Data Mining
Quality Analytics

Domain Adaptation using Stochastic Neighborhood Embedding (d-SNE)

2020.05.22 Seminar
발표자 : 정승섭

• 목차

• Introduction

한국: Domain 1

미국: Domain 2

• Introduction

한국: Domain 1

classifier

미국: Domain 2

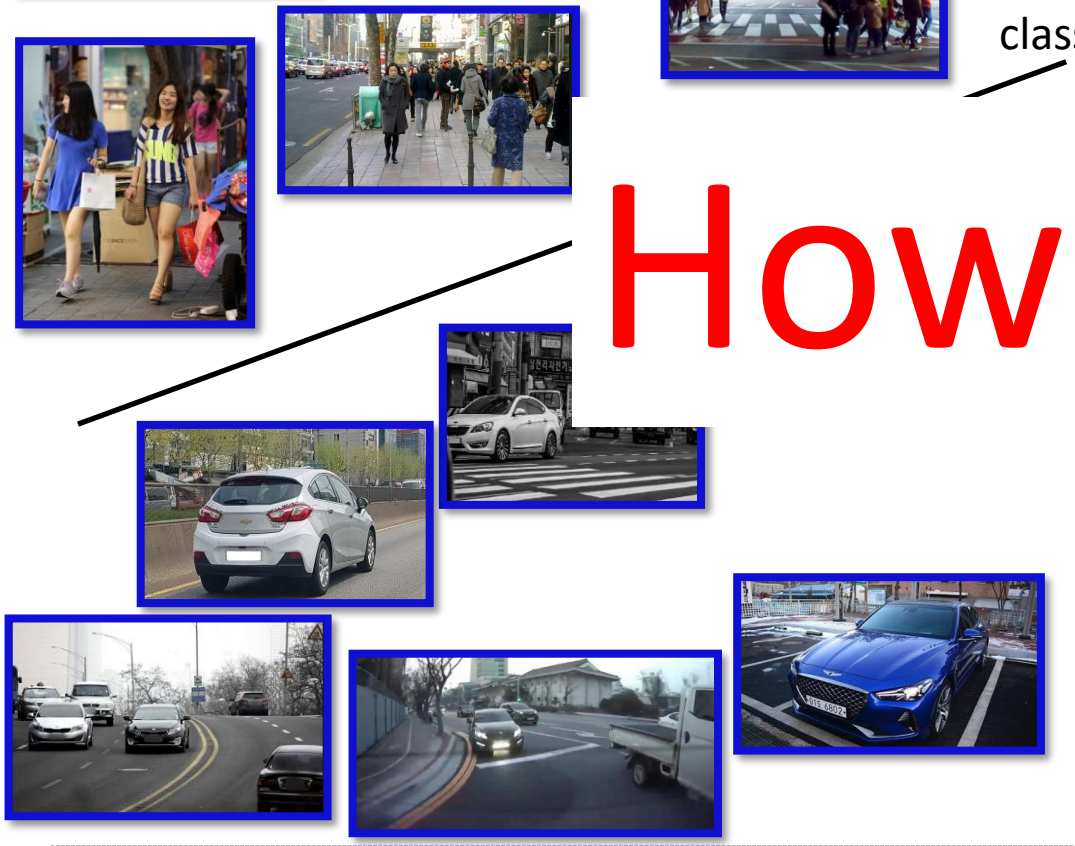
classifier

• Introduction

한국: Domain 1

classifier

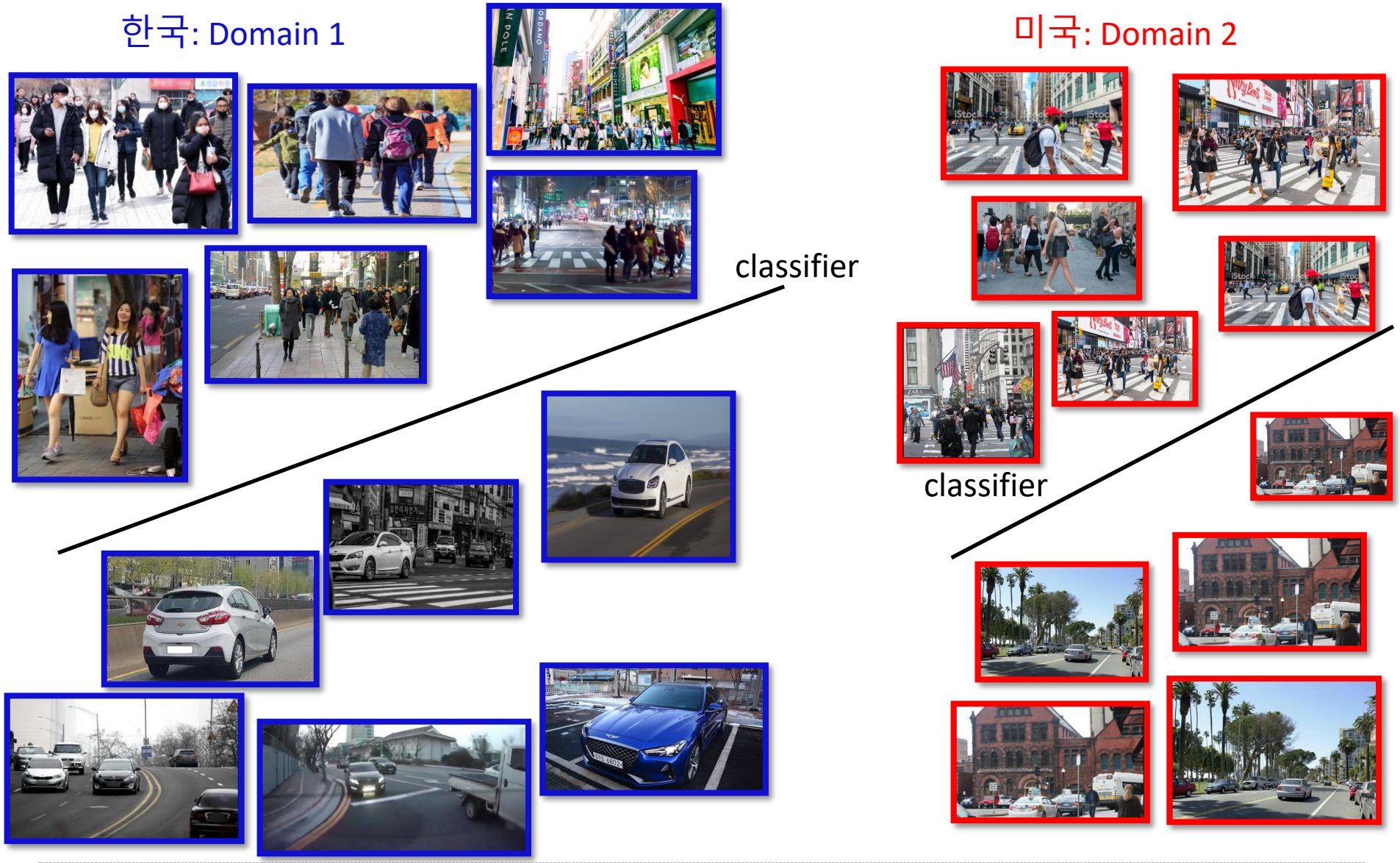
How?



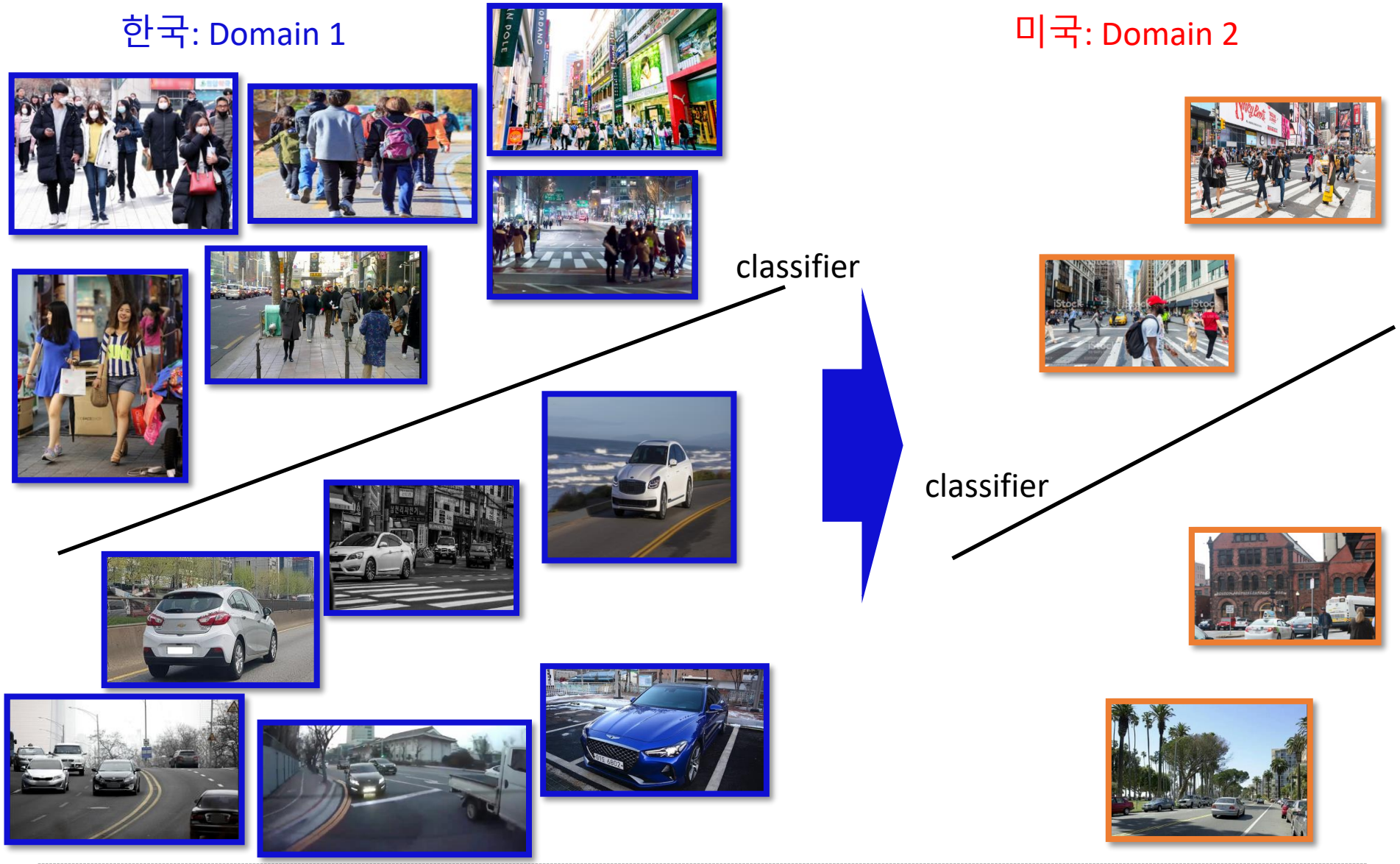
classifier

미국: Domain 2

• Introduction



• Introduction



Domain Adaptation

영역(Domain)이 약간 달라졌을 때

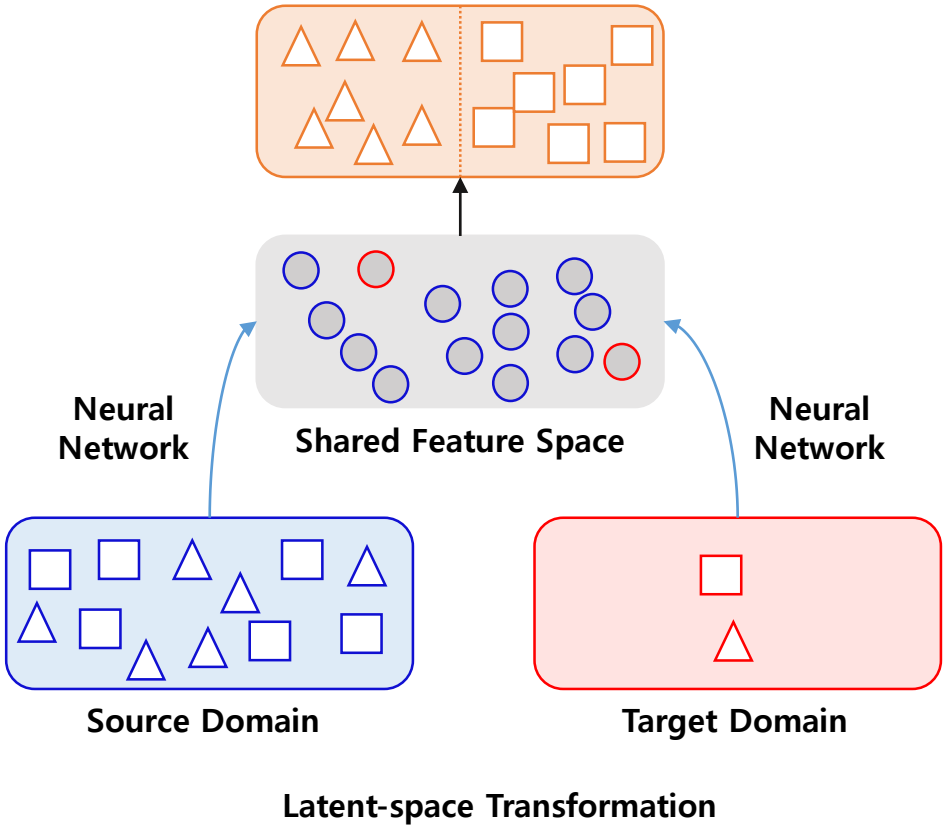
다르지만 관련 있는 새로운 영역에 기존 영역의 정보를 적응(Adaptation)시키는것

