k-means, gradients, and smart initialization

Noah Bergam NJB2154@COLUMBIA.EDU
Department of Mathematics
Columbia University
New York, NY 10027

Abstract

We discuss the classic k-means clustering problem from the perspective of gradient-based optimization. This is motivated by the fact that the popular Lloyd's method for k-means is equivalent to the Newton-Raphson method, and approximate versions of Lloyd's method are equivalent to stochastic gradient descent. In this writeup, we discuss Lloyd's method in the broader context of k-means hardness and approximability. We then provide a fleshed-out proof of the convergence of k-means++, an initialization scheme for Lloyd's algorithm which reduces its expected worst-case approximation ratio from unbounded to $O(\log(k))$.
Keywords: unsupervised learning, k-means clustering, gradient methods

1. Introduction

k-means is a classic clustering problem and among the most well-studied topics in unsupervised learning, by both practitioners and theorists. It is an NP-hard optimization problem for $k \geq 2$ (Dasgupta, 2008) and $d \geq 2$ (Vattani, 2009a), and it is even NP-hard to approximate for variable k and d (Awasthi et al., 2015). Nonetheless, a wide variety of approximation schemes have been developed. In this writeup, we work towards a proof of correctness for the $O(\log k)$ randomized approximation algorithm known as k-means++ (Arthur and Vassilvitskii, 2007).

The standard formulation of the k-means problem is as follows: given $X = (x_1, ..., x_n) \subset \mathbb{R}^d$, find a set of k *cluster centers* $W = (w_1, ..., w_k) \subset \mathbb{R}^d$ which minimize the sum of squared distance from each point to its closest center. In other words, we seek to minimize the following objective:

$$(w_1^*, ..., w_k^*) = \arg\min_W \underbrace{\sum_{x \in X} \min_{w \in W} ||x - w||^2}_{\phi_W(X)}$$

If $\phi_W(x)$ denotes the distance between a set of cluster centers W and a point x, then the k-means objective can be rewritten as $L_W(X) = \sum_{x \in X} \phi_W(x)^2$.

The k-means problem can be seen alternatively as a *continuous* optimization problem (finding the best placement of centers in Euclidean space) and a *combinatorial* one (finding the best partition of the data, see section 1.1). In this article, we explore both perspectives, but we place particular focus on the continuous perspective, exploring how Lloyd's method—the simplest and most well-known algorithm for k-means—is a gradient-based optimization technique, namely, an instantiation of Newton's method.

The article proceeds as follows:

©2023 Noah Bergam.

- We review basic results on k-means hardness and approximation, including (Dasgupta, 2008)'s proof of k-means hardness and (Wang and Song, 2011)'s dynamic programming solution to k-means in one dimension.
- We review basic results on gradient descent and Newton optimization, before discussing the observation by (Bottou and Bengio, 1994) that Lloyd's method is simply the Newton-Raphson method in disguise.
- Finally, we prove the $O(\log k)$ approximation guarantee of k-means++, which uses a probabilistic farthest-first scheme to sample the starting cluster centers.

The motivation of this study is the rising interest in understanding gradient-based techniques in the context of neural network training and other continuous optimization problems. Results like k-means++ illustrate the importance of initialization and can be informative for future work in using gradient-based methods to solve even combinatorial problems.

1.1 Combinatorial Formulation

Observe that any given placement of cluster centers induces a partition of the data points into "clusters" $C_1, ..., C_k$ based on their closest center (in fact, the centers partition the entire Euclidean space in which they live: this is called a *Voronoi partition*). Furthermore, for a fixed partition, the optimal cluster centers are given by $w_i = \frac{1}{|C_i|}\sum_{x \in C_i} x_i$, the mean of the cluster. This observation, in addition to a simple identity about variance, lends itself to the following interpretation of k-means:

Lemma 1 (Combinatorial Formulation) *The k-means problem can alternatively be formulated as finding the partition of points which minimizes the variance within each partition. In other words:*

$$(C_1^*, ..., C_k^*) = \arg\min_{\sqcup_i C_i = X} \sum_{i=1}^{k} \frac{1}{2|C_j|} \sum_{x,y \in C_j} ||x - y||^2 \qquad w_i^* = \frac{1}{|C_i|}\sum_{x \in C_i} x$$

Proof Note that this problem is a strict relaxation, as there are some partitions of the points which do not admit a representation as cluster centers. But it is not hard to see that such partitions will never be optimal.

For a fixed clustering, we have the optimal placement of each center is given by the mean, because the mean is the L2 minimizer. This is evident by a simple analysis of stationary points: $\frac{\partial}{\partial w}\sum_{i=1}^{n} ||x - w||^2 = 0 \implies w^* = \frac{1}{n}\sum_{i=1}^{n} x$, and the Hessian is positive-semidefinite so this is indeed a minimizer.

Hence, the minimizer of the original k-means is equivalent to the minimizer of:

$$\sum_{j=1}^{k} \sum_{x \in C_j} ||x - \mu_j||^2$$

From here, we make use of the following fact from elementary probability: let A, B be independent and identically distributed random variables taking values in $X = (x_1, ..., x_n)$

with equal probabilities. Applying definition of expectation and independence, we have:

$$\mathbb{E}((A-B)^2) = \mathbb{E}(A^2) + \mathbb{E}(B^2) - 2\mathbb{E}(A)\mathbb{E}(B) = 2\mathbb{E}(A^2) - 2\mathbb{E}(B)^2 = 2\mathrm{Var}(A)$$

Unpacking the meaning of the expectations for these discrete uniform random variables gives us the following identity:

$$\frac{1}{|X|^2}\sum_{x,y\in X}\|x-y\|^2 = \frac{2}{|X|}\sum_{x\in X}\|x-\mu_X\|^2$$

Substituting this identity into the inner summation completes the proof. ■

1.2 Lloyd's Method

This idea that the mean of the induced partition gives us a local solution motivates *Lloyd's method*, also known as the k-means algorithm or (less commonly) Voronoi iterations. The algorithm is as follows:

Algorithm 1 Lloyd's algorithm

Require: $X = \{x_1, ..., x_n\} \subset \mathbb{R}^d, k \in \mathbb{N}_{>1}$
  Initialize arbitrary $W = (w_1, ..., w_k)$.
  while W changes **do**
    Let $C_i = \{x \in X : \forall w \in W, ||x - w_i|| \leq ||x - w||\}$ (cluster induced by w_i).
    Update $w_i = \frac{1}{|C_i|}\sum_{x\in C_i} x$.
  end while
  Return W the set of cluster centers.

