# Data Mining Summer Term 2020

**Institute of Neural Information Processing**
PD Dr. F. Schwenker

Assignment 7 (Submission until June 23, 2020)

## Exercise 1 (10 points): *t*-Distributed Stochastic Neighbor Embedding [Pen and Paper]

Exploring a dataset is very important and usually one of the first steps when tackling a new data mining problem. A visual representation can help here and offers insights in the dataset. However, it is quite common to deal with higher-dimensional data and we do not have a direct way of visualizing data points in more than three dimensions. Hence, we need to fall back on alternatives like projection methods which try to find good representations of the higher-dimensional data points in the lower-dimensional space. In this exercise, we want to take a look at the popular *t*-Distributed Stochastic Neighbor Embedding (*t*-SNE) algorithm.

In its heart, *t*-SNE is a non-linear dimensionality reduction algorithm with the main goal of creating useful visualizations. It is worth noting that this was the design principle from the beginning unlike, for example, Principal Component Analysis (PCA) which has other uses as well.

Mathematically, we have $n$ data points $\boldsymbol{x}_i \in \mathbb{R}^d$ in a high-dimensional space (feature space) and we want to calculate $n$ projections $\boldsymbol{y}_i \in \mathbb{R}^r$ in a low-dimensional space (projection space). $r$ is usually 1, 2 or 3 allowing us to visualize the projections. *t*-SNE tries to perform this projection in a way so that the local structure of the high-dimensional data is also visible in the low-dimensional projections. This means, to some extent, that nearest-neighbour relationships are visible in the projections. However, neither distances nor density information is preserved by the projections and this was, again, a design choice. This is, for example, in contrast to the multidimensional scaling (MDS) algorithm which tries to preserve distances. For further details see e.g. [2].

Instead of comparing the $\boldsymbol{x}_i$ in the feature space directly with the points $\boldsymbol{y}_i$ in the projection space, *t*-SNE uses a probability distribution for each point (in both spaces) and compares those distributions instead. This is used to model the neighbourhood of each point with uncertainty allowing for smooth borders. This means that there is a chance, at least to some degree, that one point is the neighbour to another point. For the feature points $\boldsymbol{x}_i$, this probability is defined as

$$p_{j|i} = \frac{e^{-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\left\|x_i - x_k\right\|^2 / 2\sigma_i^2}} \quad \text{with} \quad p_{i|i} = 0. \tag{1}$$

This gives us a discrete probability distribution $P_i = (p_{1|i}, p_{2|i}, \ldots, p_{n|i})$ for each point $\boldsymbol{x}_i$. We can interpret $p_{j|i}$ as the probability that the point $\boldsymbol{x}_j$ is selected as a neighbour for the point $\boldsymbol{x}_i$ and that neighbours are selected proportionally to a probability density of a Gaussian distribution centred around the point $\boldsymbol{x}_i$. Note that each point $\boldsymbol{x}_i$ has its own probability distribution satisfying

$$\sum_{j=1}^{n} p_{j|i} = 1 \quad \text{and} \quad \forall j : p_{j|i} \geq 0. \tag{2}$$

In a similar way, we define a probability for each projection point $\boldsymbol{y}_i$

$$q_{ij} = \frac{1}{n} \frac{\left(1 + \left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \left\|\boldsymbol{y}_i - \boldsymbol{y}_k\right\|^2\right)^{-1}} \quad \text{with} \quad q_{ii} = 0. \tag{3}$$

Even though the underlying idea is the same as before, there are some differences compared to Equation 1. For once, we don't feed the distances to a Gaussian function but rather to a Student's $t$-distribution with one degree of freedom. This means that we can interpret $q_{ij}$ as the probability that the point $\boldsymbol{y}_j$ is selected as a neighbour for the point $\boldsymbol{y}_i$ and that neighbours are selected proportionally to a probability density of a Student's $t$-distribution with one degree of freedom centred around the point $\boldsymbol{y}_i$. Don't worry, we come back to this point later and clarify what this means and why we choose a different underlying distribution in the projection space[1]. Both underlying distributions are compared with each other in Figure 1.

