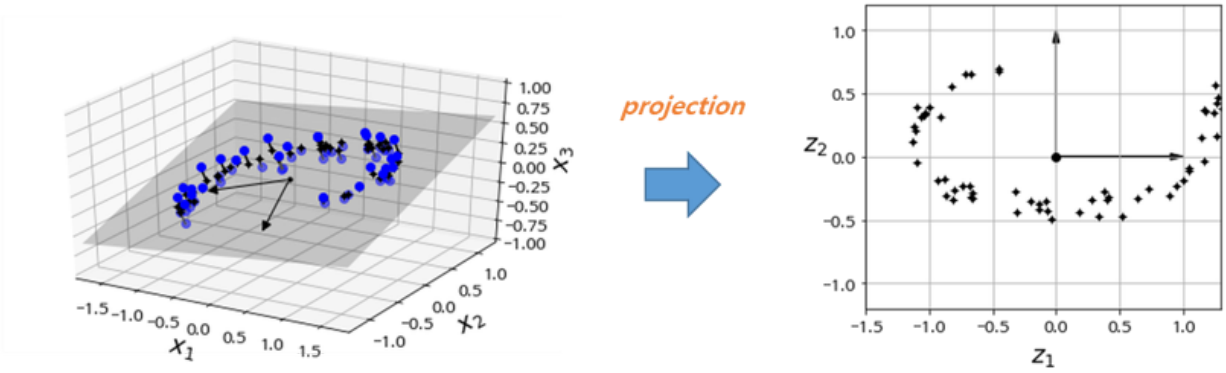
PCA, t-SNE, UMAP

PCA

PCA란?

- n 개의 관측치와 p개의 변수로 구성된 데이터를 상관 관계가 없는 k개의 변수로 구성된 데이터로 요약하는 방식. 이 때 요약된 변수는 기존 변수들의 선형 조합으로 생성된다.
- 원래 데이터의 분산을 최대한 보존하는 새로운 축을 찾고 그 축에 데이터를 projection 하는 기법.



https://img1.daumcdn.net/thumb/R1280x0/?scode=mtistory2&fname=https%3A%2F%2Ft1.daumcdn.net%2Fcfile%2Ftistory%2F99EABC3E5B8A48781B

위의 그림은 3차원으로 표현되는 데이터들을 2차원 공간으로 projection하여 2차원 data set으로 만든 것!

- 주요 목적

1. 데이터의 차원 축소 ($n \times p$ 행렬에서 $n \times k$ 행렬으로 요약. $k << p$
2. 데이터의 시각화 및 해석, 분석

즉 다음과 같은 행렬을 3개의 주성분만을 이용하여 차원 축소를 하게 되면 다음과 같이 데이터셋이 변하게 된다. 이 때 Z_1, Z_2, Z_3는 기존 변수인 $X_1, \ldots, X_p$의 선형 조합으로 새롭게 생성된 변수이다.

관측치/변수	X_1	$\ldots$	X_i	X_p	$\ldots$
N_1	X_{11}	$\ldots$	X_{1i}	X_{1p}	$\ldots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
N_3	X_{i1}	$\ldots$	X_{ii}	X_{ip}	$\ldots$

관측치/변수	X_1	$\cdots$	X_i	X_p	$\cdots$
N_n	X_{n1}	$\cdots$	X_{ni}	X_{np}	$\cdots$

관측치/변수	Z_1	Z_2	Z_3
N_1			
$\cdots$			
N_3			
$\cdots$			
N_n			

- 과정

X_1	X_2	X_3
0.2	5.6	3.56
0.45	5.89	2.4
0.33	6.37	1.95
0.54	7.9	1.32
0.77	0.78	0.98

1. Normalize $(E(X_i) = 0, Var(X_i) = 1)$

X_1	X_2	X_3
-1.193	-1.030	1.501
-0.037	-0.765	0.354
-0.592	-0.326	-0.091
0.379	1.074	-0.714
1.443	1.046	-1.050

2. Covariance(Correlation) matrix 구하기. 여기서는 정규화했으므로 correlation matrix 사용.

$Corr(X)$는 다음과 같다.