Note that there are two degenerate situations that can occur over the course of this algorithm.

- A center might have no points assigned to it (say, because it starts out much farther away than the rest of the points). In this case, the algorithm will produce fewer than k clusters.
- A point may be equally close to two centers, in which case its assignment to a cluster is made arbitrarily.

When providing lower bounds on the behavior of Lloyd's method (e.g. with respect to runtime or error) it is useful to distinguish if the lower bound relies on one of these cases.

1.3 Behavior of Lloyd's method

The good news about Lloyd's algorithm is that it monotonically decreases the k-means loss and it never visits the same clustering twice. It does, however, have undesirable worst-case behavior in terms of both approximation ratio and runtime.

Lemma 2 *Lloyd's algorithm has unbounded approximation ratio, even for* $d = 1$.

Proof Take points $X = (0, 1, N, N + M)$ and $k = 3$. For $N > M > 1$, the optimal placement of centers is $W^* = (1/2, N, N + M)$ achieving loss $1/2$. But if you initialize $(0, 1, N + \frac{M}{2})$, you achieve loss $\frac{M^2}{2}$ and Lloyd's algorithm terminates. Sending $M \to \infty$ with $N > M$, we have $L(W)/L(W^*) = M^2 \to \infty$. ■

Lemma 3 (worst-case runtime, (Vattani, 2009b)) *There exists $X = (x_i)_{i\in[n]} \subset \mathbb{R}^2$ for which Lloyd's algorithm takes $2^{\Omega(n)}$ steps before terminating, without ever falling into a degenerate configuration.*

It was however noted that k-means has more reasonable *smoothed runtime complexity* (Spielman and Teng, 2003), analogous to the simplex method for convex optimization.

Lemma 4 (smoothed runtime, (Arthur et al., 2011)) *Fix $X = (x_i)_{i\in[n]} \subset \mathbb{R}^d$. Let $X' = (x_i + \epsilon_i)_{i\in[n]}$ where $\epsilon_i \sim \mathcal{N}(0, \sigma)$ are independent noise variables. Then the expected runtime of k-means on X' is at most polynomial in n and $1/\epsilon$.*

The lower bounds in Lemmas 3 and 4 require fairly intensive constructions, which we omit for the sake of brevity.

2. Basic Algorithmic Results on k-means

In this section, we review the simple solution to k-means in one dimension, an elegant hardness proof for k-means in higher dimensions, and some of the methods that go into more involved approximation algorithms for k-means.

2.1 k-means dynamic programming solution for $d = 1$

In one-dimension, it very much pays to look at k-means as a combinatorial problem. The ordering of the real number line makes the k-means problem amenable to a simple dynamic programming based solution, as pointed out by ((Wang and Song, 2011)).

The intuition for the algorithm is as follows. Label the given points $x_1, ..., x_n$ in ascending order. Let $C_1, ..., C_k$ be the optimal clustering of these points: it is necessarily the case that these clusters are connected, in the sense that $i < j$ implies that for $x \in C_i$, $y \in C_j$ we have $x \leq y$. If $D[n, k]$ is the k-means cost of the optimal, and j is the smallest element in C_k, then:

$$D[n, k] = D[j - 1, k - 1] + d(x_j, ..., x_i)$$

where $d(X) = \sum_{x\in X} ||x - \mu_X||^2$. In other words, we have the contribution from the first $k - 1$ clusters and the contribution from the last one.

This leads us to the more general recurrence relation: for $D[i, m]$ the optimal k-means cost for the first i points using k cluster centers, we search for the j that best accomplishes the split for the last cluster.

$$D[i, m] = \min_{m\leq j\leq i} \{D[j - 1, m - 1] + d(x_j, , , x_i)\}$$

The base case is $D[i, m] = 0$ for m or i zero (since the summations in the k-means loss would be null). The desired answer of course is $D[n, k]$, which we build up to; it is easy to get the corresponding clustering by making an auxiliary matrix which records the smallest entry in cluster m (same formula as $D[i, m]$ but with an arg min rather than a min). The naive implementation of this algorithm is $O(n^3k)$ since the trellis is $(n + 1) \times (k + 1)$ and each entry takes $O(n^2)$ times to compute: $O(n)$ rounds of computing $d(\cdot)$, which is naively $O(n)$. The computation for $d(\cdot)$ could be made constant time however by using another recursion. This would reduce the overall runtime to $O(n^2k)$.

2.2 Hardness

As seen above, k-means is easy to solve in one dimension. It is however NP-hard for $k \geq 2$, as shown by (Dasgupta, 2008)'s reduction from NAE-3SAT*. It was later shown that it is also hard for $d \geq 2$: (Vattani, 2009a) showed via a reduction from Exact Cover by 3-Sets, while (Mahajan et al., 2012) showed via a reduction from Planar 3SAT. (Awasthi et al., 2015) showed it was NP-hard via a reduction from vertex cover on triangle-free graphs.

In this section we briefly review the simple NP-hardness reduction by (Dasgupta, 2008).

