % it under the terms of the GNU General Public ...
           License as published by
39      % the Free Software Foundation, either version 3 of ...
           the License, or
40      % any later version.
41      %
42      % This program is distributed in the hope that it ...
           will be useful,
43      % but WITHOUT ANY WARRANTY; without even the ...
           implied warranty of
44      % MERCHANTABILITY or FITNESS FOR A PARTICULAR ...
           PURPOSE.  See the
45      % GNU General Public License for more details.
46      %
47      % You should have received a copy of the GNU ...
           General Public License
48      % along with this program.  If not, see ...
           <http://www.gnu.org/licenses/>.
49      %
50      %
51      % 'This program comes with ABSOLUTELY NO WARRANTY;' ...
52      % 'for details type view source. This is free ...
           software, and' ...
53      % 'you are welcome to redistribute it display under ...
           certain' ...
54      % 'conditions; see <http://www.gnu.org/licenses/>', ...
55      % ('Copyright (C) 2016  Camden Bock, Bates College'))
56
57
58
59      %------------- BEGIN CODE --------------
60
61      properties
62          data
63          m_idx
64          k
65          max_iter
```

```
66          inputType
67          trialsNo
68          mixedClustering
69          numericClustering
70          origionalData
71          significances
72          data_discrete
73          normalizedData
74          tempvar
75          all_dist
76          silh_mean
77          performance
78          idx
79      end
80
81      methods
82          function obj = mixedclust(data, k, max_iter, ...
                inputType,trialsNo)
83
84              [dn, ¬] = size(data);
85              [¬, ¬] = size(inputType);
86              if nargin < 4
87                  trialsNo = 1;
88                  inputType = [];
89              elseif nargin < 3
90                  max_iter = 1000;
91              elseif nargin < 2
92                  display('Not Enough Arguments')
93              end
94              obj.tempvar.dn = dn;
95              obj.trialsNo = trialsNo;
96              obj.data = data;
97              obj.k = k;
98              obj.max_iter = max_iter;
99              obj.inputType = inputType;
100
101             % replace NaN entrieies
102             obj.data(isnan(obj.data)) = 1;
103             obj = normalize(obj);
104             obj = discretize(obj);
105             obj = sigpairs(obj);
106             obj = signif(obj);
107         end
108         function visulaizeNum(obj, trialnum)
109             if nargin < 2
```

```
110                 trialnum = 1;
111             end
112             pointClusterVis(obj.numericClust, trialnum)
113         end
114         function visualizeMix(obj, trialnum)
115             if nargin < 2
116                 trialnum = 1;
117             end
118             pointClusterVis(obj.mixedClust, trialnum)
119         end
120         function obj = normalize(obj)
121             obj.tempvar.m_distance = ...
122                 zeros(obj.tempvar.dn,obj.k,obj.trialsNo);
123             obj.tempvar.n_distance = ...
124                 zeros(obj.tempvar.dn,obj.k,obj.trialsNo);
125
126
127
128             %% Mixed KMeans
129             obj.tempvar.silhouette_mixed_mean = ...
                    zeros(1,obj.trialsNo);
130
131             obj.m_idx = zeros(obj.tempvar.dn,obj.trialsNo);
132
133             [obj.tempvar.n,obj.tempvar.m] = size(obj.data);
134             obj.tempvar.idx_num = find(¬obj.inputType);
135
136             %% Normalize Numeric Data
137             for i=1:numel(obj.tempvar.idx_num)
138                 obj.data(:,obj.tempvar.idx_num(i)) = ...
139                     (obj.data(:,obj.tempvar.idx_num(i)) ...
                            - ...
140                     repmat(min(obj.data(:,obj.tempvar.idx_num(i))),...
141                     size(obj.data(:,obj.tempvar.idx_num(i))))) ...
                            ...
142                     /(max(obj.data(:,obj.tempvar.idx_num(i)))...
143                     -min(obj.data(:,obj.tempvar.idx_num(i))));
144             end
145             obj.normalizedData = obj.data;
146         end
147         function obj = discretize(obj)
148             %% Discretize Numeric Data
149             obj.data_discrete = obj.data;
150             %ensure k << N
151             %             if obj.tempvar.dn >1000
```

```
152             %                 max_k = ...
                    round(obj.tempvar.dn/200);
153             %             else
154             max_k = 20;
155             %             end
156             obj.tempvar.idx_cat = find(-1*obj.inputType+1);
157             for i=1:numel(obj.tempvar.idx_cat)
158                 silh_avg = zeros(max_k,1);
159                 data_num = ...
                        obj.data(:,obj.tempvar.idx_cat(i));
160                 for k_iter=1:max_k
161
162                     [idx,¬,¬,D]=...
163                         kmeans(data_num,k_iter+1,'dist', ...
                                ...
164                         'sqeuclidean','MaxIter',100,...
165                         'Options',statset('UseParallel',1));
166                     [Drow,¬] = size(D);
167                     silh = zeros(1,Drow);
168                     for drow = 1:Drow
169                         [a_drow,excl_D] = min(D(drow,:));
170                         b_drow = ...
                                min(D(drow,[1:(excl_D-1),...
171                              (excl_D+1):end]));
172                         silh(drow) = (b_drow-a_drow)...
173                             /max(a_drow,b_drow);
174                     end
175                     %Ensure selection has >1 unique value
176                     if numel(unique(idx))>1
177                         silh_avg(k_iter) = mean(silh);
178                     else
179                         silh_avg(k_iter) = -10000000;
180                     end
181
182                 end
183                 [¬,k_best] = max(silh_avg);
184                 k_best = k_best+1;
185                 obj.data_discrete(:,obj.tempvar.idx_cat(i)) ...
                        = ...
186                     kmeans(obj.data(:,obj.tempvar.idx_cat(i)),...
187                     k_best);
188             end
189         end
190         function obj = sigpairs(obj)
191             D = obj.data_discrete;
```

