K-means in Space: A Radiation Sensitivity Evaluation

Kiri L. Wagstaff KIRI.WAGSTAFF@JPL.NASA.GOV
Benjamin Bornstein BENJAMIN.BORNSTEIN@JPL.NASA.GOV
Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91109 USA

Abstract

Spacecraft increasingly employ onboard data analysis to inform further data collection and prioritization decisions. However, many spacecraft operate in high-radiation environments in which the reliability of data-intensive computation is not known. This paper presents the first study of radiation sensitivity for k-means clustering. Our key findings are 1) k-means data structures differ in sensitivity, which is not determined solely by the amount of memory exposed; 2) no special radiation protection is needed below a data-set-dependent radiation threshold, enabling the use of faster, smaller, and cheaper onboard memory; and 3) subsampling improves radiation tolerance slightly, but the use of kd-trees unfortunately reduces tolerance. Our conclusions can help tailor k-means for use in future high-radiation environments.

1. Introduction

Onboard machine learning and data analysis methods are necessary for accommodating the increasingly ambitious goals of future spacecraft missions, such as autonomous and opportunistic exploration of remote targets like Titan and Europa. Support vector machines (SVMs) are already being used onboard the EO-1 Earth orbiter to classify hyperspectral image data and autonomously trigger additional observations of key events such as lake ice thaw (Castano et al., 2005). Clustering methods can provide highly compressed summaries of numeric data, and they are particularly useful for imaging applications, since they can provide segmentations of a scene as well as a summary of the main components (cluster centers). In an onboard setting, clustering results can inform local decisions about which follow-up observations to make and how to prioritize data for transmission to Earth. The summaries can also be transmitted preemptively, providing ground operators with a quick look that enables them to decide whether to consume the bandwidth needed to download the full data set.

Onboard clustering requires adapting to the onboard computing environment, which differs significantly from the desktop environment in terms of computational power, available memory, storage space, and the impact of a crash or reboot. Critically, onboard computing for planetary missions must also cope with high radiation doses, which can alter data during analysis and corrupt the results. These effects cannot be modeled as Gaussian data set noise, because 1) the impact of a flipped bit can vary by orders of magnitude, depending on the bit affected, and 2) ongoing data alteration occurs throughout the analysis process. If the analysis results are used to inform subsequent autonomous decisions about data collection and path planning, the possibility of corruption presents a major mission risk. This issue is generally addressed by using either fully radiation-hardened memory (usually with redundant components and voting) or special bit encoding schemes (consuming additional bits to detect and correct errors). Both strategies increase the total amount of memory required, therefore increasing mass and cost. Both can also result in memory with slower response times due to the additional complexity.

In this paper, we investigate how much radiation protection is needed and whether some (or all) of the data used by a clustering algorithm could be stored in smaller, cheaper, and faster unhardened RAM. No one has yet examined the innate radiation tolerance (or sensitivity) of any machine learning or data analysis algorithms. We seek to answer these questions: 1) What is the relationship between radiation-induced bit error rates and actual algorithm errors? 2) Which k-means variants are the most radiation-tolerant? 3) Which data structures (if any) could be stored in unhardened memory without sacrificing accuracy?

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

We analyzed the behavior of the k-means clustering algorithm (MacQueen, 1967), selected due to its simplicity, widespread use, and applicability to multiple domains. We ran k-means in environments with increasing radiation levels using a software radiation simulator (BITFLIPS), testing behavior on benchmark and satellite image data sets. We also studied the sensitivity of individual data structures used by the algorithm to determine which ones require the most radiation protection and which can make use of cheaper, faster, lower-mass memory. Finally, we evaluated two k-means variants with faster runtimes (and therefore less exposure to radiation): subsampling and k-means with kd-trees. Subsampling provided a slight reduction in sensitivity, while the use of kd-trees *increased* algorithm sensitivity.

Although motivated by space applications, this work also applies to any domain in which bit errors occur while clustering. Removing the assumption of incorruptible memory raises new reliability issues and opens up new areas for machine learning innovation.