Definition 5 *A 3CNF Boolean formula $\phi : \{0, 1\}^n \mapsto \{0, 1\}$ in 3CNF with clauses $C_1, ..., C_m$ is said to be in* NAE-3SAT* *if:*

1. *There exists an assignment $x = (x_1, ..., x_n)$ such that for each clause C_i, only one or two of its literals are satisfied.*
2. *Each clause has exactly three literals.*
3. *Given any pair of variables x_i, x_j, they appear together in a clause at most once in the forms (x_i, x_j) or $(\overline{x_i}, \overline{x_j})$ and at most once in the forms $(\overline{x_i}, x_j)$ or $(x_i, \overline{x_j})$*

The nice thing about this language, for our purposes, is that a satisfiable instance *minimizes* the number of literals that both (1) appear in the same clause and (2) have the same sign. We use this to our advantage in the reduction: the optimal clustering will always "save" on costs if it can place same clause literals in opposite clusters.

Definition 6 *The **generalized 2-means problem** is as follows: given $L \in \mathbb{R}$ and $D = D_{ij}$ a metric on n elements , return TRUE if and only if there exists a partition $C_1 \sqcup C_2 = [n]$ such that:*

$$\sum_{j=1}^{2} \frac{1}{2|C_j|} \sum_{i,i' \in C_j} D_{ii'} \leq L$$

The reduction proceeds as follows. Let $0 < \delta < \Delta < 1$ such that $4\delta m \leq \Delta \leq 1 - 2\delta n$, say $\delta = \frac{1}{5m+2n}$ and $\Delta = 5\delta m$. The lower bound is to penalize placing opposite sign literals in the same cluster and make the reduction work; the upper bound is to ensure embeddability of the resulting matrix.

- Given ϕ, construct D on $2n$ variables, call then $x_1, ..., x_n, \overline{x}_1, ..., \overline{x}_n$ such that:

  $D(\alpha, \alpha) = 0$.

$D(\alpha, \overline{\alpha}) = 1 + \Delta$.

$D(\alpha, \beta) = 1 + \delta$ if α, β or $\overline{\alpha}, \overline{\beta}$ appear in the same clause (denote $\alpha \sim \beta$).

$D(\alpha, \beta) = 1$ otherwise.

- Let $L = n - 1 + \frac{2\delta m}{n}$. This will be our loss threshold.

To justify this reduction, we need to show that accept instances map to accept instances and reject instances map to reject instances.

- (YES to YES) If ϕ is NAE-3SAT*, then if you partition by positive and negative literals, the k-means cost is given by:

$$\text{COST} = \frac{1}{2n} \sum_{i,i' \in C_1} D_{ii'} + \frac{1}{2n} \sum_{i,i' \in C_2} D_{ii'}$$

Note that in the assignment, each clause generates 6 same-clause relations: for instance, $(x \vee \overline{y} \vee z)$ generates $x \sim \overline{y}$ as well as $\overline{x} \sim y$, and so on. So it suffices to just look at summation over one cluster, due to the symmetry of things. From there, we note that there are $n^2 - n = 2\binom{n}{2}$ interactions within the cluster, of which $2m$ are interactions between same-clause literals. Hence the loss becomes:

$$\text{COST} = \frac{1}{n} \sum_{i,i' \in C_1} D_{ii'} = \frac{2}{n}\left(\binom{n}{2} + m\delta\right) = n - 1 + \frac{2\delta}{m} = L$$

- (NO to NO) If ϕ is NOT NAE-3SAT*, we can establish the following in sequence:

(1) Placing a variable and its negation in the same cluster attains loss $> L$.

Proof Consider a clustering with n and n' on each side, where a variable and its negation appear in the same cluster. Then:

$$\text{COST} \geq \frac{1}{n'}(\binom{n'}{2} + \Delta) + \frac{1}{2n - n'}\binom{2n - n'}{2} = n - 1 + \frac{\Delta}{n'} \geq n - 1 + \frac{\Delta}{2n}$$

Since $\Delta > 4\delta m$, then the cost is $> n - 1 + \frac{2\delta}{m} = L$. ■

(2) Any other clustering corresponds to an assignment of variables. By definition of NAE-3SAT*, any assignment of variables will yield more than $2m$ same-clause same-sign (and hence same-cluster) interactions, which would yield a cost $> L$ as well.

((Dasgupta, 2008)) furthermore shows that the matrix $D = D(\phi)$ is Euclidean embeddable in dimension at most $2n$, by using the fact that an $N \times N$ symmetric matrix is Euclidean embeddable if and only if $u^T D u \leq 0$ for $u \in \mathbb{R}^N$ such that $\sum_{i=1}^N u_i = 0$

2.3 State of Approximation Algorithms

In light of the hardness of approximation result by (Awasthi et al., 2015), the best we can hope for is efficient constant-factor approximation or a poly-time approximation scheme for fixed k or d.

The first example of the latter was (Matoušek, 2000)'s $O(n(\log n)^k \epsilon^{-2k^2 d})$ algorithm. The algorithm works by cutting down the search space and then performing a brute force search. The first reduction is to only consider centroids from an ϵ-approximate centroid set which would contain a $(1+\epsilon)$-optimal solution. He then shows that every such near-optimal solution is a *well-spread k-tuple*, of which there are at most $O(n\epsilon^{-k^2 d})$ for an n-point set. The algorithm proceeds by producing all of these k-tuples and choosing the one with the lowest cost.

The first constant-factor approximation was given by (Kanungo et al., 2002)'s local swap method. This algorithm proceeds also by using Matousek's ϵ-approximate centroid set construction, which can be done in $O(n \log n + n\epsilon^{-d} \log(1/\epsilon))$, suffering from the curse of dimensionality but independent of k. (Kanungo et al., 2002) proposes a single-swap heuristic, which works by selecting a random set of centers and improves the solution by removing one center at a time and replacing. This achieves $(25+\epsilon)$ approximation; a slightly more involved multiple-swap heuristic achieves a $(9+\epsilon)$ approximation.

