A scalable system for executing and scoring K-means clustering techniques and its impact on applications in agriculture

Nevena Golubovic*, Chandra Krintz and Rich Wolski

Department of Computer Science,
University of California, Santa Barbara,
5112 Harold Frank Hall Santa Barbara, CA, 93106, USA
Email: nevena@cs.ucsb.edu,
Email: ckrintz@cs.ucsb.edu,
Email: rich@cs.ucsb.edu
*Corresponding author

Balaji Sethuramasamyraja

Department of Industrial Technology,
California State University,
Jordan College of Ag Sciences and Technology,
Fresno 2255 East Barstow Avenue,
M/S IT9, Fresno, CA, 93740, USA
Email: balajis@csufresno.edu

Bo Liu

BioResource and Agricultural Engineering Department,
California Polytechnic State University,
8-106, 1 Grand Ave. San Luis Obispo, CA, 93407, USA
Email: bliu17@calpoly.edu

Abstract: We present Centaurus – a scalable, open source, clustering service for K-means clustering of correlated, multidimensional data. Centaurus provides users with automatic deployment via public or private cloud resources, model selection (using Bayesian information criterion), and data visualisation. We apply Centaurus to a real-world, agricultural analytics application and compare its results to the industry standard clustering approach. The application uses soil electrical conductivity (EC) measurements, GPS coordinates, and elevation data from a field to produce a 'map' of differing soil zones (so that management can be specialised for each). We use Centaurus and these datasets to empirically evaluate the impact of considering multiple K-means variants and large numbers of experiments. We show that Centaurus yields more consistent and useful clusterings than the competitive approach for use in zone-based soil decision-support applications where a 'hard' decision is required.

Keywords: K-means clustering; cloud computing.

Reference to this paper should be made as follows: Golubovic, N., Krintz, C., Wolski, R., Sethuramasamyraja, B., and Liu, B. (2018) 'A scalable system for executing and scoring K-means clustering techniques and its impact on applications in agriculture', *Int. J. Big Data Intelligence*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Nevena Golubovic is a PhD candidate and Research Assistant at the Computer Science department of UC Santa Barbara. Her research interests include building scalable data analytics systems for sensor networks with a current focus on data-driven farming. She received her Diploma in Mathematics and Computer Science from the University of Belgrade, Serbia.

Chandra Krintz is a Professor of Computer Science (CS) at UC Santa Barbara and a Chief Scientist at AppScale Systems Inc. She holds a MS/PhD in CS from the UC San Diego. Her research interests include programming systems, cloud and big data computing, and the internet of things (IoT). She has supervised and mentored over 60 students and she has led several educational and outreach programs that introduce young people to computer science.

Rich Wolski is a Professor of Computer Science at the UC Santa Barbara, and co-founder of Eucalyptus Systems Inc. He received his MS and PhD from the UC Davis (while a research scientist at Lawrence Livermore National Laboratory) he has also held positions at the UC San Diego,

and the University of Tennessee, the San Diego Supercomputer Center and Lawrence Berkeley National Laboratory. He has led several national scale research efforts in the area of distributed systems and he is the Progenitor of the Eucalyptus open source cloud project.

Balaji Sethuramasamyraja is an Associate Professor and Program Director of Agricultural Systems Management and Precision Agriculture Technology in the Department of Industrial Technology, Jordan College of Agricultural Sciences and Technology at California State University, Fresno. He is an Agricultural Engineer involved in applied engineering/technology research in precision agriculture, irrigation technology, and sustainable management practices. He is also Chair of the American Society of Ag and Biological Engineers California Nevada section.

Bo Liu is an Assistant Professor at BioResource and Agricultural Engineering Department, Cal Poly State University. His research and publication interests include agricultural robotics, artificial intelligence and control systems. He has taught a number of courses on sensors and data acquisition systems, agricultural automation, electricity, and Solidworks. He is a member of ASABE (American Society of Agricultural and Biological Engineers) and IEEE (Institute of Electrical and Electronics Engineers).

This paper is a revised and expanded version of a paper entitled 'CENTAURUS: a cloud service for K-means clustering' presented at the 3rd Intl. Conf. on Big Data Intelligence (DataCom), 2017 IEEE 15th Intl., Orlando, Florida, 6–10 November 2017.

1 Introduction

Sensing, data analytics, and cloud computing have revolutionised the way people with differing levels of expertise make decisions in support of research and business interests. Through their decreasing cost and increasing connectivity, sensing technologies are becoming increasingly accessible in wide range of human endeavors from agriculture and farming to social networks and information retrieval.

Unsupervised clustering is commonly employed to partition observation data into related groups. Clustering is a popular form of learning that requires minimal human intervention and precludes the need for model training and data labeling. Lloyd's clustering algorithm (Lloyd, 1982), commonly called K-means, is the most widely used approach to unsupervised clustering. The algorithm partitions data into K clusters based on their *distance* to specific points in a multi-dimensional space.

While K-means in its most basic form is simple to understand and implement, there are a myriad of algorithm variants that differ in how they calculate distances. Other K-means implementations require 'hyper-parameters' that control 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 an approach for 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 describes the data – a process known as model selection. From a systems perspective, Centaurus defines a pluggable framework into which clustering algorithms, K-means variants, and scoring metrics can be chosen. When users upload their data, Centaurus executes and automatically scales its K-means variants concurrently using public or private cloud resources. Centaurus automates experimentation with different hyper-parameters and provides a set of data and diagnostic visualisations so that users can best interpret its results and recommendation.

We integrate six K-means variants into Centaurus, each of which implementing a different point-distance computation. 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 using randomly selected cluster centres) and recommends the highest scoring algorithm configuration as the 'best' clustering to the user. We implement Centaurus using production-quality, open-source software and validate it using synthetic datasets with known clusters.

We then apply Centaurus to a real-world, agricultural analytics application and compare its results to the industry standard clustering approach. The application analyses fine-grained soil 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 customise management or operations for different zones on the farm (application of water, fertiliser, pesticides, etc.) (Moral et al., 2010; Fortes et al., 2015; Corwin and Lesch, 2003).

We analyse Centaurus clustering for EC measurements taken from three farms. We illustrate both the impact of including multiple K-means variants in the pool of algorithms and the benefits of executing multiple randomised trials on the quality of the clustering. Finally, we compare Centaurus to an alternative state of the art clustering tool [MZA (Fridgen et al., 2004)] for farm management zone identification and show that Centaurus yields more consistent and utilitarian clusterings for use in zone-based soil decision-support applications where a 'hard' decision is required.

In the sections that follow, we first overview related work (Section 2) and provide background on the K-means algorithms that Centaurus implements (Section 3). In Section 4, we overview the design and implementation of Centaurus. We then detail our datasets (Section 5), present our experimental methodology, analysis, and results (Section 6) and conclude in Section 7.

2 Related work

Extensive studies of K-means demonstrate its popularity for data processing and many surveys are available to interested readers (Jain et al., 1999; Berkhin, 2006). In this section we focus on K-means clustering for multivariate correlated data. We also discuss the application and need for such systems in the context of farm analytics when analysing soil electro-conductivity.

To integrate K-means into Centaurus, we leverage Murphy's (1998) work in the domain of Gaussian mixture models (GMMs). This work identifies multiple ways of computing the covariance matrices and using them to determine distances and log likelihood. To the best of our knowledge there is no prior work on using all six variants of cluster covariance computation within a K-means system. We also utilise the K-means++ (Arthur and Vassilvitskii, 2007) work for cluster centre initialisation.