2. Background and Related Work

Radiation affects RAM by flipping individual bits. This is referred to as a "single-event upset" or SEU. The number of SEUs experienced depends on the radiation environment and the physical configuration of memory. The amount of radiation protection provided by a component is characterized in terms of the SEUs experienced in low-Earth orbit (LEO) (Johnson Space Center, 1996):

	SEUs per bit per day (LEO)
Commercial	10^{-5}
Rad-tolerant	10^{-7} to 10^{-8}
Rad-hard	10^{-10} to 10^{-12}

Rad-hard SRAM has been developed that admits less than 10^{-10} SEUs per bit-day (Hoang et al., 2007), but it is typically 3–10 times more expensive than commercial RAM. A common hardware technique for achieving this protection is Triple-Modular Redundancy (TMR), in which three identical components perform the same memory operations and then vote on the result (Lyons & Vanderkulk, 1962). Alternatively, error detection and correction (EDAC) codes provide software-based protection, employing a "memory scrubber" process to run continually in the background to correct errors (Shirvani et al., 2000). Algorithm-specific tests can be devised to detect SEU-induced computational errors, although these assume error-free data storage in memory (Turmon et al., 2003). It is worth noting that these devices and techniques were generally developed for use in Earth or Mars orbit; radiation levels are even higher in orbit around planets with strong magnetic fields such as Jupiter or Saturn.

Most prior work analyzing the sensitivity of k-means has focused on its sensitivity to the choice of initial starting point (Bradley & Fayyad, 1998; Peña et al., 1999, etc.) or its sensitivity to noise in the data. The latter has exclusively been studied for what we might call "static" noise, which is present in the data prior to clustering and does not change during analysis. This kind of noise can be addressed empirically by assigning density-based weights to decrease the influence of (presumed noisy) outliers (Jolion & Rosenfeld, 1989) or, if the type of noise is known (e.g., Gaussian measurement error), it can be modeled as part of the clustering process (Kumar & Patel, 2007).

The problem we study presents unique challenges because the noise introduced by SEUs is dynamic and unpredictable. Further, SEUs affect not only the input data but also any data structures that are created and stored in memory. Finally, the magnitude of a single SEU varies widely depending on the bit struck and the data storage specification, e.g., 2^0 to 2^7 in a simple 8-bit character case. For floating-point values, the impact can be dramatically worse, especially if the SEU strikes a bit in the exponent. On the other hand, if SEUs strike bits that are not used, or that are updated with correct values before their next read, the algorithm may be able to withstand significant radiation levels without a loss in accuracy. In this paper, we investigate sensitivity quantitatively for k-means clustering algorithms.

3. Radiation Sensitivity Analysis

This section presents the experimental methodology we used to study radiation sensitivity, which can be applied to any machine learning algorithm. It also describes the different k-means methods we studied.

3.1. Methodology

We developed a software radiation simulator called BITFLIPS (Basic Instrumentation Tool for Fault Localized Injection of Probabilistic SEUs) that is based on the Valgrind debugging/profiling tool (Nethercote & Seward, 2007). BITFLIPS monitors all data structures used and injects simulated SEUs (probabilistically) at a user-specified rate (in SEUs per kB per second). It also allows the selective exposure of individual data structures. SEUs are injected uniformly at random; in reality, the physical layout of SRAM will affect the distribution of hits.

This study focuses exclusively on the impact of SEUs on memory used to store the data to be clustered and memory consumed by data structures generated in the process of clustering. We did not simulate SEUs in the processor, registers, or code memory (generally much smaller than secondary storage RAM).

We evaluated clustering performance using the Adjusted Rand Index or ARI (Hubert & Arabie, 1985), which calculates the agreement of pairwise assignments and normalizes for the expected value. An ARI of 1.0 indicates perfect agreement, and an ARI of 0 is that expected by random chance; it is negative for anti-correlated partitions. Perturbing radiation destroys any convergence guarantees, so we capped the number of k-means iterations at 30, well above the maximum number of iterations (11) needed for either data set.

3.2. K-means Algorithms

We implemented the basic k-means algorithm as well as two variants designed to increase its efficiency in different ways. K-means (Algorithm 1) takes in a data set D containing n items and a user-specified number of clusters k. It returns the list of clusters C, each represented by its centroid. The clusters are initialized by selecting k items randomly from D, and then the n items are arbitrarily assigned to cluster 0. The main loop of the algorithm alternates between two steps: assign each item d_i to its closest cluster center (lines 9-12), and update each cluster centroid to be the mean of its constituent items (lines 13-15). We used the Euclidean distance metric. Iterations continue until no item assignments change.

