2019-06-19

CVPR2019 paper review

Domain Adaptation using Stochastic Neighborhood Embedding

Introduction

- 논문 고려 문제:
  - Pretrained model $\mathcal{M}_{\mathcal{D}^s}$
  - User has smaller dataset
  - Label spaces are the same

$$\mathcal{D}^s = \{(x_i^s, y_i^s)\}_{i=1}^{N^s}$$
$$\mathcal{D}^t = \{(x_j^t, y_j^t)\}_{j=1}^{N^t} \qquad N^t << N^s$$
$$\text{i.e., } \{y^s, y^t\} \in [0, 1, \ldots c-1]$$

  - The user should be able to repurpose the model M_{Ds} to work with dataset D^t

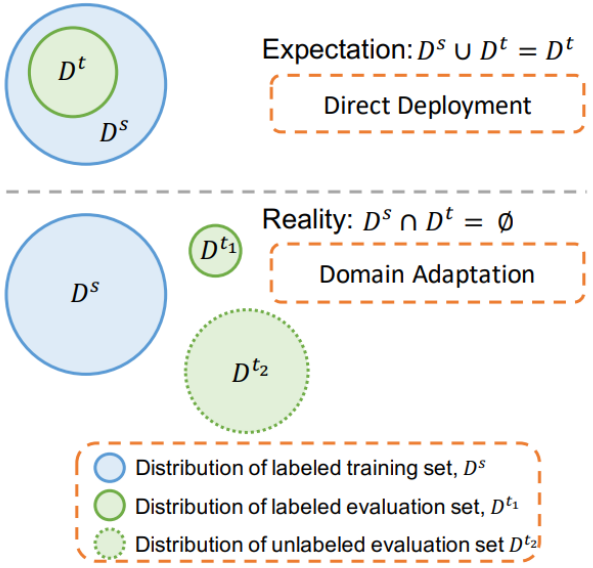


Figure 1. Domain adaptation in the true data space: Expectation vs. Reality.

- 이상적인 상황이 아니기 때문에 Domain adaptation 필요

Introduction

- Domain adaptation:
  - 다른 domain의 지식이 target domain에서 잘 작용하기 위해 재활용 되는 것
- Domain generalization:
  - 만약 위의 솔루션이 두 도메인 모두에서 동일하게 잘 작동할 수 있는 경우, 이를 domain generalization이라 한다
- ImageNet data
  - Popularizes the idea of repurposing networks trained on this dataset to be used as generic feature extractors
- Domain adaptation은 source와 target의 label이 동일하다고 생각