영역: 데이터의 분포

- 기존에 모델이 동작하던 영역을 **소스(source)도메인**, 새로운 영역을 **타겟(target)도메인**

• Introduction

목표: 소스도메인의 정보를 타겟도메인에 적응시켜 성능을 높일 수 있는 확률을 올려 주는것

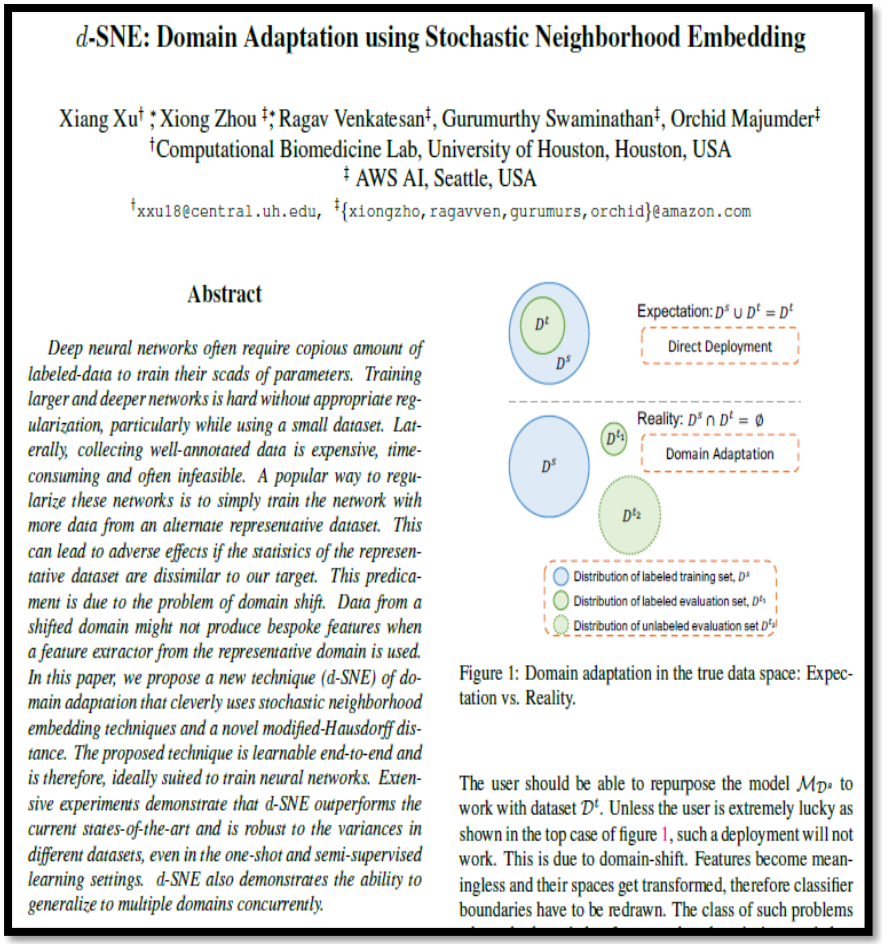


- Latent space위에서 그 데이터의 loss function과 관계를 이해해서 새롭게 모델을 트레이닝

• d-SNE

❖ Computer Vision and Pattern Recognition(CVPR), 2019

d-SNE: Domain adaptation using stochastic neighborhood embedding
X Xu, X Zhou, R Venkatesan... - Proceedings of the ..., 2019 - openaccess.thecvf.com
On the one hand, deep neural networks are effective in learning large datasets. On the other, they are inefficient with their data usage. They often require copious amount of labeled-data to train their scads of parameters. Training larger and deeper networks is hard ...
☆ 99 9회 인용 관련 학술자료 전체 9개의 버전 »



d-SNE: Domain Adaptation using Stochastic Neighborhood Embedding

Xiang Xu[†], Xiong Zhou [‡], Ragav Venkatesan[‡], Gurumurthy Swaminathan[‡], Orchid Majumder[‡]
[†]Computational Biomedicine Lab, University of Houston, Houston, USA
[‡] AWS AI, Seattle, USA
[†]xxu18@central.uh.edu, [‡]{xiongzho,ragavven,gurumurs,orchid}@amazon.com

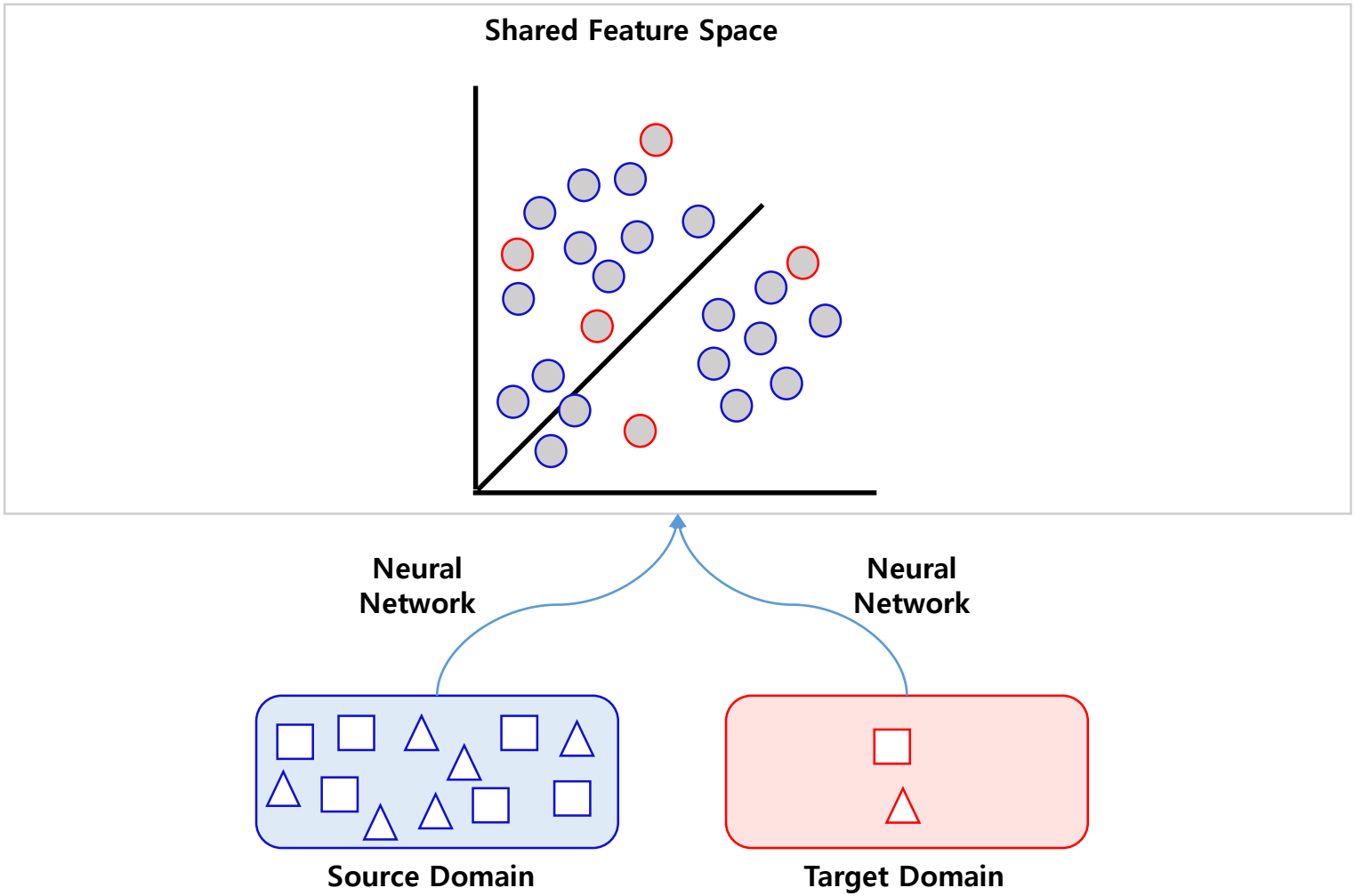
Abstract

Deep neural networks often require copious amount of labeled-data to train their scads of parameters. Training larger and deeper networks is hard without appropriate regularization, particularly while using a small dataset. Laterally, collecting well-annotated data is expensive, time-consuming and often infeasible. A popular way to regularize these networks is to simply train the network with more data from an alternate representative dataset. This can lead to adverse effects if the statistics of the representative dataset are dissimilar to our target. This predicament is due to the problem of domain shift. Data from a shifted domain might not produce bespoke features when a feature extractor from the representative domain is used. In this paper, we propose a new technique (d-SNE) of domain adaptation that cleverly uses stochastic neighborhood embedding techniques and a novel modified-Hausdorff distance. The proposed technique is learnable end-to-end and is therefore, ideally suited to train neural networks. Extensive experiments demonstrate that d-SNE outperforms the current states-of-the-art and is robust to the variances in different datasets, even in the one-shot and semi-supervised learning settings. d-SNE also demonstrates the ability to generalize to multiple domains concurrently.

Figure 1: Domain adaptation in the true data space: Expectation vs. Reality.

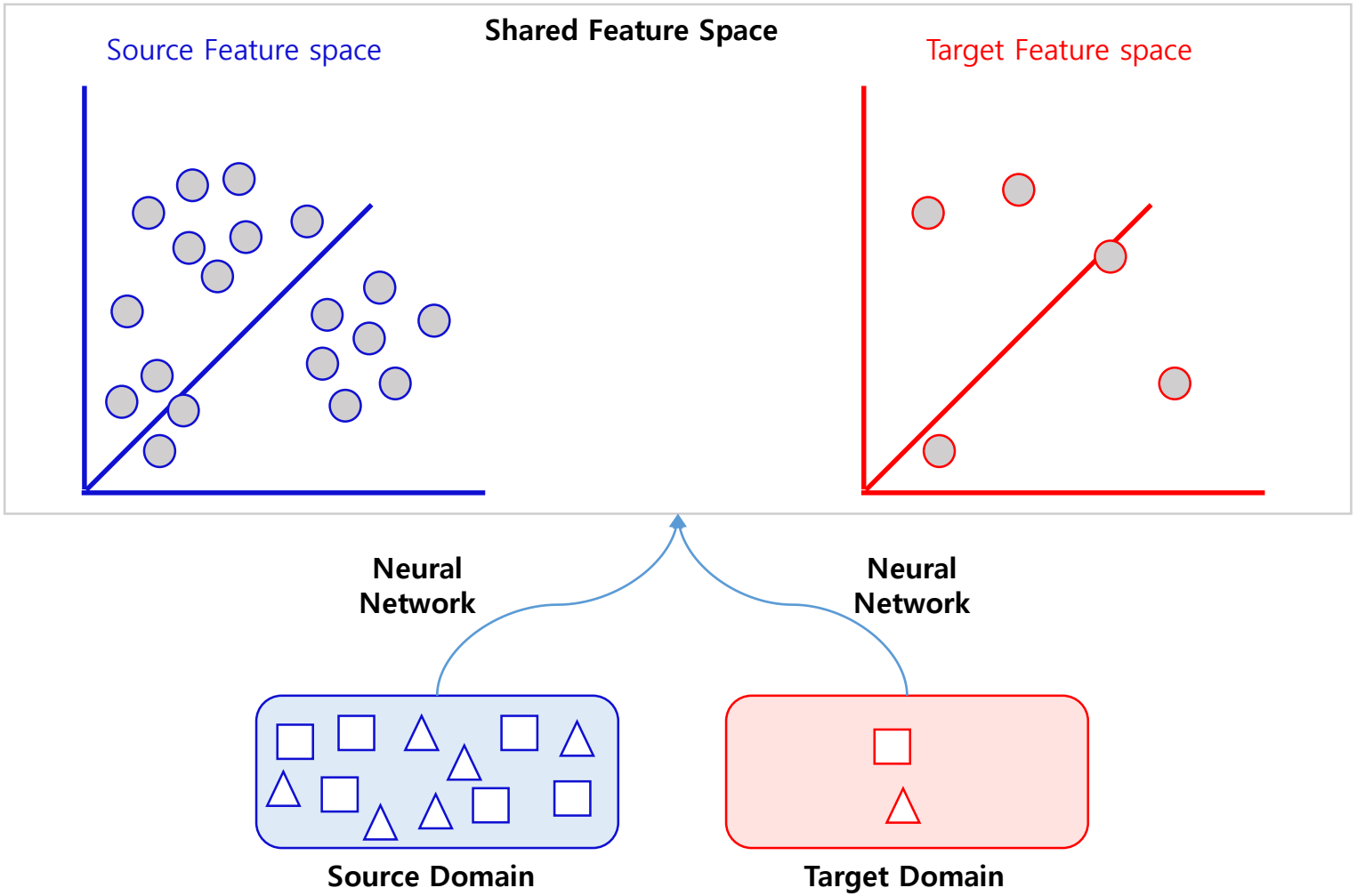
The user should be able to repurpose the model $\mathcal{M}_{\mathcal{D}^s}$ to work with dataset $\mathcal{D}^t$. Unless the user is extremely lucky as shown in the top case of figure 1, such a deployment will not work. This is due to domain-shift. Features become meaningless and their spaces get transformed, therefore classifier boundaries have to be redrawn. The class of such problems

