

Contents

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

Dimensionality reduction

1 The curse of dimensionality

- Feature space의 차원이 높으면 연산에 걸리는 시간이 증가할 뿐아니라 변수들의 관계를 파악하기 어렵다.
- 단위사각형에서 무작위로 선택한 두 점의 평균거리는 0.52이고, 단위육면체의 경우 0.66이다. 1백만차원의 hypercube에선 거리의 기대값은 408.25가 된다.
- 훈련의 어려움과 함께 새로운 관찰값을 평가할 때, 이 값은 기존의 표본들과 '멀리' 떨어져 있을 가능성이 높다. 따라서 예측은 지나친 extrapolation에 근거하게 되고 overfitting에 취약하게 된다.
- 가장 좋은 대안은 더 많은 자료를 모으는 것이지만 현실적으로 불가능한 경우가 많으므로 차선으로 dimensionality reduction을 사용할 수 있다.

2 Dimensionality reduction

- Dimensionality reduction의 기본 전제는 Manifold Hypothesis이다.

Most real-world high-dimensional datasets lie close to a much lower-dimensional manifolds embedded within the high-dimensional space.

거리를 이용...

2.1 Projection

선형변환, matrix factorization

2.2 Manifold learning

- 높은 차원의 정보를 구조화하여 가장 유사한 구조를 낮은 차원에 구현하는 것이다.
- 문제는 '가까운' 점과 '유사성'의 정의이다.

rotation과 scaling에 강하다., graph layout

- Overfitting, curse of dimensionality: Feature의 수와 모형의 performance는 비례하지 않는다.
- Visualization: Correlation 등의 descriptive statistics 만으로 자료의 특징을 이해하기 어렵다.
- Efficiency: feature의 수와 모형의 복잡도는 비례하기 때문에 훈련에 차원의 소비가 많다.
- PCA, t-SNE, UMAP
 - visualization
 - preprocessing feature extraction
 - compression

Contents ↗ *

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

3 Unsupervised learning

- Label이 없는 자료를 사용한 분석 방법으로 자료의 구조를 이해하는데 도움
- TensorBoard Embedding Projector: [3D represenation \(<https://projector.tensorflow.org/>\)](https://projector.tensorflow.org/)
- Classification
 - K-Means algorithm

4 principal component analysis (dimension reduction)

- PCA는 nonsupervised learning algorithm으로 dimension reduction의 기본 모형

4.1 Diagonalizing the covariance

- 일반적으로 분산이 클수록 더 많은 정보를 포함하고 있는 경우가 많다.
- 자료의 '전체적인' 분산 중에서 어떤 'feature'가 분산을 많이 설명하는지는 대부분 자료분석의 핵심이다.

$$\text{Cov}(x_1, x_2, x_3) = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ 1 & 3 & 4 \end{bmatrix}, \quad \text{Cov}(z_1, z_2) = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$

- 자료는 K 의 feature로 구성되어 있고 feature들 사이에 어느정도의 상관관계가 있는 것이 일반적이므로 특정 feature의 기여도를 계산하기 쉽지 않다.
- Feature들의 선형결합으로 새로운 변수(혹은 basis)들을 정의하고 직교조건을 이용하여 이들 변수들의 설명력을 계산한다.

Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_variance_ratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.03449)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

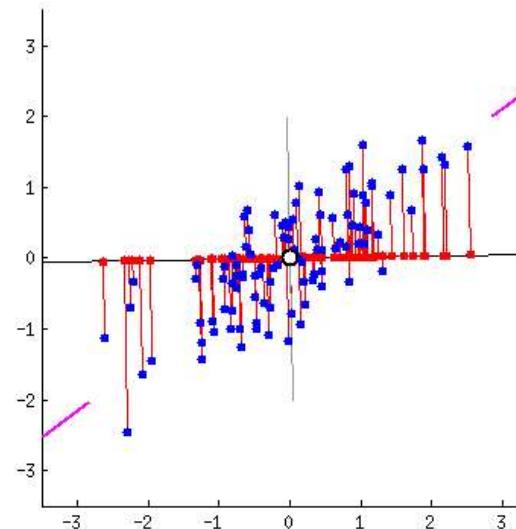
4.2 Formulation

maximum variance

minimum projection error

PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized (Hotelling, 1933). Equivalently, it can be defined as the linear projection that minimizes the average projection cost, defined as the mean squared distance between the data points and their projections (Pearson, 1901).

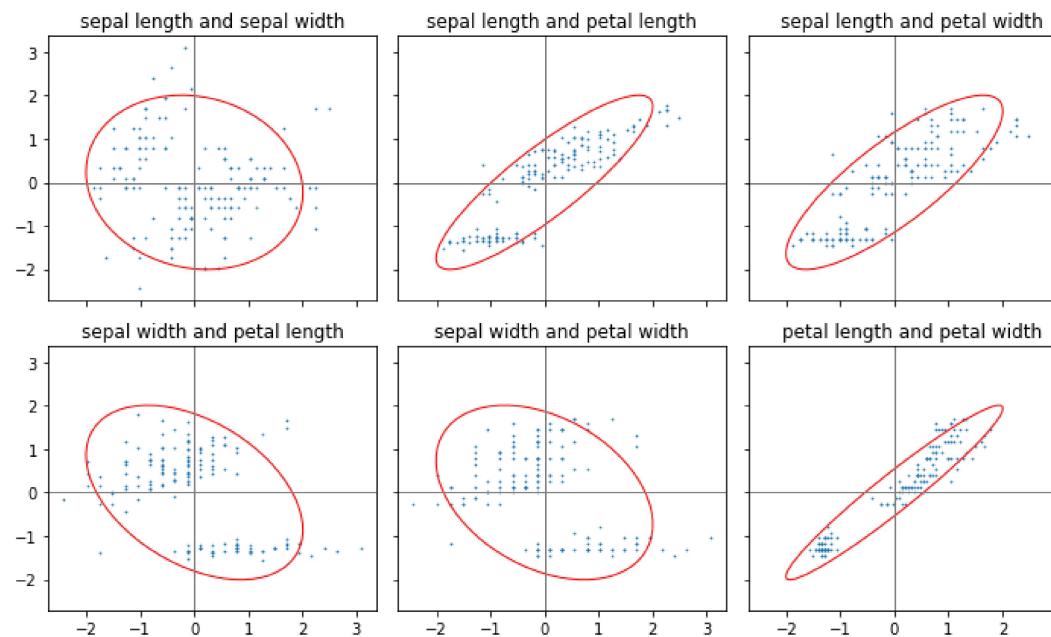
- PCA는 표본의 특징을 가장 잘 반영할 수 있는 (타원체의) 축 혹은 basis를 찾는 문제로 요약할 수 있다.
 - 계산과정은 일차원 projection을 반복적으로 적용하는 것과 동일하지만 전체모형을 projection으로 이해하기는 어렵다.
 - 분산이 가장 큰 방향이 해당 축이 된다.
 - MSE를 가장 극소화하는 1D projection plane
 - 그림 설명 블로그 (<http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/>)



<https://www.machinelearningplus.com/machine-learning/principal-components-analysis-pca-better-explained/> (<https://www.machinelearningplus.com/machine-learning/principal-components-analysis-pca-better-explained/>)

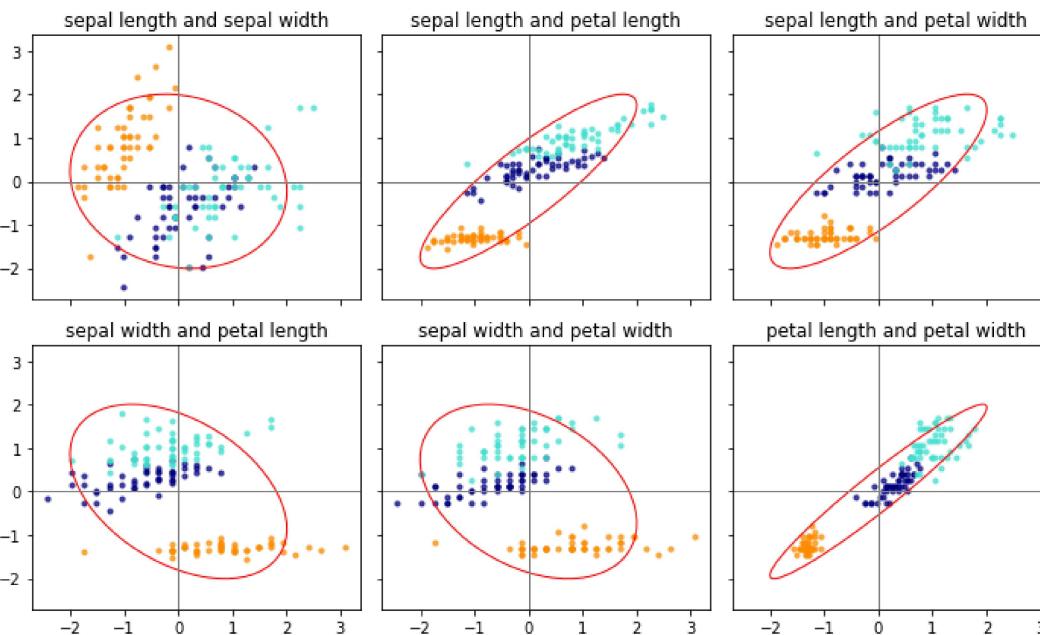
Contents ⚙ *

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - ▼ 11.1 Approximate high dimensional structures
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity



Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity



4.3 Orthonormal coordinate system

4.3.1 Diagonalizable matrix

Theorem 1 A is diagonalizable if and only if A has linearly independent eigenvectors.

$$A = PDP^{-1} \text{ where } D = \text{diag}(\lambda_j)$$

Theorem 2 Real symmetric matrices are diagonalizable by orthonormal matrices.

$$A = QDQ^T \text{ where } QQ^T = Q^TQ = I, D = \text{diag}(\lambda_j)$$

Contents

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional structures
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

4.3.2 Principal components

- Feature의 covariance matrix 는 실수를 원소로 하는 대칭행렬이고 positive-definite 이므로 모든 eigenvalue는 양수이고 orthonormal한 eigenvector가 존재한다.

$$\text{Cov}(X) \propto X^T X = W \Lambda W^T, \quad \Lambda = \text{diag}(\lambda_j), \quad \lambda_j > 0$$

- $W_{col,j}^T x$ 는 vector x 의 $W_{col,j}$ 에 대한 projection scalar 이므로 W 의 column으로 구성된 orthogonal basis의 j 번째 축의 좌표가 된다.
- Eigenvector로 이루어진 orthogonal basis가 생성하는 vector space에서 covariance matrix에 대응하는 행렬은 대각행렬이다.
- 새로운 coordinate system에서 각 principal component의 분산은 eigenvalue이고 공분산은 0이다.

$$\text{Cov}(XW) = W^T \text{Cov}(X) W \propto W^T X^T X W = W^T (W \Lambda W^T) W = (W^T W) \Lambda (W^T W) = \Lambda$$

- 각 축에 해당하는 새로운 변수 $W_{col,j}^T x$ 가 전체 분산 중에서 설명하는 비율은 해당 eigenvalue의 상대적인 크기로 계산한다.

$$\text{Explained Variance of principal component } j = \frac{\lambda_j}{\sum_{i=1}^K \lambda_i}$$

Contents

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.03404)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

4.4 Principal component의 해석

- 분산은 scale의 영향을 받으므로 모든 feature는 평균 0, 분산이 1이 되도록 scaling한다.
- Xw_1 의 분산을 가장 크게하는 일차원 subspace 혹은 basis w_1 가 first component이다.
- w_1 와 직교하면서 분산을 가장 크게하는 두 번째 basis w_2 를 찾는다.
- 필요한 만큼 앞에서 구한 (모든) basis들과 직교하면서 분산을 가장 크게하는 basis를 찾는 작업을 반복한다.