The second difference is the prefactor $1/n$. This changes the meaning a bit since we do not have $n$ probability distributions like it was the case in Equation 1 but rather only one covering all point combinations. What is more, the distribution now satisfies

$$\sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} = 1 \quad \text{and} \quad \forall i, j \,:\, q_{ij} \geq 0. \tag{4}$$

Actually, this is already what we want and we are going to see that Equation 1 is only an intermediate step before we also arrive at a single distribution for the feature space. You can also think of $p_{j|i}$ and $q_{ij}$ as two $n \times n$ matrices $P$ and $Q$ with the row index $i$ and the column index $j$. In this way, Equation 2 means that each row of $P$ sums up to 1 whereby Equation 4 tells us that the whole matrix $Q$ sums up to 1.

Generally speaking, our goal is to find $n$ projection points $\boldsymbol{y}_i$ so that the corresponding probability matrix $Q$ is as similar as possible to the matrix $P$. We do so by optimizing a cost function, calculating the gradients and applying gradient descent. We are approaching this goal step by step and also covering some important details on the way. As an example, we use the data points shown in Figure 4 and our goal is to get a one-dimensional projection of these points.[2]

1. Let's start with the probabilities $p_{j|i}$ in the feature space. There are two things we need to take care of before we can use these probabilities. First, note that each data point $\boldsymbol{x}_i$ uses its own scaling parameter $\sigma_i$ in Equation 1. We need to find values for these parameters. Second, the corresponding matrix $P$, as noted before, does not sum up to 1 yet (only the rows). We need to fix this as well.

[1] Note that you already know by now where $t$-SNE has its name from: we work with probabilities (hence stochastic in the name) and we use the Student's $t$-distribution (hence the $t$ in the name).

[2] If you want additional information about $t$-SNE, there is, besides the original paper [3], also a video introduction worth pointing to.

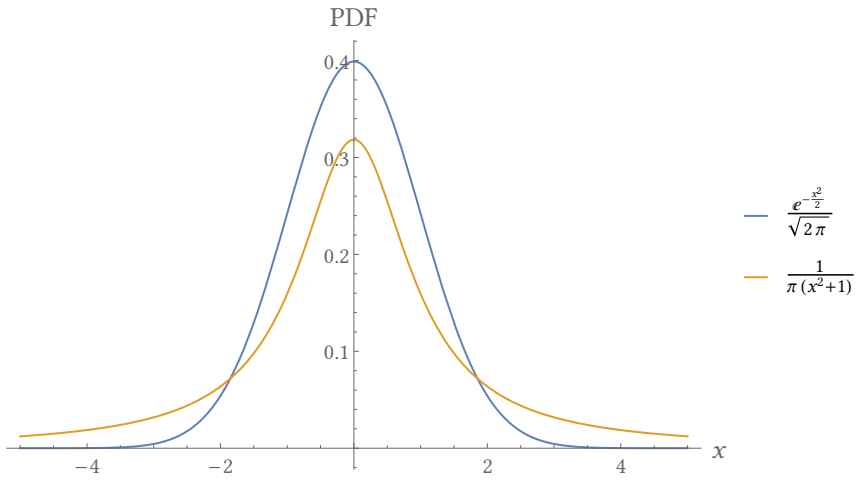



**Figure 1**: Comparison between the standard normal distribution (blue) and the Student's $t$-distribution (orange) with one degree of freedom. Note that the constant factors $1/\sqrt{2\pi}$ and $1/\pi$ are not shown in Equation 1 respectively Equation 3 since they are not relevant here.

a) The parameter $\sigma_i$ defines a region around each data point $\boldsymbol{x}_i$ where other data points $\boldsymbol{x}_j$ have a chance to be selected as neighbours. You can think of this as a geometric way to control how many neighbours you want to consider in the corresponding probability distribution $P_i$. We define this parameter per data point and not globally because an appropriate value of $\sigma_i$ depends on the density of the region around $\boldsymbol{x}_i$. Smaller values of $\sigma_i$ are usually more appropriate in dense regions and larger values of $\sigma_i$ are better suited when the data points $\boldsymbol{x}_i$ are located further away from each other.

