t-SNE Notes

Iakovidis Ioannis
email iakoviid@auth.gr

December 27, 2019

1 Introduction

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.

Let $x_i \in \mathbf{R^d}, i = 1, \ldots, n$ be the data samples in high dimensional space and $y_i \in \mathbf{R^s}, i = 1, \ldots, n$ be the data samples in low dimensional space, s is usually 2 or 3. The goal of t-SNE is to preserve local structure, points that are similar (close) in the high dimensional space are mapped to similar (close) points in the low dimensional space. This is achieved by minimizing the divergence between similarity weights that are defined in the high and low dimensional space.

More specifically from the distances d_{ij} of $(x_i)_{i=1}^n$ we define

$$p_{i|j} = \frac{e^{-d_{ij}^2/2\sigma_i^2}}{\sum_{l \neq i} e^{-d_{il}^2/2\sigma_i^2}} \quad \text{and} \quad p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$$

as the similarity weights in the high dimensional space. The value $p_{j|i}$ is interpreted as the distribution of the other points given x_i or the probability that point j is a neighbor of point i. The parameters N and σ_i are chosen. For the low dimensional space we define

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

as similarity weights but this time we use the t-distribution. For notation purposes let $P = \{p_{ij}\}$, $Q = \{q_{ij}\}$ $n \times n$ matrices and $X = [x_1, \ldots, x_n]$, $Y = [y_1, \ldots, y_n]$ $d \times n$ and $s \times n$ matrices respectively.

And then we take KL divergence

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

Minimization can be done with gradient descent. Note that this function is non convex and hence we will halt at a local minimum.

2 Computing the Gradient

Here with straight calculations we compute the gradient. First of all:

$$\begin{aligned}
C(Y) &= \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \\
&= \sum_{i\neq j} p_{ij} \log p_{ij} - \sum_{i\neq j} p_{ij} \log q_{ij} \\
&= \sum_{i\neq j} p_{ij} \log p_{ij} + \sum_{i\neq j} p_{ij} \log f_{ij} + \sum_{i\neq j} p_{ij} \log Z \\
&= \sum_{i\neq j} p_{ij} \log p_{ij} + \sum_{i\neq j} p_{ij} \log f_{ij} + \log Z.
\end{aligned}$$

Where $f_{ij} = 1 + \|y_i - y_j\|^2$ and $Z = \sum_{k\neq l}(1 + \|y_k - y_l\|^2)^{-1}$.
With some computation:

$$\begin{aligned}
\frac{\partial \sum_{i\neq j} p_{ij} \log f_{ij}}{\partial y_m} &= \sum_{i\neq m} p_{im} \frac{\partial \log f_{im}}{\partial y_m} + \sum_{i\neq m} p_{mi} \frac{\partial \log f_{mi}}{\partial y_m} \\
&= 2\sum_{i\neq m} p_{mi} \frac{\partial \log f_{mi}}{\partial y_m} \\
&= 2\sum_{i\neq m} p_{mi} \frac{1}{f_{mi}} \frac{\partial f_{mi}}{\partial y_m} \\
&= 2\sum_{i\neq m} p_{mi} \frac{2(y_m - y_i)}{1 + \|y_m - y_i\|^2} \\
&= 4\sum_{i\neq m} p_{mi} \frac{Z(y_m - y_i)}{Z(1 + \|y_m - y_i\|^2)} \\
&= 4\sum_{i\neq m} p_{mi} q_{mi} Z(y_m - y_i).
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \log Z}{\partial y_m} &= \frac{\partial Z}{\partial y_m} \frac{1}{Z} \\
&= \frac{2}{Z} \sum_{i\neq m} \frac{\partial (1 + \|y_m - y_i\|^2)^{-1}}{\partial y_m} \\
&= -\frac{2}{Z} \sum_{k\neq m} \frac{(y_m - y_i)}{1 + \|y_m - y_i\|^2)^2} \\
&= -\frac{2}{Z} \sum_{i\neq m} \frac{2(y_m - y_i)}{1 + \|y_m - y_i\|^2} \\
&= -4\sum_{i\neq m} q_{mi}^2 Z(y_m - y_i).
\end{aligned}$$

So

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

Also we can make the computation so that it generalizes more easily to changes in the objective function or the similarity weights. Observe how the objective function C depends on the embedded points. Let more general $q_{ij} = \frac{w_{ij}}{\sum_{kl} w_{kl}}$ where w_{ij} is the similarity weight of y_i and y_j depended on their distance d_{ij}.

So when computing the gradient we:

1. The distances d_ij are generated from the coordinates,y_i, y_j.
2. The similarity weights w_{ij} are generated as a function of the distances, d_ij.
3. The output probabilities, q_{ij}, are normalized versions of the similarity weights, w_{ij}.
4. The cost function, C is normally a divergence of some kind, and hence expressed in terms of the output probabilities, q_{ij}.

