CENTAURUS: A Cloud Service for K-means Clustering

Nevena Golubovic, Angad Gill, Chandra Krintz, and Rich Wolski
Dept of Computer Science
Univ. of California, Santa Barbara

***Abstract*—We present CENTAURUS, a scalable, easy to use, cloud service and pluggable framework for k-means clustering that automatically deploys and executes multiple k-means variants concurrently, and then scores them to provide a clustering recommendation. CENTAURUS scores clustering results using Bayesian Information Criterion to determine the best model fit across cluster results. CENTAURUS visualization and diagnostic tools help users interpret clustering results. We empirically evaluate CENTAURUS and compare it to MZA, a popular desktop tool that uses k-means clustering to extract farm management zones from soil electroconductivity data. We show that CENTAURUS produces better results, is more scalable, and requires less guidance from the user.**

***Index Terms*—K-means Clustering, Mahalanobis, Cloud.**

I. INTRODUCTION

The environment in which we live is increasingly accessible via sensing, observation, and monitoring. As a result, data resources, and the opportunity to analyze them, has grown explosively as "analytics" have become critical to business, government, social, and scientific advances. As a result of this pervasiveness, analysis must now become a tool available to people with an ever widening range of expertise and skill sets.

Statistical clustering, also known as a separation of measurements into related groups, is a key requirement for solving many analytics problems. Lloyd's algorithm [1], commonly called *k-means*, is one of the most widely used approaches [2]. K-means is an unsupervised learning algorithm, requiring no training or labeling, that partitions data into K clusters, based on their "distance" from K centers in a multi-dimensional space. Its basic form is simple to implement and has become an indispensable component of pattern recognition, data mining, image processing, information retrieval, and recommendation applications across fields ranging from marketing and advertising to astronomy and agriculture.

While conceptually simple, there are a myriad of k-means algorithm variants based on how distances are calculated in the problem space. Some k-means implementations also require "hyper parameters" that control for the amount of statistical variation in clustering solutions. Identifying which algorithm variant and set of implementation parameters to use in a given analytics setting is often challenging and error prone for novices and experts alike.

In this paper, we present CENTAURUS as an approach to simplifying the application of k-means through the use of cloud-computing. CENTAURUS is a web-accessible, cloud-hosted service that automatically deploys and executes multiple k-means variants concurrently, producing multiple models. It then scores the models to select the one that best fits the data – a process known as model selection. It also allows for the experimentation with different hyper parameters and provides a set of data and diagnostic visualizations so that users can best interpret its results. From a systems perspective, CENTAURUS defines a pluggable framework into which clustering algorithms and k-means variants can be chosen. When users upload their data, CENTAURUS executes and automatically scales its k-means variants concurrently using public or private cloud resources. To perform model selection, CENTAURUS employs a scoring component based on information criteria. CENTAURUS computes a score for each result (across variants, cluster sizes, and repeat runs) and provides a recommendation of the best clustering to the user. Users can also employ CENTAURUS to visualize their data, its clusterings, and scores, and experiment with different parameterizations of the system (e.g., the number of repeat runs, the combination of features to cluster, and the dimensions to display).

We implement CENTAURUS using production-quality, open-source software and validate it using synthetic datasets with known clusters. We also apply CENTAURUS in the context of a real-world, agricultural analytics application and compare its results to the industry standard clustering approach. The application analyzes fine-grained soil electrical conductivity (EC) measurements, GPS coordinates, and elevation data from a field to produce a "map" of differing soil zones. These zones are then used by farmers and farm consultants to customize management of different zones on the farm (application of water, fertilizer, pesticides, etc.) [3]–[6]. We compare CENTAURUS to the state of the art clustering tool (MZA [3]) for farm management zone identification and show that CENTAURUS is more robust, obtains more accurate clusters, and requires significantly less input and effort from its users.

In the sections that follow, we motivate our work and discuss related research. We then describe the general form of the k-means algorithm and variants for computing covariance matrices that CENTAURUS employs (Section III). In Section IV, we detail CENTAURUS's model scoring, system architecture, and implementation. Finally, we present our datasets, an empirical evaluation of CENTAURUS, and our conclusions.

II. RELATED WORK

To design and implement CENTAURUS, we leverage Murphy [7]. This prior work identifies multiple ways of computing the covariance matrices and using them to determine distances and log likelihood.

The research and system that is most closely related to CENTAURUS, is MZA [3]. MZA is a computer program widely used by farmers to identify clusters in soil electroconductivity (EC) data to aid farm zone identification and

to optimize management. MZA uses fuzzy k-means [8], [9], computes the global covariance matrix, and employs either Euclidean, diagonal, or Mahalanobis distance to compute distance between points. MZA computes the covariance matrix based on all the data points and uses this same matrix in each iteration. MZA compares clusters using two different scoring metrics: fuzziness performance index (FPI) [10] and normalized classification entropy (NCE) [9].

We compare MZA and CENTAURUS in Section VI using synthetic data and show that CENTAURUS achieves a lower error percentage than MZA. Moreover, CENTAURUS resolves many of the limitations of MZA (which is only available as desktop software, does not account for poor initial cluster assignments, and places a burden on the user to determine which cluster size and k-means variant to employ).

EZZone [11] is a web service that is similar to CENTAURUS in that it provides univariate clustering based on the Jenks natural break optimization [12]. Users input the number of clusters and the tool reports the Goodness of Variance Fit, a measure of the homogeneity of the clusters, as a metric of how well the clustering performed.