• d-SNE

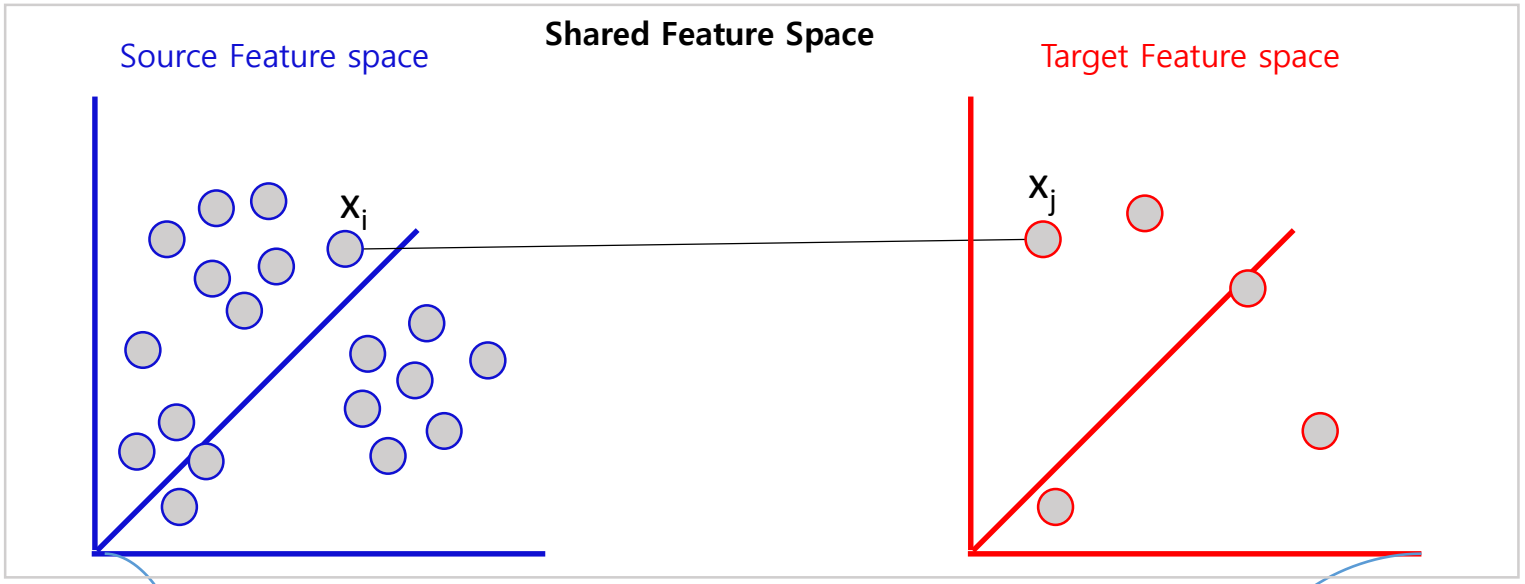


소스도메인과 타겟도메인 각각의 neural networks를 통해 만들어진 feature값들의 공간 (Latent Space)

• d-SNE


Shared Feature Space
Source Feature space
Target Feature space
Neural Network
Neural Network
Source Domain
Target Domain

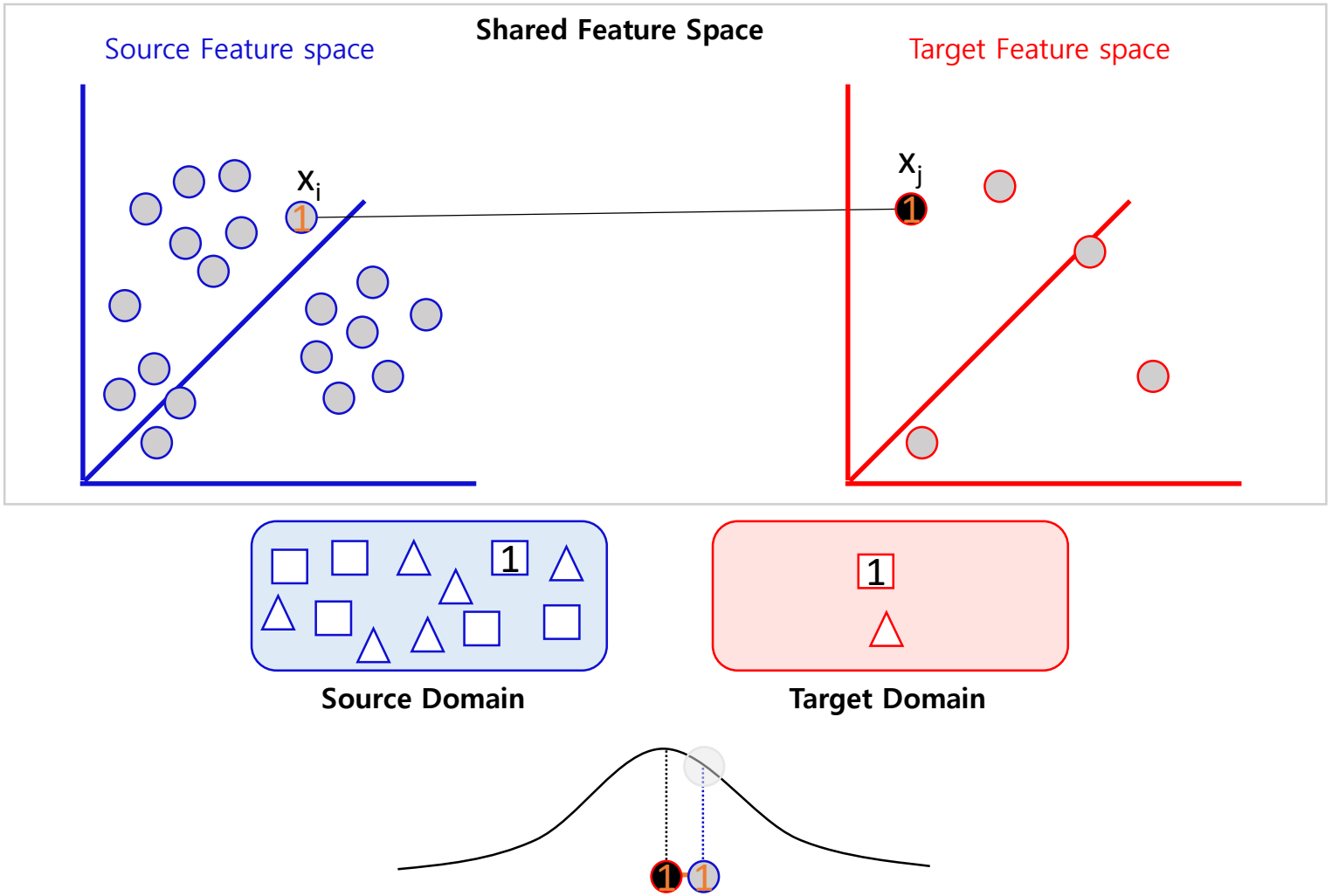
• d-SNE



$$d\left(x_i^s, x_j^t\right) = \left\|\Phi_{D^s}(x_i^s) - \Phi_{D^t}\left(x_j^t\right)\right\|_2^2$$

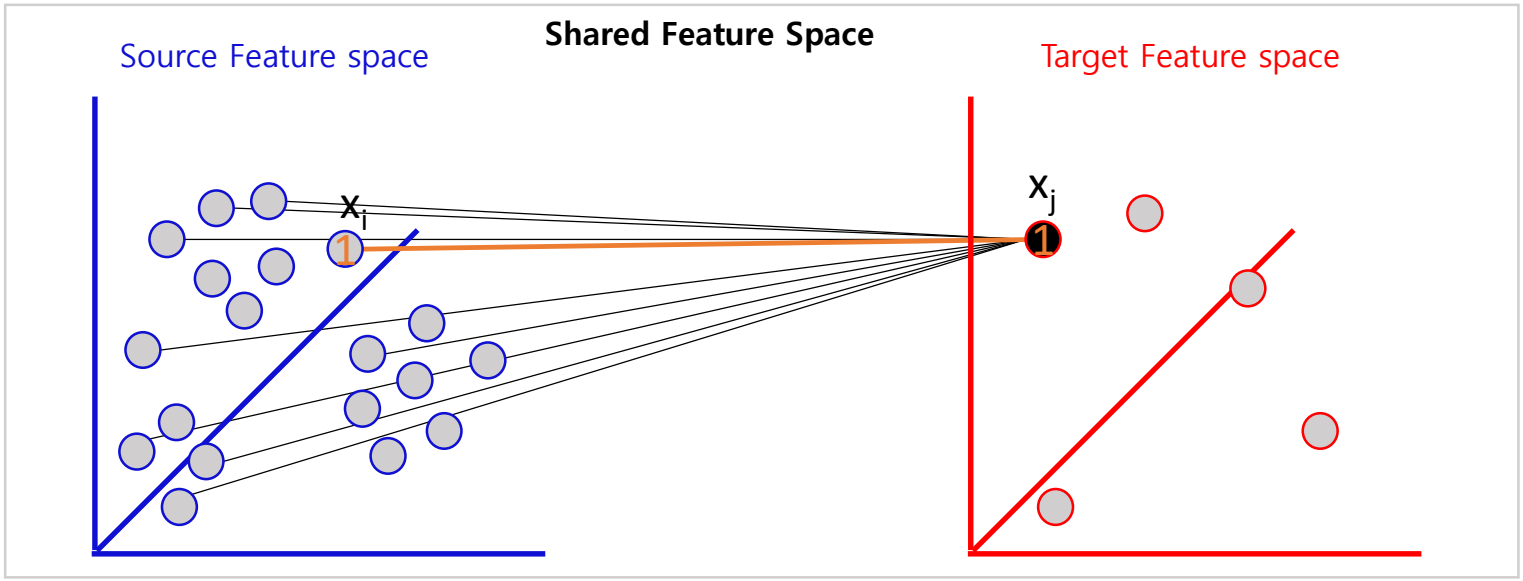
Source feature space의 샘플 x_i와 Target feature space의 샘플 x_j를 유클리디안 거리로 계산

• d-SNE

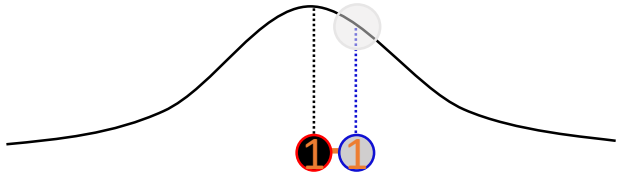


타겟샘플 x_j를 기준으로 소스샘플 x_i가 같은 label을 갖을 확률을 정규분포를 기반으로 판단
x_j, x_i 가 같은 label을 가질 경우 거리상 가까울 것이고 확률도 높아진다.