The research and system that is most closely related to Centaurus, is MZA (Fridgen et al., 2004) – a computer program widely used by farmers to identify clusters in soil electro-conductivity (EC) data to aid farm zone identification and to optimise management. MZA uses fuzzy K-means (Dunn, 1974; Bezdek, 2013), computes a global covariance (i.e., one covariance matrix spanning all clusters) and employs either Euclidean (Heath et al., 1956), diagonal, or Mahalanobis distance to compute distance between points. MZA computes the covariance matrix once from all data points and uses this same matrix in each iteration. MZA compares clusters using two different scoring metrics: fuzziness performance index (FPI) (Odeh et al., 1992) and normalised classification entropy (NCE) (Bezdek, 2013). Centaurus attempts to address some 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, K-means variant, and scoring metric to employ). We also show that although MZA provides multiple scoring metrics (Centaurus provides a single scoring metric) to compare cluster quality, the MZA metrics commonly produce different 'recommended' clusterings.

The authors of x-means (Pelleg et al., 2000) use Bayesian information criterion (BIC) (Schwarz, 1978) (which Centaurus also employs) as a score for the univariate normal distribution. Our work differs in that we extend the algorithm and scoring to multivariate distributions and account for different ways of covariance matrix computation in the clustering algorithm. We provide six different ways of computing covariance matrix for K-means for multivariate data and examples that illustrate the differences.

Different parallel computational models were used to speed up the K-means cluster initialisation (Bahmani et al., 2012), or its overall runtime (Zhao et al., 2009). Our work differs in that we provide not only a scalable system but include K-means variants, flexibility for a user to select any one or all of the variants, as well as a scoring and recommendation system. Finally, Centaurus is also pluggable enabling other algorithms to be added and compared.

3 K-means variants

The K-means algorithm attempts to find a set of cluster centres that describe the distribution of the points in the dataset by minimising the sum of the squared distances between each point and its cluster centre. For a given number of clusters K, it first assigns the cluster centres by randomly selecting K points from the dataset. It then alternates between assigning points to the cluster represented by the nearest centre, and recomputing the centres (Lloyd, 1982; Bishop, 2006), while decreasing the overall sum of squared distances (Linde et al., 1980).

The sum-of-squared distances between data points and their assigned cluster centres 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, that it is possible to use distance metrics other than Euclidean distance to compute per-cluster differences in variance, or covariance between data features [e.g., Mahalanobis distance (Mahalanobis, 1936)]. Thus, for a given dataset, the algorithm can generate a number of different K-means clusterings – one for each combination of starting centres, distance metrics, and a method used to compute the covariance matrix. Centaurus integrates both Euclidian and Mahalanobis distance. Computation of Mahalanobis distance requires computation of a covariance matrix for the dataset.

In Centaurus we integrate six different methods for computing covariance matrices for K-means algorithm: *full-tied*, *full-untied*, *diagonal-tied*, *diagonal-untied*, *spherical-tied*, and *spherical-untied* (Murphy, 1998, 2012; Bishop, 2006; Cerioli, 2005). Each of these methods is defined as:

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

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

  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. This approach ignores the covariance observed between the dimensions of the dataset.
- *Spherical*: set covariance matrix diagonal elements to the variance computed across all dimensions and set

off-diagonal elements to zero. This method is 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.

We implement K-means in its general form using Mahalanobis distance in Centaurus using the following steps:

1 We use K-means++ (Arthur and Vassilvitskii, 2007) to randomly select K points from the data and assign these as the initial cluster centres $\boldsymbol{\mu}^{(k)}$, where K is the number of clusters, k is the cluster index, and $k = 1, \dots, K$.

2 For data points having d dimensions, compute initial covariance matrix $\sum$ using all data points:

$$\Sigma_{ij} = \frac{1}{n}\sum_{p=1}^{n}(x_i^{(p)} - \mu_i)(x_j^{(p)} - \mu_j)$$

where, $\sum_{ij}$ is $(i,j)^{\text{th}}$ component of the matrix $\sum$, $x_i^{(p)}$ 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 centre using Mahalanobis distance metric:

$$D(\boldsymbol{x}^{(p)}, \boldsymbol{\mu}^{(k)}) = \left((\boldsymbol{x}^{(p)} - \boldsymbol{\mu}^{(k)})^T \sum^{-1}(\boldsymbol{x}^{(p)} - \boldsymbol{\mu}^{(k)})\right)^{1/2}$$

where, $\boldsymbol{x}^{(p)}$ is the d-dimensional vector of components of the p^{th} data point, $\boldsymbol{\mu}^{(k)}$ is the centre of the k^{th} cluster.

4 Compute covariance matrix $\sum^{(k)}$ for each cluster using their cluster centre $\boldsymbol{\mu}^{(k)}$.

5 Compute the cluster centres $\boldsymbol{\mu}^{(k)}$: for each point 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 centre.

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 centres from the old cluster centres.

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 and a list of cluster centres that correspond to the maximum likelihood estimates of the cluster means. We use the interpretation of K-means as the 'hard' cluster assignment of GMM to compute the maximum log-likelihood [for use by the BIC (Schwarz, 1978) or the Akaike information criterion (Akaike, 1974)] in order to compare the local optima generated from different variants of K-means and, ultimately, to choose the 'best' one (Pelleg et al., 2000). We discuss the use of information criteria as a 'scoring' method across multiple runs of multiple variants in Section 4.1.

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 (Bishop, 2006; Murphy, 2012), as:

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

where, p is a data point having d dimensions, 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 centre, $\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu}^{(k)}, \sum^{(k)}\right)$ is the Gaussian probability density function with $\boldsymbol{\mu}^{(k)}$ mean and $\sum^{(k)}$ covariance.

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:

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

4 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 visualisation, and browser-based user interaction. Centaurus implements six different variants of K-means (described in Section 3). The complete system is described in detail in Golubovic et al. (2017). Centaurus provides a scalable execution environment with a deployment of multiple K-means algorithm parameterisations and variants running in parallel. Centaurus also provides a user interface, visualisation tools for viewing the results in different ways, and a scoring metric for model selection which it uses to provide a recommendation of the best-performing K-means algorithm configuration for the given data.

For this work, we implement and employ a simple web interface for Centaurus. Using this interface, a user uploads the data file via a web browser and selects the maximum value for K. The user can also specify the maximum number of computational resources to be used which limits Centaurus parallelism but provides constrains on resources. Resources are virtual servers each with a single processor. Centaurus runs single, sequential K-means algorithm instances in parallel using multiple virtual servers. Centaurus adds virtual servers dynamically as the workload increases, up to a specified maximum (the default is ten servers). Centaurus then scores the model using a scoring metric based on the BIC (Schwarz, 1978). When all runs complete, Centaurus reports the result with the highest BIC score as its recommended clustering.

4.1 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.

Centaurus scoring takes label assignments from a clustering result for a particular K value and returns a score based on the BIC. BIC uses the log maximum likelihood to rate a particular clustering and then subtracts a 'penalty' function that captures the number of parameters that must have been estimated to generate the clustering, scaled by the sample size.

We compute the BIC score for a full tied clustering with K clusters as:

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

where, $\hat{\boldsymbol{\mu}}$ is the maximum likelihood estimator for the cluster centres, $\hat{\sum}$ is the d-dimensional maximum likelihood estimator for the cluster covariance matrices, $l(\boldsymbol{X}|\hat{\boldsymbol{\mu}}, \hat{\sum})$ 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 centres, and $(K \cdot \frac{d \cdot (d+1)}{2})$ parameters for each of the K symmetric cluster covariance mattrices:

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

when a single covariance matrix is used for all clusters (the Untied variants), the factor of K in the third term is set to 1. Similarly, when the off-diagonal elements are zero, the fraction in the third term is either d (for the diagonal variants) or 1 (for the Spherical variants). For BIC, the penalty function is r_K multiplied by $\frac{\log n}{2}$ where n is the total number of points.

