Uniform Manifold Approximation and Projection (UMAP)

Statistical Machine Learning (ENGG*6600*02)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Summer 2023

Data Graph in the Input Space

Data Graph in the Input Space

- **Uniform Manifold Approximation and Projection (UMAP)** was proposed in 2018 [1].
- Consider a training dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ where n is the sample size and d is the dimensionality.
- We construct a k-**Nearest Neighbors (kNN) graph** for this dataset. It has been empirically observed that UMAP requires **fewer number of neighbors** than t-SNE [2]. Its default value is $k = 15$. We denote the j-th neighbor of $\boldsymbol{x}_i$ by $\boldsymbol{x}_{i,j}$. Let $\mathcal{N}_i$ denote the set of neighbor points for the point $\boldsymbol{x}_i$, i.e., $\mathcal{N}_i := \{\boldsymbol{x}_{i,1}, \ldots, \boldsymbol{x}_{i,k}\}$.
- We treat neighborhood relationship between points stochastically. Inspired by SNE [3] and t-SNE [4, 5], we use the **Gaussian or Radial Basis Function (RBF)** kernel for the measure of similarity between points in the **input space**. The probability that a point $\boldsymbol{x}_i$ has the point $\boldsymbol{x}_j$ as its neighbor can be computed by the similarity of these points:

$$p_{j|i} := \begin{cases} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 - \rho_i}{\sigma_i}\right) & \text{if } \boldsymbol{x}_j \in \mathcal{N}_i \\ 0 & \text{Otherwise,} \end{cases} \tag{1}$$

  where $\|.\|_2$ denotes the ℓ_2 norm.
- The ρ_i is the distance from $\boldsymbol{x}_i$ to its nearest neighbor:

$$\rho_i := \min\{\|\boldsymbol{x}_i - \boldsymbol{x}_{i,j}\|_2 \,|\, 1 \leq j \leq k\}. \tag{2}$$

Data Graph in the Input Space

- The σ_i is the scale parameter which is calculated such that the total similarity of point $\boldsymbol{x}_i$ to its k nearest neighbors is normalized. By binary search, we find σ_i to satisfy:

$$\sum_{j=1}^{k} \exp\Big(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{i,j}\|_2 - \rho_i}{\sigma_i}\Big) = \log_2(k). \tag{3}$$

- Note that t-SNE [4] has a similar search for its scale using entropy as perplexity. These searches **make the neighborhoods of various points behave similarly** because the **scale for a point in a dense region of dataset becomes small** while the **scale of a point in a sparse region of data becomes large**.
- In other words, UMAP and t-SNE both assume (or approximate) that **points are uniformly distributed on an underlying low-dimensional manifold**. This approximation is also included in the name of UMAP.
- Eq. (1):

$$p_{j|i} := \begin{cases} \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 - \rho_i}{\sigma_i}\right) & \text{if } \boldsymbol{x}_j \in \mathcal{N}_i \\ 0 & \text{Otherwise,} \end{cases}$$

is a **directional similarity** measure. To have a **symmetric** measure with respect to i and j, we symmetrize it as:

$$\mathbb{R} \ni p_{ij} := p_{j|i} + p_{i|j} - p_{j|i}\, p_{i|j}. \tag{4}$$

This is a symmetric measure of similarity between points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the input space.

Data Graph in the Embedding Space

Data Graph in the Embedding Space

- Let the **embeddings** of points be $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n] \in \mathbb{R}^{p \times n}$ where p is the dimensionality of embedding space and is smaller than input dimensionality, i.e., $p \ll d$. Note that $\boldsymbol{y}_i$ is the embedding corresponding to $\boldsymbol{x}_i$.
- In the **embedding space**, the probability that a point $\boldsymbol{y}_i$ has the point $\boldsymbol{y}_j$ as its neighbor can be computed by the similarity of these points:

$$\mathbb{R} \ni q_{ij} := (1 + a \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})^{-1}, \tag{5}$$

  which is **symmetric** with respect to i and j.
- The variables $a > 0$ and $b > 0$ are hyperparameters determined by the user. By default, we have $a \approx 1.929$ and $b \approx 0.7915$ [1], although it has been empirically seen that setting $a = b = 1$ does not qualitatively impact the results [6].