1	0.8417	-0.8840
0.8417	1	-0.9133
-0.8840	-0.9133	1

3. Correlation matrix에 대한 eigenvalue 및 eigenvector 찾기

$\lambda_1 = 0.0786, \lambda_2 = 0.1618, \lambda_3 = 2.7596$

$e_1^T = [0.2590 \quad 0.5502 \quad 0.7938]$

$e_2^T = [0.7798 \quad -0.6041 \quad 0.1643]$

$e_3^T = [0.5699 \quad 0.5765 \quad -0.5855]$

이 때 $\lambda_3 > \lambda_2 > \lambda_1$

4. eigenvector와 X를 내적하여 Z_1, Z_2, Z_3 만들기

$Z_1 = e_3^T X = 0.5699X_1 + 0.5765X_2 - 0.5855X_3 = \quad [-2.1527 \quad -0.6692 \quad -0.4718 \quad 1.2533 \quad 2.0404]$

$Z_2 = e_2^T X = [-0.0615 \quad 0.4912 \quad -0.2798 \quad -0.4703 \quad 0.3204] Z_3 = e_1^T X = [0.3160 \quad -0.1493 \quad +0.4047 \quad 0.1223 \quad 0.1157]$

Z_1	Z_2	Z_3
-2.1527	-0.0615	0.3160
-0.6692	0.4912	-0.1493
-0.4718	-0.2798	-0.4047
1.2533	-0.4703	0.1223
2.0404	0.3204	0.1157

이 때 세션에서 배웠듯이 Z_1, Z_2, Z_3는 서로 독립이고 Covariance matrix는 diagonal이며, diagonal element는 $\lambda_3, \lambda_2, \lambda_1$이다. 즉 $Cov(Z)$는

2.7596	0	0
0	0.1618	0
0	0	0.0786

이 때 첫번째 주성분은 total population cariance의 약 $\frac{\lambda_3}{\lambda_3+\lambda_2+\lambda_1} = 0.920$ 정도를 설명할 수 있다. 혹은 데이터를 두 개의 주성분을 이용하여 2차원으로 표현하고 싶을 때에는 다음과 같이 Z_1, Z_2만을 이용하여 데이터를 요약할 수 있다.

Z_1	Z_2
-2.1527	-0.0615
-0.6692	0.4912

Z_1	Z_2
-0.4718	-0.2798
1.2533	-0.4703
2.0404	0.3204

t-SNE

Van der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research, 9, 2579-2605.

- t-SNE(t-distributed Stochastic Neighbor Embedding)는 데이터의 차원 축소에 사용되는 머신 러닝 알고리즘 중 하나로, 2002년 샘 로이스와 제프리 힌튼에 의해 개발된 Stochastic Neighbor Embedding을 확장시킨 방법이다.
- 비선형적인 차원 축소 방법이며, 데이터의 시각화가 목적이다. 따라서 t-SNE는 주로 시각화(2, 3차원) 툴로 사용 된다.

Introduction

$\mathcal{X} = x_1, \ldots, x_n$는 high-dimensonsal data set

$\mathcal{Y} = y_1, \ldots, y_n$는 two or three-dimensional data set

차원 축소의 과정에서의 목표는 고차원 데이터의 구조를 가능한 한 많이 저차원 map에 보존하는 것이다. 고차원 데이터의 경우 매우 유사한 datapoint를 고차원에서뿐만 아니라 저차원에서도 그 관계를 유지해야 하는 것이 중요하다. 이는 PCA와 같은 linear technique으로는 불가능하다.

Stochastic Neighbor Embedding

t-SNE를 이해하기 위해선 먼저 SNE(Stochastic Neighbor Embedding) 방법에 대해 이해해야 한다.

우선 datapointe들의 high-dimensional Euclidean distances를 유사성을 나타내는 조건부 확률로 변환해야 한다. x_i가 x_i를 중심으로 하는 Gaussian distribution 하에서, x_j를 선택할 확률!