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 30 data points in the current system.

4.2 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). Advanced users can modify the following Centaurus parameters: maximum number of clusters to fit to the data (K); number of experiments (N) per K to run; number of times to initialise the K-means clustering (M); 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.); whether to scale the data so that each dimension has zero mean and unit standard deviation ($scale$).

Centaurus considers each parameterisation that the user chooses (including the default) as a 'job'. Each job consists of multiple tasks (experiment runs) that Centaurus deploys. Users can check the status of a job or view the report for a job (when completed). The status page provides an overview of all the tasks with a progress bar for the percentage of tasks completed and a table showing task parameters and outcomes available for download and visualisation.

Centaurus has a report page to provide its recommendation. The recommendation consists of the number of clusters and K-means variant that produced the best BIC score. This page also shows the cluster assignments and spatial plots using longitude and latitude (if included in the original dataset). For additional analysis users can select 'advanced report' to see the correlation among features in the dataset, BIC scores for each K-means variant, best clusterings for each one of the variants, etc.

We implement Centaurus using Python 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 (Nurmi et al., 2009; Aristotle Cloud Federation, 2018) and integrates virtual servers with different CPU, memory, and storage capabilities based on the requirement of a particular component. Centaurus consists of five primary components:

1 *Frontend*: we couple the Python Flask (2018) (v0.12.1) web framework with Gunicorn (2018) (v19.7.1) web server and NGINX (2018) (v1.4.6) reverse proxy server to provide a robust application hosting service.

2 *Backend worker*: we use Python Celery (2018) (v4.0.2), a distributed computation framework, to perform data analysis computation tasks asynchronously and at scale (Lunacek et al., 2013). We leverage auto-scaling 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 (2018) (v3.2.4) message broker to send information about each job from the frontend to the workers. This enables frontend to

quickly offload its work to the queue where it is sent systematically to the workers as and when they become available.

4 *Backend satabase*: We support both PostgreSQL (v9.5.11) SQL database (PostgreSQL, 2018) and MongoDB Community Edition (v3.4.4) NoSQL database (Mongodb, 2018) to store parameters and results for jobs and tasks. MongoDB had advantages in fast writing of experiment results but analysis cost was high, while PostgreSQL had high cost of writing but offering high performance for data analysis once the experiments were completed.

5 *Backend file store*: we use Amazon simple storage system (S3) (Amazon S3, 2018) to store user uploaded files.

Other packages that Centaurus leverages include Numpy (Walt et al., 2011) (v1.12.1), Pandas (McKinney et al., 2010) (v0.19.2), SciKit-Learn (Pedregosa et al., 2011) (v0.18.1), and SciPy (Jones et al., 2001) (v0.19.0) for data processing and Matplotlib (Hunter, 2007) (v2.0.1) and Seaborn (2018) (v0.7.1) for data visualisation.

5 Datasets

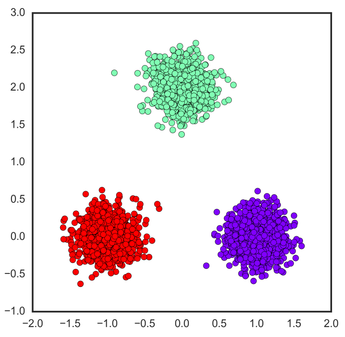
We use both synthetic and real-world datasets to evaluate Centaurus empirically. We generate the synthetic datasets with known clusters (as 'ground truth'), which we use to validate and measure the accuracy of the Centaurus implementation. Using the real-world application data from precision agriculture, we also compare the results generated by Centaurus for management zone determination to the industry standard and use them to illustrate the Centaurus visualisation capabilities.

5.1 Synthetic datasets

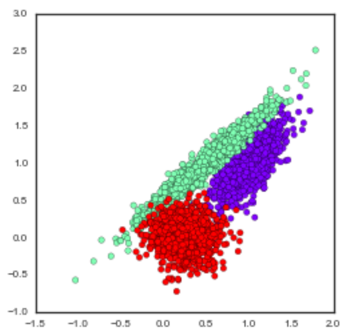
We first create multiple two-dimensional synthetic datasets using multivariate Gaussian distributions. The datasets have three clusters with 1,000 points per cluster and varying degrees of inter-dimensional correlation in each cluster. Figure 1 shows these datasets with their ground truth cluster assignments.

- *Dataset-1* clusters have no correlation and equal standard deviations of 0.2 for each dimension. Cluster centres are set at positions $(1, 0)$, $(-1, 0)$, and $(0, 2)$ as can be seen in Figure 1.
- In *Dataset-2* cluster centred at $(0.25, 0)$ has two dimensions that are independent (not correlated) with the same standard deviations of 0.2. The cluster centred at $(1, 1)$ has correlation 0.70 between dimensions, while the cluster centred 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 centred at $(1, 0)$ has two independent dimensions, while the clusters centred 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.

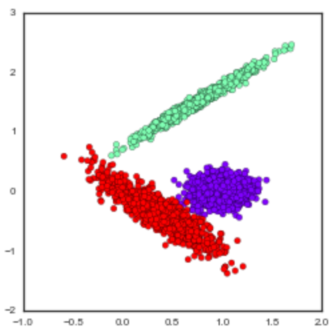
Figure 1 Synthetic datasets shown with ground truth assignment, (a) Dataset-1 (b) Dataset-2 (c) Dataset-3 (see online version for colours)



(a)



(b)



(c)

5.2 Application datasets

Our farm data consists of electrical conductivity (EC) measurements of soil measured (at 30 cm and 90 cm depths) using an instrument manufactured by Veris Technologies Inc. (2018). The surveyor collects EC as well as the GPS coordinates and elevation data associated with each EC measurement. The approach produces a data file containing five dimensions of data: longitude, latitude, elevation, EC at 30 cm depth (EC1), and EC at 90 cm depth (EC2).

Three different farms listed below represent good variety of different management practices and EC sampling capabilities. The data sampling is not uniform, the distance between data points is influenced by the type of plant; if it is perennial plant (e.g., trees at Cal Poly) we can only gather readings between rows, whereas an empty field (e.g., after season corn field at UNL) allows for narrow row spacing. Sampling is further influenced by the speed at which the sensor was driven. At the speed of 10 km/h the sensor produces around 275 readings per hectare.

The EC datasets come from three different farms:

- *Cal Poly dataset* is a 4.85 ha lemon farm at California Polytechnic State University, San Louis Obispo, California from which we have collected 3,232 data points.
- *Sedgwick* is a 12.1 ha field located in the Santa Ynez Valley, California from which we have collected 2,675 data points.
- *UNL* is a 36.8 ha field at University of Nebraska, Lincoln, from which we have 5,823 data points. It is mostly used for corn and soybean.

6 Results

In this section, we evaluate Centaurus K-means cluster quality for multiple K-means variants. We first compare cluster resolution power of the variants using synthetic datasets. We then detail the clustering that Centaurus determines for Veris (Veris Technologies Inc., 2018) EC measurements taken from three farms. From this data, we illustrate both the advantage of including multiple K-means variants in the pool of algorithms that Centaurus implements and the effect of executing multiple randomised trials on the quality of the clustering. Finally, we compare Centaurus clusterings to those produced by MZA for the same Veris EC data.

