UNIVERSITY OF WARSAW

Distance Metric Learning

Foundation, Methods and Applications

Hung Son Nguyen

MML'2023

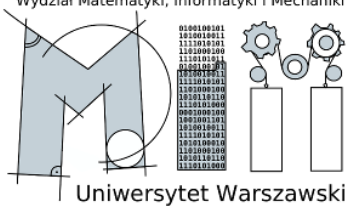


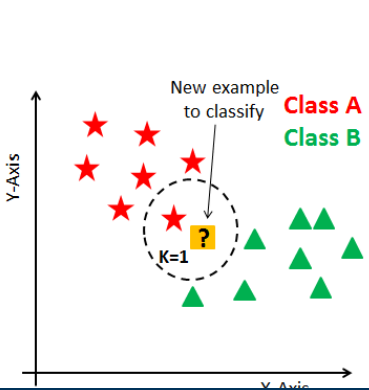
Table of contents

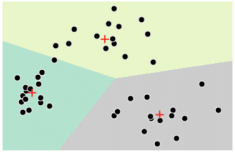
Introduction

Motivation

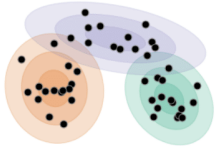
Motivation

- Similarity / distance judgments are essential components of many human cognitive processes (see e.g., [Tversky, 1977])
  - Compare perceptual or conceptual representations
  - Perform recognition, categorization...
- Underlie most ML and data mining techniques: k-means, clustering ...

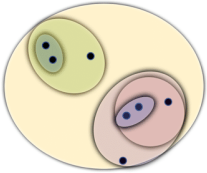


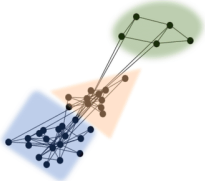


K-means clustering



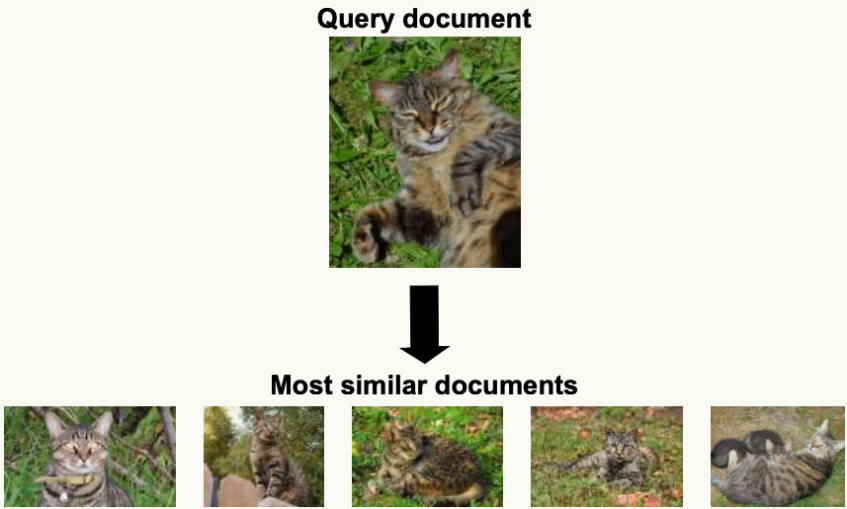
Mixture model (Gaussian)





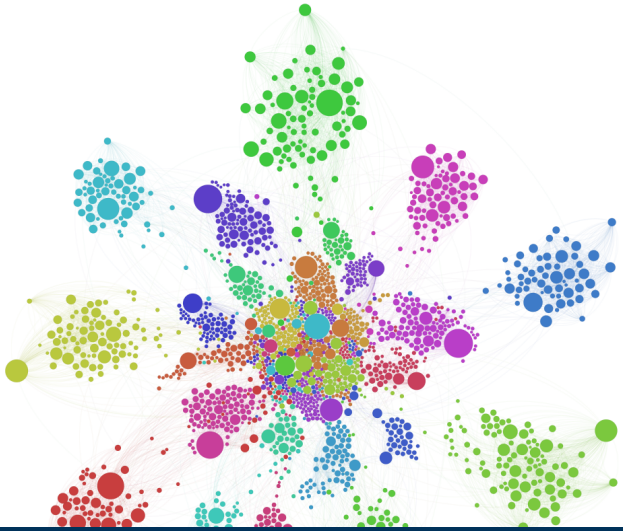
Motivation

Information Retrieval



Motivation

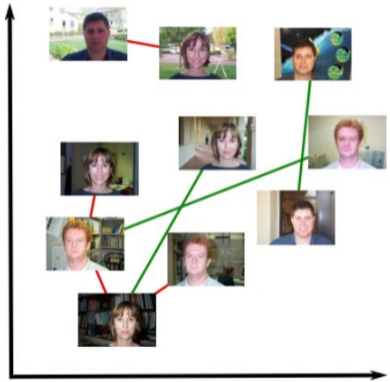
Data Visualization



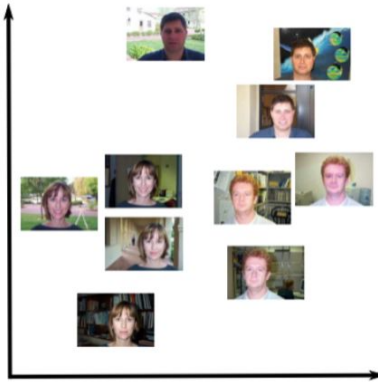
Motivation

- Choice of similarity is crucial to the performance
- Humans weight features differently depending on context
  - Facial recognition vs. determining facial expression
- Fundamental question: how to appropriately measure similarity or distance for a given task?
- Metric learning = infer the similarity or distance automatically from data
- Note: we will refer to **distance or similarity** indistinctly as **metric**

