Web-Scale K-Means Clustering

D. Sculley
Google, Inc. Pittsburgh. PA USA
dsculley@google.com

ABSTRACT

We present two modifications to the popular k-means clustering algorithm to address the extreme requirements for latency, scalability, and sparsity encountered in user-facing web applications. First, we propose the use of mini-batch optimization for k-means clustering. This reduces computation cost by orders of magnitude compared to the classic batch algorithm while yielding significantly better solutions than online stochastic gradient descent. Second, we achieve sparsity with projected gradient descent, and give a fast ϵ-accurate projection onto the $L1$-ball. Source code is freely available: `http://code.google.com/p/sofia-ml`

Categories and Subject Descriptors

I.5.3 [**Computing Methodologies**]: Pattern Recognition—*Clustering*

General Terms

Algorithms, Performance, Experimentation

Keywords

unsupervised clustering, scalability, sparse solutions

1. CLUSTERING AND THE WEB

Unsupervised clustering is an important task in a range of web-based applications, including grouping search results, near-duplicate detection, and news aggregation to name but a few. Lloyd's classic k-means algorithm remains a popular choice for real-world clustering tasks [6]. However, the standard batch algorithm is slow for large data sets. Even optimized batch k-means variants exploiting triangle inequality [3] cannot cheaply meet the latency needs of user-facing applications when clustering results on large data sets are required in a fraction of a second.

This paper proposes a mini-batch k-means variant that yields excellent clustering results with low computation cost on large data sets. We also give methods for learning sparse cluster centers that reduce storage and network cost.

2. MINI-BATCH K-MEANS

The k-means optimization problem is to find the set C of cluster centers $\mathbf{c} \in \mathbb{R}^m$, with $|C| = k$, to minimize over a set X of examples $\mathbf{x} \in \mathbb{R}^m$ the following objective function:

$$\min \sum_{\mathbf{x} \in X} ||f(C, \mathbf{x}) - \mathbf{x}||^2$$

Here, $f(C, \mathbf{x})$ returns the nearest cluster center $\mathbf{c} \in C$ to $\mathbf{x}$ using Euclidean distance. It is well known that although this problem is NP-hard in general, gradient descent methods converge to a local optimum when seeded with an initial set of k examples drawn uniformly at random from X [1].

The classic batch k-means algorithm is expensive for large data sets, requiring $O(kns)$ computation time where n is the number of examples and s is the maximum number of non-zero elements in any example vector. Bottou and Bengio proposed an online, stochastic gradient descent (SGD) variant that computed a gradient descent step on one example at a time [1]. While SGD converges quickly on large data sets, it finds lower quality solutions than the batch algorithm due to stochastic noise [1].

Algorithm 1 Mini-batch k-Means.

```
Given: k, mini-batch size b, iterations t, data set X
Initialize each c ∈ C with an x picked randomly from X
v ← 0
for i = 1 to t do
    M ← b examples picked randomly from X
    for x ∈ M do
        d[x] ← f(C, x)    // Cache the center nearest to x
    end for
    for x ∈ M do
        c ← d[x]          // Get cached center for this x
        v[c] ← v[c] + 1   // Update per-center counts
        η ← 1 / v[c]      // Get per-center learning rate
        c ← (1 − η)c + ηx // Take gradient step
    end for
end for
```

We propose the use of mini-batch optimization for k-means clustering, given in Algorithm 1. The motivation behind this method is that mini-batches tend to have lower stochastic noise than individual examples in SGD (allowing convergence to better solutions) but do not suffer increased computational cost when data sets grow large with redundant examples. We use per-center learning rates for fast convergence, in the manner of [1]; convergence properties follow closely from this prior result [1].

Experiments. We tested the mini-batch k-means against both Lloyd's batch k-means [6] and the SGD variant of [1]. We used the standard `RCV1` collection of documents [4] for

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

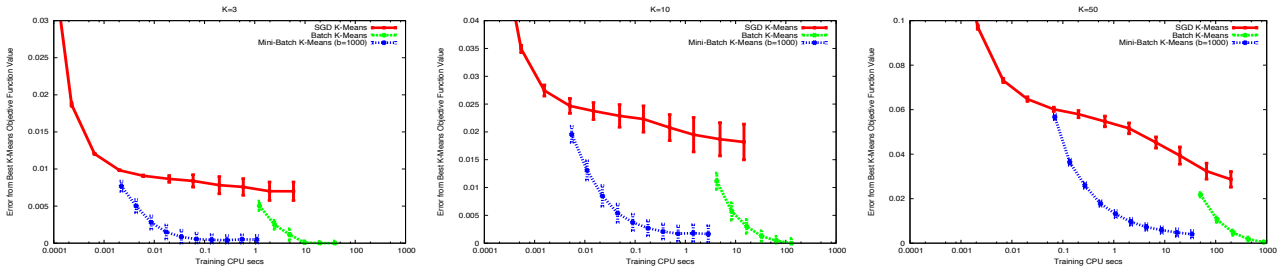

Figure 1: Convergence Speed. The mini-batch method (blue) is orders of magnitude faster than the full batch method (green), while converging to significantly better solutions than the online SGD method (red).

our experiments. To assess performance at scale, the set of 781,265 examples were used for training and the remaining 23,149 examples for testing. On each trial, the same random initial cluster centers were used for each method. We evaluated the learned cluster centers using the k-means objective function on the held-out test set; we report fractional error from the best value found by the batch algorithm run to convergence. We set the mini-batch b to 1000 based on separate initial tests; results were robust for a range of b.

