T Distributed Stochastic Neighbourhood Embedding

- A dimension reduction method that tries to maintain high-dimensional patterns
- Usually used to visually represent high-dimensional patterns in fewer dimensional spaces, by maintaining near and far distances between nodes in both datasets
- This is done by matching **distance distributions** in both spaces using **conditional probabilities**
    - We assume distances in both spaces are distributed using **Gaussian distributions**

Details

- We start by placing the nodes in random positions in our low dimensional space
- It's an **iterative algorithm -** each iteration t-SNE moves the nodes depending on "forces" exerted on them by it's neighbouring nodes in the low dimensional space, with their forces dependant on their relative distance in higher dimensions
    - This means we need some way to gauge the distance between pairs of points

Iterative Process

- **Simplified**
    - For each point, we calculate the distances between the source point $\mathbf{x_i}$ and its neighbours and plot them on a Gaussian distribution centred at our source point
        - For each pair of points, we look at $\mathbf{x_j}$'s **y-value** on the Gaussian distribution, called a **similarity**
        - Close points will have higher values than distant points
- $$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$
  $$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k\neq i} exp(-||y_i - y_k||^2)}$$
    - $\mathbf{p_{j|i}}$ **-** our "closeness function" use in high dimensions, which computes the **similarity metric**
        - When $\mathbf{x_i}$ and $\mathbf{x_j}$ are similar, the norm is close to 0, the exponent of which will be **close to p = 1**
        - When they are different, the norm will be high, and because we make it negative, the exponent of a very negative number is going to be very small, thus **close to p = 0**
    - $\mathbf{q_{j|i}}$ **-** used in lower dimensions, where we omit the $2\sigma^2$ term as we assume the variance in lower dimensions is $1/\sqrt{2}$, so it will cancel out when you plug int into the term above
- We then try to make our distribution of $\mathbf{p_{j|i}}$ be as close to the distribution of $\mathbf{q_{j|i}}$ as possible
    - We make them similar by minimising a cost function that measures their **KL-divergence**
    - $$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log\left(\frac{p_{j|i}}{q_{j|i}}\right)$$
        - We optimise the cost function using **gradient descent**, using the gradient of D_{KL}
- $$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$
    - *By minimising the KL-divergence of our similarity measure, we maintain high dimensional structure*
- In this algorithm, **σ** is a **parameter** we set, and there are ways to choose a possibly optimal value
- We use another parameter, the **momentum term**
    - Speeds up optimisation and avoids local minimum by accelerating gradients moving in one direction
- $$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta\frac{\delta C}{\delta y} + \alpha(t)(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)})$$
  $\alpha(t)$: Momentum at iteration t
  $\mathcal{Y}^{(t)}$: Solution at iteration t
  η: Learning rate
    - This is the function we use to **update our y**

t-SNE

- Uses a **symmetric cost function** where $\mathbf{p_{i|j} == p_{j|i}}$

- *We use the **Student t-distribution** to compute **similarity** in **low dimensional space***