Noname manuscript No.
(will be inserted by the editor)

Stochastic Backward Euler: An Implicit Gradient Descent Algorithm for k-means Clustering

Penghang Yin · Minh Pham · Adam Oberman · Stanley Osher

Received: date / Accepted: date

Abstract In this paper, we propose an implicit gradient descent algorithm for the classic k-means problem. The implicit gradient step or backward Euler is solved via stochastic fixed-point iteration, in which we randomly sample a mini-batch gradient in every iteration. It is the average of the fixed-point trajectory that is carried over to the next gradient step. We draw connections between the proposed stochastic backward Euler and the recent entropy stochastic gradient descent (Entropy-SGD) for improving the training of deep neural networks. Numerical experiments on various synthetic and real datasets show that the proposed algorithm provides better clustering results compared to k-means algorithms in the sense that it decreased the objective function (the cluster) and is much more robust to initialization.

Keywords k-means · backward Euler · implicit gradient descent · fixed-point iteration · mini-batch gradient

1 Introduction

The k-means method appeared in vector quantization in signal processing, and had now become popular for clustering analysis in data mining. In the seminal paper [13], Lloyd proposed a two-step alternating algorithm that quickly converges to a local minimum. Lloyd's algorithm is also known as an instance of the more general Expectation-Maximization (EM) algorithm applied to Gaussian mixtures. In [5], Bottou and Bengio cast Lloyd's algorithm as Newton's method, which explains its fast convergence.

Penghang Yin · Minh Pham · Stanley Osher
Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90095.
E-mail: (yph, minhrose, sjo)@math.ucla.edu

Adam Oberman
Department of Mathematics and Statistics, McGill University, Montreal, Canada.
E-mail: adam.oberman@mcgill.ca

Aiming to speed up Lloyd's algorithm, Elkan [9] proposed to keep track of the distances between the computed centroids and data points, and then cleverly leverage the triangle inequality to eliminate unnecessary computations of the distances. Similar techniques can be found in [8]. It is worth noting that these algorithms do not improve the clustering quality of Lloyd's algorithm, but only achieve acceleration. However, there are well known examples where poor initialization can lead to low quality local minima for Lloyd's algorithm. Random initialization has been used to avoid these low quality fixed points. The article [2] introduced a smart initialization scheme such that the initial centroids are well-separated, which gives more robust clustering than random initialization.

We are motivated by problems with very large data sets, where the cost of a single iteration of Lloyd's algorithm can be expensive. Mini-batch [17,18] was later introduced to adapt k-means for large scale data with high dimensions. The centroids are updated using a randomly selected mini-batch rather than all of the data. Mini-batch (stochastic) k-means has a flavor of stochastic gradient descent whose benefits are twofold. First, it dramatically reduces the per-iteration cost for updating the centroids and thus is able to handle big data efficiently. Second, similar to its successful application to deep learning [11], mini-batch gradient introduces noise in minimization and may help to bypass some bad local minima. Furthermore, the aforementioned Elkan's technique can be combined with mini-batch k-means for further acceleration [15].

In this paper, we propose a backward Euler based algorithm for k-means clustering. Fixed-point iteration is performed to solve the implicit gradient step. As is done for stochastic mini-batch k-means, we compute the gradient only using a mini-batch of samples instead of the whole data, which enables us to handle massive data. Unlike the standard fixed-point iteration, the resulting stochastic fixed-point iteration outputs an average over its trajectory. Extensive experiments show that, with proper choice of step size for stochastic backward Euler, the proposed algorithm can improve over EM and Mini-batch EM and locate an improved minimum with decreased objective value.

In other words, while Lloyd's algorithm is effective with *a full gradient oracle* we achieve better performance with the weaker *mini-batch gradient oracle*. We are motivated by recent work by two of the authors [6] which applied a similar algorithm to accelerate the training of Deep Neural Networks.

2 Stochastic backward Euler

The celebrated proximal point algorithm (PPA) [16] for minimizing some function $f(x)$ is:

$$x^{k+1} = \mathrm{prox}_{\gamma f}(x^k) := \arg\min_x f(x) + \frac{1}{2\gamma}\|x - x^k\|^2. \tag{1}$$

PPA has the advantage of being monotonically decreasing, which is guaranteed for any step size $\gamma > 0$. Indeed, by the definition of x^{k+1} in (1), we have

$$f(x^{k+1}) \leq f(x^k) - \frac{1}{2\gamma}\|x^{k+1} - x^k\|^2.$$

When $\gamma \in [c, \frac{1}{L(\nabla f)})$ for any $c > 0$ with $L(\nabla f)$ being the Lipschitz constant of ∇f, the (subsequential) convergence to a stationary point is established in [10]. If f is differentiable at x^{k+1}, it is easy to check that the following optimality condition to (1) holds

$$\nabla f(x^{k+1}) + \frac{1}{\gamma}(x^{k+1} - x^k) = 0.$$

By rearranging the terms, we arrive at implicit gradient descent or the so-called backward Euler:

$$x^{k+1} = x^k - \gamma \nabla f(x^{k+1}). \tag{2}$$

Fixed point iteration is a viable option for solving (2) if ∇f has the Lipschitz constant $L(\nabla f)$ and $\gamma < \frac{1}{L(\nabla f)}$, which is essentially the gradient descent on (1) with step size γ.

