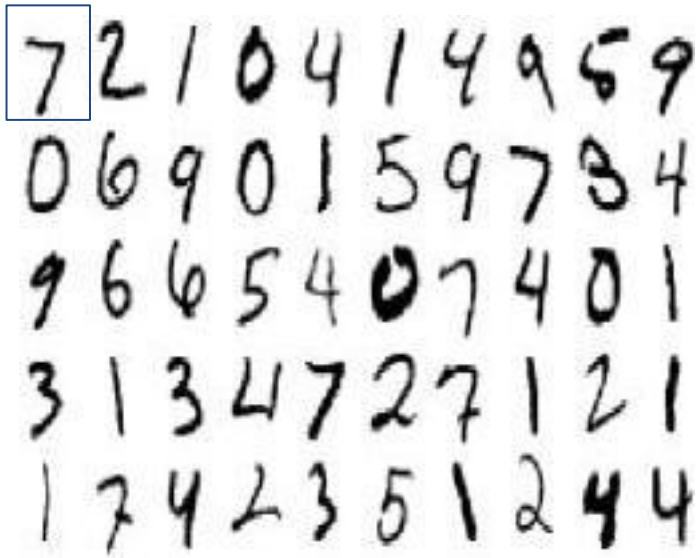
FORNAX

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Krzysztof Kolasiński Listopad - 2016

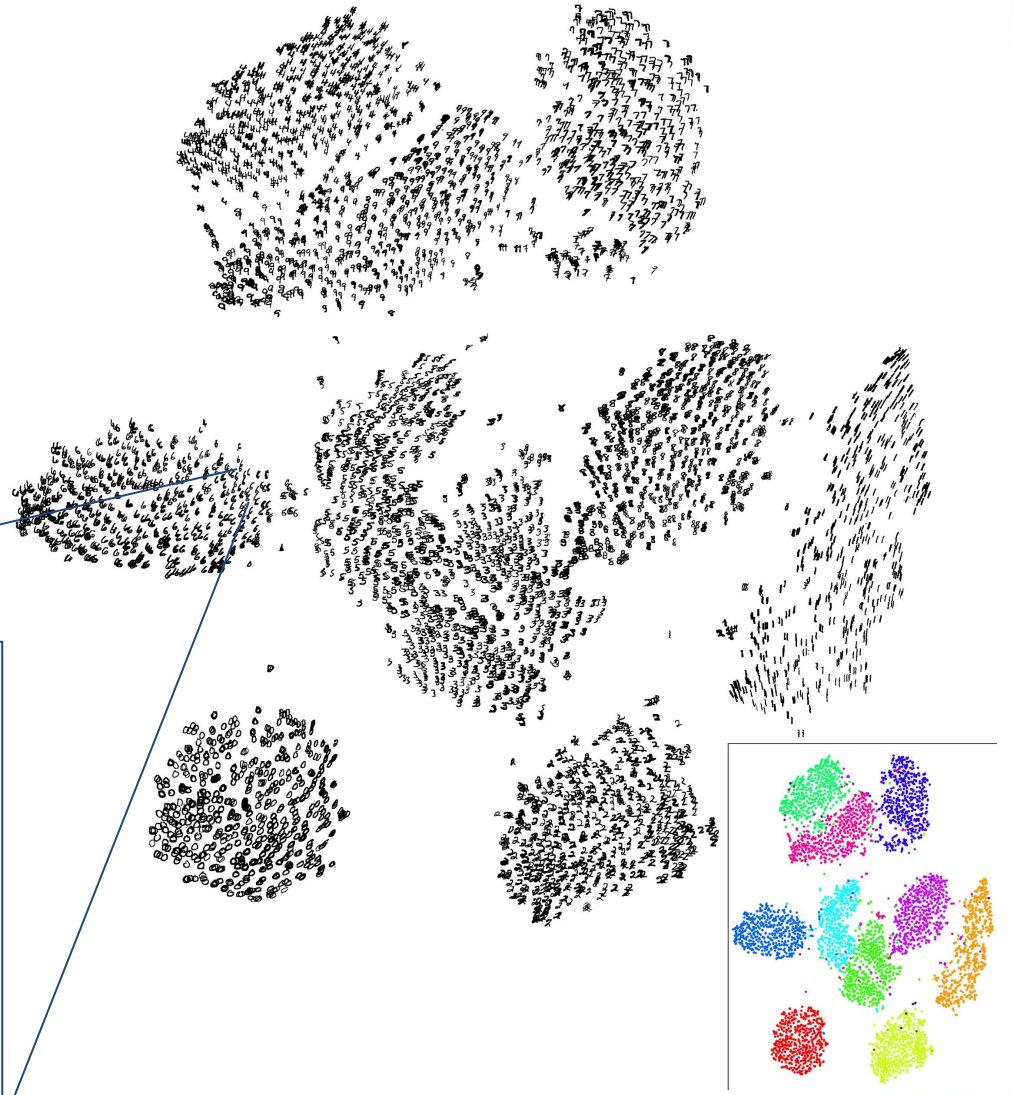
What is t-SNE ?

A technique for **dimensionality reduction** that is particularly well suited for the **visualization of high-dimensional datasets**.



t-SNE

[3D movie result]

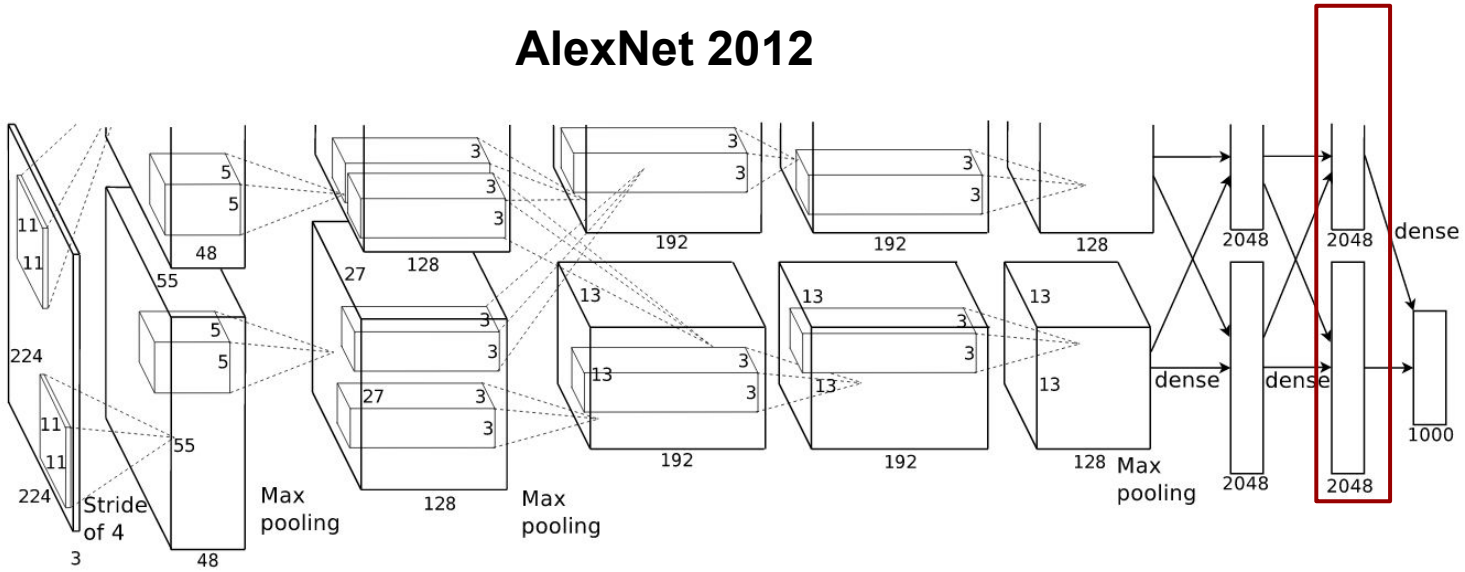


What is t-SNE ?

Another example by [A. Karpathy]:

IMAGENET

50 000 images, 1000 classes

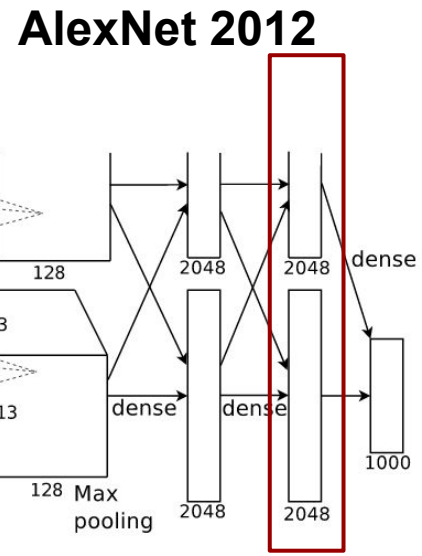


4096

extracted the 4096-dimensional fc7 CNN

What is t-SNE ?

