Non-linear Dimension Reduction Techniques

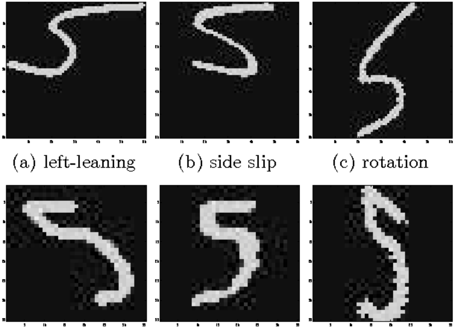
Emmanuelle Dankwa, Natalia Garcia Martin, Deborah Sulem

OxWaSP

October 18, 2019

Motivation for dimensionality reduction

- Unsupervised learning: insight on data and visualization
- High-dimensional data with **structure**
- Data lives on a low-dimensional manifold
- Example: hand-written digits under distorsions



(a) left-leaning (b) side slip (c) rotation

(d) right-leaning (e) side slip (f) rotation

Quick overview of methods

Many techniques to find representations in a latent space (embeddings):

- linear / non-linear methods
- probabilistic / deterministic model on the latent space
- mapping from a latent space to the data space or the reverse (proximity data methods)
- convex / non-convex objective function

Some challenges:

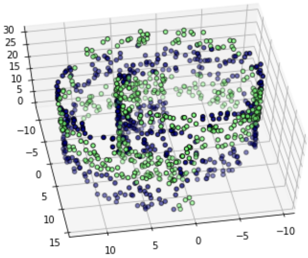
- preserving neighborhoods
- handling missing data
- projecting new data points in the latent space
- handling non-Gaussian noise models

Setting and datasets

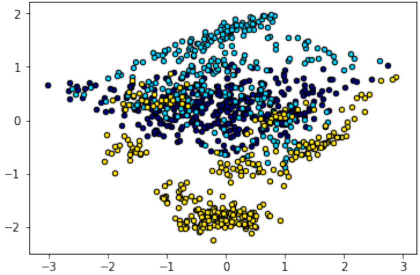
Some notations:

- N number of data points
- D dimension of the observation space
- Q dimension of the latent space
- $\boldsymbol{Y} \in \mathbf{R}^{N\times D}$ (centered) data matrix
- $\boldsymbol{X} \in \mathbf{R}^{N\times Q}$ latent variables matrix

Applications



(a) The Swiss roll



(b) Oil flow dataset

Metrics for dimension reduction

- With labelled data: **One-Nearest-Neighbour classification error**
- Without labels: k-neighborhood preservation
  - **Trustworthiness**:

$$T(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{n=1}^{N} \sum_{j \in U_n^{(k)}} r(n,j) - k$$

  - **Continuity**:

$$C(k) = 1 - \frac{2}{nk(2n-3k-1)} \sum_{n=1}^{N} \sum_{j \in V_n^{(k)}} \hat{r}(n,j) - k$$

Building block 1: Gaussian Processes

- Class of probabilistic models which specify distributions over function spaces
- Definition: collection of RV, any finite number of which have a joint Gaussian distribution

$$f(\boldsymbol{x}) \sim GP(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$$

- $m(\boldsymbol{x}) = E\left[f(\boldsymbol{x})\right]$
- $k(\boldsymbol{x}, \boldsymbol{x}') = E\left[\left(f(\boldsymbol{x}) - m(\boldsymbol{x})\right)\left(f(\boldsymbol{x}') - m(\boldsymbol{x}')\right)\right]$

Building block 2: Principal Component Analysis

- D-dimensional observed data $\boldsymbol{Y} \in \mathbb{R}^{N \times D}$
- Lower-dimensional representation $\boldsymbol{X} \in \mathbb{R}^{N \times Q}$

1. Standardise the data
2. Obtain the eigenvectors and eigenvalues from the covariance matrix
3. Sort eigenvalues in descending order and choose the Q eigenvectors that correspond to the Q largest eigenvalues
4. Construct the projection matrix $\boldsymbol{M}$ from the selected Q eigenvectors
5. Transform the original dataset $\boldsymbol{Y}$ via $\boldsymbol{M}$ to obtain a Q-dimensional feature subspace $\boldsymbol{X}$

Building block 2: Principal Component Analysis

- Latent-variable formulation: probabilistic PCA (Tipping and Bishop, 1999)
- Extension to non-linear mappings using Gaussian processes: GP-LVM (Lawrence, 2004)