In assigning items to their closest clusters, k-means greedily seeks a solution with minimal variance V from items to their assigned cluster centers:

$$V = \frac{1}{n} \sum_{i}^{n} dist(d_i, c_{a_i}) \quad (1)$$

where $dist()$ is the same metric used in line 7 of Algorithm 1, usually Euclidean.

Algorithm 1 K-means (MacQueen, 1967)

1: **Inputs:** data set D, number of clusters k
2: **Outputs:** clusters $C = \{c_j, j = 1 \dots k\}$
3: Let $n = |D|$.
4: Initialize k clusters with randomly chosen $d \in D$.
5: Assign all items to cluster 0, $a_i = 0, i = 1 \dots n$.
6: **repeat**
7: Assign each $d \in D$ to its closest cluster in C.
8: Update each cluster c_j as mean of $\{d_i | a_i = j\}$.
9: **until** A does not change

Table 1. Vulnerability of k-means to radiation-induced SEUs (single-event upsets), expressed in terms of individual data structures, including how much memory is exposed and which lines of Algorithm 1 are affected.

		Bytes of RAM exposed	Alg. 1 lines impacted
Assignments	a_i	$4n$	8, 9
Centers	c_j	$4kf$	7
Data	d_i	$4nf$	4, 7, 8

Radiation Impact. The data structures used by k-means are the assignments of items to clusters, the cluster centers, and the data itself. Assuming 4-byte data values (floats) and 4-byte cluster assignments (ints), and letting f be the number of features, we can calculate the amount of memory consumed by each data structure (see Table 1). We also identify which lines of Algorithm 1 depend on each data structure (by reading the stored values). These are the locations where k-means is vulnerable to SEUs.

Since radiation is cumulative, an obvious question is whether a more efficient version of k-means could tolerate higher radiation rates. We investigated two variants on the k-means algorithm that were designed to speed up computation. Due to reduced runtime, they should experience fewer SEUs at the same level of radiation.

3.2.1. K-means with Subsampling

A simple approach to reducing runtime is to perform an initial pass of k-means with a subsample of the data set to obtain a set of approximate clusters $\hat{C}$ (Bradley & Fayyad, 1998). Next, k-means is run on the entire data set, initializing the clusters with $\hat{C}$ rather than with randomly chosen items from the data set (see line 4 in Algorithm 1). No additional memory is required, but both phases of k-means are subject to SEUs.

3.2.2. kd-k-means

It is possible to cluster more efficiently by building a kd-tree over the data's feature space and then traversing that tree to assign clusters to groups of items at a time (Alsabti et al., 1998; Kanungo et al., 1999; Pelleg & Moore, 1999). The binary kd-tree is a hierarchical data structure that starts with a cell covering the entire data set and successively splits the feature space into smaller cells. We refer to the tree node and its associated feature-space cell interchangeably. By specifying the maximum number of items contained in a leaf, the size and granularity of the resulting tree can be varied.

Algorithm 2 KD-filter (Kanungo et al., 2002)

```
1: Inputs: node t, candidate clusters C
2: if t is a leaf then
3:    for d ∈ t do
4:       Assign d to the closest c' ∈ C.
5:       Set c'_n = c'_n + 1 and c'_LS = c'_LS + d.
6:    end for
7: else
8:    Select the closest c' ∈ C to the center of t's cell.
9:    for c ∈ C \ {c'} do
10:      if isCloser(c', c, t) then
11:         Update C = C \ {c}. // Filter candidates
12:      end if
13:   end for
14:   if |C| = 1 then
15:      C = {c'}, so assign all d ∈ t to c'.
16:      Set c'_n = c'_n + t_n and c'_LS = c'_LS + t_LS.
17:   else
18:      KD-filter(t.left, C) // Recurse left
19:      KD-filter(t.right, C) // Recurse right
20:   end if
21: end if
```

To split a cell, we selected the feature with the largest span from minimum to maximum values within the cell (Alsabti et al., 1998). We used the median value as the splitting threshold θ (Pelleg & Moore, 1999), assigning items with values $\leq \theta$ to the "left" child and items with larger values to the "right" child. Each node in the kd-tree stores statistics sufficient to enable k-means clustering: items in the cell, t_n; the linear sum of the items, t_{LS}; and the minimum $t_{MIN,f}$ and maximum $t_{MAX,f}$ values in each dimension. Each cluster c is represented by the number of items it contains, c_n, and the linear sum of its constituents, c_{LS}.