Introduction

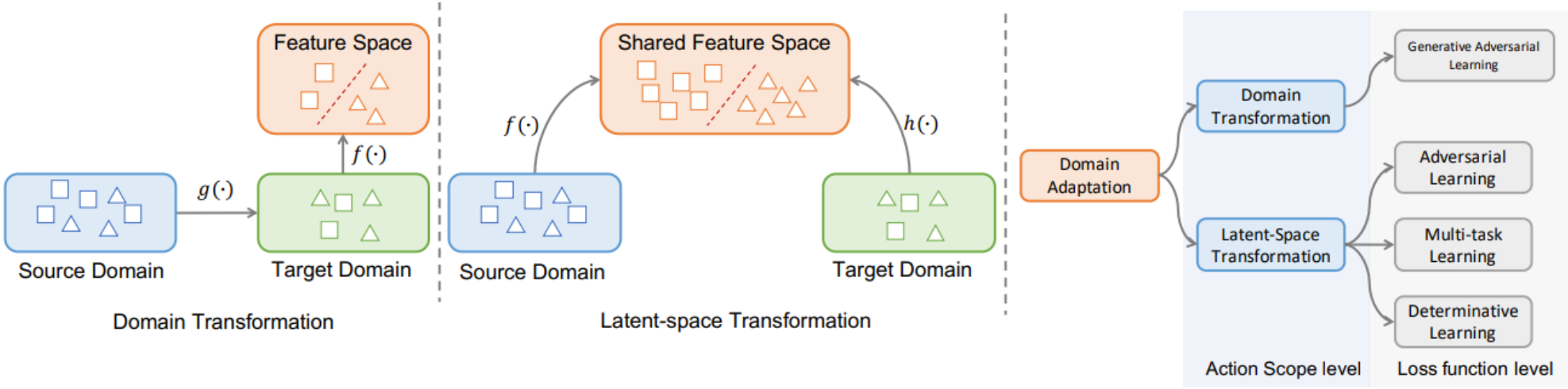


Figure 2. Various types of domain adaptation.

- Domain Transformation: pixelDA
- Latent-space Transformation: DANN

Introduction

- To create this embedding space, we use a strategy that is very similar to the popular stochastic neighborhood embedding technique (SNE)
  - SNE 방식과 유사
- To modifiy SNE for domain adaptation, we use a novel modified-Hausdorff distance metric in a min-max formulation.
- d-SNE minimizes the distance between the samples from Ds and Dt so as to maximize the margin of inter-class distance for discrimination and minimize the intra-class distance from both domains to achieve domain-invariance.
  - class 간의 격차는 최대화 시키고 class 내부의 distance는 최소화 시킨다
- End to end로 학습 가능

Introduction

- Extensive experimental results in different scenarios indicate that our algorithm is robust and outperforms the state of-the-art algorithms with only a few labeled data samples.
- The key contributions in this paper include the following:
  - SNE와 large margin nearest neighborhood를 사용해 latent space 학습 (추가자료 참고)
  - Use of a modified-Hausdorff distance and a novel min-max formulation in this space to help few-shot supervised learning.
  - Semi-supervised learning으로 확장

  - SNE는 유도에, LMNN은 전략으로, modified-Hausdorff distance는 실질적 응용으로 사용된 것 같다

Related Works

- Latent-Space Transformation: 크게 두가지 갈래로 나뉨
  - Domain adversarial learning (DAL): DANN
  - Domain multi-task learning(DMTL): Multi task Learning은 입력 데이터가 하나로 정해져 있을 때 이를 기반으로 여러 task들에 대한 예측 결과들을 동시에 출력할 수 있도록 하는 방법을 연구하는 주제다. 대상이 되는 task들을 동시에 커버할 수 있는 feature representation을 찾는다는 측면에서 Taskonomy 방법과 공통점이 일부 존재하나, Taskonomy 방법은 두 task들 간의 관계를 명시적으로 모델링한다는 측면에서 차이가 있다.

d-SNE

- Notation

$$d(x_i^s, x_j^t) = \|\Phi_{\mathcal{D}^s}(x_i^s) - \Phi_{\mathcal{D}^t}(x_j^t)\|_2^2, \quad (1)$$

$$p_{ij} = \frac{\exp(-d(x_i^s, x_j^t))}{\sum_{x \in \mathcal{D}^s} \exp(-d(x, x_j^t))}. \quad (2)$$

$$p_j = \frac{\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j^t))}{\sum_{x \in \mathcal{D}^s} \exp(-d(x, x_j^t))} = \sum_{i=0}^{N_k^s} p_{ij}, \quad (3)$$

Consider that $y_j^t = k$ and that $\mathcal{D}_k^s = \{\forall x_l^s | y_l^s = k\}$.

$$N_k^s = |\mathcal{D}_k^s|$$

same-class set $\mathcal{D}_k^s$ and a different-class set, $\mathcal{D}_{k'}^s$.