```
192
193             % define the attribute, its unique values, ...
                    and all unique pairs
194             for i=1:obj.tempvar.m
195
196                 a = D(:,i);
197                 unique_a = find(accumarray(a+1,1))-1;
198                 all_pairs = nchoosek(unique_a,2);
199                 varname = ['var', num2str(i)];
200                 obj.tempvar.(varname).all_pairs = ...
                        all_pairs;
201
202             end
203         end
204         function obj = signif(obj)
205
206             sigs = zeros(obj.tempvar.m,1);
207             parfor i=1:obj.tempvar.m
208
209                 sigs(i) = significance(obj,i);
210
211             end
212             obj.significances = sigs;
213         end
214         function obj = mclust(obj)
215             n = obj.tempvar.dn;
216             for iMixed = 1:obj.trialsNo
217                 %try
218                 curr_idx = randi([1 obj.k],n,1);
219
220                 obj = algo_distance(obj);
221
222                 new_idx = zeros(n,1);
223
224                 count = 0;
225                 while(isequal(new_idx,curr_idx)==0 &&...
226                         count<obj.max_iter)
227
228                     if count>0
229                         curr_idx = new_idx;
230                     end
231
232                     all_centers = struct;
233                     droppedACluster = 0;
234                     for i=1:obj.k
```

```
235                     curr_cluster = ...
                            obj.data(curr_idx==i,:);
236                     curr_center = ...
                            cluster_center(curr_cluster, ...
                            obj);
237                     if curr_center.cluster_size < 1
238                         droppedACluster = 1;
239                     end
240                     name = ...
                            ['center_',sprintf('%03d',i)];
241                     all_centers.(name) = curr_center;
242                 end
243
244                 silh_c = zeros(1,n);
245
246                 if droppedACluster == 0
247
248                     for i=1:n
249                         k_distances = zeros(obj.k,1);
250                         data_i = obj.data(i,:);
251                         for j=1:obj.k
252                             name_now = ['center_',...
253                                 sprintf('%03d',j)];
254                             center_now = ...
                                    all_centers.(name_now);
255                             obj.tempvar.data_i = ...
                                    data_i;
256                             obj.tempvar.center_now ...
                                    = center_now;
257                             obj.tempvar.all_dist = ...
                                    obj.all_dist;
258                             k_distances(j) = ...
                                    dist_to_center(obj);
259                         end
260
261                         [~,new_idx(i)] = ...
                                min(k_distances);
262                         min1 = min(k_distances);
263                         min2 = ...
                                min(setdiff(k_distances(:),...
264                                 min(k_distances(:))));
265                         silh_c(i) = (min2-min1)/...
266                             max(min1,min2);
267                     end
268                 else
```

```
269                     new_idx = randi([1 obj.k],n,1);
270                 end
271                 count = count+1;
272             end
273
274             idx = new_idx;
275             obj.m_idx(:,iMixed) = idx;
276             obj.silh_mean(iMixed) = mean(silh_c);
277             %                 catch
278             %                     fprintf('Error ...
                    Non-existent field categorical. ...
                    iMixed = %d', ...
279             %                         iMixed);
280             %                     display('- - - -  ...
                    execution will continue  - - - -')
281             %                     iMixed = iMixed-1;
282             %                 end
283         end
284
285     end
286     %% Significances
287     function sig = significance(obj,idx)
288         %SIGNIFICANCE: finds the significance of a ...
                categorical attribute or a
289         %discretized version of a numerical attribute
290
291         % input:
292         %   D:   the dataset of all attributes
293         %   idx: index of the attribute whose ...
                significance is to be found
294         %
295         % ouput:
296         %   sig: the significance of the attribute
297         %
298         %
299         % Copyright 2015 Ahmad Alsahaf
300         % Research fellow, Politecnico di Milano
301         % ahmadalsahaf@gmail.com
302
303         % number of attributes
304         D = obj.data_discrete;
305         m = size(D,2);
306
307         % define the attribute, its unique values, ...
                and all unique pairs
```

```
308             a = D(:,idx);
309             unique_a = find(accumarray(a+1,1))-1;
310             varname = ['var', num2str(idx)];
311             all_pairs = obj.tempvar.(varname).all_pairs;
312             %Note nchoosek is impractical for n>15
313             num_pairs = size(all_pairs,1);
314
315             % the number of all Δ distances
316             num_Δ = (m-1)*num_pairs;
317
318             % find all Δs and average them
319             feature_c = 1:m;  feature_c(idx)=[];   ...
                   %complementary feature set
320             Δ_sum = 0;                          %initialize
321             for i=1:num_pairs
322                 curr_pair = all_pairs(i,:);
323                 for j=1:(m-1)
324                     % intialize distance
325                     d = 0;
326
327                     % initalize support set
328                     w = [];
329                     w_c = [];
330
331                     % the number of categorical values ...
                           in D(:,feature_c(j))
332                     data_temp = D(:,feature_c(j))+1;
333                     unique_j = ...
                           find(accumarray(data_temp,1))-1;
334                     vj = numel(unique_j);
335
336
337                     % begin algorithm
338                     for t = 1:vj
339                         ut = unique_j(t);
340
341                         % locations
342                         ut_in_aj = ...
                               find(D(:,feature_c(j))==ut);
343                         x_in_ai = find(a==curr_pair(1));
344                         y_in_ai = find(a==curr_pair(2));
345
346                         % probabilities
```