Proposition 1 *If $\gamma < \frac{1}{L(\nabla f)}$, then we have*

(a) $f(x) + \frac{1}{2\gamma}\|x - x^k\|^2$ is strongly convex, and the proximal problem (1) has a unique solution y^*.
(b) The fixed point iteration

$$y^{l+1} = x^k - \gamma \nabla f(y^l) \tag{3}$$

generates a sequence $\{y^l\}$ converging to y^* at least linearly.

See [4, Proposition 1.2.3].

Let us consider k-means clustering for a set of data points $\{p_i\}_{i=1}^N$ in $\mathbb{R}^d$ with K centroids $\{x_j\}_{j=1}^K$. We assume each cluster contains the same number of points. Denoting $x = [x_1, \ldots, x_K]^\top \in \mathbb{R}^{Kd}$, we seek to minimize

$$\min_{x \in \mathbb{R}^{Kd}} \ \phi(x) := \frac{1}{2N} \sum_{i=1}^{N} \min_{1 \leq j \leq K} \|x_j - p_i\|^2. \tag{4}$$

Note that ϕ is non-differentiable at x if there exist p_i and $j_1 \neq j_2$ such that

$$j_1, j_2 \in \arg\min_{1 \leq j \leq K} \|x_j - p_i\|^2.$$

This means that there is a data point p_i which has two or more distinct nearest centroids x_{j_1} and x_{j_2}. The same situation may happen in the assignment step of Lloyd's algorithm. In this case, we simply assign p_i to one of the nearest

centroids. With that said, ϕ is basically piecewise differentiable. By abuse of notation, we can define the 'gradient' of ϕ at any point x by

$$\nabla\phi(x) = \frac{1}{N}[\sum_{i\in\mathcal{C}_1}(x_1 - p_i), \ldots, \sum_{i\in\mathcal{C}_K}(x_K - p_i)]^\top, \tag{5}$$

where $\mathcal{C}_j$ denotes the index set of the points that are assigned to the centroid x_j. From now on and for the rest of the paper, we denote the piecewise gradient by $\nabla\phi$ as stated in (2), and none of the results depends on the specific assignment of ambiguous data points p_i. Similarly, we can compute the 'Hessian' of ϕ as was done in [5]:

$$\nabla^2\phi(x) = \frac{1}{N}\mathrm{Diag}\big(|\mathcal{C}_1|\mathbf{1}_{(|\mathcal{C}_1|)}, \ldots, |\mathcal{C}_K|\mathbf{1}_{(|\mathcal{C}_K|)}\big),$$

where $\mathbf{1}_{(n)}$ is an n-D vector of all ones. When the number of points N is large and x_j's are distinct from each other, the jumps at discontinuities of $\nabla\phi$ caused by the ambiguity in the assignment of data points to centroids are very small. Thus with (5), one can roughly consider $\nabla\phi$ to be Lipschitz continuous with the Lipschitz constant $L(\nabla\phi) \approx \frac{1}{K}$ by ignoring these tiny jumps. In what follows, we analyze how the fixed point iteration (3) works on the piecewise differentiable ϕ with discontinuous $\nabla\phi$.

Definition 1 g is piecewise Lipschitz continuous on Ω with Lipschitz constant L, if Ω can be partitioned into a finite number of sub-domains Ω_I satisfying $\cup_I\Omega_I = \mathbb{R}^{Kd}$, $\Omega_I \cap \Omega_J = \emptyset$, $\forall\, I \neq J$, and g is Lipschitz continuous in each sub-domain Ω_I, i.e., for each Ω_I we have

$$\|g(x) - g(y)\| \leq L\|x - y\| \quad \forall x, y \in \Omega_I$$

According to the definition, we can see that $\nabla\phi$ is approximately piecewise $\frac{1}{K}$-Lipschitz continuous. But in the extreme case where just one point is assigned to each of the first $K-1$ clusters and $N-K+1$ points to the last cluster, $\nabla\phi$ only has piecewise Lipschitz constant of $\frac{N-K+1}{N} \approx 1$. The following result proves the convergence of fixed point iteration on k-means problem.

Theorem 1 *Let ϕ be the k-means objective function defined in (4). Suppose $\nabla\phi$ is piecewise L-Lipschitz. If $\gamma < 1/L$, then the fixed point iteration for minimizing $h(x) := \phi(x) + \frac{1}{2\gamma}\|x - x^k\|^2$ given by*

$$y^{l+1} = x^k - \gamma\nabla\phi(y^l)$$

with the initialization $y^0 = x^k$ satisfies

(a) $h(y^{l+1}) \leq h(y^l) - (\frac{1}{2\gamma} - \frac{L}{2})\|y^{l+1} - y^l\|^2$ and $\|y^{l+1} - y^l\| \to 0$ as $l \to \infty$.

