# Two Loss Functions in Deep Metric Learning

Xinyao Li

Department of Statistics and Data Science, SUSTech

July 20, 2023

# Deep Metric Learning (DML)

## Core Ideas

- Encouraging similarity between semantically transformed versions of an input.
- Training a network to predict whether two inputs are from the same class (or not) by making their embedding close (or far from each other).
- Positive pairs: the variants of the inputs;
  Negatives: the samples we wish to make dissimilar.

# Dimensionality Reduction by Learning an Invariant Mapping (DrLIM)

## Main Reference

R Hadsell, S Chopra, Y LeCun, Dimensionality Reduction by Learning an Invariant Mapping.

## Background

Most existing dimensionality reduction techniques (e.g. **Locally Linear Embedding**) have some shortcomings:

- Do not produce a function (or a mapping) from input to manifold that can be applied to new points whose relationship to the training points is unknown.
- Presuppose the existence of a meaningful (and computable) distance metric in the input space.
- Tend to cluster points in output space, sometimes densely enough to be considered degenerate solutions.

## Four Essential Characteristics of DrLIM

- Only needs neighborhood relationships between training samples coming from **prior knowledge** or **manual labeling**.
- Learns functions that are invariant to complicated non-linear transformations of the inputs, e.g. **lighting changes** and **geometric distortions**.
- The learned function can be used to map new samples not seen during training, with no prior knowledge.
- The mapping generated by the function is in some sense smooth and coherent in the output space.

## Notations

Given a set of input vectors $\mathcal{I} = \left\{\vec{X}_1, ..., \vec{X}_P\right\}$, where $\vec{X}_i \in \mathbb{R}^D$, $\forall i = 1, \ldots, n$.
Find a parametric function $G_W : \mathbb{R}^D \to \mathbb{R}^d$ with $d \ll D$.
Learn the parameters $W$ of the parameterized function $G_W$.
$G_W$ can **put neighbors together** and **push non-neighbors apart**.

The semantic similarity: $D_W(\vec{X}_1, \vec{X}_2) = \|G_W(\vec{X}_1), G_W(\vec{X}_2)\|_2$.

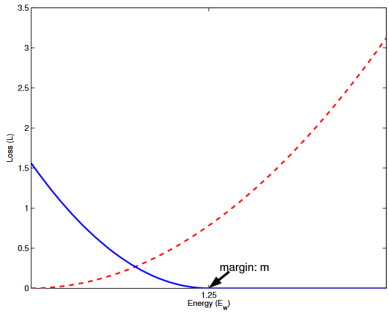
# The Contrastive Loss

The most general form of Contrastive Loss is:

$$\mathcal{L}(W) = \sum_{i=1}^{P} L(W, (Y, \vec{X}_1, \vec{X}_2)^i).$$

$$\begin{aligned} & L(W, (Y, X_1, X_2)^i) \\ &= (1-Y) L_S\left(D_W^i\right) + Y L_D\left(D_W^i\right) \\ &= (1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2, \end{aligned}$$

where $Y = 0$ if $\vec{X}_1$ and $\vec{X}_2$ are deemed similar, and $Y = 1$ if they are deemed dissimilar.



## Spring Model Analogy

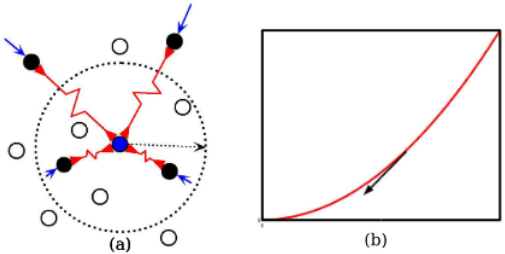
Consider the equation of a spring

$$F = -KX,$$

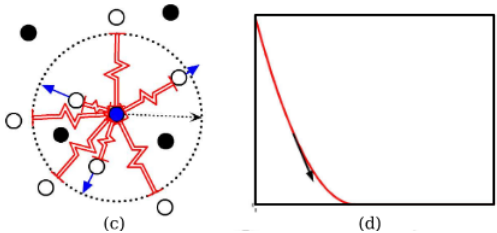
where $F$ is the force, $K$ is the spring constant and $X$ is the displacement of the spring from its rest length.

# Two Special Springs

**Attract-only**:
The rest length is equal to zero.




**m-Repulse-only**:
The rest length is equal to $m$ and there is no attractive force.



## Attract-only Spring

Each point is connected by **Attract-only** springs to **similar** points.