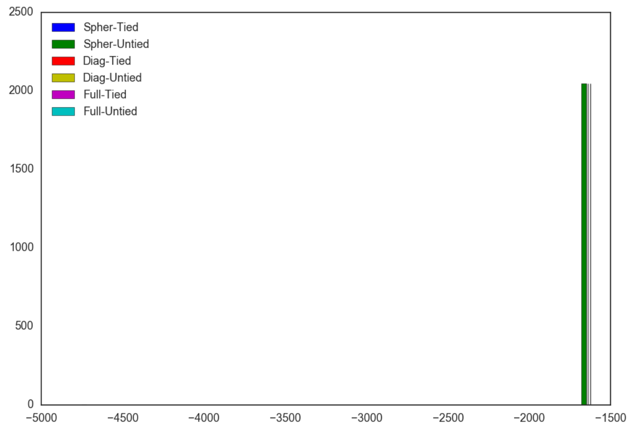
6.1 K-means variants

To illustrate the cluster resolution power of different variants of K-means, we use three synthetic datasets, each of which is drawn from a bivariate Normal distribution. For each of the six variants described in Section 3, we use Centaurus to repeat the clustering 2,048 times using a different, randomly selected, initial cluster centre assignments and to compute the BIC score for each.

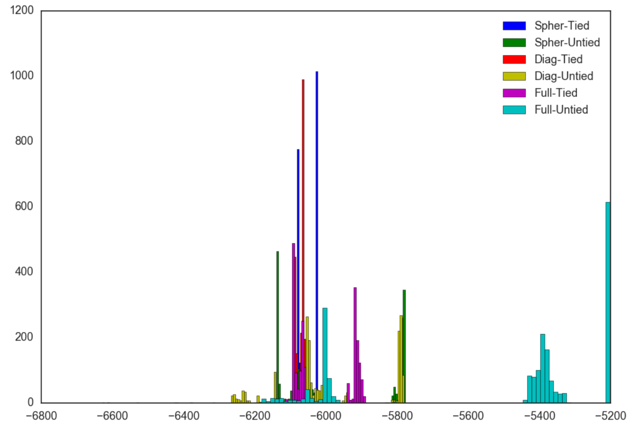
Figure 2 shows histograms of the BIC scores for each of three synthetic datasets. We divide the scores among 100 bins. For each dataset we present six histograms, one for each of the K-means variants, represented in different colors, where each variant has a total of 2,048 single K-means runs. The X-axis depicts BIC scores from experiments – farther right corresponds to larger BIC and thus higher quality clusterings.

Dataset-1 consists of well-separated clusters. All six variants perform well making the simpler (i.e., those with fewer parameters to be estimated) variants generate slightly higher BIC scores [as depicted in Figure 2(a)].

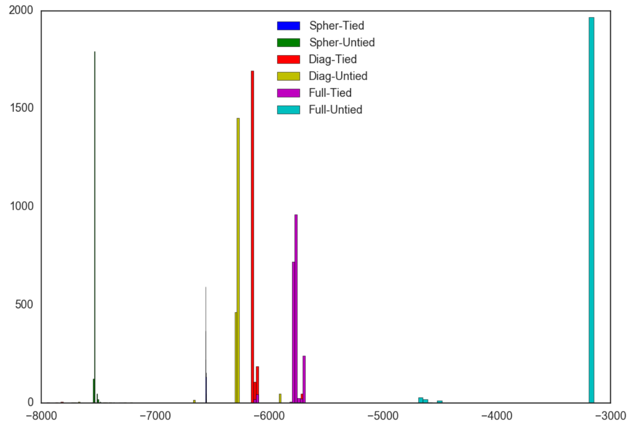
Figure 2 BIC score histograms for three synthetic datasets and six K-means variants, (a) Dataset-1 (b) Dataset-2 (c) Dataset-3 (see online version for colours)



(a)



(b)



(c)

However, for datasets where the dimensions are more highly correlated and/or where that correlation differs across clusters, the complex variants (full tied and full untied) outperform their simpler counterparts in terms of BIC score. Dataset-2 and Dataset-3 differ in that for the latter, the cross-dimensional correlation varies by synthetic cluster. Nonetheless, as shown in Figures 2(b) and 2(c), the full-untied variant (which computes a separate co-variance matrix for each cluster) performs best. These experiments, although synthetic, show the importance of considering different variants when employing K-means clustering.

6.2 Veris EC datasets

In what follows, we refer to an experiment as ten repeated clusterings [using K-means++ (Arthur and Vassilvitskii, 2007) to make initial assignments in each repetition] for each number of clusters k between 1 and 10, for each of the six K-means variants we examine in this study. Thus, each single experiment consists of $10 * 10 * 6 = 600$ individual cluster assignments using K-means.

Centaurus repeats each experiment N times, where $N = 2^i$, for $i = 0, ..., 11$. In this study, we refer to a set of N experiments as job-N. Thus job-N consists of $N * 600$ individual clusterings. Centaurus filters out any clustering with a cluster having fewer than 30 points (so that any per-cluster statistical estimates are statistically valid). To determine the best clustering from a job, Centaurus computes a BIC score for each clustering in the job and selects the one with the largest score.

6.2.1 Cluster quality analysis

To show the effect of using a large sample when determining the 'best' clustering, in Figure 3 we plot the largest observed BIC score (on the y-axis) versus the experiment number N (on the x-axis, which uses a log scale). Figures 3(a), 3(b), and Figures 3(c) show EC data from Cal Poly, Sedgwick, and UNL respectively.

As the sample size goes up, the probability of determining a clustering with the 'best' BIC score (or, at least a consistently good BIC score) should increase as well. For the Sedgwick data [Figure 3(b)] this effect is clearly visible. Once the number of experiments exceeds $N = 2^8$, there is no further improvement in BIC. However for Cal Poly and UNL, the presence of a higher BIC occurring only at $N = 2^{11}$ indicates that even more repetitions are necessary to identify a consistently 'best' clustering. Thus, for these datasets, the best clustering in the 'space' of all possible clusterings is rare since it does not occur repeatedly when the sample size is less than 1.23 million ($2^{11} * 600$).

6.2.2 Cluster specificity

One possibility is that the 'best' clustering (the one with the highest BIC score) and the next best are similar. In this case, then, a large exploration of the clustering search space may be unwarranted because the best is not substantially different from the next best (which might be more common and require less computational effort to find).

To investigate this possibility, we consider the two largest jobs from the Cal Poly dataset: the largest job with $N = 2^{11}$ experiments (job-2048) and the second largest job with $N = 2^{10}$ experiments (job-1024). The largest job, Job-2048, with twice the number of experiments of job-1024, has the best BIC score of –8,847.9 [Figure 4(a)]. This corresponds to a clustering with four clusters having cardinality of 2,188, 531, 308, and 205, respectively. The second best clustering has BIC score of -8925.4 and three clusters with cardinality 1733, 973, and 526, respectively, as shown in Figure 4(b).