• d-SNE

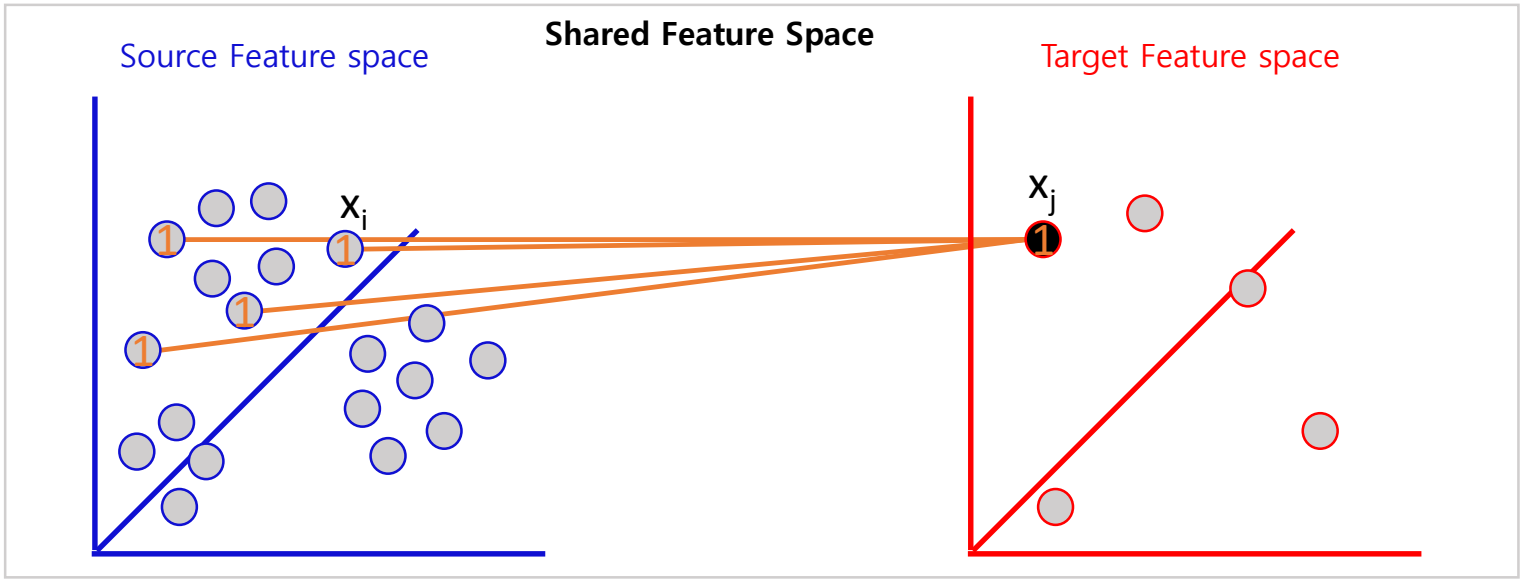


$$p_{ij} = \frac{\exp(-d(x_i^s, x_j^t))}{\sum_{x \in D^s} \exp(-d(x, x_j^t))}$$



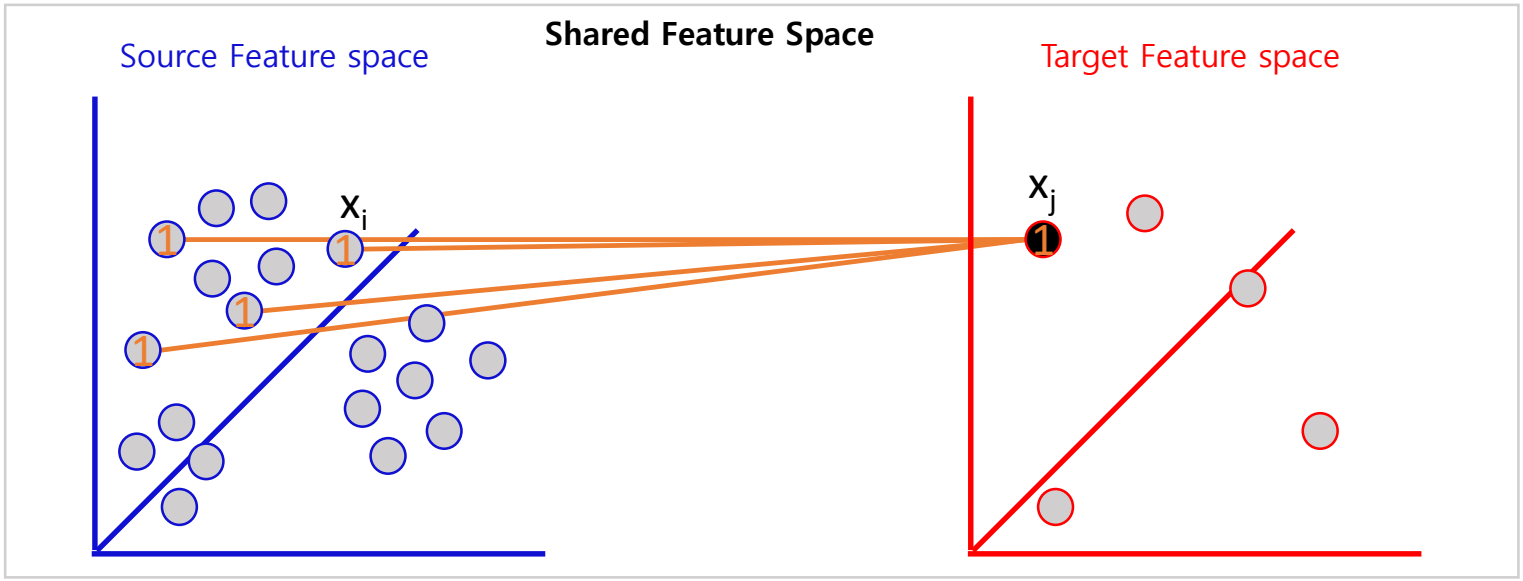
타겟샘플 x_j를 기준으로 소스샘플 x_i가 같은 label을 갖을 확률을 정규분포를 기반으로 판단
x_j, x_i 가 같은 label을 가질 경우 거리상 가까울 것이고 확률도 높아진다.

• d-SNE



타겟샘플 x_j와 같은 label을 갖는 모든 소스샘플들의 확률

• d-SNE

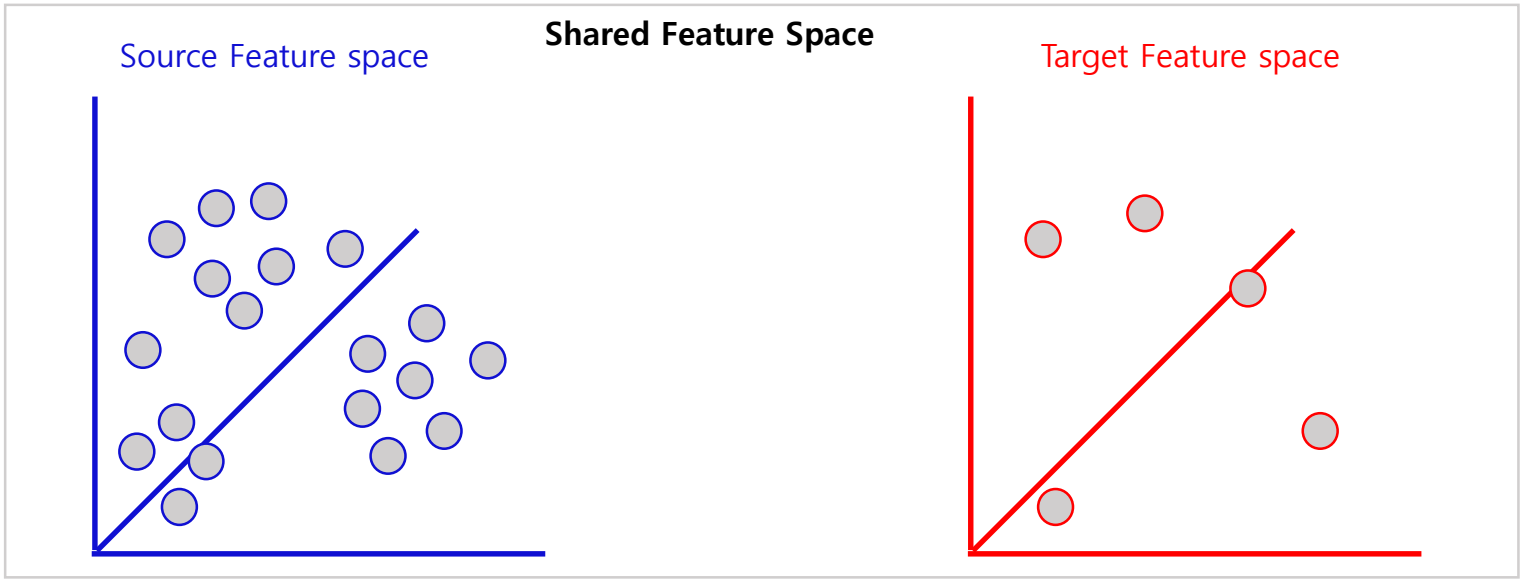


$$p_j = \frac{\sum_{x \in D_k^s} \exp(-d(x, x_j^t))}{\sum_{x \in D^s} \exp(-d(x, x_j^t))} = \sum_{i=0}^{N_k^s} p_{ij}\,,\ N_k^s = |D_k^s|$$

$$\sum_{x \in D^s} \exp\left(-d(x, x_j^t)\right) = \sum_{x \in D_k^s} \exp(-d(x, x_j^t)) + \sum_{x \in D_{\bar{k}}^s} \exp(-d(x, x_j^t))$$

타겟샘플 x_j와 같은 label을 갖는 모든 소스샘플들의 확률

• d-SNE

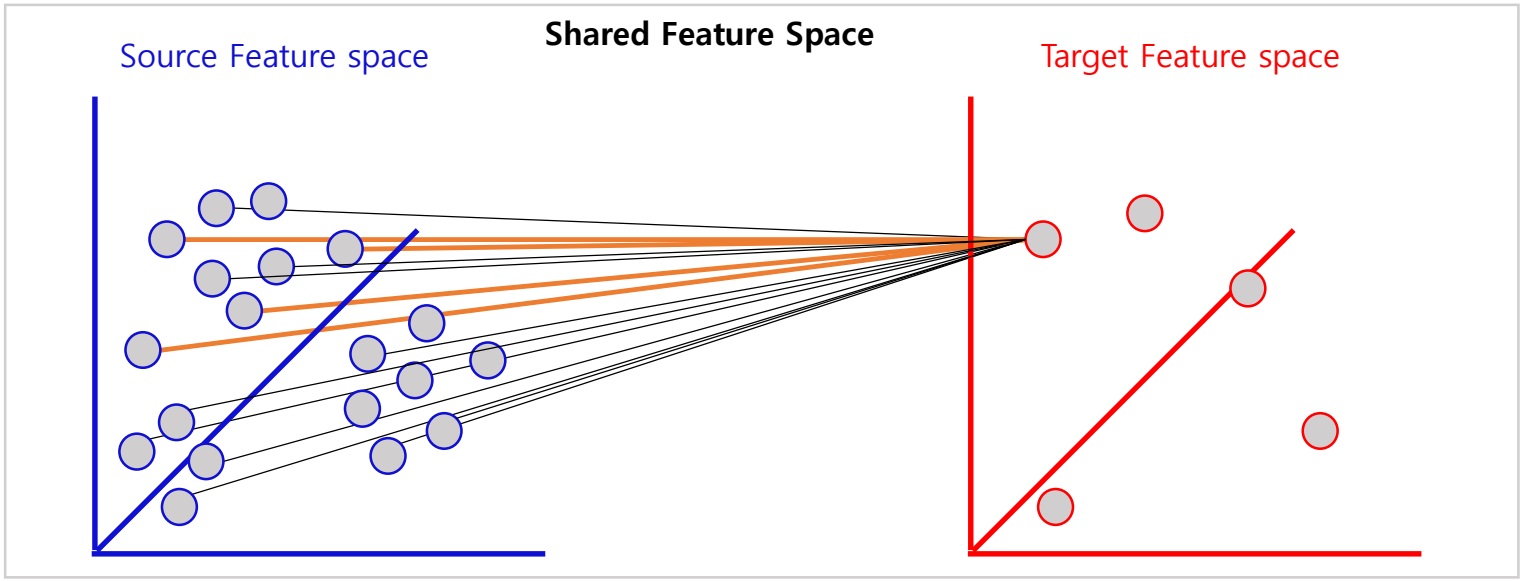


$$\sum_{x_j \in D^t} \frac{1}{p_j} = \sum_{x_j \in D^t} \left(\frac{\sum_{x \in D_k^s} \exp(-d(x, x_j))}{\sum_{x \in D_k^s} \exp(-d(x, x_j))}, for\ k = yj \right)$$

$$\mathcal{L} = \log\left(\frac{\sum_{x \in D_k^s} \exp(-d(x, x_j))}{\sum_{x \in D_k^s} \exp(-d(x, x_j))} \right), for\ k = yj)$$

Loss function을 최소화 하기 위해 역수를 취해 줌

- d-SNE

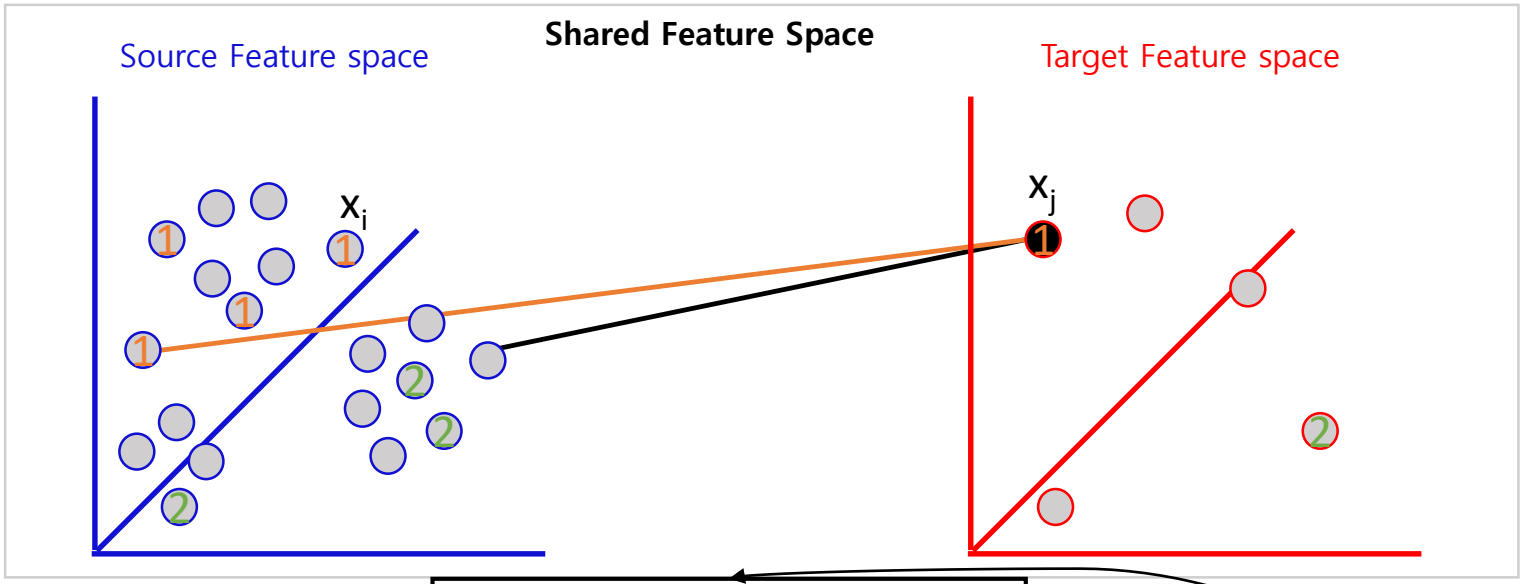


$$\sum_{x_j \in D^t} \frac{1}{p_j} = \sum_{x_j \in D^t} \left(\frac{\sum_{x \in D_k^s} \exp(-d(x, x_j))}{\sum_{x \in D_k^s} \exp(-d(x, x_j))}, for\ k = yj \right)$$

$$\mathcal{L} = \log\left(\frac{\sum_{x \in D_k^s} \exp(-d(x, x_j))}{\sum_{x \in D_k^s} \exp(-d(x, x_j))} \right), for\ k = yj)$$

Loss function을 어떻게 최적화 시킬것인가? 모든 거리를 다 계산해야 하나?