The authors of x-means [13] use Bayesian Information Criterion (BIC) [14] (which CENTAURUS also employs) as a score for the univariate normal distribution. The authors do not discuss how to extend the algorithm and scoring to multivariate distributions however. CENTAURUS provides six different ways of computing covariance matrix for k-means for multivariate data and examples that illustrate the differences. CENTAURUS is also pluggable enabling other algorithms to be added and compared.

III. K-MEANS CLUSTERING

The goal of k-means clustering algorithm is to separate similar data points (represented as vectors) into K clusters for a given K. In its basic form (that is, assuming equal cluster variances), it attempts to find a clustering that minimizes the sum of the squared distances of each point to the center of its cluster. The algorithm begins with an initial set of K centers and alternates between assigning points to the cluster represented by the nearest center, and recomputing the centers.

Finding the optimal assignment is NP-hard. However, it is typically possible to find a local optimum quickly by terminating the algorithm when cluster assignments do not change from iteration to iteration. In this case, the choice of starting centers determines the specific local optimum the algorithm will reach. Thus the termination state of the algorithm is dependent on the initial assignment. Note, also, that the sum-of-squared distances within from data points to the center of their assigned cluster provides a way to compare local optima – the lower the sum of the distances, the closer to a global optimum a specific clustering is.

Note, also, that it is possible to use different measures of distance to account for per-cluster differences in variance, or co-variance between measurements (e.g. Mahalanobis distance [15]). Thus, for a given data set, the algorithm can generate a number of different k-means clusterings – one for each combination of starting center and distance measure.

More generally, k-means is equivalent to implementing Expectation Maximization under the assumptions of a Gaussian Mixture Model (GMM) with "hard" cluster assignment (i.e. for each point there is a cluster to which it belongs with probability 1.0) [2], [16], [17]. As such, the centers correspond to the maximum likelihood estimates of the cluster means. For this reason it is possible to use information criteria based on maximum log-likelihood (e.g. the Bayesian Information Criterion [14] or the Akaike Information Criterion [18]) to compare the local optima generated from different variants of k-means and, ultimately, to choose the "best" one under the assumptions of the GMM [13]. We discuss the use of information criteria as a "scoring" method across multiple runs of multiple variants in Section IV-A.

The two most common techniques for measuring distance between points are Euclidean [19] and Mahalanobis [15] metrics. Euclidean distance is the straight line distance between two points and assumes that the dimensions of the space are orthogonal. However, dimensions often correspond to measurements that are correlated and it is possible to transform the feature space using Mahalanobis distance to correct for inter-dimensional correlation.

We implement k-means in its general form using Mahalanobis distance in CENTAURUS using the following steps:

1) Randomly select K points from the data and assign these as the initial cluster centers $\boldsymbol{\mu}^{(k)}$, where K is the number of clusters, k is the cluster index, and $k = 1, \ldots, K$.
2) Compute initial covariance matrix $\boldsymbol{\Sigma}$ using all data points:
$$\Sigma_{ij} = \frac{1}{n} \sum_{p=1}^{n} (x_i^{(p)} - \mu_i)(x_j^{(p)} - \mu_j)$$
where, Σ_{ij} is (i, j)-th component of the matrix $\boldsymbol{\Sigma}$, $x_i^{(p)}$ is the i-th component of the p-th data point, and μ_i is the i-th component of the global mean.
3) Assign all the points to the closest cluster center using Mahalanobis distance metric:
$$D(\boldsymbol{x}^{(p)}, \boldsymbol{\mu}^{(k)}) = \left((\boldsymbol{x}^{(p)} - \boldsymbol{\mu}^{(k)})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}^{(p)} - \boldsymbol{\mu}^{(k)}) \right)^{-1/2}$$
where, $\boldsymbol{\mu}^{(k)}$ is the center of the k-th cluster.
4) Compute covariance matrix $\boldsymbol{\Sigma}^{(k)}$ for each cluster using their cluster center $\boldsymbol{\mu}^{(k)}$.
5) Compute the cluster centers: For all the points in a cluster, calculate the sum of its distances to all the other points in the same cluster. Assign the point with the minimum sum as the new cluster center, $\boldsymbol{\mu}^{(k)}$.
6) Repeat (4) and (5) until convergence or completion of a maximum number of iterations. The convergence criteria is calculated by summing up the distances of new cluster centers from the old cluster centers.

The covariance matrix represents the covariance and variance observed in a sample between the dimensions of the dataset. There are multiple ways to compute the covariance

matrix in step 4 of the algorithm [7], [16], [17], [20], each of which has bearing on the output of the algorithm.

The most common methods for computing $\mathbf{\Sigma}$ are:

- *Full*: Compute the entire covariance matrix $\mathbf{\Sigma}$ and use all of its elements to compute distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$D(\boldsymbol{x}, \boldsymbol{y}) = \left((\boldsymbol{x}-\boldsymbol{y})^T \mathbf{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{y})\right)^{-1/2}$$

  This variant is commonly associated with the use of Mahalanobis distance.
- *Diagonal*: Compute the variance matrix, i.e., the covariance matrix with its off-diagonal elements set to zero. If the data are orthogonal, the full covariance matrix is a diagonal matrix with the non-zero elements representing the variance across dimensions. This approach ignores the covariance observed between the dimensions of the dataset.
- *Spherical*: Set the diagonal of the covariance matrix to the variance computed across all dimensions and set all off-diagonal elements to zero. This variant assumes that the data is best represented by a GMM in which each Gaussian (corresponding to each cluster) has a single variance across dimensions. It is also commonly referred to as using Euclidean distance.