(b) $\{y^l\}$ is bounded. Moreover, if any limit point y^* of a convergent subsequence of $\{y^l\}$ lies in the interior of some sub-domain, then the whole sequence $\{y^l\}$ converges to y^* with a locally linear rate, which is a fixed point obeying

$$y^* = x^k - \gamma\nabla\phi(y^*).$$

Proof (a) We know that ϕ is piecewise quadratic. Suppose $y^l \in \Omega_I$ (note that y^l could be on the boundary), then ϕ has a uniform expression restricted on Ω_I which is a quadratic function, denoted by ϕ_{Ω_I}. We can extend the domain of ϕ_{Ω_I} from Ω_I to the whole $\mathbb{R}^{Kd}$, and we denote the extended function still by ϕ_{Ω_I}. Since ϕ_{Ω_I} is quadratic, $\nabla\phi_{\Omega_I}$ is L-Lipschitz continuous on $\mathbb{R}^{Kd}$. Then we have the following well-known inequality

$$\begin{aligned}\phi_{\Omega_I}(y^{l+1}) &\le \phi_{\Omega_I}(y^l) + \langle \nabla\phi_{\Omega_I}(y^l), y^{l+1} - y^l\rangle + \frac{L}{2}\|y^{l+1} - y^l\|^2 \\ &= \phi(y^l) + \langle \nabla\phi(y^l), y^{l+1} - y^l\rangle + \frac{L}{2}\|y^{l+1} - y^l\|^2.\end{aligned}$$

Using the above inequality and the definition of ϕ, we have

$$\begin{aligned}h(y^{l+1}) &= \phi(y^{l+1}) + \frac{1}{2\gamma}\|y^{l+1} - x^k\|^2 \le \phi_{\Omega_I}(y^{l+1}) + \frac{1}{2\gamma}\|y^{l+1} - x^k\|^2 \\ &\le \phi(y^l) + \langle \nabla\phi(y^l), y^{l+1} - y^l\rangle + \frac{L}{2}\|y^{l+1} - y^l\|^2 + \frac{1}{2\gamma}\|y^{l+1} - x^k\|^2 \\ &= \phi(y^l) + \langle \nabla\phi(y^l), y^{l+1} - y^l\rangle + (\frac{L}{2} - \frac{1}{2\gamma})\|y^{l+1} - y^l\|^2 \\ &\qquad + \frac{1}{2\gamma}\|y^l - x^k\|^2 + \frac{1}{\gamma}\langle y^{l+1} - x^k, y^{l+1} - y^l\rangle \\ &= h(y^l) - (\frac{1}{2\gamma} - \frac{L}{2})\|y^{l+1} - y^l\|^2 + \langle \frac{1}{\gamma}(y^{l+1} - x^k) + \nabla\phi(y^l), y^{l+1} - y^l\rangle \\ &= h(y^l) - (\frac{1}{2\gamma} - \frac{L}{2})\|y^{l+1} - y^l\|^2.\end{aligned}$$

In the second equality above, we used the identity

$$\frac{1}{2}\|a - b\|^2 + \langle a, b\rangle = \frac{1}{2}\|a\|^2 + \frac{1}{2}\|b\|^2$$

with $a = y^{l+1} - y^l$ and $b = y^{l+1} - x^k$. Since $\gamma < \frac{1}{L}$, $\{h(y^l)\}$ is monotonically decreasing. Moreover, since h is bounded from below by 0, $\{h(y^l)\}$ converges and thus $\|y^{l+1} - y^l\| \to 0$ as $l \to \infty$.

(b) Since $h(y) \to \infty$ as $y \to \infty$, combining with the fact that $h(y^l) \le h(y^{l+1})$, we have $\{y^l\} \subseteq \{y \in \mathbb{R}^{Kd} : h(y) \le h(y^0)\}$ is bounded. Consider a convergent subsequence $\{y^{l_m}\}$ whose limit y^* lies in the interior of some sub-domain. Then for sufficiently large l_m, $\{y^{l_m}\}$ will always remain in the same sub-domain in which y^* lies and thus $\lim_{l_m\to\infty} \nabla\phi(y^{l_m}) = \nabla\phi(y^*)$. Since by (a), $\|y^{l+1} - y^l\| \to 0$, we have $\|\nabla\phi(y^{l+1}) - \nabla\phi(y^l)\| = \frac{1}{\gamma}\|y^l - y^{l-1}\| \to 0$ as $l \to \infty$. Therefore,

$$\begin{aligned}0 &= \lim_{l_m\to\infty} y^{l_m} - x^k + \gamma\nabla\phi(y^{l_m-1}) = \lim_{l_m\to\infty} y^{l_m} - x^k + \gamma\nabla\phi(y^{l_m}) \\ &= y^* - x^k + \gamma\phi\nabla(y^*),\end{aligned}$$

which implies y^* is a fixed point. Furthermore, by the piecewise Lipschitz condition,

$$\|y^{l_m+1} - y^*\| = \gamma\|\nabla\phi(y^{l_m}) - \nabla\phi(y^*)\| \leq L\gamma\|y^{l_m} - y^*\|.$$

Since $L\gamma < 1$, when l_m is sufficiently large, y^{l_m+1} is also in the same subdomain containing y^*. By repeatedly applying the above inequality for $l > l_m$, we conclude that $\{y^l\}$ converges to y^*.

