Fast interpolation-based *t*-SNE for data visualization

Min Dai

May 20th, 2021

daimin@zju.edu.cn

Institute of Genetics and Developmental Biology, Chinese Academy of Sciences

nature methods

Brief Communication | Published: 11 February 2019

Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data

George C. Linderman, Manas Rachh, Jeremy G. Hoskins, Stefan Steinerberger & Yuval Kluger

Nature Methods **16**, 243–245 (2019) | Cite this article

Corresponding author

Prof. Yuval Kluger, PhD

- Professor of Pathology
- Yale University

Research interest:

- bioinformatics
- machine learning
- applied mathematics
- dynamics of quantum fields

Yale University
School of Medicine

https://medicine.yale.edu/bbs/computational/profile/yuval_kluger/

Scientific question

How to reduce the complexity of the *t*-SNE algorithm?

Outline

- Background
- Algorithm
- Summary
- Discussion

What is *t*-SNE?

(three slides from Aug 22 2019 talk)

Journal of Machine Learning Research 1 (2008) 1-48

Published: 2008

Visualizing Data using t-SNE

Geoffrey E. Hinton

Laurens van der Maaten L.VANDERMAATEN@MICC.UNIMAAS.NL
MICC-IKAT
Maastricht University
P.O. Box 616, 6200 MD Maastricht, The Netherlands

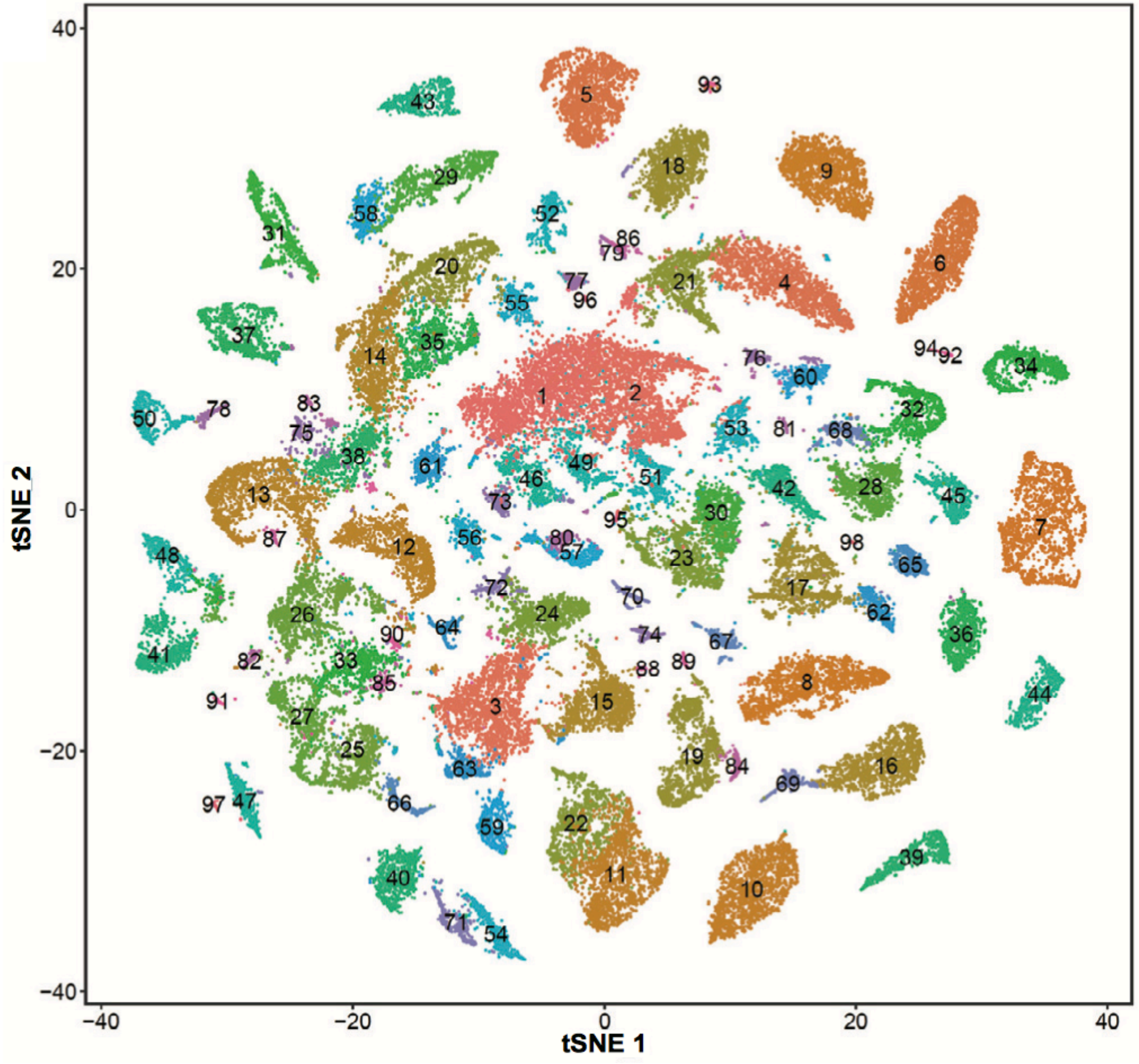
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

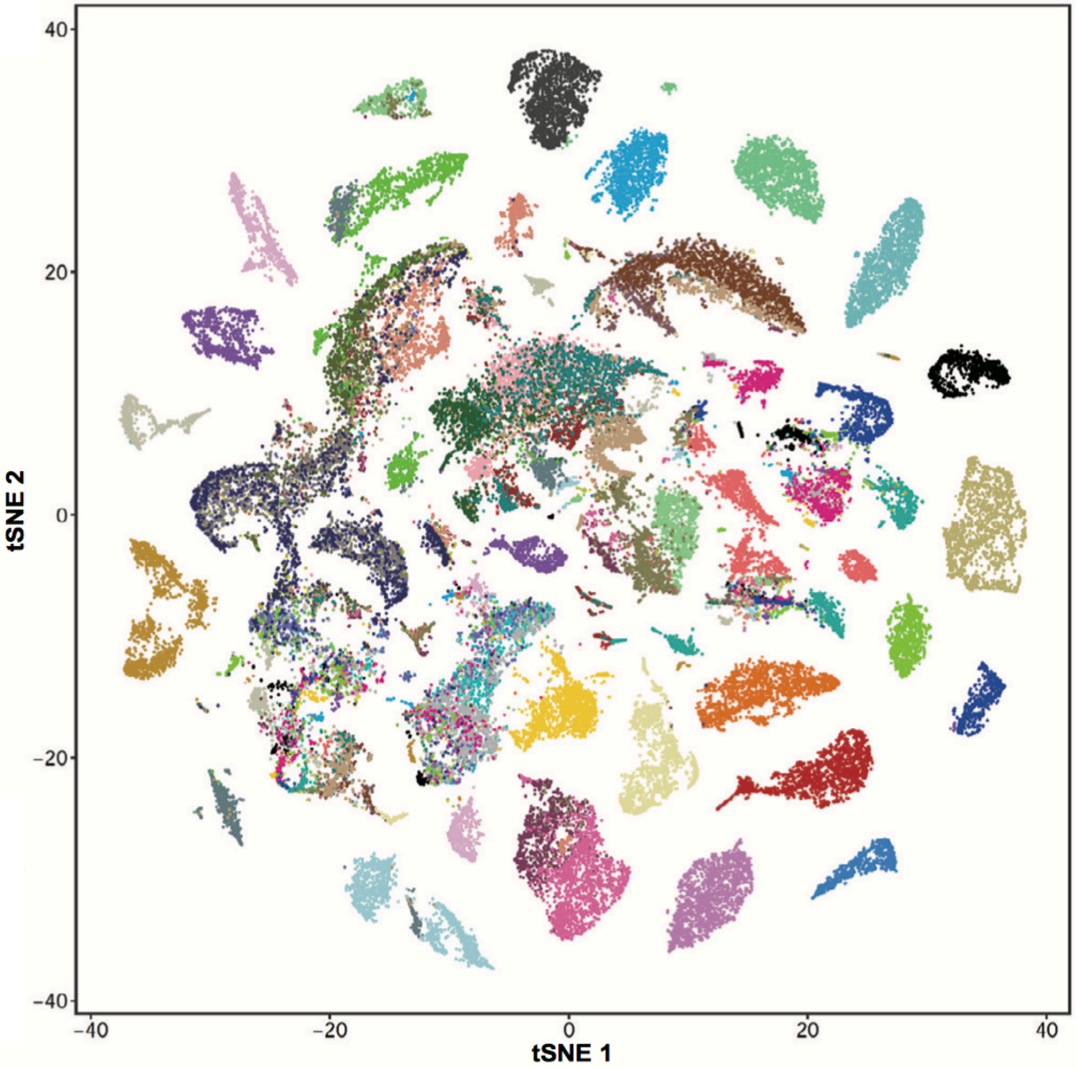
https://awards.acm.org/about/2018-turing