```
347                         p_ux = ...
                                numel(x_in_ai(ismembc(x_in_ai, ...
                                ut_in_aj))) ...
348                              /numel(x_in_ai);
349                         p_uy = ...
                                numel(y_in_ai(ismembc(y_in_ai, ...
                                ut_in_aj))) ...
350                              /numel(y_in_ai);
351
352                         % conditions
353                         if p_ux≥ p_uy
354                              w = [w;ut];           ...
                                    %update support set
355                              d = d+p_ux;           ...
                                    %update distance
356                         else
357                              w_c = [w_c;p_uy];     ...
                                    %update complement ...
                                    support set
358                              d = d+p_uy;           ...
                                    %update distance
359                         end
360
361
362                     end
363                     Δ = d-1;                        ...
                           %restrict distance to [0,1]
364                     Δ_sum = Δ_sum + Δ;
365                 end
366             end
367             % find average distance, which is the ...
                   significance
368             sig = Δ_sum/num_Δ;
369         end
370
371
372         %% Algo Distance
373
374         function obj = algo_distance(obj)
375             % Copyright 2015 Ahmad Alsahaf
376             % Research fellow, Politecnico di Milano
377             % ahmadalsahaf@gmail.com
378             data_discrete = obj.data_discrete;
379
380             % data dimenionsality
```

```
381         [¬,m] = size(data_discrete);
382
383         %intialize distance vector; which contains ...
               all distances between all pairs
384         all_dist = [];
385
386         for i = 1:m
387             % define ai, the current attribute
388             ai = data_discrete(:,i);
389
390             % find all pairs of unique values in ...
                   current feature
391             unique_ai = find(accumarray(ai+1,1))-1;
392             all_pairs = nchoosek(unique_ai,2);
393
394             % find complement feature set
395             feat_c = 1:m;  feat_c(i) = [];
396
397             for j= 1:size(all_pairs,1)
398                 % initialize sum and define current ...
                       pair
399                 sum_Δ = 0;
400                 curr_pair = all_pairs(j,:);
401
402                 % find distance between the pair ...
                       for all Aj
403                 for k = 1:m-1
404                     % define aj
405                     aj = data_discrete(:,feat_c(k));
406
407                     % update the sum
408                     % intialize distance
409                     d = 0;
410
411                     % initalize support set
412                     w = [];
413                     w_c = [];
414
415                     % the number of categorical ...
                           values in aj
416                     unique_j = ...
                           find(accumarray(aj+1,1))-1;
417
418                     vj = numel(unique_j);
419
```

```
420                         % begin algorithm
421                         for t = 1:vj
422                             ut = unique_j(t);
423
424                             % locations
425                             ut_in_aj = find(aj==ut);
426                             x_in_ai = ...
                                    find(ai==curr_pair(1));
427                             y_in_ai = ...
                                    find(ai==curr_pair(2));
428
429                             % probabilities
430                             p_ux = ...
                                    numel(x_in_ai(ismembc(x_in_ai, ...
                                    ut_in_aj))) ...
431                                  /numel(x_in_ai);
432                             p_uy = ...
                                    numel(y_in_ai(ismembc(y_in_ai, ...
                                    ut_in_aj))) ...
433                                  /numel(y_in_ai);
434
435                             % conditions
436                             if p_ux≥ p_uy
437                                 w = [w;ut];          ...
                                        %update support set
438                                 d = d+p_ux;          ...
                                        %update distance
439                             else
440                                 w_c = [w_c;p_uy];    ...
                                        %update complement ...
                                        support set
441                                 d = d+p_uy;          ...
                                        %update distance
442                             end
443
444
445                         end
446                         Δ = d-1;                      ...
                               %restrict distance to [0,1]
447                         sum_Δ = sum_Δ + Δ;
448                     end
449                     %         update the distance vector
450                     sum_Δ = sum_Δ/(m-1);
451
```

```
452                 %         arranged as ...
                        [attribute_idx,first_value(lower), ...
                        ...
453                 %         ...
                        second_value(higher),distance];
454                 pair_sorted = sort(curr_pair,'ascend');
455                 all_dist = [all_dist; ...
456                     i,pair_sorted(1),pair_sorted(2),...
457                     sum_Δ];
458                 obj.all_dist = all_dist;
459             end
460         end
461
462     end
463
464     %% Cluster Center
465     function [ center ] = cluster_center(cluster, obj)
466         %CLUSTER_CENTER find cluster centers for ...
                mixed attributes
467
468         % inputs:
469         %     cluster:    the members of the cluster
470         %     input_type: binary index indicating ...
                the type of attributes...
471         %     (1 for categorical)
472         %
473         % output:
474         %     center:     the center of the cluster
475         % Copyright 2015 Ahmad Alsahaf
476         % Research fellow, Politecnico di Milano
477         % ahmadalsahaf@gmail.com
478
479         % intialize a structure variable to save ...
                the centers
480
481         input_type = obj.inputType;
482         center = struct;
483
484         % cluster dimensions, and numerical and ...
                categorical feature indices
485         [n,¬] = size(cluster);
486         center.cluster_size = n;
487         cat_idx = find(input_type);
488         num_idx = find(¬input_type);
489
```

