Advanced Data Analytics
Lecture week 2: UMAP

Ian T. Nabney

Overview

Uniform Manifold Approximation and Projection for dimensionality reduction

- Understand the fundamental nature of the UMAP algorithm
- Able to compare UMAP to t-SNE

Foundations of UMAP

- At a high level, UMAP uses local manifold approximations and patches together their local representations to construct a topological representation of the high dimensional data.
- Given some low-dimensional representation of the data, a similar process can be used to construct an equivalent topological representation.
- UMAP then optimizes the layout of the data representation in the low-dimensional space, to minimize the cross-entropy between the two topological representations.
- We will cut to the computational point of view of the algorithm.

Computational view of UMAP

- UMAP is in the class of k-neighbour based graph learning algorithms, such as Isomap (and t-SNE).
- In the first phase a particular weighted k-neighbour graph is constructed.
- In the second phase a low-dimensional layout of this graph is computed.

University of BRISTOL

Two key hyperparameters

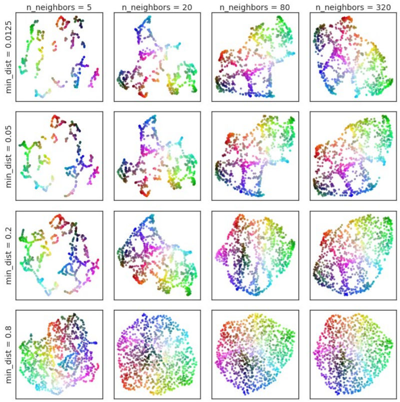
Number of neighbours

- The number of neighbours *n* the local scale at which to approximate the manifold as roughly flat, with the manifold estimation averaging over the *n* neighbours.
- Manifold features that occur at a smaller scale than within the *n* nearest-neighbours of points will be lost, while large scale manifold features that cannot be seen by patching together locally flat charts at the scale of *n* nearest-neighbours may not be well detected.

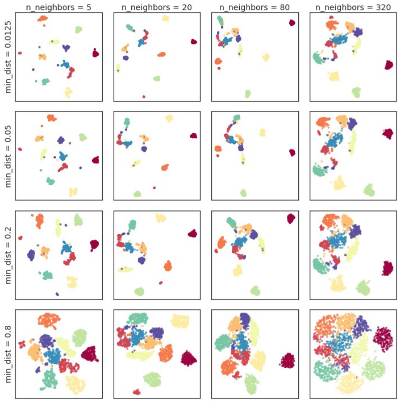
min-dist

- This parameter determines how closely points can be packed together in the low-dimensional representation.
- Low values of min-dist result in densely packed regions, but more faithfully represent the manifold structure.
- Increasing the value of min-dist forces the embedding to spread points out more, assisting visualization (and avoiding potential overplotting issues).

Examples of hyperparameter choice



Toy data from 3-dimensional colour cube.

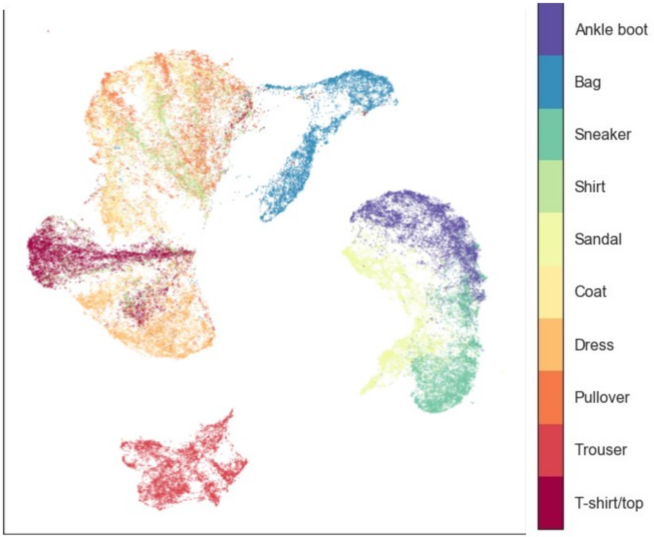


PenDigits dataset, where each point is an 8×8 grayscale image of a hand-written digit.