3. Gradients and k-means

In this section we take a slight detour and review basic guarantees for gradient descent and Newton's method in the context of convex optimization.

3.1 Gradient Descent

The classical guarantee for gradient-based optimization is an $O(1/\epsilon^2)$ approximation scheme for convex, Lipschitz functions (this was shown to be tight by Nesterov). We provide a brief sketch of this result, following (Bansal and Gupta, 2019)'s potential-based approach.

Definition 7 *Given $f : \mathbb{R}^n \mapsto \mathbb{R}$ a differentiable function, a starting point $x_0 \in \mathbb{R}^n$, a step size $\eta > 0$, and an integer T, we define* ***gradient descent*** *as follows:*

- *For $i = 1, ..., T$:*

  Let $x_i = x_{i-1} - \eta(\nabla f)$

- *Return $\hat{x} = \frac{1}{T}\sum_{i=1}^{T} x_i$*

This vanilla version of gradient descent outputs the time-average of its estimates rather than x_T itself. This makes it substantially easier to analyze, as we will see with the proof of Theorem 1. Note that there are similar results for non-time-averaged methods, see (Nesterov et al., 2018).

Theorem 8 *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a convex, differentiable, and G-Lipschitz. Let x^* be a minimizer of f. For all $\epsilon > 0$, gradient descent run for $T = \frac{G^2\|x_0 - x^*\|}{\epsilon^2}$ steps with step size $\eta = \frac{\|x_0 - x^*\|}{G\sqrt{T}}$ returns $\hat{x}$ such that:*

$$f(\hat{x}) \leq f(x^*) + \epsilon$$

Proof It suffices to show the following inequality. With this, the result follows easily from convexity of f and the choices of η and T.

$$\sum_{i=1}^{T} f(x_t) \leq Tf(x^*) + \frac{1}{2}\eta G^2 + \frac{1}{2\eta T}||x_0 - x^*||^2$$

Let $\Phi_t = \frac{||x_t - x^*||^2}{2\eta}$. The idea is to look at successive steps of the potential function.

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \frac{1}{2\eta}(||x_{t+1} - x^*||^2 - ||x_t - x^*||^2) \\
&= \frac{1}{2\eta}(2\langle x_{t+1} - x_t, x_t - x^*\rangle + ||x_{t+1} - x_t||^2) \\
\text{(algorithm)} \quad &= \frac{1}{2\eta}(2\langle -\eta\nabla f(x_t), x_t - x^*\rangle + ||\eta\nabla f(x_t)||^2) \\
\text{(G-Lipschitz)} \quad &\leq \frac{1}{2\eta}(2\langle \eta\nabla f(x_t), x^* - x_t\rangle + G^2)
\end{aligned}$$

Now consider adding $f(x_t)$ to both sides.

$$\begin{aligned}
f(x_t) + (\Phi_{t+1} - \Phi_t) &\leq f(x_t) + \langle \nabla f(x_t), x^* - x_t\rangle + \frac{G^2}{2\eta} \\
\text{(convex)} \quad &\leq f(x^*) + \frac{G^2}{2\eta}
\end{aligned}$$

Sum both sides over $t \in [T]$, with fortuitous telescoping on the LHS.

$$\sum_{t=1}^{T} f(x_t) + \Phi_T - \Phi_1 \leq Tf(x^*) + \frac{G^2 T}{2\eta}$$

Since $\Phi_T > 0$ we can drop it from LHS. Rearranging yields our desired inequality. ■

The Φ used in the proof is referred to by (Bansal and Gupta, 2019) as a **potential function**. This is a relatively common construction for some algorithmic analysis, but it (Bansal and Gupta, 2019) seemed to be the first to suggest its use in analyzing gradient descent.

3.2 Newton's Method

In contrast to gradient descent, which is a first-order optimization method, Newton's method is a second-order optimization method. What this means is that the optimization considers the best-fitting *paraboloid* at a point and then finds the projection which minimizes said paraboloid.

The elementary setting for learning about Newton's method is in the context of finding roots of a continuously differentiable function. Say we have a function $f : \mathbb{R} \to \mathbb{R}$ and we want to find x^* such that $f(x^*) = 0$. The idea is to take the second order Taylor expansion and set it to zero.

$$\underbrace{f(x+t)}_{f(x^*)=0} = f(x) + tf'(x) + O(t^2)$$

Setting the LHS to zero and ignoring the higher-order terms, we have $t = f(x)/f'(x)$. This motivates the following update scheme:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

In the absence of higher-order terms—i.e. the case where we are minimizing a quadratic function—Newton's method converges in a single step. In general, for the single-variable case, we have fast convergence for strongly convex functions (i.e. functions which are bounded below by some parabola).

Minimization of some $f : \mathbb{R}^d \to \mathbb{R}$ boils down to finding the roots of $\nabla f : \mathbb{R}^d \to \mathbb{R}^d$. We can apply one-dimensional Taylor expansion to each $(\nabla f)_i : \mathbb{R}^d \to \mathbb{R}$ and after putting things together, arrive at the following system of equations:

$$\nabla f(x) + [\nabla^2 f(x)]t = \underbrace{\nabla f(x+t)}_{\text{set to zero}}$$

where we interpret this as a vector equation and $\nabla^2 f(x)$ is the Hessian matrix, i.e. $[\nabla^2 f(x)]_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$. Solving for t and then using that as our local update, we have:

$$x_{n+1} = x_n - [\nabla^2 f(x_n)]^{-1} \nabla f(x_n)$$

Although there are results for fast convergence of Newton's method on well-behaved single-variable functions, there are examples of it diverging on vector-valued functions that are strongly convex (Boyd and Vandenberghe, 2004). In order to guarantee convergence in the multivariate function minimization setting, one must use a learning rate in the update, i.e. $x_{n+1} = x_n - \eta[\nabla^2 f(x_n)]^{-1} \nabla f(x_n)$, where a small enough value of η (for stability purposes) is found using a *line search*. See (Boyd and Vandenberghe, 2004).