```
490
491             % find center for each numerical attribute
492             for i=1:numel(num_idx)
493                 curr_att = cluster(:,num_idx(i));
494                 name = ['att_',sprintf('%03d',num_idx(i))];
495                 center.numerical.(name) = mean(curr_att);
496             end
497
498             % find center for each categorical attribute
499             for i=1:numel(cat_idx)
500                 curr_att = cluster(:,cat_idx(i));
501                 name = ['att_',sprintf('%03d',cat_idx(i))];
502                 uniq_curr_att = ...
                        find(accumarray(curr_att+1,1))-1;
503
504                 for j=1:numel(uniq_curr_att)
505                     name_value = ...
                            ['value_',sprintf('%03d',j)];
506                     curr_value = uniq_curr_att(j);
507                     count_value = ...
                            numel(find(curr_att==curr_value));
508                     center.categorical.(name).(name_value).value ...
                            = curr_value;
509                     center.categorical.(name).(name_value).count ...
                            = count_value;
510                 end
511             end
512
513         end
514
515         %% Distance to Center
516
517         function theta = dist_to_center(obj)
518
519             % dist_to_center: computes the the distance ...
                    between a data point and a
520             % cluster center
521
522             % inputs:
523             %     x:              a data point
524             %     c:              a cluster center ...
                    (structure)
525             %     input_type:     binary index ...
                    indicating attributes (1 = categorical)
```

```
526             %     sig:            significance of all ...
                    attributes in the dataset
527             %     dist_all:       list of all distances ...
                    of categorical values
528             %
529             % output:
530             %     theta:  the distance between x and c
531             %
532             %
533             % Copyright 2015 Ahmad Alsahaf
534             % Research fellow, Politecnico di Milano
535             % ahmadalsahaf@gmail.com
536
537             x = obj.tempvar.data_i;
538             c = obj.tempvar.center_now;
539             input_type = obj.inputType;
540             sig = obj.significances;
541             dist_all = obj.tempvar.all_dist;
542
543             % find indices
544             cat_idx = find(input_type);
545             num_idx = find(¬input_type);
546
547             % load cluster size
548             cluster_size = c.cluster_size;
549
550             % distance for numerical attributes
551
552             % initialize numerical distance to zero
553             sum_distance_numerical_v = ...
                    zeros(1,numel(num_idx));
554
555             % find distance for each numerical ...
                    attribute and add to sum
556             for i=1:numel(num_idx)
557                 d = x(num_idx(i));
558                 name = ['att_',sprintf('%03d',num_idx(i))];
559                 num_center = c.numerical.(name);
560                 curr_significance = sig(num_idx(i));
561                 curr_dist = ...
                        (curr_significance*(d-num_center))^2;
562                 sum_distance_numerical_v(i) = curr_dist;
563             end
564             sum_distance_numerical = ...
                    sum(sum_distance_numerical_v);
```

```
565
566             % display(c)
567             % initialize categorical distance to zero
568             sum_distance_categorical = 0;
569
570             % find distance for each categorical ...
                    attribute and add to sum
571             for i=1:numel(cat_idx)
572                 % access the current categorical ...
                        attribute from structure
573                 name = ['att_',sprintf('%03d',cat_idx(i))];
574                 curr_att = c.categorical.(name);
575                 value_names = fieldnames(curr_att);
576
577                 % initialize sum for this categorical ...
                        attribute
578                 sum_categorical_current = ...
                        zeros(1,numel(value_names));
579
580                 % now access values within that ...
                        attribute in the cluster
581
582                 for j=1:numel(value_names)
583                     value_in_point = x(cat_idx(i));
584                     value_in_cluster = ...
                            curr_att.(value_names{j}).value;
585                     count_in_cluster = ...
                            curr_att.(value_names{j}).count;
586
587                     % find the distance from the list
588                     sorted_values = ...
589                         sort([value_in_point,value_in_cluster],...
590                         'ascend');
591                     idx_dist = dist_all(:,1)==...
592                         cat_idx(i)&dist_all(:,2) == ...
593                         sorted_values(1) & ...
                                dist_all(:,3) ==...
594                         sorted_values(2);
595
596                     % set distance to zero if value is ...
                            equal to center, compute dist
597                     % othewise (i.e. only update when ...
                            different values
598
```

```
599                     if (sorted_values(1) ≠ ...
                            sorted_values(2))
600                         sum_categorical_current(j) = ...
                                ((1/cluster_size)*...
601                             count_in_cluster*...
602                             dist_all(idx_dist, 4))^2;
603                     end
604                 end
605                 sum_distance_categorical = ...
                        sum(sum_categorical_current);
606             end
607             % overall distance
608             theta = sum_distance_numerical + ...
                    sum_distance_categorical;
609         end
610
611
612     end
613 end
```

APPENDIX C

Benchmark Performance - MATLAB Source

