# Metric Learning 개념과 Deep Metric Learning

## 1. 거리함수 (Distance Function)

통계, 데이터 마이닝, 머신러닝 등에서는 데이터 간의 유사도를 수치화하기 위해 일반적으로 거리함수를 이용한다. 가장 대표적은 거리 함수로는 Euclidean distance가 있으며, 두 데이터  x1과 x2에 대해 아래 식과 같이 정의된다.

$$d_E(X_1, X_2) = \sqrt{\sum_{i=1}^{d}(X_{1,i} - X_{2,i})^2}$$
$$x_1 \in \Re^d, x_2 \in \Re^d$$

또 다른 대표적인 거리함수로는 각 축 방향으로의 분산까지 고려한 Mahalanobis distance가 있으며, 두 점의 거리는 아래 식과 같이 정의된다

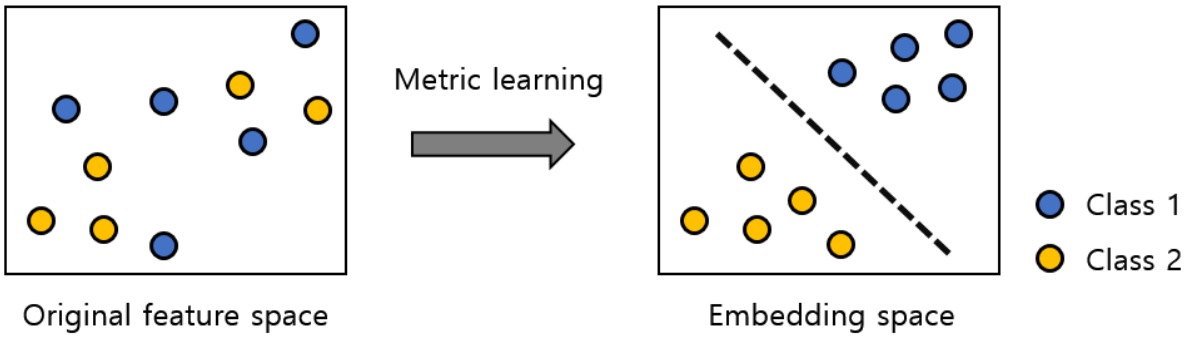
$$d_M(X_1, X_2) = \sqrt{(X_1 - X_2)^T S^{-1}(X_1 - X_2)}$$
$$x_1 \in \Re^d, x_2 \in \Re^d$$

식에서 $S$는 공분산 행렬이다. 이외에도 cosine similarity나 Wasserstein 거리와 같은 다양한 거리 함수가 있다.

이런 다양한 거리 함수 중에서 모든 데이터에 대해 적합한 거리함수는 현실적으로 존재하지 않는다. 이러한 이유로 데이터에 적합한 거리 함수를 머신러닝 알고리즘으로 직접 만드는 metric learning이 활발히 연구되고 있다.

## 2. Deep Metric Learning

데이터에 적합한 거리 함수라는 표현을 머신러닝 관점에서 다시 말하면, 데이터의 각 목표값에 대해 데이터를 구분하기 쉽게 만들어 주는 거리함수를 의미한다. 아래 그림에서 metric learning의 목적을 시각적으로 보여준다. 기존의 feature로는 분류가 쉽지 않았던 데이터에 대해 데이터를 class label 별로 잘 구분할 수 있게 만드는 metric을 학습함으로써 분류 모델을 만드는 문제가 매우 단순해 졌다.




Metric learning을 통해 학습된 새로운 거리함수는 일반적으로 embedding 함수 f를 통해 변환된 새로운 데이터 표현에 대한 Euclidean distance로 정의된다. 예를 들어, metric learning을 통해 학습된 거리 함수를 $f(x:\theta)$라고 할때, 두 데이터 x1과 x2에 대한 거리함수는 아래 식과 같이 정의된다.

$$d_\theta(X_1, X_2) = ||f(X_1) - f(X_2)||_2^2$$

따라서 **metric learning 문제의 목적은 데이터를 각 목표값에 따라 잘 구분되도록 변환하는 embedding 함수 $f$를 학습**하는 것이 된다. 이때, $f$가 단순한 선형 변환이 아니라, deep neural network일 경우에 앞에 deep을 붙여 deep metric learning이라고 한다. 아래에서는 deep metric learning의 대표적인 두가지 방법인 contrastive embedding과 triplet embedding, 그리고 deep metric learning을 회귀 문제로 확장한 log-ratio embedding에 대해 소개한다.