- Latent space에서의 거리 정의, pi는 S와 T에서 latent space로 mapping 시켜주는 각각의 NN
- P_{ij}는 x^t와 x^s 가 label이 같을 확률(S와 T모두 label이 존재)
- P_j는 x^t_j가 옳게 분류될 확률 (x^t_j 은 target domain의 data중 하나의 data)
- same set과 different class set 표기

d-SNE

추가 자료 – stochastic neighbor embedding (SNE)

- SNE란 고차원의 원공간에 존재하는 데이터 x의 이웃 간의 거리를 최대한 보존하는 저차원의 y를 학습하는 방법론
- Stochastic이란 이름이 붙은 이유는 거리 정보를 확률적으로 나타내기 때문

$$p_{j|i} = \frac{e^{-\frac{|x_i - x_j|^2}{2\sigma_i^2}}}{\sum_k e^{-\frac{|x_i - x_k|^2}{2\sigma_i^2}}}$$

$$q_{j|i} = \frac{e^{-|y_i - y_j|^2}}{\sum_k e^{-|y_i - y_k|^2}}$$

$$Cost = \sum_i KL(P_i||Q_i)$$

$$= \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- j|i는 i번째 개체 x_i가 주어졌을 때 j번째 개체 x_j가 선택될 확률
- SNE의 목적은 p와 q의 분포 차이가 최대한 작게끔 하고자 하는 것
- 측정 지표로는 KL divergence를 사용

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}, \quad q_{ij} = \frac{q_{j|i} + q_{i|j}}{2}$$

- 근데 시그마 i를 고정값으로 두고, pij와 pji의 값을 동일하게 하기 위해 왼쪽과 같은 식을 사용(계산속도 향상)

d-SNE

- Loss

$$\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j^t)) \quad + \quad \sum_{x \in \mathcal{D}_{k'}^s} \exp(-d(x, x_j^t)).$$

$$\sum_{x_j \in \mathcal{D}^t} \frac{1}{p_j} = \sum_{x_j \in \mathcal{D}^t} \left(\frac{\sum_{x \in \mathcal{D}_{k'}^s} \exp(-d(x, x_j))}{\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j))}, \text{ for } k = y_j \right). \tag{4}$$

$$\mathcal{L} = \log \left(\frac{\sum_{x \in \mathcal{D}_{k'}^s} \exp(-d(x, x_j))}{\sum_{x \in \mathcal{D}_k^s} \exp(-d(x, x_j))}, \text{ for } k = y_j \right). \tag{5}$$

- 앞의 식 (3)에서 분모는 왼쪽과 같이 나누어 짐
- 우리는 p_j를 최대화 하고 싶어하기 때문에 $1/p_j$의 log-likelihood를 최소화 하려고 한다
- 이것은 잠재적인 공간에서 클래스 간 거리에 대한 클래스 내 거리의 비율을 최소화하는 것과 같다

d-SNE

- Loss

$$\tilde{\mathcal{L}} = \sup_{x \in \mathcal{D}_k^s} \{a | a \in d(x, x_j)\} - \inf_{x \in \mathcal{D}_{k'}^s} \{b | b \in d(x, x_j)\},$$
$$\text{for } k = y_j. \quad (6)$$

- Relaxation
- exp term이 likelihood term(식 (5))에 들어가는데 이건 scaling issue를 야기하고 SGD에 좋지 않은 효과를 가져온다
- 이를 해결하기 위해 (6)의 loss를 정의하는데 결국 global distance를 optimize하는 것 대신에 same class인 경우 largest distance를 minimize하고 different class인 경우 smallest distance를 maximize하게 된다
- 이는 modified-Hausdorffian distance 개념 이다

- 이렇게 되면 사실 SNE를 사용하지 않는 것 같은데 내 생각에는 최종 loss의 유도 과정에 사용된 것이 아닌가 생각된다

d-SNE

추가 자료 – Hausdorff distance

- Hausdorff 거리는 두 집합의 비 일치도(mismatch) 를 측정하는 비선형 척도로 유클리디언 거리, 맨하탄 거리 등의 일반적인 거리 함수를 이용하는 최대 최소(maxmin) 기법이다.
- 두 점 *a*와 *b*사이의 거리 *d(a, b)*를 다음과 같 이 정의하자

$$d(a,b) = \|a-b\|$$

두 개의 유한집합 A = {a_1, a_2, …, a_m} 와 B = {b_1, b_2, …, b_3} 가 주어졌을 때, A의 한 점 *a* 에서 B까지의 Hausdorff 거리 *d(a, B)*는 다음과 같이 정의된다.