http://www.cs.toronto.edu/~hinton/absps/tsne.pdf

t-SNE has been widely used in biomedical research

- t-SNE analysis of 60,000 single cells sampled from the Mouse Cell Atlas

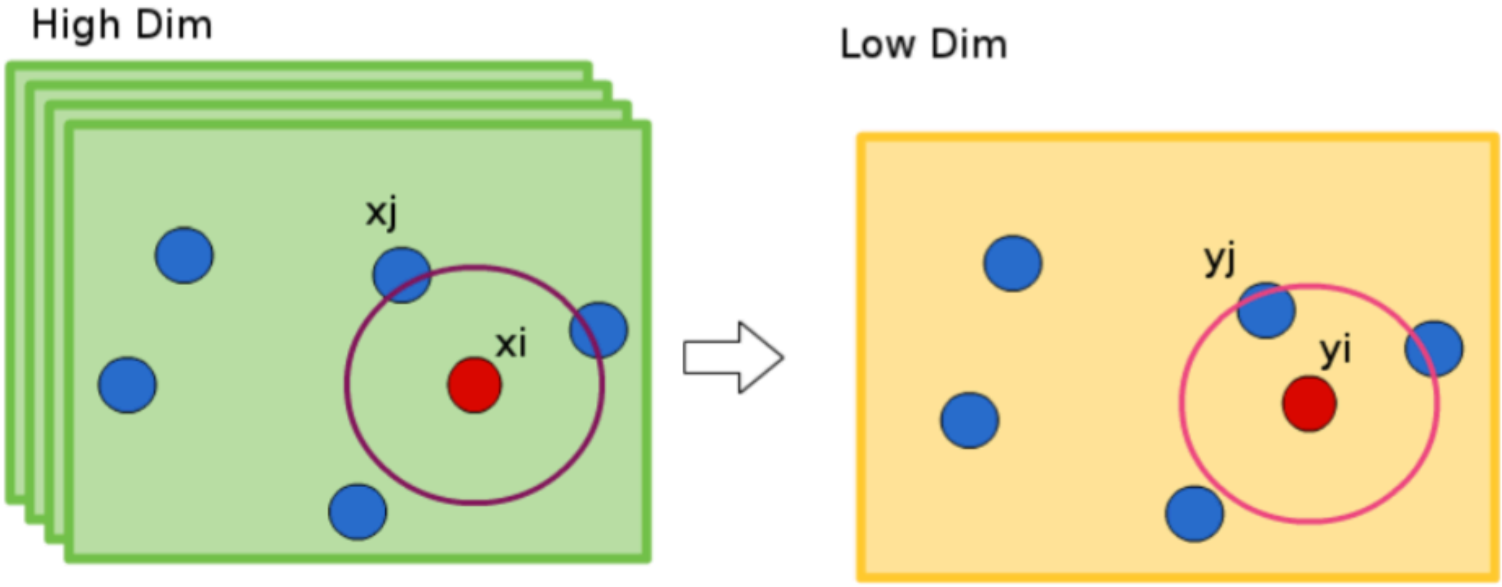




Han et al., Cell, 2018

t-SNE preserves the local structure of the high-dimensional data

- Measure pairwise similarities between high-dimensional and low-dimensional objects

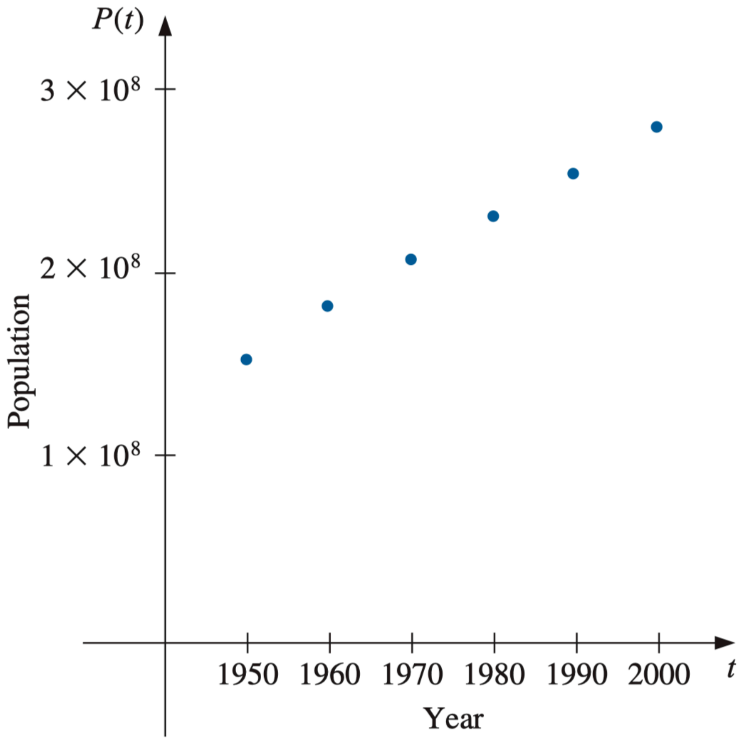


https://www.slideshare.net/ssuserb667a8/visualization-data-using-tsne

What is interpolation?