Figure 3 Largest observed BIC score vs number of experiments (log scale), (a) Cal Poly (b) Sedgwick (c) UNL (see online version for colours)

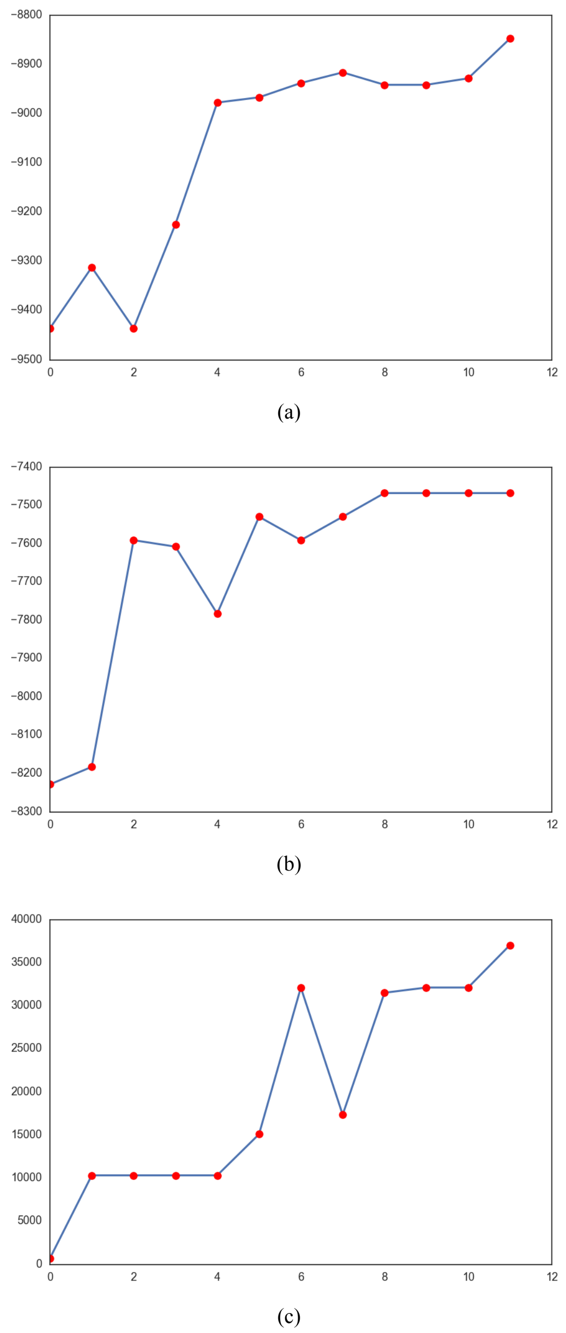
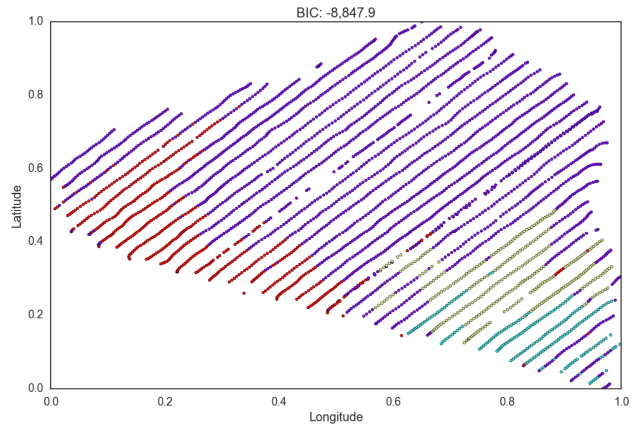
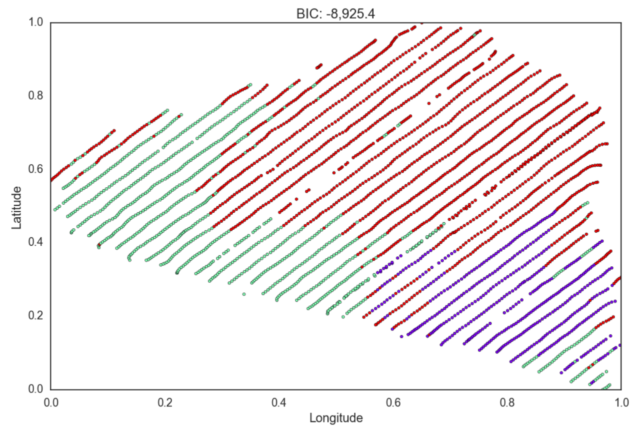


Figure 4(c) shows the difference between these two clusterings. A specific data point is shown (i.e., is considered 'different') if it has a different cluster number assignment (is in a different cluster) when we rank clusters by cardinality. For this data, clearly these clusterings differ. Thus, doubling the number of experiments from 1,024 to 2,048 allows Centaurus to find a clustering with a better BIC score.

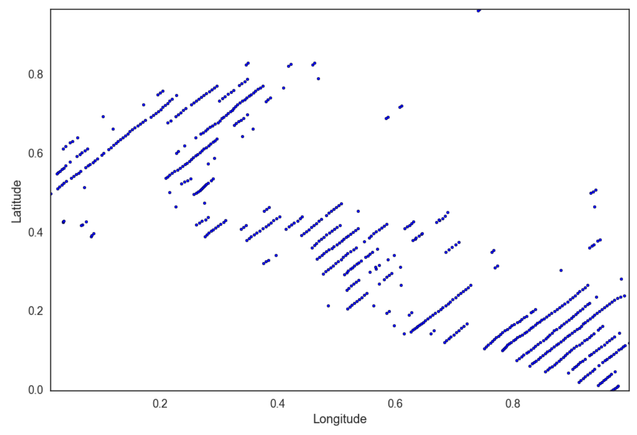
Figure 4 Clusterings of Cal Poly dataset, (a) best with BIC –8,847.9 (b) second best with BIC –8,925.4 (c) differences (see online version for colours)



(a)



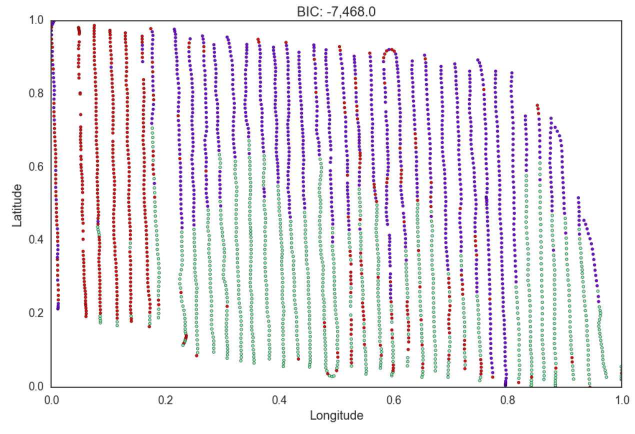
(b)



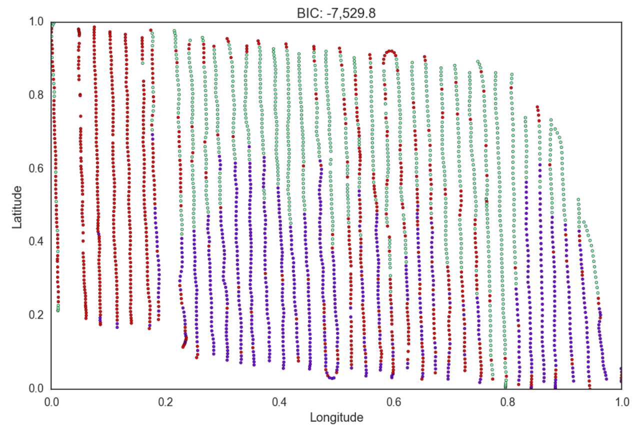
(c)

The Sedgwick dataset has more stable outcome in terms of the best BIC score when increasing the number of experiments. Figure 3(b) shows that even with 256 experiments (150 K K-means runs), we achieve the same maximum BIC score as with 2,048 experiments. The best result has a BIC score of –7,468.0 and three clusters with 1,111, 996, and 568 elements [as shown in Figure 5(a)]. This result is consistent over many repeated jobs with a sufficiently large number of experiments, i.e., any job with more than 256 experiments produced this same clustering as the one corresponding to the largest score.

