Fast algorithms for dimensionality reduction and data visualization

Manas Rachh

Yale University

Acknowledgements

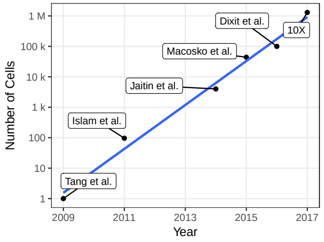
- George Linderman (Yale)
- Jeremy Hoskins (Yale)
- Stefan Steinerberger (Yale)
- Yuval Kluger (Yale)
- Vladimir Rokhlin (Yale)
- Mark Tygert (Facebook)

Introduction

- Applications
  - Single-cell RNA-sequencing (scRNA-seq)
  - Latent representations in deep learning
  - Astronomy
  - ...and much more
- t-SNE implementations scale poorly to large datasets (e.g. 8 hours for dataset of 1 million points in 500 dimensional space)
- FFT-accelerated Interpolation-based t-SNE (FIt-SNE), for faster t-SNE (30 min for same dataset)
- Out-of-Core PCA (oocPCA) for datasets that don't fit in memory

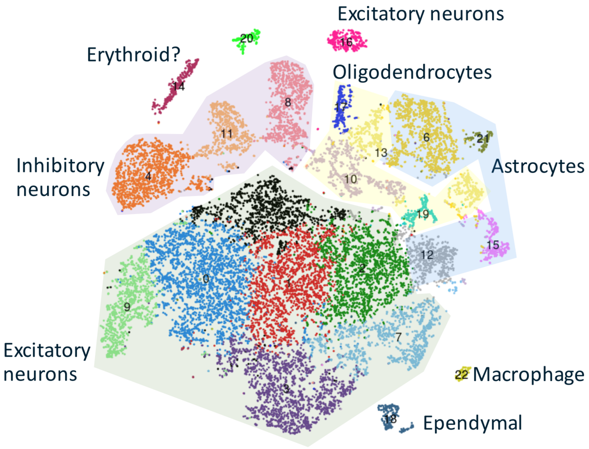
Applications: scRNA-seq

- Bulk RNA-seq averages expression across all cells
- Single cell RNA-seq measures expression in individual cells
- Results tabulated as an expression matrix
  - columns are genes (∼30,000)
  - rows are cells ($\sim 10^3$ to 10^6)
- Number of cells growing rapidly



Applications: scRNA-seq

For example, t-SNE of 1.3 million brain cells[1]



[1] 10X Genomics (2016)

t-SNE Optimization

- Input: d-dimensional dataset $X = \{x_1, x_2, \ldots, x_N\} \subset \mathbb{R}^d$
- Output: s-dimensional embedding $Y = \{y_1, y_2, \ldots, y_N\} \subset \mathbb{R}^s$, $s \ll d$
- Goal: x_i and x_j close in the input space $\implies$ y_i and y_j are also close
- Affinities between points x_i and x_j in the input space, p_{ij} - 'Gaussian'

$$p_{i|j} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)} \qquad \text{and} \qquad p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}.$$

- Affinities between points y_i and y_j - Cauchy kernel

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}.$$

- Minimize Kullback-Leibler divergence

$$C(\mathcal{Y}) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

Gradient Descent

- Minimize $C(\mathcal{Y})$ via gradient descent

$$\frac{\partial C}{\partial y_i} = 4Z \sum_{j \neq i} (p_{ij} - q_{ij}) q_{ij} (y_i - y_j)$$

- Z is a global normalization constant

$$Z = \sum_{\substack{j=1}}^{N} \sum_{\substack{\ell=1 \\ \ell \neq j}}^{N} \frac{1}{(1 + \|y_\ell - y_j\|^2)}$$

- Split into two parts

$$\frac{1}{4} \frac{\partial C}{\partial y_i} = \underbrace{Z \sum_{j \neq i} p_{ij} q_{ij} (y_i - y_j)}_{F_{\text{attr},i}} - \underbrace{Z \sum_{j \neq i} q_{ij}^2 (y_i - y_j)}_{F_{\text{rep},i}}$$