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

x_j에 대한 x_i의 유사성$p_{j|i}$는, 위와 같이 Gaussian distribution을 이용하여 나타낼 수 있다. 위의 식을 통해 x_i와 x_j가 가까울수록 $e^{-||x_i-x_j||^2}$은 커진다(더 많은 가중치를 준다)는 것을 알 수 있다. 이 때 σ_i는 x_i를 중심으로 하는 Gaussian distribution의 variance이다.

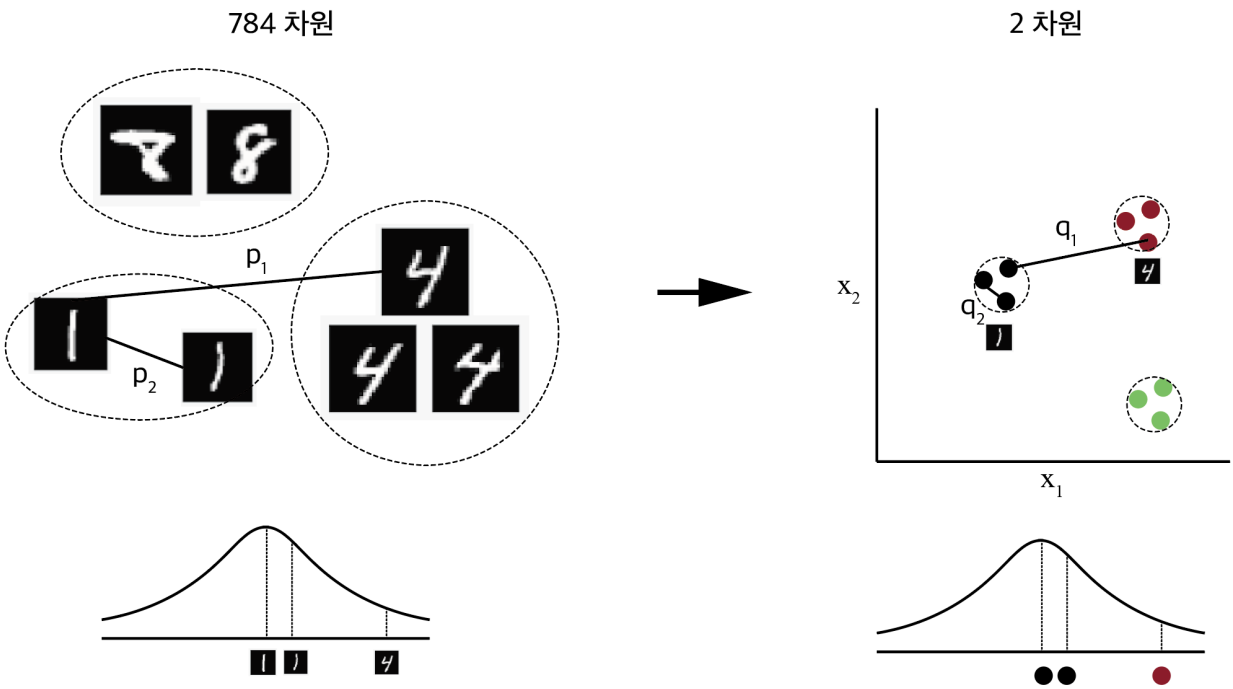
$$q_{j|i} = \frac{exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} exp(-\|y_i - y_k\|^2)}$$

마찬가지로 low-dimensional에서의 datapoint 또한 위와 같이 $q_{j|i}$로 나타낼 수 있다.

이 때 우리의 목표는 고차원(기존) 데이터와 저차원(축소된 차원에서의) 데이터의 유사성을 최소화하는 것이고, 이는 $p_{j|i}$와 $q_{j|i}$의 차이를 최소화해야 한다는 것을 의미한다. 따라서 다음과 같이 Cost 함수를 Kullback-Leibler divergence로 둔다면, Kullback-Leibler divergence를 최소화하는 것은 곧 $p_{j|i}$와 $q_{j|i}$의 차이를 최소화한다는 것을 의미하게 된다. $p_{j|i}$와 $q_{j|i}$의 차이가 작아질 수록 Cost값은 작아진다!