Related method: Kernel PCA

- Extends conventional PCA to a high dimensional feature space using the kernel trick
- Linear kernel

$$k(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y}$$

- RBF kernel

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\gamma ||\boldsymbol{x} - \boldsymbol{y}||^2\right), \gamma > 0$$

- Sigmoid kernel

$$k(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\gamma \boldsymbol{x}^T \boldsymbol{y} + r)$$

- Polynomial kernel

$$k(\boldsymbol{x}, \boldsymbol{y}) = (\gamma \boldsymbol{x}^T \boldsymbol{y} + r)^d$$

KPCA

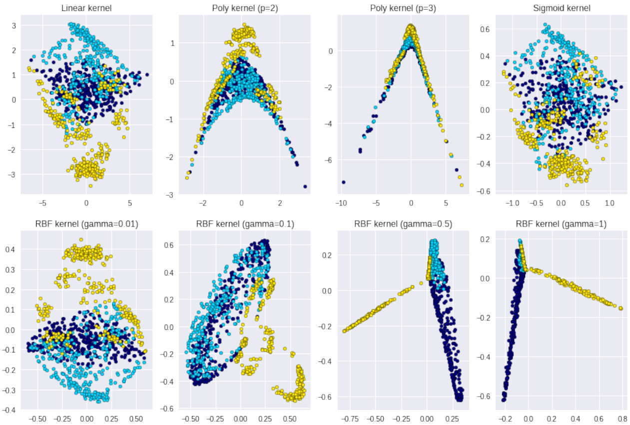


Figure: KPCA performance on oil flow dataset.

KPCA

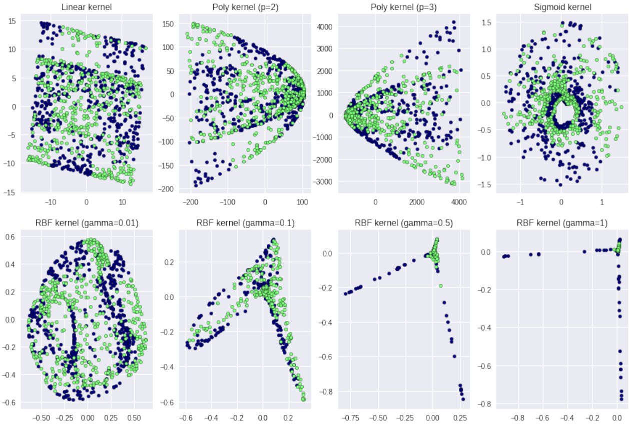


Figure: KPCA performance on swiss roll dataset.

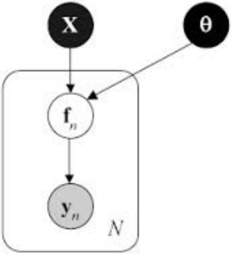
Gaussian Process Latent Variable Model [Lawrence, 2004]

Probabilistic model with noise ϵ:

$$\boldsymbol{Y} = f(\boldsymbol{X}) + \epsilon$$

Probabilistic PCA: linear mapping with parameters $\boldsymbol{W} \in \mathbf{R}^{D \times Q}$

$$y_n = \boldsymbol{W} x_n + \epsilon_n$$



- Assume spherical Gaussian noise $\epsilon_n \sim \mathcal{N}(0_D, \beta^{-1}\mathbf{I}_D)$
- Normal prior on the latent variables $x_n \sim \mathcal{N}(0_Q, \mathbf{I}_Q)$
- Marginalizing the likelihood over $\boldsymbol{X}$:

$$p(\boldsymbol{Y}|\boldsymbol{W}, \beta) = \Pi_{n=1}^{N} \mathcal{N}(y_n; 0_D, \boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} + \beta^{-1}\mathbf{I}_D)$$

- Maximize over the parameters $\boldsymbol{W}$

Gaussian Process Latent Variable Model

Dual Probabilistic PCA:

- Conjugate prior on the parameters $p(\boldsymbol{W}) = \Pi_{d=1}^{D} \mathcal{N}(w_d; 0_Q, \mathbf{I}_Q)$
- Marginal likelihood over $\boldsymbol{W}$:

$$p(\boldsymbol{Y}|\boldsymbol{X}, \beta) = \Pi_{n=1}^{N} \mathcal{N}(y_n; 0_D, \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}} + \beta^{-1}\mathbf{I}_D)$$

- Let $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}} + \beta^{-1}\mathbf{I}_D$, the log-likelihood is then:

$$L = -\frac{DN}{2}\log 2\pi - \frac{N}{2}\log det(\boldsymbol{K}) - \frac{1}{2}tr(\boldsymbol{K}^{-1}\boldsymbol{Y}\boldsymbol{Y}^{T})$$