Basic idea of metric learning



Metric Learning



Basic recipe for metric learning

1. Choose a parametric distance or similarity function
   - Say, a distance $D_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ function parameterized by $\mathbf{M}$
2. Collect similarity constrains on data pairs/triplets of the following forms:
   - Must-link/cannot-link constraints (sometimes called positive/negative pairs):

$$\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \text{ and } \mathbf{x}_j \text{should be similar}\},$$
$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j) : \mathbf{x}_i \text{ and } \mathbf{x}_j \text{should be dissimilar}\},$$

   - Relative constraints (sometimes called training triplets):

$$\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) : \mathbf{x}_i \text{ should be more similar to } \mathbf{x}_j \text{ than to } \mathbf{x}_k\}.$$

3. Design a loss function: $\ell(\mathbf{M}, \mathcal{S}, \mathcal{D}, \mathcal{R})$ (a penalty when training constraints are violated), a regularizer: $R(\mathbf{M})$ and choose a regularization parameter $\lambda > 0$.
4. Learning: (estimate parameters s.t. metric best agrees with constraints)

$$\mathbf{M}_{best} = \min_{\mathbf{M}} [\ell(\mathbf{M}, \mathcal{S}, \mathcal{D}, \mathcal{R}) + \lambda R(\mathbf{M})]$$



'similar'



'dissimilar'

(b) pairs

D( , ) < D( , )

D( , ) < D( , )

(c) triplets

D( , ) < D( , )

D( , ) < D( , )

(d) quadruplets

Some examples of DML algorithms

Some examples of DML algorithms

Linear matrix learning

Mahalanobis distance

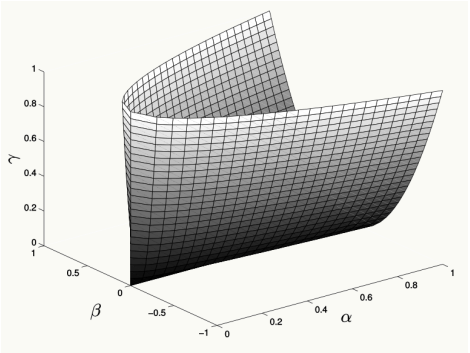
- Mahalanobis distance to refer to generalized quadratic distances, defined as

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')}$$

  and parameterized by $\mathbf{M} \in \mathbb{S}_+^d$, where $\mathbb{S}_+^d$ is the cone of symmetric positive semi-definite (PSD) $d \times d$ real-valued matrices.
- If $\mathbf{M} \in \mathbb{S}_+^d$, then $d_{\mathbf{M}}$ satisfies the properties of a pseudo-distance.

$\mathbf{M} \in \mathbb{S}_+^d$ (also denote $\mathbf{M} \succeq 0$) iff

- Its eigenvalues are all nonnegative
- $\mathbf{x}^T \mathbf{M}^T \mathbf{x} \geq 0$, for all $\mathbf{x} \in \mathbf{R}^d$
- $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L} \in \mathbb{R}^{k \times d}$ and k is the rank of $\mathbf{M}$



- Note that when $\mathbf{M}$ is the identity matrix, we recover the Euclidean distance.
- Otherwise, $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L} \in \mathbb{R}^{k \times d}$ and k is the rank of $\mathbf{M}$. We can then rewrite $d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}')$ as follows:

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') &= \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')} = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{L}^T \mathbf{L} (\mathbf{x} - \mathbf{x}')} \\ &= \sqrt{(\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}')^T (\mathbf{L}\mathbf{x} - \mathbf{L}\mathbf{x}')} \end{aligned}$$

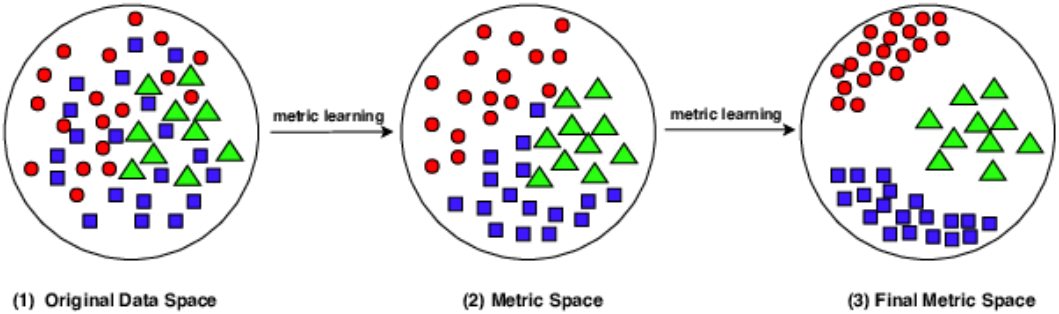
Thus, a Mahalanobis distance implicitly corresponds to computing the Euclidean distance after the linear projection of the data defined by the transformation matrix $\mathbf{L}$

Learning Mahalanobis distance

The goal is to adapt some pairwise real-valued metric function:

$$d_{\mathbf{M}}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')}$$

to the problem of interest using the information brought by training examples



The existing methods learn the positive semi-definite matrix $\mathbf{M}$ in $d_{\mathbf{M}}$.