Background

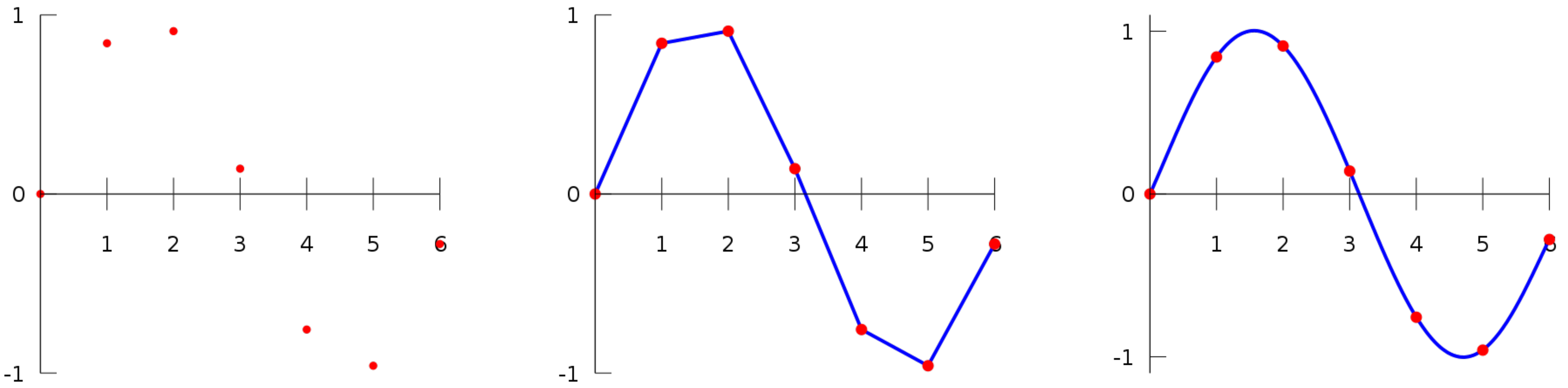
- How to provide a reasonable estimate of the population in 1975?



Year	1950	1960	1970	1980	1990	2000
Population (in thousands)	151,326	179,323	203,302	226,542	249,633	281,422

Interpolation

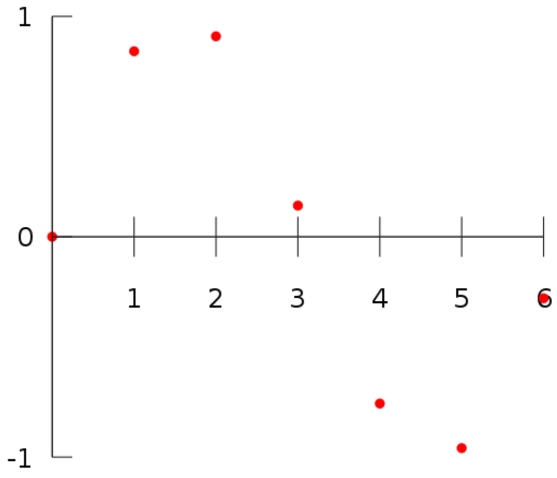
- Interpolation is a type of estimation, a method of constructing new data points within the range of a discrete set of known data points.
- Given a number of data points, obtained by sampling or experimentation, which represent the values of a function for a limited number of values of the independent variable.
    - It is often required to interpolate, i.e., estimate the value of that function for an intermediate value of the independent variable.

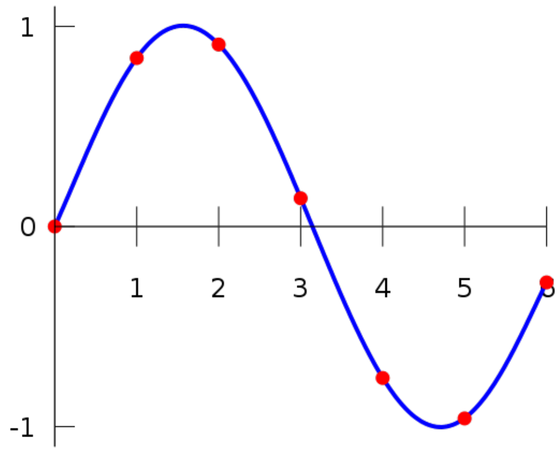


Wikipedia

Interpolation

- So we have $y_i = f(x_i)$ at $n + 1$ points $x_o, x_1, \ldots, x_i, \ldots, x_n$ and $x_j > x_{j-1}$
  - (often but not always evenly spaced)
- In general, we do not know the underlying function $f(x)$
- Conceptually, interpolation consists of two stages:
  - Develop a simple function $P(x)$ that
    - Approximates $f(x)$
    - Passes through all the points x_i
  - Evaluate $f(x_t)$ where $x_0 < x_t < x_n$