- Maximize L over $\boldsymbol{K}$, i.e $\{\boldsymbol{X}, \beta\}$

GP-LVM: Replace $\boldsymbol{K}$ by a non-linear covariance matrix on the latent variables

Gaussian Process Latent Variable Model

GP-LVM:

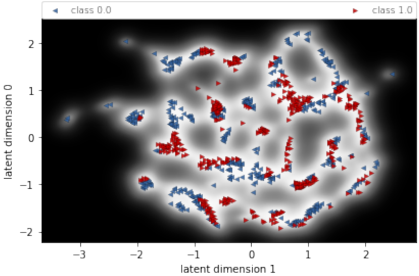
- Each dimension of the marginal likelihood is an independent **Gaussian process**
- Choice of covariance function determines the class of functions considered
- Example: **Radial Basis Function** kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(\frac{-||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\lambda^2}\right).$$

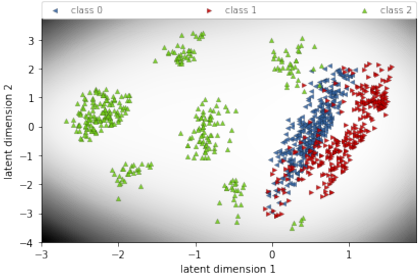
- Optimization with gradient-based method
- Implementation in `GPy`

Gaussian Process Latent Variable Model

Examples:



(a) The Swiss roll (error: 8.0 %)



(b) Oil flow dataset (error: 0.1 %)

Autoencoders

- Encoder $\boldsymbol{h} = f(\boldsymbol{x})$ which transforms the input to a hidden code
- Decoder which reconstructs the input from hidden code: $\boldsymbol{r} = g(\boldsymbol{h})$
- Minimise $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$

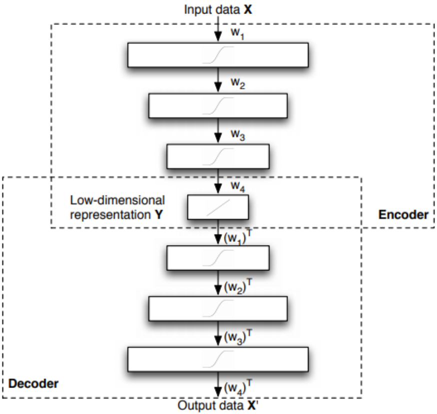


Figure: Example of autoencoder structure (van der Maaten et al., 2009).

Autoencoders

Figure: Autoencoder performance (3 hidden layers) on oil flow dataset.

Generative Topographic Mapping I

General Idea:

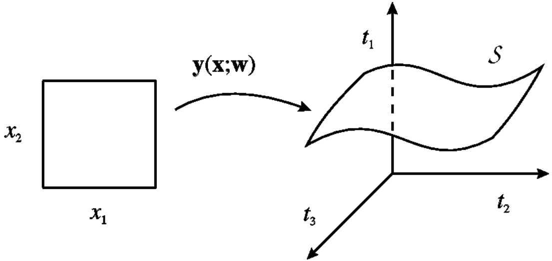


Image Credit: Bishop et al. (1998)

- GTM models a map from a d-dimensional latent space onto a d-dimensional manifold embedded in D-dimensional data space.

Generative Topographic Mapping (GTM) II

- Consider the following:
  - $\phi(\boldsymbol{x}; W)$: non-linear function mapping from latent space, X to data space Y, where W is a parameter matrix of weights.
  - Let $\phi(\boldsymbol{x}; W) = \boldsymbol{q}$. Then, the distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ and W is chosen to be Gaussian:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}, \beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left\{-\frac{\beta}{2}||\boldsymbol{q} - \boldsymbol{y}||\right\}$$

where β^{-1} is the variance of the distribution.

Generative Topographic Mapping (GTM) II

- Consider the following:
  - $\phi(\boldsymbol{x}; W)$: non-linear function mapping from latent space, X to data space Y, where W is a parameter matrix of weights.
  - Let $\phi(\boldsymbol{x}; W) = \boldsymbol{q}$. Then, the distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ and W is chosen to be Gaussian:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}, \beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left\{-\frac{\beta}{2}||\boldsymbol{q} - \boldsymbol{y}||\right\}$$

    where β^{-1} is the variance of the distribution.
- Integrating over the latent variables, we obtain:

$$p(\boldsymbol{y}|\boldsymbol{W}, \beta) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}, \beta)p(\boldsymbol{x})d\boldsymbol{x},$$

  with $p(\boldsymbol{x}) = \frac{1}{K}\sum_{i=1}^{K}\delta(\boldsymbol{x} - \boldsymbol{x_i})$.