   i. Show that Equation 2 is true.

   ii. Take a look at this animation[3] to get a feeling of how $\sigma_i$ influences the probability distribution $P_i$. It uses the (arbitrarily chosen) data point $\boldsymbol{x}_9$ and shows the distribution $P_9$ for varying $\sigma_9$. Explain why the distribution $P_9$ has a high peak for the data point $\boldsymbol{x}_{13}$ for the very small value $\sigma_9 = 0.1$ even though this data point is not included in the "main Gaussian area". Is it a coincidence that the high peak occurs for the data point $\boldsymbol{x}_{13}$?

   iii. Calculate the probability distribution for the extreme case $\lim_{\sigma_9 \to \infty} P_9$ (the result should be a vector with $n = 13$ elements).

   iv. To be a little bit more precise, we have to name $p_{j|i}$ a conditional probability given $\boldsymbol{x}_i$ and retrieving the probability for the possible neighbour $\boldsymbol{x}_j$. This is because Equation 1 is not symmetric in general, i.e. $p_{j|i} \neq p_{i|j}$ is possible. Explain

[3] https://milania.de/showcase/t-Distributed_Stochastic_Neighbor_Embedding#fig:tSNE_AnimationGaussian

why this is the case and name which part of Equation 1 is responsible for this behaviour.

b) We now know what the parameter $\sigma_i$ does and why we need it. The remaining question is how we select it. *t*-SNE uses an indirect approach here. Each $\sigma_i$ value leads to a different probability distribution $P_i$ for the point $\boldsymbol{x}_i$. For each distribution $P_i$ we can calculate the corresponding entropy

$$H(P_i) = -\sum_{j=1}^{n} p_{j|i} \cdot \log_2(p_{j|i})$$

and for the entropy, we can calculate again something which is called the perplexity

$$\text{Perp}(P_i) = 2^{H(P_i)}.$$

The approach of *t*-SNE is to let the user define a global value for the perplexity so that the algorithm can derive the corresponding $\sigma_i$ values for each data point $\boldsymbol{x}_i$. That is, the $\sigma_i$ values are adjusted in a way so that the new entropy $H(P_i)$ fits to the given perplexity value (which is the same for all data points). A binary search is used to find the $\sigma_i$ values in the implementation[4] but we are only interested in how we can interpret the relationship between $\text{Perp}(P_i)$ and $\sigma_i$ here.

The authors of *t*-SNE describe this parameter as: "The perplexity can be interpreted as a smooth measure of the effective number of neighbors" [3, p. 2582]. This is because $P_i$ has the information about the number of neighbours with influence and so we can interpret $H(P_i)$ as the number of bits we need to encode these neighbours. For example, with $H(P_i) = 3$ we have three bits to encode the neighbour number and since we can encode $2^3 = 8$ possible neighbours in this configuration, the perplexity $\text{Perp}(P_i)$ effectively measures the number of considered neighbours for each data point $\boldsymbol{x}_i$.

i. Figure 2 shows the relationship between the scaling factor and the perplexity for the two data points $\boldsymbol{x}_4$ and $\boldsymbol{x}_{13}$ and Figure 3 the corresponding probability distributions $P_i$ for the $\sigma_i$ values which correspond to a given perplexity of (roughly) Perp = 2. Use this information to fill out the missing values in Table 1.

ii. Explain why there is a plateau in both curves of Figure 2 at around $\sigma_i \in [0.5; 1.5]$ and why the blue line reaches this plateau before the orange line.

iii. Sketch how the two curves of Figure 2 proceed further, i.e. how they look like for higher $\sigma_i$ values. Hint: use the interpretation of the perplexity.