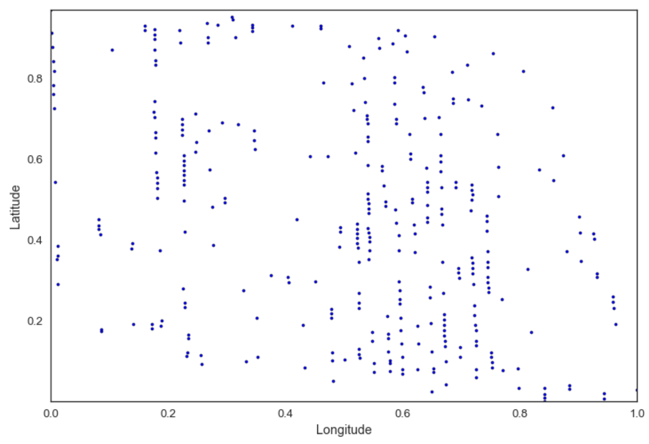
Figure 5 Clusterings of Sedgwick dataset, (a) best with BIC: –7,468.0 (b) second best with BIC: –7,529.8 (c) differences (see online version for colours)



(a)

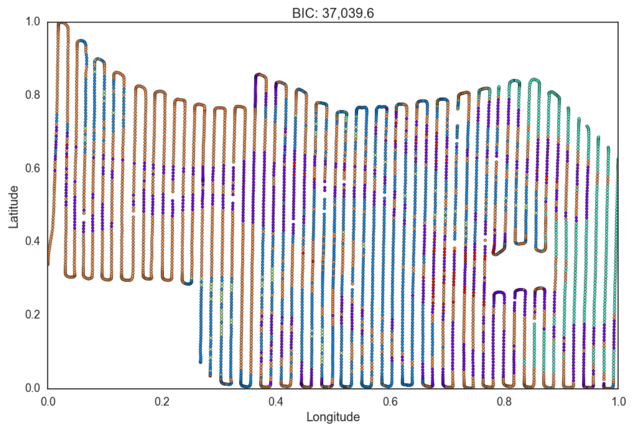


(b)

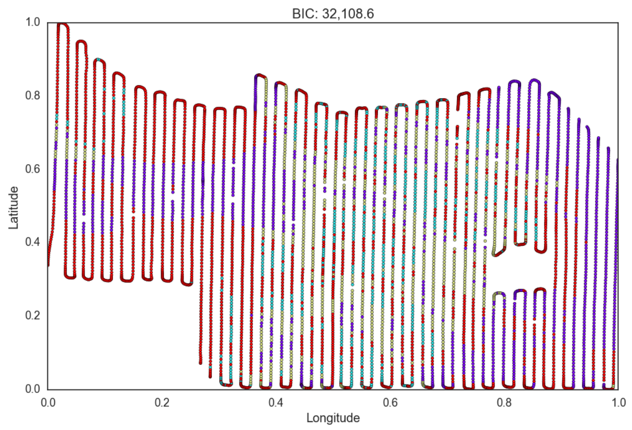


(c)

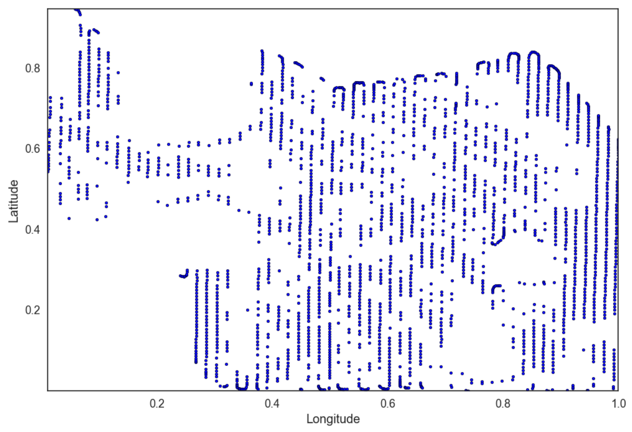
Figure 6 Clusterings of UNL dataset, (a) best with BIC: 37,039.6 (b) second best with BIC: 32,108.6 (c) differences (see online version for colours)



(a)



(b)



(c)

The second best clustering agrees with the best result on the number of clusters ($k = 3$) with cluster cardinalities of 963, 879, and 833, and a BIC score of –7,529.8 [as shown in Figure 5(b)]. While these clusters do differ, Figure 5(c) shows that the differences are scattered spatially. Thus the best and second best clusterings may not differ in terms of actionable insight.

For the UNL field, the best and second best clusterings (and their respective BIC scores) are shown in Figure 6. These are both from job-2048. The best clustering has six clusters with cardinalities 2,424, 1,493, 1,138, 561, 111, and 70, respectively. The second best clustering has four clusters with cardinalities 2,730, 1,615, 838, and 614, respectively. From these features and the differences shown in Figure 6(c) it is clear the best and second best clustering are dissimilar.

Further, the second best clustering from job-2048 [shown in Figure 6(b)] is the best clustering in job-64, job-512, and job-1024 respectively. As with the Cal Poly data (but not the Sedgwick data), doubling the number of experiments from 1,024 to 2,048 'exposed' a better and significantly different clustering.

6.2.3 K-means variants

Unlike the results for the synthetic datasets, the best clustering for the Veris EC datasets is produced by the full untied variant for sufficiently large job sizes. This result is somewhat surprising since the full untied variant incurs the largest score penalty in the BIC score computation among all of the variants. The score is penalised for the mean, variance, and co-variance estimates from each cluster. The other variants require fewer parameter estimates (and thus have a lower penalty). Related work has also argued for using fewer estimated parameters to produce the best clustering (Fridgen et al., 2004) leading to an expectation that a simpler variant [e.g., full tied as in Fridgen et al. (2004)] would produce the best clustering, but is not the case for these datasets. Because Centaurus considers all variants, it will find the best clustering even if this effect is not general to all Veris data.

6.3 Comparison with MZA

We next compare the Centaurus performance against 'management zone analysis' [MZA (Fridgen et al., 2004)] for the Veris EC farm datasets. MZA is a popular methodology with concomitant software for clustering Veris EC data. Results for such clusterings are available from Fridgen et al. (2004); Odeh et al. (1992); Corwin and Lesch (2003).MZA requires users to set a real numbered parameter known as the 'fuzziness index' that controls the degree of specificity of the algorithm. The authors of Fridgen et al. (2004) use a value of 1.5 in their experiments, an additionally suggest that values between 1.2 and 1.5 are appropriate for clustering soil EC measurements.

For the chosen fuzziness parameter m (for $m > 1.0$) and the maximum number of clusters K, MZA runs a single fuzzy clustering for each k $(2, \ldots, K)$. MZA scores the resulting clusterings using two metrics: FPI (Odeh et al., 1992), and normalised classification entropy (NCE) (Odeh et al., 1992; Bezdek, 2013). FPI is a measure of the degree of separation between partitions (lower fuzziness means a higher degree of separation) while NCE measures the disorganisation of each one of the fuzzy partitions. The authors of Fridgen et al. (2004) and Odeh et al. (1992) suggest that the best clustering is the one with the smallest value of k that also has the smallest scores for both metrics among all clusterings.