Learning Mahalanobis distance

Optimisation model:

$$\min_{\mathbf{M}} f(\mathbf{M}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \quad (1)$$

$$s.t.\ g(\mathbf{M}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} d_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq 1 \quad (2)$$

$$\mathbf{M} \succeq 0 \quad (3)$$

If $\mathbf{M} = diag(m_1, \cdots, m_d)$, the problem is equivalent to:

$$\min_{\mathbf{M}} F(\mathbf{M}) = f(\mathbf{M}) - \ln(g(\mathbf{M}))$$
$$s.t.\ \mathbf{M} \succeq 0$$

(The Newton-Raphson method)

Alternative model:

$$\max_{\mathbf{M}} g(\mathbf{M}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}} d_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j)$$
$$s.t.\ \mathbf{M} \in \mathcal{P} \text{ and } \mathbf{M} \in \mathbb{S}^d_+$$

where $\mathcal{P} = \{\mathbf{M} : f(\mathbf{M}) \leq 1\}$.

Gradient ascent iterative method:

Projection steps:

$$\mathbf{M}_1 = \underset{\mathbf{M}'}{\operatorname{argmin}}\{\|\mathbf{M}' - \mathbf{M}\|_F : \mathbf{M}' \in \mathcal{P}\}$$

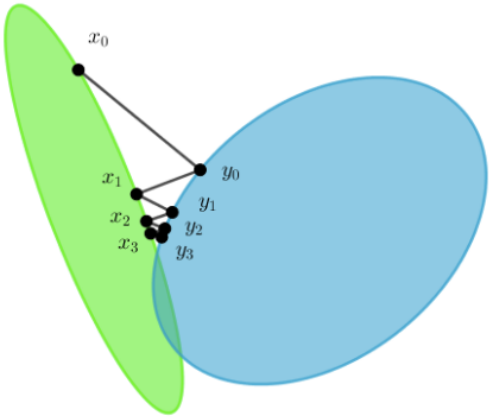
$$\mathbf{M}_2 = \underset{\mathbf{M}'}{\operatorname{argmin}}\{\|\mathbf{M}' - \mathbf{M}_1\|_F : \mathbf{M}' \in \mathbb{S}^d_+\}$$

Gradient ascent step:

$$\mathbf{M} = \mathbf{M}_2 + \alpha(\nabla_{\mathbf{M}} g(\mathbf{M}_2))_{\perp \nabla_{\mathbf{M}} f}$$

projected gradient descent

- Project onto distance constraint: $O(d^2)$ steps
- Project onto $\mathbb{S}^d_+$: $O(d^3)$ steps



Learning with triplet constraints

Target: information retrieval [Schultz and Joachims, 2003]

Model

$$\min_{\mathbf{M} \in \mathbb{S}_d^+, \geq 0} \sum_{i,j,k} \xi_{i,j,k} + \lambda \|\mathbf{M}\|_F^2$$

$$s.t. \quad d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq \xi_{i,j,k} \quad \text{for all } (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in \mathcal{R}$$

- Regularization by Frobenius norm $\|\mathbf{M}\|_F^2 = \sum m_{ij}^2$
- Formulation is convex
- One large margin soft constraint per triplet
- Can be solved with similar techniques as SVM

Some examples of DML algorithms

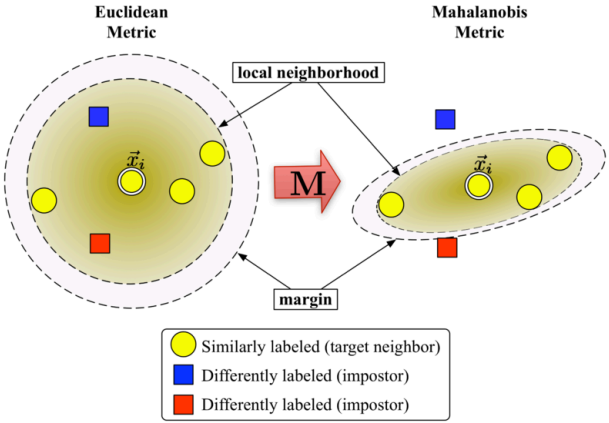
Large Margin Nearest Neighbor

Large Margin Nearest Neighbor

- Constraints: derived from labeled data

$$\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) : y_i = y_j, \mathbf{x}_j \text{ belongs to } k\text{-nearest neightborhood of } \mathbf{x}_i\}$$

$$\mathcal{R} = \{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) : (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{S}, y_k \neq y_i\}$$



Large Margin Nearest Neighbor

Target task: k-NN classification [Weinberger et al., 2005]

Model

$$\min_{\mathbf{M}\in\mathbb{S}_d^+,\,\geq 0} (1-\mu) \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}} d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) + \mu \sum_{i,j,k} \xi_{i,j,k}$$

$$s.t. \quad d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i,\mathbf{x}_j) \geq \xi_{i,j,k} \quad \text{for all } (\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R}$$

$\mu \in [0,1]$ is the trade-off parameter

Mahalanobis distance learning in other tasks

- Learning to rank [McFee and Lanckriet, 2010]
- Multi-task learning [Parameswaran and Weinberger, 2010]
- Transfer learning [Zhang and Yeung, 2010], [J Zhen, 2017]
- Semi-supervised learning [Hoi et al., 2008]

Some examples of DML algorithms

Learning feature selection

Feature AwaRe Metric learning (Farm)