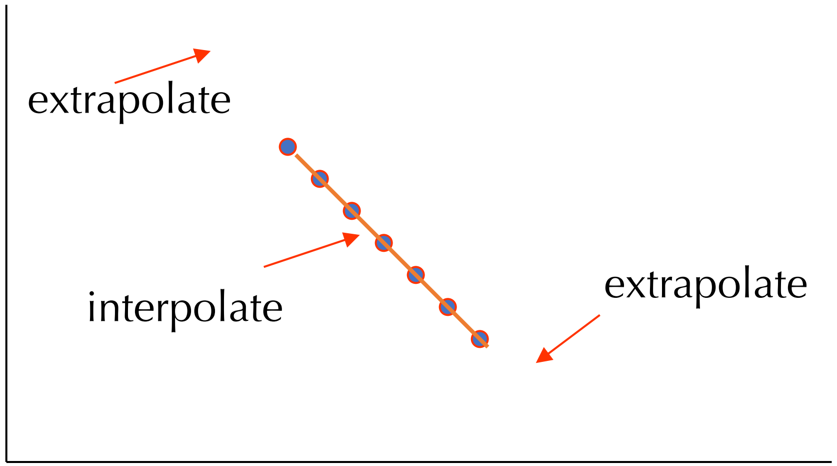




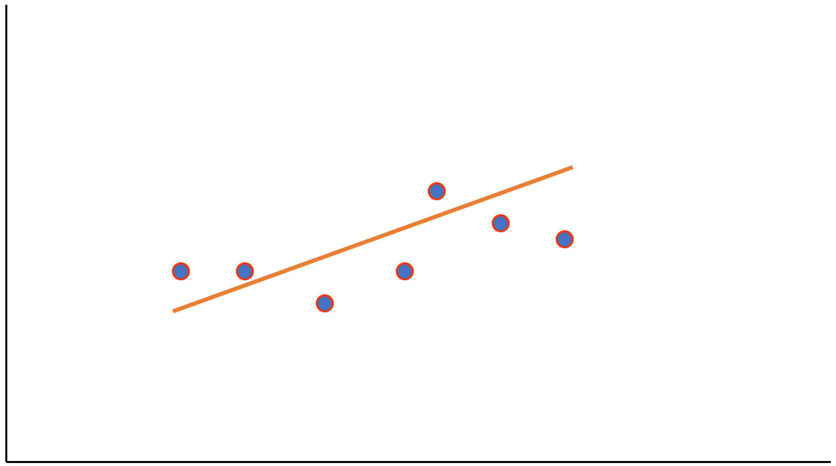
Wikipedia
Roger Crawfis

Interpolation vs. Regression

- Different approaches depending on the quality of the data



Interpolation



Regression

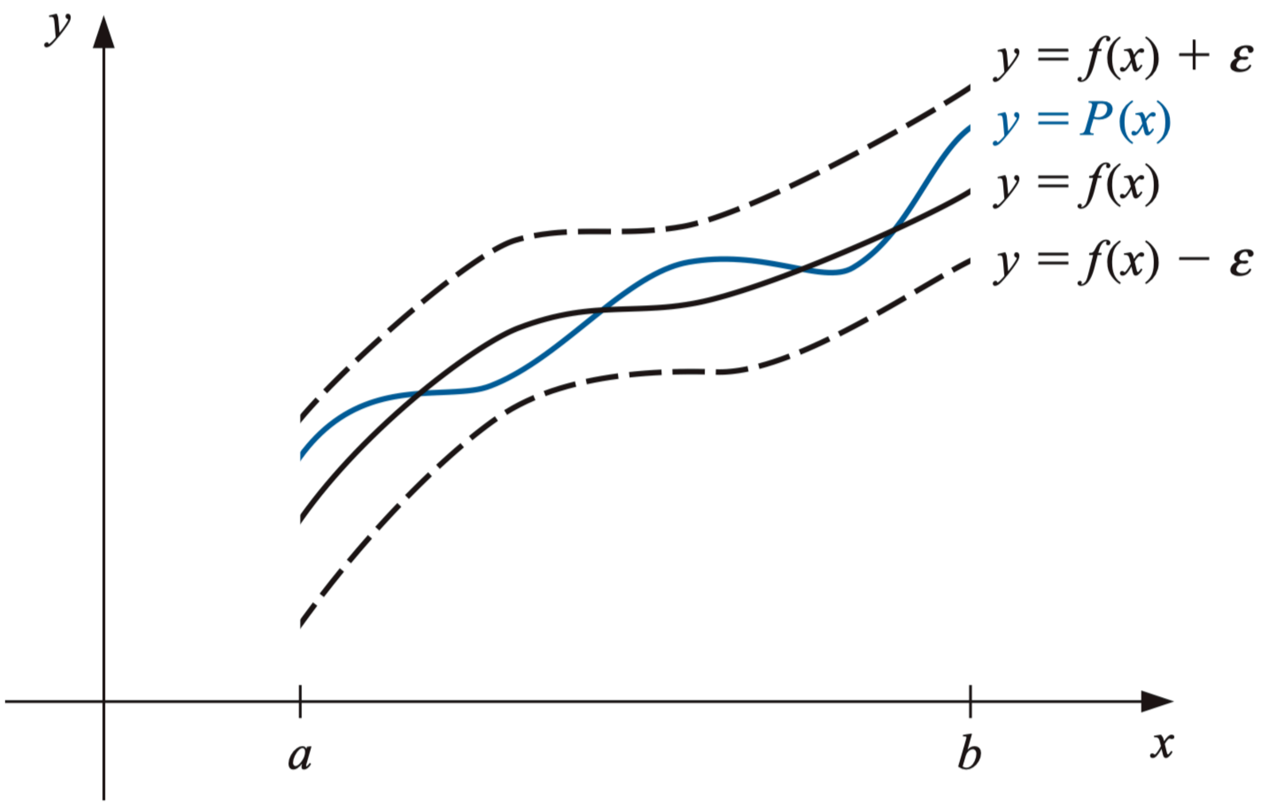
- Pretty confident: there is a polynomial relationship
- Little/no scatter
- Want to find an expression that passes exactly through all the points

- Unsure what the relationship is
- Clear scatter
- Want to find an expression that captures the trend: minimize some measure of the error of all the points…

Credit to Roger Crawfis

Why using polynomials in function approximation?

- Uniformly approximate continuous functions (Weierstrass approximation theorem)
- The derivative and indefinite integral of a polynomial are easy to determine and are also polynomials



Definitions from calculus

- The **limit** statement $\lim_{x \to a} f(x) = L$ means that for any $\varepsilon > 0$, there is a $\delta > 0$ such that $|f(x) - L| < \varepsilon$ whenever $0 < |x - a| < \delta$.

- A function f is **continuous** at x if $\lim_{h \to 0} f(x + h) = f(x)$.

- If $\lim_{h \to 0} \frac{1}{h}[f(x + h) - f(x)]$ exits, it is denoted by $f'(x)$ or $\frac{d}{dx} f(x)$ and is termed the **derivative** of f at x.

Weierstrass approximation theorem

- Suppose that f is defined and continuous on $[a, b]$. For each $\epsilon > 0$, there exists a polynomial $P(x)$, with the property that

$$|f(x) - P(x)| < \epsilon, \qquad \text{for all } x \text{ in } [a, b]$$

- Given any function, defined and continuous on a closed and bounded interval, there exists a polynomial that is as "close" to the given function as desired

- Polynomials:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

where n is a nonnegative integer and $a_0, \ldots, a_n$ are real constants.

Polynomial interpolation

- **Existence** – does there exist a polynomial that exactly passes through the $n + 1$ data points?

- **Uniqueness** – Is there more than one such polynomial?

Credit to Roger Crawfis

Existence of polynomial interpolation

- Summation of terms, such that:
  - Equal to $f(x)$ at a data point
  - Equal to zero at all other data points
  - Each term is a n^{th}-degree polynomial

$$P_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

$$L_i(x) = \prod_{k=0, k \neq i}^{n} \frac{(x - x_k)}{(x_i - x_k)}$$

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

Credit to Roger Crawfis

Formally expressed as a theorem

- If $x_o, x_1, \ldots, x_n$ are $n+1$ distinct numbers and f is a function whose values are given at these numbers, then a unique polynomial $P(x)$ of degree at most n exists with

$$f(x_i) = P(x_i), \qquad \text{for each } i = 0, 1, \ldots, n$$

- This polynomial is given by

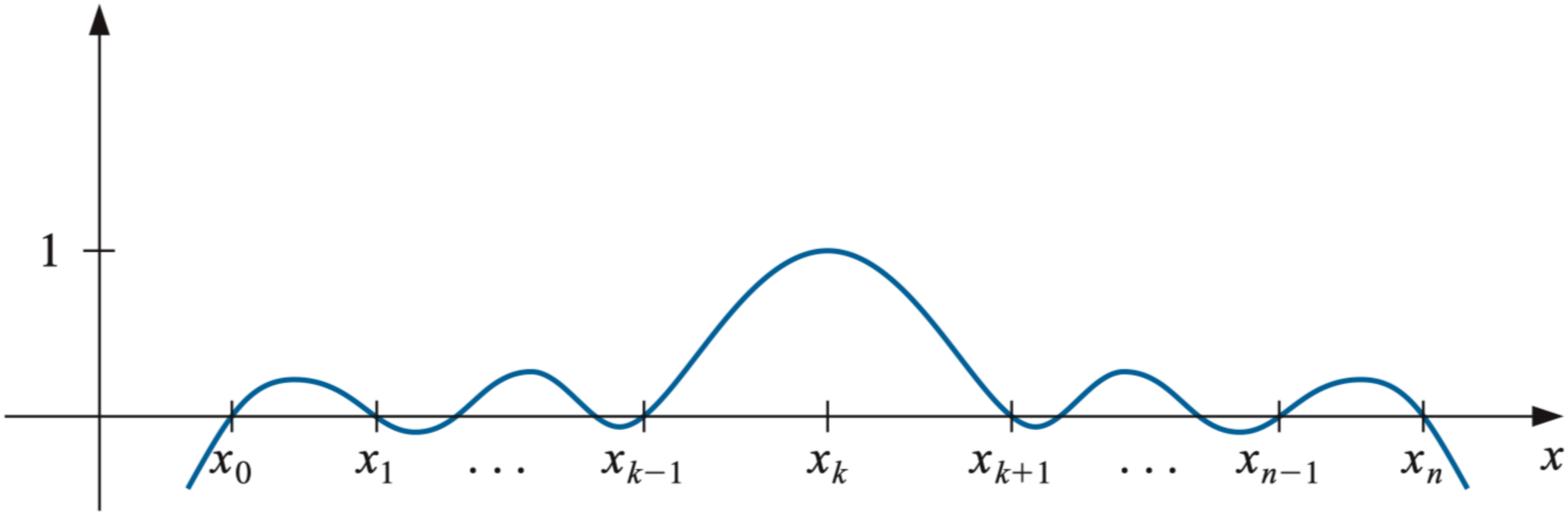
$$P(x) = f(x_0)L_0(x) + \cdots + f(x_n)L_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