When clustering, the kd-tree is used to speed up a single k-means iteration by recursively propagating a list of cluster centers down the tree until a node is reached in which all points can be assigned to a single cluster. The main challenge in using a kd-tree effectively is in how decisions are made to "filter" (Alsabti et al., 1998; Kanungo et al., 1999; Kanungo et al., 2002) or "blacklist" (Pelleg & Moore, 1999) the list of candidate clusters at each node. We used the filtering algorithm of Kanungo et al. (2002), shown in Algorithm 2. It begins with a kd-tree node t and list of candidate clusters C. If t is a leaf, a single greedy assignment pass is done over all items in the leaf (lines 3-6). Otherwise, the cluster closest to the center of the cell is selected as the best candidate c' and all other clusters are checked for possible filtering (lines 9-13). The $isCloser(c', c, t)$ function returns true if cluster c' is closer than cluster c to every point in cell t. If so, cluster c can be removed from the list of candidates, since it is dominated by cluster c'. The details of how to calculate $isCloser()$ efficiently, using $t_{MIN,f}$ and $t_{MAX,f}$, are provided by Kanungo et al. (2002).

Radiation Impact. Given a kd-tree T, we replace lines 7-8 in Algorithm 1 with a call to KD-filter($T.root$, C). This approach has the potential to drastically decrease runtime, although it is less effective if dimensionality is high (Pelleg & Moore, 1999). However, it also increases the radiation exposure because the kd-tree must be stored in memory. In our implementation, each node of the kd-tree consumes 4 bytes to store t_n (an integer) and $3f$ 4-byte floats to store t_{LS}, t_{MIN}, and t_{MAX} for f dimensions, for a total of $12f+4$ bytes. The number of nodes depends on the data distribution in feature space and the maximum size of a leaf node.

As a practical note, we found that kd-k-means absolutely required that t_n be protected from radiation, since this variable was used to loop over all points within a node. When we later refer to exposing "the kd-tree", this includes only t_{LS}, t_{MIN}, and t_{MAX}. We also did not expose the algorithm while building the kd-tree, only while clustering. Finally, to avoid segmentation faults, we stored the kd-tree in a dynamically resized array of nodes rather than using pointers.

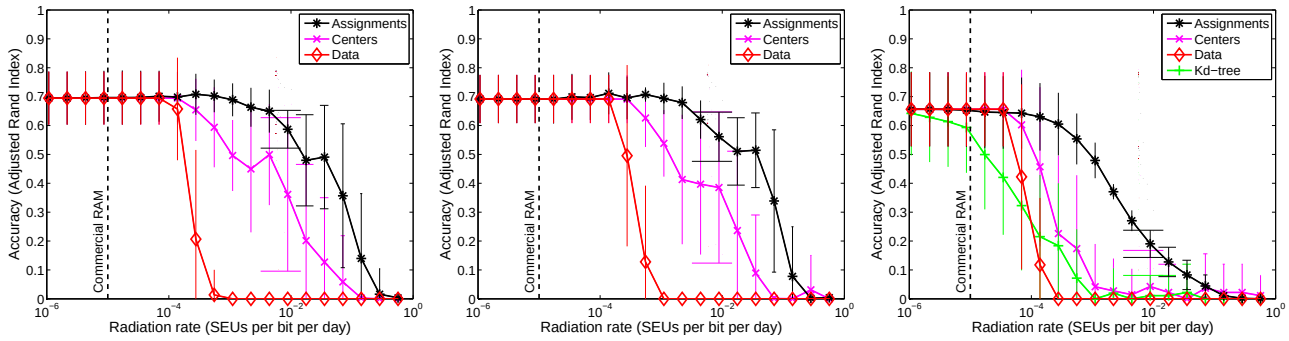
4. Experimental Results

4.1. Benchmark: Iris Data Set

We first examine the impact of radiation when clustering the Iris data set (Asuncion & Newman, 2007), with $n = 150$ items, $k = 3$ classes (types of iris), and $f = 4$ features (length and width of sepal and petal). We chose this data set to due to its wide familiarity and the known property that two of its three classes are not linearly separable. Because of this property, we do not expect k-means clustering to find the correct partition, which has a much higher variance than the local optima to which k-means will be drawn.

