Supplementary material for:
Linear-time training of nonlinear low-dimensional embeddings

Miguel Á. Carreira-Perpiñán and Max Vladymyrov
Electrical Engineering and Computer Science, University of California, Merced
{mcarreira-perpinan, mvladymyrov}@ucmerced.edu

February 21, 2014

1 Fast Gauss Transform approximation

The fast Gauss transform (FGT; Greengard and Strain, 1991) uses the idea of the fast multipole methods (FMM) to compute approximately in linear time the Gaussian interaction between the points $\mathbf{x}_n \in \mathbb{R}^d, n = 1, \dots, N$ of the form

$$Q(\mathbf{x}_n) = \sum_{m=1}^{N} q_m \exp(-\left\|(\mathbf{x}_n - \mathbf{x}_m)/\sigma\right\|^2). \tag{1}$$

The algorithm starts by normalizing the points to lie in the unit hypercube and dividing the space into boxes of side $\sqrt{2}\sigma r$, where $r < 1/\sqrt{2}$ is a user parameter. Then, there exist three different ways to compute the approximation to (1) (cf. fig. 2 from our main paper):

- We can use a Hermite expansion for each box $\mathcal{B}$ (with center $\mathbf{s}_{\mathcal{B}}$) and evaluate it for all target points $\mathbf{t}$[1]:

$$Q(\mathbf{t}) = \sum_{\mathcal{B}} \sum_{\boldsymbol{\alpha}<p} A_{\boldsymbol{\alpha}}^{\mathcal{B}} h_{\boldsymbol{\alpha}}\Big(\frac{\mathbf{t}-\mathbf{s}_{\mathcal{B}}}{\sigma}\Big) + \epsilon_H(p), \tag{2}$$

  where $A_{\boldsymbol{\alpha}}^{\mathcal{B}} = \frac{1}{\boldsymbol{\alpha}!}\sum_{\mathbf{s}_j \in \mathcal{B}} q_j (\frac{\mathbf{s}_j - \mathbf{s}_{\mathcal{B}}}{\sigma})^{\boldsymbol{\alpha}}$ and $\epsilon_H(p)$ is a known error term, defined below.

- For every source point in a box $\mathcal{B}$ we can we can form a Taylor series of the target points $\mathbf{t}$ in the nearby boxes $\mathcal{C}$ with corresponding centers $\mathbf{t}_{\mathcal{C}}$:

$$Q(\mathbf{t}) = \sum_{\boldsymbol{\beta}<p} \Bigg(\sum_{\mathcal{B}} C_{\boldsymbol{\beta}}^{\mathcal{BC}}\Bigg) \Big(\frac{\mathbf{t}-\mathbf{t}_{\mathcal{C}}}{\sigma}\Big)^{\boldsymbol{\beta}} + \epsilon_T(p), \tag{3}$$

  where $C_{\boldsymbol{\beta}}^{\mathcal{BC}} = \frac{1}{\boldsymbol{\beta}!}\sum_{\mathbf{s}_j \in \mathcal{B}} q_j h_{\boldsymbol{\beta}}(\frac{\mathbf{s}_j - \mathbf{t}_{\mathcal{C}}}{\sigma})$ and $\epsilon_T(p)$ is a known error term, defined below.

- We can further approximate (2) by expanding $h_{\boldsymbol{\alpha}}(\mathbf{t})$ as a Taylor series to get

$$Q(\mathbf{t}) = \sum_{\boldsymbol{\beta}<p} \Bigg(\sum_{\mathcal{B}} \widehat{C}_{\boldsymbol{\beta}}^{\mathcal{BC}}\Bigg) \Big(\frac{\mathbf{t}-\mathbf{t}_{\mathcal{C}}}{\sigma}\Big)^{\boldsymbol{\beta}} + \epsilon_{TH}(p), \tag{4}$$

  where $\widehat{C}_{\boldsymbol{\beta}}^{\mathcal{BC}} = \frac{1}{\boldsymbol{\beta}!}\sum_{\boldsymbol{\alpha}<p} A_{\boldsymbol{\alpha}}^{\mathcal{B}} (-1)^{|\boldsymbol{\alpha}|} h_{\boldsymbol{\alpha}+\boldsymbol{\beta}}(\frac{\mathbf{s}_{\mathcal{B}} - \mathbf{t}_{\mathcal{C}}}{\sigma})$ and $\epsilon_{TH}(p)$ is a known error term, defined below.

Computing multiindex sums over $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ scales as $\mathcal{O}(p^d N)$, which is the bottleneck computation. The rest of the computations add a linear cost to the algorithm and depend of the number of boxes included around every source box $\mathcal{B}$.

The choice of the method depends on the number of source points $N_{\mathcal{B}}$ in a given box $\mathcal{B}$ and the number of target points $M_{\mathcal{C}}$ in a given box $\mathcal{C}$. The user provides two cutoff parameters $\bar{N}_{\mathcal{B}}$ and $\bar{M}_{\mathcal{C}}$. Now, one of the following can occur:

[1] We use multi-index notation: $\boldsymbol{\alpha} \geq 0 \Rightarrow \alpha_1, \dots, \alpha_d \geq 0$; $\boldsymbol{\alpha}! = \alpha_1! \cdots \alpha_d!$; $\mathbf{t}^{\boldsymbol{\alpha}} = t_1^{\alpha_1} \cdots t_d^{\alpha_d}$ for $\boldsymbol{\alpha} \in \mathbb{N}^d$, $\mathbf{t} \in \mathbb{R}^d$.

- If $N_{\mathcal{B}} < \bar{N}_{\mathcal{B}}$ and $M_{\mathcal{C}} < \bar{M}_{\mathcal{C}}$ use exact evaluation (1).
- If $N_{\mathcal{B}} < \bar{N}_{\mathcal{B}}$ and $M_{\mathcal{C}} \geq \bar{M}_{\mathcal{C}}$ use Hermite expansion (2).
- If $N_{\mathcal{B}} \geq \bar{N}_a nd\mathcal{B}$ and $M_{\mathcal{C}} < \bar{M}_{\mathcal{C}}$ use Taylor expansion (3).
- If $N_{\mathcal{B}} \geq \bar{N}_{\mathcal{B}}$ and $M_{\mathcal{C}} \geq \bar{M}_{\mathcal{C}}$ use Hermite expansion followed by a Taylor expansion (4).