- where, for each $i = 0, 1, \ldots, n$,

$$L_i(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0)(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)} = \prod_{k=0, k \neq i}^{n} \frac{(x - x_k)}{(x_i - x_k)}$$

A sketch of the graph of a typical $L_i(x)$ (when n is even)

$$L_i(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)} = \prod_{k=0,k\neq i}^{n} \frac{(x-x_k)}{(x_i-x_k)}$$

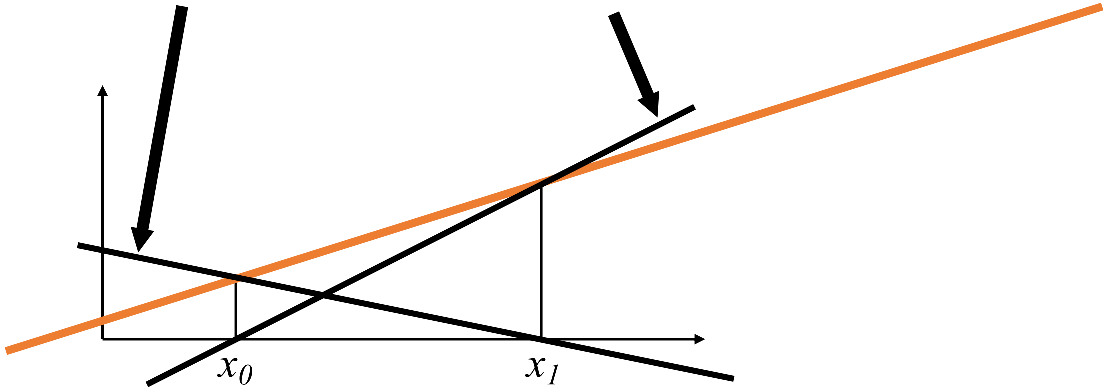


Linear interpolation

- Summation of two lines:

$$P_1(x) = \sum_{i=0}^{1} L_i(x) f(x_i)$$

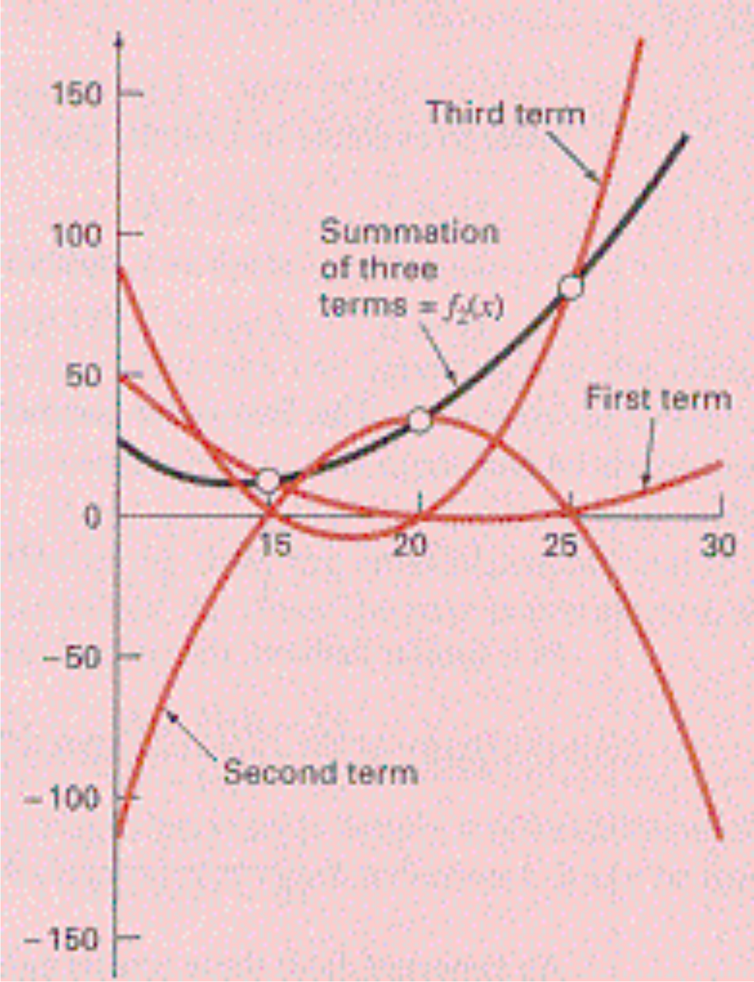
$$= \frac{(x - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1)$$



Credit to Roger Crawfis

Lagrange polynomials

- 2^{nd} order case => quadratic polynomials



Credit to Roger Crawfis

Untangling the *t*-SNE algorithm

t-distributed stochastic neighbor embedding (t-SNE)

- Given a d-dimensional dataset $X = \{x_1, x_2, \dots, x_N\} \subset \mathbb{R}^d$, t-SNE aims to compute the low-dimensional embedding

$$Y = \{y_1, y_2, \dots, y_N\} \subset \mathbb{R}^s$$

- where $s \ll d$, such that if two points x_i and x_j are close in the input space, then their corresponding points y_i and y_j are also close. Affinities between points and in the input space, p_{ij}, are defined as

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

- where σ_i is the bandwidth of the Gaussian distribution

t-distributed stochastic neighbor embedding

- Similarly, the affinity between points y_i and y_j in the embedding space is defined using the Cauchy kernel