[4] An alternative approach would be to test a bunch of values in a limited range and select the one which comes closest to the target perplexity.

c) The next part is to adjust the matrix $P$. For this, we define new probabilities

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

This formula changes two aspects. For one thing, the whole matrix $P$ now sums up to 1. For another thing, we symmetrize the conditional probabilities by averaging in both directions. The second adjustment makes the probabilities easier to handle and also results in simpler gradients which are faster to compute [3, p. 2584]. Show that Equation 4 does now also hold for the matrix $P$, i.e.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} = 1 \quad \text{and} \quad \forall i, j \,:\, p_{ij} \geq 0.$$

2. Since we now have valid probability distributions $P$ and $Q$ in both spaces, we can enter the optimization step. We need to find projection points $\boldsymbol{y}_i$ where, ideally, we have equal probabilities $p_{ij} = q_{ij}$. Usually, this is something we cannot achieve in practice for all point combinations but the goal is to find a solution which reaches this state as close as possible. For this, we need a cost function which compares $P$ and $Q$ and tells us how good our current solution is. We use the Kullback–Leibler divergence

$$D_Q(P) = \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} \cdot \log_2\left(\frac{p_{ij}}{q_{ij}}\right) \tag{5}$$

for this job. This is a measure designed to compare probability distributions and has its roots in the field of information theory. Note that $p_{ij}$ is fixed since our probabilities from the feature space don't change anymore. However, $q_{ij}$ depends on the current choice of our projection points $\boldsymbol{y}_i$ and this allows us to minimize Equation 5. The gradient based on this cost function is given as [3, p. 2586]

$$\frac{\partial D_Q(P)}{\partial \boldsymbol{y}_i} = 4\sum_{j=1}^{n}(p_{ij} - q_{ij})(\boldsymbol{y}_i - \boldsymbol{y}_j)\left(1 + \left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|^2\right)^{-1}. \tag{6}$$

We can iteratively approach to a local minimum with gradient descent and randomly initialized starting positions for the projections points $\boldsymbol{y}_i$. Based on a projection point $\boldsymbol{y}_i$, we can interpret the gradient as a sum of attraction and repelling forces from all other projection points $\boldsymbol{y}_{j\neq i}$ defined by $p_{ij} - q_{ij}$ in the scaled direction of $\boldsymbol{y}_i - \boldsymbol{y}_j$.

a) To see how these forces interact and how Equation 6 behaves, we consider a new toy example dataset shown in Table 2. We are interested in the movement of the projection point $y_1$ and your task is to calculate the missing values of Table 2.

b) Draw the projection points $y_i$ in a 1D-plot and mark for each projection point $y_{j\neq 1}$ whether it exerts an attraction or repelling force on $y_1$.

c) Calculate the negative gradient $-\partial D_Q(P)/\partial y_1$ (negative because we want to a minimum and not a maximum) to see in which direction the projection point $y_1$ moves.

d) In another animation[5], you can see how $t$-SNE evolves for the dataset shown in Figure 4. Note how the matrix $Q$ becomes more and more similar to the matrix $P$ with increasing iterations. Explain how you can tell from the matrix plot ($Q$ or $P$) how many natural clusters there are in the dataset and how many data points each cluster contains.

3. We covered the complete $t$-SNE algorithm by now. One remaining aspect which needs clarification is why we use a Student's $t$-distribution instead of a Gaussian for the probabilities $q_{ij}$ (Equation 3). The underlying reason for this choice is that it tries to solve the so-called crowding problem which can occur when mapping points from a high-dimensional to a low-dimensional space.

   a) Let's look at the problem first. The thing is that higher dimensions offer just much more room for our data points than lower-dimensions. This has a critical impact on the distances between the points. For example, the number of choices to place points with similar distance to each other grows with the number of dimensions and there is no way to model this behaviour exactly in lower dimensions. This leads to imbalanced distance distributions.