In the paper, for simplicity, we assume $\bar{N}_{\mathcal{B}} = \bar{M}_{\mathcal{C}} = M_0$.

The approximation errors are given in Baxter and Roussos (2002) with an extended derivation from **?**:

$$\epsilon_H(p) \leq \frac{\sum_{m=1}^N q_m}{(1-r)^d} \sum_{k=0}^{d-1} \binom{d}{k} (1-r^p)^k \left(\frac{r^p}{\sqrt{p!}} \right)^{d-k}, \tag{5}$$

$$\epsilon_T(p) \leq \frac{\sum_{m=1}^N q_m}{(1-r)^d} \sum_{k=0}^{d-1} \binom{d}{k} (1-r^p)^k \left(\frac{r^p}{\sqrt{p!}} \right)^{d-k}, \tag{6}$$

$$\epsilon_{TH}(p) \leq \epsilon_T(p) + \frac{\sum_{m=1}^N q_m}{(1-\sqrt{2}r)^{2d}} \left(\sum_{k=0}^{d-1} \binom{d}{k} (1-(\sqrt{2}r)^p)^k \left(\frac{(\sqrt{2}r)^p}{\sqrt{p!}} \right)^{d-k} \right)^2. \tag{7}$$

2 A simple noise model

In order to understand the effect of noisy updates to an optimization algorithm, we will use a simple noise model. At iteration k during the optimization of an objective function $E(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^n$, if using exact gradient evaluations, we would move from the previous iterate $\mathbf{x}_{k-1}$ to the current one $\mathbf{x}_k$ without error (for example, for a gradient descent step, $\mathbf{x}_k = \mathbf{x}_{k-1} - \eta \nabla f(\mathbf{x}_{k-1})$). However, if using an inexact gradient, we would move to $\mathbf{x}_k + \boldsymbol{\epsilon}_k$, incurring an error $\boldsymbol{\epsilon}_k$. In our case, $\boldsymbol{\epsilon}_k$ is caused by using an approximate method and is a deterministic function of $\mathbf{x}_{k-1}$ and the method parameters (θ for Barnes-Hut, p for fast multipole methods, etc.). Let us model $\boldsymbol{\epsilon}_k$ as a zero-mean Gaussian with variance σ^2 in each dimension. The fundamental assumption is that, although $\boldsymbol{\epsilon}_k$ is deterministic at each iterate, over a sequence of iterates we expect it not to have a preferred direction (i.e., no systematic error). The value of σ corresponds to the accuracy level of the method, where $\sigma = 0$ means no error ($\theta = 0$, $p \to \infty$). In practice, σ will be quite small. Then we have the following result.

Theorem 2.1. *Let $E(\mathbf{x})$ be a real function with $\mathbf{x} \in \mathbb{R}^n$. Call $\Delta E(\mathbf{x})$ and $\delta E(\mathbf{x})$ the absolute and relative error, respectively, incurred at point $\mathbf{x} \in \mathbb{R}^n$ upon a perturbation of $\mathbf{x}$ that follows a Gaussian noise model $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$. Call $\mu_\Delta(\mathbf{x}) = \langle \Delta E(\mathbf{x}) \rangle$, $v_\Delta(\mathbf{x}) = \langle (\Delta E(\mathbf{x}) - \langle \Delta E(\mathbf{x}) \rangle)^2 \rangle$, $\mu_\delta(\mathbf{x}) = \langle \delta E(\mathbf{x}) \rangle$ and $v_\delta(\mathbf{x}) = \langle (\delta E(\mathbf{x}) - \langle \delta E(\mathbf{x}) \rangle)^2 \rangle$ the expected errors and their variances under the noise model. Assume E has derivatives up to order four that are continuous and have finite expectations under the noise model. Call $\mathbf{g}(\mathbf{x}) = \nabla E(\mathbf{x})$ and $\mathbf{H}(\mathbf{x}) = \nabla^2 E(\mathbf{x})$ the gradient and Hessian at that point, respectively, and $\mathbf{J_H}(\mathbf{x})$ the $n \times n$ Jacobian matrix of the Hessian diagonal elements, i.e., $(\mathbf{J_H}(\mathbf{x}))_{ij} = \partial h_{ii}/\partial x_j = \partial^3 E(\mathbf{x})/\partial x_i^2 \partial x_j$. Then, the expected errors and their variances satisfy, $\forall \mathbf{x} \in \mathbb{R}^n$:*

$$\mu_\Delta(\mathbf{x}) = \frac{1}{2}\sigma^2 \operatorname{tr}(\mathbf{H}(\mathbf{x})) + \mathcal{O}(\sigma^4) \qquad v_\Delta(\mathbf{x}) = \sigma^2 \|\mathbf{g}(\mathbf{x})\|^2 + \sigma^4 \left(\frac{1}{2} \|\mathbf{H}(\mathbf{x})\|_F^2 + \mathbf{1}^T \mathbf{J_H}(\mathbf{x}) \mathbf{g}(\mathbf{x}) \right) + \mathcal{O}(\sigma^6) \tag{8}$$

$$\mu_\delta(\mathbf{x}) = \frac{\mu_\Delta(\mathbf{x})}{E(\mathbf{x})} \qquad v_\delta(\mathbf{x}) = \frac{v_\Delta(\mathbf{x})}{E(\mathbf{x})^2}. \tag{9}$$

If $\|\mathbf{H}(\mathbf{x})\|_2 \leq M \ \forall \mathbf{x} \in \mathbb{R}^n$ for some $M > 0$, then $\forall \mathbf{x} \in \mathbb{R}^n$:

$$|\langle \Delta E(\mathbf{x}) \rangle| \leq \frac{1}{2}\sigma^2 n M. \tag{10}$$