4.1.1. Exposing All Data Structures

We tested regular k-means, k-means with subsampling, and kd-k-means at a variety of radiation rates (see Figure 1). As the radiation rate increased beyond 6.9×10^{-5}, regular k-means accuracy dropped from an average of 0.69 to 0.32 and then 0.01. However, at the level of protection provided by simple commercial RAM (1.0×10^{-5}), no impact to performance was observed, even though SEUs were happening (up to 117 per run). Therefore, for clustering this data set, radiation-hardened memory would be entirely unnec-



(a) Regular k-means (b) K-means with 40% subsample (c) Kd-k-means (leaf size = 1)

Figure 2. Radiation sensitivity of individual data structures used by k-means algorithms (Iris data set). Each curve is the average over 40 trials, with bars showing one standard deviation.

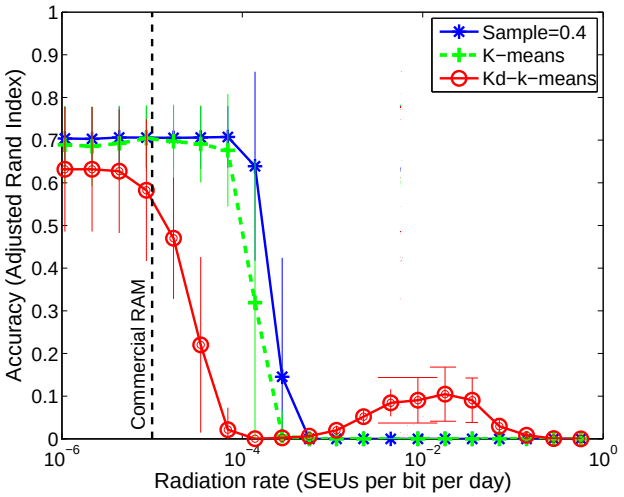


Figure 1. Radiation sensitivity of three k-means variants for the Iris data set ($n = 150$, $k = 3$, $f = 4$). All data structures (data, assignments, cluster centers, and kd-trees) were exposed to radiation. Each curve is the average over 100 trials, with bars showing one standard deviation. The vertical bar shows the level of protection provided by commercial RAM. Note log scale on the x axis.

essary in LEO. However, these results were obtained on 3-GHz CPUs, much faster than any CPU used onboard spacecraft. For example, the RAD750 runs at 133 MHz, an order of magnitude slower, so it would experience more SEUs due to a longer runtime. Scaled by CPU speed, the estimated rate at which performance would drop on a RAD750 would instead be 1.5×10^{-6}, which would likely require rad-tolerant (although not rad-hard) RAM.

The other two algorithms both showed interesting behavior. Subsampling 40% of the data in the first clustering pass provided a small increase in radiation tolerance at radiation levels exceeding 1.0×10^{-4} (similar results were observed for subsamples from 10-50% of the data).

Kd-k-means showed significantly more sensitivity to radiation than the regular k-means algorithm. Any reduction in runtime was offset by the increased memory consumption (and exposure). Kd-k-means also exhibited a brief *improvement* in performance for SEU rates between 1.1×10^{-3} and 7.1×10^{-2}. While initially intriguing, this effect is spurious. At rates higher than 1×10^{-4}, the algorithm always reached the 30-iteration cap and terminated with a partial solution. Between 1×10^{-4} and 1×10^{-3}, the algorithm assigned all items to a single cluster, yielding an ARI of 0.0. Between 1×10^{-3} and 7.1×10^{-2}, radiation effects were felt in the cluster assignment values and most of the items were assigned to non-existent clusters. Unlike regular k-means and subsampling k-means, however, some remaining information in the kd-tree permitted this algorithm to assign a few items to valid clusters, increasing the ARI above 0.0. For rates greater than 7.1×10^{-2}, the algorithm had no items assigned to any valid clusters (ARI = 0.0).