      To make this more concrete, we want to analyse the volume of a unit sphere (which serves as a measure for the number of data points) in different dimensions and measure the ratio between distances $< 0.5$ and distances $> 0.5$. Fill out Table 3 and explain how the distribution of distances changes with increasing dimensions.

   b) By comparing the two distributions in Figure 1, we see that distances in the feature space (modelled by Gaussians) effectively map to different distances in the projection space (modelled by the Student's $t$-distribution). Based on this observation, we can define two regions:

      I. The region $R_1$ where distances in the feature space map to smaller distances in the projection space and

      II. the region $R_2$ where they map to larger distances.

      Mark those two regions in Figure 1.

   c) Consider distances from the second region $R_2$ and explain why this leads to less crowded projection points.

   d) Figure 5 (b) shows two projections for the feature points shown in Figure 5 (a). One projection corresponds to the $t$-SNE algorithm with the use of a Student's $t$-distribution for $q_{ij}$ and the other projection corresponds to the SNE algorithm [1]

[5] https://milania.de/showcase/t-Distributed_Stochastic_Neighbor_Embedding#fig:tSNE_Animation

where also a Gaussian distribution is used for $q_{ij}$. Explain which projection (A or B) maps to which algorithm ($t$-SNE or SNE).

# References


[1] Geoffrey E Hinton and Sam T. Roweis. "Stochastic Neighbor Embedding". In: *Advances in Neural Information Processing Systems 15*. Ed. by S. Becker, S. Thrun, and K. Obermayer. MIT Press, 2003, pp. 857–864. URL: http://papers.nips.cc/paper/2276-stochastic-neighbor-embedding.pdf.

[2] *interpretation - K-means clustering on the output of t-SNE - Cross Validated*. Feb. 23, 2018. URL: https://stats.stackexchange.com/questions/263539/k-means-clustering-on-the-output-of-t-sne (visited on 06/07/2018).

[3] Laurens van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE". In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.


**Table 1**: Relationship between the scaling $\sigma_i$ and the perplexity Perp($P_i$) for two data points. Decide about the scale factor $\sigma_i$, the probability distribution $P_i$ of Figure 3 (C or D) and select which of the two curves in Figure 2 (A or B) fits to each point.

| Data point $\boldsymbol{x}_i$ | Scaling $\sigma_i$ | Distribution $P_i$ | Entropy $H(P_i)$ | Perplexity Perp($P_i$) | Curve |
|---|---|---|---|---|---|
| $\boldsymbol{x}_4$ | | | 0.993761 | 1.99137 | |
| $\boldsymbol{x}_{13}$ | | | 0.988433 | 1.98403 | |

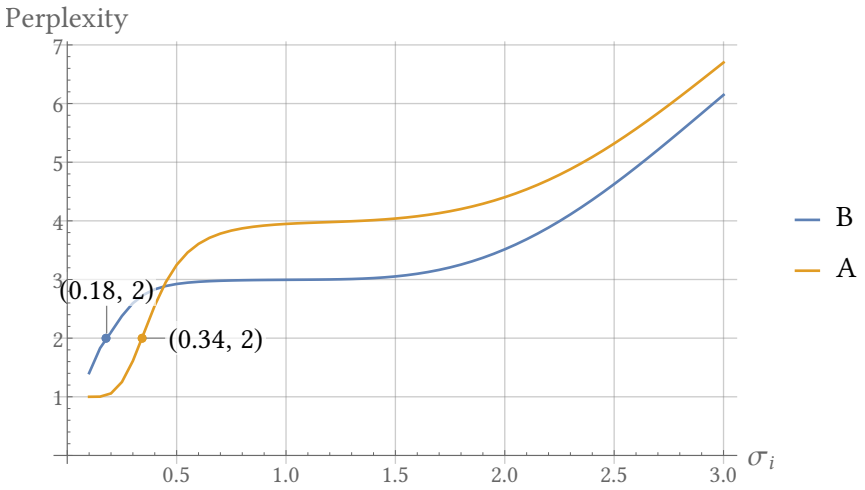



**Figure 2**: Relationship between the scaling factor $\sigma_i$ and the perplexity Perp($P_i$). One curve belongs to the point $\boldsymbol{x}_4$ and one to $\boldsymbol{x}_{13}$.