In addition, each of these approaches for computing the covariance matrix can be *Tied* or *Untied*. *Tied* means that we compute a covariance matrix per cluster, take the average across all clusters, and then *use the averaged covariance matrix* to compute distance. *Untied* means that we compute a *separate covariance matrix for each cluster*, which we use to compute distance. Using a tied set of covariance matrices assumes that the covariance among dimensions is the same across all clusters, and that the variation in the observed covariance matrices is due to sampling variation. Using an untied set of covariance matrices assumes that each cluster is different in terms of its covariance between dimensions.

CENTAURUS considers all six combinations of methods for computing covariance matrices: *Full-Tied*, *Full-Untied*, *Diagonal-Tied*, *Diagonal-Untied*, *Spherical-Tied*, and *Spherical-Untied*. The output of the algorithm is a list of cluster labels, one per data point, indicating the cluster index to which the data point belongs.

Once the labels are computed for each data point, we can compute the likelihood (a function of the data given the model) using the equation for GMM with hard assignment [16], [17], as:

$$f\left(\boldsymbol{X}|\boldsymbol{\mu}, \mathbf{\Sigma}\right) = \prod_{p=1}^{n} \prod_{k=1}^{K} \pi_k^{\mathbb{1}_{pk}} \cdot \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}^{(k)}, \mathbf{\Sigma}^{(k)}\right)^{\mathbb{1}_{pk}}$$

where, p is a data point, k is a cluster index, π_k is the ratio of the number of points in cluster k and the total number of points, and $\mathbb{1}_{pk}$ is an identity coefficient that is 1 if the point p belongs to the cluster k and 0 otherwise, $\boldsymbol{\mu}^{(k)}$ is the k-th cluster center.

The log-likelihood function is needed to compute information criteria that CENTAURUS uses to score a particular clustering. We compute the log-likelihood function as:

$$\begin{aligned} & l\left(\boldsymbol{X}|\boldsymbol{\mu}, \mathbf{\Sigma}\right) = \ln f\left(\boldsymbol{X}|\boldsymbol{\mu}, \mathbf{\Sigma}\right) \\ = & \sum_{k=1}^{K} n_k \cdot \left(\ln\left(\frac{n_k}{n}\right) - \frac{d}{2}\ln\left(2\pi\right) - \frac{1}{2}\ln|\mathbf{\Sigma}^{(k)}|\right) \\ & - \frac{1}{2}\sum_{p=1}^{n}\sum_{k=1}^{K} \mathbb{1}_{pk} \cdot (\boldsymbol{x}^{(p)} - \boldsymbol{\mu}^{(k)})^T (\mathbf{\Sigma}^{(k)})^{-1} (\boldsymbol{x}^{(p)} - \boldsymbol{\mu}^{(k)}) \end{aligned}$$

IV. THE CENTAURUS SYSTEM

CENTAURUS implements a service for k-means clustering that takes advantage of cloud-based, large-scale distributed computation, automatic scaling (where computational resources are added or removed on-demand), data management to support visualization, and browser-based user interaction. The system implements the six different variants of k-means (described in Section III) and runs them each for a succession of values of K ranging from 1 to a user-assigned large number, *max_k*. For each clustering, CENTAURUS computes a pair of scores based on both the Bayesian Information Criterion (BIC) [14] and the Akaike Information Criterion (AIC) [18]. It also allows the user to change the number of independent, randomly seeded runs (referred to as *experiments*) to account for statistical variation. Finally, it provides ways for the user to graph and visualize both two-dimensional "slices" of all clusterings as well as the relative BIC and AIC scores. It also implements a decision support feature in which the "best" clustering is identified based on BIC score across all variants.

CENTAURUS is extensible in that different clustering algorithms can be "plugged in" easily, and automatically deployed with and compared against others. For this work, we plug in the k-means variants described in the previous section. The variants include different distance computations (Euclidean and Mahalanobis), input data scaling (e.g. whether or not to scale each dimension to have zero mean and unit standard deviation), and the six combinations of covariance matrices.

A. CENTAURUS *Scoring*

CENTAURUS performs N experiments for a particular K value (where $K = 1, ..., max\_k$), each of which consists of M initial cluster assignments to the k-means algorithm. Each algorithm iterates until convergence or a maximum number of iterations is reached (in CENTAURUS this value is 300). Thus, CENTAURUS executes $N * M$ runs of an algorithm for each value of K. Across M initial cluster assignments, CENTAURUS chooses the best performing one using the maximum log likelihood.

The scoring component takes label assignments from a clustering result for a particular K value and returns a score. CENTAURUS then computes the average score (across the N experiment runs) and uses it as part of its recommendation and visualization services.

We currently integrate two different information criteria as plug-ins to CENTAURUS: BIC and AIC. BIC and AIC measure

the goodness of fit of an estimated statistical model. In our case, we use them to measure the fit of the models (clustering results) that are output from the various k-means algorithms that CENTAURUS implements. When a user requests a single recommendation, CENTAURUS uses the BIC score to make this recommendation.