```
%Testing of Benchmark Data Sets
nTrials = 1;
sendEmail('Benchmark Begin!')

for i=1:5
    if i==1
        display('Iris')
        iris = ...
            clusteringCompare('iris.all.csv',[],5,2,nTrials);
        sendEmail('Iris Done!')
%     elseif i==2
%
%         display('Lenses')
%         lenses = ...
    clusteringCompare('lenses.all.csv',1:4,5,2,nTrials);
%         sendEmail('Lenses Done!')
    elseif i==3

        display('Heart')
        heartlabels = [2,3,6,7,9,11,12,13];
        heart = clusteringCompare('Heart2.csv',...
            heartlabels,14,2,nTrials);
        sendEmail('Heart Done!')

    elseif i==4

        display('Vote')
        vote = ...
            clusteringCompare('vote.all.csv',1:16,17,2,nTrials);
        sendEmail('Vote Done!')

    elseif i==5

        display('Australian')
```

```
32          australianlabels = [1,4,6,8,9,11,12];
33          australian = ...
                clusteringCompare('australian.all.csv',...
34              australianlabels,15,2,nTrials);
35          sendEmail('Australian Done!')
36
37      end
38
39  %         pause(15*60)
40  end
41
42  statsSummary
```

```
1   classdef clusteringCompare
2       %CLUSTERINGCOMPARE Summary of this class goes here
3       %   Detailed explanation goes here
4       %
5       % Author: Camden Glenn Bock
6       % 598 Bates College, Lewistion, ME 04240
7       % cbock@bates.edu, camdenbock@gmail.com
8       % http://www.camdenbock.com
9       % December 2015; Last Revision: 12/30/2015
10      %
11      %% Copyright (C) 2016  Camden Bock - GPL v. 3.0
12      %
13      % 'This program is liscensed under GPL v3.0'
14      % 'This program is modified from Ahmad Alsahaf`s ...
            package: ...
15      % amjams/mixedkmeans'.
16      %
17      % This program is free software: you can ...
            redistribute it and/or modify
18      % it under the terms of the GNU General Public ...
            License as published by
19      % the Free Software Foundation, either version 3 of ...
            the License, or
20      % any later version.
21      %
22      % This program is distributed in the hope that it ...
            will be useful,
23      % but WITHOUT ANY WARRANTY; without even the ...
            implied warranty of
24      % MERCHANTABILITY or FITNESS FOR A PARTICULAR ...
            PURPOSE.  See the
```

```
25      % GNU General Public License for more details.
26      %
27      % You should have received a copy of the GNU ...
           General Public License
28      % along with this program.  If not, see ...
           <http://www.gnu.org/licenses/>.
29      %
30      %
31      % 'This program comes with ABSOLUTELY NO WARRANTY;' ...
32      % 'for details type view source. This is free ...
           software, and' ...
33      % 'you are welcome to redistribute it display under ...
           certain' ...
34      % 'conditions; see <http://www.gnu.org/licenses/>', ...
35      % ('Copyright (C) 2016  Camden Bock, Bates College'))
36
37
38
39      %% ------------- BEGIN CODE --------------
40
41      properties
42          mixedClust
43          numericClust
44          output
45          trialsNo
46          inputType
47          data
48          tempvar
49      end
50      methods
51          function obj = clusteringCompare(filename, ...
               catAttributes, ncols, ...
52                  startRow, trialsNo)
53              if nargin < 5
54                  trialsNo = 1;
55              elseif nargin < 4
56                  startRow = 1;
57              elseif nargin < 3
58                  display('not enough inputs')
59              end
60              obj.trialsNo = trialsNo;
61              obj = dataimport(obj,filename, ...
                   catAttributes, ncols, startRow);
62              [n,m] = size(obj.data);
```

```
63              obj.mixedClust = mixedclust(obj.data, ...
                    obj.tempvar.k, 1000,...
64                   obj.inputType,trialsNo);
65              obj.mixedClust = mclust(obj.mixedClust);
66              obj.mixedClust.idx = ...
                    obj.mixedClust.m_idx(:,:,1);
67              obj.numericClust = struct;
68              obj.numericClust.idx = zeros(m,obj.trialsNo);
69              obj.numericClust.D = ...
                    zeros(m,obj.tempvar.k,obj.trialsNo);
70              obj.numericClust.silh = zeros(n,obj.trialsNo);
71              obj.numericClust.avg_silh = ...
                    zeros(1,obj.trialsNo);
72              for i=1:obj.trialsNo
73                   [idx,¬,¬,D] = ...
                        kmeans(obj.data,obj.tempvar.k);
74                   obj.numericClust.idx = idx;
75                   obj.numericClust.dist(:,:,i) = D;
76                   silh = zeros(n,1);
77                   for j = 1:n
78                        x = sort(D(j,:));
79                        silh(j) = (x(2)-x(1))/x(2);
80                   end
81                   obj.numericClust.silh(:,i) = silh;
82                   obj.numericClust.avg_silh(i) = mean(silh);
83              end
84              obj = compareOut(obj);
85          end
86          function obj = compareOut(obj)
87              obj.numericClust.performance = ...
                    zeros(1,obj.trialsNo);
88              obj.mixedClust.performance = ...
                    zeros(1,obj.trialsNo);
89              for nameidx=1:2
90                   if nameidx==1
91                        name = 'mixedClust';
92                   elseif nameidx==2
93                        name = 'numericClusst';
94                   end
95                   for i=1:obj.trialsNo
96                        idx = obj.(name).idx(:,:,i);
97                        k = numel(unique(obj.output));
98                        ErrorMatrix = zeros(k);
99                        output_values = unique(obj.output);
100                       for emCol = 1:k
```