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

위와 같은 형태의 SNE의 cost function를 통해 local structure를 보존할 수 있다.



우리의 목표는 cost를 최소화, 즉 high dimensional에서의 $p_{j|i}$와 low-dimensional $q_{j|i}$의 차이를 최소화하는 것! 즉 그림에서 (p_1, q_1), (p_2, q_2)는 가능한 비슷해야 한다.

이 때 다시 $p_{j|i}$를 살펴보면, σ_i를 선택해야 한다. 이 때 optimal한 σ_i의 값이 하나만 있다고 보기 어려운데, 이는 데이터의 밀도가 다를 수 있기 때문이다. 데이터의 밀도가 높은 부분에서는 작은 σ_i 값이 더 적절할 것이다.

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

이 때 특정한 값의 σ_i는 다음과 같은 entropy를 생성하게 되고, 이를 통해 perplexity를 정의할 수 있게 된다. SNE는 perplexity를 고정하고 최적의 σ_i값에 대한 binary search를 진행하게 된다.

$$Perp(P_i) = 2^{H(P_i)}$$

where $H(P_i)$ is the Shannon etropy

$$H(P_i) = -\sum_j p_{j|i} log_2 p_{j|i}$$

즉 Perplexity는 $p_{j|i}$에서의 σ_i 값을 결정하기 위한 변수로서, 5-50 사이에서 Robust한 특성을 갖는다. Perplexity는 고려되는 이웃 node의 수를 결정하게 되고, 높은 perplexity의 값을 사용할 수록 더 많은 이웃 node를 고려하게 되고(perplexity와 σ_i는 monotone increase), 이는 데이터를 좀 더 전체적인 관점에서 고려한다는 것을 의미한다. 반면, 낮은 perplexity의 경우 더 적은 이웃 node를 고려하게 되고, 이는 좀 더 데이터의 지역적인(local) 구조를 고려한다는 것을 의미한다.

최종적으로 gradient descent method를 통해 low-dimensional data representation Υ을 찾게 된다.

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j).$$

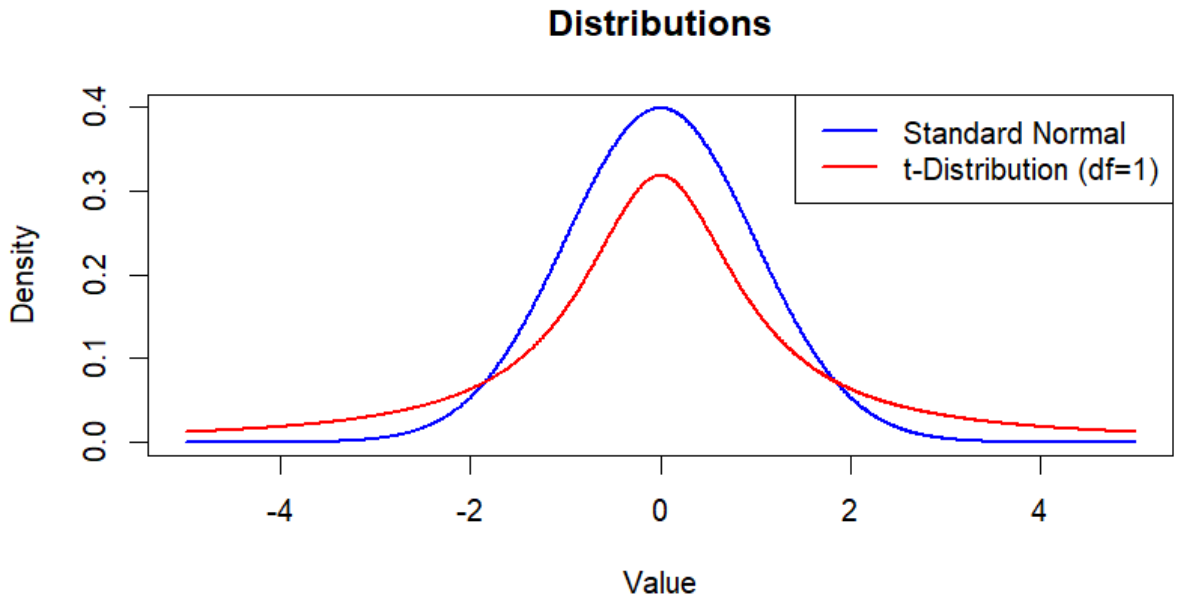
$$\Upsilon^{(t)} = \Upsilon^{(t-1)} + \eta \frac{\partial C}{\partial \Upsilon} + \alpha(t)(\Upsilon^{(t-1)} - \Upsilon^{(t-2)}).$$