We compute the BIC score for a model with K clusters as:

$$BIC_K = l(\boldsymbol{X}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) - \frac{r_K}{2} \log n$$

where, $\hat{\boldsymbol{\mu}}$ is the maximum likelihood estimator for the cluster centers, $\hat{\boldsymbol{\Sigma}}$ is the maximum likelihood estimator for the cluster covariance matrices, $l(\boldsymbol{X}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ is the maximum log likelihood, and r_K is the number of free parameters in the model. r_K is computed as the sum of $K-1$ cluster probabilities (π_k), $K \cdot d$ coordinate parameters for all the cluster centers, and $\frac{d \cdot (d+1)}{2}$ parameters for a symmetric cluster covariance matrix:

$$r_K = (K-1) + K \cdot d + \frac{d(d+1)}{2}$$

Similarly, we compute the AIC score for a model with K clusters as:

$$AIC_K = l(\boldsymbol{X}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) - r_K$$

Note that because these techniques require estimates of the covariance matrix for each cluster, there must be a minimum number of data points per cluster for this estimate to be meaningful. As a result, CENTAURUS discards (does not score or consider in the scoring average) any clustering result which has one or more clusters with fewer elements than this minimum. This minimum threshold is user configurable with a default setting of 10 data points in the current system.

B. Implementation

The CENTAURUS implementation consists of a user-facing web service and distributed cloud-enabled backend. Users upload their datasets to the web service Frontend as files in a simple format: as comma-separated values (CSV files). Users can then modify the following CENTAURUS parameters:

- *n_exp*: The number of experiments (N) per K to run. The default is 3 with a minimum of 1 and maximum of 100.
- *max_k*: Maximum number of clusters to fit to the data. CENTAURUS runs a set of experiments for clusters of size 1 through max_k. The default is 10 with a minimum of 1 and maximum of 15.
- *n_init*: Number of times to initialize the k-means clustering (M). The default is 10 with a minimum of 1 and maximum of 100.
- *covars*: The type(s) of covariance matrix to use for the analysis. All options – *Full-Tied*, *Full-Untied*, *Diagonal-Tied*, *Diagonal-Untied*, *Spherical-Tied*, and *Spherical-Untied* – are selected by default.
- *scale*: Scale the data so that each dimension has zero mean and unit standard deviation. This option is selected by default.

CENTAURUS considers each parameterization that the user chooses (including the default) as a "job". Each job consists of multiple tasks (experiment runs) that CENTAURUS deploys. Users can also use the service to check the status of a job or to view the report for a job (when completed). The status page provides an overview of all the tasks for a job showing a progress bar for the percentage of tasks completed and a table showing task parameters and outcomes.

CENTAURUS uses the report page to provide its recommendation. The recommendation consists of the number of clusters and k-means variant that produces the best BIC score. This page also shows the cluster assignments, spatial plots using longitude and latitude (if included in the original data set), BIC and AIC scores plots. Finally, CENTAURUS provides cluster labels in CSV files that the user can download.

C. System Architecture

The software architecture of CENTAURUS is shown in Figure 1. We implement CENTAURUS using Python v3.4.3 and integrate a number of open source software, packages, and cloud services. The cloud system is a private cloud that runs Eucalyptus software v4.4 [21], [22] and integrates virtual servers with 2 CPUs and 1GB of memory each. CENTAURUS consists of five primary components:

1) *Frontend*: We couple the Python Flask [23] (v0.12.1) web framework with Gunicorn [24] (v19.7.1) web server and NGINX [25] (v1.4.6) reverse proxy server to provide a robust application hosting service.
2) *Backend Worker*: We use Python Celery [26] (v4.0.2), a distributed computation framework, to perform analysis computation tasks asynchronously and at scale [27]. We leverage autoscaling groups in Eucalyptus to automatically grow and shrink the number of workers performing the computation according to the demand for each job.
3) *Backend Queue*: We use RabbitMQ [28] (v3.2.4) message broker to send information about each job from the Frontend to the Workers. This enables to Frontend to quickly off-load its work to the Queue where it is sent systematically to the Workers as and when they become available.
4) *Backend Database*: We use MongoDB [29] Community Edition (v3.4.4) database to store parameters and results for jobs and tasks.
5) *Backend File Store*: We use Amazon Simple Storage System (S3) [30] to store files uploaded by users.

Other packages that CENTAURUS leverages include Numpy [31] (v1.12.1), Pandas [32] (v0.19.2), SciKit-Learn [33] (v0.18.1), and SciPy [34] (v0.19.0) for data processing. CENTAURUS uses Matplotlib [35] (v2.0.1) and Seaborn [36] (v0.7.1) to provide data visualization and plots.

V. DATASETS

We use both synthetic and real-world datasets to evaluate CENTAURUS empirically. We generate the synthetic data sets with known clusters (as "ground truth"), which we use to validate and measure the accuracy of the CENTAURUS implementation. Using the real-world application data from precision agriculture, we also compare the results generated

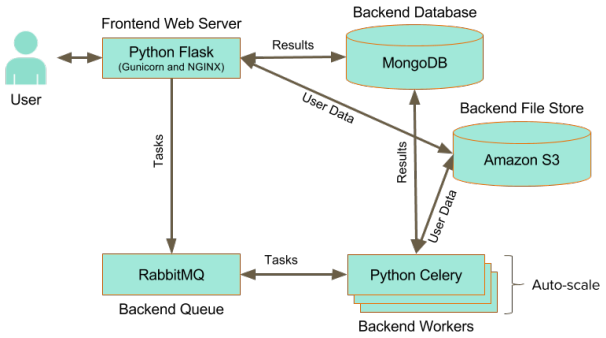
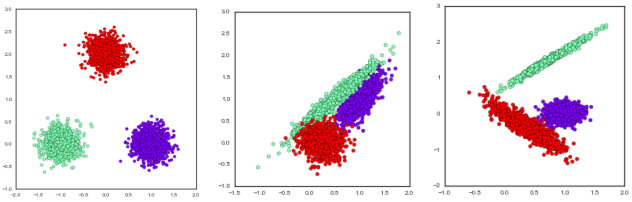


Fig. 1: System architecture for CENTAURUS.