```
101                         for emRow = 1:k
102                             output_emRow = ...
                                    output_values(emRow);
103                             for oRow = 1:length(obj.output)
104                                 if (obj.output(oRow) ≠ ...
                                        output_emRow)...
105                                         && (idx(oRow) ...
                                            == emCol);
106                                     ErrorMatrix(emCol, ...
                                        emRow)...
107                                     = ...
                                            ErrorMatrix(emCol,emRow)+1;
108                                 end
109                             end
110                         end
111                     end
112                     [mEM, nEM] = size(ErrorMatrix);
113                     if mEM≠nEM
114                         display('Warning, matrix must ...
                                be square');
115                     end
116                     [¬, count] = ...
                            assignmentoptimal(ErrorMatrix);
117                     obj.(name).performance(i) = ...
                            1-count/length(idx);
118                 end
119             end
120         end
121         function obj = dataimport(obj,filename, ...
                catAttributes, ncols, startRow)
122
123             delimiter = ',';
124             endRow = inf;
125
126             %% Format string for each line of text:  ...
                    Autmoated for user input
127             inputBlock = ('%f');
128             formatSpec = char(1:(2*ncols + 8));
129             for i = 2:2:(2*ncols)
130                 formatSpec((i-1):i) = inputBlock;
131             end
132             formatSpec((2*ncols+1):(2*ncols+8))=...
133                 ('%[^\n\r]');
134
135             %% Open the text file.
```

```
136             fileID = fopen(filename,'r');
137
138             %% Read columns of data according to format ...
                   string.
139             % This call is based on the structure of ...
                   the file used to generate this
140             % code. If an error occurs for a different ...
                   file, try regenerating the code
141             % from the Import Tool.
142             dataArray = textscan(fileID, ...
                   formatSpec, endRow(1)-startRow(1)+1, ...
143                'Delimiter', delimiter, 'HeaderLines', ...
                      startRow(1)-1, ...
144                'ReturnOnError', false);
145             for block=2:length(startRow)
146                frewind(fileID);
147                dataArrayBlock = textscan(fileID, ...
                      formatSpec, ...
148                    endRow(block)-startRow(block)+1, ...
                         'Delimiter', delimiter, ...
149                    'HeaderLines', startRow(block)-1, ...
                         'ReturnOnError', false);
150                for col=1:length(dataArray)
151                    dataArray{col} = [dataArray{col}; ...
                         dataArrayBlock{col}];
152                end
153             end
154
155             %% Close the text file.
156             fclose(fileID);
157
158             %% Create output variable
159             datasource = [dataArray{1:end-1}];
160
161             %% Options for alsahaf_mixed_kmeans
162             obj.data = datasource(:,1:(end-1));
163             obj.output = datasource(:,end);
164             if min(min(obj.output))==0
165                obj.output = obj.output + 1;
166             elseif min(min(obj.output))≠1
167                disp('Error: Datasource may not have ...
                      proper categorical assignments')
168             end
169
170             obj.tempvar.k = length(unique(obj.output));
```

```
171
172             [¬,dc] = size(obj.data);
173             obj.inputType = zeros(1,dc);
174             for q=1:length(catAttributes)
175                 obj.inputType(catAttributes(q)) = 1;
176                 if min(min(obj.data(:, ...
                        catAttributes(q)))) == 0
177                     obj.data(:,catAttributes(q)) =...
178                         obj.data(:, catAttributes(q)) + 1;
179                 elseif min(min(obj.data(:, ...
                        catAttributes(q)))) ≠1
180                     disp('Error: Datasource may not ...
                            have proper categorical ...
                            assignments')
181                 end
182             end
183             obj.tempvar.inputType = obj.inputType;
184         end
185         function visulaizeNum(obj, trialnum)
186             if nargin < 2
187                 trialnum = 1;
188             end
189             pointClusterVis(obj.numericClust, trialnum)
190         end
191         function visualizeMix(obj, trialnum)
192             if nargin < 2
193                 trialnum = 1;
194             end
195             pointClusterVis(obj.mixedClust, trialnum)
196         end
197     end
198
199 end
```

```
1 %Stats for 100trail clusterings
2 % varnames = {'australian','heart','iris','lenses','vote'};
3 % clustnames = {'mixedClust','numericClust'};
4 % propnames = ...
      ['k','idx','silhouette','performance','distance'];
5 v = 5;
6 n = 100;
7
8 summary = struct;
9 summary.perf = zeros(v,n);
```