4.4.1 First principal component

$$\max_{\|w\|=1} \sum_{i=1}^n (w^\top x_i)^2 = \max_{\|w\|=1} w^\top X^T X w$$

$$w = \arg \max_w \frac{w^\top X^T X w}{w^\top w}$$

- 위 문제의 일계조건은 다음과 같다.

$$\frac{2(w^\top w)X^T X w - 2(w^\top X^T X w)w}{w^\top w} = 0$$

- $w^\top w$ 은 scalar 이므로 다음과 같이 정리할 수 있다.

$$X^T X w = \frac{(w^\top X^T X w)}{w^\top w} w$$

- $\lambda = \frac{(w^\top X^T X w)}{w^\top w}$ 라고 하면,

$$(X^T X)w = \lambda w$$

- 즉 목적함수의 극대값은 $X^T X$ 의 가장 큰 eigenvalue에 해당하며, 최적해는 해당 eigenvalue에 대응하는 eigenvector가 된다.

Contents

1	The curse of dimensionality
▼ 2	Dimensionality reduction
2.1	Projection
2.2	Manifold learning
3	Unsupervised learning
▼ 4	principal component analysis (dime)
4.1	Diagonalizing the covariance
4.2	Formulation
4.3	Orthonormal coordinate system
4.3.1	Diagonalizable matrix
4.3.2	Principal components
▼ 4.4	Principal component의 해석
4.4.1	First principal component
4.4.2	Second principal component
▼ 4.5	Sklearn implementation
4.5.1	n_component
4.5.2	whiten
4.5.3	explained_varianceratio
▼ 5	Kernel PCA
5.1	Eigenvector
5.2	Kernel matrix
▼ 5.3	Normalization
5.3.1	centered features
5.3.2	orthonormal basis
5.4	Dimension reduction
▼ 5.5	Memory
5.5.1	Incremental PCA
5.5.2	Randomized PCA
5.6	PCA의 가정과 한계
6	Manifold learning algorithm
7	Multi-Dimensional Scaling (MDS)
8	Locally Linear Embedding (LLE)
9	ISOMAP
▼ 10	T-SNE, t-distributed stochastic nei
10.1	Manifold approximation
10.2	Low dimensional embedding
10.3	training
10.4	TSNE in 3D
10.5	t-SNE의 가정과 한계
▼ 11	UMAP (https://arxiv.org/abs/1802.03404)
▼ 11.1	Approximate high dimensional
11.1.1	Simplices
11.1.2	Simplicial complex
11.1.3	Varying distance 과 fuzzy c
11.1.4	fuzzy simplicial complex
11.1.5	Local connectivity
11.1.6	similarity

4.4.2 Second principal component

- 첫 번째 축을 찾은 후, 해당 축에 대한 정보를 제거한 vector(rejection)에 동일한 방법을 적용하여 새로운 축을 정의할 수 있다.
- 위 과정은 전체 feature의 수만큼 반복해서 적용할 수 있으며, s 번째 component는 위 과정을 반복 적용해서 구할 수 있다.
- 각 단계별로 이전 단계에서 사용한 정보를 모두 제거하고 남아있는 정보에 대해 MSE를 가장 작게하는 축을 식별한다.

$$\begin{aligned}\hat{X}_{(p)} &= \hat{X}_{(p-1)} - (X\mathbf{w}_{(p-1)})\mathbf{w}_{(p-1)}^\top, \quad 1 \leq p \leq K \\ &= X - \sum_{s=1}^{p-1} (X\mathbf{w}_s)\mathbf{w}_s^\top\end{aligned}$$

- 만일 $\text{Cov}(X)$ 가 대각행렬이라면 $\mathbf{w}_j = e_j$ 와 같은 형태가 된다. 즉, 전단계에서 정보를 추출해낸 column은 모두 0으로 바꾸어 계산한다.

$$\begin{aligned}\mathbf{w}_{(p)} &= \arg \max_{\|\mathbf{w}\|=1} \mathbf{w}^\top \hat{X}_{(p)}^T \hat{X}_{(p)} \mathbf{w} \\ &= \arg \max_{\mathbf{w}} \frac{\mathbf{w}^\top \hat{X}_{(p)}^T \hat{X}_{(p)} \mathbf{w}}{\mathbf{w}^\top \mathbf{w}}\end{aligned}$$

- 위 극대화 문제의 해는 크기 순으로 정리한 p 번째 eigenvalue에 해당하는 eigenvector가 된다.
- 실제 계산에선 singular value decomposition을 이용하여 eigenvector를 직접 계산한다.
- 아래 표현에서 V 행렬의 각 열이 principal component이다. 아래 Σ 는 분산이 아니라 singular values를 원소로 하는 대각행렬이다.

$$(X - \bar{X}) = U\Sigma V^\top, \quad \Sigma_{K \times n} = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \end{bmatrix}$$

Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

4.5 Sklearn implementation

```
sklearn.decomposition.PCA(n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10, power_iteration_normalizer='auto', random_state=None)
```

4.5.1 n_component

- n_components는 principal component의 갯수를 지정한다. 정수를 사용하여 갯수를 직접 지정하거나, 전체 분산에서 설명이 되어야할 비율을 0과 1 사이의 실수로 지정할 수 있다.
- n_components=='mle'는 [Minka's MLE](https://vismod.media.mit.edu/tech-reports/TR-514.pdf) (<https://vismod.media.mit.edu/tech-reports/TR-514.pdf>)로 hypothesis의 likelihood를 극대화하는 principal component의 갯수 d 를 계산한다. 다음은 인용한 논문의 식 (78)을 우리 notation을 사용하여 정리한 것이다.

$$\Pr(d|K) = \left(\prod_{j=1}^d \lambda_j \right)^{-n/2} \hat{v}^{-N(K-d)/2} n^{-(m+K)/2}$$

$$\hat{v} = \frac{\sum_{j=d+1}^K \lambda_j}{K-d}, \quad m = Kd - d(d+1)$$

- None 은 가능한 많은 축을 식별한다. n_components = min(n_samples, n_features) - 1

4.5.2 whiten

- Sphering 혹은 whitening trick은 feature들을 서로 독립적으로 만들기 위해 사용할 수 있다.

$$\text{Var}\left(\Sigma_X^{-1/2} X\right) = I_K$$

- 정보의 손실을 있지만 PCA의 결과를 적용할 모형의 가정에 따라 성능향상에 도움이 될 수 있다.

4.5.3 explained_varianceratio

- Attribute explained_variance_ratio_ 는 전체 분산에서 각 principal component가 설명할 수 있는 비율을 보여준다.

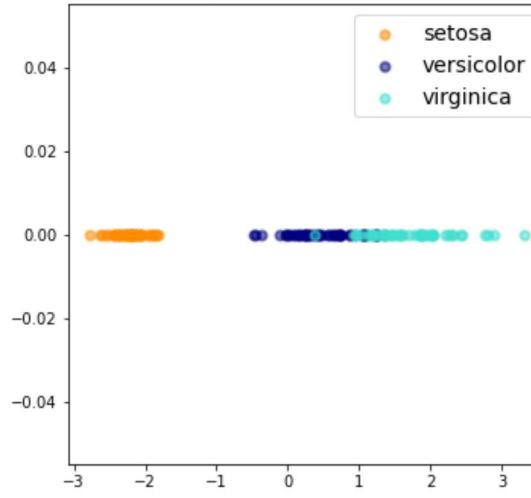
```
pca.explained_variance_ratio_
>>> array([0.72962445, 0.22850762, 0.03668922])
```

Contents ⚙

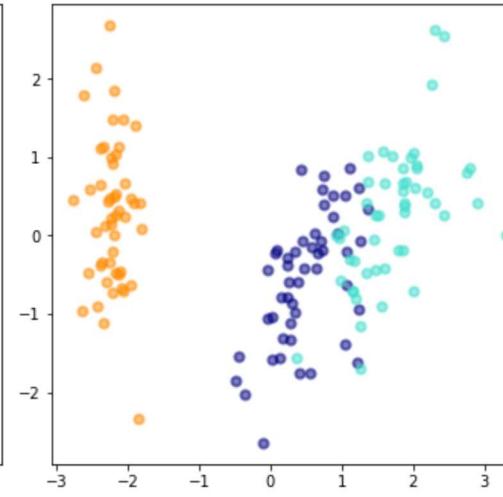
- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

Dimensionality Reduction with PCA

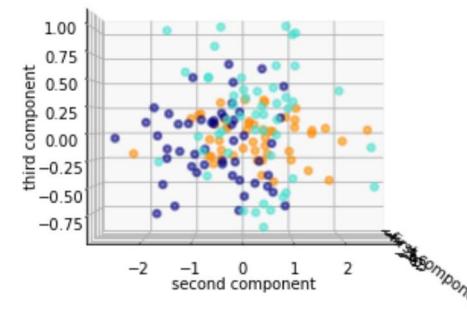
Single Component PCA



PCA with Two Components



PCA with Three Components



sepal length sepal width petal length petal width

0	0.521066	-0.269347	0.580413	0.564857
1	0.377418	0.923296	0.024492	0.066942
2	-0.719566	0.244382	0.142126	0.634273

Contents

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.0>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

5 Kernel PCA

- PCA 역시 $X^T X$ 로 요약한 정보만을 이용하므로 SVM에서 소개한 kernel trick을 적용하기 좋은 모형이다.

$$\text{Cov}(X) = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \cdot \mathbf{x}_i^T = \frac{X^T X}{n}$$

- Feature를 새로운 feature space F 에 대응시켜 이 변환된 값에 PCA를 적용한다.

$$\varphi: \mathbb{R}^K \rightarrow \mathbb{R}^D$$

- $\varphi(\mathbf{x}_i)$ 가 centered 되어 있다고 가정하자.

$$\mu = \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i) = 0$$

$$C = \text{Cov}(\varphi(X)) = \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i) \varphi(\mathbf{x}_i)^T$$

- PCA와 동일한 방법으로 $\varphi(\mathbf{x})$ 공간에서 principal component를 추출한다. 다만 kernel function을 이용하므로 유도과정이 약간 성가시다. [참고자료](http://www.cs.haifa.ac.il/~rita/uml_course/lectures/KPCA.pdf) (http://www.cs.haifa.ac.il/~rita/uml_course/lectures/KPCA.pdf)

5.1 Eigenvector

- 일반적으로 AA^T 의 $\lambda \neq 0$ 에 대응하는 eigenvector는 A의 column space의 vector이다.

$$AA^T v = \lambda v, \quad v = A \begin{pmatrix} 1 \\ \vdots \\ \lambda \\ \vdots \end{pmatrix}$$

- 위 결과를 $Cv = \lambda v$ 에 적용하면 eigenvector를 다음과 같이 표현할 수 있다.

$$v = \frac{1}{\lambda n} \sum_{i=1}^n \varphi(\mathbf{x}_i) \varphi(\mathbf{x}_i)^T v = \sum_{i=1}^n \frac{\varphi(\mathbf{x}_i)^T v}{\lambda n} \varphi(\mathbf{x}_i) = \sum_{i=1}^n \alpha_i \varphi(\mathbf{x}_i)$$

scalar

- 따라서 eigenvector를 계산하기 위해서는 $\varphi(\mathbf{x}_i)$ 가 필요하지만, feature map은 명시적으로 고려하지 않으므로 kernel function의 형태로 바꾸어야 한다.