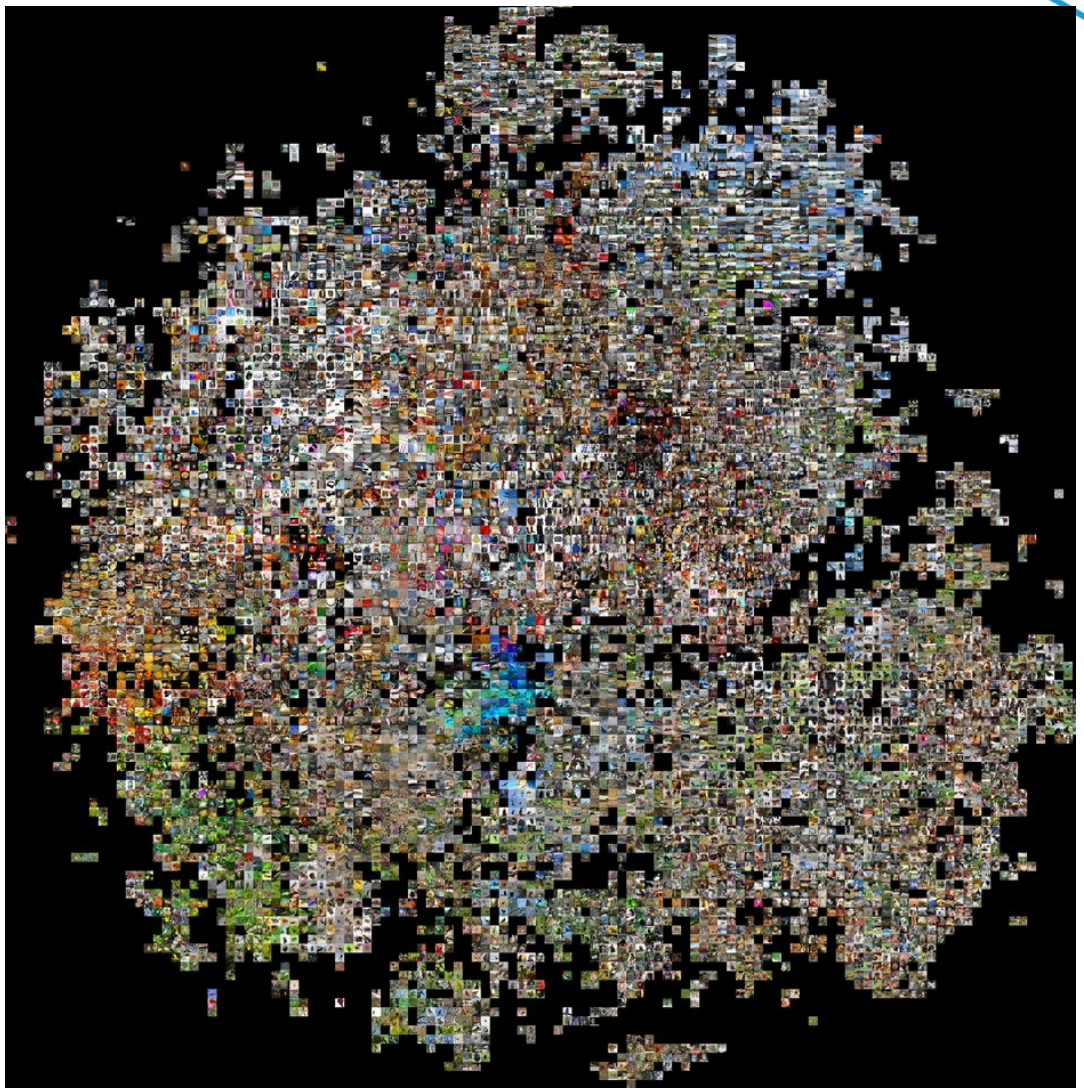
Another example by A. Karpathy:



4096

extracted the 4096-dimensional fc7 CNN

t-SNE



plot on 2d surface, to each point then an image is assigned

Note:

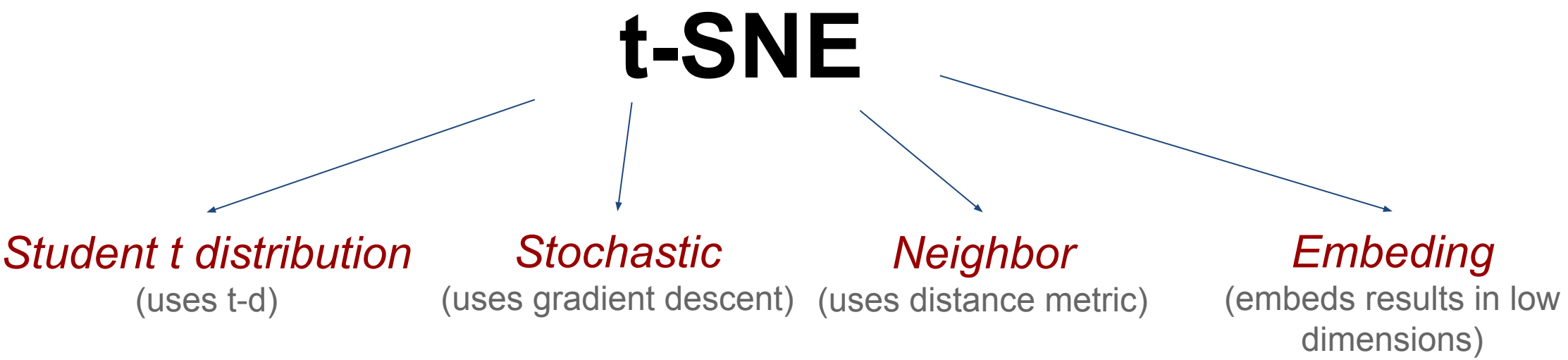
t-SNE, can be used to check if created features or model generalize well and reveal correlation between data.

What is t-SNE ?

In my opinion: A n-body problem where the force is defined with special metric

Similar objects are attracted, **dissimilar** opposite.

A theory behind t-SNE



A theory behind t-SNE

the theory will go with simple examples to get intuition (**FORTRAN**)

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

normalization

where σ_i is the variance of the Gaussian that is centered on datapoint x_i
we set the value of $p_{i|i}$ to zero

explained later

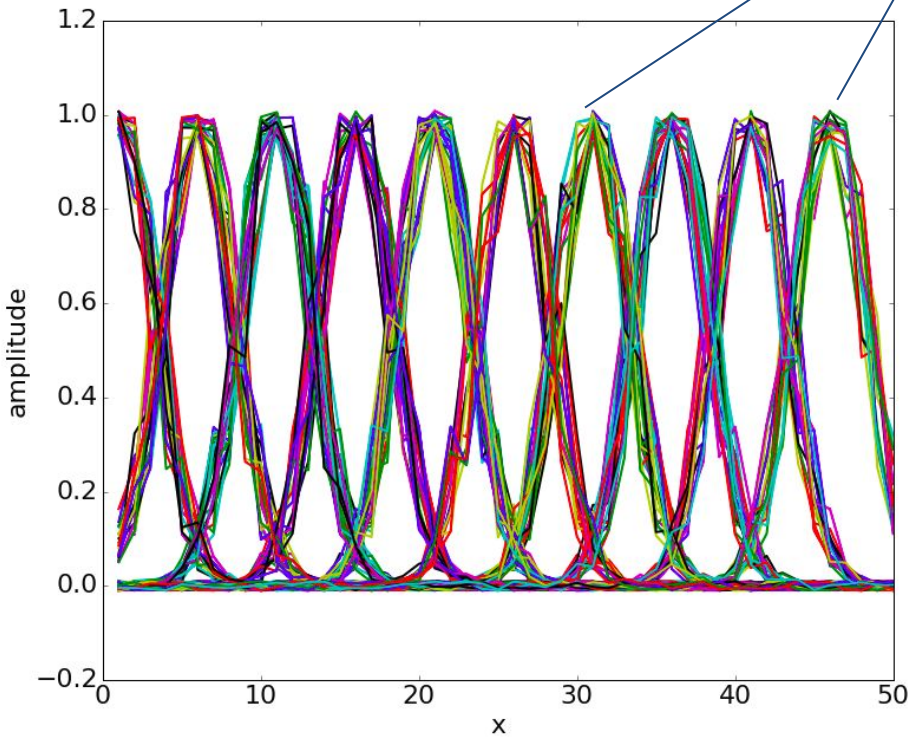
A theory behind t-SNE

the theory will go with simple examples to get intuition (**FORTRAN**)

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

normalization



Example data generated for testing:
we have 10 classes in 50 dimensional vectors.

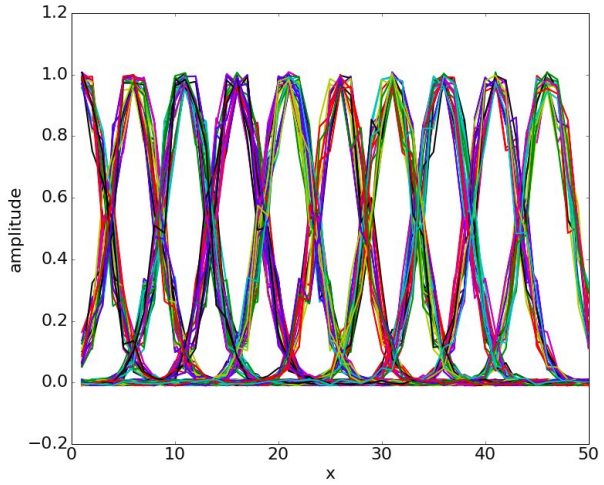
what we can see from the pij definition?