- Decouple the feature aware metric $\hat{\mathbf{M}}$ into a full metric $\mathbf{M}$ and a diagonal weight separately as: $\hat{M} = diag(\mathbf{w})\mathbf{M}\,diag(\mathbf{w})$
- Consequently, distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ using new metric $\hat{\mathbf{M}}$:

$$dist_{\hat{\mathbf{M}}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T\, diag(\mathbf{w})\mathbf{M}\, diag(\mathbf{w})(\mathbf{x}_i - \mathbf{x}_j)$$

- Therefore, objective function of Farm method is:

$$\min_{\mathbf{M},\mathbf{w}} \sum_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in\mathcal{R}} \ell(dist^2_{\hat{\mathbf{M}}}(\mathbf{x}_i, \mathbf{x}_k) - dist^2_{\hat{\mathbf{M}}}(\mathbf{x}_i, \mathbf{x}_j)) + \lambda_1\Omega(\mathbf{M}) + \lambda_2\|\mathbf{w}\|_1$$

where Ω is a regularizer and lost function $\ell()$ is a decreasing convex function.

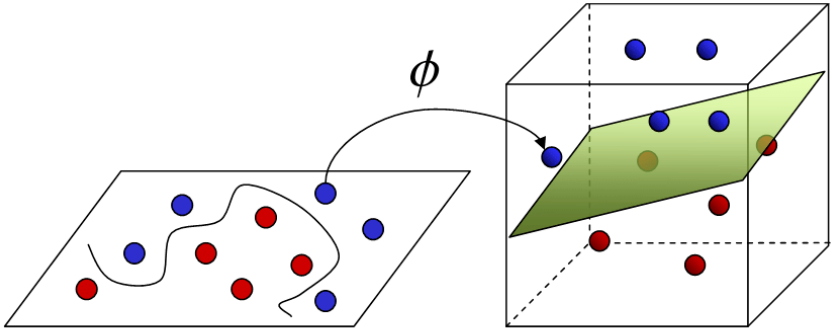
Implementation detail

- The regularizer Ω on full metric component M is also flexible:
  - when $\Omega(\mathbf{M}) = \|\mathbf{M}\|_F^2$, it can be used to prevent overfitting;
  - when $\Omega(\mathbf{M}) = \|\mathbf{M}\|_1$, elements in $\mathbf{M}$ are also sparse.
  - when $\Omega(\mathbf{M}) = \|\mathbf{M}\|_* = Tr(\sqrt{\mathbf{M}^* \cdot \mathbf{M}})$ (where $\mathbf{M}^*$ is the Hermitian conjugate of $\mathbf{M}$), $\mathbf{M}$ should be low rank and sparse on principal components is forced, so the combined metric $\hat{\mathbf{M}}$ is a sparse and low rank one.
- Lost function: $\ell(x) = \begin{cases} 0 & \text{if } x \geq 1 \\ 1/2 - x & \text{if } x \leq 0 \\ 1/2(1-x)^2 & \text{otherwise.} \end{cases}$
- Optimization strategy: alternative style, i.e., optimize on $\mathbf{w}$ with $\mathbf{M}$ fixed and vice versa
  1. Fix $\mathbf{w}$ and solve $\mathbf{M}$:
  2. Fix $\mathbf{M}$ and solve $\mathbf{w}$:

Some examples of DML algorithms

Nonlinear extension

Kernelization



Kernelization

Definition (Kernel function)

A symmetric function K is a kernel if there exists a mapping function $\phi : \mathcal{X} \to \mathbb{H}$ from the instance space $\mathcal{X}$ to a Hilbert space $\mathbb{H}$ such that K can be written as an inner product in $\mathbb{H}$:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

Equivalently, K is a kernel if it is positive semi-definite (PSD), i.e.,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$$

for all finite sequences of $x_1, \cdots, x_n \in \mathcal{X}$ and $c_1, \cdots, c_n \in \mathbb{R}$.

Kernelization

- Notations
  - Kernel $K(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$, training data $\{\mathbf{x}_i\}_{i=1}^n$
  - Let $\phi_i = \phi(\mathbf{x}_i) \in \mathbb{R}^D$, $\mathbf{\Phi} = [\phi_1, \cdots, \phi_n] \mathbb{R}^{n \times D}$
- Mahalanobis distance in kernel space:

$$d_{\mathbf{M}}(\phi_i, \phi_j) = (\phi_i - \phi_j)^T \mathbf{M} (\phi_i - \phi_j) = (\phi_i - \phi_j)^T \mathbf{L}^T \mathbf{L} (\phi_i - \phi_j)$$

- Setting $\mathbf{L}^T = \mathbf{\Phi} \mathbf{U}^T$, where $\mathbf{U} \in \mathbb{R}^{D \times n}$

$$d^2_{\mathbf{M}}(\phi(x), \phi(x')) = (\mathbf{k} - \mathbf{k}')^T \mathbf{M} (\mathbf{k} - \mathbf{k}')$$