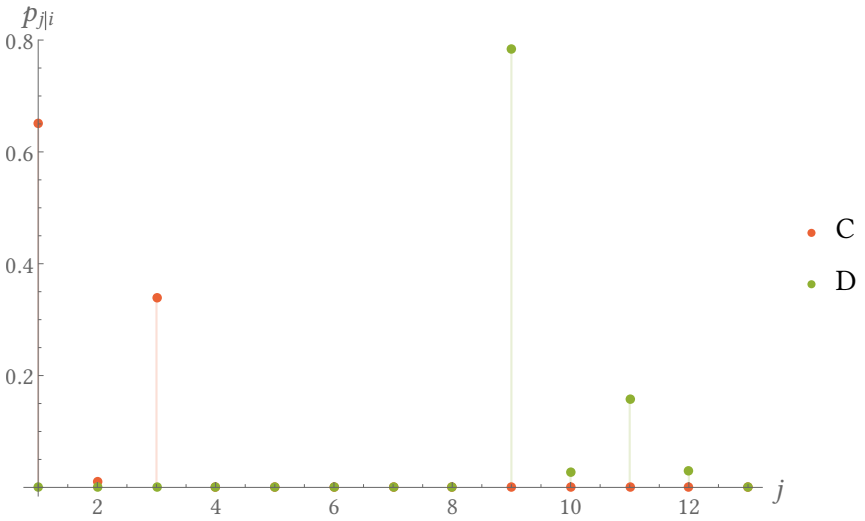



**Figure 3**: Probability distributions for the two points $\boldsymbol{x}_4$ and $\boldsymbol{x}_{13}$ with the corresponding $\sigma_i$ value of Figure 2 which corresponds to a perplexity of (roughly) Perp = 2.