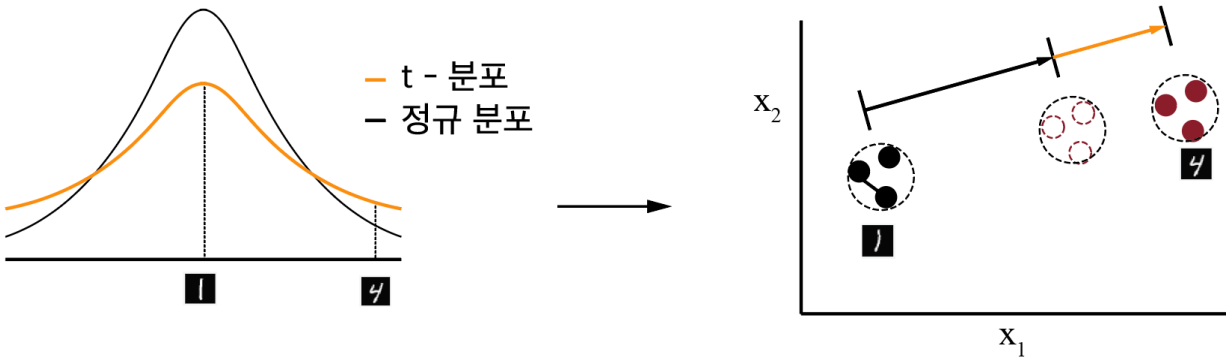
$\Upsilon^{(t)}$는 iteration t에서의 solution, η는 learning rate, $\alpha(t)$는 iteration t에서의 momentum을 의미한다. Numper of Iteration은 학습이 수행되는 횟수이며, Learning Time은 한 번의 학습에 변수가 변화되는 정도를 나타낸다. Momentum은 학습 과정에 Global Minimum이 아닌 Local Minimum에서 값이 결정되는 것을 방지하기 위해 필요한 변수로서, 0.5 - 0.8 사이의 값이 사용된다. 즉 알고리즘 구현을 위해 필수적으로 입력되어야 하는 변수는 number of iterations, perplexity, learning rate, momentum 세 가지 이다.

t-SNE

이 때 SNE가 전제하는 확률 분포는 정규 분포이지만, 정규 분포는 꼬리가 두텁지 않아서 i번째 개체에서 적당히 떨어져 있는 이웃 j와 아주 많이 떨어져 있는 이웃 k가 선택될 확률이 크게 차이가 나지 않게 된다. 또한 높은 차원 공간에서는 분포의 중심에서 먼 부분의 데이터 비중이 높기 때문에 데이터 일부분의 정보를 고차원에서 유지하기가 어렵다. 이러한 문제로 인하여 구분을 좀 더 잘하기 위해 정규 분포보다 꼬리가 두터운 t분포를 쓰게 되며 꼬리가 부분이 상대적으로 더 두텁게 degree of freedom = 1로 적용하여 사용한다. 즉 t분포의 경우, tail 부분에서 거리에 따른 확률 값의 변화가 정규분포에 비해서 크기 때문에 거리에 따른 데이터의 구분을 더 잘하게 되고, 이는 global한 structure를 반영할 수 있게 된다. 이 t 분포를 적용한 것이 t-SNE가 된다.