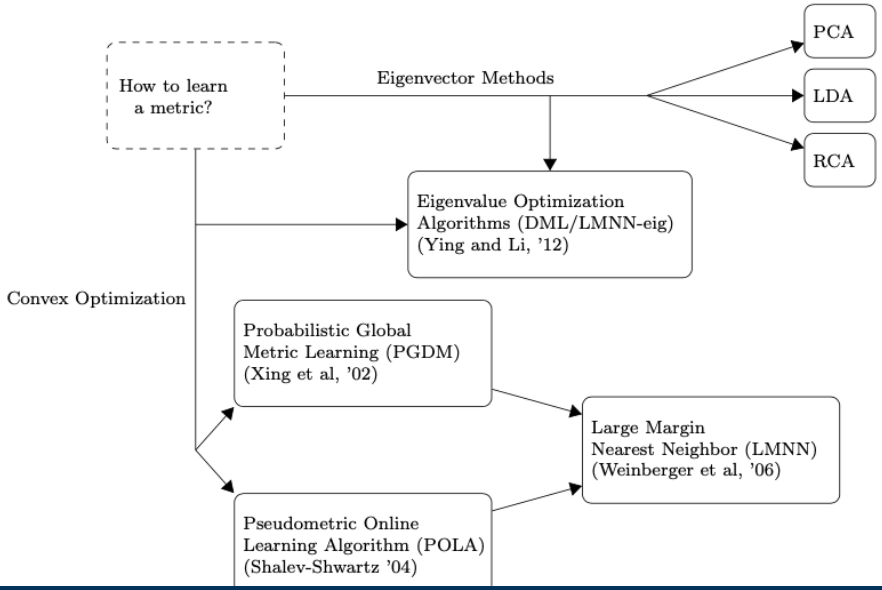
  where $\mathbf{M} \in \mathbf{U}^T \mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{k} = \mathbf{\Phi}^T \phi(\mathbf{x}) = [K(\mathbf{x}_1, \mathbf{x}), \cdots, K(\mathbf{x}_n, \mathbf{x})]^T$
- Justified by a representer theorem [Chatpatanasiri et al., 2010]

Kernelization

- Similar trick as kernel SVM
  - Use a nonlinear kernel (e.g., Gaussian RBF)
  - Inexpensive computations through the kernel
  - Nonlinear metric learning while retaining convexity
- Need to learn $O(n^2)$ parameters
- Linear metric learning algorithm must be kernelized
  - Interface to data limited to inner products only
  - Several algorithms shown to be kernelizable
- General trick [Chatpatanasiri et al., 2010]:
  1. Kernel PCA: nonlinear mapping to low-dimensional space
  2. Apply linear metric learning algorithm to transformed data

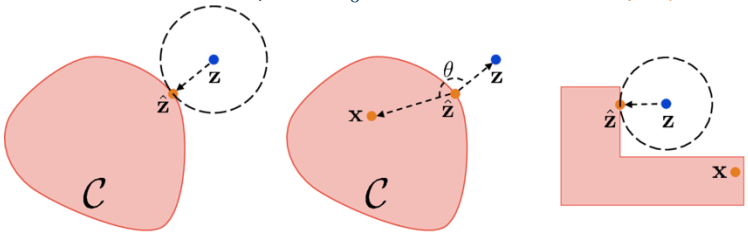
Mathematical foundations of DML

General overview of the approaches to DML



Convex Analysis

- **Convex projection theorem:** the distance from x to a convex close set K is materialized in the point x_0, which is called the projection of x to K



- Convex optimization mechanisms:
  - Gradient Descent method
  - Project Gradient method
  - Iterated Projections method
  - Sub-gradient Descent method

Matrix Analysis

- **Matrix decomposition theorem** for any matrix $\mathbf{M} \in \mathbb{S}_+^d$, there exists $L \in \mathbb{R}^{d \times d}$ s.t. $\mathbf{M} = L^T L$, and L is unique except for an isometry
- **Frobenius inner product:** With the Frobenius product we convert the matrices set in a Hilbert space, and therefore can apply the convex analysis theory studied in the previous section.
- **semidefinite programming:** semidefinite projection theorem states that we can compute the projection of a matrix onto the positive semidefinite cone by performing an eigenvalue decomposition, nullifying the negative eigenvalues and recomposing the matrix with the new eigenvalues
- **PCA:** principal components analysis

$$\max_{L \in \mathbb{R}^{d' \times d}} tr(LAL^T) \qquad s.t.: \qquad LL^T = I$$

where A is a symmetric matrix of dimension d.

Information Theory

- **Mahalanobis distance:** the generalization of Euclidean distance
- Other distances: Matusita, Bhattacharyya ...
- **Kullback-Leibler divergence:** The relative entropy or the Kullback-Leibler divergence, defined for probability distributions p and q, and X the random variable corresponding to p as

$$KL(p\|q) = \mathbf{E}_p \left[\frac{\log(p)}{\log(q)} \right]$$

- **The Jeffrey divergence** or the symmetric relative entropy, defined for p, q and X in the same conditions as above, as

$$KL(p\|q) = KL(p\|q) + KL(q\|p)$$

Information Theoretic Metric Learning (ITML)

- Let $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$, assume

$$\mathcal{S} = \{(i,j) : d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) < l\} \qquad \mathcal{D} = \{(i,j) : d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) > u\}$$

- Given $\mathbf{M} \in \mathbb{S}_+^d$ we can construct a normal distribution:

$$p(x|\mathbf{M}) = \frac{1}{(2\pi)^{n/2}(\det(\mathbf{M})^{1/2})} \exp((x-\mu)^T \mathbf{M}^{-1}(x-\mu))$$

- We measure the closeness between $\mathbf{M}_0$ and $\mathbf{M}$ through the Kullback-Leibler divergence between their corresponding gaussian distributions

$$KL(p(x|\mathbf{M}_0), p(x|\mathbf{M})) = \int p(x|\mathbf{M}_0) \log \frac{p(x|\mathbf{M}_0)}{p(x|\mathbf{M})} dx$$