We're going to chain those individual bits together via the chain rule for partial derivatives. The chain of variable dependencies is $C \to q \to w \to d \to y$. So

$$\frac{\partial C}{\partial \mathbf{y_m}} = \sum_{ij} \frac{\partial C}{\partial q_{ij}} \sum_{kl} \frac{\partial q_{ij}}{\partial w_{kl}} \sum_{pq} \frac{\partial w_{kl}}{\partial d_{pq}} \frac{\partial d_{pq}}{\partial \mathbf{y_m}}$$

In the relationship between w, and d is such that any cross terms are 0, i.e. unless $k = p$ and $l = q$ those derivatives evaluate to 0. Also, either $k = m$ or $l = m$, otherwise $\partial d_{kl}/\partial \mathbf{y_m} = 0$. So

$$\begin{aligned}
\frac{\partial C}{\partial \mathbf{y_m}} &= \sum_{ij} \frac{\partial C}{\partial q_{ij}} \sum_{kl} \frac{\partial q_{ij}}{\partial w_{kl}} \sum_{pq} \frac{\partial w_{kl}}{\partial d_{pq}} \frac{\partial d_{pq}}{\partial \mathbf{y_m}} \\
&= \sum_{ij} \frac{\partial C}{\partial q_{ij}} \sum_{kl} \frac{\partial q_{ij}}{\partial w_{kl}} \frac{\partial w_{kl}}{\partial d_{kl}} \frac{\partial d_{kl}}{\partial \mathbf{y_m}} \\
&= \sum_{ij} \frac{\partial C}{\partial q_{ij}} \sum_{l} \frac{\partial q_{ij}}{\partial w_{ml}} \frac{\partial w_{ml}}{\partial d_{ml}} \frac{\partial d_{ml}}{\partial \mathbf{y_m}} + \sum_{ij} \frac{\partial C}{\partial q_{ij}} \sum_{k} \frac{\partial q_{ij}}{\partial w_{km}} \frac{\partial w_{km}}{\partial d_{km}} \frac{\partial d_{km}}{\partial \mathbf{y_m}}
\end{aligned}$$

Now to rearrange some of the grouping of the summations:

$$\begin{aligned}
\frac{\partial C}{\partial \mathbf{y_m}} &= \sum_{l} \Big(\sum_{ij} \frac{\partial C}{\partial q_{ij}} \frac{\partial q_{ij}}{\partial w_{ml}} \Big) \frac{\partial w_{ml}}{\partial d_{ml}} \frac{\partial d_{ml}}{\partial \mathbf{y_m}} + \sum_{k} \Big(\sum_{ij} \frac{\partial C}{\partial q_{ij}} \frac{\partial q_{ij}}{\partial w_{km}} \Big) \frac{\partial w_{km}}{\partial d_{km}} \frac{\partial d_{km}}{\partial \mathbf{y_m}} \\
&= \sum_{l} f_{ml} \frac{\partial w_{ml}}{\partial d_{ml}} \frac{\partial d_{ml}}{\partial \mathbf{y_m}} + \sum_{k} f_{km} \frac{\partial w_{km}}{\partial d_{km}} \frac{\partial d_{km}}{\partial \mathbf{y_m}} \\
&= \sum_{k} f_{mk} \frac{\partial w_{km}}{\partial d_{km}} \frac{\partial d_{km}}{\partial \mathbf{y_m}} + \sum_{k} f_{km} \frac{\partial w_{km}}{\partial d_{km}} \frac{\partial d_{km}}{\partial \mathbf{y_m}}, \text{Assume w and d are symmetric} \\
&= \sum_{k} \Big(f_{mk} + f_{km} \Big) \frac{\partial w_{km}}{\partial d_{km}} \frac{\partial d_{km}}{\partial \mathbf{y_m}}
\end{aligned}$$

Now

$$\frac{\partial d_{ij}}{\partial \mathbf{y_i}} = \frac{1}{d_{ij}} (\mathbf{y_i} - \mathbf{y_j}) \text{ for } d_{ij} = \left[\sum_{l}^{K} (y_{il} - y_{jl})^2 \right]^{1/2}$$

And

$$\frac{\partial w_{ij}}{\partial d_{ij}} = -\frac{2d_{ij}}{(1+d_{ij}^2)} \text{ for } w_{ij} = \frac{1}{1+d_{ij}^2}$$