Optimization Cost Function

Optimization Cost Function

- UMAP aims to **make the data graph in the low-dimensional embedding space similar to the data graph in the high-dimensional embedding space**. In other words, we treat Eqs. (4) and (5) as probability distributions and minimize the difference of these distributions to make similarities of points in the embedding space as the similarities of points in the input space. A measure for the difference of these similarities of graphs is the **fuzzy cross-entropy** defined as:

$$c_1 := \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \Big(p_{ij} \ln(\frac{p_{ij}}{q_{ij}}) + (1 - p_{ij}) \ln(\frac{1 - p_{ij}}{1 - q_{ij}}) \Big), \tag{6}$$

  where ln(.) is the natural logarithm. The definition of this cross-entropy is in the field of **fuzzy category theory** (see our tutorial paper [7] for more information).
- The **first term** in Eq. (6) is the **attractive force** which attracts the embeddings of neighbor points toward each other. This term should only appear when $p_{ij} \neq 0$ which means either $\boldsymbol{x}_j$ is a neighbor of $\boldsymbol{x}_i$, or $\boldsymbol{x}_i$ is a neighbor of $\boldsymbol{x}_j$, or both (see Eq. (4), $\mathbb{R} \ni p_{ij} := p_{j|i} + p_{i|j} - p_{j|i}\, p_{i|j}$).
- The **second term** in Eq. (6) is the **repulsive force** which repulses the embeddings of non-neighbor points away from each other.
- As the number of all permutations of non-neighbor points is very large, computation of the second term is non-tractable in big data. Inspired by Word2Vec [8] and LargeVis [9], UMAP uses **negative sampling** where, for every point $\boldsymbol{x}_i$, m points are sampled randomly from the training dataset and treat them as non-negative (negative) points for $\boldsymbol{x}_i$.
- As the dataset is usually large, i.e. $m \ll n$, the sampled points **will be actual negative points with high probability**.

Optimization Cost Function

- We had:

$$c_1 := \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \Big(p_{ij} \ln(\frac{p_{ij}}{q_{ij}}) + (1 - p_{ij}) \ln(\frac{1 - p_{ij}}{1 - q_{ij}}) \Big).$$

- The summation over the second term in Eq. (6) is computed only over these negative samples rather than **all** negative points.
- UMAP changes the data graph in the embedding space to make it similar to the data graph in the input space.
- Eq. (6) is the cost function which is minimized in UMAP where the optimization variables are $\{y_i\}_{i=1}^{n}$:

$$\min_{\{\boldsymbol{y}_i\}_{i=1}^{n}} c_1 := \min_{\{\boldsymbol{y}_i\}_{i=1}^{n}} \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \Big(p_{ij} \ln(p_{ij}) - p_{ij} \ln(q_{ij}) + (1 - p_{ij}) \ln(1 - p_{ij}) - (1 - p_{ij}) \ln(1 - q_{ij}) \Big).$$

- According to Eqs. (1), (4), and (5), in contrast to q_{ij}, the p_{ij} is independent of the optimization variables $\{y_i\}_{i=1}^{n}$. Hence, we can drop the constant terms to revise the cost function:

$$c_2 := -\sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \Big(p_{ij} \ln(q_{ij}) + (1 - p_{ij}) \ln(1 - q_{ij}) \Big), \tag{7}$$

which should be minimized.