Contents

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

5.2 Kernel matrix

- j 번째 eigenvector v_j 에 대응하는 가중값을 $\alpha_j = (\alpha_{j1}, \dots, \alpha_{jn})$ 으로 표시하고, 위 결과들을 $Cv_j = \lambda_j v_j$ 에 대입하여 정리해보자.

$$\left(\frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i) \varphi(\mathbf{x}_i)^\top \right) \left(\sum_{h=1}^n \alpha_{jh} \varphi(\mathbf{x}_h) \right) = \lambda_j \left(\sum_{h=1}^n \alpha_{jh} \varphi(\mathbf{x}_h) \right)$$

$$\sum_{i=1}^n \left(\varphi(\mathbf{x}_i) \sum_{h=1}^n \alpha_{jh} k(\mathbf{x}_i, \mathbf{x}_h) \right) = n \lambda_j \sum_{h=1}^n \alpha_{jh} \varphi(\mathbf{x}_h)$$

- 양변에 $\varphi(\mathbf{x}_1)^\top$ 을 왼쪽에서 곱해주면

$$\sum_{i=1}^n \left(\varphi(\mathbf{x}_1)^\top \varphi(\mathbf{x}_i) \sum_{h=1}^n \alpha_{jh} k(\mathbf{x}_i, \mathbf{x}_h) \right) = n \lambda_j \sum_{h=1}^n \alpha_{jh} \varphi(\mathbf{x}_1)^\top \varphi(\mathbf{x}_h)$$

$$\sum_{i=1}^n \left(k(\mathbf{x}_1, \mathbf{x}_i) \sum_{h=1}^n \alpha_{jh} k(\mathbf{x}_i, \mathbf{x}_h) \right) = n \lambda_j \sum_{h=1}^n \alpha_{jh} k(\mathbf{x}_1, \mathbf{x}_h)$$

- K 를 $i, j = 1, \dots, n$ 에 대해 $k(\mathbf{x}_i, \mathbf{x}_j)$ 를 원소로 하는 kernel matrix 혹은 Gram matrix 라고 하면, 다음과 같이 정리할 수 있다.

$$K^2 \alpha_j = n \lambda_j K \alpha_j$$

- 좌우변의 K 는 $\lambda_j \neq 0$ 에 대응하는 eigenvector α_j 를 결정하는데 아무런 영향을 미치지 않으므로 소거해도 무방하다. (아래 조건을 만족하는 λ_j 와 α_j 는 위 조건 역시 만족 한다.)

$$K \alpha_j = n \lambda_j \alpha_j$$

- K 는 kernel 함수로 계산이 가능하므로 K 의 eigenvector를 찾는 문제가 된다.

Contents

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

5.3 Normalization

5.3.1 centered features

- 일반적으로 $\varphi(\mathbf{x}_i)$ 의 평균은 0이 아니므로 이를 조정해주어야 한다.
- 먼저 centered feature를 구한 후 이를 kernel matrix에 반영한다.

$$\tilde{\varphi}(\mathbf{x}_i) = \varphi(\mathbf{x}_i) - \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i)$$

$$\begin{aligned}\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) &= \varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j) \\ &= \left(\varphi(\mathbf{x}_i) - \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i) \right)^\top \left(\varphi(\mathbf{x}_j) - \frac{1}{n} \sum_{i=1}^n \varphi(\mathbf{x}_i) \right) \\ &= K(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{n} \sum_{h=1}^n K(\mathbf{x}_i, \mathbf{x}_h) - \frac{1}{n} \sum_{h=1}^n K(\mathbf{x}_j, \mathbf{x}_h) - \frac{1}{n^2} \sum_{h=1}^n \sum_{k=1}^n K(\mathbf{x}_h, \mathbf{x}_k)\end{aligned}$$

- 행렬형태로 바꾸면 다음과 같다.

$$\tilde{K}(\mathbf{x}_i, \mathbf{x}_j) = K - 2\mathbf{1}_{1/n} K + \mathbf{1}_{1/n} \mathbf{1}_{1/n}^\top, \quad \mathbf{1}_{1/n} = \left[\frac{1}{n} \right]_{n \times n}$$

5.3.2 orthonormal basis

- Eigenvector는 유일하게 결정되지 않으므로 mapping에 사용할 orthonormal 한 basis를 구하기 위해선 unit vector 조건이 필요하다.

$$\mathbf{v}_j^\top \mathbf{v}_j = 1 \Rightarrow \sum_{i=1}^n \sum_{j=1}^n \alpha_{ji} \alpha_{ij} \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_l) = 1 \Rightarrow \alpha_j^\top K \alpha_j = 1$$

- $K \alpha_j = n \lambda_j \alpha_j$ 의 양변에 α_j^\top 를 왼쪽에서 곱하고 위 조건을 대입하면 다음과 같이 정리할 수 있다.

$$n \lambda_j \alpha_j^\top \alpha_j = 1$$

- 임의의 관찰값 \mathbf{x} 의 j 번째 principal component에 대한 scalar project은 다음과 같이 계산한다.

$$\varphi(\mathbf{x}) \cdot \mathbf{v}_j = \sum_{i=1}^n \alpha_{ji} \varphi(\mathbf{x}) \cdot \varphi(\mathbf{x}_i) = \sum_{i=1}^n \alpha_{ji} k(\mathbf{x}, \mathbf{x}_i)$$

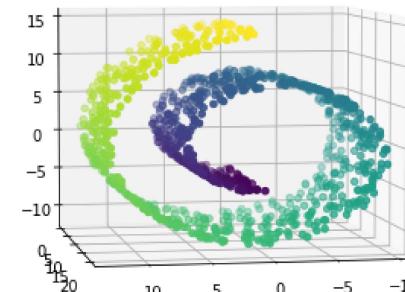
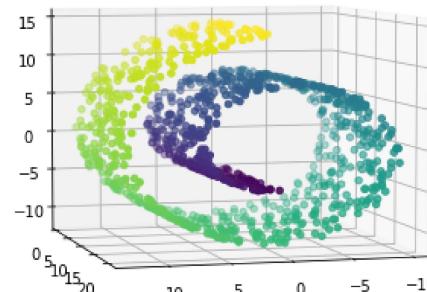
Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

5.4 Dimension reduction

- K 차원의 input space를 d 차원으로 줄이려면 kernel matrix \tilde{K} 의 값이 가장 큰 d 개의 eigenvalue에 대응하는 eigenvector를 basis로 하는 vector space를 선택한다.

Swiss Roll in 3D



```
sklearn.decomposition.KernelPCA(n_components=None, *, kernel='linear', gamma=None, degree=3, coef0=1, kernel_params=None, alpha=1.0, fit_inverse_transform=False, eigen_solver='auto', tol=0, max_iter=None, iterated_power='auto', remove_zero_eig=False, random_state=None, copy_X=True, n_jobs=None)
```

kernel: 'linear' (default), 'poly', 'rbf', 'sigmoid', 'cosine', 'precomputed'

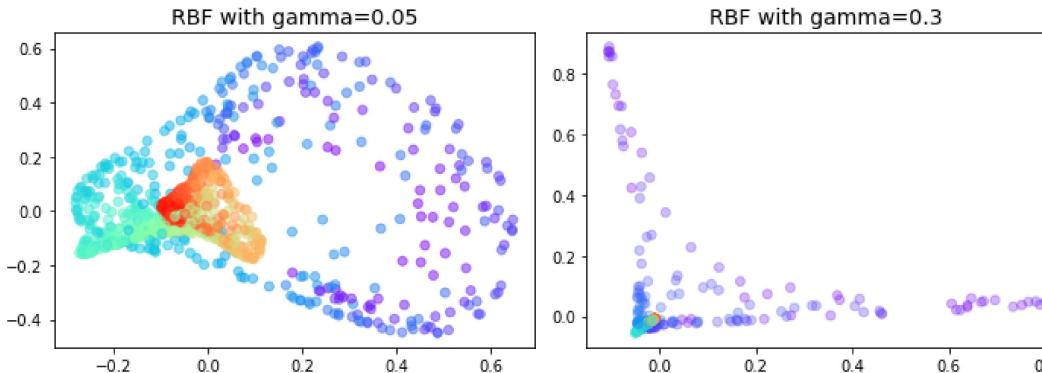
gamma: Kernel coefficient for rbf, poly and sigmoid kernels. Ignored by other kernels. If gamma is None, then it is set to 1/n_features



Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
topology
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

Kernel PCA with RBF Kernel



Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

```
from scipy.spatial.distance import pdist, squareform
from scipy import exp
from scipy.linalg import eigh
import numpy as np
def rbf_kernel_pca(X, gamma, n_components):
    """
    RBF kernel PCA implementation.

    Parameters
    -----
    X: {NumPy ndarray}, shape = [n_examples, n_features]
        Input data
    gamma: float
        Tuning parameter of the RBF kernel
    n_components: int
        Number of principal components to return

    Returns
    -----
    X_pc: {NumPy ndarray}, shape = [n_examples, k_features]
        Projected dataset
    """
    # Calculate pairwise squared Euclidean distances
    # in the MxN dimensional dataset.
    sq_dists = pdist(X, 'sqeuclidean')
    # Convert pairwise distances into a square matrix.
    mat_sq_dists = squareform(sq_dists)
    # Compute the symmetric kernel matrix.
    K = exp(-gamma * mat_sq_dists)
    # Center the kernel matrix.
    N = K.shape[0]
    one_n = np.ones((N,N)) / N
    K = K - one_n.dot(K) - K.dot(one_n) + one_n.dot(K).dot(one_n)
    # Obtaining eigenpairs from the centered kernel matrix
    # scipy.linalg.eigh returns them in ascending order
    eigvals, eigvecs = eigh(K)
    eigvals, eigvecs = eigvals[::-1], eigvecs[:, ::-1]
    # Collect the top k eigenvectors (projected examples)
    X_pc = np.column_stack([eigvecs[:, i]
                           for i in range(n_components)])
    return X_pc
```

Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

5.5 Memory

- PCA에선 SVD algorithm을 적용하기 위해 전체 자료를 모두 메모리에 올려야 한다.
- Sklearn은 메모리 문제를 해결하기 위한 몇 가지 algorithm을 제공하고 있다.

5.5.1 Incremental PCA

- 표본을 여러 batch로 나누어 계산한다. IncrementalPCA와 partial_fit method를 사용한다.
- 새로운 표본이 추가되는 경우 update에 유리하다.
- 자료의 크기가 아주 크다면 Numpy의 np.memmap으로 disk 공간을 memory로 활용할 수 있다. IncrementalPCA와 함께 사용한다.

5.5.2 Randomized PCA

- PCA의 keyword로 축소할 principal component 만 근사하므로 속도가 빠르다.