The results (Fig. 1) show a clear win for mini-batch k-means. The mini-batch method converged to a near optimal value several orders of magnitude faster than the full batch method, and also achieved significantly better solutions than SGD. Additional experiments (omitted for space) showed that mini-batch k-means is several times faster on large data sets than batch k-means exploiting triangle inequality [3].

For small values of k, the mini-batch methods were able to produce near-best cluster centers for nearly a million documents in a fraction of a CPU second on a single ordinary 2.4 GHz machine. This makes real-time clustering practical for user-facing applications.

3. SPARSE CLUSTER CENTERS

We modify mini-batch k-means to find sparse cluster centers, allowing for compact storage and low network cost. The intuition for seeking sparse cluster centers for document clusters is that term frequencies follow a power-law distribution. Many terms in a given cluster will only occur in one or two documents, giving them very low weight in the cluster center. It is likely that for a locally optimal center $\mathbf{c}$, there is a nerby point $\mathbf{c}'$ with many fewer non-zero values.

Sparsity may be induced in gradient descent using the projected-gradient method, projecting a given $\mathbf{v}$ to the nearest point in an $L1$-ball of radius λ after each update [2]. Thus, for mini-batch k-means we achieve sparsity by performing an $L1$-ball projection on each cluster center $\mathbf{c}$ after each mini-batch iteration.

Algorithm 2 ϵ-L1: an ϵ-Accurate Projection to L1 Ball.

Given: ϵ tolerance, L1-ball radius λ, vector $\mathbf{c} \in \mathbb{R}^m$
if $||\mathbf{c}||_1 \leq \lambda + \epsilon$ **then exit**
$upper \leftarrow ||\mathbf{c}||_\infty$; $lower \leftarrow 0$; $current \leftarrow ||\mathbf{c}||_1$
while $current > \lambda(1+\epsilon)$ **or** $current < \lambda$ **do**
  $\theta \leftarrow \frac{upper+lower}{2.0}$ // Get $L1$ value for this θ
  $current \leftarrow \sum_{\mathbf{c}_i \neq 0} \max(0, |\mathbf{c}_i| - \theta)$
  if $current \leq \lambda$ **then** $upper \leftarrow \theta$ **else** $lower \leftarrow \theta$
end while
for $i = 1$ to m **do**
  $\mathbf{c}_i \leftarrow sign(\mathbf{c}_i) * \max(0, |\mathbf{c}_i| - \theta)$ // Do the projection
end for

Fast $L1$ Projections. Applying $L1$ constraints to k-means clustering has been studied in forthcoming work by Witten and Tibshirani [5]. There, a hard $L1$ constraint was applied in the full batch setting of maximizing between-cluster distance for k-means (rather than minimizing the k-means objective function directly); the work did not discuss how to perform this projection efficiently.

The projection to the $L1$ ball can be performed effectively using, for example, the linear time $L1$-ball projection algorithm of Duchi *et al.* [2], referred to here as LTL1P. We give an alternate method in Algorithm 2, observing that the exact $L1$ radius is not critical for sparsity. This simple approximation algorithm uses bisection to find a value θ that projects $\mathbf{c}$ to an $L1$ ball with radius between λ and $(1+\epsilon)\lambda$. Our method is easy to implement and is also significantly faster in practice than LTL1P due to memory concurrency.

METHOD	λ	#NON-ZERO'S	TEST OBJECTIVE	CPUs
full batch	-	200,319	0 (baseline)	133.96
LTL1P	5.0	46,446	.004 (.002-.006)	0.51
ϵ-L1	5.0	44,060	.007 (.005-.008)	0.27
LTL1P	1.0	3,181	.018 (.016-.019)	0.48
ϵ-L1	1.0	2,547	.028 (.027-.029)	0.19

Results. Using the same set-up as above, we tested Duchi *et al.*'s linear time algorithm and our ϵ-accurate projection for mini-batch k-means, with a range of λ values. The value of ϵ was arbitrarily set to 0.01. We report values for $k = 10$, $b = 1000$, and $t = 16$ (results for other values qualitatively similar). Compared with the full batch method, we achieve much sparser solutions. The approximate projection is roughly twice as fast as LTL1P and finds sparser solutions, but gives slightly worse performance on the test set. These results show that sparse clustering may cheaply be achieved with low latency for user-facing applications.

4. REFERENCES

[1] L. Bottou and Y. Bengio. Convergence properties of the kmeans algorithm. In *Advances in Neural Information Processing Systems*. 1995.

[2] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, 2008.

[3] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML '03: Proceedings of the 20th international conference on Machine learning*, 2003.

[4] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5, 2004.

[5] D. Witten and R. Tibshirani. A framework for feature selection in clustering. To Appear: *Journal of the American Statistical Association*, 2010.

[6] X. Wu and V. Kumar. *The Top Ten Algorithms in Data Mining*. Chapman & Hall/CRC, 2009.