universität wien

Approximative Linear t-SNE Using k-Means

Sonja Biedermann, BSc

April 20, 2021

Fakultät für Informatik
Universität Wien

Agenda

1. Introduction
2. t-SNE And Its Variants
3. Approximative Linear t-SNE
4. Evaluation
5. Conclusion

Introduction

Introduction

t-SNE And Its Variants

SNE

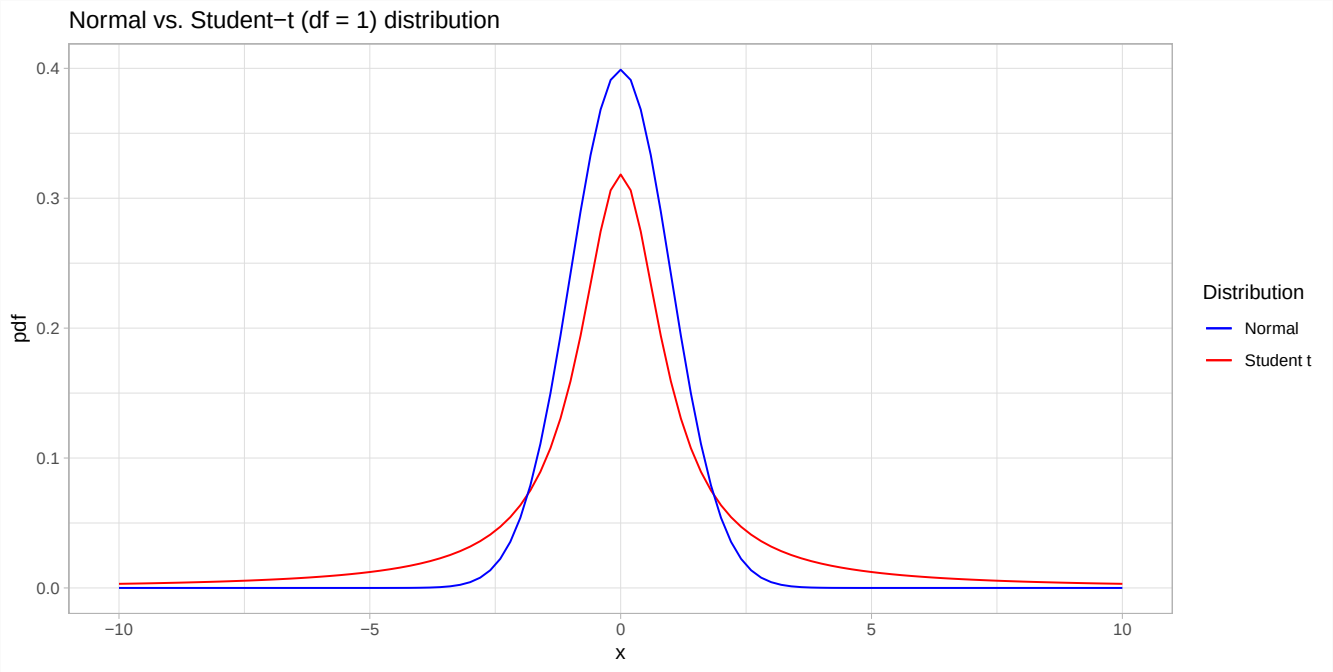
- Stochastic Neighborhood Embedding: two probability distributions
  - high dimensional: $p_{j|i}$, probability that i picks j as its neighbor
  - low dimensional: $q_{j|i}$
- optimization problem
- objective: Kullback-Leibler divergence

$$KL(P \,||\, Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- intuition: neighborhood encoded as distribution
- find low-dimensional points whose neighborhood has similar encoding
- optimized by gradient descent — difficult, not convex, many local minima

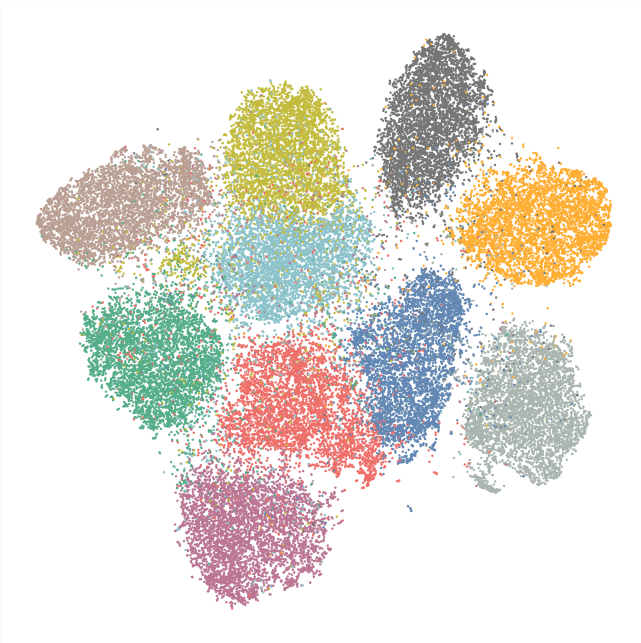
t-SNE

- qualitative issue of SNE: crowding problem
  - lower dimensional space has much less volume
  - many points moderately far away, not enough space
- solution: fat-tailed distribution in low-dimensional space