Generative Topographic Mapping (GTM) II

- Consider the following:
  - $\phi(\boldsymbol{x}; W)$: non-linear function mapping from latent space, X to data space Y, where W is a parameter matrix of weights.
  - Let $\phi(\boldsymbol{x}; W) = \boldsymbol{q}$. Then, the distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ and W is chosen to be Gaussian:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}, \beta) = \left(\frac{\beta}{2\pi}\right)^{D/2} \exp\left\{-\frac{\beta}{2}||\boldsymbol{q} - \boldsymbol{y}||\right\}$$

  where β^{-1} is the variance of the distribution.
- Integrating over the latent variables, we obtain:

$$p(\boldsymbol{y}|\boldsymbol{W}, \beta) = \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}, \beta)p(\boldsymbol{x})d\boldsymbol{x},$$

  with $p(\boldsymbol{x}) = \frac{1}{K}\sum_{i=1}^{K}\delta(\boldsymbol{x} - \boldsymbol{x_i})$.
- Assuming $\boldsymbol{y}_n$ is i.i.d, parameters $\boldsymbol{W}$ and β are optimized using

$$\boldsymbol{L} = \sum_{n=1}^{N} \ln\left\{\frac{1}{K}\sum_{i=1}^{K} p(\boldsymbol{y}_n|\boldsymbol{x}_i, \boldsymbol{W}, \beta)\right\}.$$

t-Distributed Stochastic Neighbour Embedding (tSNE) I

Stochastic Neighbour Embedding (SNE)

- Interprets distances between data points as Gaussian conditional probabilities.
- Consider $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ in the observed D-dimensional data set. Then, $p_{j|i}$ the probability that $\boldsymbol{y}_j$ is a neighbour of $\boldsymbol{y}_i$ given as

$$p_{j|i} = \frac{\exp(||\boldsymbol{y}_i - \boldsymbol{y}_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(||\boldsymbol{y}_i - \boldsymbol{y}_k||^2/2\sigma_i^2)}.$$

t-Distributed Stochastic Neighbour Embedding (tSNE) I

Stochastic Neighbour Embedding (SNE)

- Interprets distances between data points as Gaussian conditional probabilities.
- Consider $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ in the observed D-dimensional data set. Then, $p_{j|i}$ the probability that $\boldsymbol{y}_j$ is a neighbour of $\boldsymbol{y}_i$ given as

$$p_{j|i} = \frac{\exp(||\boldsymbol{y}_i - \boldsymbol{y}_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(||\boldsymbol{y}_i - \boldsymbol{y}_k||^2/2\sigma_i^2)}.$$

- Similarly, for corresponding points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the d-dimensional space, we have

$$q_{j|i} = \frac{\exp(||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(||\boldsymbol{x}_i - \boldsymbol{x}_k||^2/2\sigma_i^2)}.$$

t-Distributed Stochastic Neighbour Embedding (tSNE) I

Stochastic Neighbour Embedding (SNE)

- Interprets distances between data points as Gaussian conditional probabilities.
- Consider $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ in the observed D-dimensional data set. Then, $p_{j|i}$ the probability that $\boldsymbol{y}_j$ is a neighbour of $\boldsymbol{y}_i$ given as

$$p_{j|i} = \frac{\exp(||\boldsymbol{y}_i - \boldsymbol{y}_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(||\boldsymbol{y}_i - \boldsymbol{y}_k||^2/2\sigma_i^2)}.$$

- Similarly, for corresponding points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the d-dimensional space, we have

$$q_{j|i} = \frac{\exp(||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(||\boldsymbol{x}_i - \boldsymbol{x}_k||^2/2\sigma_i^2)}.$$

- SNE minimises the objective function

$$\sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

t-Distributed Stochastic Neighbour Embedding (tSNE) II

Limitations of SNE

- Objective function can be difficult to optimize
- "Crowding" problem

t-Distributed Stochastic Neighbour Embedding (tSNE) II

Limitations of SNE

- Objective function can be difficult to optimize
- "Crowding" problem

tSNE (Van Der Maaten et al., 2009) comes to the rescue!

How?

- "Symmetrizes" objective function to obtain simpler gradients without losing quality of visualization. Modified objective:

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

  P,Q: joint probability distributions

  $p_{ij} = p_{ji}; q_{ij} = q_{ji} \forall i, j.$

- Models q_{ij} with a student-t distribution (which has heavier tails than the Gaussian).