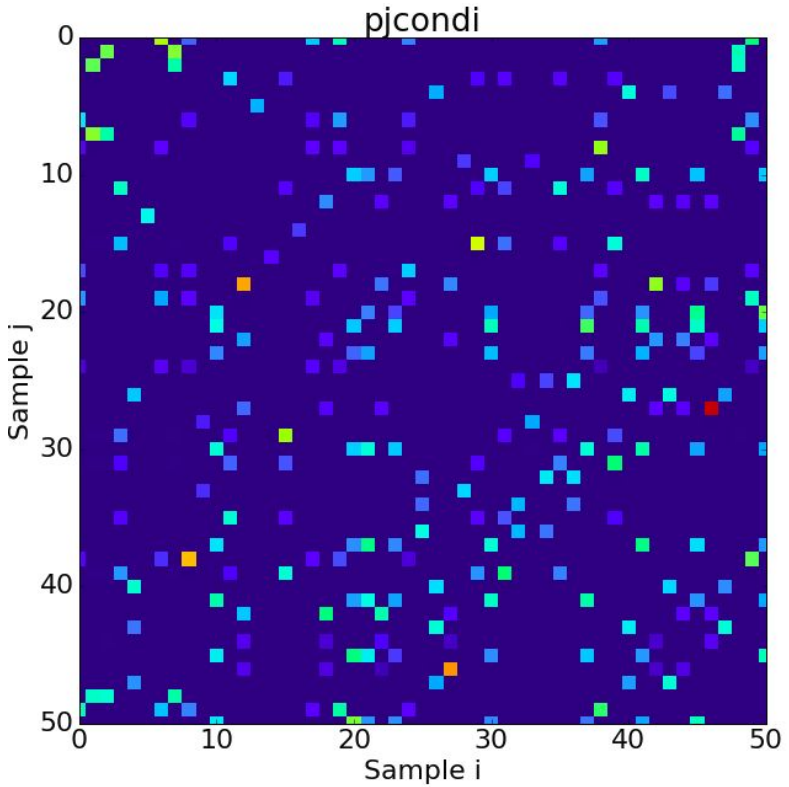
- we can globally shuffle elements of those vectors and not change pij. **Very GOOD!**
- **(in my opinion)**, we need vector elements to be of the same order, otherwise we may have dominating direction. **Probably** we need to normalize somehow input for tSNE. **Not GOOD!**
- pij - defined above is not symmetric, which physically is not correct…(**can be fixed**)

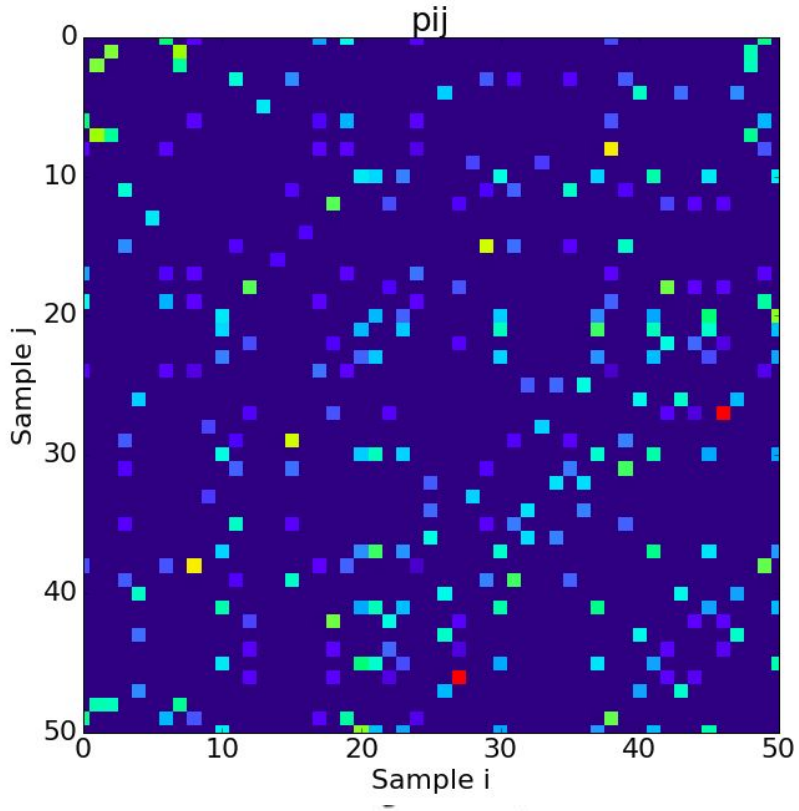
A theory behind t-SNE

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$







Lets then symmetrize it and normalize: sum(pij) = 1

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

A theory behind t-SNE

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

What about sigmas?

The value of **sigma_i** define neighborhood of surrounding vectors, **physically** it is an interaction distance.

each high-dimensional datapoint, x_i. It is not likely that there is a single value of σ_i that is optimal for all datapoints in the data set because the density of the data is likely to vary. In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions. Any particular value of

What t-SNE does?

perplexity = złożoność ?

Define sigma for each vector **x** such that the **perplexity** is constant for it, **physically** we want to have same number of neighbors for each x.

The perplexity can be interpreted as a smooth measure of the effective number of neighbors.

A theory behind t-SNE

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

```
subroutine computePCONDi(i)
  integer :: j,i,k
  real*8 :: denom
  do j = 1 , numSamples
    pjcondi(j,i) = exp(-norm2(X(i,:)-X(j,:))/(2*sigma(i)**2))
  enddo
  pjcondi(i,i) = 0
  pjcondi(:,i) = pjcondi(:,i) / sum(pjcondi(:,i))
end subroutine computePCONDi
```

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

What is perplexity?

(input parameter for t-SNE)

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

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

Shannon entropy

```
real*8 function getPerplexity(i) result(perp)
  integer :: i,j

  call computePCONDi(i)
  perp = 2 ** sum(-pjcondi(:,i)*log2(pjcondi(:,i) + 1D-06))
end function getPerplexity
```

A theory behind t-SNE

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

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

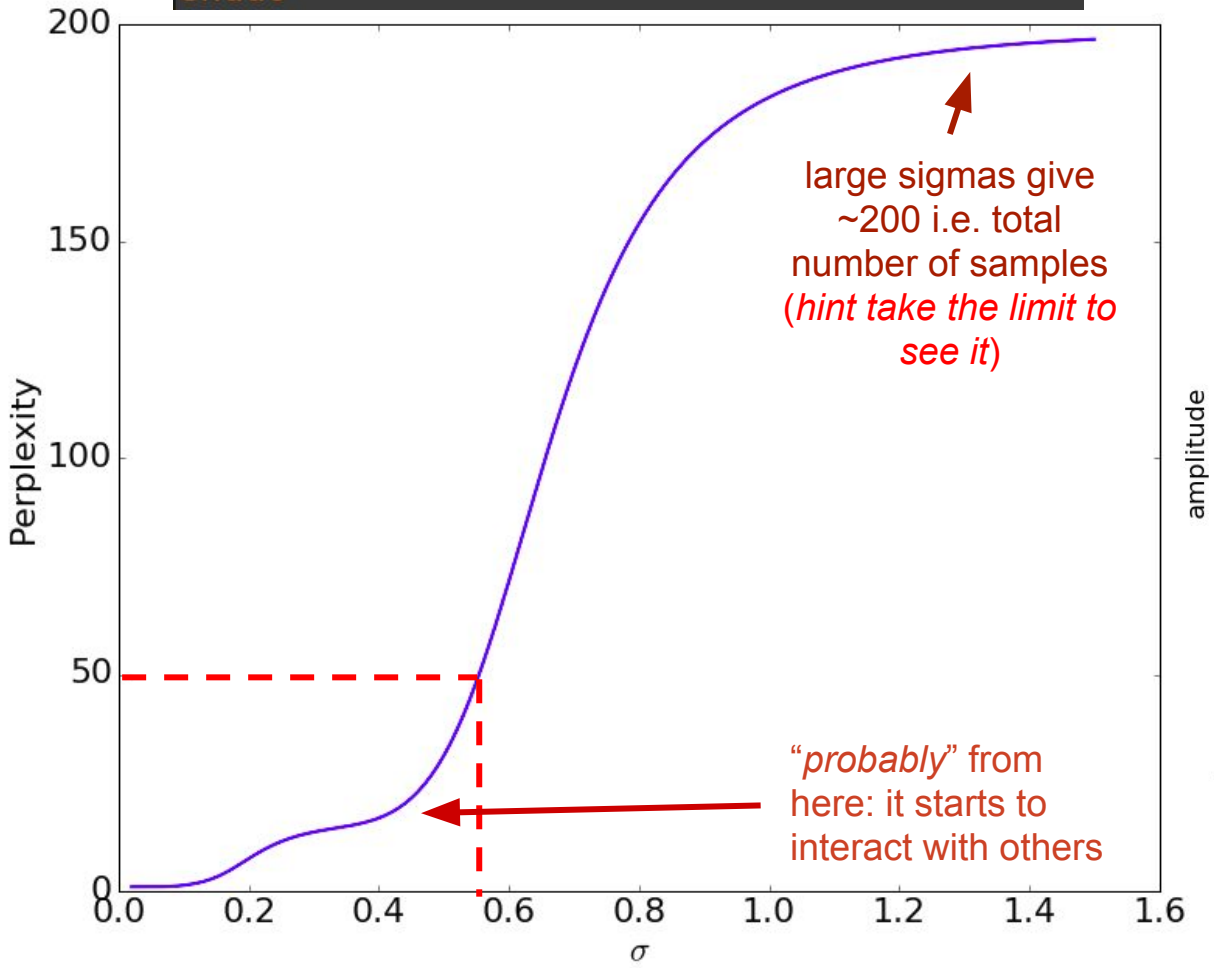
$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

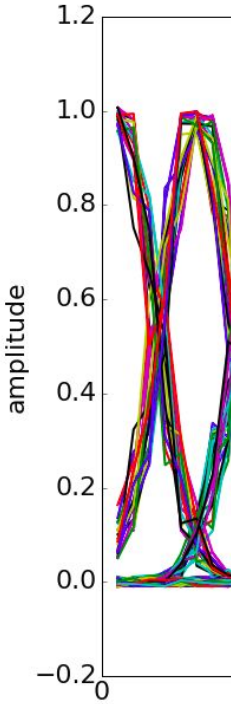
What t-SNE does?