4.1.2. Exposing Single Data Structures

We also examined the sensitivity of individual data structures used by each algorithm. Figure 2 shows performance for k-means, subsampling, and kd-k-means when either the item assignments, the cluster centers, the input data, or the kd-tree (if used) was exposed to radiation. Initially we might assume that sensitivity increases with the amount of memory exposed. Indeed, the most impact was observed when the input data was exposed, which at 2400 bytes was the largest component of memory consumption. However, all three algorithms were least sensitive to exposure of the item assignments (600 bytes); errors in the cluster centers (48 bytes) had much more impact. Modifica-

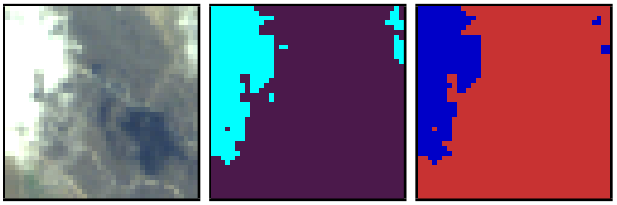
tions to a cluster center can affect assignment decisions for several items (Algorithm 1, line 7), while perturbing an item assignment may have only minor impact to a single cluster center (line 8). Here, subsampling provided a slight, but not significant, increase in tolerance. For kd-k-means, exposing the kd-tree hurt performance more than exposing the input data at lower radiation levels, but had less impact at higher levels.

An interesting, and concerning, effect of using kd-trees to cluster is revealed by comparing the individual "centers" curves. The curve in Figure 2(c) drops off much more quickly than it does in Figure 2(a), indicating that the use of a kd-tree in fact increases the sensitivity of the algorithm to SEUs in the cluster centers, even when the kd-tree itself is protected. Each iteration of kd-k-means propagates the cluster centers down through the kd-tree, and a perturbed cluster center is likely to be filtered from the candidate list earlier than it should be, at which point it will never be considered as a possible host for any item lower in the tree. In contrast, regular k-means will at least consider each possible cluster as a potential host for each item, so small changes can be tolerated, and since the cluster centers are recomputed at each iteration from the (protected) data, those errors will be erased. In effect, the efficiency advantage of kd-k-means, which leverages geometric reasoning to reduce the number of cluster-item associations to consider, increases its sensitivity to radiation.

4.2. Satellite Multispectral Data

We are not aware of any application in which there is a need to analyze the Iris data set in a high-radiation environment. Therefore, we also experimented with real data collected in Earth orbit. This data consists of hyperspectral observations from the Hyperion instrument onboard the EO-1 Earth orbiter. Although Hyperion observes at 242 wavelengths, onboard processing can only be done on a (selectable) subset of up to 12 bands (Castano et al., 2005). We extracted the same 11 bands used by the onboard pixel SWIL (snow, water, ice, and land) classifier described by Castano et al. (2005). They range from 426 to 2284 nm, covering visible and near-IR wavelengths.

The data set we used was collected on October 3, 2002. It covers part of the Quinghai Province in China and includes clouds, cultivated land, and lakes. The original image is very large (256 by 2904 pixels) and, while it can be clustered in a reasonable amount of time in regular operation, the additional overhead imposed by BITFLIPS (tracking each exposed block of memory) precluded us from analyzing it in simulation. Instead,



(a) RGB (b) Labels (c) K-means

Figure 3. Satellite image of Quinghai Province, China, observed on October 3, 2002. (a) The 11-band data shown in RGB. Clouds are on the left and right sides, with land and a lake between them. (b) Manually assigned labels: bright = cloudy and dark = clear. (c) Regular k-means clustering output (using all 11 spectral bands, no radiation) achieves 0.940 agreement with the labels.

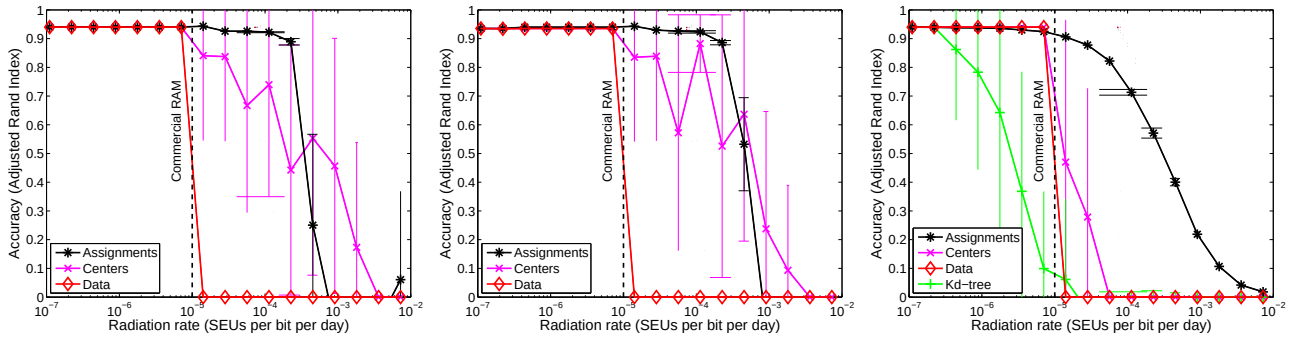
we focused on a small region within the image that contains 1600 pixels, an order of magnitude more data points than the Iris data set contains. The dimensionality is also higher, leading to a much larger memory profile just to store the data (70,400 bytes). This region is shown in Figure 3(a) and contains all of the elements of the larger image (cloud, land, and water).