$$d(a,B) = \min_{b\in B} \mathrm{d}(a,b) = \min_{b\in B} \|a-b\| \quad (2)$$

집합 A로부터 집합 B까지의 Hausdorff 거리 *h(A, B)*는 다음과 같이 정의된다.

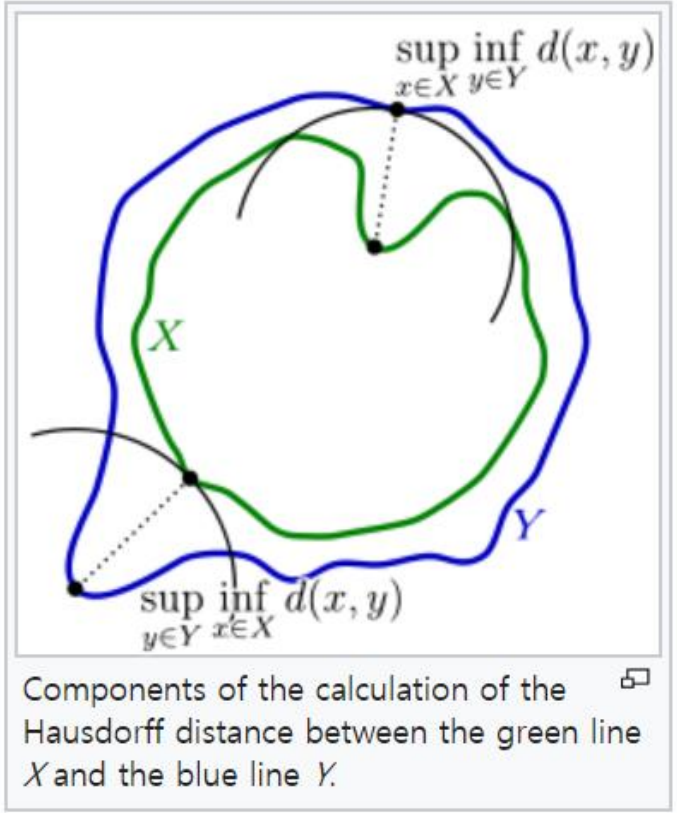
$$\begin{aligned} h(A,B) &= \max_{a\in A} d(a,B) \\ &= \max_{a\in A} \min_{b\in B} \mathrm{d}(a,b) \\ &= \max_{a\in A} \min_{b\in B} \|a-b\| \end{aligned} \quad (3)$$

유사한 방법으로, 집합 B로부터 집합 A까지의 Hausdorff 거리 *h(B , A)*는 다음과 같이 정의된다.

$$\begin{aligned} h(B,A) &= \max_{a\in B} d(b,A) \\ &= \max_{b\in B} \min_{a\in A} d(\mathrm{b},a) \\ &= \max_{b\in B} \min_{a\in A} \|a-b\| \end{aligned} \quad (4)$$

집합 A와 집합 B사이의 Hausdorff 거리 *H(A, B)*는 앞에서 구한 *h(A, B)*와 *h(B , A)*를 이용하여 다음과 같 이 정의된다.

$$H(A,B) = \max(h(A,B), h(B,A)) \quad (5)$$



Components of the calculation of the Hausdorff distance between the green line *X* and the blue line *Y*.