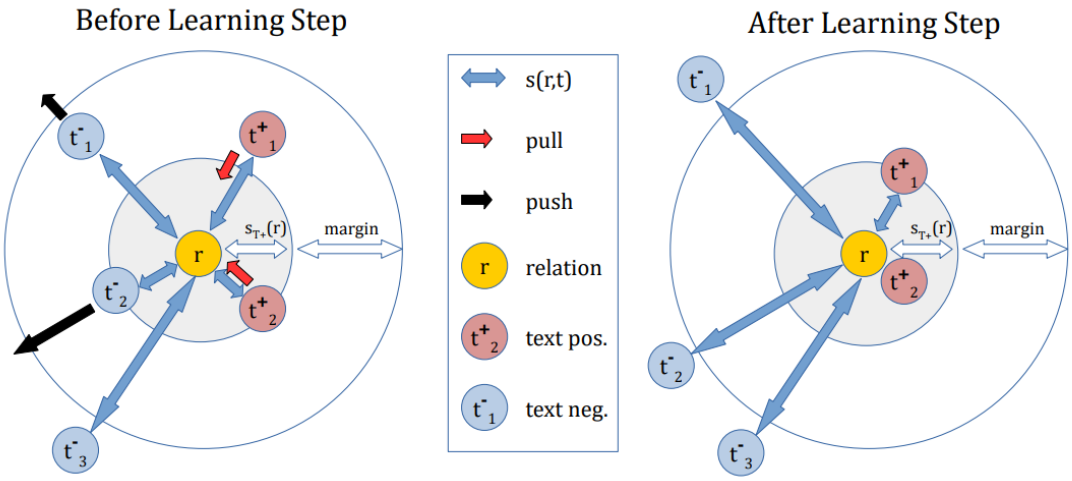
## 3. Contrastive Embedding

Contrastive embedding은 이진 분류에 이용될 수 있는 metric learning이다. 각각의 tuple $(X_i, X_j, y_{ij})$에 대해 contrastive loss는 다음과 같이 정의된다.

$$L_C = \frac{1}{N}\sum_{i=1}^{N/2}\sum_{j=1}^{N/2} y_{ij} d(X_i, X_j) + (1 - y_{ij})\{max(0, \alpha - d(X_i, X_j))\}$$

위 식 contrastive loss에서는 embedding network $f$에 대해 $d(x_i, x_j) = ||f(x_i) - f(x_j)||_2^2$로 정의되며, $y_{ij}$는 $X_i$와 $X_j$가 같은 class이면 1이고, 아니면 0이다. $y_{ij}$가 1인 경우, 거리가 0이면 $L_c$는 0이 되고 거리가 d라면 $L_c$는 d가 된다. $y_{ij}$가 0인 경우, 거리가 0이면 $L_c$는 $\alpha$이 되고 거리가 d라면 $L_c$는 ($\alpha$-d or 0)이 된다. ⇒ 즉, $X_i$와 $X_j$의 현재거리 간격에서 $\alpha$거리 간격이 되도록 하기에 $\alpha$ < d 라면 더이상 멀어지지 않도록 $L_c$는 0이고, $\alpha$ > d라면 $\alpha$거리가 되도록 $\alpha$ - d 만큼이 $L_c$가 된다.

즉, 간단히 말하자면 같은 class의 경우 거리가 0이 되도록 다른 class의 경우 거리가 $\alpha$(margin)이 되도록 학습한다.




이로 인해 **contrastive Loss 단점**은 margin parameter인 $\alpha$를 설정하기가 어렵고, 크게 차이나는 Neg Sample이나, 적게 차이나는 Neg Sample이 동일한 Margin($\alpha$) 차이가 난다는 것이다.

학습은 contrastive embedding의 $L_c$를 최소화하도록 $f$의 model parameter를 학습시킴으로써 데이터를 잘 구분할 수 있는 새로운 embedding을 만들어낸다.

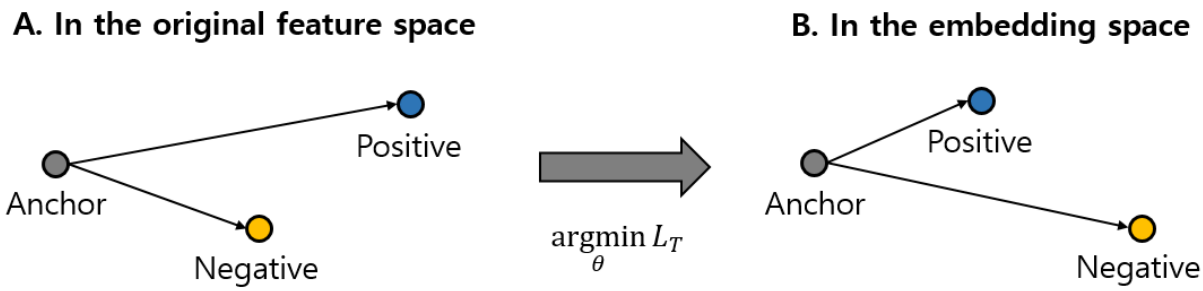
## 4. Triplet Embedding

Triplet embedding은 다중 분류 (multi-class classification)에 이용되는 metric learning이다. embedding을 위한 triplet loss는 주어진 데이터셋에서 선택된 데이터인 anchor, 그리고 anchor와 동일한 class label을 갖는 positive sample, 다른 class label을 갖는 negative sample로 아래 식과 같이 정의된다.

$$L_T = \frac{3}{2N}\sum_{i=1}^{M/3} max(0, d(x_i, x_{i,p}) - d(x_i, x_{i,n}) + \alpha)$$

이 식에서 $x_{i,p}$와 $x_{i,n}$은 각각 현재 선택된 anchor $x_i$의 positive sample과 negative sample이다. 위 식의 triplet loss를 $f$의 model parameter $\theta$에 대해 최소화함으로써 embedding space에서 anchor와 positive sample거리는 가까워지고, negative sample과는 거리가 멀어진다.