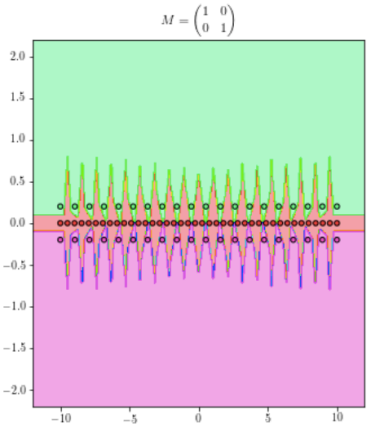
- DML Problem:

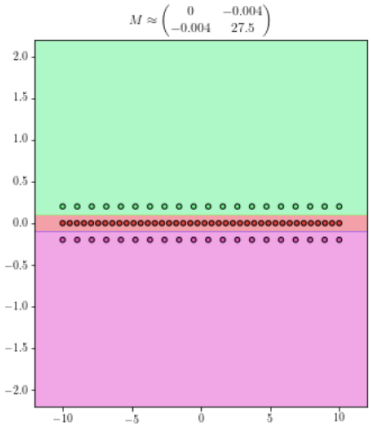
$$\min_{\mathbf{M} \in \mathbb{S}_+^d} KL(p(x|\mathbf{M}_0), p(x|\mathbf{M})) \qquad s.t. \qquad d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) < l, \quad (i,j) \in \mathcal{S}$$

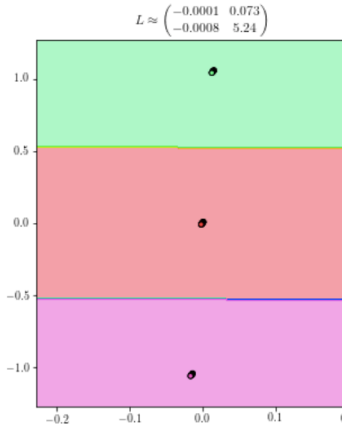
$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) > u, \quad (i,j) \in \mathcal{D}$$

Application of DML in Machine Learning

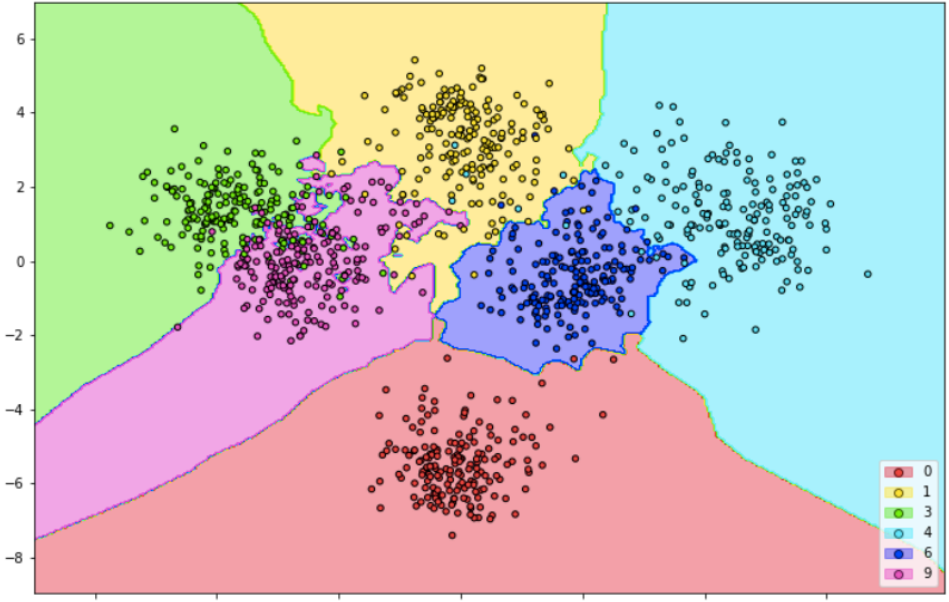
Improve the performance of distance-based classifiers







Dimensionality reduction



Other

- Axes selection and data rearrangement.
- Improving the performance of clustering algorithms
- Semi-supervised learning.

Prospects and Challenges in Distance Metric Learning

Prospects of DML

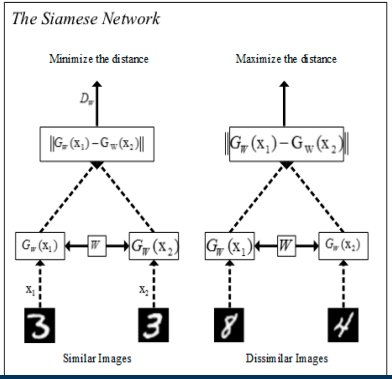
- **Big data:**
  - although many of DML algorithms, especially those based on gradient descent, are quite slow and do not scale well with the number of samples, they can be largely parallelized in both matrix computations and gradient descent batches.
  - DML can be extended to handle Big Data by developing specialized algorithms and integrating them with frameworks such as Spark and Cloud Computing architectures
- **Application of DML to singular supervised learning problems:**
  - regression,
  - multi-dimensional classification,
  - ordinal classification,
  - multi-output learning
  - and even transfer learning.