```
10 summary.silh = zeros(v,n);
11
12 summary.k = zeros(1,v);
13 summary.mixMaxP = zeros(v,1);
14 summary.mixAvgP = zeros(v,1);
15 summary.mixMedP = zeros(v,1);
16 summary.mixMaxS = zeros(v,1);
17 summary.mixAvgS = zeros(v,1);
18 summary.mixMedS = zeros(v,1);
19
20 summary.numMaxP = zeros(v,1);
21 summary.numAvgP = zeros(v,1);
22 summary.numMedP = zeros(v,1);
23 summary.numMaxS = zeros(v,1);
24 summary.numAvgS = zeros(v,1);
25 summary.numMedS = zeros(v,1);
26 for i=1:v
27     if i==1
28         this = australian;
29     elseif i==2
30         this = heart;
31     elseif i==3
32         this = iris;
33     elseif i==4
34         this = lenses;
35     elseif i==5
36         this = vote;
37     end
38     now = this.mixedClust;
39
40     summary.k(i) = now.k;
41     summary.mixMaxP(i) = max(now.performance);
42     summary.mixAvgP(i) = mean(now.performance);
43     summary.mixMedP(i) = median(now.performance);
44     summary.mixMaxS(i) = min(now.silhouette);
45     summary.mixAvgS(i) = mean(now.silhouette);
46     summary.mixMedS(i) = median(now.silhouette);
47
48     now = this.numericClust;
49     summary.numMaxP(i) = max(now.performance);
50     summary.numAvgP(i) = mean(now.performance);
51     summary.numMedP(i) = median(now.performance);
52     summary.numMaxS(i) = min(now.silhouette);
53     summary.numAvgS(i) = mean(now.silhouette);
54     summary.numMedS(i) = median(now.silhouette);
```

```
55 end
56 clear cols
57 clear propnames
58 clear subname
59 clear this
60 clear v
61 clear varnames
62 clear i
63 clear ans
64 clear n
65 clear now
66 clear clustnames
67
68 figure
69 bar([summary.mixMaxP,summary.numMaxP]);
70 hold on
71 title 'Benchmark Maximum Performance Ratio'
72 hold off
73 fig2plotly();
74
75 figure
76 bar([summary.mixAvgP,summary.numAvgP]);
77 hold on
78 title 'Benchmark Mean Performance Ratio'
79 hold off
80 fig2plotly();
81
82 figure
83 bar([summary.mixMedP,summary.numMedP]);
84 hold on
85 title 'Benchmark Median Performance Ratio'
86 hold off
87 fig2plotly();
88 close all
```

APPENDIX D

Visualization of Data Point 2D and 3D

```
1  function pointClusterVis(clustering,trialnum)
2  %POINTCLUSTERVIS plots polar and cylinderical ...
       visualizations of clustering
3  %
4  % Input:
5  %     clustering             - struct
6  %             .k             - number of means
7  %             .idx           - cluster assignments ...
       (indicies) for each trial
8  %             .distances     - distance from each point ...
       to each center
9  %     trialnum               - trial to visualize ...
       (default 1)
10 %
11 % Author: Camden Glenn Bock
12 % 598 Bates College, Lewistion, ME 04240
13 % cbock@bates.edu, camdenbock@gmail.com
14 % http://www.camdenbock.com
15 % December 2015; Last Revision: 12/30/2015
16 %
17 %% Copyright (C) 2016  Camden Bock - GPL v. 3.0
18 %
19 % 'This program is liscensed under GPL v3.0'
20 % 'This program is modified from Ahmad Alsahaf`s ...
       package: ...
21 % amjams/mixedkmeans'.
22 %
23 % This program is free software: you can redistribute ...
       it and/or modify
24 % it under the terms of the GNU General Public License ...
       as published by
25 % the Free Software Foundation, either version 3 of the ...
       License, or
26 % any later version.
```

```
27 %
28 % This program is distributed in the hope that it will ...
      be useful,
29 % but WITHOUT ANY WARRANTY; without even the implied ...
      warranty of
30 % MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  ...
      See the
31 % GNU General Public License for more details.
32 %
33 % You should have received a copy of the GNU General ...
      Public License
34 % along with this program.  If not, see ...
      <http://www.gnu.org/licenses/>.
35 %
36 %
37 % 'This program comes with ABSOLUTELY NO WARRANTY;' ...
38 % 'for details type view source. This is free software, ...
      and' ...
39 % 'you are welcome to redistribute it display under ...
      certain' ...
40 % 'conditions; see <http://www.gnu.org/licenses/>', ...
41 % ('Copyright (C) 2016  Camden Bock, Bates College'))
42
43
44
45 %------------- BEGIN CODE --------------
46
47 if nargin < 2
48     trialnum = 1;
49 elseif nargin < 1
50     display('not enough arguments')
51 end
52
53 idx = clustering.idx(:,trialnum);
54 distances = clustering.distance(:,:,trialnum);
55 rownum = 1:length(idx);
56
57 [idx,I] = sort(idx);
58 distances = distances(I,:);
59
60 k = numel(unique(idx));
61 X = zeros(length(rownum),k+1);
62 Y = zeros(length(rownum),k+1);
63 Z = zeros(length(rownum),k+1);
64 C = zeros(length(rownum),k+1);
```