- d-SNE

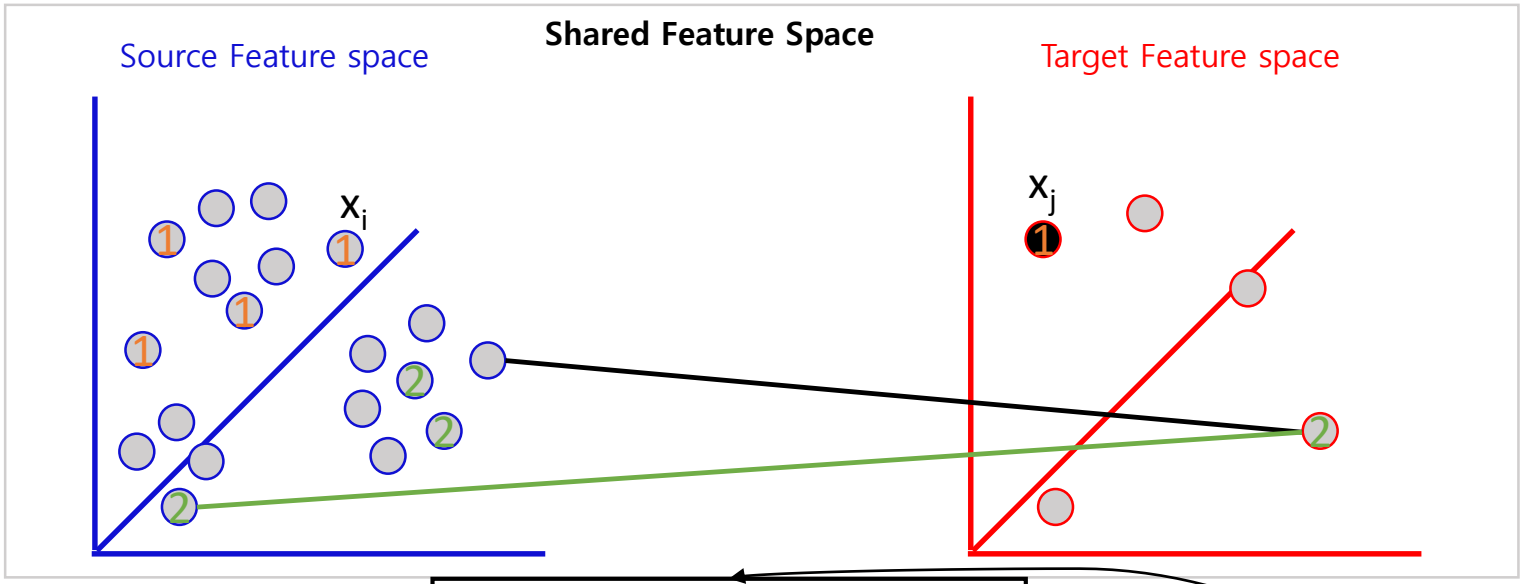


$$\mathcal{L} = \log\left(\frac{\sum_{x \in D_k^s} \exp(-d(x, x_j))}{\sum_{x \in D_k^s} \exp(-d(x, x_j))}\right), for\ k = yj$$

$$\tilde{\mathcal{L}} = \sup_{x \in D_k^s}\{a | a \in d(x, xj)\} - \inf_{x \in D_k^s}\{b | b \in d(x, xj)\}, for\ k = yj$$

동일한 label을 가지고 있는 샘플들의 거리 중 가장 거리가 먼것을 최소화
다른 label을 가지고 있는 샘플들의 거리 중 가장 거리가 가까운것을 최대화

- d-SNE

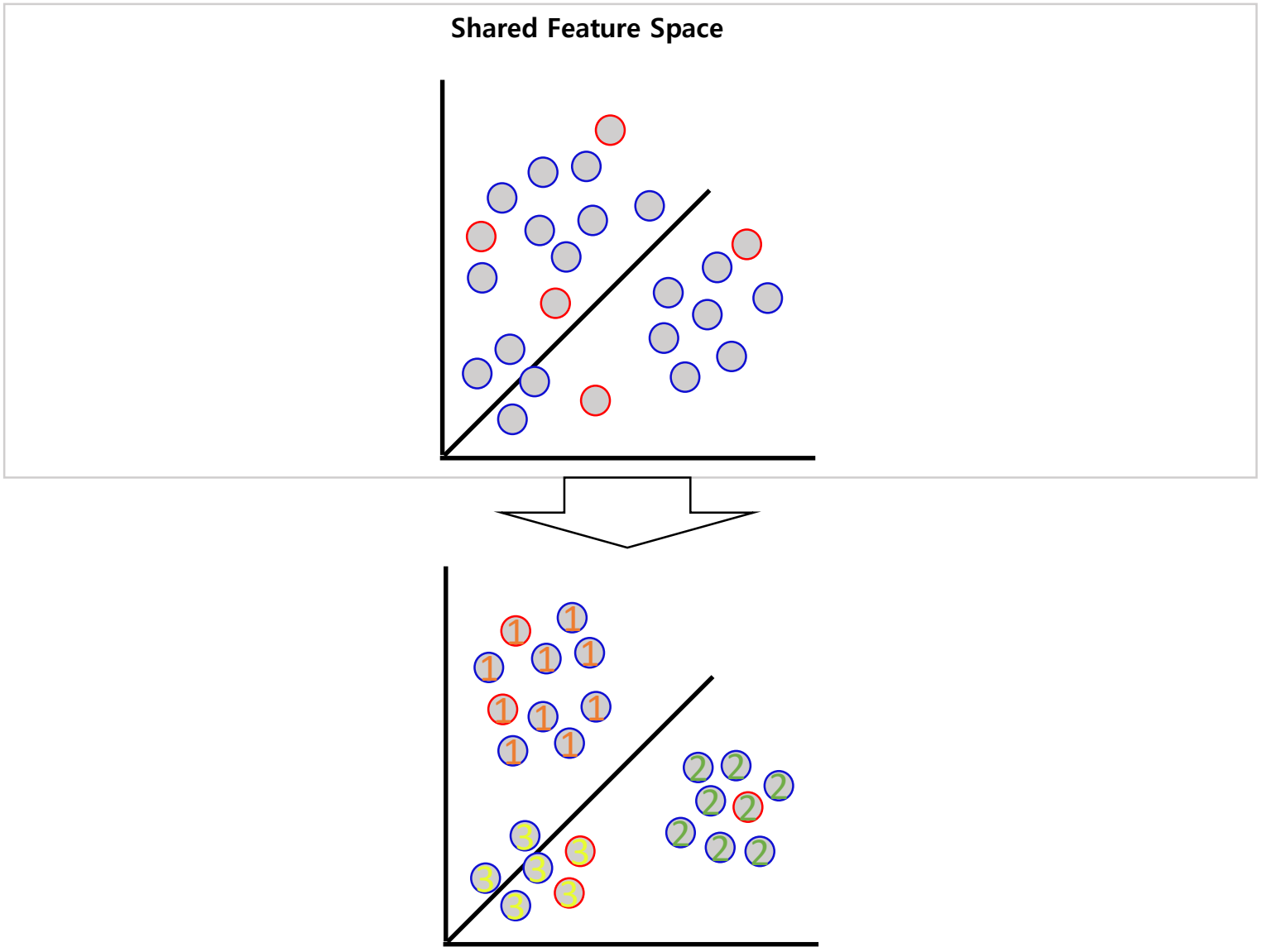


$$\mathcal{L} = \log\left(\frac{\sum_{x \in D_k^s} \exp(-d(x, x_j))}{\sum_{x \in D_k^s} \exp(-d(x, x_j))}\right), for\ k = yj$$

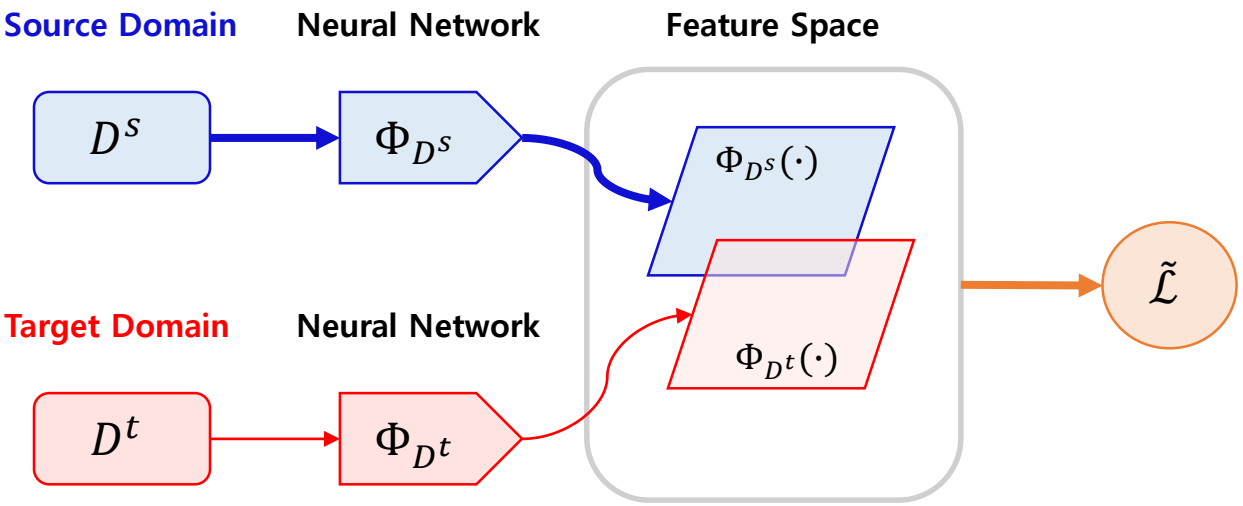
$$\tilde{\mathcal{L}} = \sup_{x \in D_k^s}\{a | a \in d(x, xj)\} - \inf_{x \in D_k^s}\{b | b \in d(x, xj)\}, for\ k = yj$$

목표: 같은 label인 점끼리는 최대한 가깝게 다른 label인 점끼리는 최대한 멀게 하자

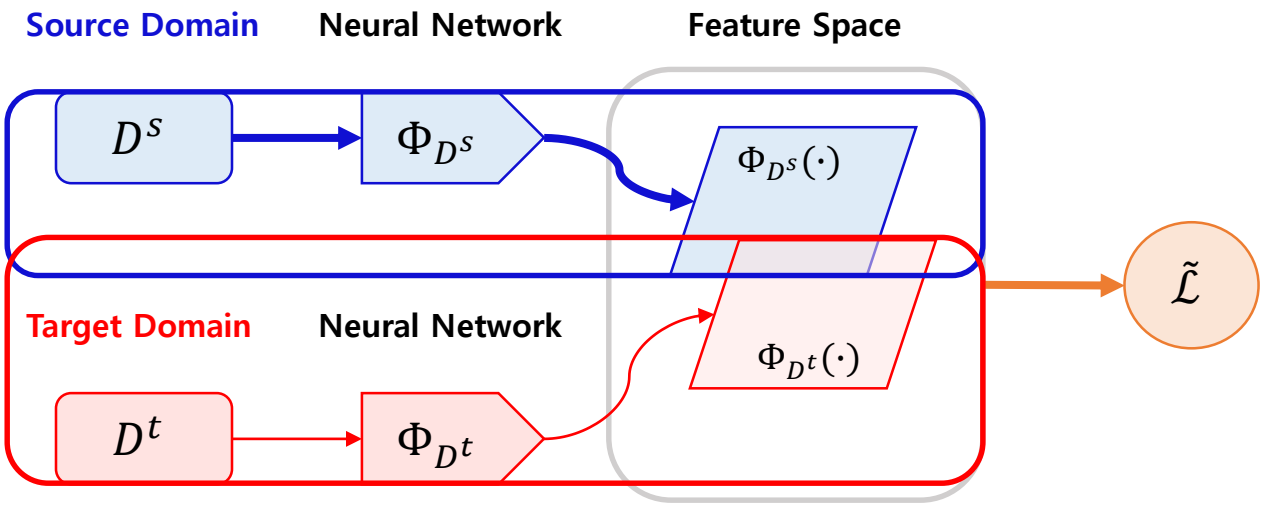
• d-SNE



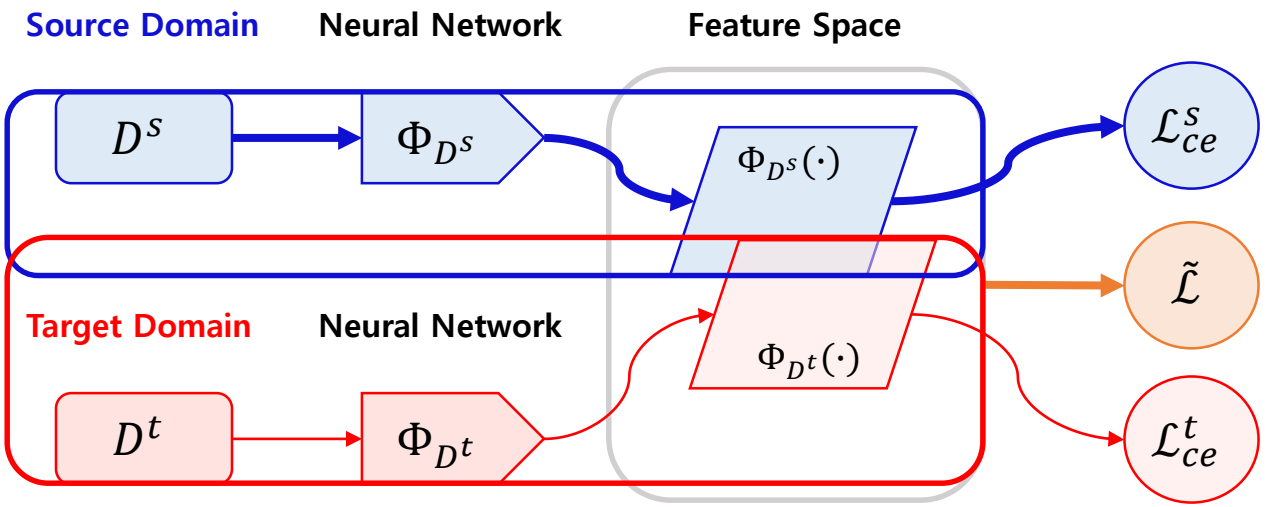
• d-SNE


Source Domain
Neural Network
Feature Space
D^s
Φ_{D^s}
$\Phi_{D^s}(\cdot)$
Target Domain
Neural Network
D^t
Φ_{D^t}
$\Phi_{D^t}(\cdot)$
$\tilde{\mathcal{L}}$