Consider the loss function

$$L_S(W, \vec{X_1}, \vec{X_2}) = \frac{1}{2}(D_W)^2.$$

The gradient of $L_S$ is

$$\frac{\partial L_S}{\partial W} = D_W \frac{\partial D_W}{\partial W}.$$

The gradient $\frac{\partial L_S}{\partial W}$ of $L_S$ gives the attractive force between the two points. $\frac{\partial D_W}{\partial W}$ defines the spring constant $K$.

## m-Repulse-only Spring

Each point is connected by **m-Repulse-only** springs to **dissimilar** points.

Consider the loss function

$$L_D(W, \vec{X_1}, \vec{X_2}) = \frac{1}{2}\{max(0, m - D_W)\}^2.$$

When $D_W > m$, $\frac{\partial L_D}{\partial W} = 0$.
When $D_W < m$, then

$$\frac{\partial L_D}{\partial W} = -(m - D_W)\frac{\partial D_W}{\partial W}.$$

# The Algorithm

**Step 1**: For each input sample $\vec{X}_i$, do the following:

(a) Using prior knowledge find the set of samples $\mathcal{S}_{\vec{X}_i} = \{\vec{X}_j\}_{j=1}^{p}$, such that $\vec{X}_j$ is deemed similar to $\vec{X}_i$.

(b) Pair the sample $\vec{X}_i$ with all the other training samples and label the pairs so that: $Y_{ij} = 0$ if $\vec{X}_j \in \mathcal{S}_{\vec{X}_i}$, and $Y_{ij} = 1$ otherwise.

Combine all the pairs to form the labeled training set.

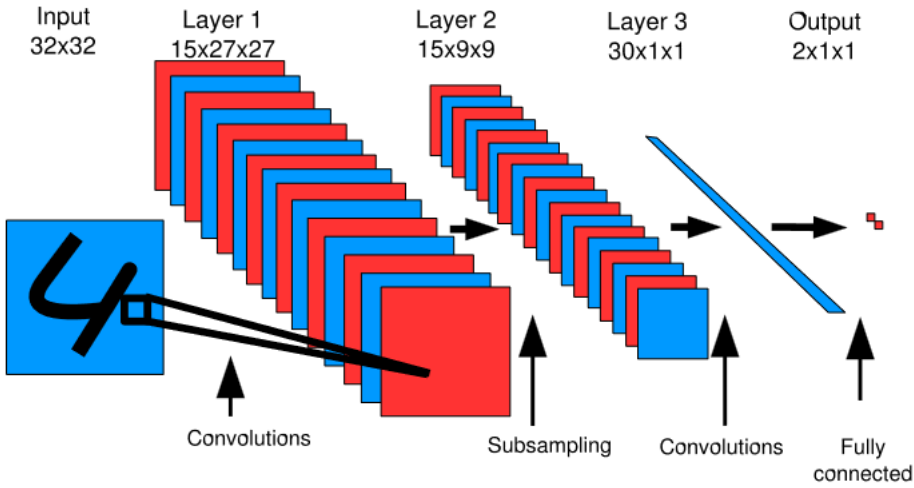
**Step 2**: Repeat until convergence:

(a) For each pair $(\vec{X}_i, \vec{X}_j)$ in the training set, do

i. If $Y_{ij} = 0$, then update $W$ to decrease $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

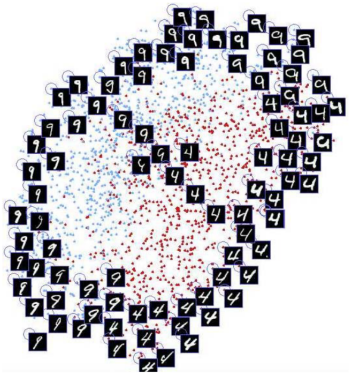
ii. If $Y_{ij} = 1$, then update $W$ to increase $D_W = \|G_W(\vec{X}_i) - G_W(\vec{X}_j)\|_2$

# Training Architecture

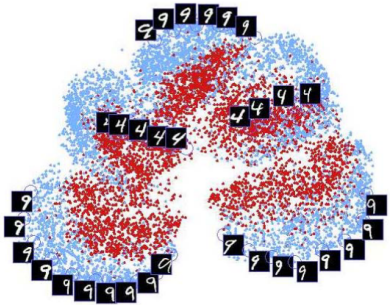


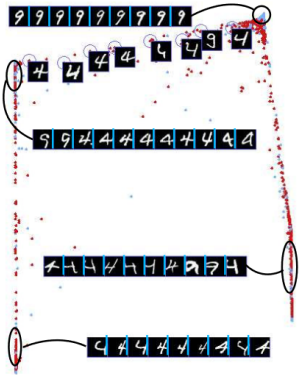
## Experiments on MNIST

Produce the set $S_{X_i}$ by choosing the 5 nearest neighbors.
All other possible pairs are labeled dissimilar.