t-SNE

- popular due to visual properties
- problems:
  - difficult to optimize objective
  - computational complexity $\mathcal{O}(n^2)$
  - often misinterpreted



Related Work (Selection)

Barnes-Hut t-SNE

- use trees to
  - speed up nearest neighbor search
  - summarize points to speed up computation
- computational complexity $\mathcal{O}(n \log n)$
- introduces quality-speed tradeoff parameter θ

Related Work (Selection)

FIt-SNE

- interpolates onto equispaced grid
- uses FFT to compute convolution
- uses random tree based library[1] to find ANNs
- computational complexity $\mathcal{O}(n)$

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

Related Work (Selection)

UMAP

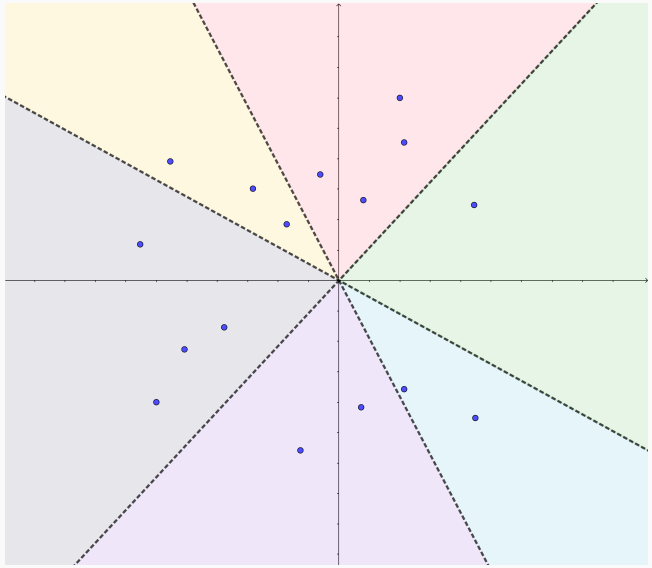
- another common criticism of t-SNE is the lack of theoretical foundation
- UMAP borrows from topological analysis and manifold theory to create theoretical foundation
- similar to t-SNE, but
  - objective is cross entropy of two fuzzy set
  - different kernel in high-dimensional and low-dimensional space
  - different normalization
- objective allows usage of negative sampling
- empirical complexity of $\mathcal{O}(n^{1.14})$ [2]

[2] due to NNDescent, otherwise $\mathcal{O}(n)$

Approximative Linear t-SNE

Using Approximative Nearest Neighbors: LSH

- ANN scheme of choice: Locality Sensitive Hashing
- LSH family: Cross Polytope
- LSH intuition: similar items should collide into the same hash bucket
- widely used in similarity search



Using k-Means

- summarize points by using k-Means to segment low-dimensional space
- fixed k (in contrast to data dependent in Barnes-Hut t-SNE)
- clustering need not be good—run for 10 iterations
- cluster centers (centroids) are used to summarize entire cluster