- Direct calculation: $O(N^2)$

Repulsion term - F_{rep}

$$\boxed{F_{\text{rep},k}(m) = \left(\sum_{\substack{\ell=1 \\ \ell \neq k}}^{N} \frac{y_\ell(m) - y_k(m)}{(1 + \|y_\ell - y_k\|^2)^2} \right) \Bigg/ \left(\sum_{\substack{j=1 \\ \ell \neq j}}^{N} \sum_{\ell=1}^{N} \frac{1}{(1 + \|y_\ell - y_j\|^2)} \right) ,}$$

- Combinations of

$$\sum_{j=1}^{N} K(y_i, y_j)\sigma_j$$

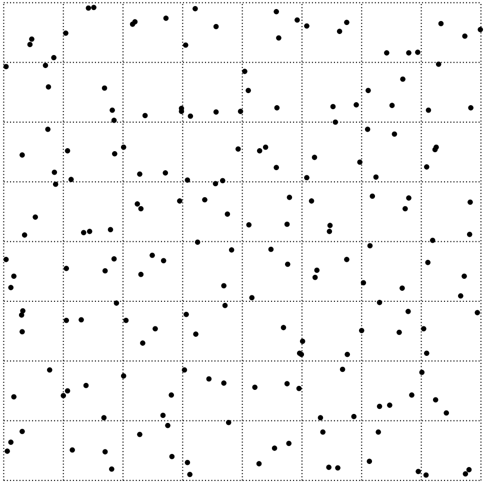
where

$$K(y,z) = \frac{1}{1 + \|y - z\|^2} \quad \text{or} \quad K(y,z) = \frac{1}{(1 + \|y - z\|^2)^2}$$

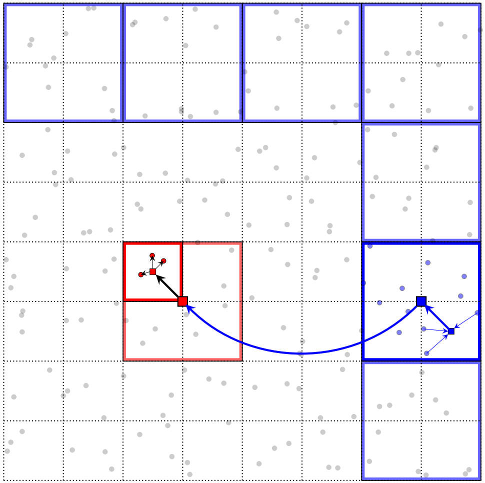
- Existing methods: Tree Codes/ Fast-multipole methods (FMMs)

