# t-SNE

## Visualization of high dimensional data with stochastic neighbor embedding

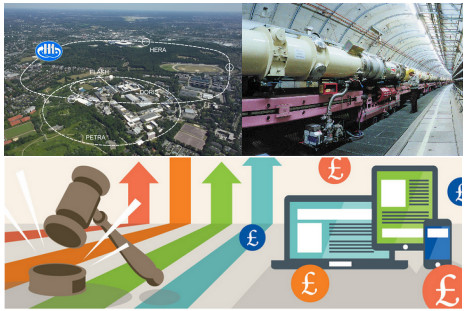
Philipp Pahl

linkedin.com/in/philipppahl

Data Meets Nice People, 02/27/2014

# Introduction

- Particle physicist at H1 (Experiment) / HERA (Collider) / DESY (Institute)
- Back-end developer and data scientist role at a Hamburg based “real time advertising” start-up
- Passionate about knowledge extraction from big and complex data sets and to turn it into a useful application



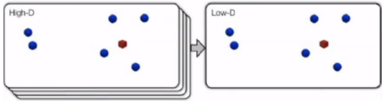
# Motivation

- The problem with big data samples, especially for exploration tasks
  - Large response times (large amount of data)
  - Many degrees of freedom / dimensions, which are difficult to capture
    - What are the features of the data?
    - What is the structure of the data? (Clusters, which entities are far apart of each other)
    - What are the relations between the entities

t-SNE helps to solve these issues

# Introduction to t-SNE

- The key idea is that you have a high dimensional metric space which contains your data and you try to arrange the points in a lower dimensional space, such that the pairwise similarity between the points is preserved

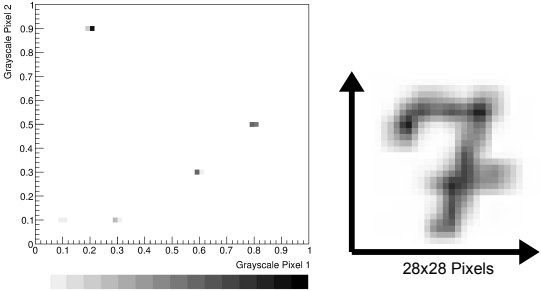



- Publications by G. Hinton and L. van der Maaten
  - G. Hinton, S. Roweis introduced SNE in 2002 ("no t!") [1]
  - L. van der Maaten, G. Hinton introduced the "t" in 2008 [2]
  - L. van der Maaten published "Barnes-Hut-SNE" in 2013 [3]
  - This talk is inspired by the Google Tech Talk [4] by LvdM[1]

[1]Some of the figures are taken from the aforementioned publications

# Example data set: MNIST[2] handwritten digits

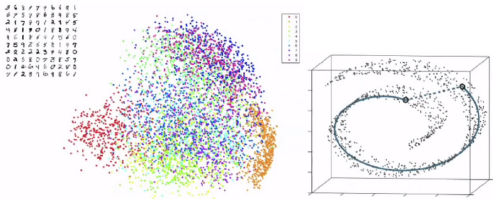
- Every pixel has a certain gray scale
- Every gray scale value defines a value in one dimension
- 28 x 28 = 784 pixels, i.e. 784 dimensions




[2]Mixed National Institute of Standards and Technology

# Linear approach (PCA) not suited for visualization

- PCA is mainly concerned with preserving large pairwise distances, since it uses squared errors
- Sometimes you are more interested in preserving the local structure
- Example of non-linear dimensionality reduction is "Local Linear Embedding" (LLE)
  - Has also the "crowding problem"



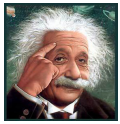
# The basic SNE algorithm I

- The similarity of the points $i$ and $j$

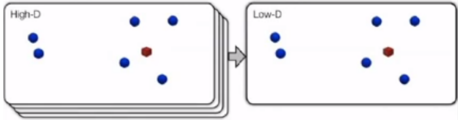
$$p_{ij} = \frac{\exp(-d_{ij}^2)}{\sum_{k \neq i} \exp(-d_{ik}^2)}$$

- The distance is defined as

$$d_{ij}^2 = \frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_i^2}$$



- Where the bandwidth is chosen such that the perplexity of the distribution over the neighbors equals $k$
- i.e. the metric $d_{ij}$ is a local property




- Perplexity:

$$Perp(P_i) = 2^{H(P_i)}$$

- "The perplexity can be interpreted as a smooth measure of the effective number of neighbors."
- Entropy:

$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$$

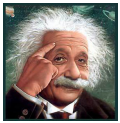
# The basic SNE algorithm II

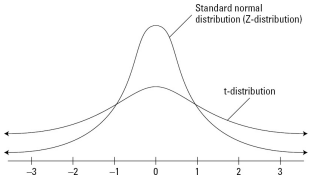
- The probabilities are symmetrized:

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

- Measure pairwise similarity in the low dimensional space using a t-distribution