Using k-Means

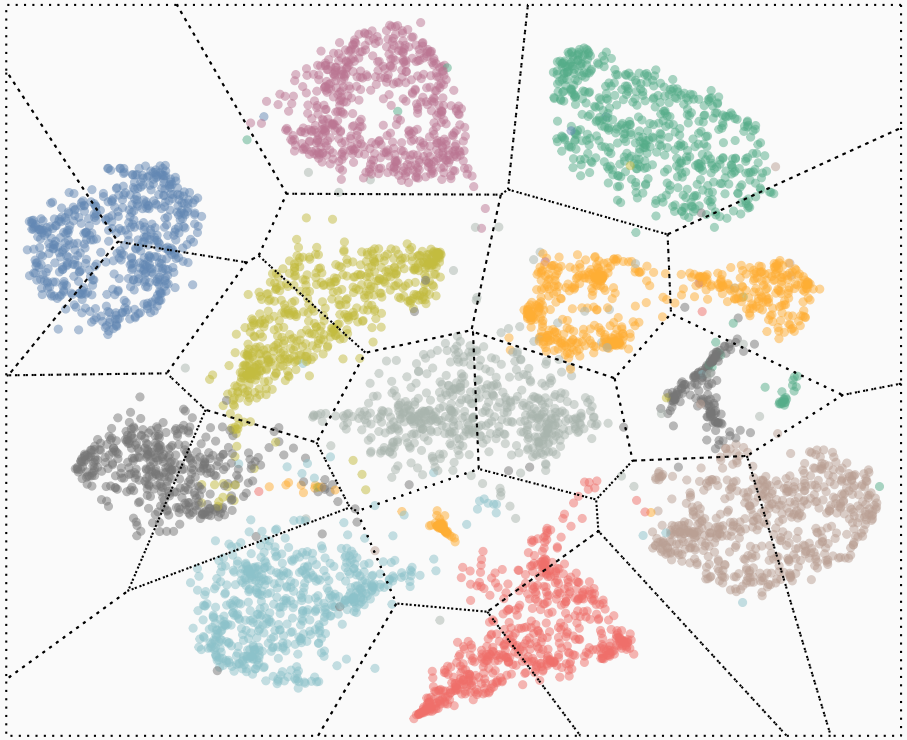


Figure 1: Voronoi diagram implied by k-Means clustering

Putting it all together: t-SNE's gradient

$$\begin{aligned}\frac{\partial KL(P||Q)}{\partial y_i} &= \sum_{j \neq i} p_{ij} q_{ij} Z(y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z(y_i - y_j) \\ &= F_{\text{attr}} - F_{\text{rep}}\end{aligned}$$

where $Z = \sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}$

- speeding up F_{attr} is simple
  - use only k nearest neighbors $\rightarrow$ P becomes sparse
  - parts of q_{ij} cancel out Z
  - computation is constant

Putting it all together: t-SNE's gradient

$$F_{\text{rep}} = \sum_{j \neq i} -q_{ij}^2 Z(y_i - y_j)$$

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

- more difficult situation
- trick: approximate $F_{\text{rep}} Z = \sum_{j \neq i} -(q_{ij} Z)^2 (y_i - y_j)$
- summarize points by cluster centroids $\bar{y}_c,\ c = 1 \ldots k$

Putting it all together: t-SNE's gradient

Final result

$$F_{\text{rep}}Z = \sum_{i=1}^{n}\sum_{c=1}^{k} -\frac{y_i - \bar{y}_c}{1 + ||y_i - \bar{y}_c||^2} \cdot n_c$$

$$\hat{Z} = \sum_{i=1}^{n}\sum_{c=1}^{k} \frac{n_c}{1 + ||y_i - \bar{y}_c||^2}$$

where n_c is the number of points in a cluster.

Evaluation

Experimental Results: Nearest Neighborhood Purity

data	bhtsne	fitsne	umap	ktsne
emailEuCore	0.3127	0.3160	**0.3301**	0.0722
coil-20	0.5906	0.5641	**0.5920**	0.4922
cora	0.6331	0.6283	0.6232	**0.6404**
citeseer	0.4217	0.4262	0.4280	**0.4297**
optdigits	0.9677	0.9656	**0.9736**	0.9585
youtube*	0.7978	0.8011	**0.8124**	0.7946
dblp	0.3393	**0.3401**	0.3370	0.3100
pubmed	**0.7007**	0.6985	0.6986	0.6948
cifar	0.8784	0.8773	**0.8801**	0.8519
mnist	0.9536	**0.9539**	0.9521	0.9380
fashion_mnist	0.7417	**0.7426**	0.7034	0.6889
com-dblp*	0.5303	**0.5907**	0.5855	0.4580
com-amazon*	0.6982	0.7449	**0.7606**	0.6047
svhn	0.1236	0.1238	**0.1240**	0.1234
hollywood*	0.6695	0.7894	**0.8172**	0.7844
timit	0.3125	0.3445	**0.3641**	0.3481
wiki-topcats*	0.5060	0.7477	**0.8939**	0.7968

Table 1: Average nearest neighborhood purity, $k = 100$. Best in bold.