SNE performs **binary search** for sigma that produces fixed value of perplexity for each X.

Note: This function is monotonic with sigma - binary search is safe

```
do sigma = 0.01 , 1.5 , 0.01
  dataXsigm(i) = sigma
  write(111,*), sigma , getPerplexity(i)
enddo
```



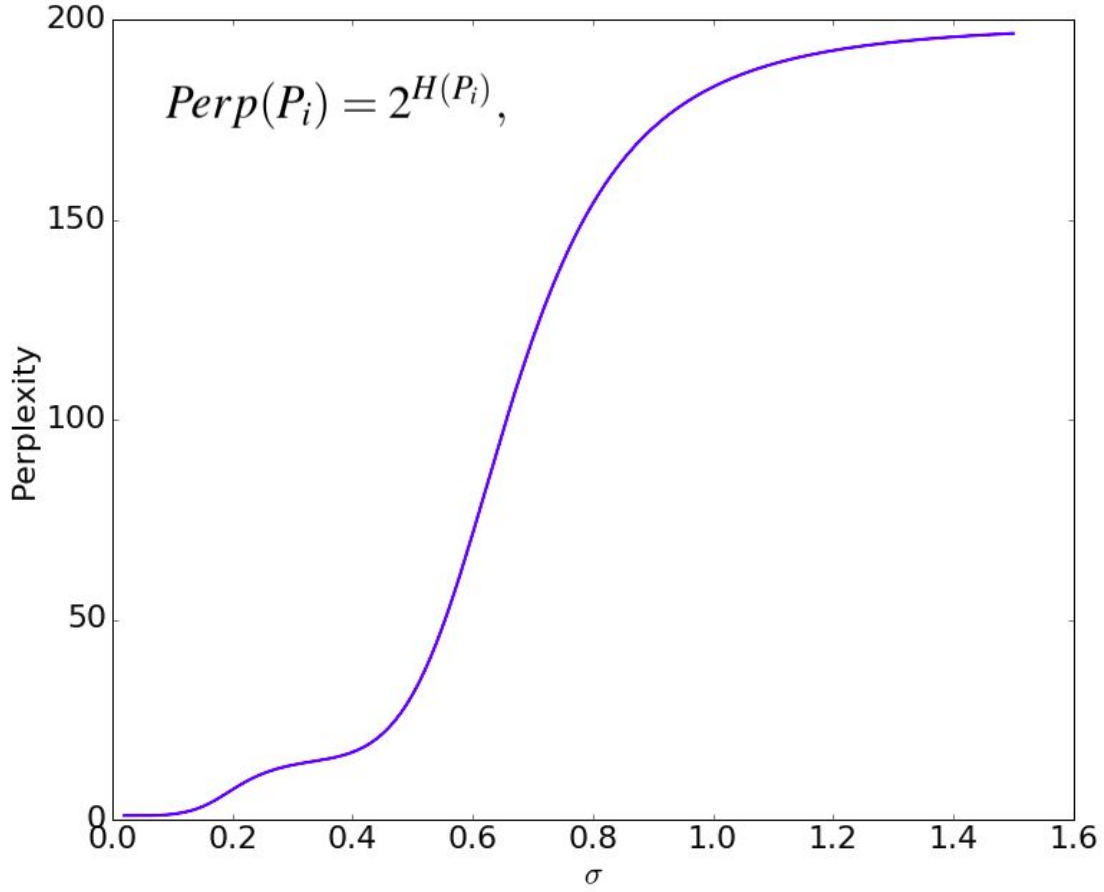


Perplexity in function of sigma for a selected vector

A theory behind t-SNE

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

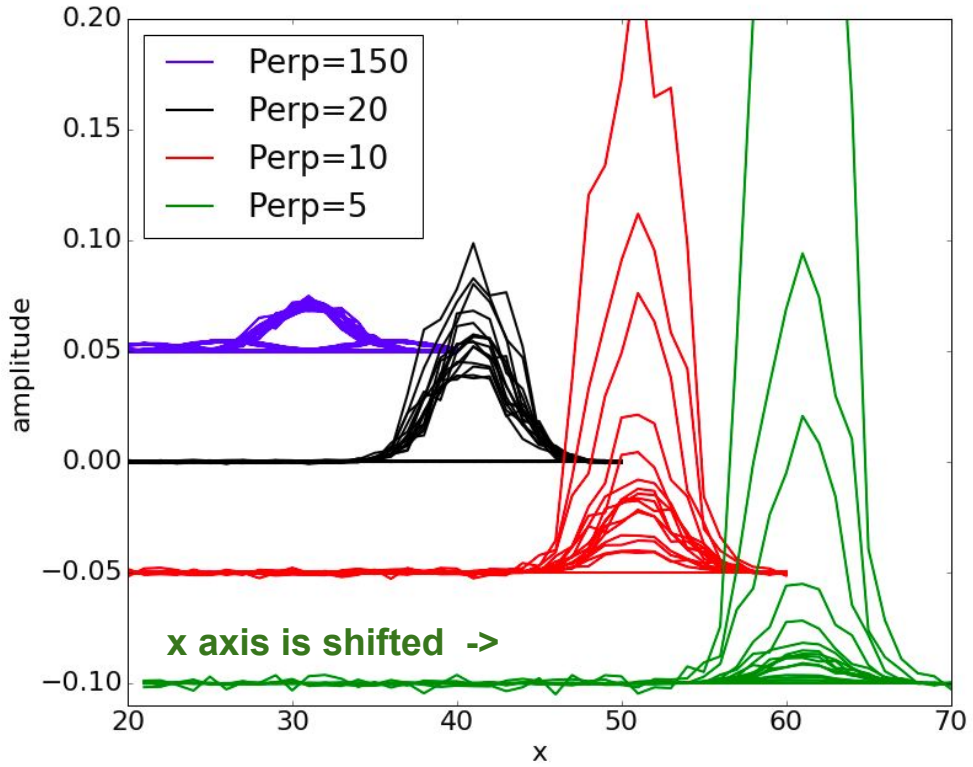
$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$



Perplexity in function of sigma for a selected vector

The effect of the perplexity on neighbor visibility:

```
do j = 1 , sampleDim
  write(111,"(5000e20.10)"),j+0.0,dataX(:,j)*pjcondi(:,i)
end do
```



What i-th vector is seeing around? For large sigmas all samples become equally visible.

A theory behind t-SNE

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

The value of perplexity really matters:

Nice visualizations: http://distill.pub/2016/misread-tsne/ !!!

A theory behind t-SNE

- We convert high dimensional distances to conditional probabilities of picking j when i has sigma_i:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)},$$

Calculation of pij probabilities is the first step for t-SNE:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \qquad Perp(P_i) = 2^{H(P_i)}, \qquad H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}.$$

- **Finally the second step:** we introduce another set of variables Y, of the same number of samples what X but is low dimensional. **In practice 2 or 3** dimensional vectors e.g. **$Y = (y_1, y_2)$ for visualization**

We define joint probabilities as:

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

Student t-distribution with one degree of freedom: Cauchy distribution

TL;TW: this kind of approximation has nice properties, one of them is **long tail** (for long range interactions), is similar to Gauss, faster to compute (no exponential). **See paper for more details**.

NOTE !!!