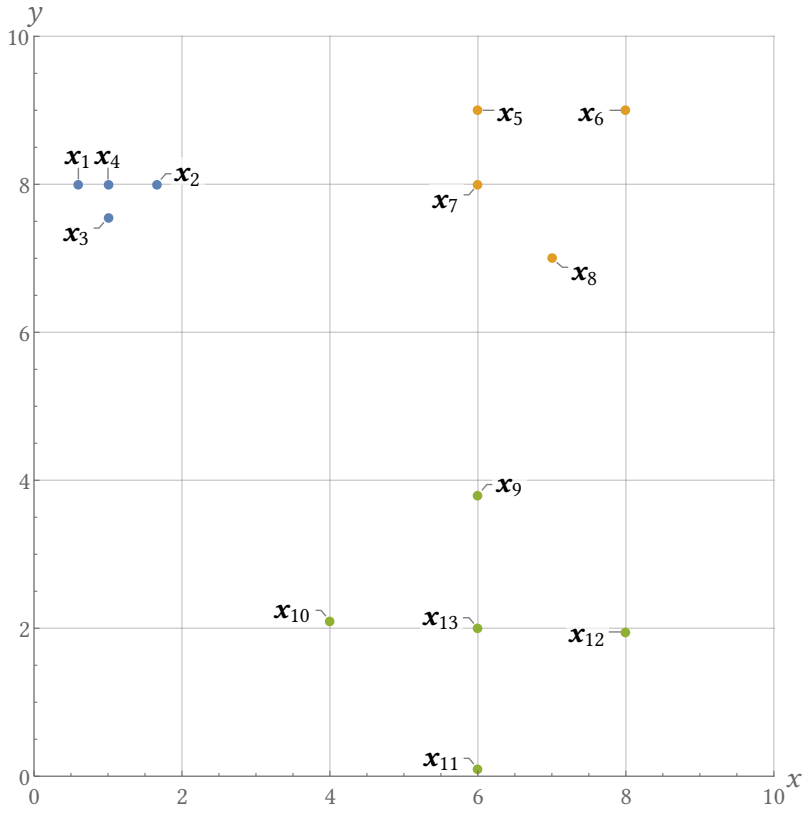


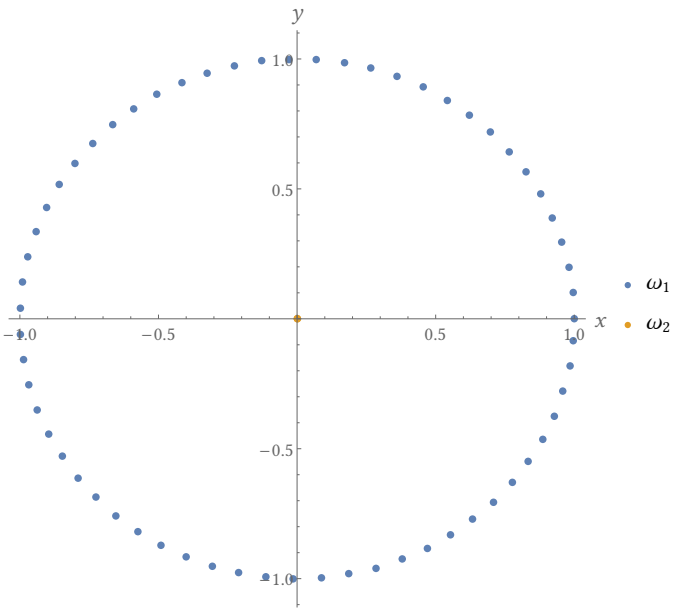

**Figure 4**: Example dataset used to explore the *t*-SNE algorithm. The colours correspond to clusters obtained by the *k*-means algorithm with $k = 3$.

**Table 2**: Toy example dataset consisting of four one-dimensional projection points $y_i$. The last column is intended for one summand, i.e. the product of the three factors in Equation 6. You can think of the corresponding data points (which are not shown) as a two-cluster set with the point $\boldsymbol{x}_2$ in one and the points $\boldsymbol{x}_1, \boldsymbol{x}_3$ and $\boldsymbol{x}_4$ in a second cluster. Note that this is also reflected by the probabilities $p_{1j}$.

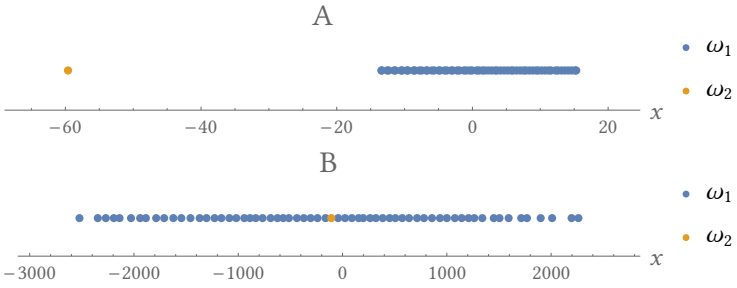
| $y_j$ | $p_{1j}$ | $q_{1j}$ | $p_{1j} - q_{1j}$ | $y_1 - y_j$ | $\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}$ | Summand |
|---|---|---|---|---|---|---|
| $y_1 = 0$ | 0.00 | | | | | |
| $y_2 = 1$ | 0.04 | | | | | |
| $y_3 = 2$ | 0.15 | | | | | |
| $y_4 = 3$ | 0.20 | | | | | |

**Table 3**: The effect of the crowding problem analysed on a hyper-sphere. Based on the volume in different dimension, calculate the ratio of distances < 0.5 and distances > 0.5 for a hyper-sphere with unit length ($r = 1$).

| Dimension | Volume | Distances < 0.5 (ratio) | Distances > 0.5 (ratio) |
|---|---|---|---|
| $\mathbb{R}^1$ | $2r$ | | |
| $\mathbb{R}^2$ | $\pi r^2$ | | |
| $\mathbb{R}^3$ | $\frac{4}{3}\pi r^3$ | | |




(a) Data points




(b) Projections

**Figure 5**: Dataset with corresponding projections obtained by using two different algorithms. The data points are arranged in a circle with one point in the centre (visualized as different classes). In both cases, a perplexity of Perp = 3 was specified and the algorithm was repeated multiple times with different initial starting points. Shown are the results with the lowest obtained loss.