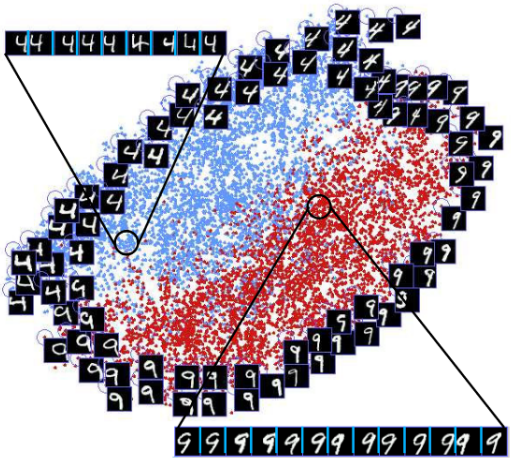


# Comparison with LLE under Transformation





# Results of Test Samples



# FaceNet

## Main References

F Schroff et al, FaceNet: A Unified Embedding for Face Recognition and Clustering.
A Hermans et al, In Defense of the Triplet Loss for Person Re-Identification.

## Cores

- Directly learning a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity.
- **Minimizing** the distance between an anchor and a **positive** and **maximizing** the distance between the anchor and a **negative**.

# Triplet Loss


**Anchor**: $x_i^a$.
**Positive**: $x_i^p$.
**Negative**: $x_i^n$.

## Triplet Loss

We want

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2,$$
$$\forall \left(f(x_i^a), f(x_i^p), f(x_i^n)\right) \in \mathcal{T},$$

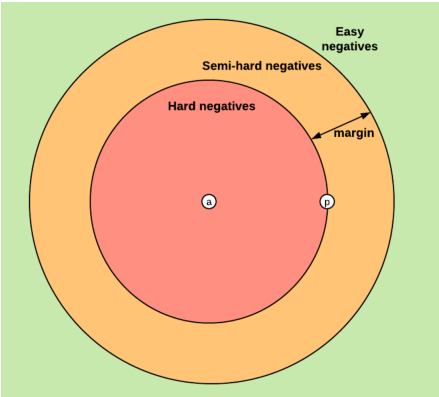
where $\alpha$ is a margin that is enforced between positive and negative pairs, $\mathcal{T}$ is the set of all possible triplets in the training set with cardinality $N$.

The Triplet loss is defined as:

$$\mathcal{L} = \sum_i^N \left[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha\right]_+ .$$

# Triplet Selection

- **easy triplets**: $d(a,p) + \alpha < d(a,n)$;
- **hard triplets**: $d(a,n) < d(a,p)$;
- **semi-hard triplets**: $d(a,p) < d(a,n) < d(a,p) + \alpha$.



# Triplet Selection


## FaceNet

Pick a random semi-hard negative for every pair of anchor and positive, and train on these triplets.

## Batch Hard

For each anchor, select the hardest positive (**biggest** $d(a, p)$) and the hardest negative among the batch.

# Deep Metric Learning versus Contrastive SSL

## Paradigm Shift Between Deep Metric Learning and Contrastive SSL

| **Deep Metric Learning** | | **Contrastive SSL** |
|---|---|---|
| positive/negative pairs come from labels or fixed transforms e.g. two halves of an image | $\Longrightarrow$ | positive pairs come from designed DAs that are continuously sampled, negative pairs are all non-positive pairs regardless of class membership |
| Hard-Negative Sampling for each mini-batch | $\Longrightarrow$ | random sampling |
| encoder DN | $\Longrightarrow$ | encoder DN + projector MLP |
| small dataset (N<200k) | $\Longrightarrow$ | large dataset |
| zero-shot k-NN validation | $\Longrightarrow$ | -zero-shot k-NN validation<br>-zero/few-shot/fine-tuning linear probing |