- no sigmas (model simplification, perplexity recomputation for each iteration step would be difficult)
- of course we put $q_{ii} = 0$

A theory behind t-SNE

- **Finally the second step:** we introduce another set of variables Y, of the same number of samples what X but is low dimensional. **In practice 2 or 3** dimensional vectors e.g. **$Y = (y_1, y_2)$ for visualization**

We define joint probabilities as:

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

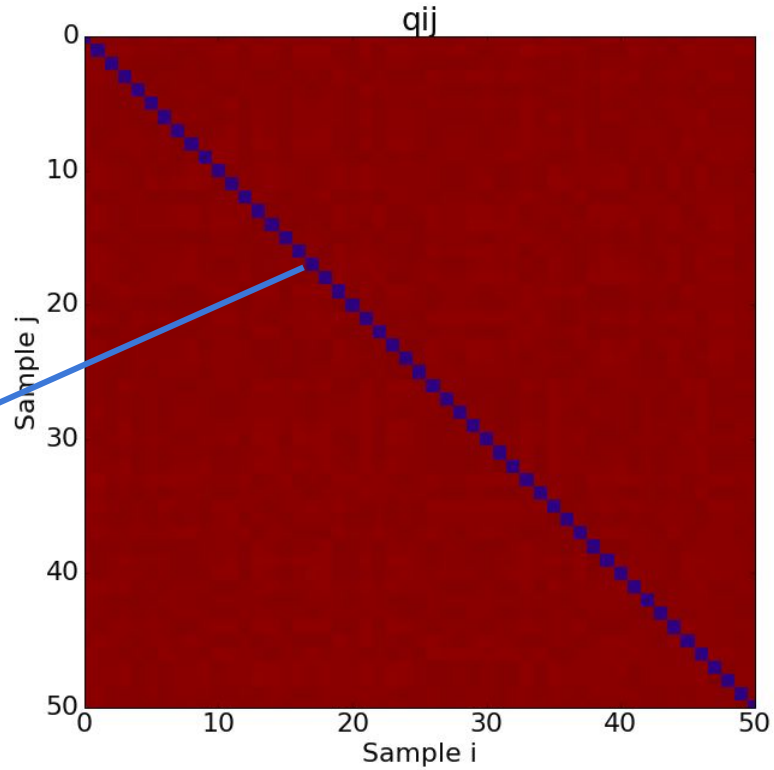
Student t-distribution with one degree of freedom: Cauchy distribution

$\mathbf{q_{ij}}$ is initialized with random values:

```
do i = 1,  numSamples
  Y(i,:) = (/ rand()-0.5 , rand()-0.5 /)/100.0
enddo
```

```
subroutine updateQij()
  integer :: i,j
  do i = 1 , numSamples
    do j = i+1 , numSamples
      qij(i,j) = 1/(1 + norm2(Y(i,:)-Y(j,:)))
      qij(j,i) = qij(i,j)
    enddo
    qij(i,i) = 0
  enddo
  qij = qij / sum(qij)
end subroutine updateQij
```

$q_{ii} = 0$



note the $O(N^2)$ complexity

A theory behind t-SNE

- Very basic assumption:

We want to have $\mathbf{p_{ij}}$ to be equal to $\mathbf{q_{ij}}$ i.e. low dimensional vectors **Y** correctly model similarities between high-dimensional data **X**.

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

but how to do it?

=

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

We can define a measure between two distributions which will tell as how close those distributions are:

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

Kullback–Leibler divergence

We are going to minimize **C** with respect to **Y** vectors

Notes:

- C is asymmetric
- q_{ij} is force to be large as p_{ij}
- but q_{ij} cannot be larger than one, due to normalization, hence stability

$\mathbf{p_{ij}}$ are computed once in the first stage of **t-SNE**

A theory behind t-SNE

- **Gradient descent**:

We want to have $\mathbf{p_{ij}}$ to be equal to $\mathbf{q_{ij}}$ i.e. low dimensional vectors **Y** correctly model similarities between high-dimensional data **X**.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \qquad q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}.$$

We minimize **Kullback–Leibler** divergence

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

gradient step

learning rate

momentum term

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right),$$

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \left(1 + \|y_i - y_j\|^2\right)^{-1}$$

gradients derived in original paper

A theory behind t-SNE

- **Relation to physics**

gradients derived in original paper

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$

mass term: may be positive or negative

force direction

force dumping

PHYSICS BELOW!

- **Force from gravity:**

$$\mathbf{F}_{ij} = \frac{G m_i m_j (\mathbf{q}_j - \mathbf{q}_i)}{\|\mathbf{q}_j - \mathbf{q}_i\|^3},$$

- **Total force which acts on particle i**

$$m_i \frac{d^2 \mathbf{q}_i}{dt^2} = \sum_{j=1, j\neq i}^{N} \frac{G m_i m_j (\mathbf{q}_j - \mathbf{q}_i)}{\|\mathbf{q}_j - \mathbf{q}_i\|^3} = \frac{\partial U}{\partial \mathbf{q}_i}$$

From wikipedia: *N-body problem*

second order

How does it connect to Gradient descent method:

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right),$$

A theory behind t-SNE

- **Relation to physics**

$$\frac{\delta C}{\delta y_i} = 4\sum_i (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta\frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right),$$

From basic physics we have: acceleration is proportional to Force

$$A = \frac{d^2Y}{dt^2} = F(t)/m \equiv \frac{\delta C}{\delta y} \qquad \begin{aligned} A &= \frac{dV}{dt} \\ V &= \frac{dY}{dt} \end{aligned}$$

Finite difference approximation gives:

$$A^{(t)} \approx \frac{V^{(t)} - V^{(t-1)}}{\Delta t} \Longrightarrow V^{(t)} = V^{(t-1)} + \Delta t\frac{\delta C}{\delta y}^{(t)}$$

Let's do the same with coordinates Y:

$$Y^{(t)} = Y^{(t-1)} + \Delta t V^{(t)} \qquad Y^{(t)} = Y^{(t-1)} + \Delta t\left(V^{(t-1)} + \Delta t\frac{\delta C}{\delta y}^{(t)}\right)$$

$$\begin{aligned} Y^{(t)} &= Y^{(t-1)} + \Delta t^2\frac{\delta C}{\delta y}^{(t)} + \Delta t V^{(t-1)} \\ &= Y^{(t-1)} + \Delta t^2\frac{\delta C}{\delta y}^{(t)} + \left(Y^{(t-1)} - Y^{(t-2)}\right) \end{aligned}$$

Looks familiar ;)

A theory behind t-SNE

- **Relation to physics**

$$\frac{\delta C}{\delta y_i} = 4\sum_i (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta\frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right),$$

Force

From basic physics we have: acceleration is proportional to Force

$$A = \frac{d^2Y}{dt^2} = F(t)/m \equiv \frac{\delta C}{\delta y}$$

$$Y^{(t)} = Y^{(t-1)} + \Delta t^2 \frac{\delta C}{\delta y}^{(t)} + \left(Y^{(t-1)} - Y^{(t-2)}\right)$$

without alpha this equation will not give a **steady state**! We need dumping trick **alpha=0.9**

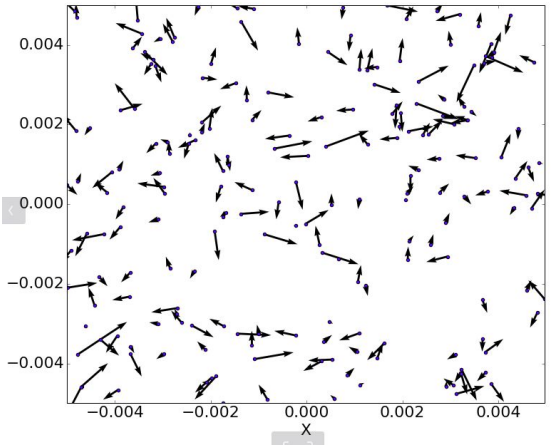
t-SNE solves this, only the force is different.

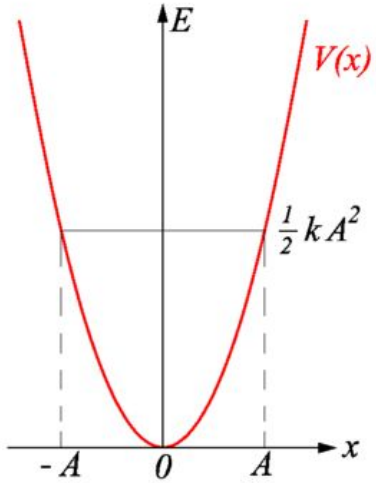
Optimization Methods for t-SNE

- **We do gradient descent**

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right),$$

  - Adaptive learning rate: Jacobs (1988) (*see tSNE paper Sec 3.4*)
  - Dimensionality reduction on X before processing (**sklearn** has PCA for initial **Y** vectors)
  - add L2 norm (**"early compression"**)





derivative of this gives force which keeps particles in the middle increasing mixing between them

- **"early exaggeration"** - increase interaction between particles by multiplying p_{ij} by some number e.g. 4 in the initial stages of the optimization

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$

mass term is always positive it so force is attractive, natural clusters arises

Vanilla implementation in Fortran

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right),$$

minus

```
subroutine tSNEsolver(eta,reg,T)
  real*8 ,intent(in) :: eta, reg
  integer,intent(in) :: T
  integer :: i
  real*8  :: alpha, exgg , earlycmp

  do i = 1, T
    alpha = 0.8 ; if(i < 250) alpha    = 0.5
    exgg  = 1   ; if(i < 50)  exgg     = 4.0

    call updateQij
    call computeGradients(exgg)

    dataYtmp = dataY &
             - eta * dCdy &
             - reg * dataY &
             + alpha*(dataY - dataYold)

    dataYold = dataY
    dataY    = dataYtmp
  enddo
end subroutine tSNEsolver
```