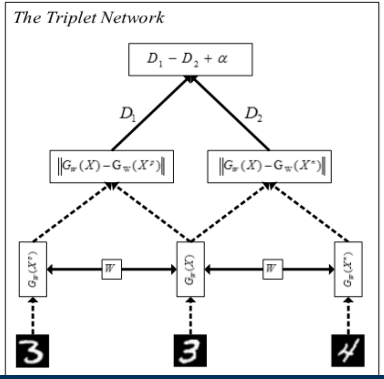
Prospects of DML: Hybridization

- **Hybridization with feature selection techniques:**
  - to solve high dimensional data problems.
  - to combine DML with feature selection techniques prepared for very high dimensional data
- **Hybridization with shallow learning techniques:**
  - DML with Naive-Bayes: obtaining a Naive-Bayes classifier whose feature distributions are determined by the nearest neighbors of each class;
  - DML with neural networks, to find the best neural network architecture;
  - DML with random forests, by exploiting the relationship between voting points and potential nearest neighbors;
  - DML with ensemble methods, like bootstrap;
  - DML with SVM, training them locally in neighborhoods
  - DML with rule-learning algorithms, obtaining the so-called nested generalized exemplar algorithms

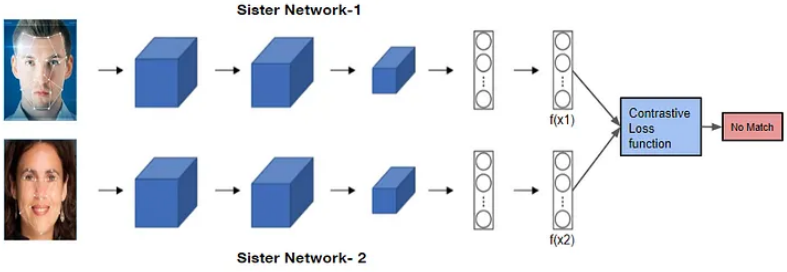
Prospects of DML: Hybridization

- **Hybridization with deep learning techniques:**
  - use the k-nearest neighbors classifier to provide interpretability and robustness to deep neural networks.
  - deep metric learning: use of neural networks to learn distances
    - Deep Metric Learning by Siamese Convolutional Neural Networks
    - Triplet Convolutional Neural Networks





Deep metric learning: siamese CNN



- Prepare the dataset in a pair-based format. Label these pair as similar or dissimilar based on their ground truth.
- Choose a suitable loss function according to your dataset.
- Extract the features from the input pairs, calculate the similarity index, and compute the gradients.
- While training, update the shared parameters using the computed gradients.
- Sister NN share the same weights and parameters.

Siamese CNN: Loss function

- **Contrastive Loss Function:** More formally, we suppose that we have a pair (I_i, I_j) and a label Y that is equal to 0 if the samples are similar and 1 otherwise. To extract a low-dimensional representation of each sample, we use a CNN f that encodes the input images I_i and I_j into an embedding space where $x_i = f(I_i)$ and $x_j = f(I_j)$. The contrastive loss is defined as:

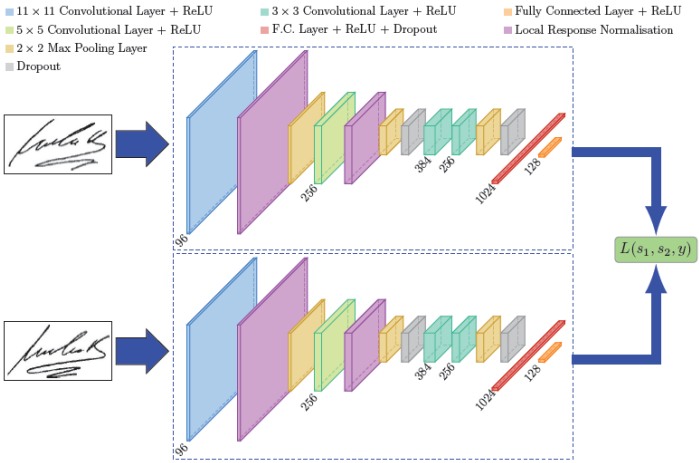
$$\mathbf{L} = (1 - \mathbf{Y}) * ||x_i - x_j||^2 + Y * max(0, m - ||x_i - x_j||^2)$$

  where m is a hyperparameter, defining the lower bound distance between dissimilar samples.

- **Triplet Loss Function**

Siamese CNN example: SigNet

While training, the feature vectors need to have two properties to make the few-shot learning strategy work:

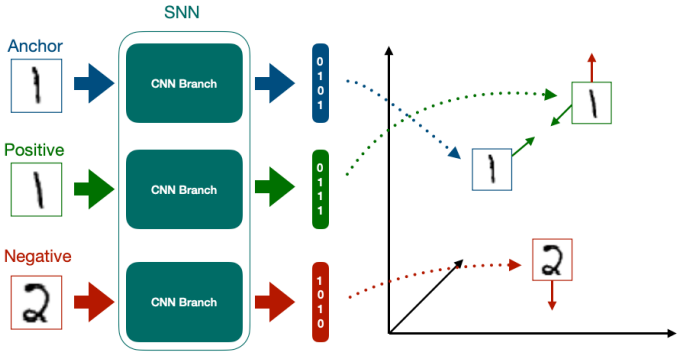


1. the feature vectors of similar and dissimilar pairs should be descriptive, informative, and distinct enough from each other so that segregation can be learned effectively.
2. the feature vectors of similar image pairs should be similar enough, and those for dissimilar pairs should be dissimilar enough so that the model can quickly learn semantic similarity.