Proof. Throughout, we define $\langle \cdot \rangle$ to be the expectation under the Gaussian noise model $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$:

$$\langle \mathbf{f}(\boldsymbol{\epsilon}) \rangle = (2\pi\sigma^2)^{-\frac{n}{2}} \int_{\mathbb{R}^n} \mathbf{f}(\boldsymbol{\epsilon}) \exp(\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} / 2\sigma^2) \, d\boldsymbol{\epsilon}.$$

In particular, we have:

$$\langle \epsilon_i \rangle = 0, \ \langle \epsilon_i^2 \rangle = \sigma^2, \ \langle \epsilon_i^4 \rangle = 3\sigma^2 \text{ for } i = 1, \dots, n; \ \langle f(\epsilon_i) g(\epsilon_j) \rangle = \langle f(\epsilon_i) \rangle \langle g(\epsilon_j) \rangle \text{ if } i \neq j.$$

To prove expression (8), we expand E around $\mathbf{x}$ as a Taylor series to fourth order with remainder:

$$\Delta E(\mathbf{x}) = E(\mathbf{x}+\boldsymbol{\epsilon}) - E(\mathbf{x}) = \mathbf{g}^T\boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon} + \frac{1}{6}\sum_{i,j,k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k + \frac{1}{24}\sum_{i,j,k,l=1}^{n} q_{ijkl}\epsilon_i\epsilon_j\epsilon_k\epsilon_l \tag{11}$$

where

$$\mathbf{g} = \nabla E(\mathbf{x}), \quad \mathbf{H} = \nabla^2 E(\mathbf{x}), \quad t_{ijk} = \frac{\partial^3 E(\mathbf{x})}{\partial x_i \partial x_j \partial x_k}, \quad q_{ijkl} = \frac{\partial^4 E(\mathbf{x}+\xi\boldsymbol{\epsilon})}{\partial x_i \partial x_j \partial x_k \partial x_l}$$

are the corresponding derivatives (omitting the dependence on $\mathbf{x}$ to simplify the notation) and $\xi \in (0,1)$ depends on $\boldsymbol{\epsilon}$. Let us now compute the expectation of each term in (11). The first- and third-order terms (on $\mathbf{g}$ and t_{ijk}) vanish under the expectation because they are odd functions of at least one ϵ_i. For the second-order term, write the Hessian in terms of its eigenvectors and eigenvalues as $\mathbf{H} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$ with $\mathbf{U}$ orthogonal and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$ diagonal (by the spectral theorem), so we can change variables $\mathbf{u} = \mathbf{U}^T\boldsymbol{\epsilon}$. Then its expectation is:

$$\begin{aligned}
\left\langle \frac{1}{2}\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon} \right\rangle &= (2\pi\sigma^2)^{-\frac{n}{2}} \int_{\mathbb{R}^n} \left(\frac{1}{2}\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon}\right) \exp\left(\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}/2\sigma^2\right) d\boldsymbol{\epsilon} \\
&= (2\pi\sigma^2)^{-\frac{n}{2}} \int_{\mathbb{R}^n} \left(\frac{1}{2}\mathbf{u}^T\boldsymbol{\Lambda}\mathbf{u}\right) \exp\left(\mathbf{u}^T\mathbf{u}/2\sigma^2\right) d\mathbf{u} \\
&= (2\pi\sigma^2)^{-\frac{n}{2}} \sum_{i=1}^{n} \lambda_i \int_{\mathbb{R}^n} \frac{1}{2}u_i^2 \exp\left(\mathbf{u}^T\mathbf{u}/2\sigma^2\right) d\mathbf{u} \\
&= \sum_{i=1}^{n} \lambda_i \left\langle \frac{1}{2}u_i^2 \right\rangle = \frac{1}{2}\sigma^2 \sum_{i=1}^{n} \lambda_i = \frac{1}{2}\sigma^2 \operatorname{tr}(\mathbf{H}).
\end{aligned}$$

The fourth-order term can be handled in a similar way and its expectation, which is finite by assumption, has the form $\mathcal{O}(\sigma^4)$. Hence $\langle \Delta E(\mathbf{x}) \rangle = \frac{1}{2}\sigma^2 \operatorname{tr}(\mathbf{H}(\mathbf{x})) + \mathcal{O}(\sigma^4)$.

For the variance of the absolute error we have $\left\langle (\Delta E(\mathbf{x}) - \langle \Delta E(\mathbf{x}) \rangle)^2 \right\rangle = \left\langle (\Delta E(\mathbf{x}))^2 \right\rangle - (\langle \Delta E(\mathbf{x}) \rangle)^2$. The second moment has the form

$$\begin{aligned}
\left\langle (\Delta E(\mathbf{x}))^2 \right\rangle &= \left\langle \left(\mathbf{g}^T\boldsymbol{\epsilon} + \frac{1}{2}\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon} + \frac{1}{6}\sum_{i,j,k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k + \text{higher-order terms} \right)^2 \right\rangle \\
&= \left\langle \boldsymbol{\epsilon}^T\mathbf{g}\mathbf{g}^T\boldsymbol{\epsilon} \right\rangle + \frac{1}{4}\left\langle (\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon})^2 \right\rangle + \left\langle (\mathbf{g}^T\boldsymbol{\epsilon})(\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon}) \right\rangle + \left\langle \frac{1}{3}(\mathbf{g}^T\boldsymbol{\epsilon}) \sum_{i,j,k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k \right\rangle + \langle \text{higher-order terms} \rangle.
\end{aligned}$$

Again, the higher-order terms either vanish if they are odd, as is also the case for $\left\langle (\mathbf{g}^T\boldsymbol{\epsilon})(\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon}) \right\rangle$, or are assumed to be integrable, and contribute a term $\mathcal{O}(\sigma^6)$. The other three terms are as follows. For the first term we have:

$$\left\langle \boldsymbol{\epsilon}^T\mathbf{g}\mathbf{g}^T\boldsymbol{\epsilon} \right\rangle = \sigma^2 \operatorname{tr}\left(\mathbf{g}\mathbf{g}^T\right) = \sigma^2 \|\mathbf{g}\|^2.$$