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$$

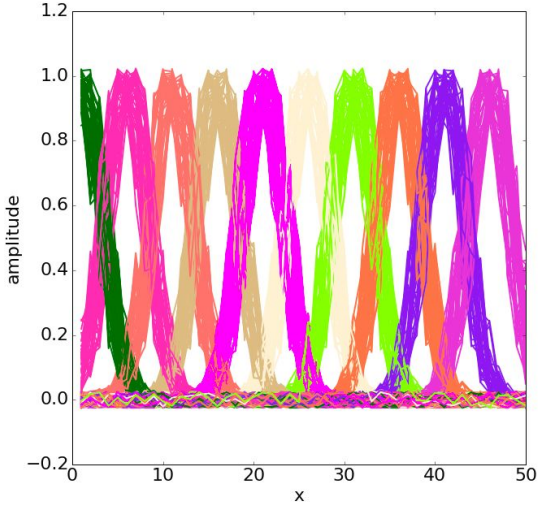
```
subroutine computeGradients(exgg)
  real*8 :: exgg
  integer :: i,j
  real*8 ,dimension(:) , pointer :: yi,yj
  do i = 1, numSamples
    dCdy(i,:) = 0
    yi => dataY(i,:)
    do j = 1 , numSamples
       yj => dataY(j,:)
       ! gradient
       dCdy(i,:) = dCdy(i,:) &
                 + 4*(exgg * pij(i,j) - qij(i,j)) & ! MASS
                 * ( yi - yj ) &                    ! DIRECTION
                 * (1 + norm2(yi - yj))**(-1)       ! DUMPING
    enddo
  enddo
end subroutine computeGradients
```

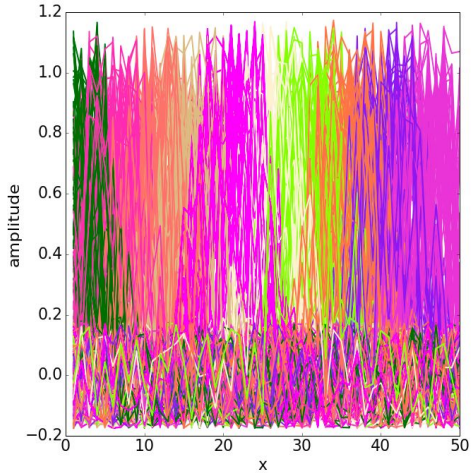
Unfortunately:

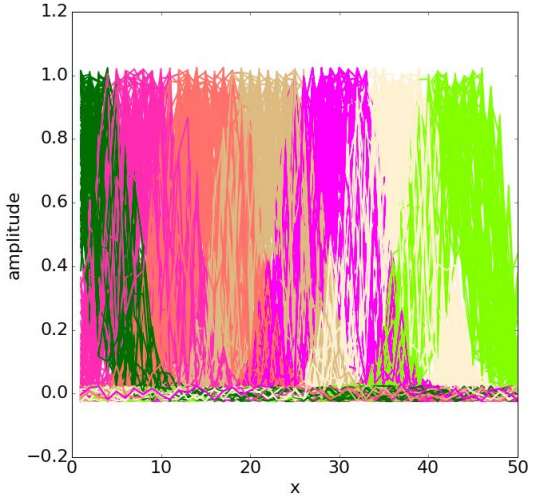
This has complexity $O(N^2)$:(

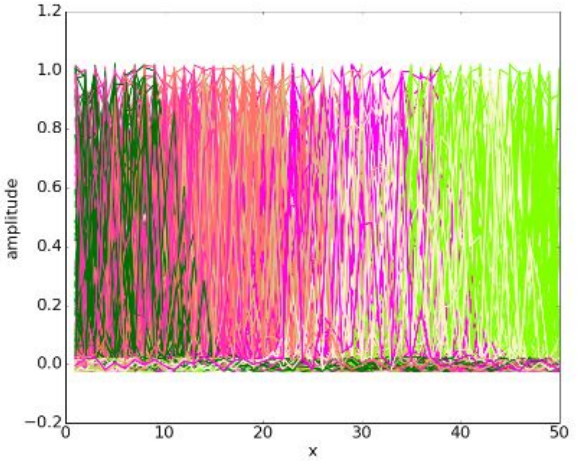
Vanilla examples

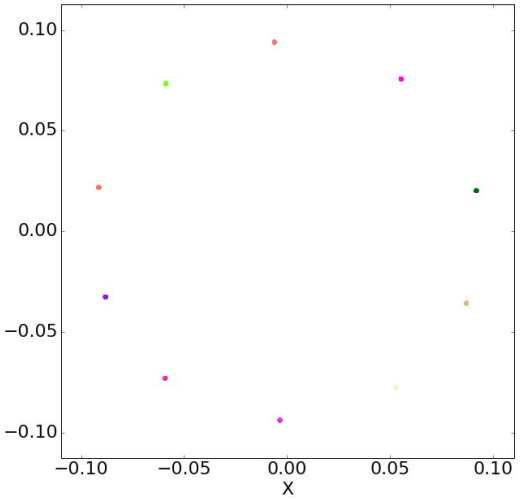
Well correlated data:

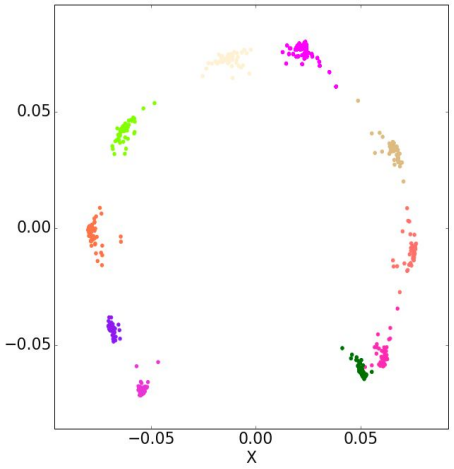


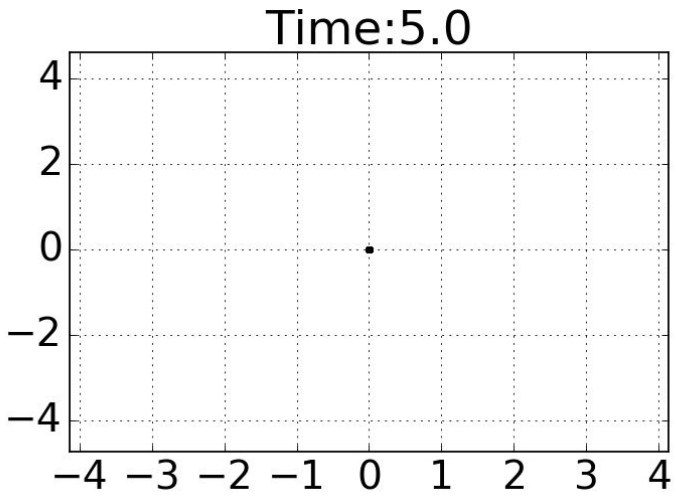


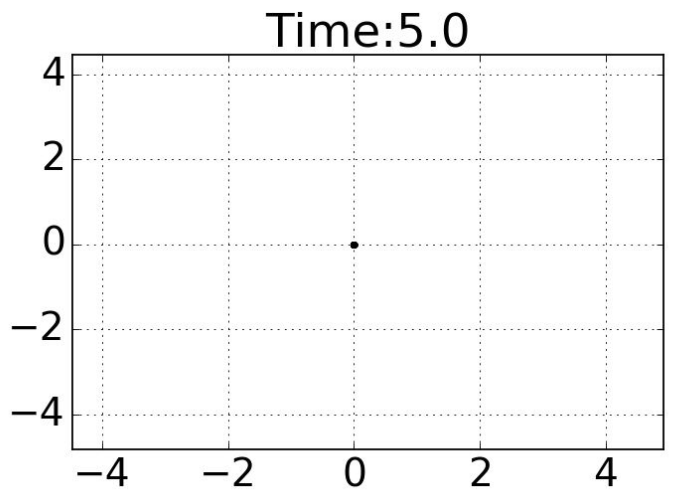


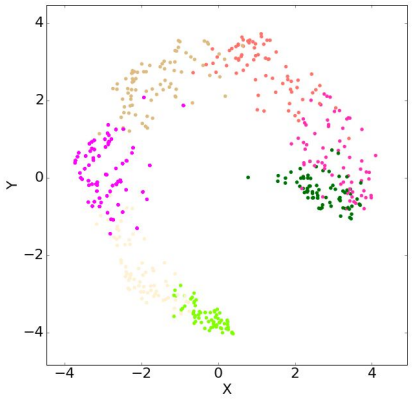












Other extensions and improvements

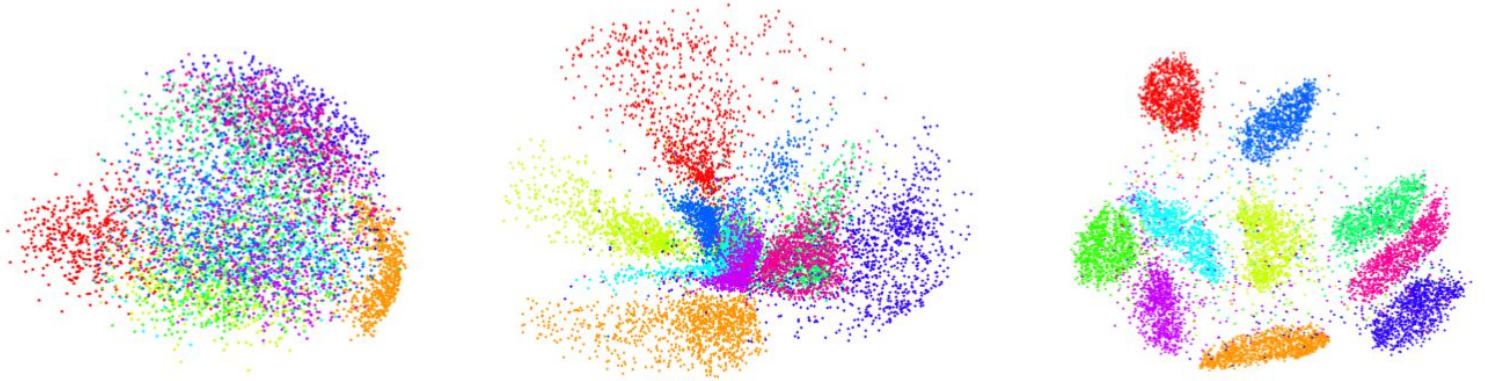
- L.J.P. van der Maaten and G.E. Hinton. **Visualizing High-Dimensional Data Using t-SNE**. *Journal of Machine Learning Research* 9(Nov):2579-2605, **2008**. **PDF** **[Supplemental material]** **[Talk]**

So far we have discussed this paper.