Clustering satellite data is generally an unsupervised process designed to summarize the data or identify major components. However, we labeled the pixels in this image to provide a reference against which to evaluate the clustering results. We assigned each pixel to one of two classes: cloudy and clear (see Figure 3(b)). This is a relatively easy clustering problem for which the regular k-means algorithm achieves 0.940 accuracy (ARI), as shown in Figure 3(c).

4.2.1. Exposing Single Data Structures

Due to space limitations, we focus on the Quinghai results obtained when exposing individual data structures (Figure 4). The behavior was similar to that observed on the Iris data set, with some important differences. Due to the larger data set and longer run times, performance degraded for all algorithms at lower radiation levels. Commercial RAM would no longer provide sufficient protection for the input data, even when analyzed by a 3-GHz processor. However, radiation-tolerant RAM (ensuring a rate of less than 1×10^{-7} to 10^{-8}) would be sufficient protection when using regular or subsampling k-means, even scaling processing time down to a 133-MHz processor. Further, the data assignments could safely be stored in commercial RAM.

Unlike the experience with Iris, subsampling (10%) did not improve radiation sensitivity (Figure 4(b)). According to Figure 4(c), kd-k-means definitely requires



(a) Regular k-means (b) K-means with 10% subsample (c) Kd-k-means (leaf size = 100)

Figure 4. Radiation sensitivity of individual data structures used by k-means algorithms (Quinghai Province data set). Each curve is the average over 10 trials, with bars showing one standard deviation.

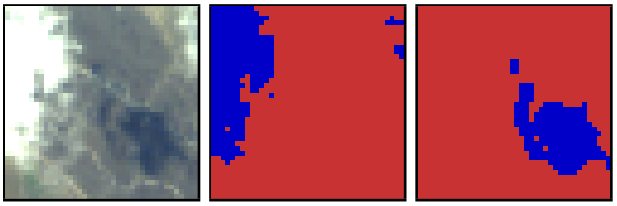
rad-tolerant memory for the kd-tree and input data. The other components could be stored in commercial RAM, at some risk of errors. On a 133-MHz processor, all data structures should be stored in rad-tolerant or even rad-hard memory.

Because this is image data, we can gain additional insights by visually browsing the output partitions. Figure 5(b) shows a superior result achieved *only* in the presence of radiation, with an ARI of 0.946 (radiation rate 1.3×10^{-6}). This solution, which provides a better match to the labels, has a higher variance (3.061×10^6) than that of the solution consistently found by k-means without radiation (3.058×10^6) and therefore could only be reached with outside influence. In this case, it was achieved by the k-means algorithm running with the item assignments exposed.

Figure 5(c) shows another result obtained by k-means with assignments exposed, at a higher radiation rate (6.7×10^{-4}). This solution does not match the labels at all, and in fact is anti-correlated with them, yielding an ARI of -0.102. However, visual inspection of the data indicates that this result is not random incoherence. The small cluster is associated with the lake instead of the clouds. This result has extremely high variance (1.321×10^7) and almost certainly would never have been found by regular k-means.

5. Conclusions and Future Work

We have conducted the first quantitative analysis of radiation sensitivity for the k-means clustering algorithm. The same process can be applied to quantify the radiation sensitivity of any machine learning algorithm. Given relevant data and knowledge about the rate of radiation in the target environment, we can estimate the amount of radiation protection needed. Specifically, we have focused on



(a) RGB (b) ARI 0.946 (c) ARI -0.102

Figure 5. Radiation-perturbed clustering output for Quinghai Province. (a) Original data. (b) A solution superior to any found by k-means without radiation. (c) A "poor" solution with a reasonable physical interpretation.

the impact of radiation-induced single-event upsets (SEUs) which flip individual bits in RAM. We have observed, in benchmark and satellite data sets, that there is a threshold radiation rate below which, although SEUs are happening, the clustering results are not affected. If this threshold is high enough (1×10^{-5} SEUs/bit/day), regular commercial RAM could be used in space without any impact to performance.