(a) Dataset-1 (b) Dataset-2 (c) Dataset-3

Fig. 2: Synthetic datasets shown with ground truth assignment.

by CENTAURUS for management zone determination to the industry standard and use them to illustrate the CENTAURUS visualization capabilities.

A. Synthetic Datasets

We first create multiple 2-dimensional synthetic datasets using multivariate Gaussian distributions. The datasets have three clusters with 1,000 points per cluster and varying degrees of inter-dimensional correlation in each cluster. Figure 2 shows these datasets with their ground truth cluster assignments.

The clusters in *Dataset-1* have no correlation and have equal standard deviations of 0.2 for each dimension. Cluster centers are set at positions $(1, 0)$, $(-1, 0)$, and $(0, 2)$ as can be seen in Figure 2a.

In *Dataset-2*, the cluster centered at $(0.25, 0)$ has two dimensions that are independent (not correlated) with the same standard deviations of 0.2. The cluster centered at $(1, 1)$ has correlation 0.70 between dimensions, while the cluster centered at $(0.5, 1)$ has correlation 0.97 between dimensions. When all three clusters are combined the correlation between the two dimensions is 0.75.

In *Dataset-3*, the cluster centered at $(1, 0)$ has two independent dimensions, while the clusters centered at $(0.75, 1.5)$ and $(0.35, -0.35)$ have correlations of 0.98 and -0.89 respectively. The correlation of the entire set is 0.2.

B. Application Datasets

The farm data that we use consists of measurement of electrical conductivity (EC) of soil measured (at 30cm and 90cm depths) using an instrument manufactured by Veris Technologies Inc. [37]. The surveyor collects EC as well as the GPS cordinates and elevation data associated with each EC measurement. The approach produces a data file containing

Variant	Dataset-1	Dataset-2	Dataset-3
Full-Untied	0.0%	3.6%	0.1%
Full-Tied	0.0%	37.6%	57.5%
Diagonal-Untied	0.0%	26.2%	26.0%
Diagonal-Tied	0.0%	34.4%	55.2%
Spherical-Untied	0.0%	27.3%	11.2%
Spherical-Tied	0.0%	34.4%	56.6%

TABLE I: Percentage error (out of 3,000 points per dataset) for the six k-means variants of CENTAURUS for the synthetic datasets. Values are the percentage of points incorrectly labeled by the variant (i.e. assigned to the wrong cluster).

five dimensions of data: longitude, latitude, elevation, EC at 30cm depth (EC1), and EC at 90cm depth (EC2).

The EC datasets come from three different farms. *Cal Poly*: a 12-acre lemon field at California Polytechnic State University, San Louis Obispo, California for which we have 3,233 data points. *UNL*: a 91-acre field at University of Nebraska, Lincoln, for which we have 5,823 data points. *Sedgwick*: a 30-acre field located in the Santa Ynez Valley, California for which we have 7,920 data points.

VI. RESULTS

To evaluate the efficacy of CENTAURUS, we deploy the service and run it on the datasets described in Section V for each of the k-means variants described in Section III. In this section, we refer to the variants as *Full-Untied*, *Full-Tied*, *Diagonal-Untied*, *Diagonal-Tied*, *Spherical-Untied*, and *Spherical-Tied*.

For the results that follow, we parameterize CENTAURUS with $K = 1, \ldots, 10$ and 100 experiments each with 100 random initial cluster center assignments (for a total of 10,000 k-means algorithm invocations per variant). CENTAURUS stores the cluster assignments (labels) for each experiment, which is the result with the largest log-likelihood value across initial assignments. This CENTAURUS instance only considers clustering results when all clusters have at least 10 points, in its computation of BIC and AIC. Finally, as described above, CENTAURUS reports the result with the highest average BIC score the "best" clustering across every K considered for all variants.

A. Validation Using Synthetic Data

For the datasets with known clusters (those that we have generated synthetically) we report classification percentage error, i.e. the percentage of incorrectly classified points out of all the points in the dataset (3,000 data points per dataset in this case). Table I shows these results for each of the synthetic datasets (Dataset-1, Dataset-2, and Dataset-3) for each of the six k-means variants.

Note that Dataset-1 was generated using a GMM where all dimensions are independent of each other and are identically distributed. Thus the "perfect" classification results (0% error) generated by the Full and Diagonal methods indicate that they correctly disregard any observed sample variance or covariance.

The results for Full-Untied with Dataset-2 and Dataset-3 illustrate CENTAURUS 's ability to correct for cross-dimensional

correlation. The generating GMM in both cases is untied (i.e. each cluster has a distinct covariance matrix). Also, unlike in Dataset-1 where there are three distinct clusters with separated centers, we purposefully placed the cluster centers of Dataset-2 and Dataset-3 near each other and generated distributions that overlap. Doing so poses challenges for k-means clustering and all variants misclassified some points.

B. Real Datasets

We have used CENTAURUS to analyze the farm EC datasets collected from the Cal Poly, UNL, and Sedgwick locations. These datasets consist of five features: longitude, latitude, EC1, EC2, and elevation. For reasons of brevity, we present an analysis of the data only from Cal Poly as it is representative of all of our experiments.

