t-SNE

Visualizing high dimensional data in a 2D space

Emil K Svensson

2017-05-24

{r setup, include=FALSE} knitr::opts_chunk$set(echo = FALSE)

Disclaimer

Everything not referenced in is this presentation is from the paper Visualizing High-Dimensional Data Using t-SNE. Written by L.J.P. van der Maaten and G.E. Hinton. published in the Journal of Machine Learning Research 9(Nov):2579-2605, 2008.

Overview

- What is high dimensional data?
- Why is t-SNE necessary?
- How does it work?
- How does it preform?

High Dimensional Data

- Lot of data, both columns and rows
- Data Mining, searching for underlying structures
- Finding patterns - hard for humans, easy for computers
- Easiest way to understand is to visualize

The problems with visualization

- Humans are limited in preciving more than 3 dimensions
- Visualization techniques for datasets with more dimensions.
- PCA, Chernoff-faces, Multidimensional Scaling (MDS)

What is t-SNE trying to achive?

- Reduce the number of dimensions
- Conserve the local structure of the data
- Display similar but ambigious objects close to eachother

Number example

- MINST data set
- Contains data from handwritten digits
- Every digit is a picture of 28 x 28 = 784 columns (pixels)
- Similar digits close to eachother and well separated

Picture

SNE

- t-SNE builds on the work of SNE, stochastic neighbohur embedding
- Suffers from “The crowding problem”
- t-distributed kernels instead of gaussian

The math behind the magic

- Perplexity
- Pairwise distances
- t-distributed kernel (Cauchy distributed)
- Gradient Decent
- Kullback libler divergence

Perplexity

- A tunable hyperparameter that is required
- The relationship between local and global structure of data
- Has a impact on the final result
- Supposed to be stable between Perplexity between 5 - 50

The Cost Function

- Compare two different distributions
- Kullback Leibler Divergence

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

Gradient Decent

- Optimization algorithm
- Not guaranteed to arrive at the global minimum but a local
- Each itteration trying to get closer to the minimum
- Running gradient decent on the derivative of this expression

$$\frac{\delta C}{\delta y_{ij}} = 4\sum_{j}(p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_k - y_l||^2)^{-1}$$

Code example of gradient decent

```
# set up a stepsize and no. itterations
alpha = 0.003
iter = 500
# define the gradient of f(x) = x^4 - 3*x^3 + 2
gradient = function(x) return((4*x^3) - (9*x^2))
# randomly initialize a value to x
x = floor(runif(1)*10)
# create a vector to contain all xs for all steps
x.All = vector("numeric",iter)

# gradient descent method to find the minimum
for(i in 1:iter){
        x = x - alpha*gradient(x)
        x.All[i] = x
        print(x)}
```

Source: Gradient Decent (2017,may 18) In Wikipedia Retrived 2017-05-22 from

The parts

- What are we minimizing?
- The high dimensional probabilities

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

- The low dimensional probabilities

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

Pseudocode

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$
    for *t=1* **to** T **do**
        compute low-dimensional affinities q_{ij} (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
    end
end

Figure 1:

Conclusions

- Why shouldn't we use t-SNE?
- The result varies from each run
- Hard to set hyper-parameters
- Hard to grasp every component of the algorithm

Conclusions

- Why should we use t-SNE?
- It works on real life data sets
- Can handle variation on low and high level
- Manages to handle ambigous data (words with different meanings)