t-SNE

* **High Dimensional Data**

- can be predicted by 30+ features.
- High dim data is hard to visualize & work with
- Embedding to low dim spaces helps visualize the data

* **Earlier Techniques - Direct Visualization**

• Parallel co-ordinates:

- allows for comparison of multiple data records by using parallel lines to connect points based on multiple numerical variable.

• Chernoff Faces:

- Symbolizing data using faces
- each emoji has an equ emotion

Ex: LA life : emoji - emotion - ranked categories

- Direct methods does not preserve ordinal nature of features.

Distance Preservation MDS, Isomap

Topology/geometrical Preservation Isomap

Information preservation PCA

capturing most of the variance

MNIST dataset

images → 0 1 2 ... 9 8x8 pixels

High Dim Data — A selection from the 64 dimensional digits dataset.

Ex: MDS: Colors have been clustered well, performed better than PCA.

✓ ISOMAP: clearly made clusters of all digits clearly.

unsupervised → all dimensionality techniques.

Isomap - preserves global structure & focusses on geometry of data. the geodesic distance, which can be useful for understanding the underlying manifold or shape of the data

- computationally expensive (∵ size of dataset or dim ↑)
- Also requires careful parameter selection for optimal results. (like # of neighbours).

Q- what to do when understanding local relationships is crucial, such as nbd relationships?

Ans: t-SNE (only looked locally).

- t-SNE provides better visualizations than other methods.
- It helps to uncover patterns, clusters & relationships in the data

* Stochastic Neighbour Embedding (SNE)

→ An unsupervised technique which focusses on preserving nbd's, instead of preserving distances.

Goal: To find a low-d map ... difference b/w the P (high-d) & Q (low-d) distributions.

* t-SNE Algo — looking at neighbors

Ex: 8x8 pixel ⇒ 64 dimensions ⇓ 1 Sample.

Data: data set $X = \{x_1, x_2, x_3, \ldots x_n\}$

cost fn parameters: perplexity Perp.

optimization parameters: # of iterations T, learning rate η, momentum $\alpha(t)$

Result: low-dim data representn $Y^T = \{y_1, y_2, \ldots y_n\}$

begin:

compute pairwise affinities $p_{j|i}$ with perplexity Perp (using eqn 1)

set p_{ij} (or p_{ij}) $= \dfrac{p_{j|i} + p_{i|j}}{2n}$

Sample initial solution $Y^{(0)} = \{y_1, y_2, \ldots y_n\}$ from $\mathcal{N}(0, 10^{-4} I)$

for $t = 1$ to T do
  compute low-dimensional affinities q_{ij} (eqn 4)
  compute grad $\frac{\delta C}{\delta Y}$
  Set $Y^{(t)} = Y^{(t-1)} + \eta \frac{\delta C}{\delta Y} + \alpha(t)\left(Y^{(t-1)} - Y^{(t-2)}\right)$
end

end

→ The difference b/w the high-d & low-d maps are minimized using grad descent.

Compute probabilities P that x_i & x_j are neighbours in high-d space.

perplexity ⇒ how many neighbors
(Measures the effective # of neighbors, usually b/w 5 & 50).

Optimization ⇒ first define how many times we want to improve (usually 5000).
Parameters to speed up optimization & avoid poor local minima.

# of iterations: Number of steps need to reach the optimization as GD is iterative.

learning rate: GD steps in the dirn the error is the min. η defines how big will be the step.

$\alpha(t)$: Momentum encourages a step that is in the same direction as previous steps.

$q_{ij} - p_{ij} = C$ ← difference

Prob distribution of samples
How far am I from original prob distr^n in high-d.
Slope (in which direction

1) Random guess
2) Probability distribution
3) distances (difference as small as possible)
4) Slope (dir^n in which diff ↑)
5) learning rate: step size

compute pairwise affinities $p_{j|i}$

↳ The probability that x_i would choose x_j as its neighbours in the high-d space.

x_i that are close (low euclidean dist) returns a high value.

x_i that are far (high euclidean dist) returns a lower value.

σ is the variance, change of variance will change the values we assign to the distances.

Var will be different (∵ clusters points are very close to each other.)

(using eqn 1)

The probability that x_i would choose x_j as its neighbours, in the high-d space.

Set p_{ij}

→ The low-d points are moved around to minimize the difference b/w the two distributions.

→ we find a low-d represen^n that captures the high-d data after successive iterations.

Perplexity

optimization

scale divergence Kullback - Leibler

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

↳ calc diff b/w 2 prob distributions

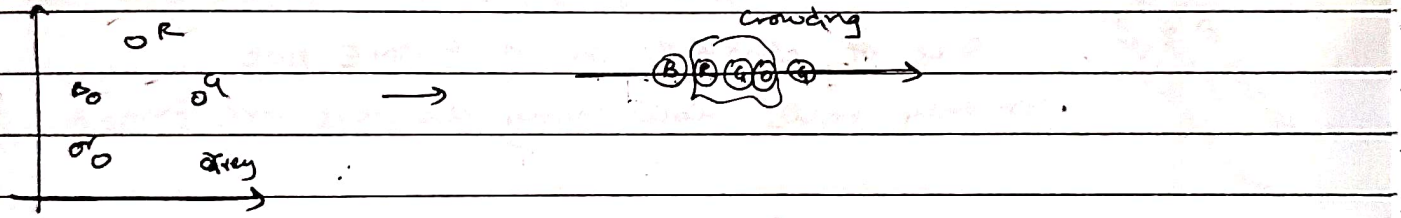
* Gaussian vs Student t-distribution

→ Student's have longer tails compared to Gaussian.

→ Gives high probabilities to points that are further away.

→ Desirable, as we have limited low-d space & want to focus on modelling the close high-d points.

Crowding in SNE



probability distribution

* Dependence on hyperparameters

perplexity	2	5	30	50	100
step	5000	5000	5000	5000	5000

# of datapoints & steps ⇒ similar

(always work with values 5-50 for perplexity)

low values → noises

high values → more than # of points.

projecting in low dim, → crowding
can't give imp to distances

perplexity	30	30	30	30	30
step	10	20	50	120	1000

seperated clusters

SNE → projecting in low dimension

clusters with different standard deviations & sizes

↳ has seperated ~~the~~ into clusters
↳ from original graph, we cannot ~~see~~ relative size of clusters in a t-SNE plot.
↳ only tells how many clusters are formed.

spread of clusters
how close/far

SNE → • good for interpreting # of clusters
• not good for understanding size, order of closeness in low & high.

• Fine tuning perplexity

* Take home Points:

understanding t-SNE → Data Visualization
• Mainly used for visualization purposes, not dim reduct

Running t-SNE
→ always run multiple trials
→ use apt perplexity
→ let the samples stabilize (iterations).

classmate
Date
Page

Reading t-SNE

- Do not give importance to distances b/w far away points
- Do not give importance to density of clusters.
- Do not infer anything from a single o/p.