• d-SNE


Source Domain
Neural Network
Feature Space
D^s
Φ_{D^s}
$\Phi_{D^s}(\cdot)$
Target Domain
Neural Network
$\Phi_{D^t}(\cdot)$
D^t
Φ_{D^t}
$\tilde{\mathcal{L}}$

• d-SNE



Learning formulation: $\underset{w_s, w_t}{argmin}\ \tilde{\mathcal{L}} + \alpha\mathcal{L}_{ce}^s + \beta\mathcal{L}_{ce}^t$

• Result

Source Domain
MNIST

Target Domain
USPS

$\lvert\mathcal{D}_k^t\rvert$, $\forall k$	0	1	3	5	7
CCSA [16]	65.40	85.00	90.10	92.40	92.90
FADA [15]	65.40	89.10	91.90	93.40	94.40
d-SNE	73.01	**92.90**	**93.55**	**95.13**	**96.13**

National Institute of Standards and Technology(MNIST)에서 손으로 쓴 글자 데이터셋에서 숫자만 뽑아낸 데이터셋
US Postal Service(USPS)에서 손으로 쓴 숫자 데이터셋
각 숫자 클래스마다 트레이닝을 하기 위해 샘플을 0 to 7까지 타겟도메인에서 랜덤하게 사용

• Result

MNIST

MNIST-M

USPS

SVHN

Method	$\lvert\mathcal{D}_k^t\rvert,\ \forall k$	Setting	MNIST → MNIST-M	MNIST → USPS	USPS → MNIST	MNIST → SVHN	SVHN → MNIST
PixelDA [1]			98.20	95.90	-	-	-
ADA [10]			87.47	-	-	-	97.60
I2I [17]		$\mathcal{U}$	-	95.1	92.2	-	92.1
DIRT-T [25]			98.90	-	-	54.50	**99.40**
SE [6]			-	98.26 ± 0.11	98.07 ± 2.82	13.96 ± 4.41	99.18 ± 0.12
SBADA-GAN [21]			**99.40**	95.04	97.60	61.08	76.14
G2A [23]			-	95.30 ± 0.70	90.80 ± 1.30	-	92.40 ± 0.90
FADA [15]	7	$\mathcal{S}$	-	94.40	91.50	47.00	87.20
CCSA [16]	10		78.29 ± 2.00	97.27 ± 0.19	95.71 ± 0.42	37.63 ± 3.62	94.57 ± 0.40
d-SNE	0	$\mathcal{S}$	50.98 ± 1.64	93.16 ± 0.71	83.37 ± 0.93	26.22 ± 2.02	66.02 ± 0.72
	7		84.62 ± 0.04	97.53 ± 0.10	97.52 ± 0.08	53.19 ± 0.28	95.68 ± 0.03
	10		*87.80 ± 0.16*	**99.00 ± 0.08**	**98.49 ± 0.35**	**61.73 ± 0.47**	*96.45 ± 0.20*

National Institute of Standards and Technology(MNIST)
MNIST-M
US Postal Service(USPS)
The Street View House Numbers(SVHN)

• Result

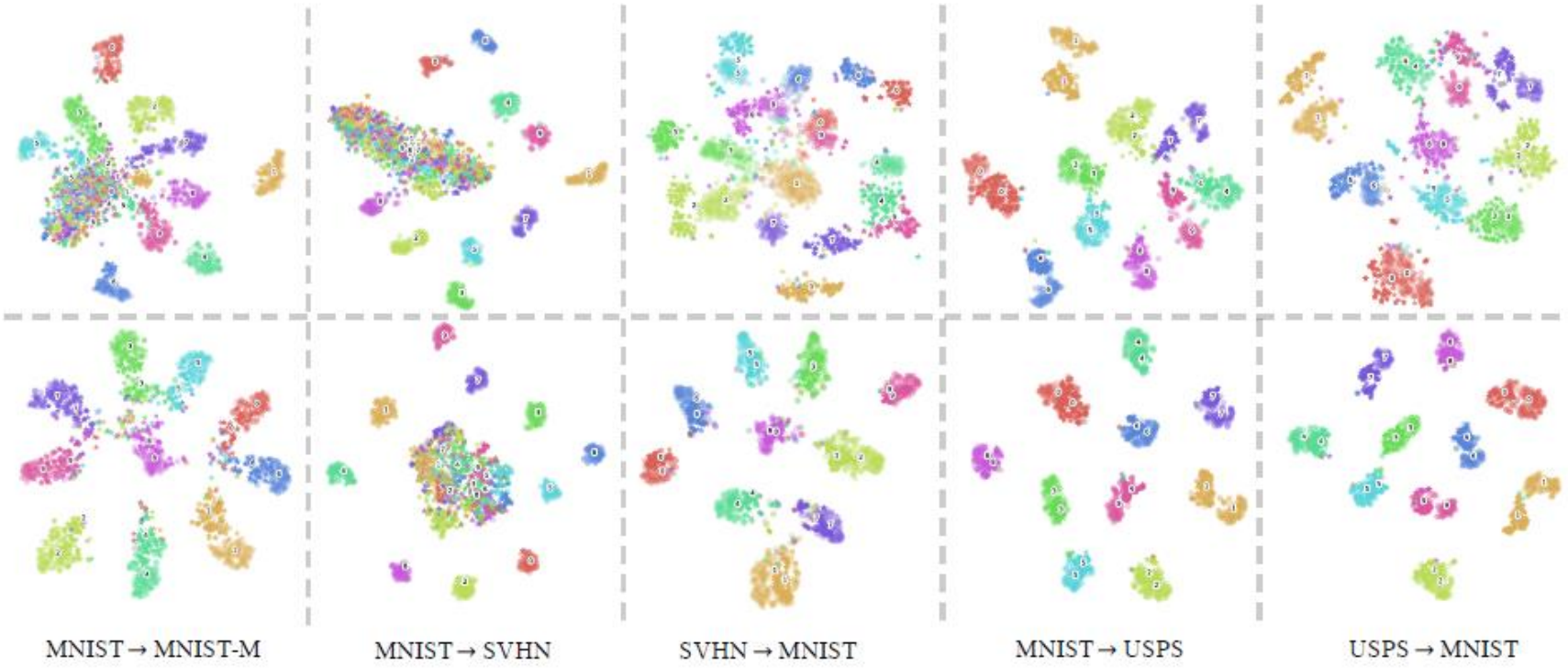


Figure 5: t-SNE visualizations without (top) and with (bottom) domain adaptations.

위쪽은 t-SNE만 사용했을때의 visualization
아래쪽은 d-SNE를 이용했을때의 visualization 훨씬 잘 구분 하는것을 보여준다.

• Conclusion

❖ Comment

① Label이 잘되어진 타겟도메인이 별로 없을 경우 분류 정확도 향상에 도움이 됨

② Modified-Hausdorff distance를 사용하여 기존 SNE나 t-SNE에 비해 더욱 빠른 연산 처리속도를 얻고도 t-SNE에 비해 더 정교한 visualization을 보여줌

③ Pre-trained 된 어떤 Neural network를 가져와 사용할 수 있음

• Result에서 참고된 논문

- CCSA: Unified Deep Supervised Domain Adaptation and Generalization, 2017
- FADA: Few-Shot Adversarial Domain Adaptation, 2017
- PixelDA: Unsupervised pixel-level domain adaptation with generative adversarial networks, 2017
- ADA: Associative Domain Adaptation, 2017
- I2I: Image to image translation for domain adaptation, 2018
- DIRT-T: A DIRT-T Approach to unsupervised Domain Adaption, 2018
- SE: Self-Ensembling for Visual Domain Adaptation, 2017
- SBADA-GAN: symmetric bi-directional adaptive GAN, 2018
- G2A: Aligning Domains using Generative Adversarial Networks, 2018

감사합니다.

Appendix

❖ SNE

Stochastic Neighbor Embedding

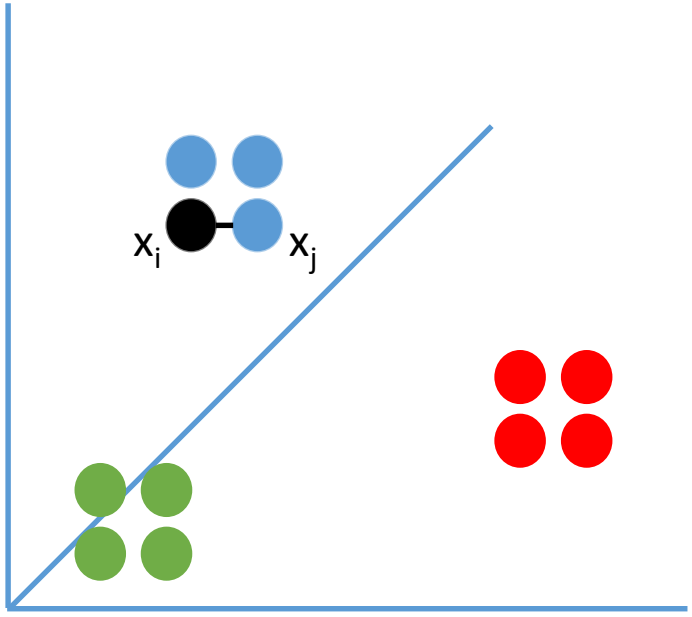
각 데이터포인트 x_i 마다 확률분포 P_i 를 정의.
확률적인 개념이 들어가기에 stochastic이 사용

그 확률분포 P_i 는 데이터포인트 x_i 가 어떤 데이터포인트 x_j를 이웃이라고 선택할 확률
x_i 와 가까운 데이터는 확률값이 크게 먼 데이터는 확률값이 작도록 P_i 를 정의

x_i 를 저차원의 y_i로 임배딩해주는데 y_i 에 대해서도 확률분포 Q_i가 P_i 와 최대한 비슷해지는 방향으로 임배딩하는것

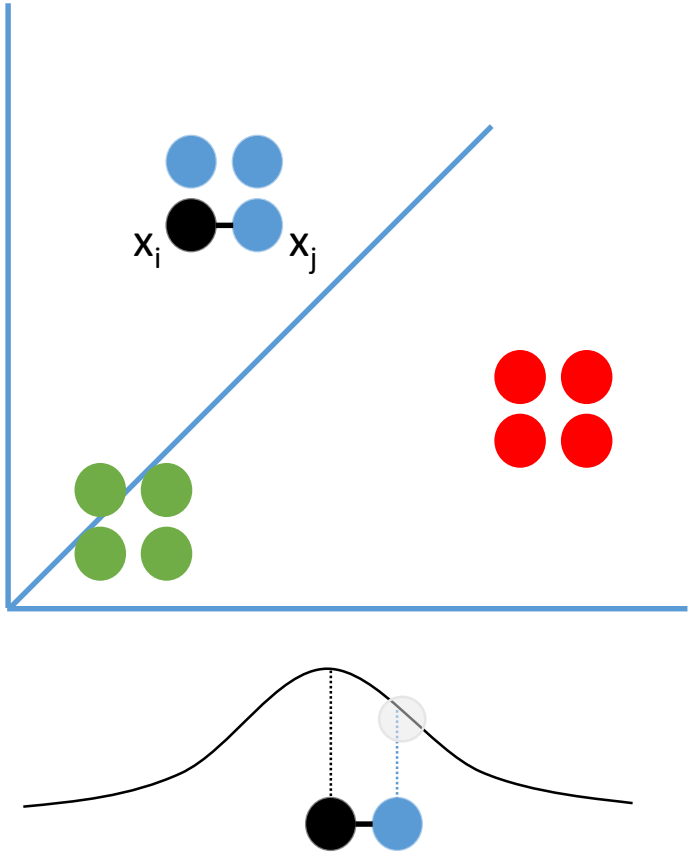
• SNE / t-SNE

❖ SNE



먼저 점 하나를 선택하고 선택한 점부터 다른 점까지의 거리를 측정

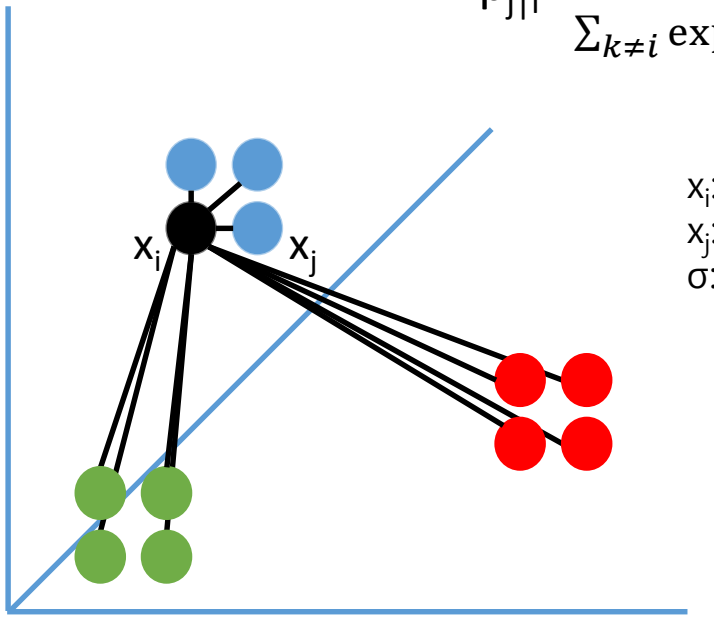
❖ SNE



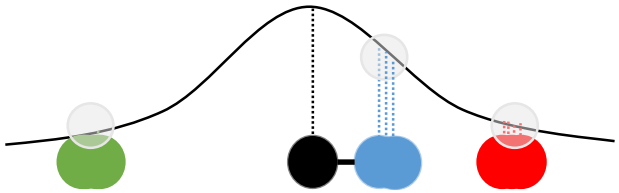
가우시안분포 그래프를 이용하여 두 점 사이의 유클리디안 거리(두 점간 최단거리)를 유사도 (Similarity) 로 변환
유사도를 가우시안분포(조건부 확률분포)를 기반으로 판단