L. Greengard, V. Rokhlin (1987)

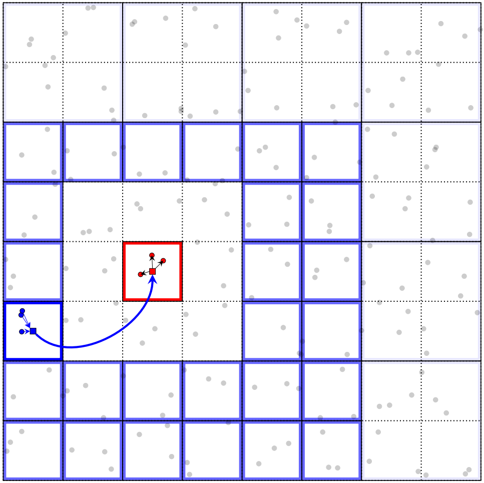
FMM illustration



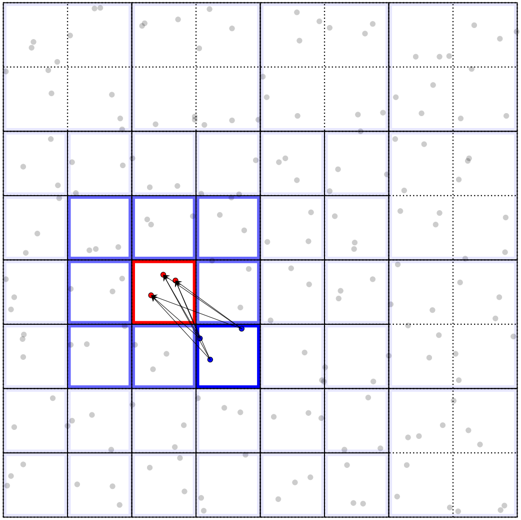
FMM illustration



FMM illustration



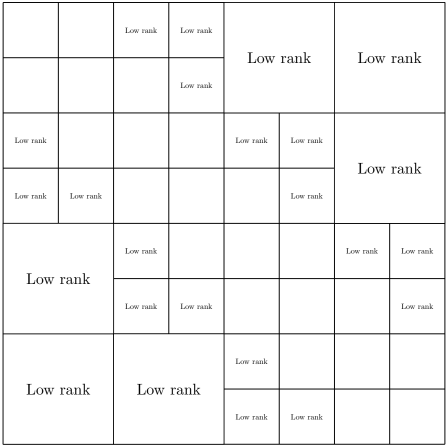
FMM illustration



FMM matrices

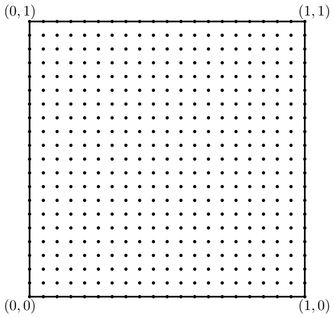
$$F_i = \sum_{j=1}^{N} K(y_i, z_j)\sigma_j$$

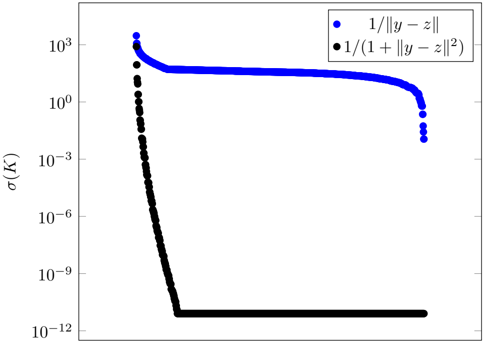
- $K(y, z)$ singular when $y = z$
- Self-interaction: full-rank
- Tree refinement strategy: $O(1)$ particles per leaf box



Self-interaction - smooth kernels

- t-SNE kernels - smooth even for $y = z$
- Even self-interaction can be compressed!





Polynomial interpolation based fast algorithms

- $K_p(y,z)$ - polynomial interpolant of $K(y,z)$ of order p
- $\tilde{y}_\ell$, $\tilde{z}_m$ - interpolation nodes, L - Lagrange polynomials; then

$$K_p(y,z) = \sum_{j=1}^{p^2} \sum_{\ell=1}^{p^2} K(\tilde{y}_\ell, \tilde{z}_m) L_{j,\tilde{y}}(y) L_{\ell,\tilde{z}}(z)$$

- Replace

$$\phi_\ell = \sum_{j=1}^{N} K(y_\ell, z_j)\sigma_j \quad \text{with} \quad \tilde{\phi}_\ell = \sum_{j=1}^{N} K_p(y_\ell, z_j)\sigma_j$$

- Relative error:

$$\frac{|\phi_\ell - \tilde{\phi}_\ell|}{|\phi_\ell|} \leq \sup |K(y,z) - K_p(y,z)|$$

- For fixed tolerance ε, p depends on smoothness of K, independent of N
- Greengard, Rokhlin, Gimbutas, Ying, Darve, Zorin, Biros, Barnett, Ho, Gillman, Martinsson,...