3.3 Lloyd-Newton Correspondence

Consider the following formulation of the k-means loss, equivalent for the purposes of optimizing the cluster centers (the 1/2 factor makes the analysis nicer).

$$L(w_1, ..., w_k) = \sum_{i=1}^{n} \frac{1}{2} \min_{j \in [k]} ||x_i - w_j||^2$$

Unfortunately but importantly, due to the minimum, this function is not differentiable everywhere. It is however differentiable outside of a set of measure zero. If you differentiate with respect to such w_i, the derivative is indeed well-defined and given by:

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^{n} (w_j - x_i) \cdot \mathbf{1}\{x_i \in C_j\} = |C_j| w_j - \sum_{i:\ x_i \in C_j} x_i$$

Note that mixed second derivatives are zero. However, taking the second derivative with respect to w_j yields:

$$\frac{\partial^2 L}{\partial w_j^2} = -|C_j| I$$

Let $N_i = |C_i|$. Then the Hessian is given by:

$$\nabla^2 f = \text{diag}(N_1 I_{N_1}, ..., N_k I_{N_k}) \tag{1}$$

Note that the Hessian written above is positive semi-definite, but this does not guarantee the convexity of the objective function because the Hessian above does not exist everywhere.

Nonetheless, the Hessian above exists almost everywhere (with respect to Lebesgue measure on the domain of possible placements of cluster centers), which makes it reasonable to conceive of a Newton update scheme.

$$w_{k+1} = w_k - \Big[\nabla^2 f(w_k)\Big]^{-1}\Big[\nabla f(w_k)\Big] \tag{2}$$

$$= w_j - \frac{1}{|C_j|}\Big(|C_j| w_j - \sum_{i:\ x_i \in C_j} x_i\Big) \tag{3}$$

$$= \frac{1}{|C_j|}\sum_{i:\ x_i \in C_j} x_i \tag{4}$$

This update scheme is precisely Lloyd's method.

3.4 Online Lloyd's Algorithm and SGD

With large datasets, it is often desirable to make iterative updates based on small subsets of the data rather than the entire dataset at once. This is the idea behind mini-batch stochastic gradient descent, for instance, which approximates the gradient using a randomly chosen mini-batch of data. It is also the idea behind (Bottou and Bengio, 1994)'s online Lloyd's algorithm and (Sculley, 2010)'s mini-batch Lloyd's algorithm. (Tang and Monteleoni, 2017) showed $O(1/t)$ convergence for these forms of so-called *stochastic k-means* (i.e. after t rounds of the algorithm, additive error with a local minimum is $O(1/t)$). The proof effectively interprets these methods as stochastic gradient descent on non-convex functions.

Similarly, (So et al., 2022) provides a convergence guarantee for the *online Lloyd's algorithm* in a streaming setting, by interpreting it as stochastic gradient descent and adapting standard techniques for SGD analysis (e.g. supermartingale convergence).

Algorithm 2 Online Lloyd's algorithm

Require: p supported in $\mathbb{R}^d$. $k \in \mathbb{N}_{>1}$.
Require: h_i learning rate.
  Initialize $W = (W_1, ..., W_k)$ arbitrarily in $\text{supp}(p)$.
  for $n = 1, 2, ...$ **do**
    Sample $X \sim p$.
    Identify closest center $i \leftarrow \arg\min_{j \in [k]} \|W_j - X\|$
    Update center $W_i \leftarrow W_i - H_i \cdot (W_i - X)$.
  end for
  Return W the set of cluster centers.

In the version of the algorithm analyzed by (So et al., 2022) and (Tang and Monteleoni, 2017), H_i is a random sequence which is made to satisfy certain convergence criteria. In the

original algorithm proposed by (Bottou and Bengio, 1994), it was actually deterministic, i.e. $H_i = \frac{1}{N_i}$ where N_i is the number of times that W_i was updates.

The streaming setting of the problem proceeds as follows: fix some distribution p supported on $\mathbb{R}^d$, and a sequence of points $\{X_i\}_{i\in\mathbb{N}}$ sampled i.i.d. from p. The cost function in this smoothed setting is then:

$$f(W_1, ..., W_k) = \int \min_{j\in[k]} \|W_j - x\|^2 p(x)dx$$

The main theorem (Theorem 5.1) from (So et al., 2022) states that, so long as f has no degenerate stationary points (i.e. two centers overlapping), then online Lloyd's with a suitable choice of stochastic learning rate H_i will asymptotically converge almost surely to a local minimum, i.e.

$$\limsup_{n\to\infty} \inf_{w\in\{\nabla f=0\}} \|W^{(n)} - w\| = 0$$

where $W^{(n)}$ are the k centers output by stochastic k-means after n iterations.

4. Smart Initialization and kmeans++

With the understanding that Lloyd's algorithm has arbitrarily bad worst-case behavior, it is natural to consider how it might perform in expectation, where the randomness is over the initialization. A natural way to do this initialization is using a *farthest-first heuristic*:

Algorithm 3 kmeans++

Require: $X = \{x_1, ..., x_n\} \subset \mathbb{R}^d, k \in \mathbb{N}_{>1}$

Pick w_1 uniformly at random from $x_1, ..., x_n$. Initialize $W = \{w_1\}$.

for $i \in \{2, ..., n\}$ **do**

Pick $x \in \mathcal{X}$ with probability:

$$p_x = \frac{\min_{w\in W} ||x - w||^2}{\sum_{z\in\mathcal{X}} \min_{w\in W} ||x - w||^2} = \frac{D^2(x; W)}{\sum_{z\in\mathcal{X}} D^2(z; W)}$$

Update $W = W \cup \{x\}$ and $X = X \setminus \{x\}$.

end for

Return W the set of cluster centers.