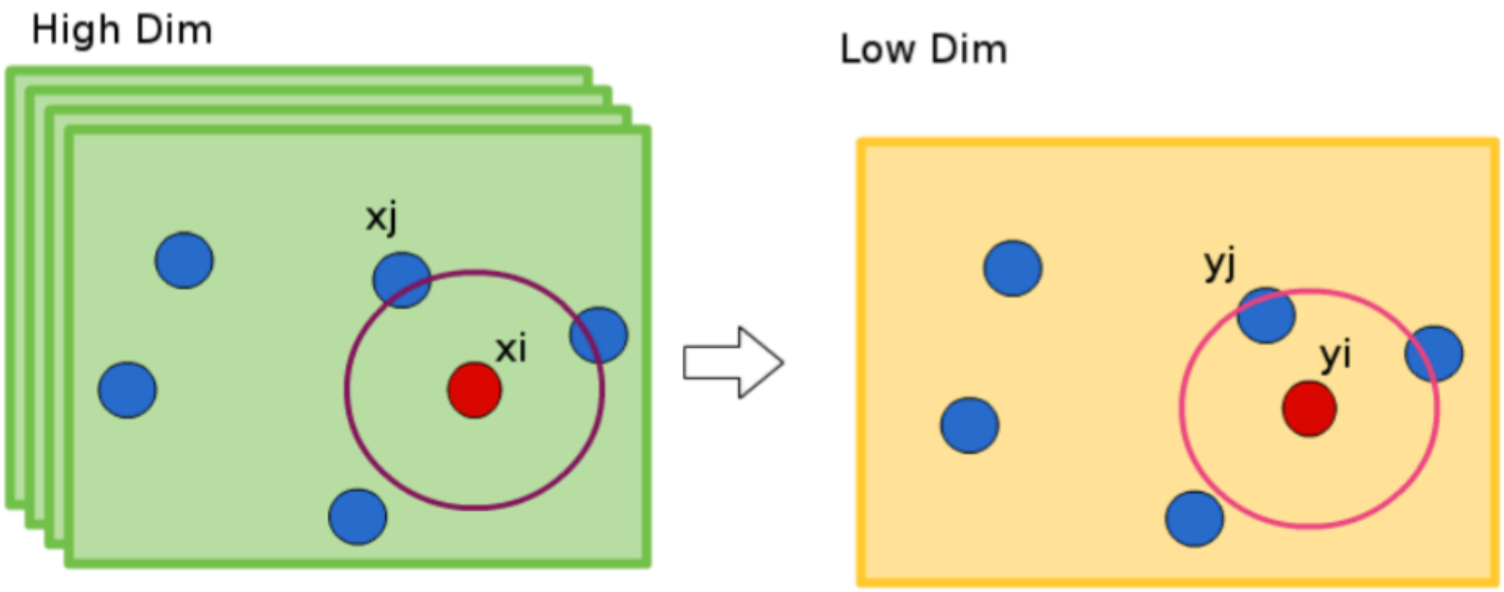
$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

- *t*-SNE finds the points $\{y_1, y_2, \ldots, y_N\}$ that minimize the Kullback–Leibler (KL) divergence between the joint distribution of points in the input space P and the joint distribution of the points in the embedding space Q,

$$C(Y) = \mathrm{KL}(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

t-SNE preserves the local structure of the high-dimensional data

- Measure pairwise similarities between high-dimensional and low-dimensional objects



https://www.slideshare.net/ssuserb667a8/visualization-data-using-tsne

t-distributed stochastic neighbor embedding

- Starting with a random initialization, the cost function $C(Y)$ is minimized by gradient descent, with the gradient

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

- where Z is a global normalization constant

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

- We split the gradient into two parts

$$\frac{1}{4}\frac{\partial C}{\partial y_i} = \sum_{j\neq i} p_{ij}q_{ij}Z(y_i - y_j) - \sum_{j\neq i} q_{ij}^2 Z(y_i - y_j)$$

attractive force between points

repulsive force between points

$$\frac{1}{4}\frac{\partial C}{\partial y_i} = F_{attr,i} - F_{rep,i}$$

Computation complexity of *t*-SNE

- The computation of the gradient at each step is an N-body simulation, where the position of each point is determined by the forces exerted on it by all other points.
- Exact computation of N-body simulations scales as $O(N^2)$, making exact *t*-SNE computationally prohibitive for datasets with tens of thousands of points.

$$\frac{1}{4}\frac{\partial C}{\partial y_i} = \sum_{j \neq i} p_{ij} q_{ij} Z(y_i - y_j) - \sum_{j \neq i} q_{ij}^2 Z(y_i - y_j)$$

$$\frac{1}{4}\frac{\partial C}{\partial y_i} = F_{attr,i} - F_{rep,i}$$

Computation complexity of *t*-SNE

- The attractive force between two points decays exponentially fast as a function of the distance between them, so that a point exerts a significant attractive force only on its nearest neighbors.
- Only nearest neighbors need to be considered when calculated $F_{attr,i}$
- Computation of $F_{rep,i}$ is the most time-consuming step in *t*-SNE

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

$$\frac{1}{4}\frac{\partial C}{\partial y_i} = \sum_{j\neq i} p_{ij}q_{ij}Z(y_i - y_j) - \sum_{j\neq i} q_{ij}^2 Z(y_i - y_j)$$

$$\frac{1}{4}\frac{\partial C}{\partial y_i} = F_{attr,i} - F_{rep,i}$$

Accelerating computation of repulsive forces in FIt-SNE

- Recall that $\{y_1, y_2, \dots, y_N\}$ is the s-dimensional embedding of a collection of d-dimensional vectors $\{x_1, x_2, \dots, x_N\}$. At each step of gradient descent, the repulsive forces are given by

$$F_{\mathrm{rep},i}(m) = \frac{\sum_{l=1,l\neq i}^{N} \frac{y_l(m) - y_i(m)}{(1 + \|y_l - y_i\|^2)^2}}{\sum_{j=1}^{N} \sum_{l=1,l\neq j}^{N} \frac{1}{\left(1 + \|y_l - y_j\|^2\right)}}$$

- where $i = 1, 2, \dots, N$; $m = 1, 2, \dots, s$; and $y_i(j)$ denotes the jth component of y_i.

- Evidently, the repulsive force between the vectors $\{y_1, y_2, \dots, y_N\}$ consists of N^2 pairwise interactions, and were it computed directly, it would require CPU time scaling as $O(N^2)$.

The authors proposed an approach enabling the computation in $O(N)$ time

Accelerating computation of repulsive forces in FIt-SNE

$$F_{\text{rep},i}(m) = \frac{\sum_{l=1,l\neq i}^{N} \frac{y_l(m) - y_i(m)}{(1 + \|y_l - y_i\|^2)^2}}{\sum_{j=1}^{N} \sum_{l=1,l\neq j}^{N} \frac{1}{\left(1 + \|y_l - y_j\|^2\right)}}$$

- By observation:
  - the repulsive forces $F_{\text{rep},i}$ defined in the above equation can be expressed as sums of the form

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

- where the kernel $K(y,z)$ is either

$$K_1(y,z) = \frac{1}{(1+\|y-z\|^2)} \quad \text{or} \quad K_2(y,z) = \frac{1}{(1+\|y-z\|^2)^2}$$

- for $y, z \in \mathbb{R}^s$. Note that both of the kernels K_1 and K_2 are smooth functions of y, z for all $y, z \in \mathbb{R}^s$.