Optimization Cost Function

- We had:

$$c_2 := -\sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\Big(p_{ij}\ln(q_{ij}) + (1-p_{ij})\ln(1-q_{ij})\Big).$$

- Two important terms in this cost function are:

$$c^a_{i,j} := -\ln(q_{ij}), \tag{8}$$

$$c^r_{i,j} := -\ln(1-q_{ij}), \tag{9}$$

and we can write:

$$c_2 := \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\Big(p_{ij}\,c^a_{i,j} + (1-p_{ij})\,c^r_{i,j}\Big) \tag{10}$$

$$\overset{(a)}{=} 2\sum_{i=1}^{n}\sum_{j=i+1}^{n}\Big(p_{ij}\,c^a_{i,j} + (1-p_{ij})\,c^r_{i,j}\Big), \tag{11}$$

where (a) is because $p_{ij} = p_{ji}$, $c^a_{i,j} = c^a_{j,i}$, and $c^r_{i,j} = c^r_{j,i}$ are symmetric.

- The Eqs. (8) and (9) are the **attractive and repulsive forces** in Eq. (7), respectively. The **attractive force** attracts the neighbor points toward each other in the embedding space while the **repulsive force** pushes the non-neighbor points (i.e., points with low probability of being neighbors) away from each other in the embedding space.
- According to Eq. (10), $c^a_{i,j}$ and $c^r_{i,j}$ occur with probability p_{ij} and $(1-p_{ij})$, respectively. For every point, we call it the **anchor** point and we call its neighbor and non-neighbor points, with large and small p_{ij}, as the **positive** and **negative** points, respectively.

The Training Algorithm of UMAP

The Training Algorithm of UMAP

Input: Training data $\{\boldsymbol{x}_i\}_{i=1}^n$
Construct kNN graph
Initialize $\{\boldsymbol{y}_i\}_{i=1}^n$ by Laplacian eigenmap
Calculate p_{ij} and q_{ij} for $\forall i, j \in \{1, \dots, n\}$ by Eqs. (4) and (5)
$\eta \leftarrow 1, \nu \leftarrow 0$
while *not converged* **do**
  $\nu \leftarrow \nu + 1$ // epoch index
  for i *from* 1 *to* n **do**
    for j *from* 1 *to* n **do**
      $u \sim U(0, 1)$
      if $u \leq p_{ij}$ **then**
        $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta \frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_i}$
        $\boldsymbol{y}_j \leftarrow \boldsymbol{y}_j - \eta \frac{\partial c_{i,j}^a}{\partial \boldsymbol{y}_j}$
        for m *iterations* **do**
          $l \sim U\{1, \dots, n\}$
          $\boldsymbol{y}_i \leftarrow \boldsymbol{y}_i - \eta \frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_i}$
          // The next line does not exist in original UMAP:
          $\boldsymbol{y}_l \leftarrow \boldsymbol{y}_l - \eta \frac{\partial c_{i,l}^r}{\partial \boldsymbol{y}_l}$
  $\eta \leftarrow 1 - \frac{\nu}{\nu_{\max}}$
Return $\{\boldsymbol{y}_i\}_{i=1}^n$

Algorithm 1: UMAP algorithm

The Training Algorithm of UMAP

- As this algorithm shows, a k**NN graph** is constructed from the training data $\{\boldsymbol{x}_i\}_{i=1}^n$.
- UMAP uses **Laplacian eigenmap** [10, 11], also called **spectral embedding**, for initializing the embeddings of points denoted by $\{\boldsymbol{y}_i\}_{i=1}^n$.
- Using Eqs. (4) and (5), p_{ij} and q_{ij} are calculated for all points.
- **Stochastic Gradient Descent (SGD)** is used for optimization where optimization is performed iteratively.
- In every iteration (epoch), we iterate over points twice with indices i and j where the i-th point is called the **anchor**. For every pair of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we update their embeddings $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with probability p_{ij} (recall Eq. (7)).
- If p_{ij} is large, it means that the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are probably neighbors (in this case, the j-th point is called the **positive** point) and their embeddings are highly likely to be updated to become close in the embedding space based on the attractive force. For implementing it, we can **sample a uniform** value from the continuous uniform distribution $U(0, 1)$ and if that is less than p_{ij}, we update the embeddings.
- We update the embeddings $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ by gradients $\partial c_{i,j}^a / \partial \boldsymbol{y}_i$ and $\partial c_{i,j}^a / \partial \boldsymbol{y}_j$, respectively, where η is the learning rate.
- For **repulsive forces**, we use **negative sampling** as was explained before. If m denotes the size of negative sample, we sample m indices from the discrete uniform distribution $U\{1, \ldots, n\}$. These are the indices of points which are considered as **negative** samples $\{\boldsymbol{y}_l\}$ where $|\{\boldsymbol{y}_l\}| = m$. As the size of dataset is usually large enough to satisfy $n \gg m$, these negative points are probably valid because many of the points are non-neighbors of the considered anchor.
- In negative sampling, the **original UMAP** [1] updates only the embedding of anchor $\boldsymbol{y}_i$ by gradient of the repulsive force $\partial c_{i,j}^a / \partial \boldsymbol{y}_i$. One can **additionally update the embedding of negative point** $\boldsymbol{y}_l$ by gradient of the repulsive force $\partial c_{i,j}^a / \partial \boldsymbol{y}_l$ [12].