To analyze this algorithm we establish some notation.

- Given a set of cluster centers C and $A \subset X$, let

$$\phi_C(A) = \sum_{a\in A} \min_{c\in C} ||x - c||^2$$

- Let C_{OPT} denote the set of optimal cluster centers and $\phi_{OPT} = \phi_{C_{OPT}}$.

  Let $\{A_1^*, ..., A_k^*\}$ denote the partition generated by C_{OPT}.

Lemma 9 *For S a point set in $\mathbb{R}^d$ and $z \in \mathbb{R}^d$, we have:*

$$\sum_{x \in S} ||x - z|| = \sum_{x \in S} \|x - \mu_S\|^2 + |S| \cdot \|z - \mu_S\|^2$$

Lemma 10 *Let A_i^* be an arbitrary cluster generated by C_{OPT}. Let c be chosen uniformly at random from C_i^*. Then:*

$$\mathbb{E}_c[\phi_{\{c\}}(A_i^*)] = 2\phi_{OPT}(A_i^*)$$

Proof Note that the expectation is taken over the choice of u in A_i^*. Each one of these centers is chosen with equal probability $1/|C_i^*|$.

$$\begin{aligned}
\sum_{c \in A_i^*} \mathbb{P}(\text{we pick } c) \cdot \phi_{\{c\}}(A_i^*) &= \sum_{c \in A_i^*} \frac{1}{|A_i^*|} \cdot \phi_{\{c\}}(A_i^*) \\
\text{(definition of } \phi) \quad &= \sum_{c \in A_i^*} \frac{1}{|A_i^*|} \sum_{v \in A_i^*} ||c - v||^2 \\
&= \frac{1}{|A_i^*|} \sum_{c,v \in A_i^*} ||c - v||^2 \\
\text{(variance identity)} \quad &= 2 \sum_{v \in A_i^*} ||v - \mu_{A_i}^*||^2 \\
&= 2 \cdot \phi_{OPT}(A_i^*)
\end{aligned}$$

In the last step, we use the fact that the optimal clustering of a given point set is given precisely by the mean (again, by the L2 minimizer argument). ■

Lemma 11 *Let A_i^* be an arbitrary cluster from C_{OPT}. Let C be an arbitrary clustering. If we pick $c \in A_i^*$ with probability according to farthest-first weighting, then:*

$$\mathbb{E}_c[\phi_{C \cup \{c\}}(A_i^*)] \leq 8\phi_{OPT}(A_i^*)$$

Proof The proof proceeds first by writing out the definition of expectation and inserting our farthest-first probabilities.

$$\begin{aligned}
\mathbb{E}_c[\phi_{C \cup \{c\}}(A_i^*)] &= \sum_{c \in A_i^*} \mathbb{P}(\text{we pick } c) \cdot \phi_{C \cup \{c\}}(A_i^*) \\
&= \sum_{c \in A_i^*} \frac{D(c;C)^2}{\sum_{a \in A_i^*} D(a;C)^2} \cdot \sum_{a \in A_i^*} D(a; C \cup \{c\})^2
\end{aligned}$$

Observe that $D(a; C \cup \{c\})^2 = \min_{b \in C \cup \{c\}} \|b - a\|^2 = \min(D(a;C), \|a - c\|^2)^2$, by simply properties of the minimum. So we have the following.

$$\mathbb{E}_c[\phi_{C \cup \{c\}}(A_i^*)] = \sum_{c \in A_i^*} \frac{\boxed{D^2(c;C)}}{\sum_{a \in A_i^*} D^2(a;C)} \cdot \sum_{a \in A_i^*} \min(D(a;C), \|a - c\|^2)^2$$

Let us analyze the boxed term in more detail, We would like to decompose it effectively into a term containing $D(a;C)^2$ and a term containing $||a-c||^2$. For the sake of simplicity we suppress the clustering in $D(\cdot)$.

By triangle inequality, we have:

$$\min_{b\in C} ||c-b|| \leq \min_{b\in C} ||a-b|| + \|a-c\|$$

$$\implies D(c) \leq D(a) + \|a-c\|$$

By applying power-mean inequality $\frac{1}{2}(a+b)^2 \leq a^2+b^2$ to the RHS, we have:

$$D^2(c) \leq 2D^2(a) + 2||a-c||^2$$

Summing over A, we have then:

$$|A| \cdot D^2(c) \leq 2\sum_{a\in A} D^2(a) + 2\sum_{a\in A} ||a-c||^2$$

Plugging this back into our inequality readily gives us our approximation.

$$\mathbb{E}_c[\phi_{C\cup\{c\}}(A_i^*)] \leq \frac{2}{|A|}\sum_{c\in A}\frac{\sum_{a\in A} D^2(a)}{\sum_{a\in A} D^2(a)}\sum_{a\in A}\min(D(a), ||a-c||^2) \tag{5}$$

$$+ \frac{2}{|A|}\sum_{c\in A}\frac{\sum_{a\in A} ||a-c||^2}{\sum_{a\in A} D(a)^2}\sum_{a\in A}\min(D(a), ||a-c||^2) \tag{6}$$

$$\leq \frac{4}{|A|}\sum_{c,a\in A}\|a-c\|^2 \tag{7}$$

$$= 8\phi_{OPT}(A) \tag{8}$$

Hence, with farthest-first weighting, our performance on the new cluster is at most 8 times the performance of optimal. ■

The bulk of the work comes into the following lemma. The visual to have in mind is that there is some fixed Voronoi partition (of the points as well as the ambient space) generated by the optimal set of cluster centers. If we have some clustering that covers some portion of this partition, and we add points to that clusterng according to farthest-first weighting, we essentially want to show that the new points are for the most part going to hit uncovered clusters.