- Ex) X 집합 중 모든 Y의 point와 가장 먼 point를 잡고 그 포인트에서 가장 가까운 Y까지의 거리를 구하면 되는 거 같다

d-SNE

추가 자료 – Modified Hausdorff distance

- Hausdorff 거리는 이상치에 매우 민감하다. – 이를 해결하기 위한 변형 중 Modified Hausdorff distance 가 있다

Dubuisson 등[3]이 제시한, 집합 A로부터 집합 B까지의 Modified Hausdorff 거리 $h_m(A, B)$는, 집합 A의 각 점으로 부터 집합 B까지의 Hausdorff 거리의 평균값으로 다음과 같이 정의되며, 이상치의 영향을 감소시키는 효과를 갖 는다.

$$
\begin{aligned}
h_M(A,B) &= \frac{1}{|A|}\sum_{a\in A} d(a,B) \\
&= \frac{1}{|A|}\sum_{a\in A}\min_{b\in B} \mathrm{d}(a,b) \\
&= \frac{1}{A}\sum_{a\in A}\min_{b\in B}\|a-b\|
\end{aligned}
\tag{7}
$$

d-SNE

End to end Learning

- 앞에서 두가지 NN을 가진다고 했다(S, T 각각) 하지만 실제적으로, 소스와 대상 도메인의 입력 데이터가 동일한 차원을 가진다면 두 도메인 간에 단일 네트워크를 공유할 수 있다

$$\underset{w_s, w_d}{\operatorname{argmin}} \tilde{\mathcal{L}} + \alpha \mathcal{L}_{\text{ce}}^{s} + \beta \mathcal{L}_{\text{ce}}^{t} \qquad (7)$$

- 최종 Loss

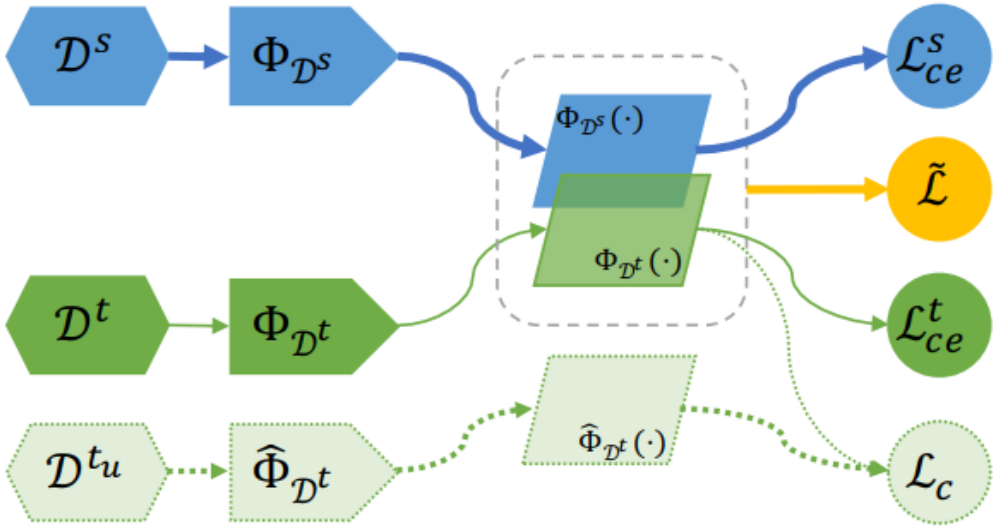


Figure 3. The learning setup. The segment in the bottom in lighter shade and dotted lines is the semi-supervised extension.