For the second term:

$$\begin{aligned}
\frac{1}{4}\left\langle (\boldsymbol{\epsilon}^T\mathbf{H}\boldsymbol{\epsilon})^2 \right\rangle &= \frac{1}{4}\left\langle (\mathbf{u}^T\boldsymbol{\Lambda}\mathbf{u})^2 \right\rangle = \frac{1}{4}\left\langle \left(\sum_{i=1}^{n} \lambda_i u_i^2\right)^2 \right\rangle = \frac{1}{4}\left\langle \sum_{i,j=1}^{n} \lambda_i\lambda_j u_i^2 u_j^2 \right\rangle = \frac{1}{4}\sum_{i\neq j}^{n} \lambda_i\lambda_j \left\langle u_i^2 u_j^2 \right\rangle + \frac{1}{4}\sum_{i=1}^{n} \lambda_i \left\langle u_i^4 \right\rangle \\
&= \frac{1}{4}\sum_{i\neq j}^{n} \lambda_i\lambda_j \left\langle u_i^2 \right\rangle \left\langle u_j^2 \right\rangle + \frac{1}{4}\sum_{i=1}^{n} \lambda_i \left\langle u_i^4 \right\rangle = \frac{1}{4}\sigma^4 \left(\left(\sum_{i=1}^{n} \lambda_i\right)^2 + 2\sum_{i=1}^{n} \lambda_i^2 \right) = \frac{1}{4}\sigma^4 \left(\operatorname{tr}(\mathbf{H})^2 + 2\operatorname{tr}\left(\boldsymbol{\Lambda}^2\right) \right) \\
&= \frac{1}{4}\sigma^4 \left(\operatorname{tr}(\mathbf{H})^2 + 2\|\mathbf{H}\|_F^2 \right)
\end{aligned}$$

where we changed variables $\mathbf{u} = \mathbf{U}^T\boldsymbol{\epsilon}$ as above and used

$$\|\mathbf{H}\|_F^2 = \operatorname{tr}\left(\mathbf{H}^T\mathbf{H}\right) = \operatorname{tr}\left(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T\right) = \operatorname{tr}\left(\boldsymbol{\Lambda}^2\right). \tag{12}$$

For the fourth term, we have

$$\left\langle \frac{1}{3}(\mathbf{g}^T\boldsymbol{\epsilon}) \sum_{i,j,k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k \right\rangle = \frac{1}{3}\sum_{l=1}^{N} g_l \left\langle \epsilon_l \sum_{i,j,k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k \right\rangle.$$

The lth expectation can be simplified as follows:

$$
\begin{aligned}
\left\langle \epsilon_l \sum_{i,j,k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k \right\rangle &= \left\langle \epsilon_l \left(\sum_{j,k=1}^{n} t_{ljk}\epsilon_l\epsilon_j\epsilon_k + \sum_{i\neq l}^{n} \sum_{j,k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k \right) \right\rangle && (13)\\
&= \left\langle \epsilon_l \left(\sum_{j,k=1}^{n} t_{ljk}\epsilon_l\epsilon_j\epsilon_k + \sum_{i\neq l}^{n} \left[\left(\sum_{k=1}^{n} t_{ilk}\epsilon_i\epsilon_l\epsilon_k \right) + \left(\sum_{j\neq l}^{n} \sum_{k=1}^{n} t_{ijk}\epsilon_i\epsilon_j\epsilon_k \right) \right] \right) \right\rangle && (14)\\
&= \left\langle \epsilon_l \left(\sum_{j=1}^{n} t_{ljj}\epsilon_l\epsilon_j^2 + \sum_{i\neq l}^{n} \left(t_{ili}\epsilon_i^2\epsilon_l + t_{iil}\epsilon_i^2\epsilon_l \right) \right) \right\rangle && (15)\\
&= \left\langle \epsilon_l^2 \left(\sum_{i=1}^{n} t_{lii}\epsilon_i^2 + \sum_{i\neq l}^{n} (t_{ili} + t_{iil})\epsilon_i^2 \right) \right\rangle && (16)\\
&= \left\langle \epsilon_l^2 \left(t_{lll}\epsilon_l^2 + \sum_{i\neq l}^{n} (t_{lii} + t_{ili} + t_{iil})\epsilon_i^2 \right) \right\rangle && (17)\\
&= t_{lll} \left\langle \epsilon_l^4 \right\rangle + \left(\sum_{i\neq l}^{n} (t_{lii} + t_{ili} + t_{iil}) \right) \left\langle \epsilon_l^2 \right\rangle^2 && (18)\\
&= \sigma^4 \left(3t_{lll} + \sum_{i\neq l}^{n} (t_{lii} + t_{ili} + t_{iil}) \right) = \sigma^4 \sum_{i=1}^{n} (t_{lii} + t_{ili} + t_{iil}) && (19)\\
&= 3\sigma^4 \sum_{i=1}^{n} t_{iil} = 3\sigma^4 (\mathbf{1}^T \mathbf{J}_{\mathbf{H}})_l && (20)
\end{aligned}
$$

where in (13) we split the i summation into $i = l$ and $i \neq l$, in (14) we split the j summation into $j = l$ and $j \neq l$, in (15) the odd terms vanish (so in the first term we must have $k = j$, in the second $k = i$ and in the third $i = j \neq l$ and $k = l$), in (17) we split the first i summation into $i = l$ and $i \neq l$, and in (20) $t_{lii} = t_{ili} = t_{iil}$ holds because the third derivatives are continuous (so we can exchange the derivative order). Hence the fourth term equals $\sigma^4 \mathbf{1}^T \mathbf{J}_{\mathbf{H}} \mathbf{g}$, and $\left\langle (\Delta E(\mathbf{x}) - \langle \Delta E(\mathbf{x}) \rangle)^2 \right\rangle = \sigma^2 \|\mathbf{g}\|^2 + \frac{1}{2}\sigma^4 \|\mathbf{H}\|_F^2 + \sigma^4 \mathbf{1}^T \mathbf{J}_{\mathbf{H}} \mathbf{g}$.