Lemma 12 *Let C be an arbitrary clustering. Fix $u > 0$ arbitrary clusters from C_{OPT}. Let X_u denote the set of points in these clusters. Let $X_c = X \setminus X_u$. Suppose we add $t \leq u$ centers to C according to farthest-first weighting. Let C' denote the resulting clustering. Then:*

$$\mathbb{E}(\phi_{C'}(X)) \leq \Big(\phi_C(X_c) + 8\phi_{OPT}(X_u)\Big)(1+H_t) + \frac{u-t}{u}\phi_C(X_u)$$

Proof We prove this by induction on t and u.

(Base Case 1): For $u > t = 0$, we did not add anything to our clustering, so $C' = C$, and note that $1 + H_t = \frac{u-t}{u} = 1$. So the desired inequality becomes:

$$\phi_C(X) \leq \phi_C(X_c) + \phi_C(X_u) + 8\phi_{OPT}(X_u)$$

which is trivial since $\phi_C(X_c) + \phi_C(X_u) = \phi_C(X)$.

(Base Case 2): Now suppose $t = u = 1$. So X_u is a single cluster, and X_c represents the points contained in the rest of the clusters. Note that the probability we pick a point from the cluster according to probabilistic farthest first weighting is as follows (no minimum to deal with because our clustering is a single cluster):

$$\sum_{t \in X_u} \frac{D(t;C)^2}{\sum_{x \in X} D(x;C)^2} = \frac{\sum_{t \in X_u} \|x - t\|^2}{\sum_{x \in X} \|x - u\|^2} = \frac{\phi_C(X_u)}{\phi_C(A)}$$

Likewise with X_u. So by law of total expectation, we have:

$$\mathbb{E}(\phi_{C'}(X)) = \mathbb{P}(c \in X_u) \cdot \mathbb{E}_c(\phi_{C \cup \{c\}}(X) \mid c \in X_u) + \mathbb{P}(c \in X_c) \cdot \mathbb{E}_c(\phi_{C \cup \{c\}}(X) \mid c \in X_c)$$

Note that the latter expectation can only be lower than $\phi_C(X)$, since adding more clusters always reduces k-means loss. For the expectation on the left, we apply Lemma 7:

$$\begin{aligned}\mathbb{E}(\phi_{C'}(X)) &\leq \frac{\phi_C(X_u)}{\phi_C(X)}\Big(\phi_C(X_c) + 8\phi_{OPT}(X_u)\Big) + \frac{\phi_C(X_c)}{\phi_C(X)} \cdot \phi_C(X) \\ &\leq 2\phi_C(X_c) + 8\phi_{OPT}(X_u)\end{aligned}$$

(Inductive Step): Let $a_1, ..., a_t$ be random cluster centers chosen by the algorithm. We condition on whether a_0 came from X_u or from X_c.

- If the first center was chosen from X_c, then the contribution is at most

$$\begin{aligned}&\leq \mathbb{P}(a_1 \in X_u) \cdot \mathbb{E}(\phi_{C \cup \{a_2,...,a_t\}}(X_u) \mid a \in X_u) \\ &\leq \mathbb{P}(a_1 \in X_u) \cdot \mathbb{E}(\phi_{C \cup \{a_2,...,a_t\}}(X_u)) \\ \textbf{(IH } (u, t-1)\textbf{)} \quad &\leq \boxed{\frac{\phi_C(X_u)}{\phi_C(X)}\Big((\phi_C(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1}) + \frac{u - (t-1)}{u} \cdot \phi_C(X_u)\Big)}\end{aligned}$$

- If the first center was chosen from X_u, we find it advantageous to split this into cases based on the u clusters contained in X_u.

  Suppose we chose our first center from some uncovered cluster $A \subset X_u$. Let p_a be the probability we choose $a \in A$ as a center given the center is somewhere in A. Then we can upper bound the contribution by:

$$\begin{aligned}&\leq \frac{\phi_C(A)}{\phi_C(X)} \sum_{a \in A} p_a \cdot \mathbb{E}(\phi_{C \cup \{a_2,...,a_t\}}(X_u \setminus A)) \\ \textbf{(IH } (u-1, t-1)\textbf{)} \quad &\leq \frac{\phi_C(A)}{\phi_C(X)} \sum_{a \in A} p_a\Big(\phi_C(X_c \cup \{a\}) + 8\phi_{OPT}(X_i \setminus A)(1 + H_{t-1}) \\ &\qquad + \frac{u-t}{u-1}(\phi_C(X_u \setminus A))\Big)\end{aligned}$$

Note that $\phi(X \setminus A) = \phi(X) - \phi(A)$. Also note that $\sum_{a \in A} p_a \phi_{\{a\}} \leq 8\phi_{OPT}(A)$

$$\leq \frac{\phi_C(A)}{\phi_C(X)}\Big((\phi_C(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1})\Big) \tag{9}$$

$$+ \frac{\phi_C(A)}{\phi_C(X)}\Big(\frac{u-t}{u-1}(\phi(X_u) - \phi(A))\Big) \tag{10}$$

Next we sum this over $A \subset X_u$ to obtain the full contribution. Note that for the first term, all that happens is the coefficient becomes $\phi_C(X_u)/\phi_C(X)$. The second term demands a more careful treatment. Observe that, by power-mean inequality:

$$\frac{1}{u}\phi_C(X_u)^2 = \frac{1}{u}\Big(\sum_{A \subset X_u} \phi(A)\Big)^2 \leq \sum_{A \subset X_u} \phi(A)^2$$

Hence, we can apply the following bound:

$$\begin{aligned}\sum_{A \subset X_u} \phi_C(A)\Big(\phi_C(X_i) - \phi_C(A)\Big) &\leq \phi_C(X_i)^2 - \sum_{A \subset X_u} \phi_C(A)^2 \\ &\leq \phi_C(X_i)^2 - \frac{1}{u}\phi_C(X_i)^2\end{aligned}$$