- Cross entropy를 각각의 도메인에 적응시켜서 regularization에 사용

d-SNE

- Semi-supervised Extension

- Unlabeled data가 performance를 향상 시킬 수 있다
- 새로운 network 파이 D_u 는 Unlabeled data를 latent space에 embedding 시키는 network 이다
- Mean-Teacher network technique을 사용해 학습 시킨다
- 우리는 consistency loss L_c를 사용한다
  - (embedding 사이의 L_2 error 사용)

$\mathcal{L}_c$ across $\hat{\Phi}_{\mathcal{D}_u^t}$ and $\Phi_{\mathcal{D}^t}$, by taking an L_2 error

- 학습 과정
  - 먼저 앞의 과정으로 unsupervised set 없이 학습을 시킴
  - Unlabeled data로 Mean-Teacher model을 학습 시킴(unlabeled의 network 초기화는 T의 weight로 함)
  - 양쪽 network의 input은 augmentation 등을 활용해서 만듦, 같은 샘플데이터 에서 생성
  - Unlabeled data의 network는 consistency loss만 사용해서 backprop으로 update, Target의 weight은 Unlabeled 의 network의 weights을 지수가중 평균해서 update한다

d-SNE

추가 자료 – Large Margin Nearest Neighbors (LMNN)

- Distance Metric Learning: 데이터들의 분포 등을 고려하여 '거리'를 새로 정의하는 분야가 존재하는데 이를 일컬어 Distance Metric Learning이라 한다.
- 즉 input data space에서 data들에 가장 적합한 형태의 어떤 metric을 learning하는 알고리즘이다.
  - Similar한 point끼리는 더 가까운 거리로 판단하게 하고, dissimilar한 point는 더 먼 거리로 판단하게 하는 metric을 학습한다

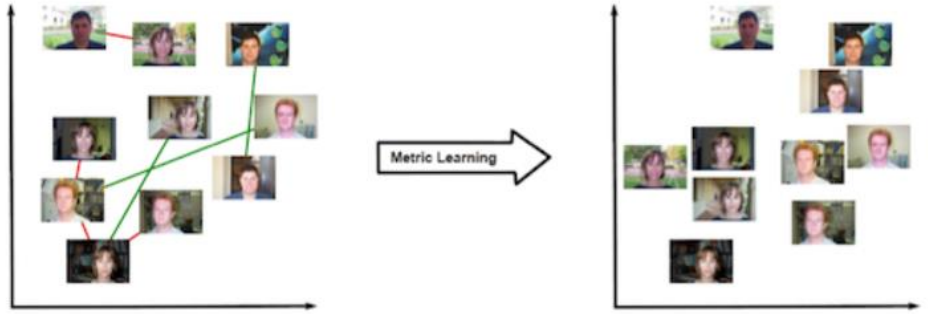
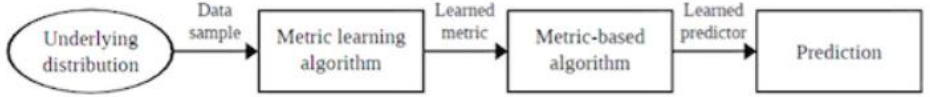


Figure 1: Illustration of metric learning applied to a face recognition task. For simplicity, images are represented as points in 2 dimensions. Pairwise constraints, shown in the left pane, are composed of images representing the same person (must-link, shown in green) or different persons (cannot-link, shown in red). We wish to adapt the metric so that there are fewer constraint violations (right pane). Images are taken from the Caltech Faces dataset.[8]



d-SNE

추가 자료 – Large Margin Nearest Neighbors (LMNN)

- Mahalanobis Distance Metric: 이 distance metric은 Euclidean distance metric이 data set의 correlation을 하나도 고려하지 않은 문제점을 해결할 수 있고, 또한 scale-invariant한 특성을 가지고 있다. 이 metric은 아래와 같이 정의된다.

$$d(p,q) = \sqrt{(\vec{p}-\vec{q})^{\top}\Omega(\vec{p}-\vec{q})}$$

- 이 때 Ω는 positive semidefinite matrix이다. 정확하게는 Ω가 covariance matrix인 metric이다.
  - 따라서 이 metric이 data set의 correlation을 포함하여 거리를 표현할 수 있는 것이다.
  - 하지만 실제 분포를 알 수 없는 임의의 데이터들에 대해서 올바른 covariance matric를 계산하는 것은 매우 어렵다.
  - 따라서 Mahalanobis metric의 Ω를 learning하는 method들도 존재하는데, 대표적으로 LMNN classification이 있다.