Remark 1 This result can be extended to objective functions that are the pointwise infimum of a set of a finite number of Lipschitz differentiable functions.

2.1 Algorithm description

Instead of using the full gradient $\nabla\phi$ in fixed-point iteration, we adopt a randomly sampled mini-batch gradient

$$\nabla_l\phi = \frac{1}{M}[\sum_{i\in\mathcal{C}_1^l}(x_1 - p_i), \ldots, \sum_{i\in\mathcal{C}_K^l}(x_K - p_i)]^\top$$

at the l-th inner iteration. Here, $\mathcal{C}_j^l$ denotes the index set of the points in the l-th mini-batch associated with the centroid x_j obeying $\sum_{j=1}^K |\mathcal{C}_j^l| = M$. The fixed-point iteration outputs a forward looking average over its trajectory. Intuitively averaging greatly stabilizes the noisy mini-batch gradients and thus smooths the descent. We summarize the proposed algorithm in Algorithm 1. Another key ingredient of our algorithm is an aggressive initial step size $\gamma^0 \approx \frac{1}{L(\nabla\phi)} \approx K$, which helps pass bad local minimum at the early stage. Unlike in deterministic backward Euler, diminishing step size is needed to ensure convergence. But γ should decay slowly because large step size is good for a global search.

Algorithm 1 Stochastic backward Euler for k-means.

Input: number of clusters K, step size $\gamma^0 \approx K$, mini-batch size M, averaging parameter $\alpha > 0$, step size decay parameter $\beta \lessapprox 1$.
Initialize: centroid x^0.

```
for k = 1, ..., omaxit do
    y^{0,k} = x^{k-1}
    x^k = y^{0,k}
    for l = 1, ..., imaxit do
        Randomly sample a mini-batch gradient ∇_l φ.
        y^{l,k} = x^{k-1} − γ^k ∇_l φ(y^{l-1,k})
        x^k = α x^k + (1 − α) y^{l,k}
    end for
    γ^k = β γ^{k-1}
end for
```

Output: x^{omaxit}

2.2 Related work

Chaudhari et al. [7] recently proposed the entropy stochastic gradient descent (Entropy-SGD) algorithm to tackle the training of deep neural networks. Relaxation techniques arising in statistical physics were used to change the energy landscape of the original non-convex objective function $f(x)$ yet with the minimizers being preserved, which allows easier minimization to obtain a 'good' minimizer with a better geometry. More precisely, they suggest to replace $f(x)$ with a modified objective function $f_\gamma(x)$ called local entropy [3] as follows

$$f_\gamma(x) := -\frac{1}{\beta}\log\big(G_{\beta^{-1}\gamma} * \exp(-\beta f(x))\big),$$

where $G_\gamma(x) = (2\pi\gamma)^{-d/2}\exp\big(-\frac{|x|^2}{2\gamma}\big)$ is the heat kernel. The connection between Entropy-SGD and nonlinear partial differential equations (PDEs) was later established in [6]. The local entropy function f_γ turns out to be the solution to the following viscous Hamilton-Jacobi (HJ) PDE at $t = \gamma$

$$u_t = -\frac{1}{2}|\nabla u|^2 + \frac{\beta^{-1}}{2}\Delta u \tag{6}$$

with the initial condition $u(x, 0) = f(x)$. In the limit $\beta^{-1} \to 0$, (6) reduces to the non-viscous HJ equation

$$u_t = -\frac{1}{2}|\nabla u|^2,$$

whose viscosity solution is exactly the Moreau envelope [14]:

$$u(x,t) = \inf_y \Big\{ f(y) + \frac{1}{2t}\|y - x\|^2 \Big\}.$$

The gradient descent dynamics for f_γ is obtained by taking the limit of the following system of stochastic differential equation as the homogenization parameter $\varepsilon \to 0$:

$$\begin{aligned} dx(s) &= -\gamma^{-1}(x - y)ds \\ dy(s) &= -\frac{1}{\varepsilon}\big[\nabla f(y) + \frac{y - x}{\gamma}\big]ds + \frac{\beta^{-1/2}}{\sqrt{\varepsilon}}dW(s) \end{aligned} \tag{7}$$

where $W(s)$ is the standard Wiener process. Specifically, we have

$$-\nabla f_\gamma(x) = -\gamma^{-1}(x - \langle y\rangle)$$

with $\langle y\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^T y(s)ds$ and $y(s)$ being the solution of (7) for fixed x. This gives rise to the implementation of Entropy-SGD [6]:

$$\begin{aligned} y^{l+1,k} &= y^{l,k} - \eta_y\Big(\nabla_l f(y^{l,k}) + \frac{y^{l,k} - x^k}{\gamma^k}\Big) + \sqrt{\eta_y\beta^{-1}}\varepsilon \quad \text{(inner loop)} \\ x^{k+1} &= x^k - \eta_x\frac{x^k - \langle y\rangle^k}{\gamma^k} \quad \text{(outer loop)} \end{aligned}$$

where η_y and η_x are the gradient step sizes for the inner and outer loops, respectively, $\langle y\rangle^k$ is the moving average of $\{y^{l,k}\}$ output from the inner loop, and $\sqrt{\eta_y\beta^{-1}}\varepsilon$ introduces the noise. Stochastic backward Euler simplifies Entropy-SGD in two aspects. First, the term $\sqrt{\eta_y\beta^{-1}}\varepsilon$ is removed in SBE as the mini-batch gradient $\nabla_l f$ itself already contains the noise. Second, the step sizes η_y and η_x are both set to γ^k, which makes the algorithm simpler with less tunable parameters.