또한 기존 SNE에서는 cost function을 optimize하는 하는 문제는 상당히 difficult했는데, t-SNE에서는 cost function을 symmetric하게 만들어 조금 더 간단하게 계산이 가능하도록 하였다.





우선 $p_{j|i}$에 대한 보정을 해야 한다. 데이터에서 노드 i 에 대한 노드 j 의 거리 가중치와 노드 j 에 대한 노드 i 의 거리 가중치 사이에는 차이가 있다. 가령 i 의 기준에선 j 가 가장 가까운 거리에 있을지라도 j 의 기준에선 i 보다 더 가까운 노드가 존재 할 수 있기 때문이다. 따라서 두 노드 사이에 Symmetric한 확률 값을 갖게 하기 위해 다음과 같이 정의한다.

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

또한 자유도가 1인 t distribution의 kernel 부분을 이용하여 다음과 같이 q_{ji}를 정의할 수 있다.

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

마지막으로 cost function은 다음과 같이 정의되고, gradient는 다음과 같아진다.

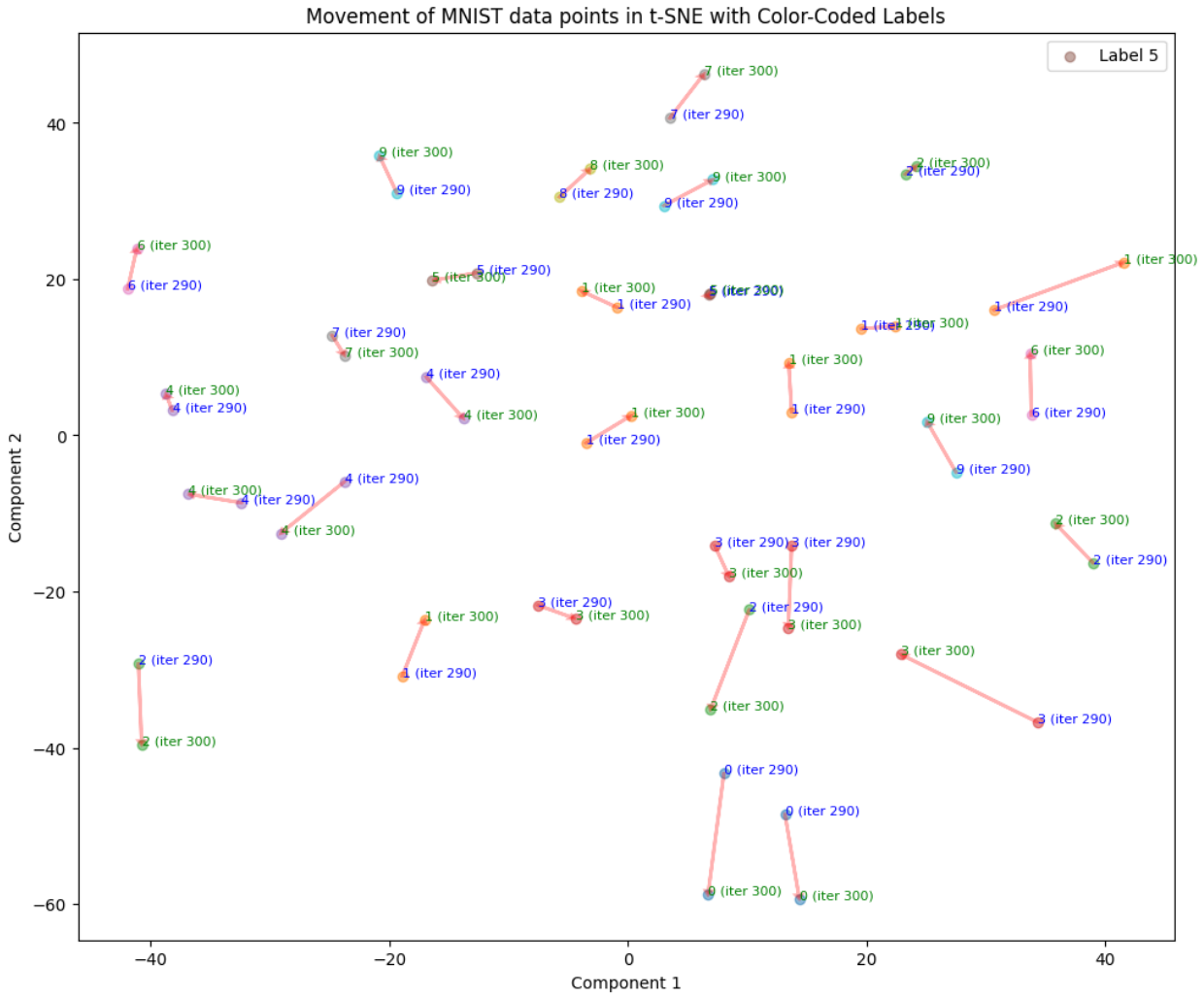
$$Cost = \ KL(P||Q) = \sum_i \sum_j p_{ji} log \frac{p_{ji}}{q_{ji}}$$