$\sum_j K_p(y_\ell, z_j)\sigma_j$

$$\tilde{\phi}_\ell = \sum_{j=1}^{N} \sum_{m=1}^{p^2} \sum_{n=1}^{p^2} K(\tilde{y}_m, \tilde{z}_n) L_{m,\tilde{y}}(y_\ell) L_{n,\tilde{z}}(z_j)\sigma_j$$
$$= \sum_{m=1}^{p^2} L_{m,\tilde{y}}(y_\ell) \left(\sum_{n=1}^{p^2} K(\tilde{y}_m, \tilde{z}_n) \left(\sum_{j=1}^{N} L_{n,\tilde{z}}(z_j)\sigma_j \right) \right)$$

- **Step 1**:

$$w_n = \sum_{j=1}^{N} L_{n,\tilde{z}}(z_j)\sigma_j \qquad \text{Work: } O(N \cdot p^2)$$

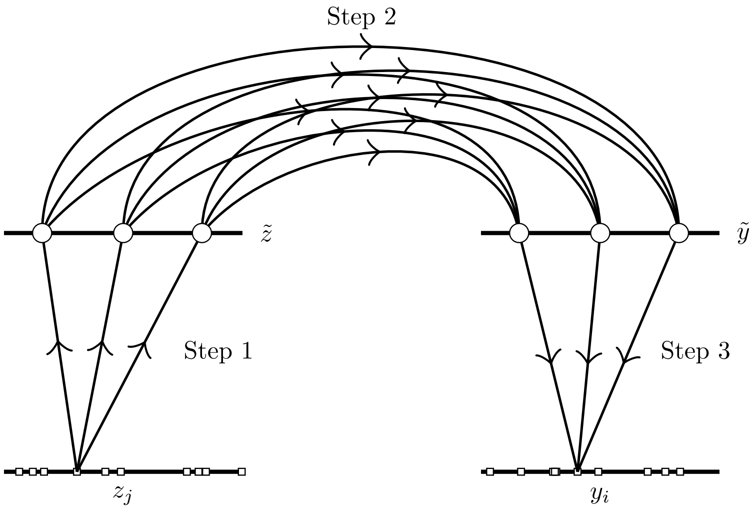
- **Step 2**:

$$v_m = \sum_{n=1}^{p^2} K(\tilde{y}_m, \tilde{z}_n) w_n \qquad \text{Work: } O(p^4)$$

- **Step 3**:

$$\tilde{\phi}_\ell = \sum_{m=1}^{p^2} L_{m,\tilde{y}}(y_\ell) v_m \qquad \text{Work: } O(M \cdot p^2)$$

Algorithm illustration



FFT accelerated interpolation based t-SNE (FIt-SNE)

- Subdivide domain into $N_{\text{int}} \times N_{\text{int}}$ boxes
- Given ε, determine p
- Equispaced interpolation nodes
- In each box, compute effective charges at interpolation nodes

$$w_{n,\ell} = \sum_{y_j \in B_\ell}^{N} L_{n,\tilde{y}^\ell}(y_j)\sigma_j \qquad \text{Work: } O(N \cdot p^2)$$

- Interaction between equispaced nodes - via FFT

$$v_{m,n} = \sum_{j=1}^{N_{\text{int}}^2} \sum_{\ell=1}^{p^2} K(\tilde{y}_{m,n}, \tilde{y}_{\ell,j}) w_{\ell,j} \qquad \text{Work: } O((N_{\text{int}} \cdot p)^2 \log (N_{\text{int}} \cdot p))$$

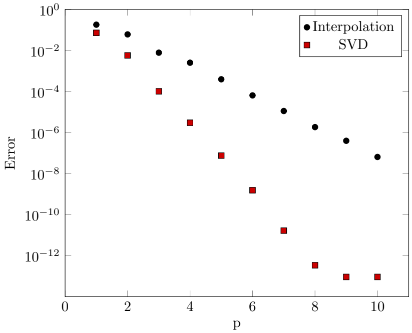
- Interpolate, for $y_i \in B_\ell$,

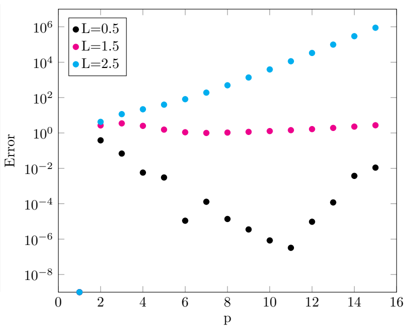
$$\tilde{\phi}_i = \sum_{j=1}^{p^2} L_{m,\tilde{y}^\ell}(y_i) v_{m,\ell} \qquad \text{Work: } O(N \cdot p^2)$$