The formula for $\delta E(\mathbf{x}) = \Delta E(\mathbf{x})/E(\mathbf{x})$ follows directly. To prove the bound (10), we apply a Taylor expansion to second order with remainder:

$$
\Delta E(\mathbf{x}) = E(\mathbf{x} + \boldsymbol{\epsilon}) - E(\mathbf{x}) = \mathbf{g}(\mathbf{x})^T \boldsymbol{\epsilon} + \frac{1}{2} \boldsymbol{\epsilon}^T \nabla^2 E(\mathbf{x} + \xi \boldsymbol{\epsilon}) \boldsymbol{\epsilon}
$$

where $\xi \in (0, 1)$ depends on $\boldsymbol{\epsilon}$. Proceeding as before:

$$
\begin{aligned}
\langle \Delta E(\mathbf{x}) \rangle &= (2\pi\sigma^2)^{-\frac{n}{2}} \int_{\mathbb{R}^n} \left(\frac{1}{2} \boldsymbol{\epsilon}^T \nabla^2 E(\mathbf{x} + \xi \boldsymbol{\epsilon}) \boldsymbol{\epsilon} \right) \exp\left(\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} / 2\sigma^2 \right) d\boldsymbol{\epsilon} \Rightarrow \\
|\langle \Delta E(\mathbf{x}) \rangle| &\leq (2\pi\sigma^2)^{-\frac{n}{2}} \frac{1}{2} \int_{\mathbb{R}^n} \left| \boldsymbol{\epsilon}^T \nabla^2 E(\mathbf{x} + \xi \boldsymbol{\epsilon}) \boldsymbol{\epsilon} \right| \exp\left(\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} / 2\sigma^2 \right) d\boldsymbol{\epsilon} \\
&\leq (2\pi\sigma^2)^{-\frac{n}{2}} \frac{1}{2} \int_{\mathbb{R}^n} M \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} \exp\left(\boldsymbol{\epsilon}^T \boldsymbol{\epsilon} / 2\sigma^2 \right) d\boldsymbol{\epsilon} = \frac{1}{2} \sigma^2 n M.
\end{aligned}
$$

□

Note that the mean error in eq. (8) depends on the point $\mathbf{x}$, i.e., the iterate where we apply the approximate step, through the trace of the Hessian at that point (and the accuracy level σ, which we assume fixed by the user). It does not depend on the gradient itself, because the linear term is an odd function that integrates to zero. The formula for the mean error is accurate when σ is small, which means the accuracy in the gradient evaluation is high. If the function E is quadratic, the formulas are exact. The bound for the mean error in eq. (10) is valid at any iterate (i.e., it does not depend on $\mathbf{x}$), but will typically be too coarse, and it also loses the information about the sign of the error.

The formula for the mean error in eq. (8) has a simple geometric interpretation (see fig. 1): while a Gaussian perturbation is symmetric in $\mathbf{x}$-space, the value of $E(\mathbf{x} + \boldsymbol{\epsilon})$ is not because of the curvature in E, so the average of the E-error is not zero.

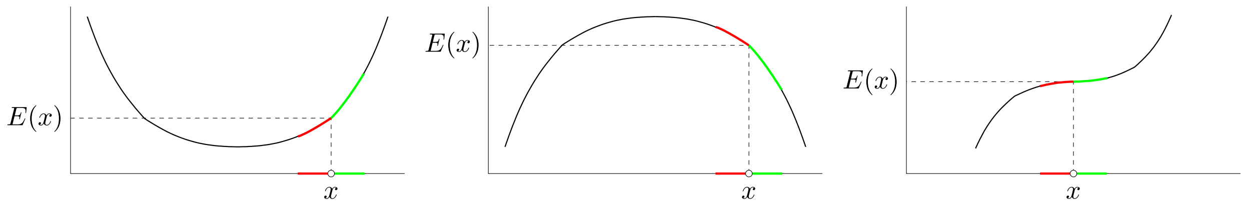

Figure 1: The result of a Gaussian perturbation to a point x on the function $E(x)$ in 1D. With positive mean curvature (left), the perturbation is equally likely to move x to the left or to the right, but points to the right have a larger, positive error in E, while points to the left have a smaller, negative error in E, and the net effect is that the perturbed E value is larger than $E(x)$ on average. With negative mean curvature (middle), the opposite is true. With zero mean curvature (right), the perturbed E value is zero to first order.

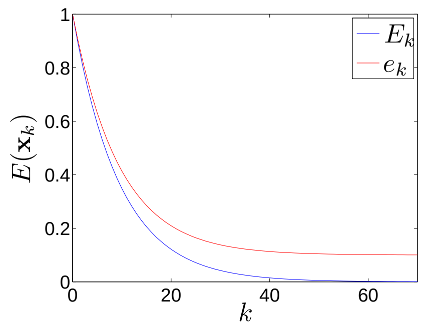

Figure 2: Illustration of the objective function sequences E_k and e_k for the optimization using exact and inexact gradients, respectively. We use $r = 0.9$, $\mu = 0.01$ and an initial $E_0 = e_0 = 1$. E_k converges to $E^* = 0$ while e_k converges to $e^* = 0.1$.