Table 1 MZA results for the farm datasets for different values of k and fuzziness coefficients (m)

K	*Cal Poly Veris EC*							
	m = 1.1		*m = 1.3*		*m = 1.5*		*m = 2.0*	
	FPI	*NCE*	*FPI*	*NCE*	*FPI*	*NCE*	*FPI*	*NCE*
2	0.044	0.016	0.120	*0.044*	0.265	0.093	0.475	*0.162*
3	0.028	*0.013*	0.095	0.048	0.170	0.088	0.361	0.189
4	0.027	0.014	*0.083*	0.046	*0.143*	*0.084*	*0.328*	0.207
5	*0.025*	0.014	0.087	0.053	0.166	0.105	0.370	0.255
6	0.027	0.016	0.098	0.062	0.180	0.122	0.393	0.291
7	0.031	0.019	0.099	0.065	0.173	0.121	0.386	0.304
K	*Sedgwick Veris EC*							
	m = 1.1		*m = 1.3*		*m = 1.5*		*m = 2.0*	
	FPI	*NCE*	*FPI*	*NCE*	*FPI*	*NCE*	*FPI*	*NCE*
2	*0.018*	*0.006*	*0.063*	*0.023*	*0.126*	*0.047*	0.413	*0.143*
3	0.020	0.010	0.081	0.040	0.145	0.074	*0.315*	0.164
4	0.025	0.013	0.080	0.044	0.140	0.081	0.324	0.201
5	0.025	0.015	0.088	0.053	0.158	0.100	0.356	0.244
6	0.026	0.016	0.091	0.057	0.172	0.116	0.383	0.281
7	0.028	0.017	0.094	0.062	0.167	0.116	0.388	0.299
K	*UNL Veris EC*							
	m = 1.1		*m = 1.3*		*m = 1.5*		*m = 2.0*	
	FPI	*NCE*	*FPI*	*NCE*	*FPI*	*NCE*	*FPI*	*NCE*
2	0.038	0.014	0.126	0.044	0.201	0.069	0.341	*0.117*
3	0.020	*0.010*	0.068	0.033	0.115	*0.057*	0.233	0.119
4	0.019	0.010	0.059	*0.033*	0.102	0.059	*0.229*	0.142
5	*0.017*	0.010	*0.056*	0.034	0.100	0.063	0.239	0.163
6	0.025	0.015	0.082	0.051	*0.094*	0.062	0.239	0.177
7	0.021	0.013	0.073	0.046	0.136	0.092	0.285	0.212

We run MZA for the three farm datasets and present results in Table 1. For this study, we set $k = 2, \dots, 7$ and consider fuzziness values of 1.1, 1.3, 1.5, and 2.0. The value of k is given in the first column. The table shows the FPI and NCE scores for each fuzziness value and for each k in the data columns. The lowest (considered the best) score is shown in italic.

The results show that MZA often recommends different clusterings depending upon the scoring metric and fuzziness value used. We first consider scores across values of m and k. In all cases, across datasets, NCE and FPI select $m = 1.1$ as producing the best clustering. This is in contrast to

1 the MZA default ($m = 1.3$)

2 the values recommended by the authors (1.2–1.5)

3 the value for m used in the original MZA study (1.5) (Fridgen et al., 2004), which all perform worse.

Unfortunately, the best performing cluster size differs between NCE and FPI for both Cal Poly (top table) and UNL (bottom table). For the Cal Poly dataset (top table), NCE reports that the best clustering is ($k = 3$, row 3, column 3). FPI reports that the best clustering is ($k = 5$, row 5, column 2). For Sedgwick (middle table), NCE and FPI agree on ($k = 2$, row 2, columns 2 and 3). For the UNL (bottom table), NCE selects ($k = 3$, row 3, column 3) and FPI selects ($k = 5$, row 5, column 2).

Moreover, FPI and NCE disagree more often than they agree for these datasets. For the Cal Poly dataset (top table) both scores agree only when $m = 1.5$ suggesting that $k = 4$ (row 4, columns 6 and 7) is the best clustering. For other values of m, MZA recommends cluster sizes that range from $k = 2$ to $k = 5$. For Sedgwick (middle table) and $m = 2.0$ (columns 8 and 9), FPI selects $k = 3$ and NCE selects $k = 2$. For UNL, no FPI-NCE pairs agree on the best clustering, with MZA recommending all values of k (except 7) for different m.

Because fine-grained EC measurements (e.g. using soil core samples and lab analysis) are not available for the Cal Poly, Sedgwick, and UNL farm plots, it is not possible to compare the MZA and Centaurus in terms of which produces a more accurate spatial maps from the Veris data. Even with expert interpretation of the conflicting MZA results for Cal Poly and UNL, we do not have access to 'ground truth' for the fields. However, it is possible to compare the two methods with the synthetic datasets shown in Figure 1. In particular, Table 2 shows the percentage of 3,000 data points (1,000 per cluster) that were incorrectly classified in terms of their cluster label Centaurus and MZA. That is, it shows the fraction (as a percentage) of data points that were mis-assigned by the 'best'

Centaurus clustering for each (first row) and same measure of error rate for MZA (second row).

Note that this evidence suggests Centaurus is more effective for some clustering problems but (again, due to a lack of ground truth) is not conclusive for the empirical data. Instead, from the empirical data we claim that Centaurus is more utilitarian than MZA because disagreement between FPI and NCE differing possible best clusterings based on user-selected values of m, can make MZA results difficult and/or error prone to interpret for non-expert users. MZA recommendations may be useful in providing an overall high level 'picture' of the Veris data clustering, but its varying recommendations are challenging to use for making 'hard' decisions (e.g. to control irrigation duration) by experts and non-experts alike. In contrast, Centaurus provides both a single 'hard' spatial clustering assignment and a way to explain (in terms of maximum likelihood and BIC penalty score) why one clustering should be preferred over another and which one is 'best' when ground truth is not available.

Table 2 Percentage error (out of 3,000 data points per dataset) for Centaurus and MZA on the synthetic datasets shown in Figure 1

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

In contrast, Centaurus is able to use its variants of K-means, a BIC-based scoring metric, and large state space exploration to determine a single 'best' clustering. The only free parameter the user must set is the size of the state space exploration (the default is N = 2,048 experiments which is 1.23 M K-means runs). As the work in this study illustrates, Centaurus can find rare and relatively unique high-quality clusterings when the state space it explores is large.

A large state space (each requiring a separate 'run' of the K-means algorithm) of course requires more computational power than MZA. MZA is a stand alone software package that runs on a laptop or desktop computer. In contrast, Centaurus is designed to run as a highly concurrent and scalable cloud service (via a browser) and uses a single processor per K-means run. As such, it automatically harnesses multiple computational resources on behalf of its users. Centaurus can be configured to constrain the number of resources (CPUs) it uses; doing so proportionately increases the time required to complete a job (each independent K-means run takes between 0.3 s and 1 s in our experiments). As part of future work, we are investigating the use of both spot instances and serverless functions to keep the cost of jobs as low as possible if/when Centaurus is deployed in the public cloud. For this work, we host Centaurus on two large private cloud systems: Aristotle (Aristotle Cloud Federation, 2018) and Jetstream (Stewart et al., 2015; Towns et al., 2014).

7 Conclusions

With this work, we present Centaurus, a scalable 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 visualise their results in multiple ways. Centaurus leverages cloud resources and services to automatically deploy, scale, and score K-means runs.