Choosing N_{int} and p

- Large p with equispaced nodes - unstable
- t-SNE kernels archetypical examples of Runge phenomenon
- $L \leq 1.4$, $p < 10$ works
- For fixed accuracy, $N_{\text{int}} \cdot p$ constant $\implies$ Computational complexity $O(N \cdot p^2)$

Runge phenomenon and equispaced interpolation





Error estimates

1-D interpolation:

- $\tilde{x}_j = -L/2 + (j - 1/2) * L/p \quad j = 1, 2, \ldots p$
- $f(x) = 1/(1+x^2)$ or $f(x) = 1/(1+x^2)^2$
- Interpolation error:

$$\left| f(x) - \sum_{j=1}^{p} L_{j,\{\tilde{x}_j\}}(x) f(\tilde{x}_j) \right| \le \frac{f^p(\zeta)}{p!} \underbrace{\prod_{j=1}^{p} |x - \tilde{x}_j|}_{\pi_p(x)}$$

- Estimates:

$$|f^p| \le \frac{p+2}{2} p! \quad |\pi_p(x)| \le \frac{(2p)!}{2^{2p} p!} \left(\frac{L}{p}\right)^p$$

- Error in 1-D

$$\left| f(x) - \sum_{j=1}^{p} L_{j,\{\tilde{x}_j\}}(x) f(\tilde{x}_j) \right| \le \frac{p+2}{\sqrt{2}} \left(\frac{L}{e}\right)^p e^{\frac{1}{24p}} .$$

Error estimates - II

- In $d-$dimensions

$$f(\boldsymbol{x}) = 1/(1+|\boldsymbol{x}|^2) \quad \text{or} \quad f(x) = 1/(1+|\boldsymbol{x}|^2)^2$$

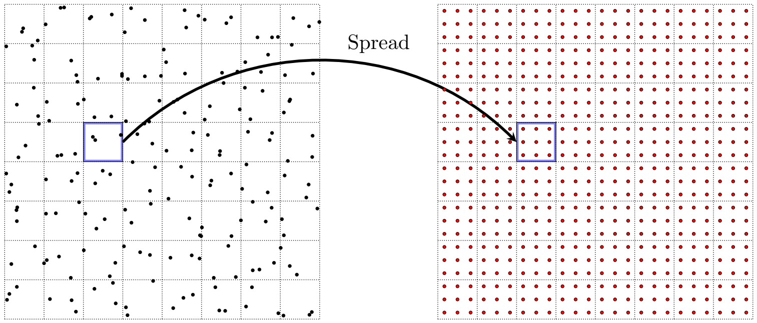
- In $d-$dimensional interpolation, estimates via error estimates along lines
- Interpolation error:

$$\left| f(\boldsymbol{x}) - \sum_{j=1}^{p} L_{j,\{\tilde{x}_j\}}(\boldsymbol{x}) f(\tilde{x}_j) \right| \leq \frac{p+2}{\sqrt{2}} \left(\frac{2^d L}{e} \right)^p e^{\frac{1}{24p}} .$$

- Not sharp for $d > 1$

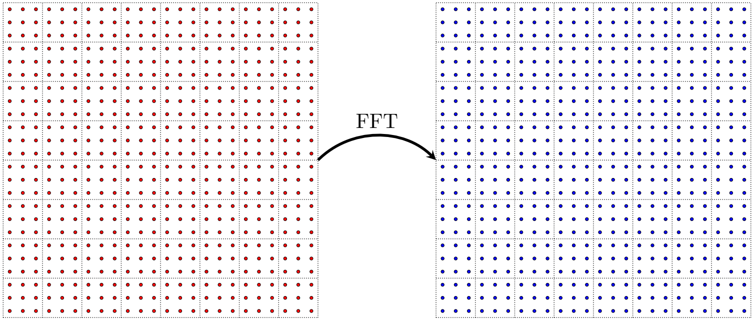
Algorithm Illustration - Step 1

$$\sum_{m=1}^{p^2} L_{m,\tilde{y}}(y_\ell)\left(\sum_{n=1}^{p^2} K(\tilde{y}_m, \tilde{z}_n)\underbrace{\left(\sum_{j=1}^{N} L_{n,\tilde{z}}(z_j)\sigma_j\right)}_{w_n}\right)$$