The behavior of the variance of the absolute error during the optimization can be characterized as follows. The variance is, to first order, proportional to the squared gradient, so we expect large variations in the error in early stages of the optimization. Near a minimizer, $\mathbf{g}(\mathbf{x}) \approx \mathbf{0}$ and so the coefficient of variation of the absolute error is

$$\frac{\sqrt{v_\Delta(\mathbf{x})}}{\mu_\Delta(\mathbf{x})} \approx \sqrt{2}\frac{\|\mathbf{H}(\mathbf{x})\|_F}{\operatorname{tr}(\mathbf{H}(\mathbf{x}))} = \sqrt{2}\frac{\|\boldsymbol{\lambda}(\mathbf{x})\|_2}{\|\boldsymbol{\lambda}(\mathbf{x})\|_1} \in \left[\sqrt{\frac{2}{n}}, \sqrt{2}\right]$$

since $\|\mathbf{x}\|_1/\sqrt{n} \leq \|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1$ $\forall \mathbf{x} \in \mathbb{R}^n$, $\operatorname{tr}(\mathbf{H}(\mathbf{x})) = \sum_{i=1}^n \lambda_i$ and, from eq. (12), $\|\mathbf{H}(\mathbf{x})\|_F^2 = \sum_{i=1}^n \lambda_i^2$, where $\boldsymbol{\lambda}(\mathbf{x}) = (\lambda_1, \dots, \lambda_n)^T \geq \mathbf{0}$ are the eigenvalues of $\mathbf{H}(\mathbf{x})$. Thus, the coefficient of variation of the absolute error is independent of the accuracy level σ and dependent only on the curvature. The ends of the interval above occur when all the eigenvalues are equal ($\sqrt{2/n}$) or at most one eigenvalue is nonzero ($\sqrt{2}$). In practice, if n is large we are likely to have many nonzero eigenvalues and thus be closer to the $\sqrt{2/n}$ end, so the coefficient of variation will be very small. Hence, near a minimizer we expect to see absolute errors with a near-constant value of $\mu_\Delta(\mathbf{x}) = \frac{1}{2}\sigma^2 \operatorname{tr}(\mathbf{H}(\mathbf{x}))$.

This allows us to characterize the behavior of the optimization near the minimizer. Assume that, if using exact gradients, we converge linearly with rate $0 < r < 1$, e.g. $E_{k+1} = rE_k$ where E_k is the exact value of the objective function $E(\mathbf{x}_k)$ at iterate k (and we assume w.l.o.g. that $E_k \to E^* = 0$). Using the approximate gradients, the sequence of objective function values is instead $e_{k+1} \approx re_k + \mu_\Delta(\mathbf{x}_k) \approx re_k + \mu$, where $\mu = \frac{1}{2}\sigma^2 \operatorname{tr}(\mathbf{H}(\mathbf{x}^*))$ and $\mathbf{x}^*$ is the minimizer. We assume a high enough accuracy $\sigma \ll 1$ so that $\mu \ll 1$ and convergence actually occurs. Then, we have that $e_k \to e^* = \frac{\mu}{1-r}$ linearly with rate r. Indeed, for the limit we have $e_{k+1} = e^* = re^* + \mu \Rightarrow e^* = \frac{\mu}{1-r}$. For the rate, we have:

$$\frac{|e_{k+1} - e^*|}{|e_k - e^*|} = \frac{re_k + \mu - \frac{\mu}{1-r}}{e_k - \frac{\mu}{1-r}} = r.$$

This means that, when using approximate gradients, the sequence of objective function values (e_k) will seem to converge, but will do so to a value e^* that is larger than the optimal one E^*, and proportional to σ^2. The iterates $\mathbf{x}_k$ will, of course, not converge but oscillate around $\mathbf{x}^*$. Fig. 2 illustrates this.

2.1 Application to optimization of embeddings

While the previous noise model is probably too simple to make quantitative predictions, it does give important qualitative predictions (always noting that σ must be small enough, i.e., the accuracy must not be too low): (1) adding noise will be beneficial only where the mean curvature $\frac{1}{n}\operatorname{tr}\left(\nabla^2 E(\mathbf{x})\right)$ is negative; (2) when the mean curvature is positive, the lower the accuracy the worse the optimization; (3) $\langle \Delta E(\mathbf{x})\rangle \,/\operatorname{tr}\left(\nabla^2 E(\mathbf{x})\right)$ should take an approximately constant value over iterates which is related to the accuracy level; and (4) $\Delta E(\mathbf{x})$ will vary widely at the beginning of the optimization and become approximately constant and equal to $\frac{1}{2}\sigma^2 \operatorname{tr}(\mathbf{H}(\mathbf{x}))$ near a minimizer. This in turn gives suggestions as to how to tune the accuracy (θ or p) during the optimization, as follows.

Let us assume that the optimization algorithm decreases the objective function, at least on average (this means the step sizes are sufficiently small and the accuracy sufficiently high), so that the optimization is effective. Thus, we expect that the early iterates will move through a region that may have negative or positive mean curvature (depending on the initialization and the objective function), but eventually they will move through a region of positive mean curvature, as they approach a minimizer. Thus, a higher accuracy will be necessary in the later stages of the optimization. As for the early stages, we can be more specific by looking at the Hessian trace for some embedding models, whose form can be obtained from Vladymyrov and Carreira-Perpiñán (2012):

- For the elastic embedding (EE) (Carreira-Perpiñán, 2010): $\operatorname{tr}\left(\nabla^2 E(\mathbf{x})\right) = 4d \operatorname{tr}(\mathbf{L})$, where $\mathbf{L}$ is the $N \times N$ graph Laplacian corresponding to the affinities in the input (high-dimensional) space and d is the dimension of the low-dimensional space.
- For s-SNE, t-SNE and other normalized models: $\operatorname{tr}\left(\nabla^2 E(\mathbf{x})\right) = 4d \operatorname{tr}(\mathbf{L}) - 16\lambda \left\|\mathbf{X}\mathbf{L}^q\right\|_F^2$, where $\mathbf{L}^q$ is a $N \times N$ graph Laplacian corresponding to the affinities learned in the low-dimensional space (see details in Vladymyrov and Carreira-Perpiñán, 2012). The usual s-SNE (Hinton and Roweis, 2003; Cook et al., 2007) and t-SNE (van der Maaten and Hinton, 2008) fix $\lambda = 1$.