Using polynomials to approximate kernels

- Let p be a positive integer. Suppose that $\tilde{z}_1, \ldots, \tilde{z}_p$ are a collection of p points on the interval I_{z_0} and that $\tilde{y}_1, \ldots, \tilde{y}_p$ are a collection of p points on the interval I_{y_0}.
- Let $K_p(y, z)$ denote a bivariate polynomial interpolant of the kernel $K(y, z)$ satisfying

$$K_p(\tilde{y}_j, \tilde{z}_l) = K(\tilde{y}_j, \tilde{z}_l), \qquad j, l = 1,2,, \ldots, p$$

Using polynomials to approximate kernels

$$K_p(\tilde{y}_j, \tilde{z}_l) = K(\tilde{y}_j, \tilde{z}_l), \qquad j, l = 1,2,, \ldots, p$$

- A simple calculation shows that $K_p(y, z)$ is given by

$$K_p(y, z) = \sum_{l=1}^{p} \sum_{j=1}^{p} K(\tilde{y}_j, \tilde{z}_l) \, L_{j,\tilde{y}}(y) L_{l,\tilde{z}}(z)$$

- where $L_{j,\tilde{y}}(y)$ and $L_{l,\tilde{z}}(z)$ are the Lagrange polynomials

$$L_{j,\tilde{y}}(y) = \prod_{j=1, j\neq l}^{p} \frac{(y - \tilde{y}_j)}{(\tilde{y}_l - \tilde{y}_j)} \quad \text{and} \quad L_{l,\tilde{z}}(z) = \prod_{j=1, j\neq l}^{p} \frac{(z - \tilde{z}_j)}{(\tilde{z}_l - \tilde{z}_j)}$$

- where $l = 1, 2, \ldots, p$. In the following, we refer to the points $\tilde{y}_1, \ldots, \tilde{y}_p$ and $\tilde{z}_1, \ldots, \tilde{z}_p$ as interpolation points.

Previous slide

Formal expressed as a theorem

- If $x_o, x_1, \dots, x_n$ are $n+1$ distinct numbers and f is a function whose values are given at these numbers, then a unique polynomial $P(x)$ of degree at most n exists with

$$f(x_i) = P(x_i), \qquad \text{for each } i = 0, 1, \dots, n$$

- This polynomial is given by

$$P(x) = f(x_0)L_0(x) + \cdots + f(x_n)L_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

- where, for each $i = 0, 1, \dots, n$,

$$L_i(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)} = \prod_{k=0, k\neq i}^{n} \frac{(x-x_k)}{(x_i-x_k)}$$

Using polynomials to approximate kernels

$$\phi(y_i) = \sum_{j=1}^{N} K\big(y_i, z_j\big) q_j \qquad K_1(y,z) = \frac{1}{(1+\|y-z\|^2)} \quad \text{or} \quad K_2(y,z) = \frac{1}{(1+\|y-z\|^2)^2}$$

- Let $\tilde{\phi}(y_i)$ denote the approximation to $\phi(y_i)$ obtained by replacing the kernel K in the above equation by its polynomial interpolant K_p, that is,

$$\tilde{\phi}(y_i) = \sum_{j=1}^{N} K_p\big(y_i, z_j\big) q_j\,, \qquad \text{for } i = 1, 2, \ldots, N$$

$$K_p(y,z) = \sum_{l=1}^{p} \sum_{j=1}^{p} K\big(\tilde{y}_j, \tilde{z}_l\big) L_{j,\tilde{y}}(y) L_{l,\tilde{z}}(z)$$

$$L_{j,\tilde{y}}(y) = \prod_{j=1, j\neq l}^{p} \frac{(y - \tilde{y}_j)}{(\tilde{y}_l - \tilde{y}_j)}$$

$$L_{l,\tilde{z}}(z) = \prod_{j=1, j\neq l}^{p} \frac{(z - \tilde{z}_j)}{(\tilde{z}_l - \tilde{z}_j)}$$

Analysis of the computation complexity

- The direct computation of $\phi(y_1), \dots, \phi(y_N)$ requires $\boldsymbol{O(N^2)}$ operations. In comparison, the values of $\tilde{\phi}(y_1), \dots, \tilde{\phi}(y_N)$ can be computed in $\boldsymbol{O(2N \cdot p + p^2)}$.

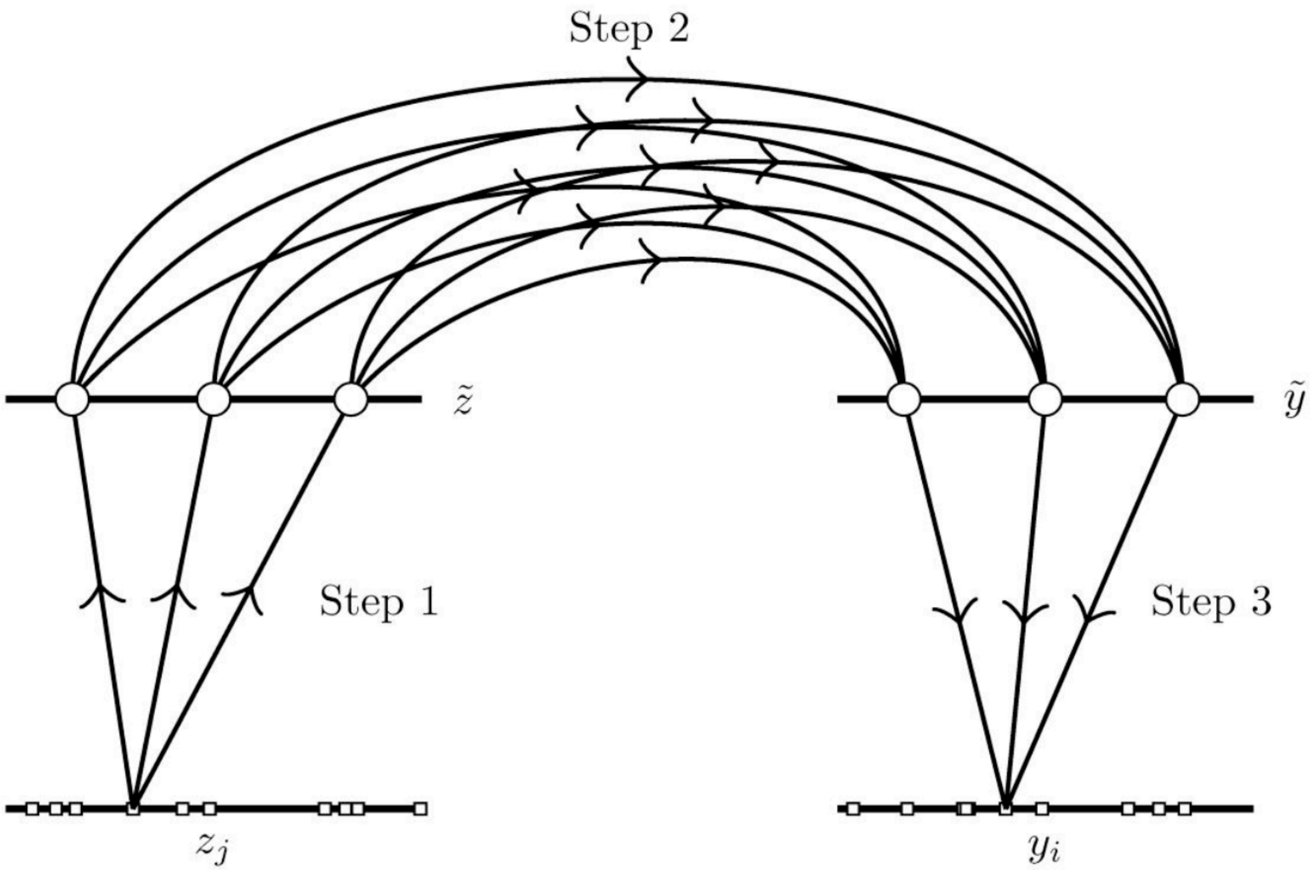
$$\tilde{\phi}(y_i) = \sum_{j=1}^{N} K_p(y_i, z_j) q_j$$