Using the BITFLIPS software radiation simulator, we tested k-means and two variants designed to decrease runtime, to determine whether they could also decrease sensitivity. Subsampling to generate better initial cluster centers provided a small benefit on the Iris data set. In contrast, k-means using a kd-tree to speed up clustering increased the algorithm's sensitivity on both data sets, especially if the cluster centers were exposed. We also found that the sensitivity of individual components (assignments, centers, and input data) differed greatly, reflecting how they were used by the algorithm rather than strictly depending on the amount of memory exposed.

We have only scratched the surface of this domain. There are many other k-means variants available for consideration as onboard clustering algorithms. A

clear next step is to quantify the sensitivity of supervised learning methods such as support vector machines, which are already in use in Earth orbit (Castano et al., 2005). Further, we would like to issue a challenge to the community to develop k-means clustering algorithms that specifically aim for reduced radiation sensitivity. It is possible that careful reordering of actions could reduce the impact of SEUs. Pairwise constraints or other domain knowledge could potentially be used to reduce sensitivity, although that information, too, would be subject to radiation. Ultimately, we seek "rad-hard" machine learning algorithms that perform robustly in extreme environments.

Acknowledgments

We thank the anonymous reviewers for their thoughtful suggestions for improving the paper. The experiments were made possible by the JPL Supercomputing Project, which is funded by the JPL Office of the Chief Information Officer. This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

Alsabti, K., Ranka, S., & Singh, V. (1998). An efficient k-means clustering algorithm. *Proceedings of the 1st Workshop on High Performance Data Mining*.

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Bradley, P. S., & Fayyad, U. M. (1998). Refining initial points for k-means clustering. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 91–99).

Castano, R., Mazzoni, D., Tang, N., Doggett, T., Chien, S., Greeley, R., Cichy, B., & Davies, A. (2005). Learning classifiers for science event detection in remote sensing imagery. *Proceedings of the Eighth International Symposium on Artificial Intelligence, Robotics, and Automation in Space*.

Hoang, T., Ross, J., Doyle, S., Rea, D., CHan, E., Neiderer, W., & Bumgarner, A. (2007). A radiation hardened 16-Mb SRAM for space applications. *Proc. of the IEEE Aerospace Conference* (pp. 1–6).

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*, 193–218.

Johnson Space Center (1996). Space radiation effects on electronic components in low-Earth orbit. NASA Reliability Practice No. PD-ED-1258, `http://www.nasa.gov/offices/oce/llis/0824.html`.

Jolion, J.-M., & Rosenfeld, A. (1989). Cluster detection in background noise. *Pattern Recognition*, *22*, 603–607.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C., Silverman, R., & Wu, A. Y. (1999). Computing nearest neighbors for moving points and applications to clustering. *Proceedings of the Tenth ACM-SIAM Symposium on Discrete Algorithms* (pp. S931–S932).

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 881–892.

Kumar, M., & Patel, N. R. (2007). Clustering data with measurement errors. *Computational Statistics and Data Analysis*, *51*, 6084–6101.

Lyons, R. E., & Vanderkulk, W. (1962). The use of triple-modular redundancy to improve computer reliability. *IBM Journal*, 200–209.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Symposium on Math, Statistics, and Probability* (pp. 281–297).

Nethercote, N., & Seward, J. (2007). Valgrind: A framework for heavyweight dynamic binary instrumentation. *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation* (pp. 89–100).

Peña, J. M., Lozano, J. A., & Larrañaga, P. (1999). An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, *20*, 1027–1040.

Pelleg, D., & Moore, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases* (pp. 277–281).

Shirvani, P. P., Saxena, N. R., & McCluskey, E. J. (2000). Software-implemented EDAC protection against SEUs. *IEEE Transactions on Reliability*, *49*, 273–284.

Turmon, M., Granat, R., Katz, D., & Lou, J. (2003). Tests and tolerances for high-performance software-implemented fault detection. *IEEE Transactions on Computers*, *52*, 579–591.