We empirically evaluate Centaurus using synthetically generated and real-world data from an agricultural analytics application, which uses EC measurements of soil (along with elevation) to identify regions that are similar. We use this data to evaluate the impact of considering multiple K-means variants and large numbers of experiments, and to compare against a popular industry standard clustering approach (MZA). We find that Centaurus overcomes many of the limitations of MZA and yields more useful clusterings for zone-based soil decision-support. Finally, we provide a detailed analysis of the results that shows that for these datasets, large numbers of experiments are necessary to identify the clustering variant that produces the highest quality clustering.

As part of future work, we are extending 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 plan to incorporate other publicly available datasets (e.g., SSURGO) to improve clustering methods when applicable. From the systems perspective, we are investigating the use of both spot instances and serverless functions to keep the cost of jobs as low as possible if Centaurus is deployed in the public cloud.

Acknowledgements

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 would like to thank Prof. Bruce A. Roberts, and Dr. Jim Yager from California State University, Fresno, and Dr. Newell R Kitchen from United States Department of Agriculture for their support on this project.

References

Akaike, H. (1974) 'A new look at the statistical model identification', *IEEE Transactions on Automatic Control*, Vol. 19, No. 6, pp.716–723.

Amazon S3 (2018) [online] http://www.aws.amazon.com/s3 (accessed 1 March 2018).

Aristotle Cloud Federation (2018) [online] https://federatedcloud.org (accessed March 2018).

Arthur, D. and Vassilvitskii, S. (2007) 'K-means++: the advantages of careful seeding', in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Society for Industrial and Applied Mathematics, pp.1027–1035.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R. and Vassilvitskii, S. (2012) 'Scalable K-means++', *Proceedings of the VLDB Endowment*, Vol. 5, No. 7, pp.622–633.

Berkhin, P. (2006) 'A survey of clustering data mining techniques', in *Grouping Multidimensional Data*, pp.25–71, Springer.

Bezdek, J.C. (2013) *Pattern Recognition with Fuzzy Objective Function Algorithms*, Springer Science and Business Media.

Bishop, C.M. (2006) 'Pattern recognition', *Machine Learning*, Vol. 128, pp.1–58.

Python Celery (2018) [online] http://www.celeryproject.org (accessed March 2018).

Cerioli, A. (2005) 'K-means cluster analysis and mahalanobis metrics: a problematic match or an overlooked opportunity', *Statistica Applicata*, Vol. 17, No. 1.

Corwin, D. and Lesch, S. (2003) 'Application of soil electrical conductivity to precision agriculture', *Agronomy Journal*, Vol. 95, No. 3, pp.455–471.

Dunn, J.C. (1974) 'Well-separated clusters and optimal fuzzy partitions', *Journal of Cybernetics*, Vol. 4, No. 1, pp.95–104.

Fortes, R., Millán, S., Prieto, M. and Campillo, C. (2015) 'A methodology based on apparent electrical conductivity and guided soil samples to improve irrigation zoning', *Precision agriculture*, Vol. 16, No. 4, pp.441–454.

Fridgen, J.J., Kitchen, N.R., Sudduth, K.A., Drummond, S.T., Wiebold, W.J. and Fraisse, C.W. (2004) 'Management zone analyst (MZA)', *Agronomy Journal*, Vol. 96, No. 1, pp.100–108.

Golubovic, N., Gill, A., Krintz, C. and Wolski, R. (2017) 'Centaurus: a cloud service for K-means clustering', in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress*, November.

Gunicorn (2018) [online] http://www.gunicorn.org (accessed March 2018).

Heath, T.L. et al. (1956) *The Thirteen Books of Euclid's Elements*, Courier Corporation.

Hunter, J. (2007) 'Matplotlib: a 2D graphics environment', *Computing in Science and Engineering*, Vol. 9, pp.90–95.

Jain, A.K., Murty, M.N. and Flynn, P.J. (1999) 'Data clustering: a review', *ACM Computing Surveys (CSUR)*, Vol. 31, No. 3, pp.264–323.

Jones, E., Oliphant, T., Peterson, P. et al. (2001) *SciPy: Open Source Scientific Tools for Python*.

Linde, Y., Buzo, A. and Gray, R. (1980) 'An algorithm for vector quantizer design', *IEEE Transactions on Communications*, Vol. 28, No. 1, pp.84–95.

Lloyd, S. (1982) 'Least squares quantization in pcm', *IEEE Transactions on Information Theory*, Vol. 28, No. 2, pp.129–137.

Lunacek, M., Braden, J. and Hauser, T. (2013) 'The scaling of many-task computing approaches in python on cluster supercomputers', in *IEEE Cluster Computing*, pp.1–8.

Mahalanobis, P.C. (1936) 'On the generalized distance in statistics', *Proceedings of the National Institute of Sciences (Calcutta)*, Vol. 2, pp.49–55.

McKinney, W. et al. (2010) 'Data structures for statistical computing in Python', in *Proceedings of the 9th Python in Science Conference*, van der Voort S, Millman J, Vol. 445, pp.51–56.

Mongodb (2018) [online] http://www.mongodb.com (accessed March 2018).

Moral, F., Terrón, J. and Da Silva, J.M. (2010) '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.

Murphy, K.P. (1998) *Fitting a Conditional Linear Gaussian Distribution*.

Murphy, K.P. (2012) *Machine Learning: a Probabilistic Perspective*, MIT Press.

NGINX (2018) [online] http://www.nginx.org (accessed March 2018).

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D. (2009) 'The eucalyptus open-source cloud-computing system', in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09*, IEEE, pp.124–131.

Odeh, I., Chittleborough, D. and McBratney, A. (1992) '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.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. et al. (2011) 'Scikit-learn: machine learning in python', *Journal of Machine Learning Research*, October, Vol. 12, pp.2825–2830.

Pelleg, D., Moore, A.W. et al. (2000) 'X-means: extending K-means with efficient estimation of the number of clusters', in *ICML*, Vol. 1, pp.727–734.

PostgreSQL (2018) [online] http://www.postgresql.org (accessed March 2018).

Python Flask (2018) [online] http://www.flask.pocoo.org (accessed March 2018).

RabbitMQ (2018) [online] http://www.rabbitmq.com (accessed March 2018).

Schwarz, G. (1978) 'Estimating the dimension of a model', *Annals of Statistics*, Vol. 6, No. 2.

Seaborn (2018) *Statistical Data Visualization*, [online] http://www.seaborn.pydata.org (accessed March 2018).

Stewart, C., Cockerill, T., Foster, I., Hancock, D., Merchant, N., Skidmore, E., Stanzione, D., Taylor, J., Tuecke, S., Turner, G., Vaughn, M. and Gaffney, N. (2015) 'Jetstream: a self-provisioned, scalable science and engineering cloud environment', in *XSEDE Conference: Scientific Advancements Enabled by Enhanced Cyberinfrastructure*, ACM: 2792774 [online] http://dx.doi.org/10.1145/2792745.2792774.

Towns, J., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A., Hazlewood, V., Lathrop, S., Lifka, D., Peterson, G.D., Roskies, R., Scott, J.R. and Wilkins-Diehr, N. (2014) 'Xsede: accelerating scientific discovery', *Computing in Science and Engineering*, September–October, Vol. 16, No. 5.

Veris Tech Inc. (2018) [online] http://www.veris.com (accessed March 2018).

Walt, S.v.d., Colbert, S.C. and Varoquaux, G. (2011) 'The numpy array: a structure for efficient numerical computation', *Computing in Science and Engineering*, Vol. 13, No. 2, pp.22–30.

Zhao, W., Ma, H. and He, Q. (2009) 'Parallel K-means clustering based on mapreduce', in *IEEE International Conference on Cloud Computing*, Springer, pp.674–679.