Why use Siamese CNN

- Unlike traditional NN in deep learning, a siamese network does not require too many instances of a class and few are enough to build a good model.
- Moreover, a new class can be added without training the entire network from scratch after the siamese neural network has been trained and deployed. The model trains by learning how similar or dissimilar image pairs are, samples from a new class can be added to the trained siamese network, and training can be resumed since the network architecture will compare the new images with the rest of the classes and update the weights and the fully connected layer.
- This behavior is unique to a network architecture that uses one-shot learning since other categories of neural networks would have to be trained from scratch on a large, class-balanced dataset for significant performance.

Deep metric learning: triple CNN and triplet loss

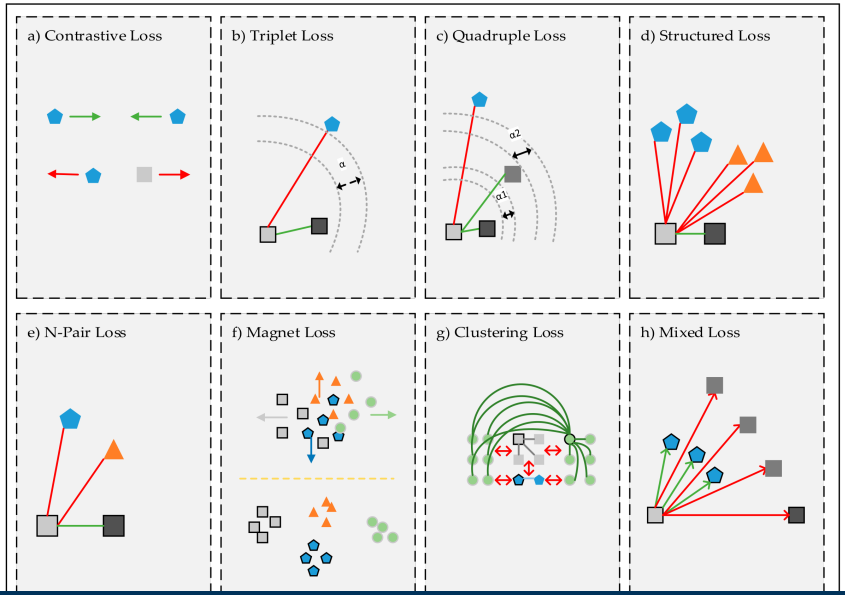


Triplet loss outperforms the contrastive loss by using triplets of samples instead of pairs. Specifically, it takes as input an anchor sample I, a positive sample I^+ and a negative sample I^-. During training, the loss enforces the distance between the anchor sample and the positive sample to be less than the distance between the anchor sample and the negative sample:

$$\mathbf{L} = max(0, ||x - x^+||^2 - ||x - x^-||^2 + m)$$

When we train a model with the triplet loss, we require fewer samples for convergence since we simultaneously update the network using both similar and dissimilar samples.

Deep metric learning: more loss functions



Prospects:

- **Other approaches for the concept of distance:**
  - Most of the current DML theory focus on Mahalanobis distances
  - other possible distances, such as local Mahalanobis distances, that lead to a multi-metric learning
- Implementation: PyDML Library
- Demo:

Challenges in DML

- **Non-linear distance metric learning:**
  - learning a Mahalanobis distance is equivalent to learning a linear map!
  - the kernelization of DML algorithms can be applied to fit non-linear data
  - **Challenge:** Extend the kernel trick to other algorithms by searching for suitable parameterizations and representer theorems.
  - **Challenge:** Ądapt classical objective functions so that they can work with non-linear distances or with non-linear transformations of the data learned by another algorithm.
- **Multi-linear distance metric learning:**
  - In many cases traditional vector representation may not be the most appropriate to fit the data properly. – e.g. lack of the consideration of neighborhood relationships between pixels in an image
  - multi-linear DML should learn distances in, e.g. tensor spaces.
- **Other optimization mechanisms.**

  – Non-convex optimization? Evolutionary algorithms: simulated annealing, particle swarm optimization or response surface methods?

Summary and Conclusions

Summary: We have studied

- The concept of DML: (1) what its applications are, (2) how to design its algorithms, and the (3) theoretical foundations of this discipline.
- Some most popular DML algorithms and their theoretical foundations, and explained different resolution techniques.
- In order to understand the theoretical foundations of DML and its algorithms, it was necessary to delve into three different mathematical theories:
  - **Convex analysis:** many optimization problems + methods for solving them.
  - **Matrix analysis:** many useful tools to help understand DML, from how to parameterize Mahalanobis distances to the optimization with eigenvectors going through the most basic algorithm of semidefinite programming.
  - **Information theory** has motivated several interesting algorithms.
- Deep metric learning seems to be an important study field for researchers.
  - Most of the recent studies were inspired by Siamese and Triplet networks.
  - Many loss functions have been designed

Can you spot the owner

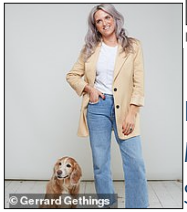
DailyMail Online

James (3) & Bingley (14), Rosalyn (10) & Bertie (6), Kim (13) & Coco (11)

James (3) & Bingley (14)
Rosalyn (10) & Bertie (6)
Kim (13) & Coco (11)

Helen (17) & Rupert (9), Desiree (1) & Stella (4), Hayley (8) & Foxy (18)

Helen (17) & Rupert (9)
Desiree (1) & Stella (4)
Hayley (8) & Foxy (18)



Barry (12) & Bubbles (2)
Maggie (5) & Jack Sparrow (16)
Skye (15) & Barley (7)