Qualitative comparison with other algorithms

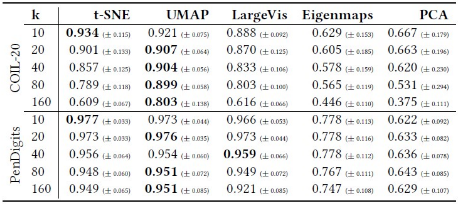
- As well as MNIST and COIL-20 also consider Fashion MNIST and Google News word vectors.
    - F-MNIST or Fashion MNIST is a dataset of 28×28 pixel grayscale images of fashion items (clothing, footwear and bags). There are 10 classes and 70000 total images. As with MNIST this is treated as 70000 different 784 dimensional vectors.
    - GoogleNewsword vectors is a dataset of 3 million words and phrases derived from a sample of Google News documents and embedded into a 300 dimensional space via word2vec.
- Compare against PCA, t-SNE, Laplacian Eigenmaps (graph layout algorithm used by UMAP for initialisation) and LargeVis (optimises a similar cost function to t-SNE but scales much better).

UMAP on F-MNIST



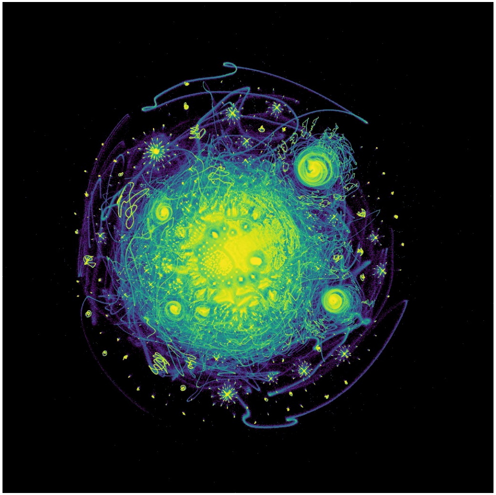
Quantitative comparison with other algorithms

- Compare algorithms by using *k*-nearest neighbour classifier performance in embedding space.
- By varying the hyper-parameter *k* we can also consider how structure preservation varies under transition from purely local to non-local, to more global structure.



	k	**t-SNE**	**UMAP**	**LargeVis**	**Eigenmaps**	**PCA**
COIL-20	10	**0.934** (± 0.115)	0.921 (± 0.075)	0.888 (± 0.092)	0.629 (± 0.153)	0.667 (± 0.179)
	20	0.901 (± 0.133)	**0.907** (± 0.064)	0.870 (± 0.125)	0.605 (± 0.185)	0.663 (± 0.196)
	40	0.857 (± 0.125)	**0.904** (± 0.056)	0.833 (± 0.106)	0.578 (± 0.159)	0.620 (± 0.230)
	80	0.789 (± 0.118)	**0.899** (± 0.058)	0.803 (± 0.100)	0.565 (± 0.119)	0.531 (± 0.294)
	160	0.609 (± 0.067)	**0.803** (± 0.138)	0.616 (± 0.066)	0.446 (± 0.110)	0.375 (± 0.111)
PenDigits	10	**0.977** (± 0.033)	0.973 (± 0.044)	0.966 (± 0.053)	0.778 (± 0.113)	0.622 (± 0.092)
	20	0.973 (± 0.033)	**0.976** (± 0.035)	0.973 (± 0.044)	0.778 (± 0.116)	0.633 (± 0.082)
	40	0.956 (± 0.064)	0.954 (± 0.060)	**0.959** (± 0.066)	0.778 (± 0.112)	0.636 (± 0.078)
	80	0.948 (± 0.060)	**0.951** (± 0.072)	0.949 (± 0.072)	0.767 (± 0.111)	0.643 (± 0.085)
	160	0.949 (± 0.065)	**0.951** (± 0.085)	0.921 (± 0.085)	0.747 (± 0.108)	0.629 (± 0.107)

Beauty of visualisation



Visualization of 30,000,000 integers as represented by binary vectors of prime divisibility, colored by density of points.