The left-hand side of Figure 3 shows CENTAURUS's visualization of the label assignments generated for the six variants, with the best one shown in bold. The longitude and latitude dimensions are used for plotting as the x-axis and y-axis, respectively, by default when included in the uploaded dataset. The clustering for these results is based on the EC2 and elevation dimensions.

The right-hand side of Figure 3 graphs the corresponding BIC/AIC scores for the Cal Poly clusters in the six graphs shown on the left. In these graphs, the x-axis is the number of clusters (K) and the y-axis is BIC and AIC score. The values are the average score across the 100 experiments. Error bars indicate the 95% confidence interval for BIC and AIC values over all repeated runs as determined by the Python Seaborn package. Higher scores indicate a better model fit. Missing values are due to experiments that result in clusters with fewer than 10 points across all 100 experiments. CENTAURUS omits these experiments in its computation of BIC and AIC score since there are too few values in one or more clusters to compute covariance in a manner that is trustworthy statistically.

C. Visualization Customization

In addition to reporting a recommendation (a data clustering that results in the best BIC score), CENTAURUS enables users to customize the clustering computation and their visualization of results. For example, a user of CENTAURUS can select any two plotting dimensions to visualize the cluster assignments. The graphs shown above use longitude and latitude dimensions as the plotting dimensions. Figure 4 shows the same clustering assignment as Figure 3, potted using dimensions EC2 and elevation. This visualization shows the clustering that the k-means variants "saw" when they clustered the data.

Figure 4 illustrates both the challenges to developing a fully automated EC mapping technique based on clustering, and the utility of CENTAURUS as a decision support tool. For the Full-Untied case, the best BIC score shows two clusters separated by what appears to be a hard linear boundary. Comparing this case to the Full-Tied case (where there is a single cluster) one either sees two centroids or not. Thus while a novice with no domain experience may have no choice but to trust the BIC score as identifying the best clustering, a more informed user can use these visualizations to support domain-specific knowledge. In this case, for example, soil samples taken from the field as well as records detailing the history of how the field has been used over time point to the validity of the two-cluster mapping.

D. Comparison with MZA

We next compare CENTAURUS against MZA for the synthetic datasets. We use the number that both FPI and NCE scores report for MZA as the optimal number of clusters. We then use the respective cluster assignment (labels) to compute the error rates. Figure 5 shows the best assignments produced by CENTAURUS and MZA and Table II shows the percentage of incorrectly classified points (out of 3,000 points) in each dataset, for the same assignments.

For MZA, the best assignment is achieved by Mahalanobis distance and for CENTAURUS the best assignment is achieved by Full-Untied. MZA clusters the Dataset-1 correctly and reports $K = 3$ as the ideal number of clusters (as does CENTAURUS).

For Dataset-2, MZA correctly identifies $K = 3$ but has a higher error rate of 13.8% than CENTAURUS' 3.6%. A possible reason for this is that MZA only considers a single initial assignment of cluster centers, which in this case converges to a local minimum that is different from the global minimum. CENTAURUS avoids this kind of error by performing several runs (10,000 in this case, specified by *n_exp* $\times$ *n_init*) of k-means algorithm before suggesting the optimal cluster assignment.

Dataset-3 consists of clusters with correlation across features. CENTAURUS provides better results than MZA for this dataset, achieving a percentage error of only 0.1% compared to MZA's 11.6%. A possible reason for this is that MZA employs a global covariance matrix and does not consider Tied and Untied options as CENTAURUS does, which results in better label assignments.

Another limitation of MZA is that it uses a free variable, called the fuzziness parameter, and multiple scoring techniques. It is challenging (especially for novices) to determine how to set the fuzziness value even though the results are highly sensitive to this value. For the results in this section, we chose the default fuzziness parameter of $m = 1.3$ as suggested by the author [10].

Furthermore, for the farm datasets, the MZA scoring metrics (NCE and FPI) do not aways agree, providing conflicting recommendation and forcing the user to choose the best clustering. In combination, these limitations make MZA hard to use as a recommendation service for growers who lack the data science background necessary to interpret its results. CENTAURUS addresses these limitations by providing high enough number of k-means runs, no free parameters, and more sophisticated ways of computing the covariance matrix in each iteration of its clustering algorithm. It uses a unique scoring method to decide what is a single best clustering that will be presented to a novice user while it provides the diagnostic capabilities that are needed for more advanced users.

Finally, we visually compare the cluster maps that MZA

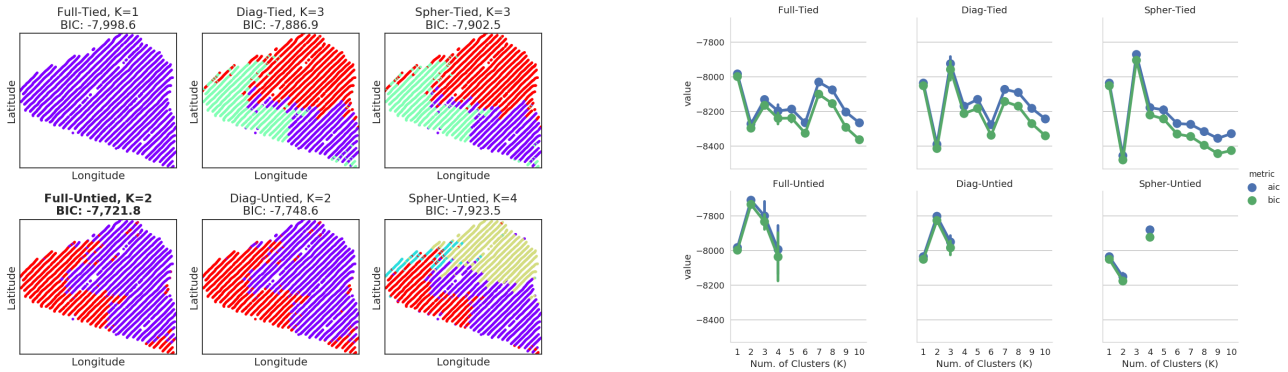


Fig. 3: CENTAURUS clustering results for all six variants (left six graphs) of EC2 and elevation dimensions from the Cal Poly farm data, plotted using longitude and latitude. The best performing number of clusters (K) for each variant is shown above each graph and the best out of the six is shown in bold. The right six graphs show the BIC and AIC scores for the corresponding clusterings.