3 Experimental results

We show by several experiments that the proposed stochastic backward Euler (SBE) gives superior clustering results compared with the state-of-the-art algorithms for k-means. SBE scales well for large problems. In practice, only a small number of fixed-point iterations are needed in the inner loop, and this seems not to depend on the size of the problem. Specifically, we chose the parameters `imaxit` $= 5$ or 10 and the averaging parameter $\alpha = 0.75$ in all experiments. Moreover, we always set $\gamma^0 = K$.

3.1 2-D synthetic Gaussian data

We generated 4000 synthetic data points in 2-D plane by multivariate normal distributions with 1000 points in each cluster. The means and covariance matrices used for Gaussian distributions are as follows:

$$\mu_1 = \begin{bmatrix}-5\\-3\end{bmatrix},\ \mu_2 = \begin{bmatrix}5\\-3\end{bmatrix},\ \mu_3 = \begin{bmatrix}0.0\\5.0\end{bmatrix},\ \mu_4 = \begin{bmatrix}2.5\\4.0\end{bmatrix};$$
$$\Sigma_1 = \begin{bmatrix}0.8 & 0.1\\0.1 & 0.8\end{bmatrix},\ \Sigma_2 = \begin{bmatrix}1.2 & 0.6\\0.6 & 0.7\end{bmatrix},\ \Sigma_3 = \begin{bmatrix}0.5 & 0.05\\0.05 & 1.6\end{bmatrix},\ \Sigma_4 = \begin{bmatrix}1.5 & 0.05\\0.05 & 0.6\end{bmatrix}.$$

For the initial centroids given below,both Lloyd's algorithm (or EM) and mini-batch EM got stuck at the same local minimum with objective value about 1.34; see the left plot of Fig. 1.

$$x_1 = \begin{bmatrix}-5.5989\\-2.7090\end{bmatrix},\ x_2 = \begin{bmatrix}-4.4572\\-4.0614\end{bmatrix},\ x_3 = \begin{bmatrix}-0.1082\\5.2889\end{bmatrix},\ x_4 = \begin{bmatrix}2.3485\\3.5286\end{bmatrix}.$$

Starting from where EM and mini-batch EM got stuck, we can see that SBE managed to jump over the trap of local minimum and arrived at a better minimum, which seems to be the global minimum; see the right plot of Fig. 1.

3.2 Iris dataset

The Iris dataset, which contains 150 4-D data samples from 3 clusters, was used for comparisons of SBE, EM as well as mini-batch EM algorithms. 100 runs were realized with the initial centroids randomly selected from the data

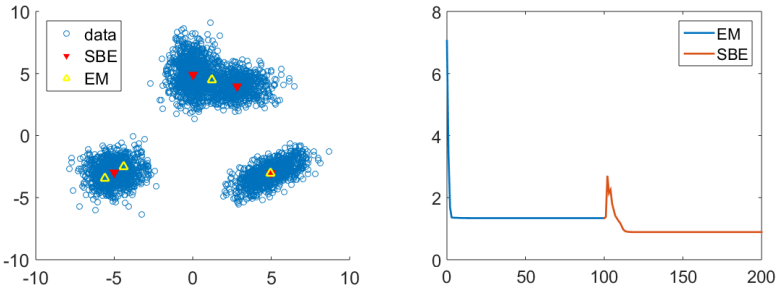

Fig. 1 Synthetic Gaussian data with 4 centroids. Left: Computed centroids by EM and SBE corresponding to the objective values 1.34 and 0.89, respectively. Right: Plot of objective value v.s. number of iteration. EM converged quickly but got trapped at a local minimum. SBE bypassed this local minimum and reached a better minimum by jumping over a hill.

samples. For the parameters, we chose mini-batch size $M = 60$, initial step size, `imaxit`$= 40$, `omaxit`$= 10$, and decay parameter $\beta = \frac{1}{1.01}$. The histograms in Fig. 2 record the frequency of objective values given by the three algorithms. Clearly there was 29% chance that EM got stuck at a local minimum whose value is about 0.48, whereas both SBE and mini-batch EM managed to locate an improved minimum valued at around 0.264 *every time*.

3.3 Gaussian data with MNIST centroids

We selected 8 hand-written digit images of dimension $28 \times 28 = 784$ from MNIST dataset shown in Fig. 3, and then generated 60,000 images from these 8 centroids by adding Gaussian noise. We compare SBE with both EM and mini-batch EM (mb-EM) [17,18] on 100 independent realizations with random initial guess. For each method, we recorded the minimum, maximum, mean and variance of the 100 objective values by the computed centroids.