5.6 PCA의 가정과 한계

- 가정
 - 선형변환을 전제로 orthogonal basis를 찾는다.
 - 분석의 목적에서 분산이 자료의 중요한 특성을 설명할 수 있다.
- 특징
 - Feature의 분산을 기준으로 표본을 가급적 멀리 흩트려뜨리는 방식으로 작용
 - Clustering보다는 dimension reduction의 의미가 강함
- 한계
 - 새로운 basis가 갖는 의미가 불분명하다.
 - 분류 등에 중요하지 않지만 분산이 큰 feature가 있다면 표본의 본질을 제대로 반영하지 못할 수 있다.
- PCA의 한계는 unsupervised learning algorithm의 일반적인 한계이다.
- Supervised learning 모형인 LDA는 class 사이의 거리를 가장 크게하는 축을 식별하는 반면 unsupervised learning 모형인 PCA는 전체 feature의 분산에 근거해 '차별화'한다.

6 Manifold learning algorithm

https://lovit.github.io/nlp/representation/2018/09/28/mds_isomap_lle/ (https://lovit.github.io/nlp/representation/2018/09/28/mds_isomap_lle/)

Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

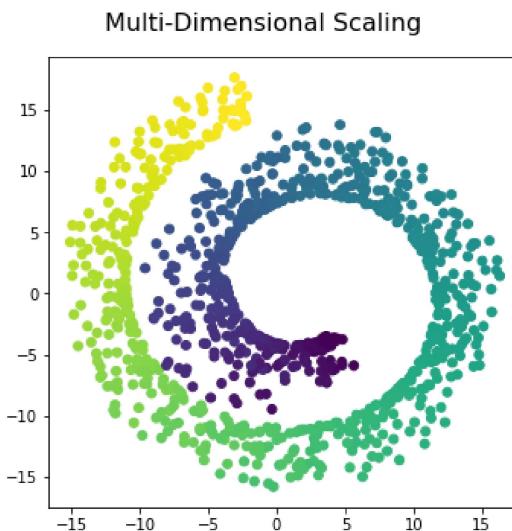
7 Multi-Dimensional Scaling (MDS)

- 원 공간에서 모든 점들 사이의 거리 w_{ij} 를 계산하고 낮은 차원의 공간에서 이 거리를 가장 잘 유지할 수 있는 점들을 찾는다.

$$(y_1, \dots, y_n) = \arg \min_{y_1, \dots, y_n} \sum_{i < j} (\|y_i - y_j\| - w_{ij})^2$$

- 차원이 높은 공간에서 '거리'가 커질수록 그 크기는 무의미할 가능성이 높지만, MDS는 가까운 거리의 점들과 먼 거리의 점들의 거리를 같은 중요도로 취급한다. 가까운 점들의 위치를 제대로 복원하는데 약점이 있다.

```
sklearn.manifold.MDS(n_components=2, *, metric=True, n_init=4, max_iter=300, verbose=0, eps=0.001, n_jobs=None, random_state=None, dissimilarity='euclidean')
```



Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.03404)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

8 Locally Linear Embedding (LLE)

- MDS와 다르게 국지적인 정보에 집중한다.
- 1. 모든 점들 사이의 거리를 계산하여 각 점 x_i 와 가까운 k 개의 점을 구한다. 이 집합을 m_i 라고 하자.
- 2. m_i 에 속한 점들의 선형결합으로 x_i 를 근사한다.

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}_{ij}} \|x_i - \sum_{j \in m_i} w_{ij} x_j\|^2, \quad w_{ij} = 0 \text{ if } j \notin m_i \text{ and } \sum_{j \in m_i} w_{ij} = 1$$

- 1. 선형결합의 정보를 잘 유지할 수 있는 y 를 계산한다.

$$\hat{\mathbf{z}} = \arg \min_{\mathbf{z}} \|z_i - \sum_{j \in m_i} \hat{w}_{ij} z_j\|^2$$

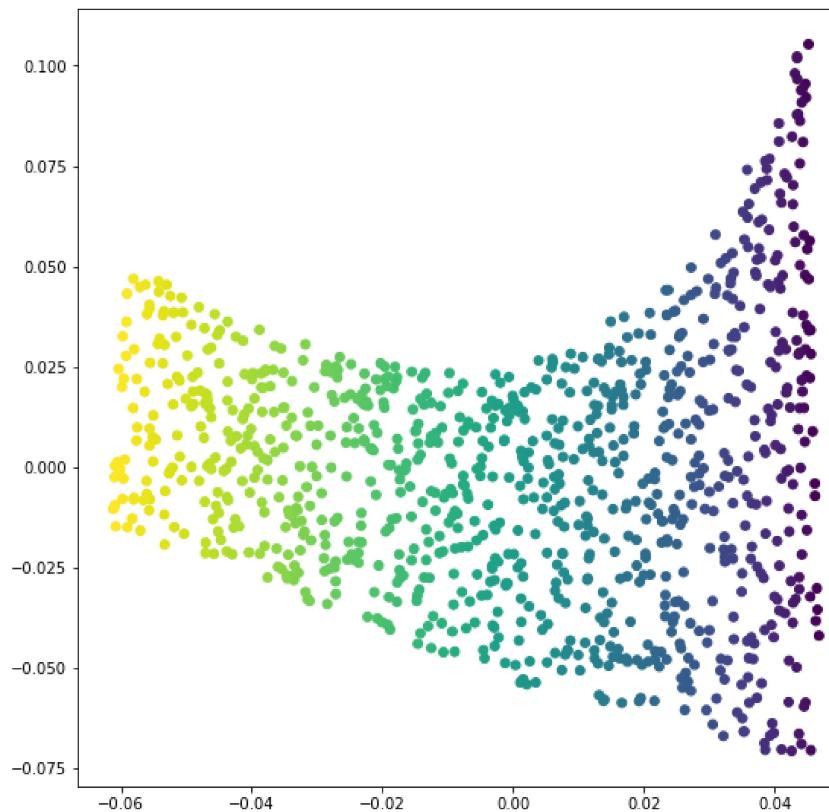
- 거리에 대한 정보만을 사용하므로 회전과 같은 변환에 구애받지 않으며, 마찬가지 이유로 결과물 역시 축의 순서나 방향이 달라질 수 있다.

```
sklearn.manifold.LocallyLinearEmbedding(*, n_neighbors=5, n_components=2, reg=0.001, eigen_solver='auto', tol=1e-06, max_iter=100, method='standard', hessian_tol=0.0001, modified_tol=1e-12, neighbors_algorithm='auto', random_state=None, n_jobs=None)
```

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03449>)
 - ▼ 11.1 Approximate high dimensional manifolds
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

Locally Linear Embedding



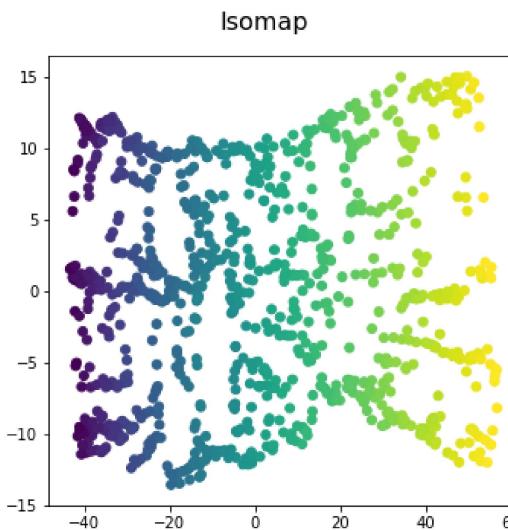
9 ISOMAP

- 가까운 점들을 통해서만 이동할 수 있는 거리 (통과해야하는 점의 수로 거리를 재며 geodesic distance, shortest-path distance) 를 사용하여 도약이 필요한 점들 사이의 거리는 정의되지 않는다.

```
sklearn.manifold.Isomap(*, n_neighbors=5, radius=None, n_components=2, eigen_solver='auto', tol=0, max_iter=None, path_method='auto', neighbors_algorithm='auto', n_jobs=None, metric='minkowski', p=2, metric_params=None)
```

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
- ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
- 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity



10 T-SNE, t-distributed stochastic neighbor embedding

- 주 목적은 시각화를 위한 low dimensionality reduction
- Input space에서 관찰값들의 결합확률을 계산하고 이 확률분포를 낮은 차원의 feature space로 mapping한다.
- 첫 번째 단계에선 Gaussian 분포를 가정하고 조건부 분포를 계산하고 두 번째 단계에선 t-분포(자유도가 1인 Cauchy distribution)를 이용하여 Kullback–Leibler divergence를 극소화하는 방법을 사용한다.
- Feature의 수가 많을 경우 PCA 등을 이용하여 차원을 축소한 후 이 자료에 t-SNE를 적용하면 속도 뿐 아니라 noise를 감소시킬 수 있다. 참고 (<https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html#sklearn-manifold-tsne>).
- 경우에 따라 feature extraction 용도로 사용하기도 하지만 t-SNE는 표본들 사이의 상대적인 거리를 유지하지 못하므로 주의해야 한다.

Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

10.1 Manifold approximation

- 점들 사이의 '거리' 혹은 유사도는 정규분포의 밀도함수로 계산한다.

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}$$

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

- Bandwidth를 결정하는 σ_i^2 는 확률계산에 고려하는 관찰값들의 분산으로 해당 표본들의 비중을 결정하며, 효율적인 계산을 위해 perplexity로 계산에 사용할 σ_i^2 를 결정한다.

$$\text{perplexity} = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}$$

$$-\sum_j p_{j|i} \log_2 p_{j|i} = \log_2 \text{perplexity}$$

- 모든 조합에 대해 Euclidean distance를 계산하므로 계산량이 많다.

Contents

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

10.2 Low dimensional embedding

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k=1}^K \sum_{k \neq i} (1 + \|y_k - y_i\|^2)^{-1}}$$

$$y_i = \arg \max_{y_i} \text{KL}(P||Q) = \arg \max_{y_i} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- 이 loss function은 두 점사이의 거리가 멀때, 혹은 p 가 0에 가까울 때 low-dimensional space에 제대로된 제약을 가지지 못한다.
- p 와 q 의 크기는 다음과 같이 근사해 보자.

$$p_{ij} \approx \exp(-d_x^2), q_{ij} \approx \frac{1}{1 + d_y^2}$$

- KL-divergence의 극한은 $p = 0, 1$ 에서 0이므로 p_{ij} 가 0에 가까울 경우 (두 점이 멀 경우) penalty는 다음과 같이 결정된다.

$$p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \approx -p_{ij} \log q_{ij} p_{ij} \approx \exp(-d_x^2), q_{ij} \approx \frac{1}{1 + d_y^2}$$

q_{ij} 의 영향을 거의 받지 않는다.

- 따라서 거리가 먼 두 점은 축소된 차원에서 멀리 위치할 수도 있지만 가깝게 위치할 수도 있게 된다.

10.3 training

- 학습에는 gradient descent를 사용하여 초기 배열에서 각 점들의 위치를 조정해 나간다.