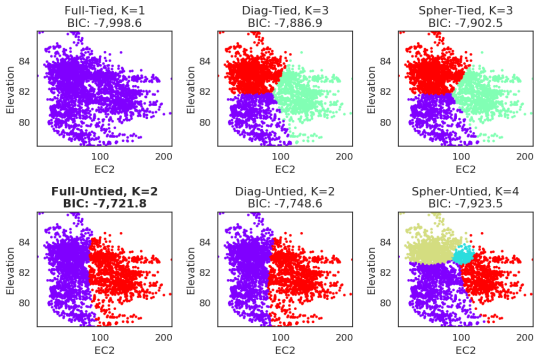
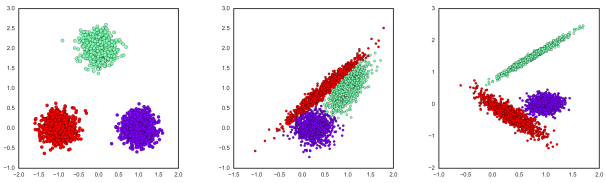
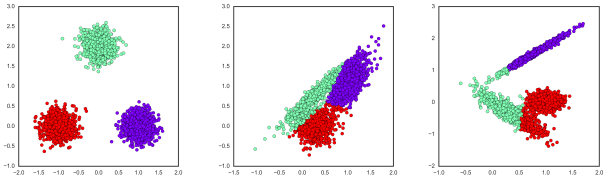


Fig. 4: CENTAURUS clustering results for Cal Poly EC2 and elevation data plotted using EC2 and elevation dimensions.



(a) CEN.: Dataset-1 (b) CEN.: Dataset-2 (c) CEN.: Dataset-3



(d) MZA: Dataset-1 (e) MZA: Dataset-2 (f) MZA: Dataset-3

Fig. 5: CENTAURUS vs. MZA clustering recommendations for the synthetic datasets.

generates with those generated by CENTAURUS. Figure 6 shows the MZA clustering of EC2 and elevation using longitude and latitude as the plotting dimensions, Mahalanobis distance, and a fuzziness exponent of 1.3.

	Dataset-1	Dataset-2	Dataset-3
CENTAURUS	0.0%	3.6%	0.1%
MZA	0.0%	13.8%	11.6%

TABLE II: Percentage error (out of 3,000 data points per dataset) for CENTAURUS and MZA on the synthetic datasets for the clustering results in Figure 5.

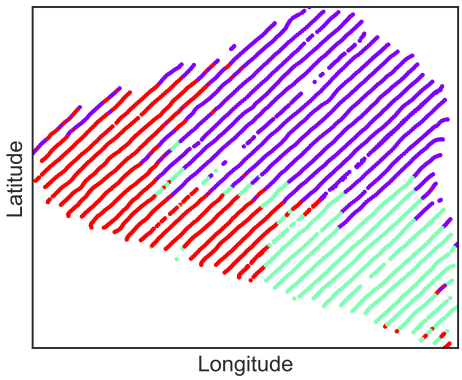


Fig. 6: Clustering assignment for Cal Poly dataset produced by MZA based on EC2 and elevation.

For this data set, MZA indicates three clusters using both FPI and NEC. Curiously, even though the resulting MZA mapping used Mahalanobis distance, it appears (visually) to be quite similar to the Diagonal-Tied and Spherical-Tied CENTAURUS mappings (both which also indicate three clusters based on BIC score) shown in Figure 3. MZA appears to choose a clustering of the Cal Poly data that corresponds to a lower BIC score than the one identified as being "best" by CENTAURUS (Full-Untied in Figure 3). Using k-means as a reference, MZA parameterized with a fuzziness exponent of 1.3 appears to "see" the Cal Poly data as having no meaningful covariance between EC2 and elevation in each cluster.

Note that the BIC scores for the Full-Untied clustering incur the highest "penalties" in the computation of a BIC score

among the six k-means variants. Recall that the formulation of BIC used by CENTAURUS (cf Section IV) attempts to avoid overfitting by subtracting $r_K * log(n)$ from the maximum log-likelihood estimate for a given clustering. For the Full-Tied case, r_K must account for the $\frac{d^2-d}{2}$ off-diagonal covariance estimates where in the Diagonal and Spherical cases, it does not (i.e. the "penalty" for estimated parameters is lower). Thus if the visual comparison is accurate, CENTAURUS finds a clustering (Full-Untied) that has a better BIC than MZA even with the additional penalty for the needed covariance estimates. Since both are making a fundamental GMM assumption about the data, CENTAURUS appears (based on information criteria) to find a better clustering. It may be that with a different fuzziness exponent MZA and CENTAURUS converge on a "best" clustering but we have yet to determine whether this convergence does, indeed, occur and, if it does, the best method for finding the fuzziness exponent that results in a consistent clustering between the two approaches.