$$= \sum_{j=1}^{N} \sum_{l=1}^{p} \sum_{m=1}^{p} K(\tilde{y}_l, \tilde{z}_m) L_{l,\tilde{y}}(y_i) L_{m,\tilde{z}}(z_j) q_j$$

$$= \underbrace{\sum_{l=1}^{p} L_{l,\tilde{y}}(y_i)}_{\boldsymbol{O(N \cdot p)}} \Bigg(\underbrace{\sum_{m=1}^{p} K(\tilde{y}_l, \tilde{z}_m)}_{\boldsymbol{O(p^2)}} \Bigg(\underbrace{\sum_{j=1}^{N} L_{m,\tilde{z}}(z_j) q_j}_{\boldsymbol{O(N \cdot p)}} \Bigg) \Bigg) \qquad \text{for } i = 1, 2, \dots, N$$

An illustration of the algorithm

- In the lower intervals, the white squares denote the locations z_j and y_i, and in the upper intervals the white circles indicate the locations of the equispaced nodes $\tilde{z}$ and $\tilde{y}$. The arrows illustrate how a point z_j communicates with a point y_i.



Experimental results

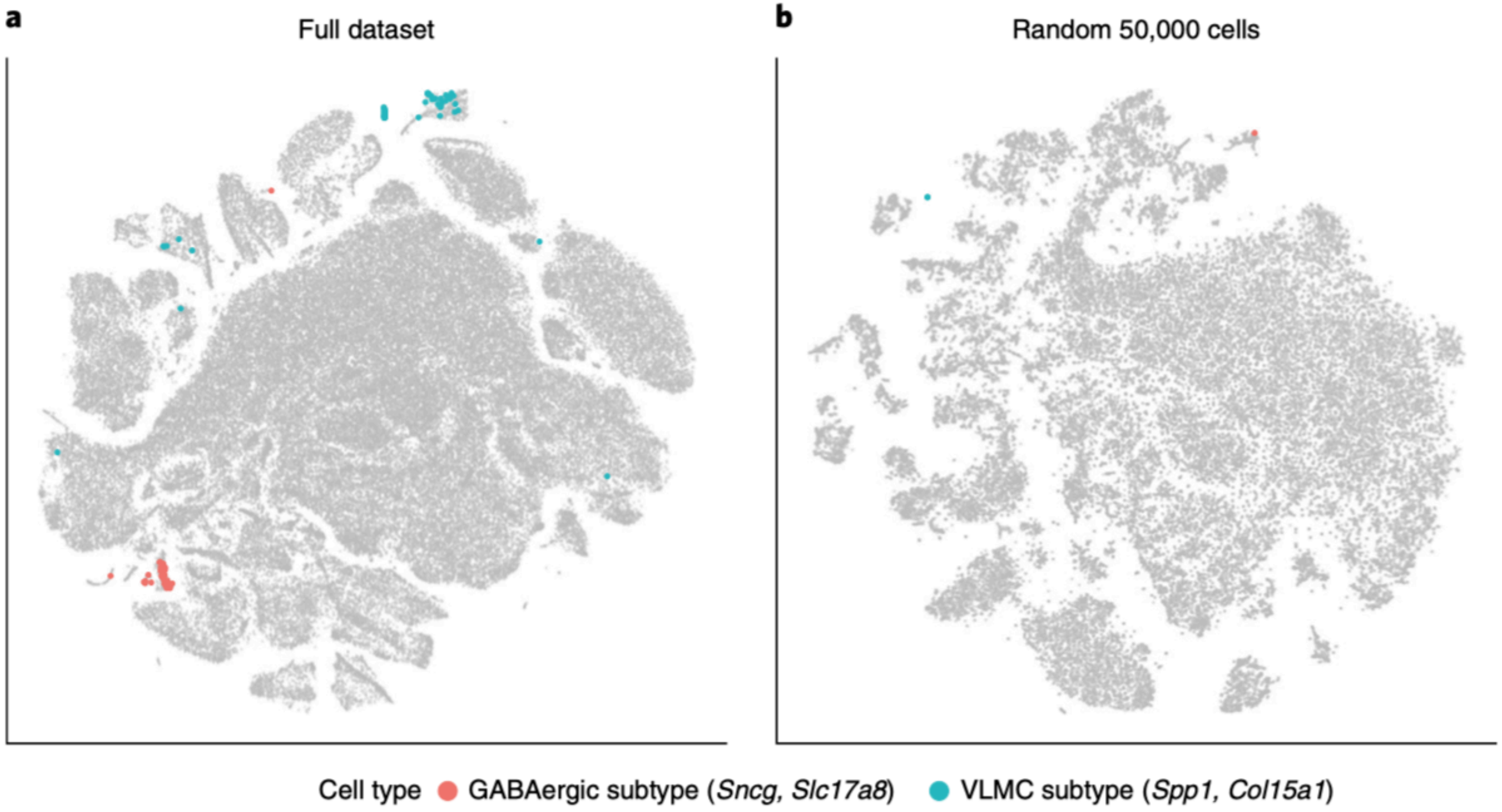
The computation complexity is remarkably reduced

Table 1 | Time taken for 1,000 iterations of the gradient descent phase of 2D t-SNE using BH t-SNE compared to our implementation (FIt-SNE), as compared on a 2017 Macbook Pro for a given number of points *N*

N	BH t-SNE	FIt-SNE
10,000	1 min	<1 min
100,000	11 min	<1 min
500,000	1 h 10 min	3 min
1,000,000	3 h 9 min	15 min

See the Methods for more details.

Identifying subpopulations in a large dataset by using marker genes



Summary

Identification of the most time-consuming part in the *t*-SNE algorithm

$\Downarrow$

Recognition of the computation problem as

polynomial interpolation

$\Downarrow$

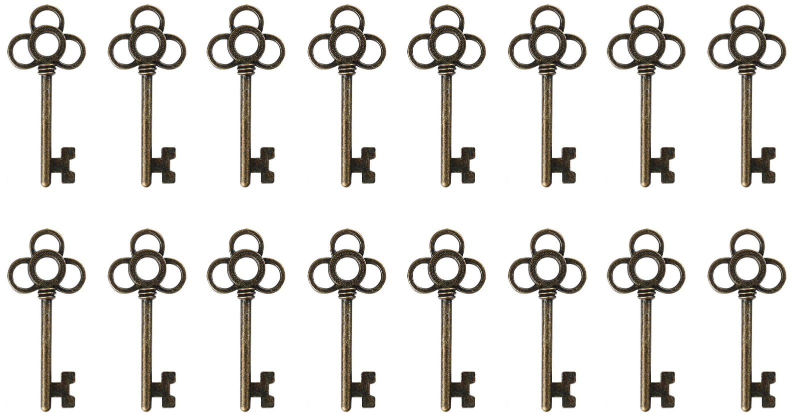
Problem solved

Discussion

Find your question

Find your approach

Don't settle



www.amazon.com

Thank you!