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} \frac{(p_{ij} - q_{ij})(y_i - y_j)}{1 + \|y_i - y_j\|^2}$$

Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

```
sklearn.manifold.TSNE(n_components=2, *, perplexity=30.0, early_exaggeration=12.0, learning_rate=200.0, n_iter=1000, n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', init='random', verbose=0, random_state=None, method='barnes_hut', angle=0.5, n_jobs=None, square_distances='legacy')
```

- n_components: int, default=2
- perplexity: float, default=30.0

The perplexity is related to the number of nearest neighbors that is used in other manifold learning algorithms. Larger datasets usually require a larger perplexity. Consider selecting a value between 5 and 50. Different values can result in significantly different results.

- learning_rate: float, default=200.0

The learning rate for t-SNE is usually in the range [10.0, 1000.0]. If the learning rate is too high, the data may look like a 'ball' with any point approximately equidistant from its nearest neighbours. If the learning rate is too low, most points may look compressed in a dense cloud with few outliers. If the cost function gets stuck in a bad local minimum increasing the learning rate may help.

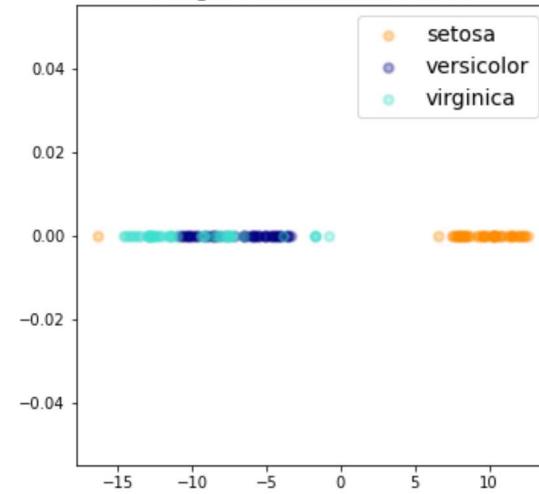
Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03008>)
 - 11.1 Approximate high dimensional
geometry
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

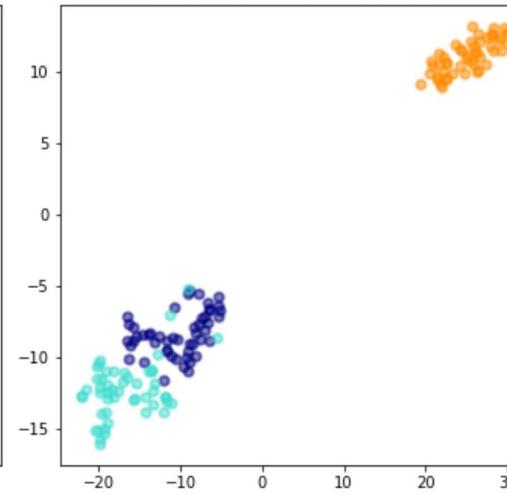
[Toggle show/hide](#)

Dimensionality Reduction with t-SNE

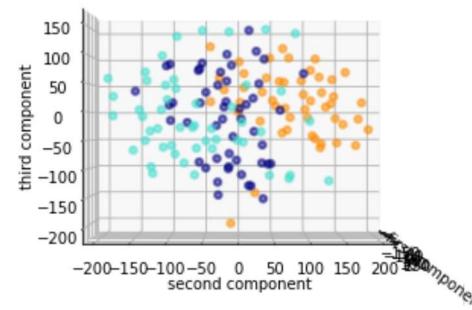
Single Dimensional t-SNE



Two Dimensional t-SNE



Three Dimensional t-SNE



PCA run time

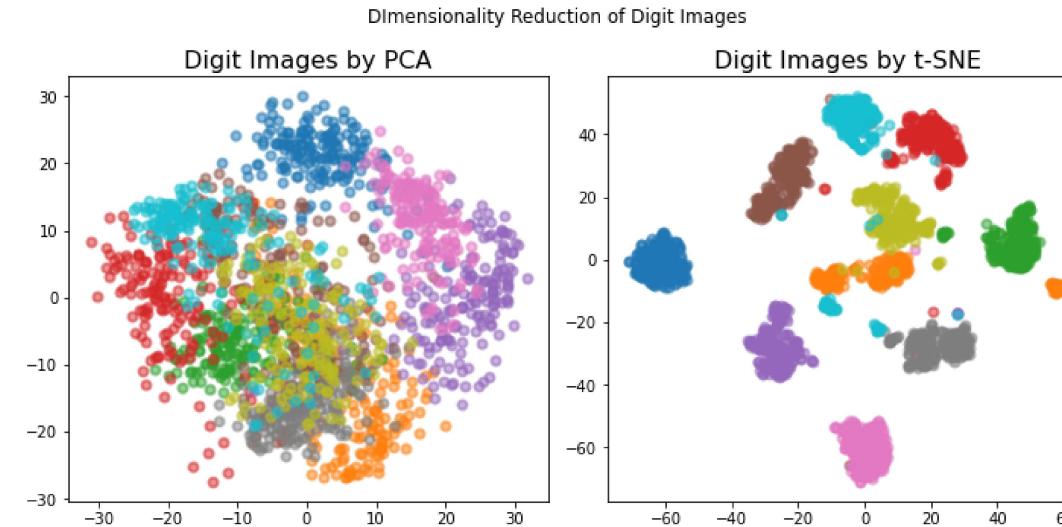
CPU times: total: 0 ns
Wall time: 8.5 ms

T-SNE run time

Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03008>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

CPU times: total: 1min 18s
Wall time: 8.97 s

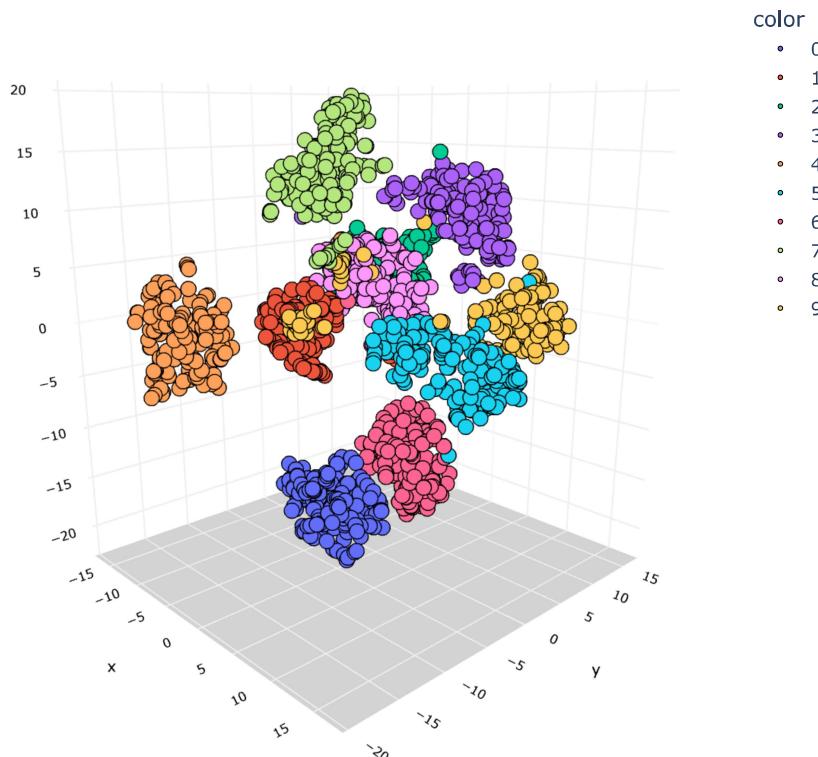


10.4 TSNE in 3D

CPU times: total: 2min 18s
Wall time: 15.4 s

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - ▼ 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity



Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
- ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
- 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03898>)
 - ▼ 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

10.5 t-SNE의 가정과 한계

- 가정
 - 표본은 특정 분포를 따른다.
- 특징
 - 유사한 특징을 갖는 표본은 가능한 가깝게 만드는 방식으로 하나의 표본과 다른 특성을 갖는 표본들은 차이를 반영하지 않는다. OVR와 유사한 방식으로 적용한다.
 - 실제 적용에선 시각화 목적이 대부분이므로 2차원 혹은 3차원으로 축소하는 것이 일반적이지만 이론적인 제한은 없다.
- 한계
 - Sample size나 perplexity가 커지면서 자원 소비가 급격히 증가한다. $\mathcal{O}(n^2)$
 - 동일한 cluster에 속한 표본의 거리는 해석이 가능하지만, cluster 사이의 유사정도를 측정하는데는 문제가 있다. 즉, cluster를 만드는 feature의 특성을 제대로 반영하지 못한다. (A와 B가 멀고 A와 C가 가깝다는 것이 A와 C가 더 유사하다는 것을 의미하진 않는다.) t-SNE는 조금만 멀면 거리가 다 비슷해진다.
 - σ 가 작으면 일정 수준 이상 떨어진 대부분의 점들은 확률이 0에 가깝다. 반면 값이 커지면 극한에선 모든 점에서 값이 같아진다. Sparse data에는 적용을 하지 않는다.
 - Feature수가 많을 경우 직접 적용이 어렵다. Autoencoder나 PCA 등을 이용해 차원을 축소한 후 사용한다.
 - Loss function이 convex하지 않으므로 초기값에 따라 결과가 달라진다.
 - 자료의 전역적인 구조를 잘 반영하지 못한다.
- t-SNE 사용상 주의할 점을 잘 정리한 [참고 사이트](https://distill.pub/2016/misread-tsne/) (<https://distill.pub/2016/misread-tsne/>)

It is highly recommended to use another dimensionality reduction method (e.g. PCA for dense data or TruncatedSVD for sparse data) to reduce the number of dimensions to a reasonable amount (e.g. 50) if the number of features is very high. This will suppress some noise and speed up the computation of pairwise distances between samples. [TSNE documentation](https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html#sklearn-manifold-tsne) (<https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html#sklearn-manifold-tsne>)

<https://towardsdatascience.com/umap-dimensionality-reduction-an-incredibly-robust-machine-learning-algorithm-b5acb01de568> (<https://towardsdatascience.com/umap-dimensionality-reduction-an-incredibly-robust-machine-learning-algorithm-b5acb01de568>)

Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03426>) (Uniform Manifold Approximation and Projection)
 - 11.1 Approximate high dimensional data space
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

11 UMAP (<https://arxiv.org/abs/1802.03426>) (Uniform Manifold Approximation and Projection)

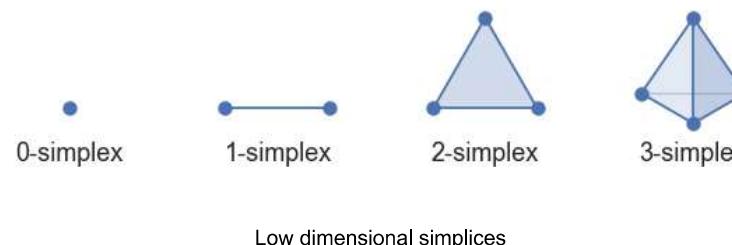
- UMAP의 전체적인 구조는 t-SNE와 굉장히 유사하지만 계산속도가 더 빠르고 전역적인 구조를 상대적으로 더 잘 반영한다는 장점이 있다.
- UMAP documentation 설명 (https://umap-learn.readthedocs.io/en/latest/how_umap_works.html)을 간단히 요약하고, 이후 설명에선 t-SNE과 비교하여 설명한다.

Assuming uniformly distributed data set on a manifold, project the approximation to a lower-dimensional space.