We first compare SBE and EM with the true number of clusters $K = 8$. For SBE, mini-batch size $M = 1000$, maximum number of iterations for backward Euler `omaxit`$=150$, maximum fixed-point iterations `imaxit`$= 10$ for SBE. We set the maximum number of iterations for EM to be 50, which was sufficient for its convergence. The results are listed in the first two rows of Table 3.3. We observed SBE always found a minimum around 15.68 up to a tiny error due to the noise from mini-batch. Moreover, note that although we run more iterations (taking the inner loop into account) for SBE than for EM, SBE actually requires less gradient evaluations and is computationally cheaper compared with EM, because we are able to take a much smaller batch size.

In the comparison between SBE and mb-EM, we reduced mini-batch size to $M = 500$, `omaxit`$= 100$, `imaxit`$= 5$ and tested for $K = 6, 8, 10$. Table 3.3 shows that with the same mini-batch size, SBE outperforms mb-EM in all three cases, in terms of both mean and variance of the objective values.

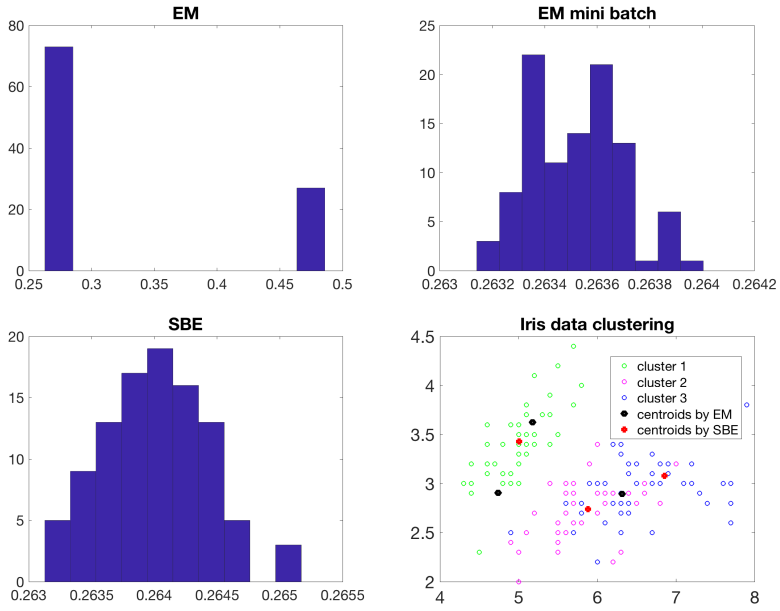

Fig. 2 The Iris dataset with 3 clusters. Top left: histogram of objective values obtained by EM in 100 trials. Top right: histogram of objective values obtained by mini-batch EM in 100 trials. Bottom left: histogram of objective values obtained by SBE (proposed) in 100 trials. Bottom right: computed centroids by EM (black) and SBE (red), corresponding to the objective values 0.48 and 0.264, respectively.

Fig. 3 8 selected images from MNIST dataset. 60,000 sample images are generated from these 8 images by adding Gaussian noise.

3.4 Raw MNIST data

In this example, We used the 60,000 images from the MNIST training set for clustering test, with 6000 samples for each digit (cluster) from 0 to 9. The comparison results are shown in Table 3.4. We conclude that SBE consistently performs better than EM and mb-EM. The histograms of objective value by the three algorithms in the case $K = 10$ are plotted in Fig. 4.

K	Method	Batch size	Max iter	Min	Max	Mean	Variance
8	EM	60000	50	15.6800	27.2828	20.0203	6.0030
	SBE	1000	(150,10)	15.6808	15.6808	15.6808	1.49×10^{-10}
6	mb-EM	500	100	20.44	23.4721	21.8393	0.67
	SBE	500	(100,5)	20.2989	21.2047	20.4939	0.0439
8	mb-EM	500	100	15.9193	18.5820	16.4009	0.7646
	SBE	500	(100,5)	15.6816	15.6821	15.6820	1.18×10^{-9}
10	mb-EM	500	100	15.9148	18.1848	16.1727	0.4332
	SBE	500	(100,5)	15.6823	15.6825	15.6824	1.5×10^{-9}

Table 1 Gaussian data generated from MNIST centroids by adding noise. Ground truth $K = 8$. Clustering results for 100 independent trails with random initialization.

K	Method	Batch size	Max iter	Min	Max	Mean	Variance
10	EM	60000	50	19.6069	19.8195	19.6725	0.0028
	SBE	1000	(150,10)	19.6087	19.7279	19.6201	5.7×10^{-4}
8	mb-EM	500	100	20.4948	20.7126	20.5958	0.0018
	SBE	500	(100,5)	20.2723	20.4104	20.3090	0.0014
10	mb-EM	500	100	19.9029	20.2347	20.0146	0.0041
	SBE	500	(100,5)	19.6103	19.7293	19.6354	0.0011
12	mb-EM	500	100	19.3978	19.7147	19.5136	0.0042
	SBE	500	(100,5)	19.0492	19.1582	19.0972	6.2×10^{-4}

Table 2 Raw MNIST training data. The ground truth number of clusters is $K = 10$. Clustering results for 100 independent trails with random initialization.