For the graph Laplacian in the input space, we have $\operatorname{tr}(\mathbf{L}) = \sum_{n \neq m}^N w_{nm}$, which is a positive constant. Thus, the mean curvature is always positive for EE, so we do not expect the noise to help anywhere. For s-SNE and t-SNE, the mean curvature can be negative if $\left\|\mathbf{X}\mathbf{L}^q\right\|_F^2$ is large enough, but this will likely not happen if, as is commonly done, one initializes $\mathbf{X}$ from small values. In summary, it seems unlikely that the mean curvature will be negative during the optimization, and therefore the inexact steps caused by the Barnes-Hut or FMM methods will reduce the objective less than exact steps on average. However, it is likely that the mean curvature will become more positive as the optimization progresses, which suggests starting with a relatively low accuracy and increasing it progressively.

It still may make sense to try to benefit from the noise whenever the mean curvature does become negative. Since the Hessian trace for s-SNE and t-SNE can be computed in linear time in the number of parameters Nd in the embedding $\mathbf{X}$, one could detect when it is negative and use a very low accuracy in the gradient evaluations.

As noted above, if the accuracy is high enough, the objective function will decrease steadily and appear to converge, even though the iterates are actually oscillating, and the objective function value we converge to is larger than the optimal one (by a factor proportional to the accuracy). This again indicates that for convergence, at least in a practical sense where we have to stop the optimization after a finite number of iterations with a finite accuracy, as the optimization progresses the accuracy has to be increased as much as computationally feasible. Upon stopping, we should expect the objective function to exceed the optimal one at least by a factor that is proportional to the accuracy.

In theorem 2.1 we try to model, as generally as possible, how inexact gradients affect the convergence and the decrease of the objective function $E(\mathbf{x})$ in terms of local properties of E at the current iterate $\mathbf{x}$ (slope, curvature, etc.). The theorem tries to be as independent as possible of the particular approximation method (FMM, BH, etc.) and NLE (SNE, *t*-SNE, EE, etc.). In contrast, the FGT bound of Baxter and Roussos (2002) only applies to Gaussian sums with the FGT method and is *independent of the iterate* $\mathbf{x}$ (it only depends on the number of terms, dimension of latent space and box width). Hence, the FMM bound can be coarse, and does not distinguish between early and late stages of the optimization, so it does not help to design adaptive schedules for the accuracy level.

3 Embeddings using FGT and Barnes-Hut approximations

Fig. 3 shows the resulting embedding of 60 000 MNIST digits after one hour of optimization using L-BFGS. The results of FGT and BH look much better than the one using exact computation. Fig. 4 shows additional embeddings of the infiniteMNIST dataset using FGT.

References

B. J. C. Baxter and G. Roussos. A new error estimate of the fast Gauss transform. *SIAM J. Sci. Comput.*, 24 (1), 257–259 2002.

M. Á. Carreira-Perpiñán. The elastic embedding algorithm for dimensionality reduction. In J. Fürnkranz and T. Joachims, editors, *Proc. of the 27th Int. Conf. Machine Learning (ICML 2010)*, pages 167–174, Haifa, Israel, June 21–25 2010.

J. Cook, I. Sutskever, A. Mnih, and G. Hinton. Visualizing similarity data with a mixture of maps. In M. Meilă and X. Shen, editors, *Proc. of the 11th Int. Workshop on Artificial Intelligence and Statistics (AISTATS 2007)*, San Juan, Puerto Rico, Mar. 21–24 2007.

L. Greengard and J. Strain. The fast Gauss transform. *SIAM J. Sci. Stat. Comput.*, 12(1):79–94, Jan. 1991.

G. Hinton and S. T. Roweis. Stochastic neighbor embedding. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 857–864. MIT Press, Cambridge, MA, 2003.

L. J. P. van der Maaten and G. E. Hinton. Visualizing data using t-SNE. *J. Machine Learning Research*, 9: 2579–2605, Nov. 2008.

M. Vladymyrov and M. Á. Carreira-Perpiñán. Partial-Hessian strategies for fast learning of nonlinear embeddings. In J. Langford and J. Pineau, editors, *Proc. of the 29th Int. Conf. Machine Learning (ICML 2012)*, pages 345–352, Edinburgh, Scotland, June 26 – July 1 2012.