11.1 Approximate high dimensional data space

- Manifold learning 기반모형으로 다차원공간 점의 배열을 구조화하기 위해 simplex를 이용한다.

11.1.1 Simplices



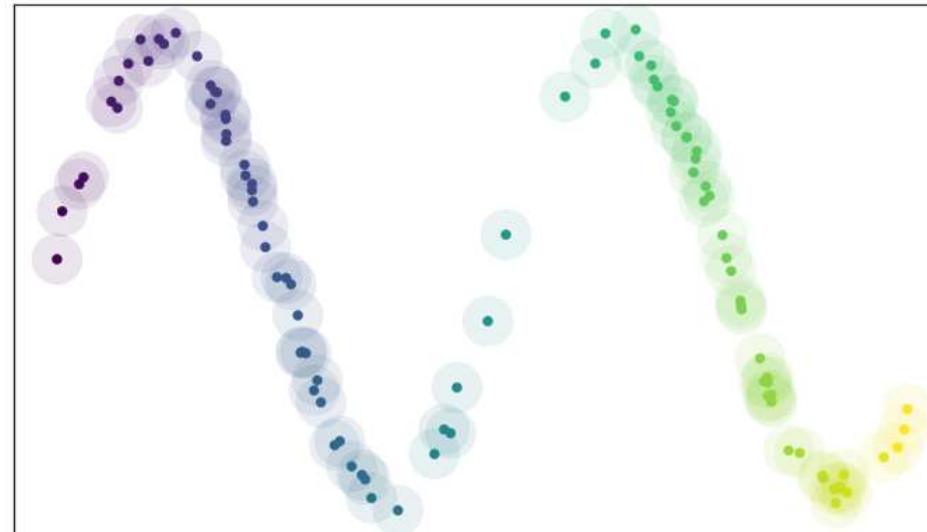
- Simplex는 k 차원 점들을 볼록결합한 것으로 생각해도 무방하며 기하학에서 볼 수 있는 개념이다.
 - 0-simplex: vertex, vertices 꼭지점, 0-faces
 - 1-simplex: edges, 1 faces, ...
- 반대로 (affinely independent한 점들로 구성한) k-simplex 차원의 점을 묘사하기 위해선 k 차원이 필요하다.
- 실제 훈련에선 1-simplex를 기초로 거리를 계산한다.

Contents ⚙

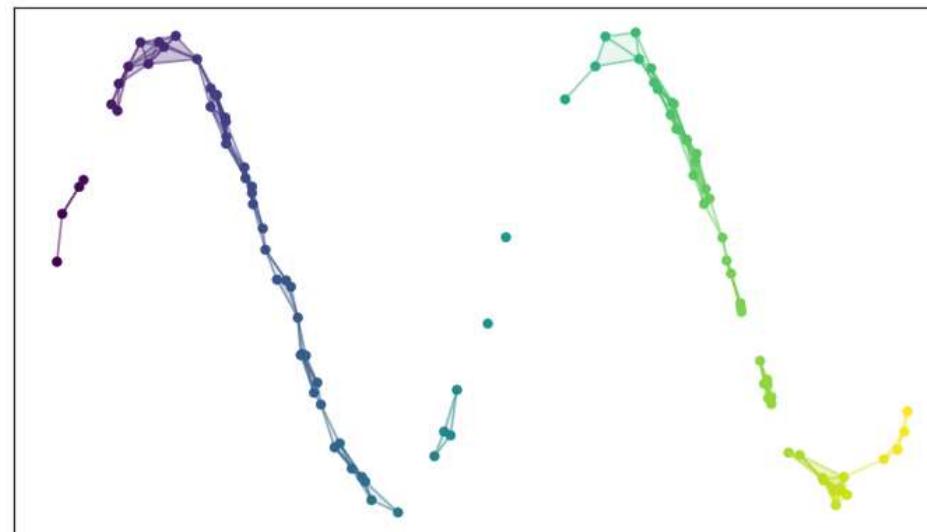
- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

11.1.2 Simplicial complex

- 각 점에서 open ball을 그려 open ball이 접하는 점들을 edge로 연결하면 점들의 위치와 open ball의 반경에 따라 k값이 결정된다.
[원리 \(https://pair-code.github.io/understanding-umap/supplement.html\)](https://pair-code.github.io/understanding-umap/supplement.html)
- k차원의 점으로 만든 k-simplex를 이어붙여 simplicial complex를 만들어 공간을 구조화한다.
- 예를들어 isolated point는 0-simplex, 가까운 두 점은 1-simplex로 구분한다.



A basic open cover of the test data



Contents ⚙ *

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - ▼ 11.1 Approximate high dimensional
topology
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

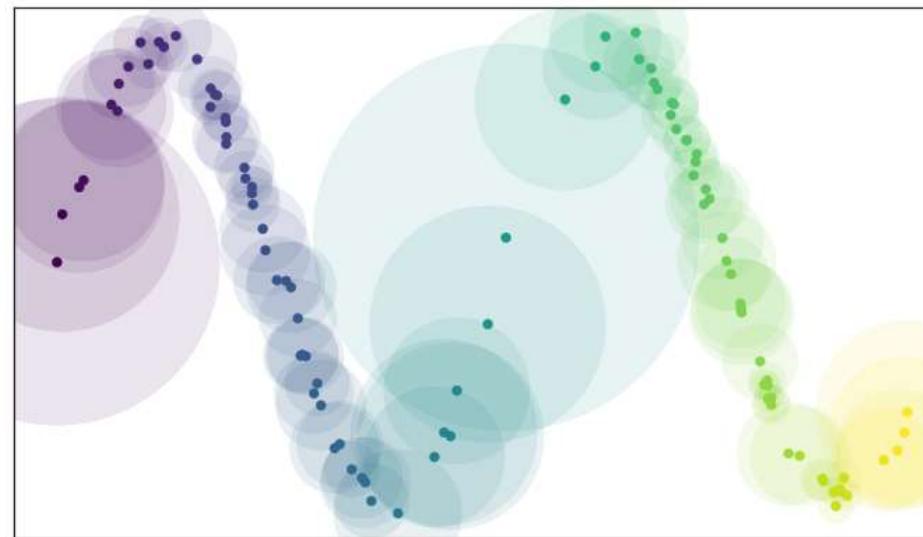
A basic open cover of the test data

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - ▼ 11.1 Approximate high dimensional structures
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

11.1.3 Varying distance 과 fuzzy connection

- 각 점들의 상대적인 위치는 가까운 k 개의 점을 이용하여 계산한다.
- 이때 모든 자료는 uniformly distributed 라고 가정하고 Riemannian geometry를 이용한 거리를 적용하는데, 실제 k 개의 점을 포함하는 원의 지름이 모두 다르므로 각 점마다 다른 거리 단위를 적용하여 unit circle 안에 '확률적으로' k 개의 점이 위치하도록 한다.
- 실제 자료의 분포는 균일하지 않으므로 국지적정보를 거리(radius)로 제한하면 표본의 밀도가 낮은 부분에선 포함되는 점이 아예 없을 수가 있다. 반면 표본의 밀도가 높은 곳에선 지나치게 많은 표본이 포함될 수 있다.
- 더구나 dimension이 높으면 점들 사이의 평균거리는 멀어지고 거리는 비슷해지는 경향이 있어, curse of dimensionality 가 문제를 더 악화시키게 되며 거리에 따라 점들을 차별화하기가 어렵다.
- 이 문제는 각 점마다 다른 거리기준을 적용하여 해결한다.



Open balls of radius one with a locally varying metric

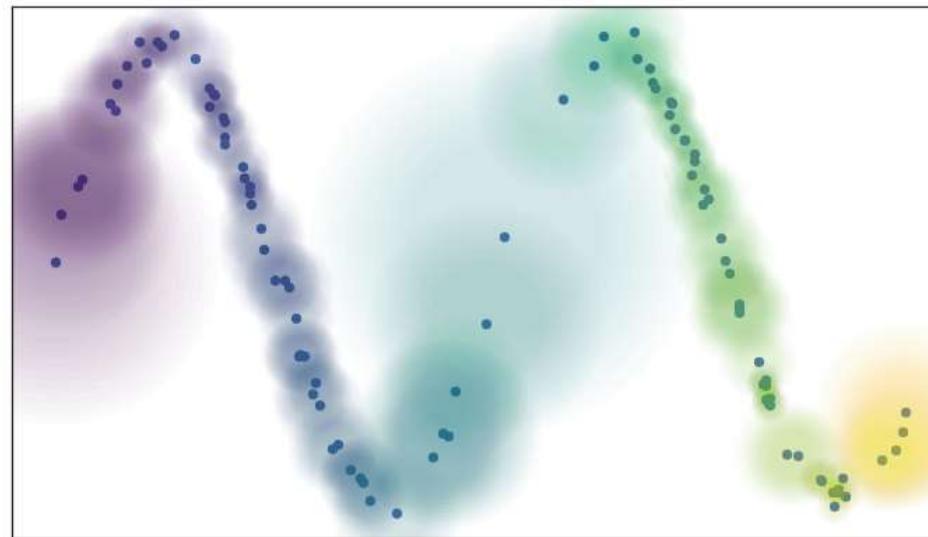
- 거리 대신 표본수를 사용하는 이유
 1. implementation이 쉽고 자료에 덜 의존한다.
 2. 국지성과 관련한 모형 tuning이 직관적이다.
 3. 각 점마다 다른 metric을 사용하므로 open set의 원소에 fuzzy value를 적용하기 쉽다.

Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
geometry
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

11.1.4 fuzzy simplicial complex

- 한 점을 중심으로한 거리로 점들의 상대적인 위치를 결정할 때 주변 점들의 거리가 멀수록 영향을 작게 만들 수 있다.
- 가까운 k 번째 점까지의 거리를 반지름으로 하고 해당 점들과의 거리를 얼마나 반영할지는 거리에 반비례하는 connection probability를 이용하여 확률적으로 결정한다.



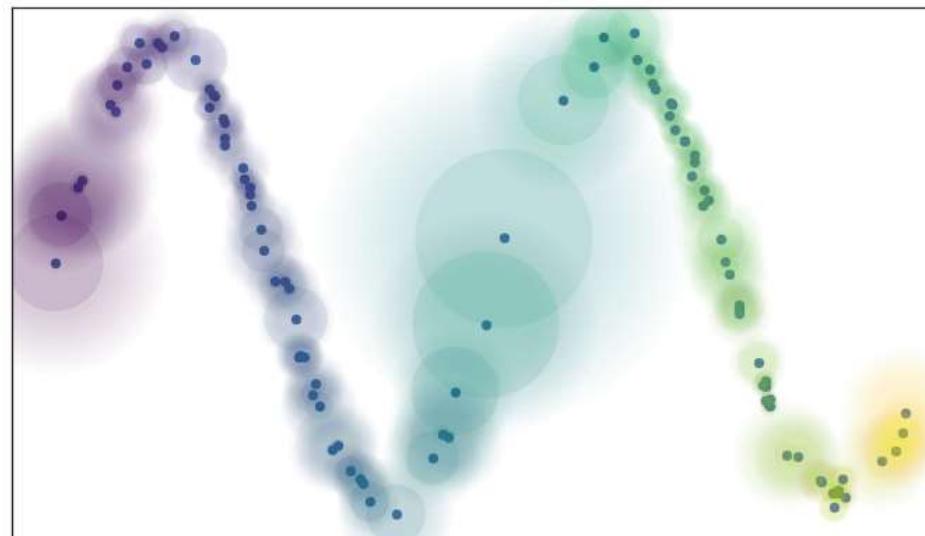
Fuzzy open balls of radius one with a locally varying metric

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - ▼ 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

11.1.5 Local connectivity

- Finite manifold approximation에서 완전히 분리된 점이나 cluster가 존재하면 전체 구조에서 상대적인 위치를 결정하기 곤란하므로 (거리의 척도가 모두 다르므로 어떤 거리가 사용되었는지 알 수가 없다) 모든 점은 하나 이상의 점과 연결되어 있어야 한다. Fuzzy confidence는 모든 점이 가장 가까운 점과는 연결되어 있고, 더 멀리 있는 점들은 거리에 따라 값이 작아지도록 한다.