K	Method	Batch size	Max iter	Min	Max	Mean	Variance
10	EM	60000	50	1.6238	3.0156	2.1406	0.0977
	SBE	1000	(150,10)	1.6238	1.6239	1.6239	2.7×10^{-10}
8	mb EM	500	100	2.3428	3.5972	2.7157	0.0666
	SBE	500	(100,5)	2.2833	2.4311	2.3274	0.0015
10	mb EM	500	100	1.6504	2.6676	2.1391	0.0712
	SBE	500	(100,5)	1.6239	1.6242	1.6240	1.37×10^{-9}
12	mb EM	500	100	1.5815	2.6189	1.7853	0.0661
	SBE	500	(100,5)	1.5326	1.5891	1.5622	9.8×10^{-5}

Table 3 MNIST features generated by LeNet-5 network. The ground truth number of clusters is $K = 10$. Clustering results for 100 independent trails with random initialization.

3.5 MNSIT features

We extracted the feature vectors of MNIST training data prior to the last layer of LeNet-5 [12]. The feature vectors have dimension 64 and lie in a better manifold compared with the raw data. The results are shown in Table 3 and Fig. 5 and 6.

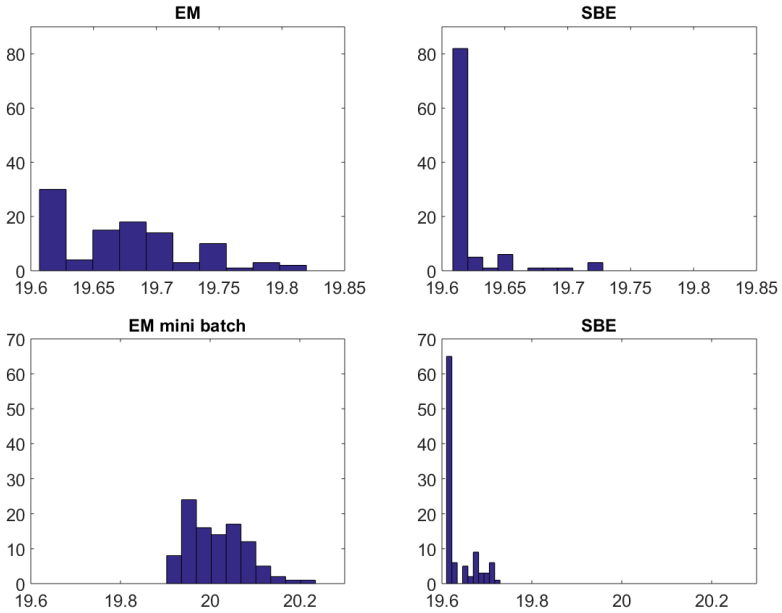

Fig. 4 Histograms of objective value for MNIST training data with ground truth number of clusters $K = 10$. Top left: EM. Top right: SBE, mini-batch size of 1000. Bottom left: mn-EM, mini-batch size of 500. Bottom right: SBE, mini-batch size of 500.

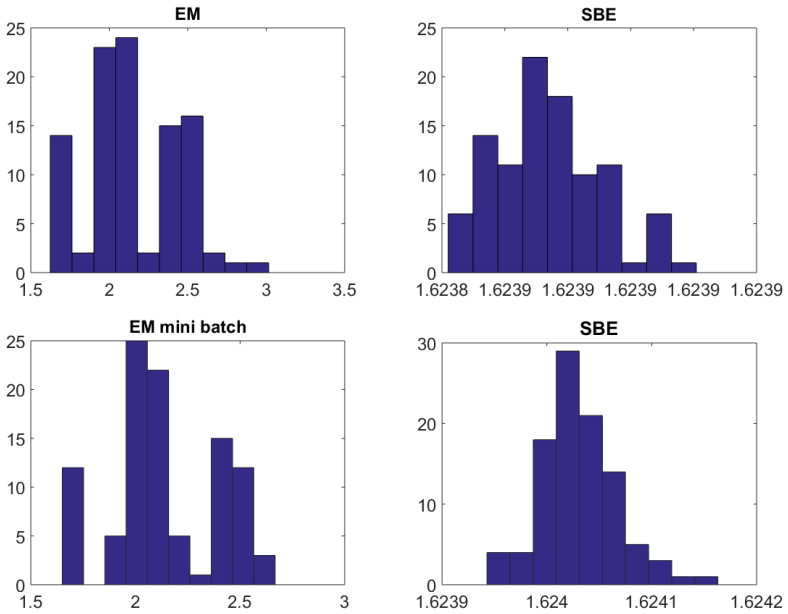

Fig. 5 Histograms of objective value for MNIST feature data with ground truth number of clusters K=10. Top left: EM. Top right: SBE, mini-batch size of 1000. Bottom left: mn-EM, mini-batch size of 500. Bottom right: SBE, mini-batch size of 500.

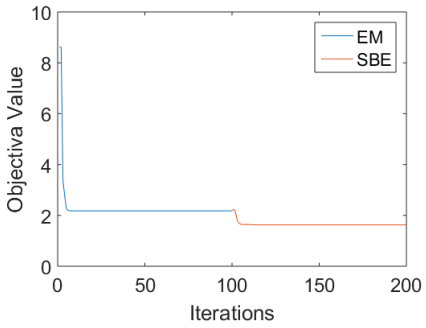

Fig. 6 Objective value for MNIST features dataset. The ground truth number of clusters is $K = 10$. EM got trapped at local minimum around 2.178. Initializing SBE with this local minimizer, an improved minimum around 1.623 was found.