❖ SNE

$$p_{j|i} = \frac{\exp(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2})} \text{ for all } j \neq I,\ p_{i|i} = 0$$



x_i: 고차원의 기준점(평균)
x_j: 고차원의 상대점
σ: x_i 를 중심으로 한 분산

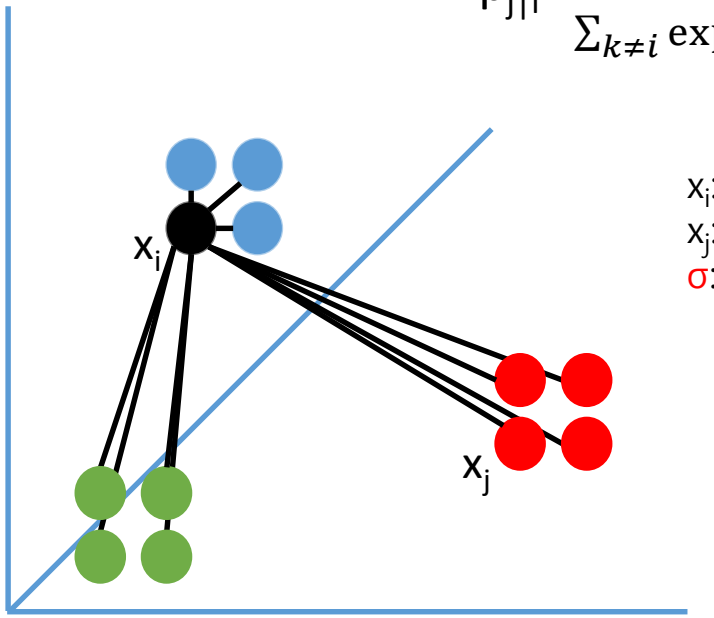


기준점과 가까울수록 유사도 값이 크고 멀어질수록 유사도 값이 작아짐

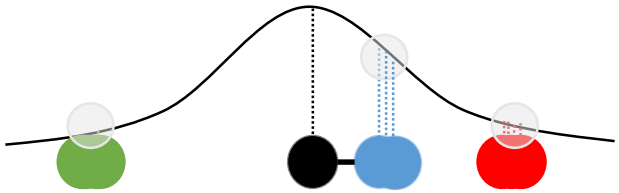
• SNE / t-SNE

❖ SNE

$$p_{j|i} = \frac{\exp(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{\|x_i - x_k\|^2}{2\sigma_i^2})} \text{ for all } j \neq I,\ p_{i|i} = 0$$



x_i: 고차원의 기준점(평균)
x_j: 고차원의 상대점
σ: x_i 를 중심으로 한 분산



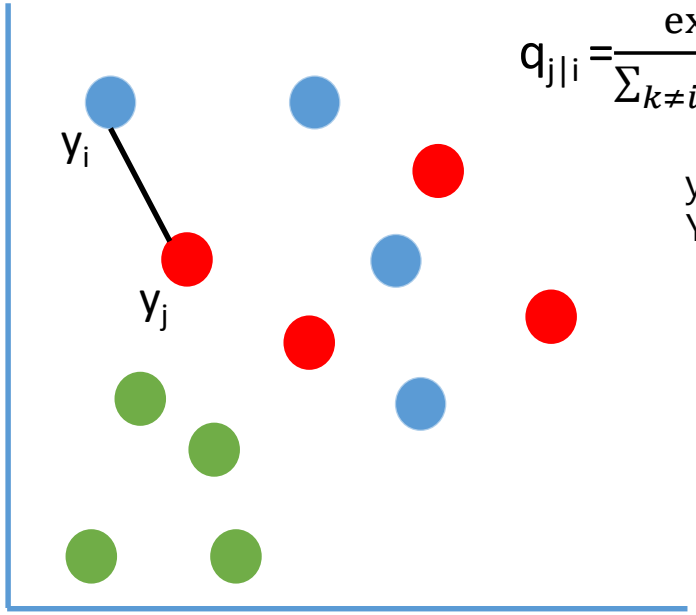
σ_i 는 각 개체마다 데이터 밀도가 달라 매번 계산되어야 하지만 반복 실험 결과 고정된 값을 사용해도 큰 문제 없음. 5~50 사이값으로 정해도 크게 문제 없다.
(Perplexity 값 설정, 학습에 영향을 주는 점들의 개수를 조절)

• SNE / t-SNE

❖ SNE

x_i, x_j → y_i, y_j

고차원 공간표현 → 저차원 공간표현



$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k \neq i} \exp(-\|y_i - y_k\|^2)} \quad \sigma = \frac{1}{\sqrt{2}}, \quad q_{i|i} = 0$$

y_i: 저차원의 기준점(평균)
Y_j: 저차원의 상대점

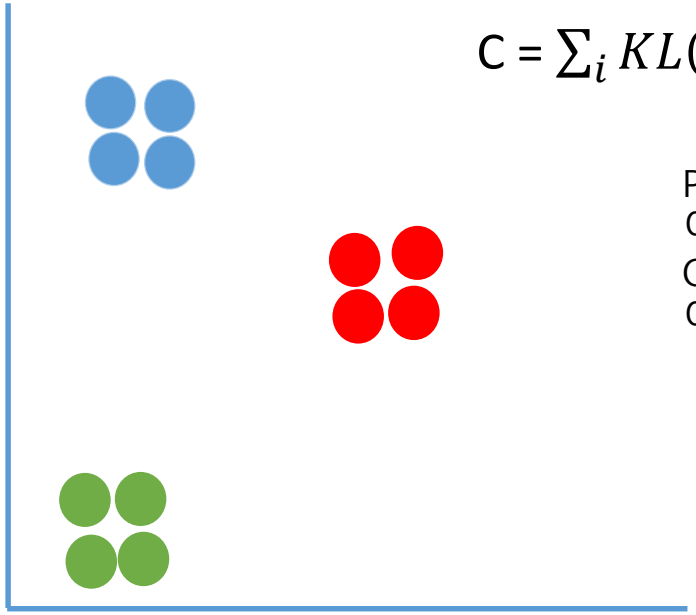
처음 저차원 공간에 y_i, y_j 을 Random하게 Initialize 하고
y_i 에 대해서도 확률분포 Q_i가 P_i와 최대한 비슷해지는 방향으로 임배딩

• SNE / t-SNE

❖ SNE

x_i, x_j → y_i, y_j

고차원 공간표현 → 저차원 공간표현



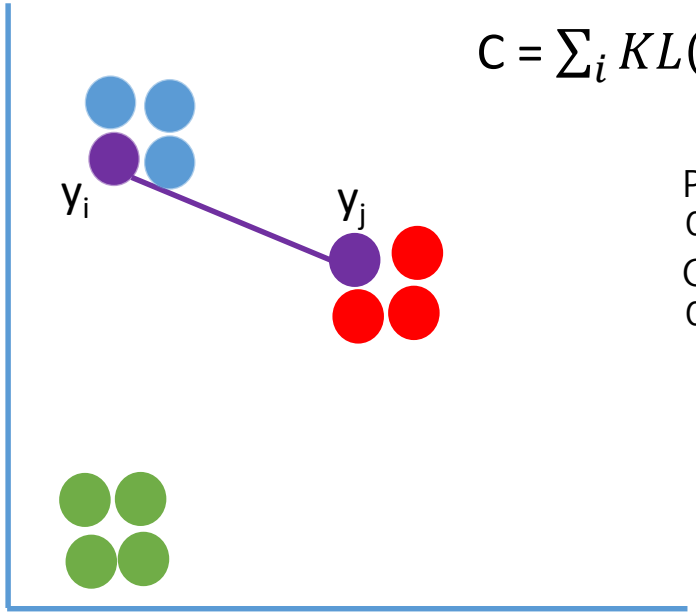
$$C = \sum_i KL(P_i|Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

P_i: 고차원 데이터 점 x_i대한 모든 데이터 점의 조건부 확률분포
Q_i: 저차원 데이터 점 y_i대한 모든 데이터 점의 조건부 확률분포

$$Minimize\ C\ by\ using\ \frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(yi - yj)$$

$p_{j|i}$ 과 $q_{j|i}$ 간의 KL divergence를 최소화 하도록 계산.
KL divergence는 한 확률 분포가 두 번째 예상 확률 분포와 어떻게 다른지 측정하는 척도
두 분포가 완전히 다르면 1, 동일하면 0의값을 가짐

❖ SNE의 문제점: Cost function의 최적화가 어려움



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

P_i: 고차원 데이터 점 x_i대한 모든 데이터 점의 조건부 확률분포
Q_i: 저차원 데이터 점 y_i대한 모든 데이터 점의 조건부 확률분포

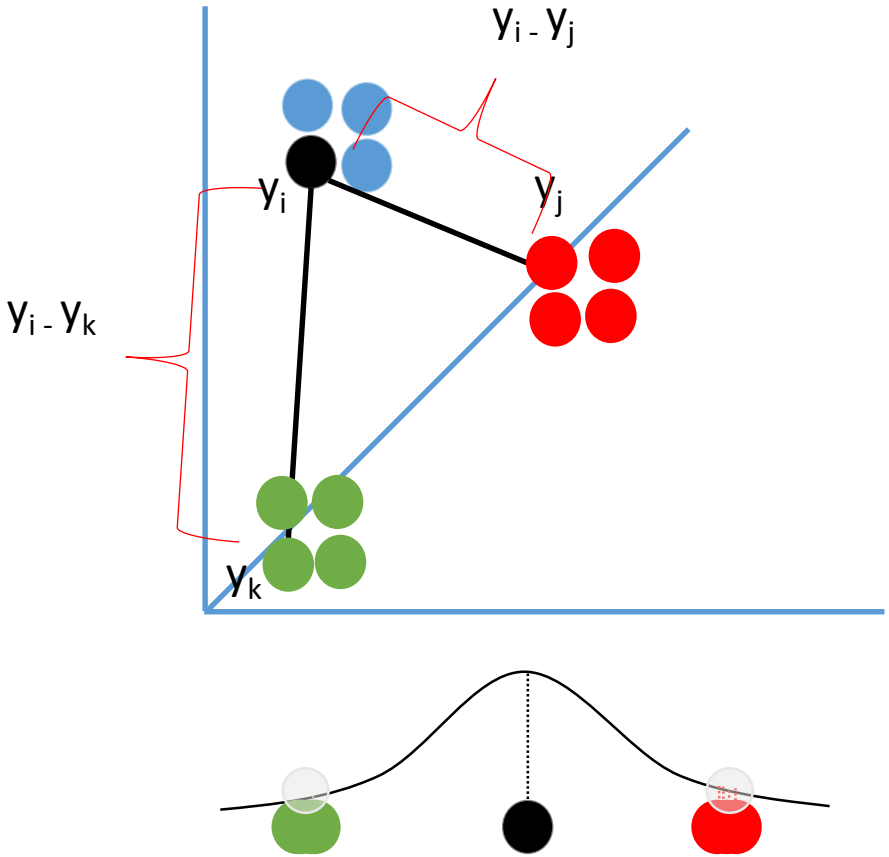
$$Minimize\ C\ by\ using\ \frac{\delta C}{\delta y_i} = 4 \sum_j (y_j - yi)(pij - qij)$$

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

$$q_{ij} = \frac{q_{j|i} + qi_{|j}}{2N}$$

$p_{j|i} = p_{i|j}$ / $q_{j|i} = q_{i|j}$ 즉 i번째 개체가 주어졌을때 j번째 개체가 이웃으로 뽑힐 확률과 j번째 개체가 주어졌을때 i번째가 이웃으로 뽑힐 확률이 동일하다고 놓아도 성능의 차이가 크지않음