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ji} - q_{ji})(y_i - y_j).$$

$$\Upsilon^{(t)} = \Upsilon^{(t-1)} + \eta \frac{\partial C}{\partial \Upsilon} + \alpha(t)(\Upsilon^{(t-1)} - \Upsilon^{(t-2)}).$$

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

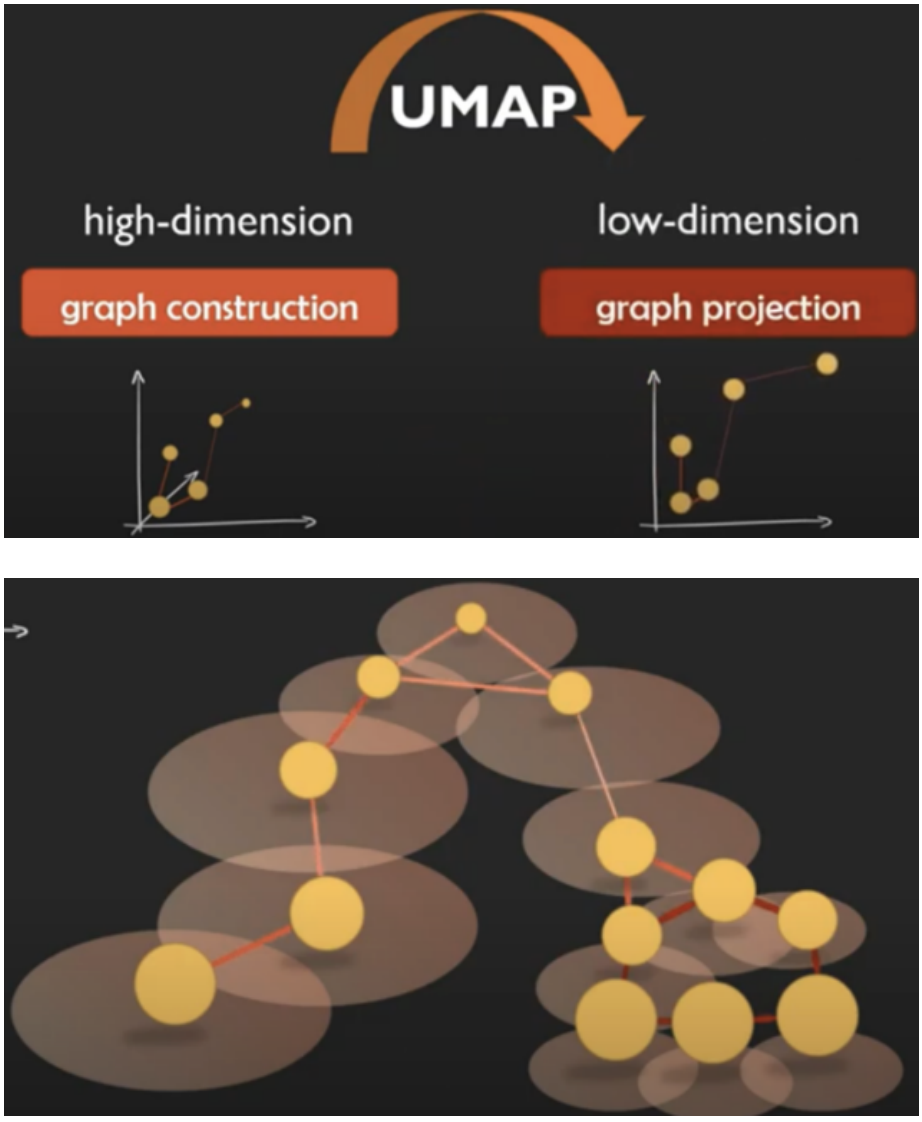
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=1 **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end