Local connectivity and fuzzy open sets

Contents

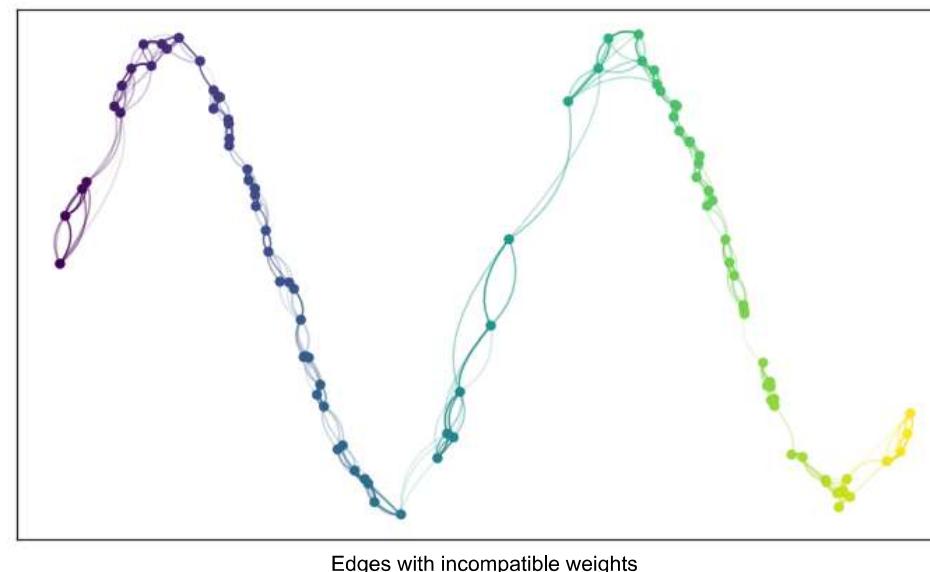
- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

11.1.6 similarity

- 표본 i 를 기준으로 한 j 와의 거리로 similarity를 측정하여 다음과 같이 계산한다. t-SNE의 정규분포의 밀도함수와 비슷한 형태이지만 정규화는 하지 않는다.

$$p_{j|i} = \exp\left(-\frac{d(x_i, x_j) - \rho_i}{\sigma_i}\right)$$

- $d(\cdot)$ 은 다양한 선택이 있다. [metric \(<https://umap-learn.readthedocs.io/en/latest/parameters.html#metric>\)](https://umap-learn.readthedocs.io/en/latest/parameters.html#metric)
- ρ_i 는 표본 i 에서 가장 가까운 표본과의 거리이다. 따라서 거리의 기준은 표본의 국지적인 분포에 영향을 받으며, 모든 관찰값에서 달라질 수 있다. 따라서 실제 거리와는 관계없이 제일 가까운 표본과의 similarity value는 항상 0이고, local connectivity를 만족한다. 두 점이 서로 제일 가깝다면 $p_{i|j} = p_{j|i}$ 가 된다.
- ρ 의 값은 sparse한 영역에선 크고 dense한 영역에선 작아진다. Outlier의 특징 역시 잘 잡아내며, sparsity를 발생시키는 high dimensional에서도 전처리 없이 만족스러운 성과를 보인다.
- σ_i 는 위치결정에 고려하는 주변 점들의 수인 k 값에 의해 결정되며, 더 많은 점을 고려할수록 similarity value를 크게 할 수 있어 전역적인 관계를 살펴보는데 유리하다.



Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

11.1.7 scope

- Perplex에 대응하는 $n_{neighbor}$ 로 고려할 주위 점의 갯수를 이용하여 가중값 σ_i 를 결정한다.
- $n_{neighbor}$ 는 UMAP에서 가장 중요한 parameter 중 하나로 similarity value의 합이 밑이 2인 $n_{neighbor}$ 의 로그값과 같아지도록 σ 를 조정하여 최종 similarity를 구한다.

$$n_{neighbor} = 2^{\sum_i p_{ij}}$$

$$\sum_i p_{ij} = \log_2(n_{neighbor})$$

- $n_{neighbor}$ 가 작을수록 거리에 따른 similarity가 급하게 줄어 local structure에 집중하고, 클수록 global structure를 반영하기 쉬워진다.

Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

11.1.8 Symmetrization

- Symmetrization는 probabilistic fuzzy union 이라고 부르는 연산을 사용하여 t-SNE와 다른 방식으로 적용한다.

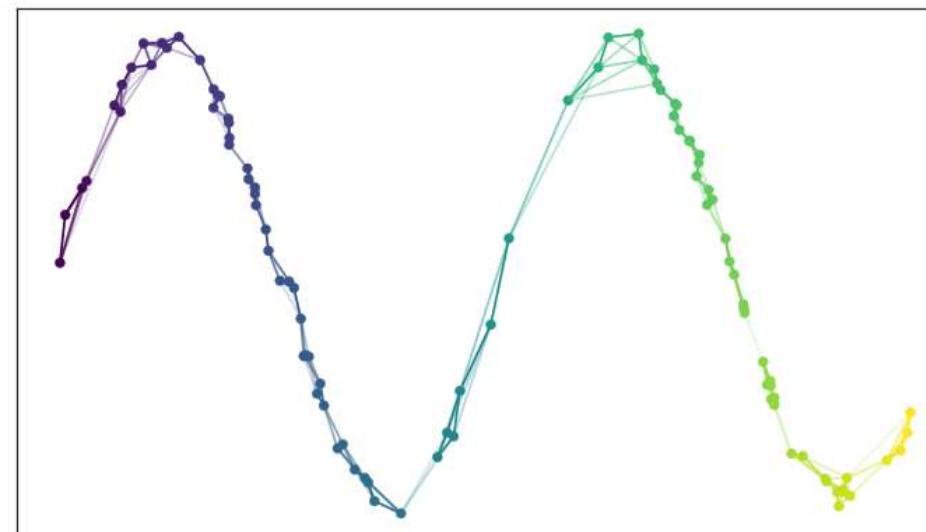
$$P_{ij} = P_{j|i} + P_{i|j} - P_{j|i}P_{i|j}$$

- Similarity value가 다음과 같은 세 점 a, b, c 를 생각해 보다.

$$P_{b|a} = P_{a|b} = 1.0, \quad P_{c|b} = 0.6, \quad P_{b|c} = 1.0$$

- c 에선 b 가 가장 가깝지만 a 와 b 가 가장 서로 가까운 경우이다.
- 대칭적으로 바꾸어보면 b 는 a 에 더 가깝지만 similarity value는 a 와 c 에 대해 같다.

$$P_{bc} = P_{c|b} + P_{b|c} - P_{c|b}P_{b|c} = (0.6 + 1.0) - (0.6 \times 1.0) = 1.0$$



Graph with combined edge weights

Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

11.2 Finding a Low Dimensional Representation

11.2.1 low dimensional embedding

- Low-dimensional embedding 공간은 통상적인 Euclidean space이다.
- 이때 낮은 차원의 공간에 manifold에서와 같이 varying metric을 사용하면 정상적인 배치가 어려워진다.
- Manifold에서 metric의 기준으로 가장 가까운 점과의 거리를 사용한 것과 같이 이에 대응하는 min_dist로 각 점 사이의 최소한의 거리 지정한다.
- min_dist의 값이 작을수록 clustering 효과가 크며 값이 클수록 전체적인 형태를 이해하는데 도움이 된다.
- 점 사이의 거리는 다음과 같이 결정한다. student-t 분포와 유사한 약간 일반적인 형태이다.

$$q_{ij} = \frac{1}{1 + a|y_i - y_j|^{2b}}, \quad a \approx 1.93, b \approx 0.79$$

- $a = b = 1$ 이면 low dimensional score는 정규화하지 않은 t-SNE의 low-dimension distribution과 같다.

11.2.2 Optimization

- Manifold와 embedding space는 동일한 0-simplices를 공유하고 있으므로 모든 1-simplices의 similarity value를 각 공간에서 해당 1-simplices가 존재할 확률로 해석할 수 있다.
- 이 확률이 Bernoulli distribution을 따른다고 하면 두 similarity vector의 비교는 cross entropy를 적용하는 것이 자연스럽다.
- 모든 가능한 1-simplices의 집합을 E 라고 하면, cross entropy loss는 다음과 같다.

$$\text{CE}(\mathbf{w}_h, \mathbf{w}_f) = \sum_{e \in E} \left[\mathbf{w}_h(e) \log \frac{\mathbf{w}_h(e)}{\mathbf{w}_f(e)} + (1 - \mathbf{w}_h(e)) \log \frac{1 - \mathbf{w}_h(e)}{1 - \mathbf{w}_f(e)} \right]$$

- 첫째 항은 $\mathbf{w}_h(e)$ 가 클 때 attractive force로 작용하며 loss를 작게하려면 $\mathbf{w}_f(e)$ 역시 커져야 한다.
- 반대로 $\mathbf{w}_h(e)$ 가 작을 때는 둘째 항이 repulsive force로 작용하여 $\mathbf{w}_f(e)$ 역시 작아져야 한다.
- t-SNE와 다르게 local connectivity와 global connectivity 사이에 적절한 균형을 이루면서 global data structure를 반영한다.
- 최적화 과정에서 stochastic gradient descent를 사용하므로 훈련마다 완전히 동일한 결과를 얻을 순 없지만 차이는 그리 크지 않다.
- t-SNE는 초기분포를 무작위로 결정하기 때문에 결과가 상당히 달라질 수 있지만 UMAP은 spectral embedding을 사용하므로 어느정도 일정한 결과를 얻을 수 있다.