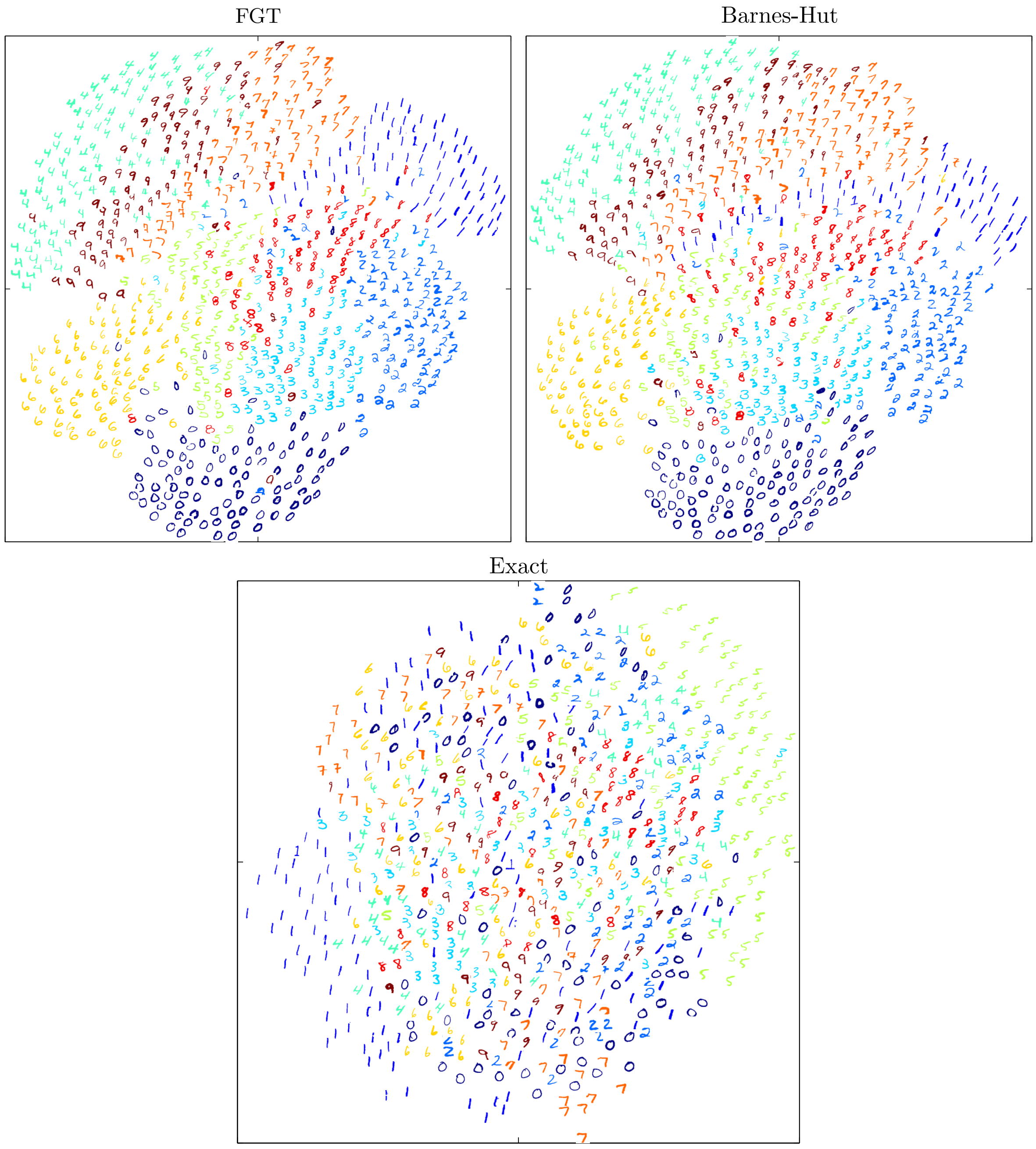

Figure 3: Embedding of 60 000 MNIST digits using FGT, BH and exact computation for L-BFGS optimization (only a subset is shown).

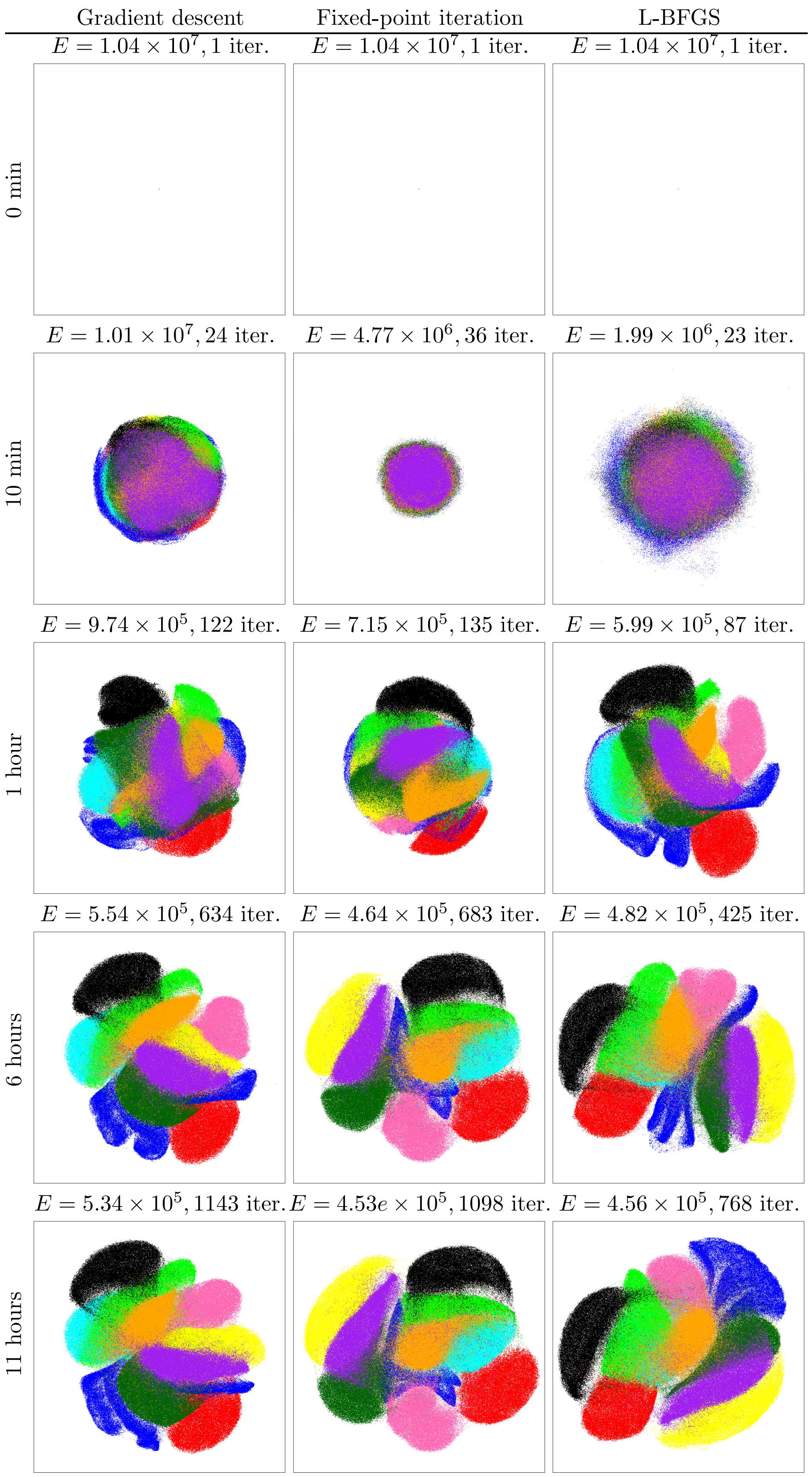

Figure 4: Embeddings of 1 020 000 digits from infinite MNIST dataset with FGT using GD, FP and L-BFGS. Shown are the resulting embedding in the beginning and after 10 min, 1 hour, 6 hours and 11 hours of optimization.