Putting this together, we have the following bound on the second case:

$$\begin{aligned}&\leq \frac{\phi_C(X_u)}{\phi_C(X)}\Big((\phi_C(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1})\Big) \\ &+ \frac{1}{\phi_C(X)} \cdot \frac{u-t}{u-1} \cdot \frac{u-1}{u} \cdot \phi(X_u)^2 \\ &= \boxed{\frac{\phi_C(X_u)}{\phi_C(X)}\Big((\phi_C(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1}) + \frac{u-t}{u}\phi(X_u)\Big)}\end{aligned}$$

Combining the bounds (boxed), and observing $\frac{\phi_C(X_c)+\phi_C(X_u)}{\phi_C(X)} = 1$, we have:

$$\begin{aligned}\mathbb{E}(\phi_{C \cup \{a_1, \ldots a_t\}}) &\leq (\phi_C(X_c) + 8\phi_{OPT}(X_u))(1 + H_{t-1}) + \frac{\phi_C(X_c) \cdot \phi_C(X_u)}{\phi_C(X) \cdot u} + \frac{u-t}{u}\phi(X_u) \\ &\leq (\phi_C(X_c) + 8\phi_{OPT}(X_u))\Big(1 + H_{t-1} + \frac{1}{u}\Big) + \frac{u-t}{u}\phi(X_u) \\ &\leq (\phi_C(X_c) + 8\phi_{OPT}(X_u))\Big(1 + H_t\Big) + \frac{u-t}{u}\phi(X_u)\end{aligned}$$

Note in the last line we use $1/u \leq 1/t$, and in the line before that, we use relatively crude bounds.

$$\frac{\phi_C(X_c) \cdot \phi_C(X_u)}{\phi_C(X) \cdot u} \leq \frac{\phi_C(X_c)}{u} \leq (\phi_C(X_c) + 8\phi_{OPT}(X_u)) \cdot \frac{1}{u}$$

This completes the induction proof. ■

Theorem 13 *Let K be a set of cluster centers generated by k-means++ on a dataset $X \subset \mathbb{R}^d$. Then we have:*

$$\mathbb{E}[\phi_K(X)] \leq 8(\log(k) + 2)\phi_{OPT}(X)$$

Proof Let $K = (c_1, ..., c_k)$ be the set of k cluster centers generated by the algorithm. Recall that the first cluster center c_1 is chosen uniformly at random and say lands on cluster C_1 in the optimal clustering.

$$\mathbb{E}[\phi_{\{c\}}(C_1)] = 2\phi_{OPT}(C_1)$$

Now we apply Lemma 8 with $C = C_1, t = u = k - 1$, we have:

$$\begin{aligned}
\mathbb{E}(\phi_K(X)) &\leq \Big(\phi_K(C_1) + 8\phi_{OPT}(X \setminus C_1)\Big)(1 + H_{k-1}) \\
&\leq \Big(\phi_K(C_1) - 8\phi_{OPT}(C_1) + 8\phi_{OPT}(X)\Big)(2 + \ln(k)) \\
&\leq 8(2 + \ln(k))\phi_{OPT}(X)
\end{aligned}$$

Note in the last step we use that $\phi_K(C_1) \leq 8\phi_{OPT}(C_1)$ as per lemma 7. ■

5. Discussion

The convergence guarantees for local iterative update methods has gained significant interest in the theory community with the rise of deep neural networks and optimization via stochastic gradient descent (Arora et al., 2018; Andoni et al., 2014; Jacot et al., 2018). The k-means++ is a triumphant example of analyzing how random initialization (with the right kind of weighting) enables us to establish a guarantee for an otherwise poorly behaved gradient-based algorithm, namely Lloyd's method.

Acknowledgments

The author would like to thank Mihalis Yannakakis and Rashida Hakim for their feedback on this review, which was submitted as a final project for COMS6232 Approximation Algorithms at Columbia.

References

Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning polynomials with neural networks. In *International conference on machine learning*, pages 1908–1916. PMLR, 2014.

Sanjeev Arora, Wei Hu, and Pravesh K Kothari. An analysis of the t-sne algorithm for data visualization. In *Conference on learning theory*, pages 1455–1462. PMLR, 2018.

David Arthur and Sergei Vassilvitskii. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *Journal of the ACM (JACM)*, 58(5):1–31, 2011.

Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. *arXiv preprint arXiv:1502.03316*, 2015.

Nikhil Bansal and Anupam Gupta. Potential-function proofs for gradient methods. *Theory of Computing*, 15(1):1–32, 2019.

Leon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. *Advances in neural information processing systems*, 7, 1994.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Sanjoy Dasgupta. The hardness of k-means clustering. 2008.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18, 2002.

Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. *Theoretical Computer Science*, 442:13–21, 2012.

Jiří Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178, 2010.

Geelon So, Gaurav Mahajan, and Sanjoy Dasgupta. Convergence of online k-means. In *International Conference on Artificial Intelligence and Statistics*, pages 8534–8569. PMLR, 2022.

Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of termination of linear programming algorithms. *Mathematical Programming*, 97:375–404, 2003.

Cheng Tang and Claire Monteleoni. Convergence rate of stochastic k-means. In *Artificial Intelligence and Statistics*, pages 1495–1503. PMLR, 2017.

Andrea Vattani. The hardness of k-means clustering in the plane. *Manuscript, accessible at http://cseweb. ucsd. edu/avattani/papers/kmeans_hardness. pdf*, 617, 2009a.

Andrea Vattani. K-means requires exponentially many iterations even in the plane. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 324–332, 2009b.

Haizhou Wang and Mingzhou Song. Ckmeans. 1d. dp: optimal k-means clustering in one dimension by dynamic programming. *The R journal*, 3(2):29, 2011.