❖ SNE의 문제점: Crowding Problem

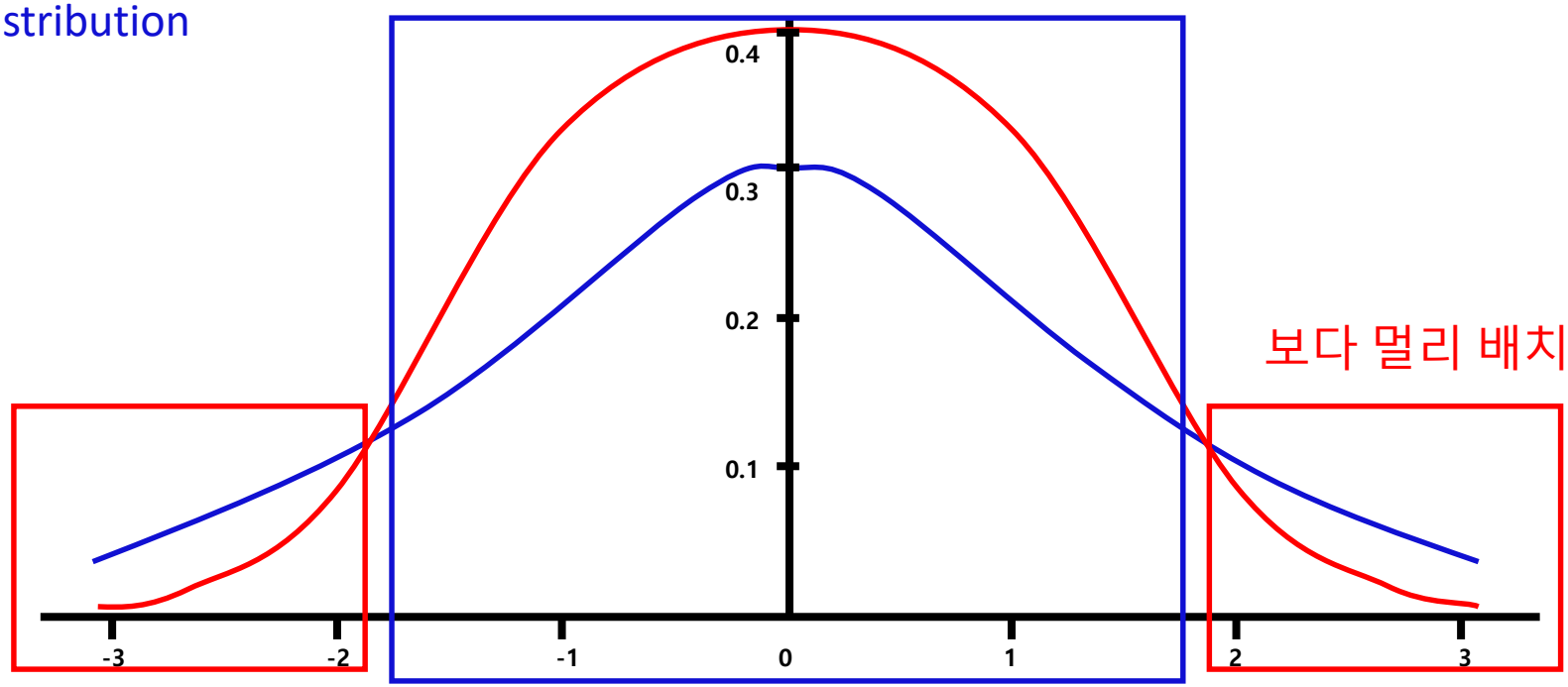


가우시안 분포의 특성인 꼬리부분이 두텁지 않아 i번째 개체에서 적당히 떨어져 있는 이웃 j와 많이 떨어져 있는 이웃 k가 선택될 확률이 크게 차이가 나지 않는 문제

❖ t-SNE

Normal distribution
t distribution

보다 가깝게 배치시킴



보다 멀리 배치시킴

가우스 분포와 자유도 1의 t분포의 비교

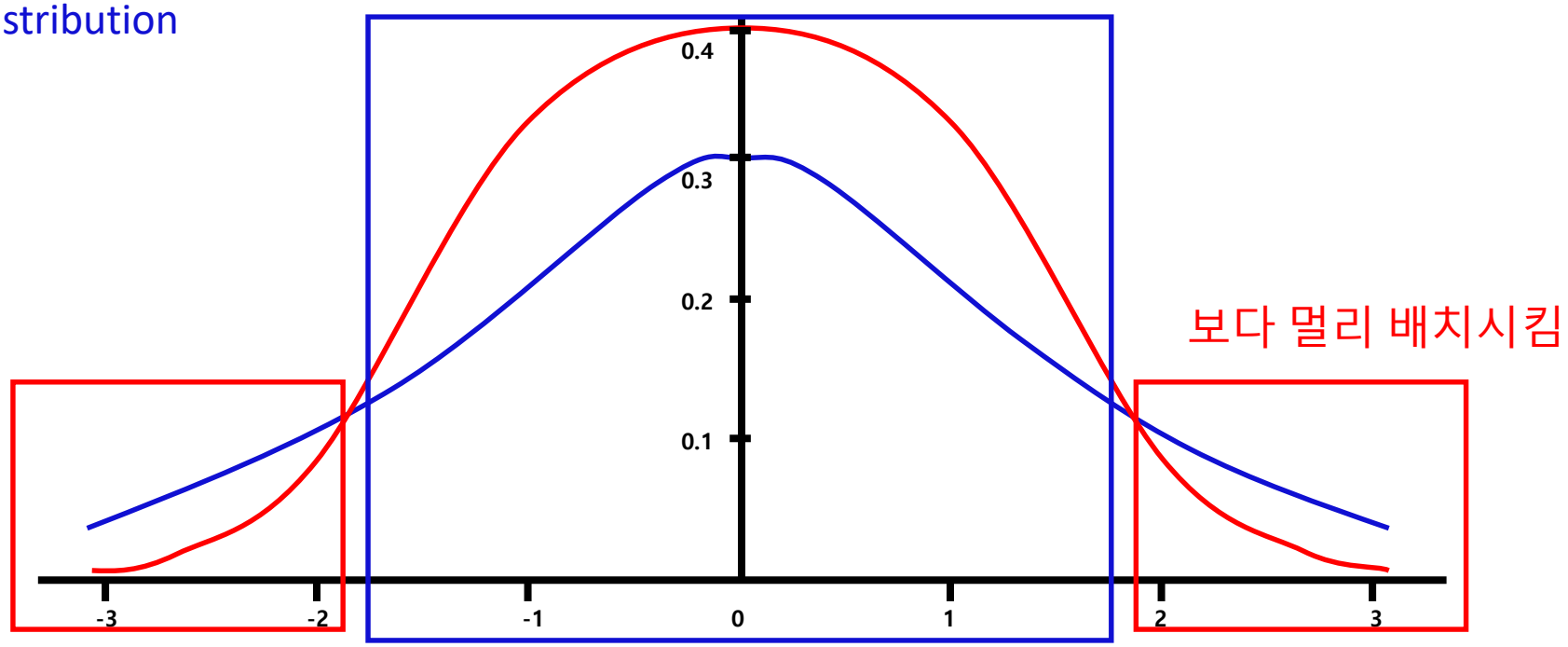
저차원에서만 t-분포를 사용하여 꼬리부분의 차이가 보이도록 사용

1. 가까운점의 q값을 실제보다 저평가 -> 더 가까워지려 함
2. 먼 점의 q값을 실제보다 고평가 -> 더 멀어짐

❖ t-SNE

Normal distribution
t distribution

보다 가깝게 배치시킴



보다 멀리 배치시킴

가우스 분포와 자유도 1의 t분포의 비교

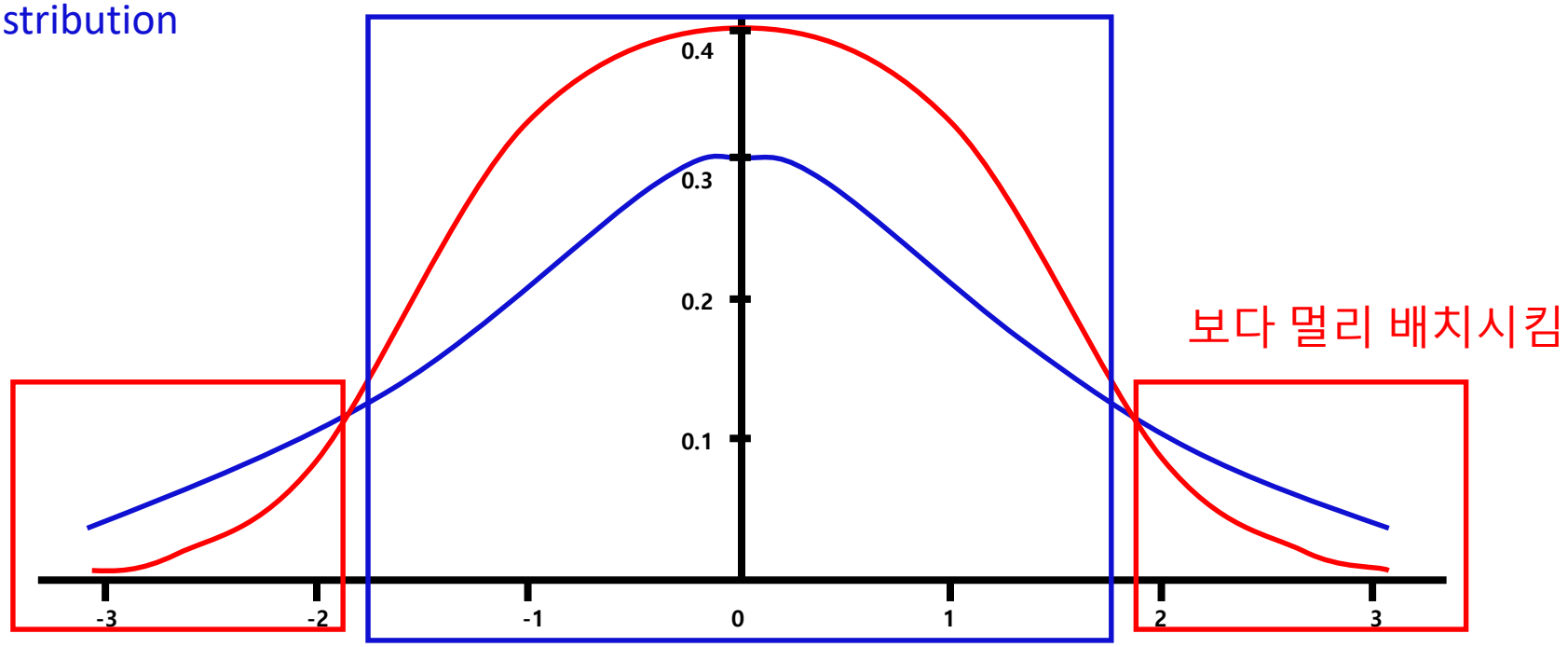
$|y_i - y_j|^2$ 가 작을수록 큰값을 보이도록 역수를 취하여 $(|y_i - y_j|^2)^{-1}$ 을 유사도로 사용
이때 0에 가까운 값의 역수는 무한대로 계산상 문제가 생기기에 $(1+|y_i - y_j|^2)^{-1}$ 을 사용.

❖ t-SNE

Normal distribution
t distribution

보다 가깝게 배치시킴

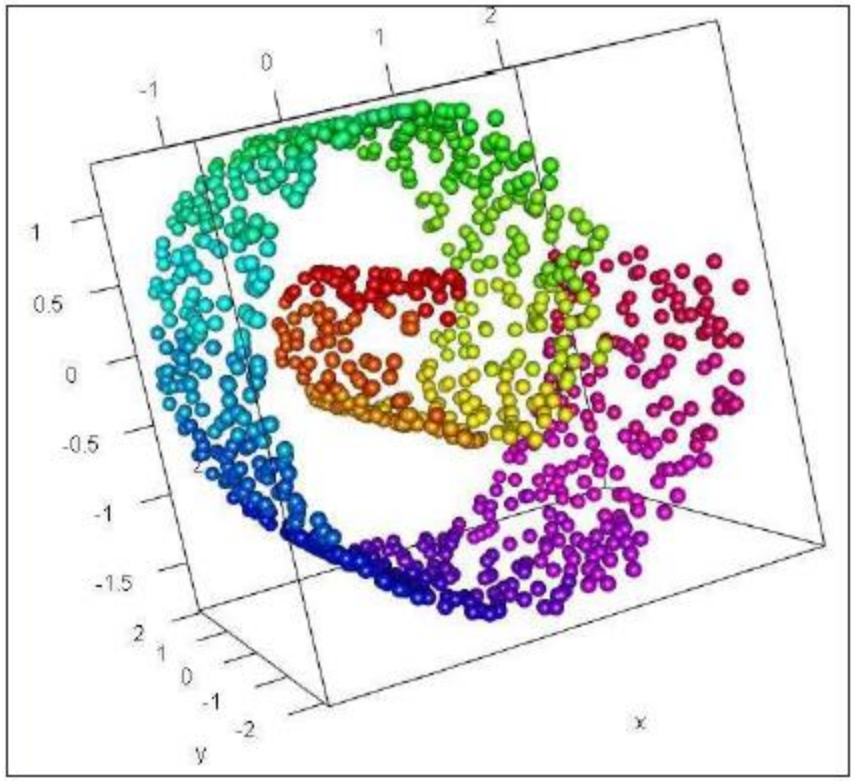
보다 멀리 배치시킴



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

$$Minimize\ C\ by\ using \frac{\delta C}{\delta y_i} = 4 \sum_j (y_j - y_i)(p_{ij} - q_{ij})\ (1 + \|y_i - y_j\|^2)^{-1}$$

❖ Visual목적의 t-SNE의 장점



1. 데이터 점 사이의 거리가 큰 것을 유사하지 않은 점으로 모델링
2. 데이터 점 사이의 거리가 작은 것을 유사한 점으로 모델링
3. Cost function의 최적화가 용이함
4. 다양체(Manifold)와 같은 구조를 유지한 체 가시화 가능