Multi-dimensional Scaling

- These methods use some measure of proximity between points in high dimensional space to deduce corresponding location of points in a low dimensional space.
- Concept:
  - Let δ_{ij} denote a proximity value between observed D-dimensional points i and j; $i, j \in (1, ..., N)$.
  - The values of δ_{ij} form an $N \times N$ matrix, $\mathbf{\Lambda}$.
  - Given $\mathbf{\Lambda}$, MDS finds a set of vectors $(\boldsymbol{x}_1, ..., \boldsymbol{x}_N) \in \mathbb{R}^d; d << D$, such that the stress,

$$A = \sqrt{\frac{\sum_i \sum_j [f(\boldsymbol{x}_i, \boldsymbol{x}_j) - \delta_{ij}]^2}{\sum_i \sum_j \delta_{ij}^2}}.$$

    is minimised. f is chosen to be monotonic to preserve the ordering in the original data.

GTM, tSNE MDS in Action!

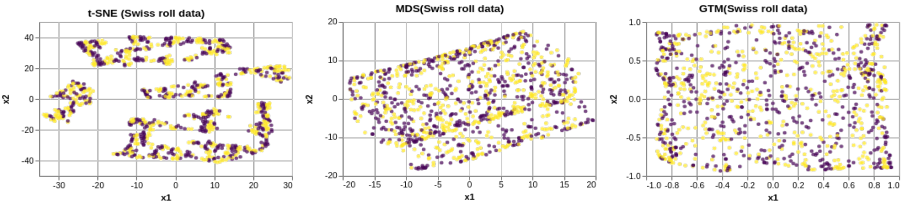


Figure: tSNE, MDS and GTM performance on artificial data

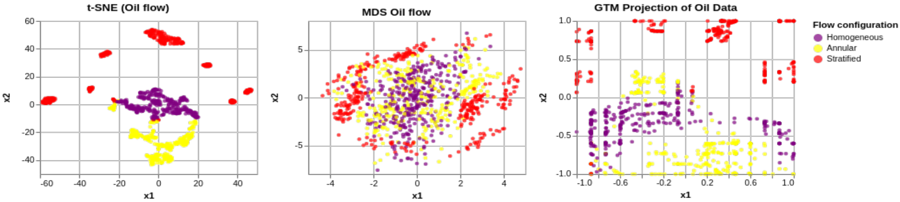


Figure: tSNE, MDS and GTM performance on oil flow dataset

General Comparisons

	Sp.GP-LVM	tSNE	MDS	GTM	KPCA p.	KPCA s.	KPCA RBF	Auto.
error (%)	8.4	28.1	30.2	33.8	25.4	13.9	25.3	23.9
trust.	0.9975	0.9996	0.8949	0.9634	0.8770	0.9473	0.8652	0.8704
continuity	0.9700	0.9916	0.9931	0.9894	0.9911	0.9962	0.9924	0.9930
wall time(s)	132	6.14	3	1.4	0.0351	0.0607	0.0751	0.638

Table: Performance metrics of various dimensionality reduction techniques on artificial data

	Sp.GP-LVM	tSNE	MDS	GTM	KPCA p.	KPCA s.	KPCA RBF	Auto.
error (%)	0.0	0.3	28.9	4.3	32.2	35.2	15.2	14.4
trust.	0.9969	0.9985	0.9306	0.9898	0.9275	0.8957	0.8182	0.9143
continuity	0.9769	0.9964	0.9248	0.9836	0.9886	0.9910	0.9578	0.9707
wall time(s)	144	6.41	3.02	1.48	0.0326	0.0785	0.0774	0.276

Table: Performance metrics of various dimensionality reduction techniques on multi-phase oil flow data

Conclusion

- GP-LVM is a quite robust framework but with a high computation cost ($O(N^3)$): variants include sparsification, hierarchical dynamic models, Bayesian optimisation.
- Further research could explore the optimal choice of hyperparameters to increase performance of dimensionality reduction techniques.
- Inference methods with application to human motion tracking, shape modelling, assisted animation.

References I

Christopher M Bishop, Markus Svensén, and Christopher KI Williams. Gtm: The generative topographic mapping. *Neural computation*, 10(1): 215–234, 1998.

Héléna Alexandra Gaspar. ugtm: A python package for data modeling and visualization using generative topographic mapping. *Journal of Open Research Software*, 6(1), 2018.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

References II

Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.