4 Discussions

At the $(k+1)$-th iteration, SBE approximately solves $x = x^k - \gamma\nabla\phi(x)$ due to the noise introduced by mini-batch gradient. Since $\nabla\phi$ is technically only piecewise Lipschitz continuous, the backward Euler may have multiple solutions. For example, for the 1-D example in Fig. 7, we get two solutions $x_{\mathrm{BE},1}$ in the leftmost valley and $x_{\mathrm{BE},2}$ in the second from the left. x^{k+1} solved by SBE is close to $x_{\mathrm{BE},2}$ as it gives a lower objective value of f_γ. Then x^{k+1} bypasses the local minimum, which explains the increase of objective value showed in the right plot of Fig. 1. We conjecture that averaging of the iterates $\{y^l\}$ enables an aggressive step size γ larger than the theoretical upper-bound $1/L$ as in Theorem 1, which also helps skip bad local minima. We did observe blowup phenomenon numerically when without this average scheme. It is of our interest to prove an improved upper-bound for γ in the future work. Similar to what was done in [1] , another direction is to analyze how exact the inner problems have to be solved in order to still guarantee the convergence of the outer Backward Euler problem.

Acknowledgement

This work was partially supported by AFOSR grant FA9550-15-1-0073 and ONR grant N00014-16-1-2157. We would like to thank Dr. Bao Wang for helpful discussions.

References

1. M. Artina, M. Fornasier, and F. Solombrino. "Linearly constrained nonsmooth and nonconvex minimization. SIAM Journal on Optimization", 23(3), 1904-1937, 2013.
2. D. Arthur and S. Vassilvitskii. "k-means++: The advantages of careful seeding", Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.

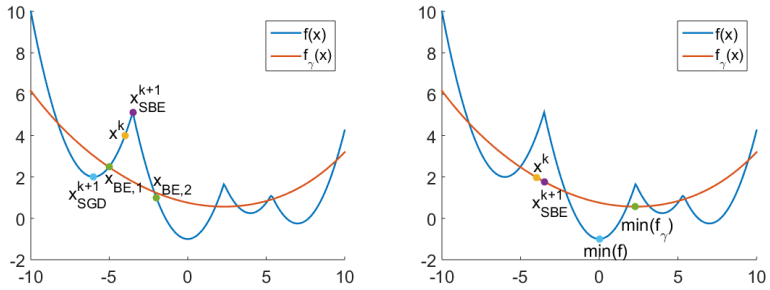

Fig. 7 Comparison the updates between SGD and SBE. Left: In the $(k+1)$-th update, SGD gives $\mathbb{E}_{\mathrm{SGD}}[x^{k+1}] = x^k - \gamma\nabla f(x^k)$ while SBE solves gradient descent on f_γ and ends up with x^{k+1}_{SBE}. Right: SBE tends to converge to the global minimum of local entropy f_γ. We must let $\gamma \to 0$ in order for SBE to converge to the true global minimum of f.

3. C. Baldassi, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina, "Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses", Physical review letters, 115(12), 128-101, 2015.
4. D.P. Bertsekas: "Nonlinear Programming" Second Edition, Athena Scientific, Belmont, Massachusetts, 2008.
5. L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms", Advances in neural information processing systems, 1995.
6. P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina. "Entropy-sgd: Biasing gradient descent into wide valleys", arXiv preprint arXiv:1611.01838, 2016.
7. P. Chaudhari, A. Oberman, S. Osher, S. Soatto, and G. Carlier, "Deep Relaxation: partial differential equations for optimizing deep neural networks", arXiv preprint arXiv:1704.04932, 2017.
8. Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz, "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup", Proceedings of the 32nd International Conference on Machine Learning, 2015.
9. C. Elkan, "Using the triangle inequality to accelerate k-means", Proceedings of the 20th International Conference on Machine Learning, 2003.
10. A. Kaplan and R. Tichatschke, "Proximal point method and nonconvex optimization", Journal of Global Optimization, 13, 389-406, 1998.
11. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", Nature, 521(7553), 436-444, 2015.
12. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based Learning Applied to Document Recognition", Proceedings of the IEEE, 86(11), 2278-2324, 1998.
13. S. Lloyd, "Least squares quantization in PCM", IEEE transactions on information theory, 28(2), 129-137, 1982.
14. J.-J. Moreau, " Proximité et dualité dans un espace hilbertien", Bulletin de la Société Mathématique de France, 93, 273–299, 1965.
15. J. Newling and F. Fleuret, "Nested Mini-Batch k-means", Advances in Neural Information Processing Systems, 2016.
16. R. Rockafellar, "Monotone operators and the proximal point algorithm", SIAM J. Control and Optimization, 14, 877-898, 1976.
17. D. Sculley, "Web-scale k-means clustering", Proceedings of the 19th international conference on World wide web, ACM, 2010.
18. C. Tang and C. Monteleoni. "Convergence rate of stochastic k-means", arXiv preprint arXiv:1610.04900, 2016.