UMAP

https://data-newbie.tistory.com/295 참고. 이론적인 배경은 리만 기하학과 위상 수학에 기반하여 상당히 어려움 ⇒ 주요 내용만 정리.

고차원에서의 데이터를 graph로 만들고, 저차원으로 graph projection



https://www.youtube.com/watch?v=6BPl81wGGP8 캡쳐본

1. data point에서 simplex 복합체로 만들어서 graph를 구성 (proven by nerve theorem)

2. 각 node에서 길이 k의 radius를 그린다. (knn 등을 이용, 정해진 nearest neighbor n개만큼을 포함하는 radius k)
3. k가 작으면 local structure, k가 크면 global structure을 좀 더 반영할 수 있다.
4. radius가 겹치는 정도로 connection의 wiehgt를 결정하고, 이러한 구조를 저차원으로 이동한다. (리만 매니폴드, 퍼지이론 등등 사용)

UMAP(Uniform Manifold Approximation and Prediction)은 고차원에서의 exponential probability distribution을 사용한다. 이 때

t-SNE와는 달리 확률로 normalize할 필요가 없다. 어떠한 거리든 다 연결할 수 있다. ρ는 각각의 데이터 x_i로부터 첫번째 이웃까지의 거리를 표현한다. 이는 각 포인트들은 최소한 가장 가까운 점들과는 반드시 연결되어야 한다는 의미이고, 이를 통해 local structure을 보존하면서, global structure 또한 보존할 수 있도록 하였다. 따라서 t-SNE보다 global structure를 더 잘 보존할 수 있다.

$$p_{ij} = exp(-\frac{d(x_i, x_j) - \rho_i}{\sigma_i})$$

UMAP은 t-SNE에서의 Perplexity 대신 the number of nearest neighbors을 사용한다.

$$k = 2^{\sum_i p_{ij}}$$

UMAP에서는 p_{ij}를 다음과 같이 정의한다.

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

UMAP에서는 q_{ij}를 다음과 같이 정의한다.

$$q_{ij} = \left(1 + a(y_i - y_j)^{2b}\right)^{-1}$$

a, b는 hyperparameter로, default로는 $a \approx 1.93, b \approx 0.79$를 사용한다고 한다. 보통은 다음과 같이 min_dist라는 hyperparameter를 설정하여 non linear least square 을 통해 a, b를 찾는다고 한다.

$$(1 + a(y_i - y_j)^{2b})^{-1} \approx \begin{cases} 1 & \text{if } |y_i - y_j| \leq \text{min\_dist} \\ e^{-(y_i - y_j) - \text{min\_dist}} & \text{if } |y_i - y_j| > \text{min\_dist} \end{cases}$$

UMAP은 binary cross entropy(CE)를 cost function으로 사용한다.

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

Gradient는 다음과 같다.

$$\frac{\delta CE}{\delta y_i} = \sum_j \left[\frac{2abd_{ij}^{2(b-1)} P(X)}{1 + ad_{ij}^{2b}} - \frac{2b(1 - P(X))}{d_{ij}^2 (1 + ad_{ij}^{2b})} \right] (y_i - y_j)$$