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l)\|^2)^{-1}}$$

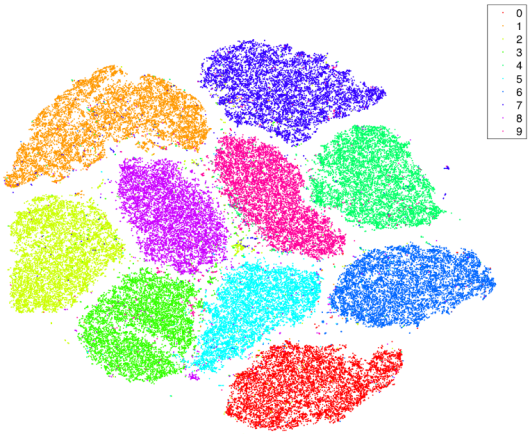


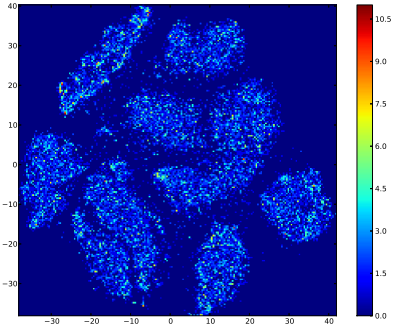



- Minimize the Kullback-Leibler divergence

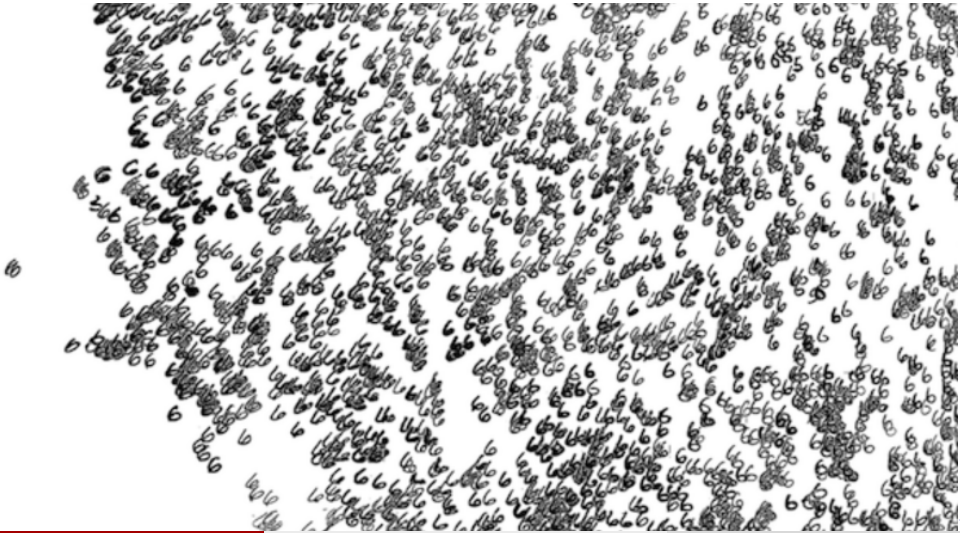
$$C(\mathcal{E}) = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

# Results from t-SNE



# Results from t-SNE II



# Implementation / Performance improvements

- Only use an effective number of nearest neighbors
  - A *Vantage Point* tree is applied to calculate $p_{ij}$ with $k$ nearest neighbors
- By far the most "expensive" part of the calculation of the embedding is the gradient of the optimization problem

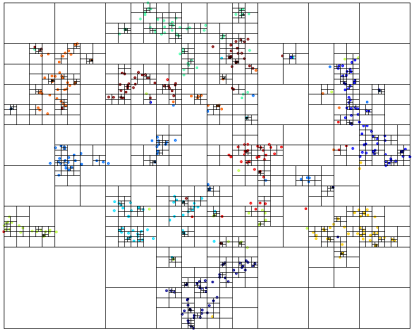
$$\frac{\partial C}{\partial \mathbf{y}_i} = 4(F_{attr} - F_{rep}) = 4\left(\sum_{j \neq i} p_{ij} q_{ij} Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j \neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right)$$

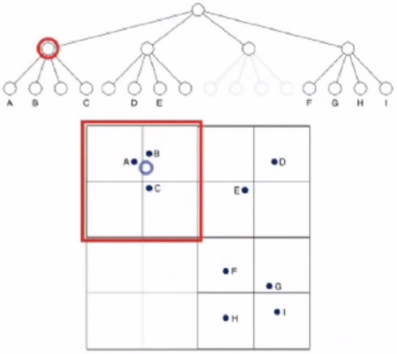
where

$$Z = (1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}$$

# Barnes-Hut algorithm

- The use of the Barnes-Hut algorithm[5] improves the calculation of the *"forces"* from $O(N^2)$ to $O(N \log N)$





# Results from deep learned NN based information

# Summary

- The *bright ideas* of t-SNE
  - The use of *affinity* in the high dimensional space: the metric is a local property
  - The use of a long tailed *Student-t distribution* in the low dimensional space
  - The use of the *Barnes-Hut* approximation
- In order to make it easier applicable I started an implementation of a JRuby extension written in Java
- The implementation is supposed to further gain speed by parallelization (multiple threads)

# Thank you

- Want to contribute?
- Start an open source project? Get it out on Bountysource?
- Do you have interesting data and want to test it with t-SNE?
- Thanks for your attention
- Questions?

# References I




G. Hinton, S. Roweis
*Stochastic neighbor embedding*
NIPS. Vol. 2. 2002.



L. van der Maaten, G. Hinton.
*Visualizing Data using t-SNE*
Journal of Machine Learning Research (2008)



L. van der Maaten
*Barnes-Hut-SNE*
arXiv:1301.3342 (2013).



L. van der Maaten
*Visualizing Data using t-SNE*
https://www.youtube.com/watch?v=RJVL80Gg3lA (2013)

# References II


J. Barnes, P. Hut
*A hierarchical $O(N \log N)$ force-calculation algorithm*
Nature, 324(4):446-449 (1986)