아래 그림은 이러한 triplet embedding의 개념을 보여준다. 일반적으로 triplet embedding에서 positive sample과 nagative sample은 랜덤 샘플링 기반으로 추출되며, 이외에도 embedding 성능을 향상시키기 위한 여러 샘플링방법이 제안되었다.

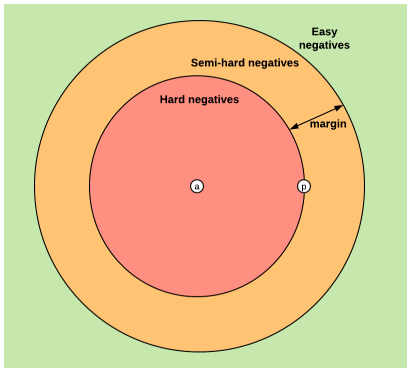



예를 들어 $d(x_i, x_{i,p}) = 0.5, d(x_i, x_{i,n}) = 0.7$ 인 경우 $max(0, d(x_i, x_{i,p}) - d(x_i, x_{i,n}) + \alpha = \alpha - 0.2$ 가 된다.
$d(x_i, x_{i,p}) = 0.7, d(x_i, x_{i,n}) = 0.5$ 인 경우 $max(0, d(x_i, x_{i,p}) - d(x_i, x_{i,n}) + \alpha = \alpha + 0.2$ 가 된다.

**triplet loss의 장점**으로는 anchor과 절대적 거리 $\alpha$를 유지하는 contrastive loss와는 달리, tiplet loss는 상대적 거리 $\alpha$ 차이를 학습한다.

triplet loss 세부 종류는 anchor와 neg, pos sample의 구성을 어떻게 하느냐에 따라 3가지로 나뉘어진다.

1. Easy Triplet : $D_{an}^2 + \alpha < D_{ap}^2$인 neg sample 사용
   - Easy Neg 샘플은 이미 $D_{an}^2 + \alpha$보다 더 떨어져있는 샘플
   - Batch를 구성(찾기)하기는 쉬우나, Gradient가 낮아 학습에서 덜 중요하다. 즉, anchor와 완전히 다른 이미지는 찾기 쉬우나, 유사하지만 다른 class를 가지는 샘플이 학습되어야 성능이 좋아짐…
2. Hard Triplet : $D_{an}^2 < D_{ap}^2$ 인 Neg Sample을 사용
   - Hard Neg 샘플은 Pos 샘플보다 anchor과 더 가까운 샘플
   - 학습에 매우 주요하며, Batch를 구성하기(찾기)가 매우 어려움
3. Semi-hard Triplet : $D_{an}^2 < D_{ap}^2 < D_{an}^2 + \alpha$인 neg sample 사용
   - Semi-hard neg 샘플은 margin에 위치한 neg 샘플
   - 학습에 매우 주요하며, Hard 샘플보다 Batch를 구성하기(찾기)가 상대적으로 쉬움




## 5. Log-Ratio Embedding

기존의 contrative embedding과 triplet embedding은 모두 분류 문제를 위해 고안되었다. Log-ratio embedding은 분류 문제를 일반화하여 회귀 (regression) 문제에도 metric learning을 적용하기 위해 제안되었다. 즉, 기존의 이산 목표 변수가 아닌 연속 목표 변수에 대해 metric learning을 확장한 것이다. 이러한 log-ratio embedding을 위한 log-ratio loss는 아래 식과 같이 정의된다.

$$L_{LR} = \frac{1}{N}\sum_{i=1}^{N}\{log\frac{d(x_i, x_{i,p})}{d(x_i, x_{i,n})} - log\frac{d(y_i, y_{i,p})}{d(y_i, y_{i,n})}\}^2$$

Log-ratio embedding에서는 위 식의의 log-ratio loss를 최소화함으로써 embedding 공간 상에서의 데이터 간의 거리 비율과 목표 변수 간의 거리 비율이 일치하도록 $f$의 model parameter를 최적화한다.

### 참고자료


- https://untitledtblog.tistory.com/164
- https://en.wikipedia.org/wiki/Mahalanobis_distance
- https://ko.wikipedia.org/wiki/%EC%9C%A0%ED%81%B4%EB%A6%AC%EB%93%9C_%EA%B1%B0%EB%A6%AC
- https://kmhana.tistory.com/17