Cited by 1806

- L.J.P. van der Maaten. **Learning a Parametric Embedding by Preserving Local Structure**. In *Proceedings of the Twelfth International Conference on Artificial Intelligence & Statistics (AI-STATS), JMLR W&CP* 5:384-391, 2009. PDF

Connect output to neural network with t-SNE and minimize C in order to create dimensionality reduction algorithm.



(a) Visualization by PCA. (b) Visualization by an autoencoder. (c) Visualization by parametric t-SNE.

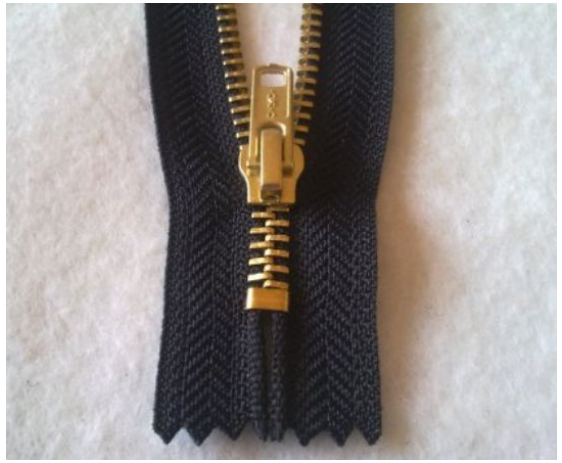
Other extensions and improvements

- L.J.P. van der Maaten and G.E. Hinton. **Visualizing Non-Metric Similarities in Multiple Maps**. *Machine Learning* 87(1):33-55, 2012. PDF

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}, \quad \text{for } \forall i \forall j : i \neq j,$$

$$q_{ij} = \frac{\sum_m \pi_i^{(m)} \pi_j^{(m)} (1 + \|\mathbf{y}_i^{(m)} - \mathbf{y}_j^{(m)}\|^2)^{-1}}{\sum_k \sum_{l \neq k} \sum_{m'} \pi_k^{(m')} \pi_l^{(m')} (1 + \|\mathbf{y}_k^{(m')} - \mathbf{y}_l^{(m')}\|^2)^{-1}},$$

zamek



zamek

zamek

???

multimap t-SNE can produce multiple maps which will maximize local structure. Number of maps is a parameter.

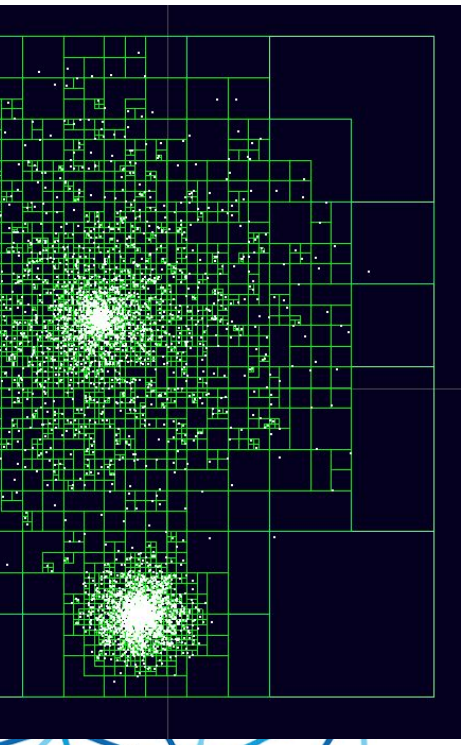
Other extensions and improvements

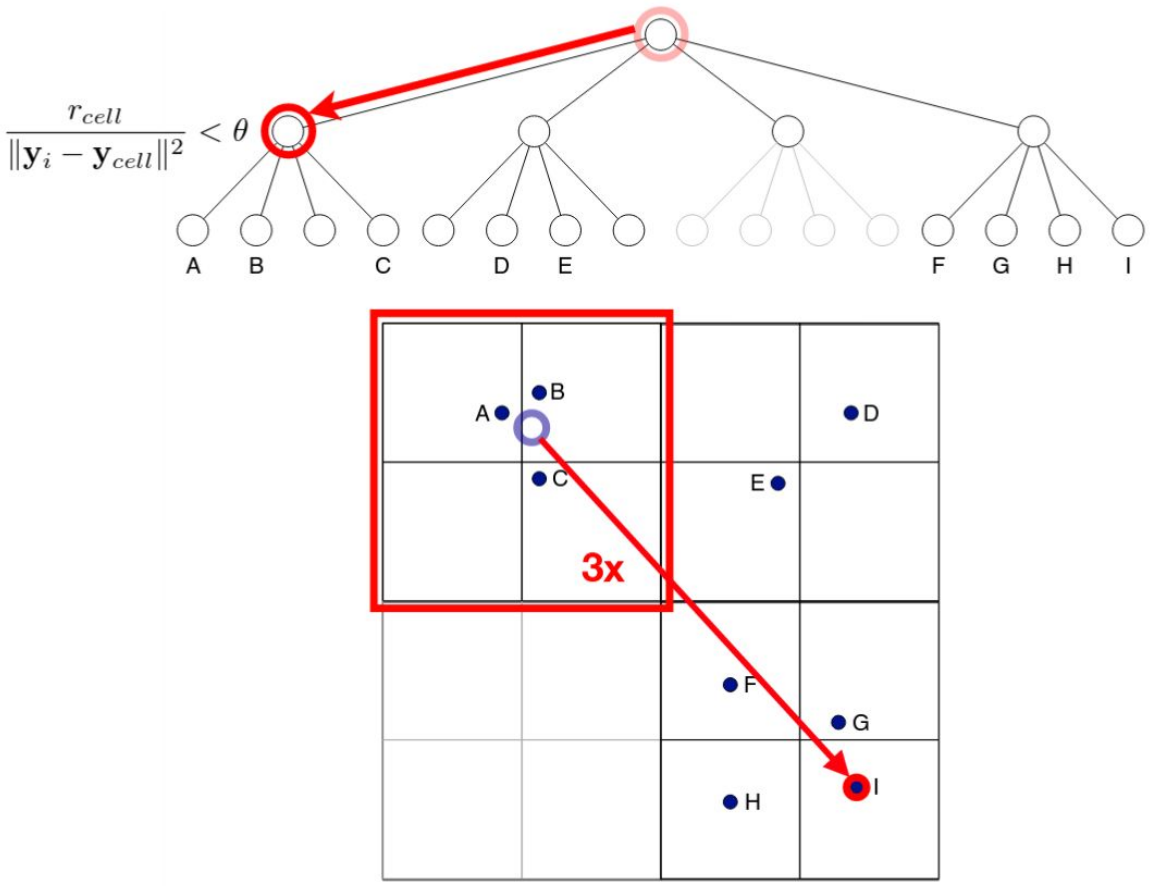
L.J.P. van der Maaten. **Accelerating t-SNE using Tree-Based Algorithms**. *Journal of Machine Learning Research* 15(Oct):3221-3245, **2014**. PDF

First t-SNE paper in 2008

Barnes–Hut simulation (1986): O(*n* log *n*)

wikipedia





Diagonal of cell

center of the mass

$$\frac{r_{cell}}{\|\mathbf{y}_i - \mathbf{y}_{cell}\|} < \theta,$$

sklearn - theta is called angle, LOL

"Angle less than 0.2 has quickly increasing computation time and angle greater 0.8 has quickly increasing error" - sklearn

Other extensions and improvements

L.J.P. van der Maaten. **Accelerating t-SNE using Tree-Based Algorithms**. *Journal of Machine Learning Research* 15(Oct):3221-3245, **2014**. PDF

https://github.com/DmitryUlyanov/Multicore-TSNE

DmitryUlyanov / **Multicore-TSNE**

Code | Issues 4 | Pull requests 0 | Projects 0

Parallel t-SNE implementation with Python and Torch wrappers.