VII. CONCLUSION

With this work, we present CENTAURUS, a scalable, easy to use, cloud service for clustering multivariate and correlated data. CENTAURUS simplifies selection of k-means clustering variants, provides a recommendation of the best variant, and enables users to visualize their results in multiple ways. CENTAURUS leverages cloud resources and services to automatically deploy, scale, and score k-means clustering jobs. We empirically evaluate CENTAURUS using synthetically generated and real datasets and compare it to the popular MZA clustering tool. Our results show that CENTAURUS provides better results than MZA and precludes many of its limitations.

In future work, we plan to extend CENTAURUS with other data analysis methods (DBSCAN, Spectral Clustering, GMM, etc.) and metrics for model evaluation and selection (Silhouette Score, Rand Index, Mutual Information, etc.). We also intend to incorporate other publicly available datasets (e.g., SSURGO) to improve clustering methods when applicable.

Acknowledgments. This work is funded in part by NSF (CNS-1703560, OAC-1541215, CCF-1539586, CNS-1218808, CNS-0905237), NIH (1R01EB014877-01), ONR NEEC (N00174-16-C-0020), Huawei Technologies, and the California Energy Commission (PON-14-304). We also thank Drs. Roberts, Sethuramasamyraja, and Yager from California State University, Fresno, Dr. Kitchen from the USDA, and Dr. Oh from the Univ. of California, Santa Barbara for their support and guidance on this project.

REFERENCES

[1] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
[2] R. O. Duda, P. E. Hart, D. G. Stork, *et al.*, *Pattern classification*, vol. 2. Wiley New York, 1973.
[3] J. J. Fridgen, N. R. Kitchen, K. A. Sudduth, S. T. Drummond, W. J. Wiebold, and C. W. Fraisse, "Management zone analyst (mza)," *Agronomy Journal*, vol. 96, no. 1, pp. 100–108, 2004.
[4] F. Moral, J. Terrón, and J. M. Da Silva, "Delineation of management zones using mobile measurements of soil apparent electrical conductivity and multivariate geostatistical techniques," *Soil and Tillage Research*, vol. 106, no. 2, pp. 335–343, 2010.
[5] R. Fortes, S. Millán, M. Prieto, and C. Campillo, "A methodology based on apparent electrical conductivity and guided soil samples to improve irrigation zoning," *Precision agriculture*, vol. 16, no. 4, pp. 441–454, 2015.
[6] D. Corwin and S. Lesch, "Application of soil electrical conductivity to precision agriculture," *Agronomy Journal*, vol. 95, no. 3, pp. 455–471, 2003.
[7] K. P. Murphy, "Fitting a conditional linear gaussian distribution," 1998.
[8] J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of cybernetics*, vol. 4, no. 1, pp. 95–104, 1974.
[9] J. C. Bezdek, *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.
[10] I. Odeh, D. Chittleborough, and A. McBratney, "Soil pattern recognition with fuzzy-c-means: application to classification and soil-landform interrelationships," *Soil Science Society of America Journal*, vol. 56, no. 2, pp. 505–516, 1992.
[11] C. Lowrance, S. Fountas, V. Liakos, and G. Vellidis, "Ezzone–an online tool for delineating management zones," *International Conference on Precision Agriculture*, 2016.
[12] G. F. Jenks, "The data model concept in statistical mapping," *International yearbook of cartography*, vol. 7, no. 1, pp. 186–190, 1967.
[13] D. Pelleg, A. W. Moore, *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters.," in *ICML*, vol. 1, pp. 727–734, 2000.
[14] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, no. 2, 1978.
[15] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.
[16] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.
[17] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.
[18] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.
[19] T. L. Heath *et al.*, *The thirteen books of Euclid's Elements*. Courier Corporation, 1956.
[20] A. Cerioli, "K-means cluster analysis and mahalanobis metrics: a problematic match or an overlooked opportunity," *Statistica Applicata*, vol. 17, no. 1, 2005.
[21] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pp. 124–131, IEEE, 2009.
[22] "Aristotle Cloud Federation." https://federatedcloud.org [Online; accessed 22-Aug-2016].
[23] "Python flask." http://flask.pocoo.org/ [Online; accessed 9-June-2017].
[24] "Gunicorn." http://gunicorn.org/ [Online; accessed 9-June-2017].
[25] "Nginx." https://nginx.org/ [Online; accessed 9-June-2017].
[26] "Celery." http://www.celeryproject.org/ [Online; accessed 9-June-2017].
[27] M. Lunacek, J. Braden, and T. Hauser, "The scaling of many-task computing approaches in python on cluster supercomputers," in *IEEE Cluster Computing*, pp. 1–8, 2013.
[28] "Rabbitmq." https://www.rabbitmq.com/ [Online; 9-Jun-2017].
[29] "Mongodb." https://www.mongodb.com/ [Online; 9-Jun-2017].
[30] "Amazon s3." https://aws.amazon.com/s3/ [Online; 9-Jun-2017].
[31] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.
[32] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, van der Voort S, Millman J, 2010.
[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.
[34] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online; accessed 9-June-2017].
[35] J. Hunter, "Matplotlib: A 2d graphics environment. computing in science& engineering 9, 90–95 (2007)," 2007.
[36] "Seaborn: statistical data visualization." http://seaborn.pydata.org/index.html [Online; accessed 9-June-2017].
[37] "Veris tech. inc.." http://www.veris.com/ [Online; 2-Jun-2017].