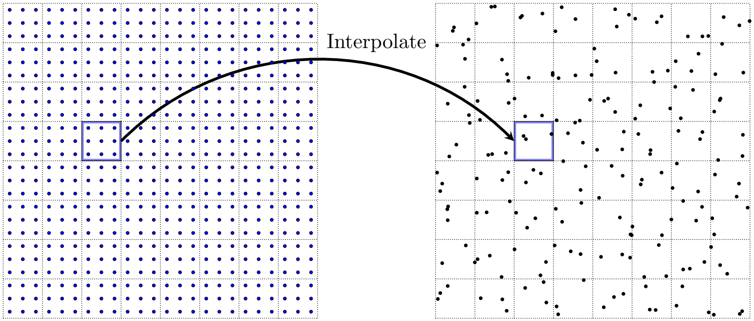
Algorithm Illustration - Step 2

$$\sum_{m=1}^{p^2} L_{m,\tilde{y}}(y_\ell) \underbrace{\left(\sum_{n=1}^{p^2} K(\tilde{y}_m, \tilde{z}_n) w_n \right)}_{v_m}$$



Algorithm Illustration - Step 3

$$\sum_{m=1}^{p^2} L_{m,\tilde{y}}(y_\ell) v_m$$



Matrix decomposition

- Matrix K block separable, all submatrices low rank
- $K_{i,j} = U_i S_{i,j} U_j^T$

$$\underbrace{\begin{bmatrix} U_1 & 0 & \dots & 0 \\ 0 & U_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{N_{\text{int}}^2} \end{bmatrix}}_{U} \cdot \underbrace{\begin{bmatrix} S_{1,1} & S_{1,2} & \dots & S_{1,N_{\text{int}}^2} \\ S_{2,1} & S_{2,2} & \dots & S_{2,N_{\text{int}}^2} \\ \vdots & \vdots & \ddots & \vdots \\ S_{N_{\text{int}}^2,1} & S_{N_{\text{int}}^2,2} & \dots & S_{N_{\text{int}}^2,N_{\text{int}}^2} \end{bmatrix}}_{S} \cdot \begin{bmatrix} U_1^T & 0 & \dots & 0 \\ 0 & U_2^T & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & U_{N_{\text{int}}^2}^T \end{bmatrix}$$

- $U_i : n_i \times p^2$ matrix
- S - Toeplitz (almost)

Attractive forces - F_{attr}

$$p_{ij} \sim \exp\left(-\|x_i - x_j\|^2/\sigma\right).$$

- Computing p_{ij} - a local calculation
- Attractive forces

$$F_{\text{attr},i} = \sum_{j \neq i} p_{ij} q_{ij} Z(y_i - y_j) \approx \sum_{j \in \text{ KNN of } i} p_{ij} q_{ij} Z(y_i - y_j).$$

- One time computation - doesn't need to be computed every iteration of gradient descent

Nearest Neighbors

- bhtsne: exact nearest neighbors
  - Using vantage point-trees
  - Slows down in high dimensions
- FIt-SNE: approximate nearest neighbors
  - Using ANNOY[2]
  - Random projections
  - Smoothing effect[3] from using near neighbors?

[2] https://github.com/spotify/annoy

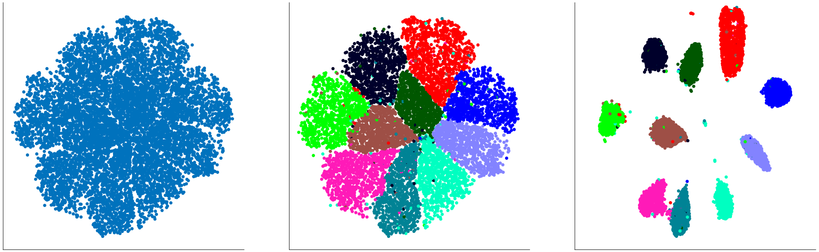
[3] G. Linderman and S. Steinerberger (2017) arXiv:1711.04712.