Experimental Results: Runtime

data	bhtsne	fitsne	umap	ktsne
emailEuCore	6.02	36.28	22.5	**4.54**
coil-20	9.74	52.67	24.16	**6.9**
cora	27.26	52.1	34.54	**12.04**
citeseer	44.52	56.02	39.49	**14.39**
optdigits	68.22	52.12	57.48	**27.46**
youtube	220.16	**50.52**	101.55	66.7
dblp	402.66	**53.22**	131.77	86.16
pubmed	478.61	**58.06**	142.53	100.36
cifar	1507.18	**70.11**	362.86	333.41
mnist	2284.13	**80.76**	512.21	466.5
fashion_mnist	2003.8	**80.63**	534.86	515.43
com-dblp	25277.6	**345.12**	2336.02	4517.64
com-amazon	27031.37	**360.54**	2486.53	5450.24
svhn	25749.11	**656.79**	6191.86	10212.89
hollywood	123778.42	**964.93**	16137.21	11624.3
timit	190110.22	**1146.17**	11170.68	13598.08
wiki-topcats	256839.2	**1421.12**	17196.37	22242.01

Table 2: Average runtime in seconds. Best average in bold.

Experimental Results: Runtime on synthetic data

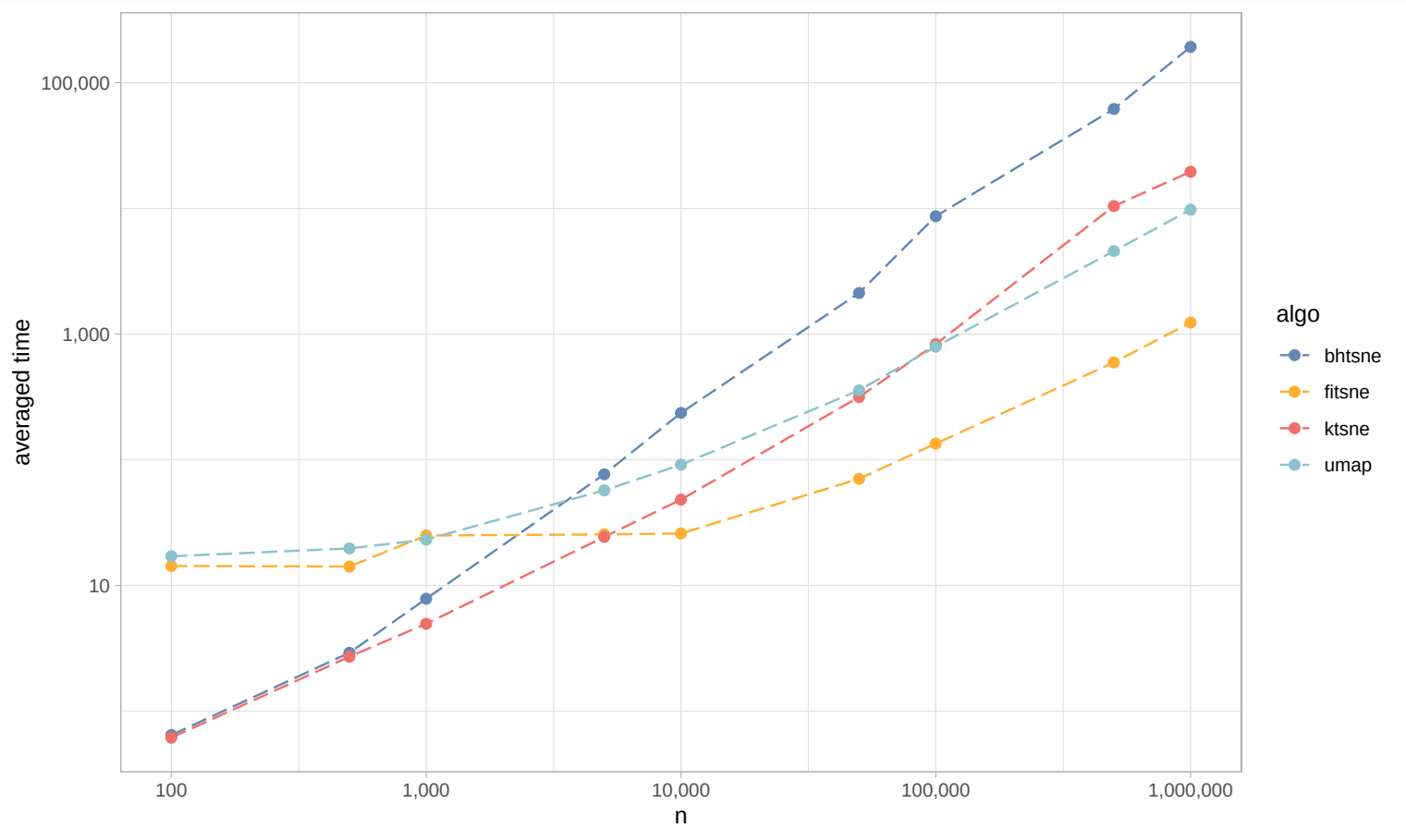


Figure 2: Average runtime in seconds on synthetic data

Conclusion

Thanks for your attention!

Any questions?