d-SNE

추가 자료 – Large Margin Nearest Neighbors (LMNN)

- LMMN논문에서는 Mahalanobis Metric으 제곱근 형태가 아니라 아래와 같은 꼴로 표현했다

$$D(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^\top \mathbf{M}(\vec{x}_i - \vec{x}_j) \qquad D(\vec{x}_i, \vec{x}_j) = ||L(\vec{x}_i - \vec{x}_j)||^2.$$

- 이 논문의 핵심 아이디어는, 위에서 표현한 Metric을 평가하는 Cost function을 design하고 이 function을 minimize 시키는 Metric을 찾아내는 것이다.

$$\varepsilon(\mathbf{L}) = \sum_{ij} \eta_{ij} ||L(\vec{x}_i - \vec{x}_j)||^2 + c \sum_{ijl} \eta_{ij}(1 - y_{il}) h[1 + ||L(\vec{x}_i - \vec{x}_j)||^2 - ||L(\vec{x}_i - \vec{x}_l)||^2]$$

1. $(\vec{x}_i, y_i)_{i=1}^n$: training set을 의미한다. 벡터 x는 input data를, scalar y는 label을 의미한다. (binary class가 아니어도 상관없다.)
2. η_{ij}: $\vec{x}_j$가 $\vec{x}_i$의 target neighbor인가 아닌가를 나타내는 binary variable. 맨 처음 learning할 때 고정되는 값이며 알고리듬이 돌아가는 동안 변하지 않는 값이다.
3. y_{ij}: label y_i와 y_j가 서로 일치하는가 하지 않는가를 나타내는 binary variable이다. 역시 변하지 않는다.
4. h(x): hinge function으로, 간단하게 표현하면 $h(x) = max(0, x)$이다. 즉, 0보다 작으면 0, 아니면 원래 값을 취하는 함수이다.
5. c: 0보다 큰 임의의 상수로, 끌어당기는 term과 밀어내는 term사이의 trade-off를 조정한다. 보통 cross validation으로 결정한다.
6. Target neighbor: 임의의 x_i와 같은 label을 가진 데이터들 중에서 가장 가까운 k개의 데이터들을 의미하며 k는 사용자가 세팅할 수 있다

- 옆의 loss에서 앞의 항은 같은 label 끼리 서로 끌어오는 term이고, 뒷 항은 서로 다른 label끼리 밀어내는 term이다.
- 이유는, 먼저 에타ij는 i와 j가 서로 target data 일 때만 1이므로 연산이 target neighbor 들에 대해서만 진행된다. 따라서 앞의 항을 minimization하는 것은 같은 label 끼리 거리를 가깝게 한다는 의미이고 뒷 항은 i,j외의 다른 label을 가지는 l들에 대해서는 최대한 거리를 멀어지도록 하는 항이다.

- 내 생각에는 그냥 같은 class의 거리는 가깝게, 다른 class의 거리는 멀게 하는 개념이 LMMN concept 인 것 같다

Reference

- http://research.sualab.com/review/2018/08/14/taskonomy-task-transfer-learning.html
  - Taskonomy: Disentangling Task Transfer Learning 리뷰
- http://sanghyukchun.github.io/38/
  - LMNN 설명
- https://ratsgo.github.io/machine%20learning/2017/04/28/tSNE/
  - SNE 설명
- http://www.ksie.ne.kr/journal/article.php?code=31920
  - Haudorff distance 설명
- https://beguru.tistory.com/50
  - Haudorff distance 설명
- https://m.blog.naver.com/PostView.nhn?blogId=pjk871387&logNo=220712856778&proxyReferer=https%3A%2F%2Fwww.google.com%2F
  - Haudorff distance 설명
- https://www.edwith.org/deeplearningai3/lecture/34893/
  - End to end learning