oocPCA for Big Data

- What if dataset is extremely large?
- Computers without enough memory to load data cannot visualize it
  - e.g. 1 million cells with $30,000$ genes requires 240GB!
- Out-of-core implementation of randomized PCA
  - Compute the top few ($\sim$ 50) principal components of a dataset without loading it entirely
- Mundane computers can visualize/analyze the largest datasets
- oocPCA computes top 50 principal components with varying memory limitations:

Memory (GB)	1	2	8	32	128	300
Time (Min)	15.9	12.8	12.7	12.0	10.5	8.4

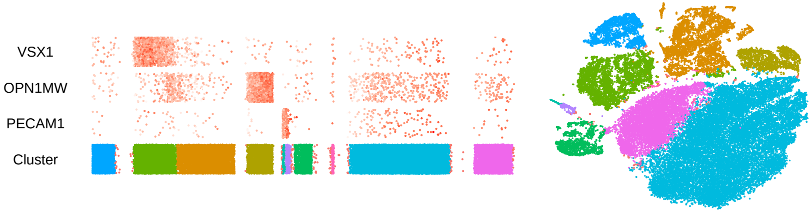
MNIST data



- 10^6 digit images from Infinite MNIST data-set
- Late exaggeration to separate clusters more effectively

G. Lindermann and S. Steinerberger (2017)

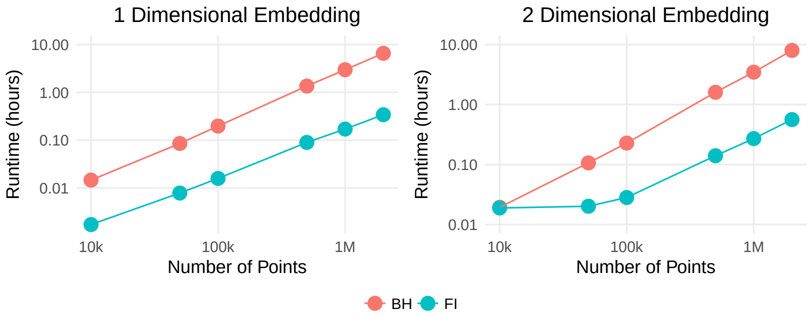
Retinal cells and t-SNE heatmaps



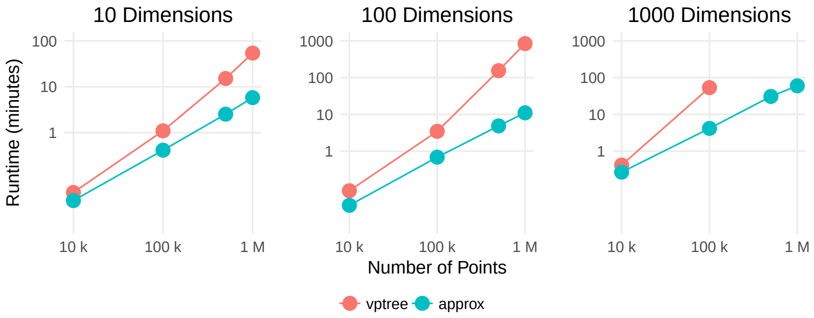
1D t-SNE heatmaps (left) vs 2D t-SNE (right) for 4.9×10^4 retinal cells

Data from Macosko et al (2016)

Numerical results - FIt-SNE



Numerical results - Fast nearest neighbors



Summary

- We developed fast algorithms for data visualization and dimensionality reduction using t-SNE which is roughly 15 times faster than the state of the art
- We presented interpolation based fast algorithms for $N - body$ interactions with smooth kernels
- Late exaggeration for better separation of clusters
- Out of core PCA for visualizing extremely large data sets on laptops
- Github: `https://github.com/KlugerLab/FIt-SNE`

Future work

- Better convergence estimates and theoretical framework
- Different affinities for input and target spaces
- Fast multipole style multi-level schemes

- Questions?

- Questions?

Thank you