So

$$\frac{\partial C}{\partial \mathbf{y_m}} = -2\sum_k \Big(f_{mk} + f_{km}\Big)\frac{(y_m - y_k)}{(1+{d_{km}}^2)}$$

$$\frac{\partial C}{\partial q_{ij}} = -\frac{p_{ij}}{q_{ij}}$$

3 Gradient Interpretation

The t-SNE gradient computation can be reformulated as an N-body simulation problem:

$$F_{attr,i} = \sum_{j\neq i} p_{ij} q_{ij} Z(y_i - y_j)$$

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

$$\frac{1}{4}\frac{\partial C(Y)}{\partial y_i} = F_{attr,i} + F_{rep,i}$$

The forces can be also viewed with matrix-vector computations in the following ways:

- $$F_{attr} = (P \odot Q)O \odot Y - (P \odot Q)Y$$
  $$F_{rep} = (Q \odot Q)O \odot Y - (Q \odot Q)Y$$
  Where Y is the $N \times 2$ matrix of embedded points, O is an $N \times 2$ matrix of ones.

-

4 First Complexity Discussion

5 Fast Multipole Methods

Essentially a subproblem of computing the gradient is to compute sums of the form:

$$u_i = \sum_{j=1}^{N} G(y_i, y_j) v_j$$

in our case for computing the repulsive forces $G(y_i, y_j) = \frac{1}{1+\|y_i - y_j\|^2}$. More generally the fast multipole methods are methods are constructed to compute sums of the form:

$$u_i = \sum_{j=1}^{N} G(t_i, y_j) v_j$$

the points $\{t_i\}_1^M$ are called targets and $\{y_i\}_1^N$ are called sources. While the kernel $G(t_i, y_j)$ depends on the distance of t_i and y_j.

Putting all evaluations of the kernel in a matrix $A : A_{ij} = G(t_i, y_j)$ we can see that $u = Ax$. It is known that if the domain of T is different than the domain of Y then we can separate the variables and have $A \approx B(X)C(Y)$, let B be $M \times P$ and C be $P \times N$ matrices. This is true because if the kernel have separate domains (does not blow up) then A can be efficiently approximated by a low rank matrix.

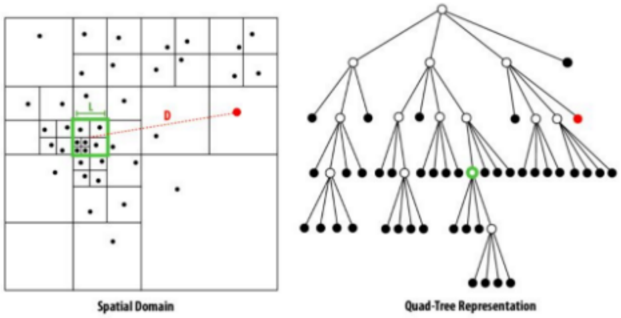
$$\begin{aligned} u_i &= \sum_{j=1}^{N} G(t_i, y_j) v_j \\ &= \sum_{j=1}^{N} \sum_{k=1}^{P} B_{ik} C_{kj} v_j \\ &= \sum_{k=1}^{P} B_{ik} \Big(\sum_{j=1}^{N} C_{kj} v_j \Big) \end{aligned}$$

6 Separation of Variables

7 Barnes–Hut

The Barnes–Hut simulation is an approximation algorithm for performing an n-body simulation. It is notable for having order $O(n \log n)$ compared to a direct-sum algorithm which would be $O(n^2)$. The space is hierarchically divided into cells (rectangular or cubic), so that only points from nearby cells need to be treated individually, and points in distant cells can be treated as a single large point centered at the cell's center of mass. This can dramatically reduce the number of particle pair interactions that must be computed.

More specifically the Barnes–Hut algorithm constructs a octtree or quadtree and for each point it does a depth first search on that tree at each node testing a condition that says if the approximation for this cell is valid.



8 Barnes–Hut an algebraic interpretation

It is clear that given a constructed space tree we can reorder the matrix $A = \{q_{ij}^2\}$ to a matrix A' that has the recursive partition:

$$A' = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

for a binary tree $y \in \mathbf{R}$ or

$$A' = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix}$$

for a quadtree $y \in \mathbf{R}^2$ similarly for an octtree.

Then for the computation for

$$u = A'v = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} v_{near} \\ v_{far} \end{bmatrix} = \begin{bmatrix} A_{11}v_{near} + A_{12}v_{far} \\ A_{21}v_{near} + A_{22}v_{far} \end{bmatrix} \approx \begin{bmatrix} A_{11}v_{near} + a_1 \\ a_2 + A_{22}v_{far} \end{bmatrix}$$

a_1 and a_2 represent the approximation of the the terms of the cells with the combined force from the center of mass. This approximation can be done recursively at any level.

9 Theoretical Analysis