This is a benchmark for `70000x784` MNIST data:

Method	Step 1 (sec)	Step 2 (sec)
MulticoreTSNE(n_jobs=1)	**912**	**350**
bhtsne	4257	1233
py_bh_tsne	1232	367
sklearn(0.18)	~5400	~20920

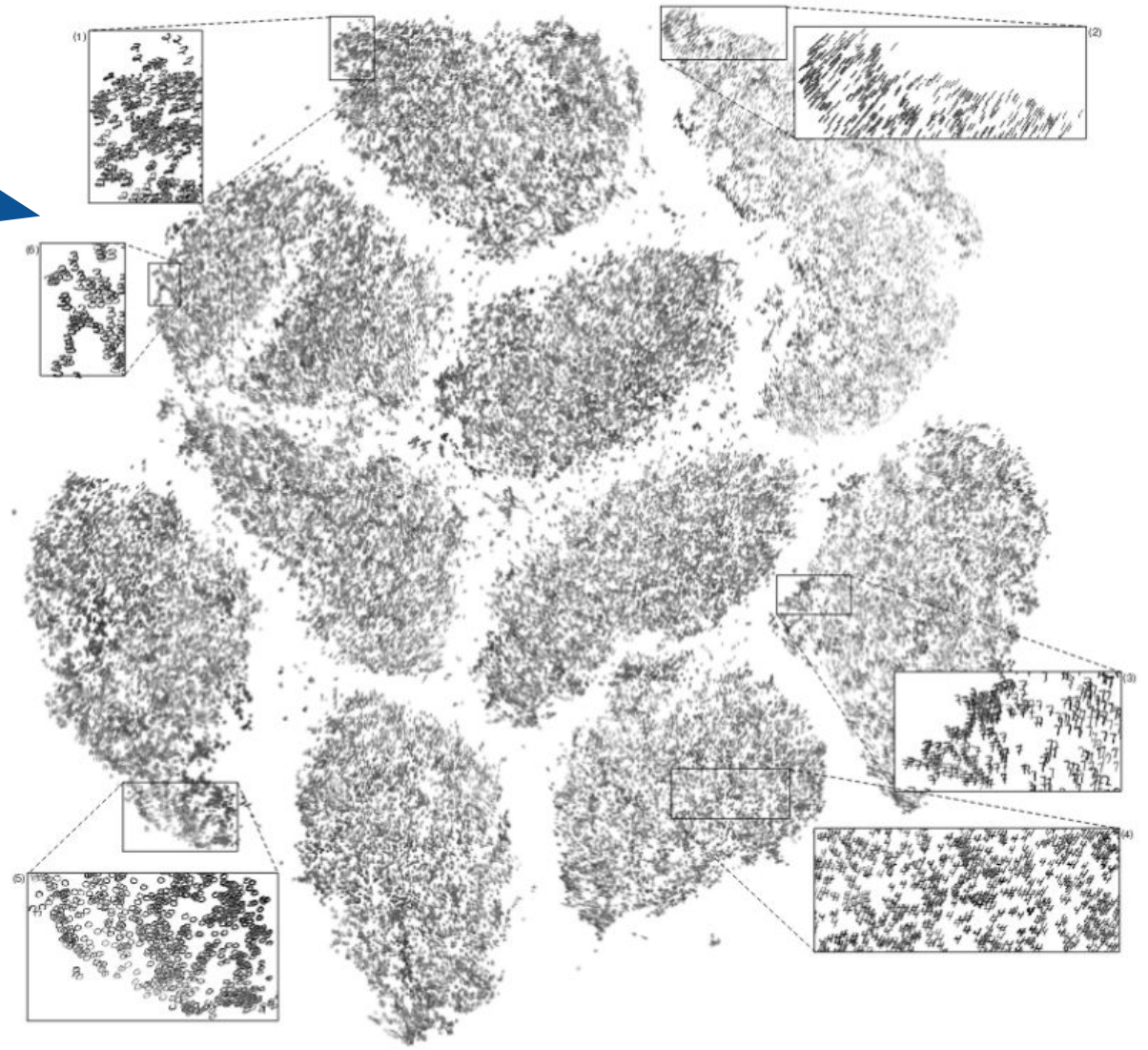


Figure 8: Barnes-Hut t-SNE visualization of all 70,000 MNIST handwritten digit images (constructed in 10 minutes and 45 seconds using $\theta = 0.5$). The insets (from the

t-SNE problems

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

- Results hard to interpret (cluster size, cluster distance do not resemble to actual ones)
- **Non deterministic** - results depend on initial conditions
- **non-parametric mapping** - having final solution we cannot embed new points into resulting map (parameteric t-SNE ?)
- (*in my opinion*) lack of state-of-art implementation (multicore on OMP, CUDA without B-H). *Quick search gives:*

CHAPTER

6

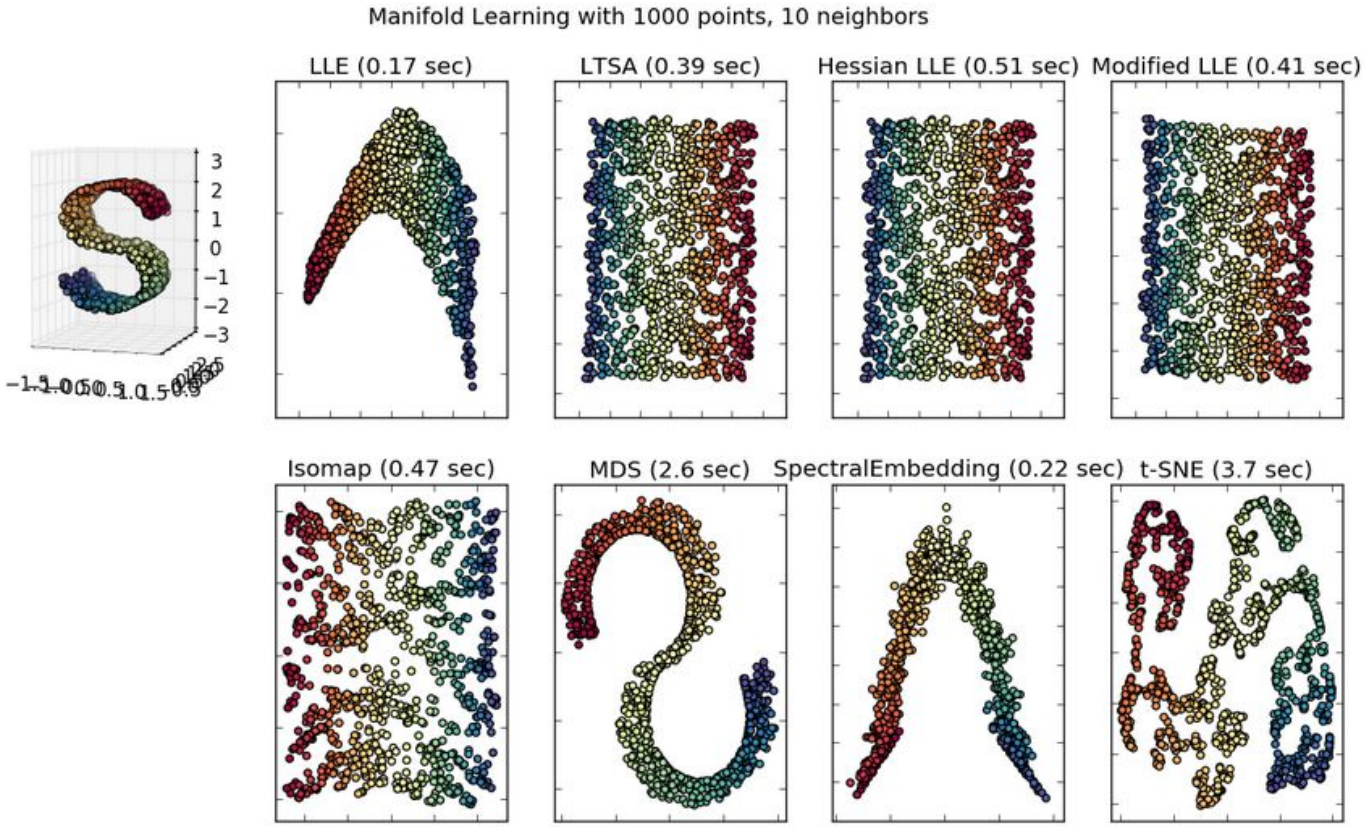
An Efficient CUDA Implementation of the Tree-Based Barnes Hut *n*-Body Algorithm

Martin Burtscher, Keshav Pingali

t-SNE problems

- **sklearn** - most general, but slow and unstable (in my case), no parametric, no multimap, a lot of parameters...

class `sklearn.manifold.` **TSNE** (*n_components=2, perplexity=30.0, early_exaggeration=4.0, learning_rate=1000.0, n_iter=1000, n_iter_without_progress=30, min_grad_norm=1e-07, metric='euclidean', init='random', verbose=0, random_state=None, method='barnes_hut', angle=0.5*) ¶ [source]



References

http://distill.pub/2016/misread-tsne/ nice visualization for intuition with t-SNE
https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf - orginal papaer
https://github.com/DmitryUlyanov/Multicore-TSNE - multicore OMP implementation of t-SNE
https://lvdmaaten.github.io/tsne/ author's github page
http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html tsne on sklearn

FORNAX

WWW.FORNAX.CO