Contents ⚙

1 The curse of dimensionality
▼ 2 Dimensionality reduction
2.1 Projection
2.2 Manifold learning
3 Unsupervised learning
▼ 4 principal component analysis (dime)
4.1 Diagonalizing the covariance
4.2 Formulation
▼ 4.3 Orthonormal coordinate system
4.3.1 Diagonalizable matrix
4.3.2 Principal components
▼ 4.4 Principal component의 해석
4.4.1 First principal component
4.4.2 Second principal component
▼ 4.5 Sklearn implementation
4.5.1 n_component
4.5.2 whiten
4.5.3 explained_varianceratio
▼ 5 Kernel PCA
5.1 Eigenvector
5.2 Kernel matrix
▼ 5.3 Normalization
5.3.1 centered features
5.3.2 orthonormal basis
5.4 Dimension reduction
▼ 5.5 Memory
5.5.1 Incremental PCA
5.5.2 Randomized PCA
5.6 PCA의 가정과 한계
6 Manifold learning algorithm
7 Multi-Dimensional Scaling (MDS)
8 Locally Linear Embedding (LLE)
9 ISOMAP
▼ 10 T-SNE, t-distributed stochastic nei
10.1 Manifold approximation
10.2 Low dimensional embedding
10.3 training
10.4 TSNE in 3D
10.5 t-SNE의 가정과 한계
▼ 11 UMAP (https://arxiv.org/abs/1802.0)
▼ 11.1 Approximate high dimensional
11.1.1 Simplices
11.1.2 Simplicial complex
11.1.3 Varying distance 과 fuzzy c
11.1.4 fuzzy simplicial complex
11.1.5 Local connectivity
11.1.6 similarity

11.3 Implementation

- UMAP은 label 정보를 이용하여 구현할 수도 있다.

```
!conda install -c conda-forge umap-learn
```

```
umap = UMAP(n_neighbors=15, # The size of local neighborhood used for manifold approximation.  
            n_components=2, # The dimension of the space to embed into.  
            metric='euclidean',  
            n_epochs=None, # larger values result in more accurate embeddings.  
            learning_rate=1.0, # The initial learning rate  
            min_dist=0.1, # The effective minimum distance between embedded points.  
            local_connectivity=1, # The local connectivity required  
            random_state=None  
)
```

- n_neighbors에 따라 국지적인 특성의 반영 정도가 결정된다.
- min_dist embeded space에서 표본들 사이의 최소한의 거리
- local_connectivity 국지적으로 고려해야 할 최소한의 관찰값 수

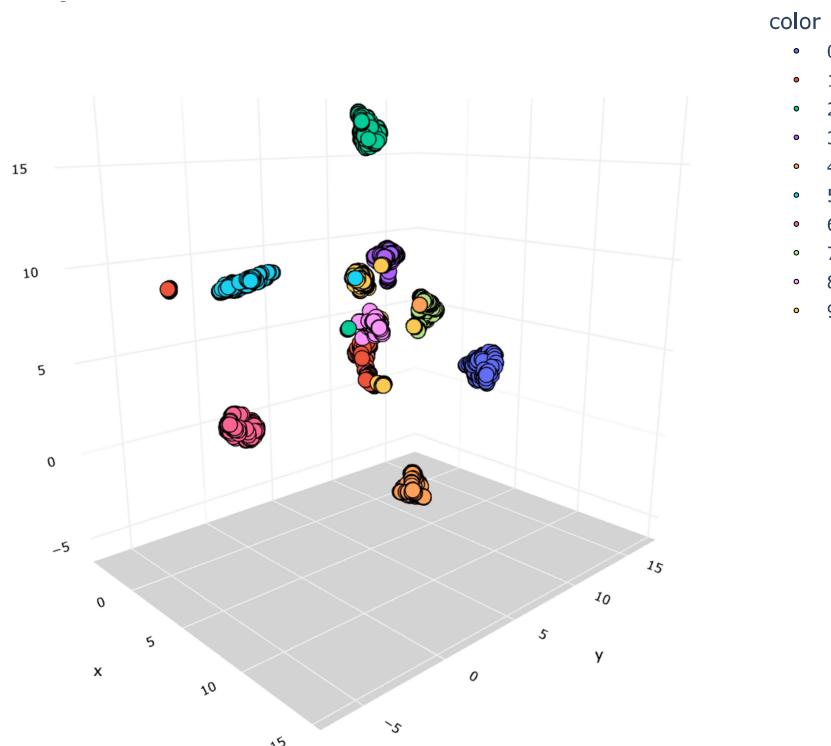
참고 블로그 (<https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668>)

11.3.1 Unsupervised

```
CPU times: total: 2.25 s  
Wall time: 11.7 s
```

Contents ⚙

- 1 The curse of dimensionality
- 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

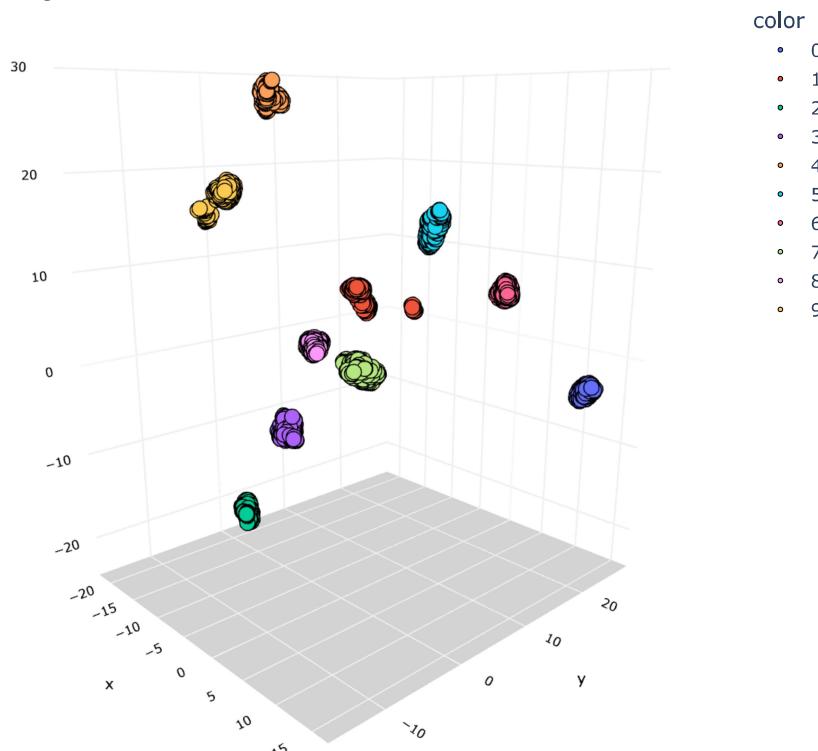


11.3.2 supervised

CPU times: total: 2.64 s
Wall time: 20.1 s

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_variance_ratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic neighbor embedding
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03449>)
 - 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy complex
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

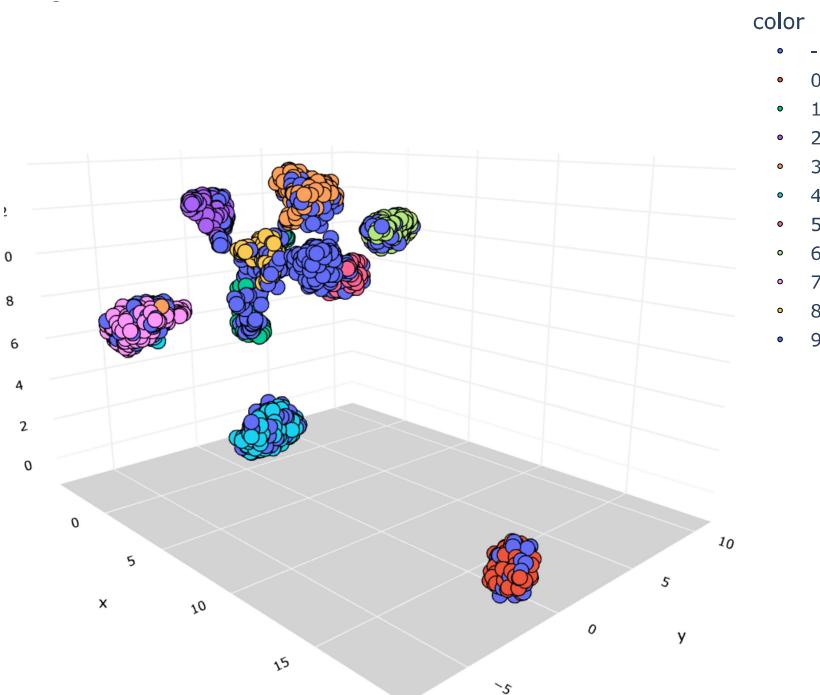


11.3.3 Semi-supervised UMAP

CPU times: total: 2.33 s
Wall time: 18.8 s

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 t-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - ▼ 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity



11.4 UMAP의 가정과 한계

11.4.1 UMAP의 장점

- 비선형 패턴을 학습할 수 있다.
- PCA 보다 cluster 들을 곧잘 분리해 낸다.
- 예측에 사용할 수도 있다.
- 정규화가 필요없으며, $n_{neighbors}$ 계산에 entropy 대신 단순한 지수함수를 사용하여 계산이 간단하다.
- 고차원 자료나 sparse 자료에 직접 적용할 수 있다.
- 국지적인 거리와 전역적인 거리를 모두 반영한다.

11.4.2 단점

- 거리는 원래 자료의 거리와 직접 비교하기 어렵다.
- 범주형 자료에 직접 적용하지 못한다.
- t-SNE나 UMAP 모두 초기값이 큰 영향을 미치며 global structure 과 관련된 특징은 제한적이다. [Kobak and Linderman, 2019](https://www.researchgate.net/publication/338071992_UMAP_does_not_preserve_global_structure_any_better_than_t-SNE_when_using_the_same_INITIALIZATION) (https://www.researchgate.net/publication/338071992_UMAP_does_not_preserve_global_structure_any_better_than_t-SNE_when_using_the_same_INITIALIZATION)

Contents ⚙

- 1 The curse of dimensionality
- ▼ 2 Dimensionality reduction
 - 2.1 Projection
 - 2.2 Manifold learning
- 3 Unsupervised learning
- ▼ 4 principal component analysis (dime)
 - 4.1 Diagonalizing the covariance
 - 4.2 Formulation
 - ▼ 4.3 Orthonormal coordinate system
 - 4.3.1 Diagonalizable matrix
 - 4.3.2 Principal components
 - ▼ 4.4 Principal component의 해석
 - 4.4.1 First principal component
 - 4.4.2 Second principal component
 - ▼ 4.5 Sklearn implementation
 - 4.5.1 n_component
 - 4.5.2 whiten
 - 4.5.3 explained_varianceratio
- ▼ 5 Kernel PCA
 - 5.1 Eigenvector
 - 5.2 Kernel matrix
 - ▼ 5.3 Normalization
 - 5.3.1 centered features
 - 5.3.2 orthonormal basis
 - 5.4 Dimension reduction
 - ▼ 5.5 Memory
 - 5.5.1 Incremental PCA
 - 5.5.2 Randomized PCA
 - 5.6 PCA의 가정과 한계
- 6 Manifold learning algorithm
- 7 Multi-Dimensional Scaling (MDS)
- 8 Locally Linear Embedding (LLE)
- 9 ISOMAP
- ▼ 10 T-SNE, t-distributed stochastic nei
 - 10.1 Manifold approximation
 - 10.2 Low dimensional embedding
 - 10.3 training
 - 10.4 TSNE in 3D
 - 10.5 t-SNE의 가정과 한계
- ▼ 11 UMAP (<https://arxiv.org/abs/1802.03404>)
 - ▼ 11.1 Approximate high dimensional
 - 11.1.1 Simplices
 - 11.1.2 Simplicial complex
 - 11.1.3 Varying distance 과 fuzzy c
 - 11.1.4 fuzzy simplicial complex
 - 11.1.5 Local connectivity
 - 11.1.6 similarity

<https://pair-code.github.io/understanding-umap/> (<https://pair-code.github.io/understanding-umap/>)