Conclusions

- Python software available from https://umap-learn.readthedocs.io/en/latest/
- McInnes, Leland, John Healy, and James Melville. 'Umap: Uniform manifold approximation and projection for dimension reduction.' arXiv preprint arXiv:1802.03426, 2018.
- Oskolkov, Nikolay, How Exactly UMAP Works, Towards Data Science blog

Constructing the weighted graph

- Given parameter k, for each x_i, compute the set $\{x_{i_1}, \ldots, x_{i_k}\}$ of the k nearest neighbours of x_i under a metric d.
- For each x_i define ρ_i and σ_i:

$$\rho_i = \min\{d(x_i, x_{i_j}) | 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\}$$

$$\log_2 k = \sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right)$$

- Define the weighted directed graph with vertices the data points $\mathbf{X}$, edges connecting each x_i with its k nearest neighbours, and the weight function

$$w(x_i, x_{i_j}) = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right)$$

- The choice of ρ_i (the distance to the nearest neighbour) ensures that x_i connects to at least one other data point with an edge of weight 1.

Graph layout

- UMAP uses a force directed graph layout algorithm in low-dimensional space with a set of attractive forces applied along edges and a set of repulsive forces applied among vertices.
- The algorithm proceeds by iteratively applying attractive and repulsive forces at each edge or vertex. This amounts to a non-convex optimization problem.
- Convergence to a local minimum is guaranteed by slowly decreasing the attractive and repulsive forces in a similar fashion to that used in simulated annealing.

$$\frac{-2ab\|\mathbf{y}_i - \mathbf{y}_j\|_2^{2(b-1)}}{1 + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2} w(x_i, x_j)(\mathbf{y}_i - \mathbf{y}_j)$$
$$\frac{2b}{(\epsilon + \|\mathbf{y}_i - \mathbf{y}_j\|_2^2)(1 + a\|\mathbf{y}_i - \mathbf{y}_j\|_2^{2b})}(1 - w(x_i, x_j))(\mathbf{y}_i - \mathbf{y}_j))$$

where a and b are hyper-parameters and ϵ is a small number to prevent division by zero.

Implementation

- Minimise cross-entropy between graphs (A, μ) and (A, ν)

$$-\sum_{a \in A} (\mu(a) \log(\nu(a)) + (1 - \mu(a)) \log(1 - \nu(a)) \tag{1}$$

  where the first term gets the clumps right and the second term gets the gaps right.

- Work with a smooth approximation to the membership function (so it can be optimised more easily)

$$\Psi(_{\mathrm{x}}, \mathbf{y}) = \begin{cases} 1 & \text{if } \|\mathbf{x} - \mathbf{y}\|_2 \leq \text{min-dist} \\ \exp(-(\|\mathbf{x} - \mathbf{y}\|_2 - \text{min-dist}) & \text{otherwise} \end{cases} \tag{2}$$

- The nearest-neighbour-descent algorithm has achieved $O(N^{1.14})$ empirically, a lot better than the naive $O(n^2)$.

Technical comparison with t-SNE

- Appendix C of paper
- UMAP does not apply normalization to either high- or low-dimensional probabilities, which dramatically reduces time of computing the high-dimensional graph since summation or integration is a computationally expensive procedure.
- UMAP uses the number of nearest neighbours instead of perplexity.
- UMAP uses a slightly different symmetrization of the high-dimensional probability
$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j}p_{j|i}$$
- UMAP uses the family of curves $q_{ij} = 1/(1 + ay^{2b})$ for modelling distance probabilities in low dimensions, not exactly Student t-distribution but with a similar shape.
- UMAP finds a and b from non-linear least-square fitting to the piecewise function with the min-dist hyperparameter.
- UMAP uses binary cross-entropy (CE) as a cost function instead of the KL-divergence like t-SNE.

$$CE(X,Y) = \sum_i \sum_j \left[p_{ij}(X) \log\left(\frac{p_{ij}(X)}{q_{ij}(Y)}\right) + (1 - p_{ij}(X)) \log\left(\frac{(1 - p_{ij}(X))}{(1 - q_{ij}(Y))}\right)\right]$$

University of BRISTOL