The Training Algorithm of UMAP

- We had:

$$c^a_{i,j} := -\ln(q_{ij}),$$
$$c^r_{i,j} := -\ln(1 - q_{ij}).$$

- The gradients of attractive and repulsive cost functions in UMAP are:

$$\frac{\partial c^a_{i,j}}{\partial \boldsymbol{y}_i} = \frac{2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)}}{(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_i - \boldsymbol{y}_j), \tag{12}$$
$$\frac{\partial c^r_{i,j}}{\partial \boldsymbol{y}_i} = \frac{-2b}{(\varepsilon + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2)(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_i - \boldsymbol{y}_j), \tag{13}$$

  where ε is a small positive number, e.g. $\varepsilon = 0.001$, for stability to prevent division by zero when $\boldsymbol{y}_i \approx \boldsymbol{y}_j$.

- Likewise, we have:

$$\frac{\partial c^a_{i,j}}{\partial \boldsymbol{y}_j} = \frac{2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)}}{(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_j - \boldsymbol{y}_i),$$
$$\frac{\partial c^r_{i,j}}{\partial \boldsymbol{y}_j} = \frac{-2b}{(\varepsilon + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2)(1 + a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_j - \boldsymbol{y}_i).$$

The Training Algorithm of UMAP

- Proof: We had:

$$
\begin{aligned}
c^a_{i,j} &:= -\ln(q_{ij}),\\
c^r_{i,j} &:= -\ln(1-q_{ij}),\\
\mathbb{R} \ni q_{ij} &:= (1+a\,\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2b})^{-1}.
\end{aligned}
$$

- For the first equation, we have:

$$
\begin{aligned}
\frac{\partial c^a_{i,j}}{\partial \boldsymbol{y}_i} &= \frac{\partial c^a_{i,j}}{\partial q_{ij}} \times \frac{\partial q_{ij}}{\partial \boldsymbol{y}_i} = \frac{-1}{q_{ij}} \times \left(\frac{-1}{(1+a\,\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2b})^2} \times 2ab(\boldsymbol{y}_i-\boldsymbol{y}_j) \times \|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2(b-1)} \right)\\
&\overset{(5)}{=} \frac{2ab\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2(b-1)}}{(1+a\,\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2b})}(\boldsymbol{y}_i-\boldsymbol{y}_j). \qquad (14)
\end{aligned}
$$

- For the second equation, we have:

$$
\begin{aligned}
\frac{\partial c^r_{i,j}}{\partial \boldsymbol{y}_i} &= \frac{\partial c^r_{i,j}}{\partial q_{ij}} \times \frac{\partial q_{ij}}{\partial \boldsymbol{y}_i}\\
&= \frac{1}{1-q_{ij}} \times \left(\frac{-1}{(1+a\,\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2b})^2} \times 2ab(\boldsymbol{y}_i-\boldsymbol{y}_j) \times \|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2(b-1)} \right)\\
&= \frac{-2ab\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2(b-1)}}{(1-q_{ij})(1+a\,\|\boldsymbol{y}_i-\boldsymbol{y}_j\|_2^{2b})^2}(\boldsymbol{y}_i-\boldsymbol{y}_j). \qquad (15)
\end{aligned}
$$

The Training Algorithm of UMAP

- Eq. (5) was: $\mathbb{R} \ni q_{ij} := (1 + a \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_2^{2b})^{-1}$.
- The term in the numerator can be simplified as:

$$
\begin{aligned}
&-2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)} = -2b(a\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2b})\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{-2} \\
&\overset{(5)}{=} -2b\,(q_{ij}^{-1} - 1)\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{-2}.
\end{aligned}
$$

- The term in the denominator can be simplified as:

$$
(1 - q_{ij})(1 + a \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_2^{2b})^2 \overset{(5)}{=} (1 - q_{ij}) q_{ij}^{-2} = q_{ij}^{-2} - q_{ij}^{-1} = q_{ij}^{-1}(q_{ij}^{-1} - 1).
$$

- Hence, Eq. (15):

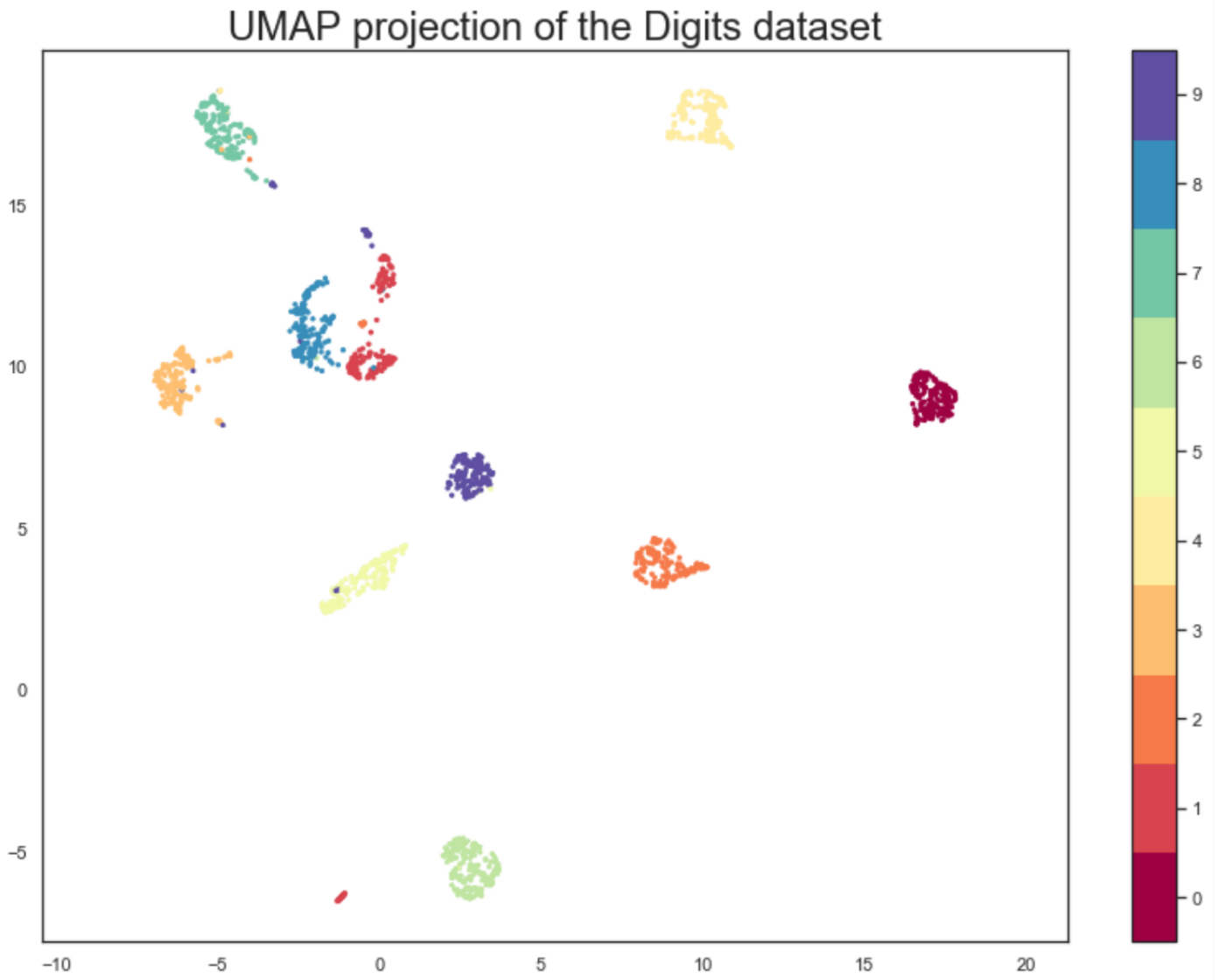
$$
\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i} = \frac{-2ab\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{2(b-1)}}{(1 - q_{ij})(1 + a \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_2^{2b})^2} (\boldsymbol{y}_i - \boldsymbol{y}_j),
$$

  can be simplified as:

$$
\begin{aligned}
\frac{\partial c_{i,j}^r}{\partial \boldsymbol{y}_i} &= \frac{-2b\,(q_{ij}^{-1} - 1)\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^{-2}}{q_{ij}^{-1}(q_{ij}^{-1} - 1)} (\boldsymbol{y}_i - \boldsymbol{y}_j) = \frac{-2b}{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2\, q_{ij}^{-1}} (\boldsymbol{y}_i - \boldsymbol{y}_j) \\
&\overset{(5)}{=} \frac{-2b}{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 \,(1 + a \left\| \boldsymbol{y}_i - \boldsymbol{y}_j \right\|_2^{2b})} (\boldsymbol{y}_i - \boldsymbol{y}_j).
\end{aligned}
$$

- If we add ε for stability to the squared distance in the denominator, the equation is obtained. Q.E.D.

Example of UMAP Embedding (Digit Dataset)



Credit of image: `https://umap-learn.readthedocs.io/en/latest/basic_usage.html`

Acknowledgment

- Some slides are based on our tutorial paper: "Uniform Manifold approximation and projection (UMAP) and its variants: tutorial and survey" [7]
- For more information on UMAP, refer to our tutorial paper [7].
- UMAP library: `https://umap-learn.readthedocs.io/en/latest/basic_usage.html`

References

[1] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[2] T. Sainburg, L. McInnes, and T. Q. Gentner, "Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning," *arXiv preprint arXiv:2009.12981*, 2020.

[3] G. E. Hinton and S. T. Roweis, "Stochastic neighbor embedding," in *Advances in neural information processing systems*, pp. 857–864, 2003.

[4] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[5] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Stochastic neighbor embedding with Gaussian and Student-t distributions: Tutorial and survey," *arXiv preprint arXiv:2009.10301*, 2020.

[6] J. N. Böhm, P. Berens, and D. Kobak, "A unifying perspective on neighbor embeddings along the attraction-repulsion spectrum," *arXiv preprint arXiv:2007.08902*, 2020.

[7] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Uniform manifold approximation and projection (umap) and its variants: tutorial and survey," *arXiv preprint arXiv:2109.02508*, 2021.

References (cont.)

[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[9] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proceedings of the 25th international conference on world wide web*, pp. 287–297, 2016.

[10] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, vol. 14, pp. 585–591, 2001.

[11] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Laplacian-based dimensionality reduction including spectral clustering, Laplacian eigenmap, locality preserving projection, graph embedding, and diffusion map: Tutorial and survey," *arXiv preprint arXiv:2106.02154*, 2021.

[12] S. Damrich and F. A. Hamprecht, "On UMAP's true loss function," *arXiv preprint arXiv:2103.14608*, 2021.