```
65
66 %plot regular polygon
67 ΔAngle = 2*pi/k;
68 theta = 0:ΔAngle:2*pi;
69
70 radiusReg = zeros(1,k+1);
71 for i=1:k+1
72     radiusReg(i) = max(max(distances));
73 end
74 figure
75 subplot(1,2,1)
76 if gpuDeviceCount > 0
77     thetaG = gpuArray(theta);
78     radiusRegG = gpuArray(radiusReg);
79     polar(thetaG,radiusRegG,'b');
80 else
81     polar(theta,radiusReg,'-b');
82 end
83 hold on
84 for i=1:length(rownum)
85     distancePoint = distances(rownum(i),:);
86     %plot datapoint point polygon
87     radiusPoint = zeros(1,k+1);
88     parfor j=1:k
89         radiusPoint(j) = distancePoint(j);
90     end
91     radiusPoint(k+1) = radiusPoint(1);
92     if gpuDeviceCount > 0
93         thetaG = gpuArray(theta);
94         radiusPointG = gpuArray(radiusPoint);
95         polar(thetaG,radiusPointG,'-r');
96     else
97         polar(theta,radiusPoint,'-r');
98     end
99     [x,y] = pol2cart(theta, radiusPoint);
100    X(i,:) = x;
101    Y(i,:) = y;
102    Z(i,:) = rownum(i);
103    for j=1:k
104        C(i,j) = idx(i);
105    end
106 end
107 hold off
108 subplot(1,2,2)
109 if gpuDeviceCount > 0
```

```
110     Xg = gpuArray(X);
111     Yg = gpuArray(Y);
112     Zg = gpuArray(Z);
113     Cg = gpuArray(C);
114     surf(Xg,Yg,Zg,Cg);
115 else
116     surf(X,Y,Z,C);
117 end
118 end
```

Bibliography

[1] Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, November 2007. URL: http://www.sciencedirect.com/science/article/pii/S0169023X0700050X, doi:10.1016/j.datak.2007.03.016.

[2] Ahmad Alsahaf. amjams mixed kmeans, 2015. URL: http://www.mathworks.com/matlabcentral/fileexchange/53489-amjams-mixed-kmeans.

[3] Alexander Andonian. Global Optima in sums of matrix elements and the Hungarian Algorithm, December 2015.

[4] Leon Bottou. Une Approche thorique de lApprentissage Connexioniste; applications la reconnaissance de la Parole. *Universite de Paris*, 1991. URL: http://www.iro.umontreal.ca/~pift6266/A06/refs/bottou-1991.pdf.

[5] Lon Bottou and Yoshua Bengio. Convergence Properties of the K-Means Algorithms. In *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press, 1995.

[6] P.S. Bradley and Usama M. Fayyad. Refining Initial Points for K-Means Clustering. *Microsoft Research*, January 1998. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=68490.

[7] Markus Buehren. Functions for the rectangular assignment problem, December 2004. URL: http://www.mathworks.com/matlabcentral/fileexchange/6543-functions-for-the-rectangular-assignment-problem/content/assignmentoptimal.m.

[8] Jadzia Cendrowska. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27(4):349–370, October 1987. URL: http://www.sciencedirect.com/science/article/pii/S0020737387800032, doi:10.1016/S0020-7373(87)80003-2.

[9] Neil Heffernan. Personal Interview, February 2016.

[10] Neil Heffernan, Cristina Heffernan, and Aviv Brest. WPI Fine Grained Skills - Skill Diagram, 2009. URL: http://teacherwiki.assistment.org/index.php/WPI_Fine_Grained_Skills.

[11] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahmi, Leonidas J Guibas, and Jascha Sohl-Dickstein. *Deep Knowledge Tracing*. Number 28 in Advances in Neural Information Processing Systems. Curran Associates, Inc., 2015. URL: http://papers.nips.cc/paper/5654-deep-knowledge-tracing.pdf.

[12] Simon Rogers and Mark Girolami. *A first course in machine learning*. CRC Press, Boca Raton, 2012.

[13] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*,

20:53–65, November 1987. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257, doi:10.1016/0377-0427(87)90125-7.

[14] Silicon Graphics International. SGI - MLC++: Datasets from UCI. URL: http://www.sgi.com/tech/mlc/db/.

[15] Jie Sun, Robert Freund, and Thomas Magnanti. *15.094J Systems Optimization: Models and Computation (SMA 5223), Spring 2004: Introduction to Convex Constrained Optimization*. (Massachusetts Institute of Technology: MIT OpenCourseWare, 2004. URL: http://ocw.mit.edu/.

[16] S Trivedi, Z.A Pardos, and N.T Heffernan. Clustering Students to Generate an Ensemble to Improve Standard Test Score Predictions. *Lecture notes in computer science.*, (6738):377–384, 2011.

[17] Zhenjie Zhang, Bing Tian, Dai Anthony, and K. H. Tung. *On the Lower Bound of Local Optimum in k-means Algorithm*. 2006.

GNU General Public License

The source code is freely available at:
`https://github.com/cam3715/mkmeans`

Cite this source code with the following paper:

MLA:
Bock, Camden Glenn, and Bonnie J. Shulman. "Mixed K-means Clustering in Computer Adaptive Learning." *SCARAB* (2016) *Department of Mathematics,* Bates College, May 2016. Web. http://scarab.bates.edu/.

APA:
Bock, C. G., & Shulman, B. J. (2016). Mixed K-means clustering in computer adaptive learning. *SCARAB*. Retrieved from http://scarab.bates.edu/.

GNU General Public License 3.0

http:///www.gnu.org/licenses/

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

(0) Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

(1) Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are

not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

(2) Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

(3) Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

(4) Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

(5) Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

(a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

(b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program

has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

(6) Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not

require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of

possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

(7) Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
(d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

(8) Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

(9) Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

(10) Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source

of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

(11) Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to

deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

(12) No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy

simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

(13) Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

(14) Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

(15) Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

(16) Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

(17) Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions